POLITECNICO DI TORINO

Master Thesis in Electrical Engineering

Control of Grid-Forming Power Converters

Author: Valentina Zito

Supervisor: Prof. Radu Bojoi

Advisor: Fabio Mandrile

July 2020

Contents

List of Tables

List of Figures

Chapter 1

Introduction

New Challenges and Scenarios for the Electrical Power System

In the last few decades, the continuous growth of electricity demand combined with the decarbonization requirements have lead to an increase in the amount of energy produced from renewable sources. The International Renewable Energy Agency (IRENA) has conducted a study proposed in [13]. From this analysis they predict that, by 2050, 86% of electricity generation will be renewable, and 60% will come from solar and wind. In Fig. 1.1, it is shown the expected energy generation and the power generation installed capacity by source. It is visible that wind and solar power technologies dominate growth in renewable-based generation in the next thirty years.

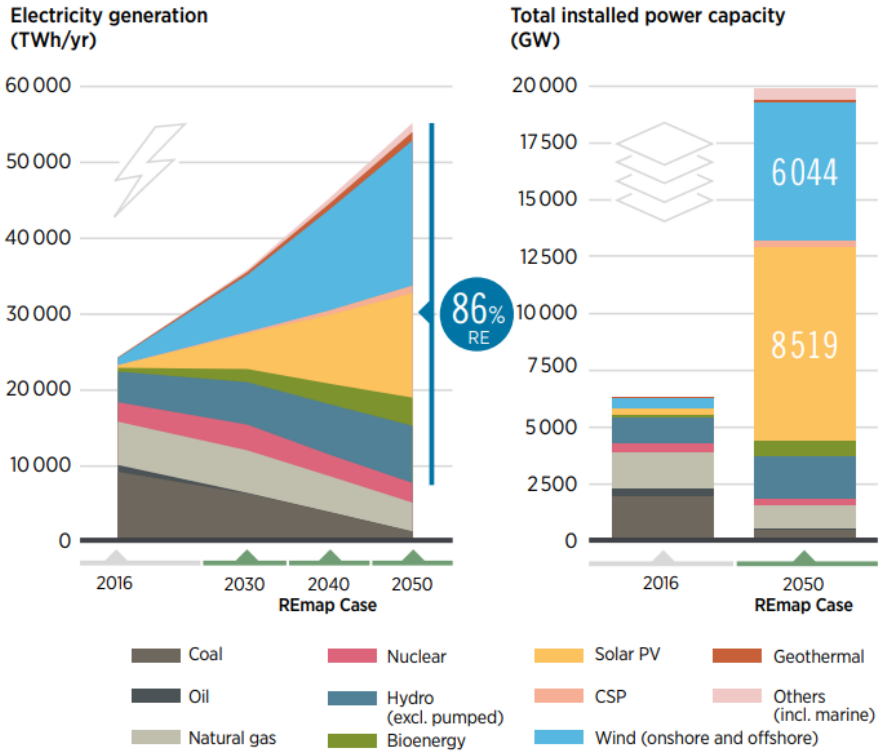


Figure 1.1: Expected energy generation and power generation installed capacity (GW) by fuel [13].

In this scenario, many new solutions will be introduced in the classical power system such as distributed generation systems (DG), which are based on renewable energies sources (RESs), electrical energy storage (EES), microgrids (MG), active demand management (ADM), smart control based on information and comunications thecnologies (ICTs).

Every RES, such as wind and solar photovoltaic (PV) source, and EES is typically connect to the grid through a power electronics interface called *inverter*.

This evolutionary transformation of the electrical system has created new challenges for research, related to the stability requirements, the reliability and the controllability of the system [16] and [14]. Fig. 1.2 shows the expected evolution of the power system structure.

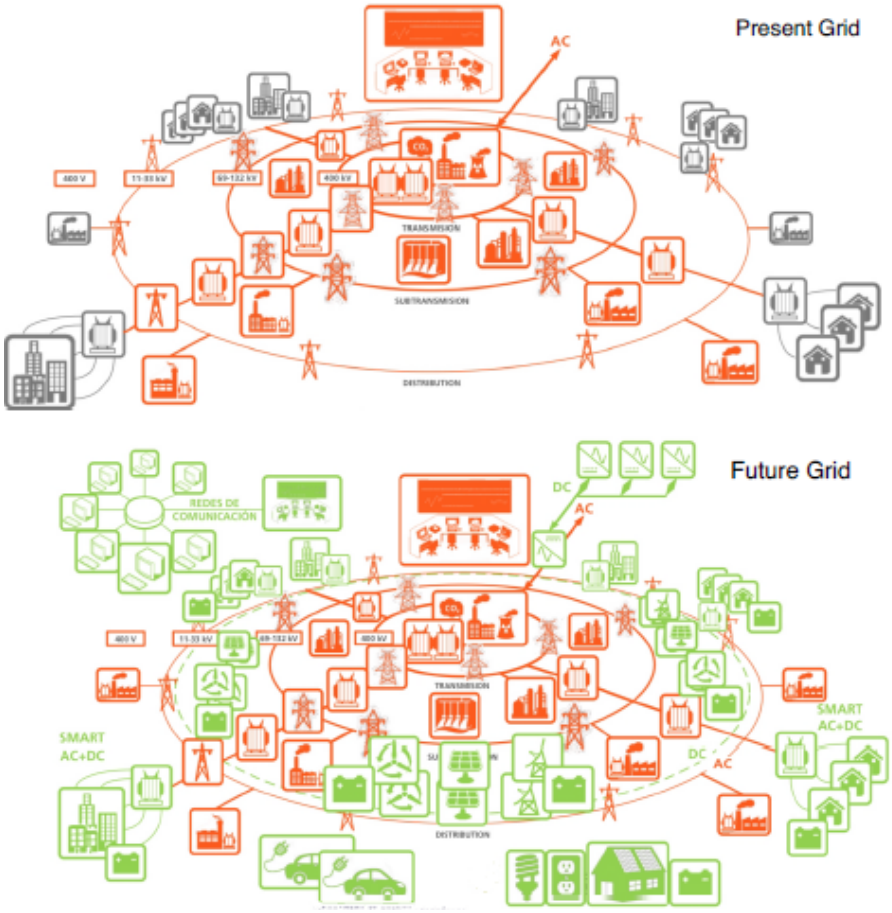


Figure 1.2: Present and Future Electric Grid [25].

As it can be seen, the future power system is expected to be fed by an huge amount of RESs, instead of the classical synchronous generators [16]. In [2] it is well explained that moving from a system completely driven by synchronous machines to a system with only power electronics introduces changes in the system's dynamic characteristics. In fact, in the first case the transient behaviour is imposed by the physical (electrical and mechanical) limitations of the machine, while in the second case the behaviour depends mainly on the characteristic of the adopted control structure.

From the literature it is known that the mechanical inertia, which is the net rotating mass across all interconnected machines, is strictly correlated to the ability of the system to counteract tackle frequency variations. In fact, if a system has

a large inertia, it can sustain better a frequency deviation thanks to the kinetic energy stored in its rotating parts. This make the system more stable.
In contrast to a synchronous generator, an inverter does not contain rotating masses, but only electronic components. For this reason they are considered to have zero inertia and therefore their transient behaviour depends only on the adopted control strategy. Moreover, the replacement of conventional synchronous generators with grid connected renewable-generation units leads to a reduction of the overall stability ([16], [2]). This environment creates less favourable grid conditions for the stable operation of the inverters used in wind e PV generation units. In fact, most of the grid connected inverter use a voltage source inverter (VSI) technology, which are controlled by a current vector control, aligned to the grid voltage vector. This operating mode is called Grid-following control and it is shown in Fig. 1.3.

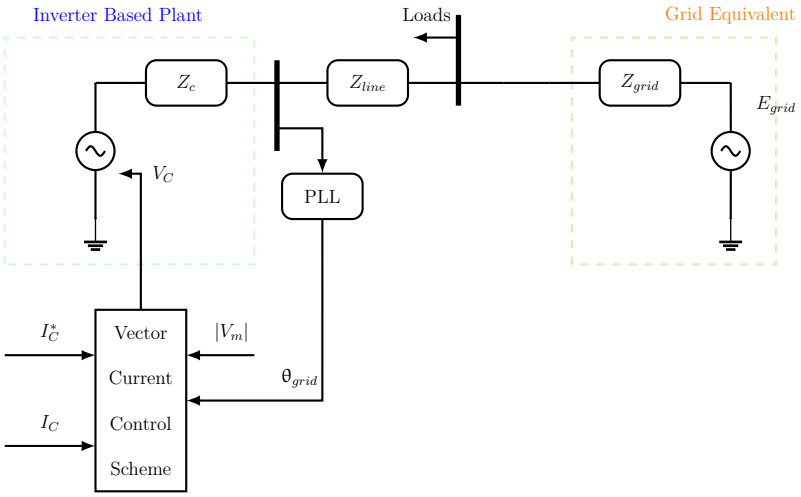


Figure 1.3: Grid Following control strategy scheme ([20]).

Basically, a Grid-Following controller [20] involves a phase-locked loop (PLL) in order to instantaneously estimate the sinusoidal voltage angle at the inverter terminals. So, to inject a controlled current into the grid, the controller "follows" the sinusoidal voltage terminals. The converter controls the current vector injected into the grid in order to achieve the desired active and reactive power flow. However, this solution works well in stiff AC grids with low frequency and voltage amplitude deviations, for example in networks with a large presence of synchronous machines that can provide a stiff voltage and frequency.

Due to the electrical evolution of the power system, another control scheme, called **Grid-Forming**, can be adopted. Extensive research is carried out on these topics. This is due to the fact that Grid-Forming power converters are able to support the operation of an AC power system under nominal, disturbed and unbalanced operations without being dependent from the synchronous generators or synchronous compensators [19]. In fact, it acts as an ideal voltage source which actively controls the magnitude of the voltage and the frequency at the point of common coupling (PCC) [21]. The Grid-Forming control strategy is widely used in MGs configuration, but it can play an important role in improving the frequency dynamics and stability of the inverter-dominated-power systems. In this case, the dependency of the frequency dynamics on the mechanical inertia can be reduced thanks to the active frequency control provided by Grid-Forming inverters.
Furthermore, another challenge to overcome is the reduction of black start capability, so the inverter-dominated systems will be required to provide sufficient starting current or the loads must be segregated in order to permit the re-powering of the grid. The black start is one of the ancillary services offered by the microgrid configuration, as it will be discuss later.

So, in order to overcome all these challenges and to exploit in a more efficient way the distributed RES, the *microgrid* configuration has a crucial role in the future power generations system. A MG is considered as a small-scale grid and it is formed by DG systems, EES devices and by loads hierarchically controlled. The RES most involved in a MG are the PV and wind power plants, which have an intermittent behaviour that depends on the environmental conditions, so the EESs assumed an important role in these small-scale grids.

The control system of the MG must be able to work in grid-connected and island operating mode, depending on the operating conditions. So, in normal operational conditions the MG helps the main grid to supply the loads and contributes to the stability. If any contingency occurs in the main grid, it is possible to maintain the continuity of supply of some portions of the entire power system by enabling the creation of active island. This is crucial for loads that need the continuity of supply as hospitals or large industries. Moreover, the possibility to create small-scale isolated networks, not connected to main power system, is useful to energize areas

that are remote and distant from any transmission and distribution infrastructure and, therefore, have no connection to the utility grid.
Typical MG elements are shown in Fig. 1.4.

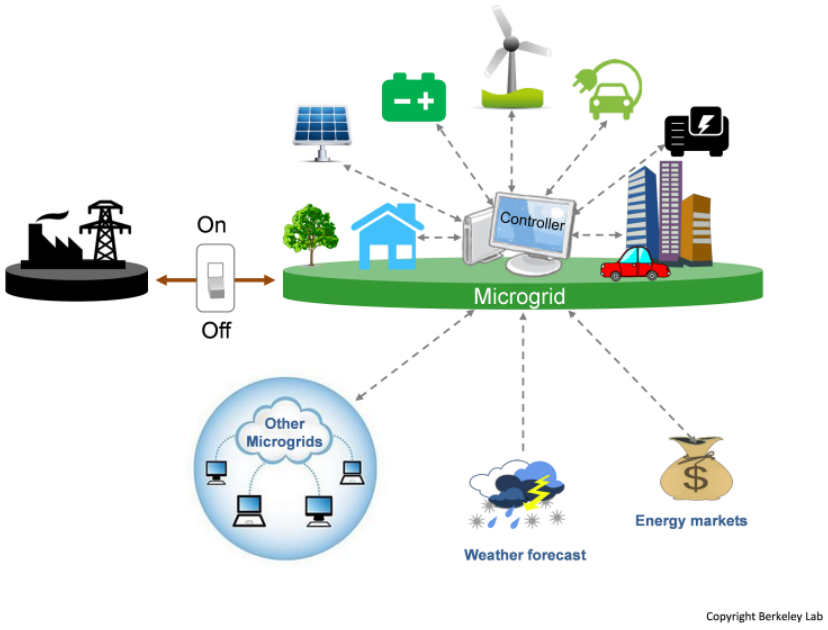


Figure 1.4: MG typical structure [6].

In a MG it is necessary to implement a hierarchical control structure managed by a multi-level controller in order to maximize the efficiency, the controllability, the operational costs and to ensure the correct operation. In fact, to find the optimum operating point of a MG it is necessary to consider multiple things: the power ratings, the distribution of loads and generations source, the amount of energy available from the stochastic RESs, the possible communications with other MGs or with main grid and also the costs. All these variables are contemplated in hierarchical control [15] that is mainly divided in three main levels:

- **Primary control level**, also called **local control**;
- **Secondary control level**, also called **MG-Supervisor Control**;
- **Tertiary control level**, also called **Grid-Supervisor Control**.

The MG hierarchical control is shown in Fig. 1.5.

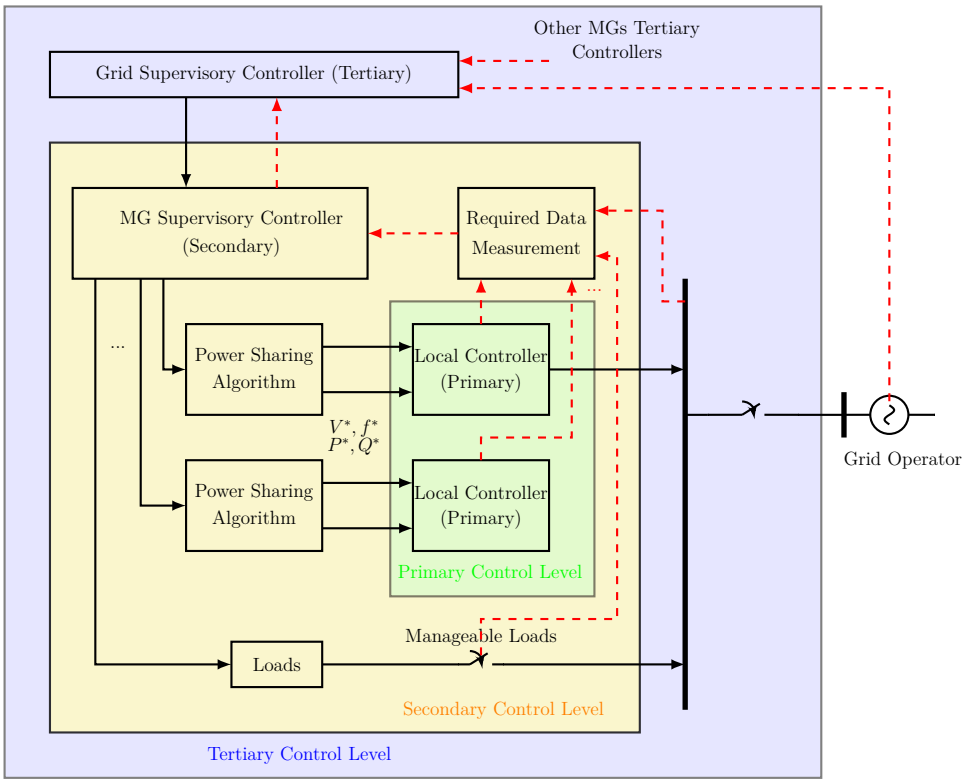


Figure 1.5: Microgrid Hierarchical Control

The Primary control level is implemented in every VSIs that can inject power into the MG. Its role is to control the local variables such as the output converter current or the frequency and the voltage at the PCC. There are two main operating modes that has to be considered: the Grid-Connected mode and the Stand-Alone (Islanded) mode.

In the first case, the MG is connected to the grid, which imposes voltage and frequency at the PCC. In this mode, the main task of the control system is to regulate the flow of active and reactive power injection according to the set-points imposed by a supervisory control. It is done adjusting the output converter current, regulating it with a grid-feeding control strategy.

In the second case, one power converter inside the MG must form the reference of voltage magnitude and frequency at the PCC. These objectives are achieved controlling it as a VSI using a voltage closed loop, that is the Grid-Forming control strategy.

The secondary control level uses a MG supervisory controller that has to restore the MG voltage and frequency at the nominal values compensating the steady-state errors. This level of control use communications and a monitoring system to coordinate the action of all the generations units inside an area. It has a slow dynamic compared with the primary control.
The tertiary control level employs a Grid Supervisory Controller to monitor and regulate the flow of active and reactive power among the MG, the main grid and the others small-scaled grids. It makes use of communication links to exchange informations with other MGs tertiary controllers and with the grid operator. Its goal is to optimize the economic performance of the grid giving optimal set-points of active and reactive power for the secondary controller. It is done by balancing the demand and the energy supply with marginal generation cost of each DG unit. There are several ancillary service that a MG could provide [14], such as:

- black-start capability, that is put into effect thanks to the coordination between the MG controllers and the transmission/distribution centralized controllers. In fact, the restoration after a complete or a partial black-out is done by energizing active island and then synchronizing them. This process is enhanced if in the MG there are sufficient EESs systems;
- transient voltage support during grid-faults and flicker reduction;
- power oscillation damping, that is allowed by the installation of EESs systems and by the coordinate action of the power controls forming the MG;
- island operation.

Control Schemes of Power Converter in AC Microgrids

Basically, there are three typology of control schemes used for power converters in AC MG ([14]):

- Grid-Forming;
- Grid-Feeding;
- Grid-Supporting.

The Grid-Forming power converters are VSIs controlled as an ideal voltage source and are in charge of setting the voltage amplitude and frequency of the local grid by using a proper control loop. This control operates at primary control level, imposing the local AC voltage. Obviously, if more than one power converters contribute to form the voltage magnitude and frequency in a single MG, because they work in parallel, the circulation of high current created by a mismatch at the output voltage/frequency must be avoided. So, a proper power sharing control must be added for each VSIs controller based on their power contribution. The most used method is the "droop control" [15].
The Grid-Forming control strategy is mostly used in island operating mode to create the reference of voltage magnitude and frequency.
The Grid-feeding power converters are mainly designed to deliver power to an energized grid. They are controlled as ideal current source and they should be perfectly synchronized with the AC voltage at the connection point. These power converters can operates in parallel with other grid-feeding power converters in grid-connected mode. In fact, they always need a system that "forms" the grid reference of voltage magnitude and frequency, for example they require the presence of at least one Grid-Forming power converter. The operation of the grid-feeding converters are often regulated by an high-level controller which set the active and reactive power reference that has to be delivered.
Moving on the last typology of power control scheme presented that is the Grid-Supporting control strategy, there are two main categories of it. In the first case it is controlled as a current source and in the second one it is regulated as a voltage source. These converters regulates their output current or voltage amplitude to

keep the value of the grid frequency and voltage amplitude close to their rated one. So, they participate to the regulation of the voltage and the frequency in both grid-connected and island modes.

Goal of the Thesis

This thesis focuses on the study of converters for MGs. These are small-scale grid which can supply local loads, by using distributed generation. The goal of this project is the analysis, simulation and experimental validation of MG converter control strategies. Moreover, an experimental test bench consisting of multiple converters has been assembled and tested.
The entire work done is divided in the following steps:

1. Study of different grid-forming control strategies and validation in PLECS simulations. It is also analysed a grid-supporting control strategy, which can be adopted in case the MG is connected to the main grid.

2. Elaboration and implementation of an unique grid-forming control C code that gives the chance to chose among the different strategies studied in the first step.

3. Experimental tests using a three phase two level inverter;

4. Mechanical design of two support structures where other two inverter units will be installed.

5. Assembly and testing of two printed circuit boards (PCBs) equipped with a field-programmable gate array (FPGA). They will be used to interface the control implemented on a rapid control prototyping system and the new converter units.

6. Implementation of the grid-forming control strategy on the new test bench. This step includes also the programming of the FPGA and the creation of communication protocol between dSpace and the board.

So, in the first chapter the innermost control layer in a MG will be analysed. In fact, various typologies of control for a Grid Forming converter will be studied to

test their functionality. In this case is considered that the MG works as an island and is not connected to the main grid. These controls will be tested by simulating various types of load (both active and passive) connected to it. The results obtained in PLECS simulation will be validated on the experimental test bench. Moreover, two modulation techniques will be compared, evaluating their effectiveness and also a study on the most suitable output filter typologies will be proposed.

In the second chapter, it is conducted a study on a Grid-supporting converter, where an external power loop is implemented. In this case, the MG is considered to work in grid-connected mode.

In the third chapter, it will be explained in detail the process of assembling and testing of the additional inverter units. This involves the assembly of PCBs, the design of a suitable mechanical assembly and the programming of an FPGA board to interface the power hardware and the controller.

Chapter 2

Grid-Forming Power Converters

The Grid-Forming technique is one of the strategies used to control a power converter that is working connected to the grid or to a microgrid. The main goal of this control strategy is to make the power converter *form the grid*. In fact, a power converter controlled as grid-forming has to set the reference voltage E^* and the reference frequency f^* at the point of common coupling (PCC).
They are controlled in closed loop to act as an ideal AC voltage sources. A simplified representation of this is shown in Fig. 2.1.

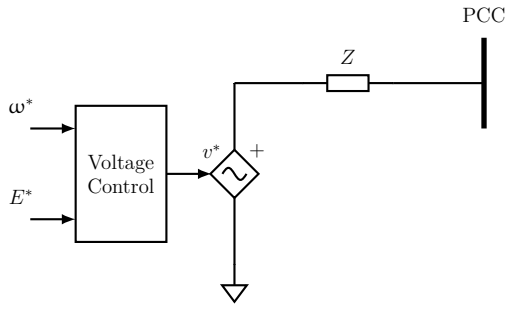


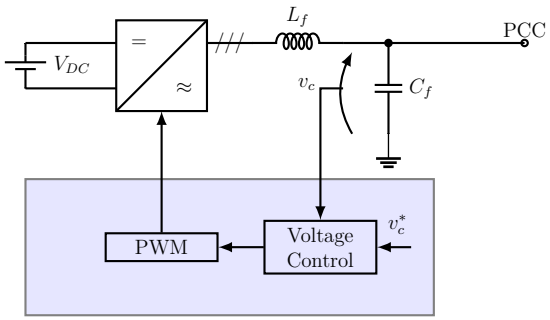
Figure 2.1: Grid-forming power converter controlled as an ideal voltage source.

In this chapter, the synthesis and the test of different Grid-Forming control

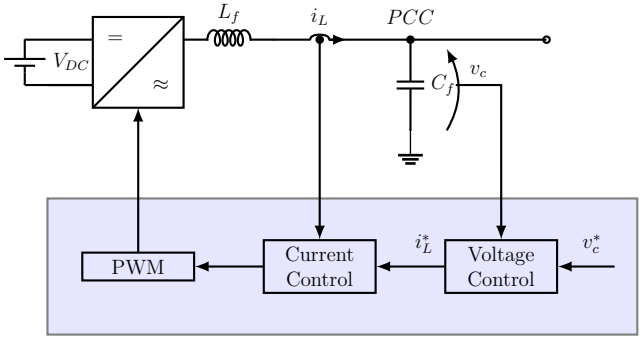
strategies for a power converter connected a microgrid that works in island operating mode will be presented.
A grid-forming power converter has an important role in a microgrid that is working in island operating mode. In fact, the AC voltage generated by it will be used as a reference for the other power converters connected to the AC microgrid bus or it will directly supply the loads. Therefore, it is important to build a stable control [15].
In this thesis two methods will be analysed:

- **Single loop Voltage Control**: a single voltage regulator controls the voltage at the PCC;
- **Dual loop Control**: a nested scheme which regulates the PCC voltage and the converter output current.

The equivalent single phase circuits are shown in Fig. 2.2.



(a) Single Loop Control.



(b) Dual Loop Control.

Figure 2.2: Grid-Forming Control Strategies equivalent single phase circuits.

The grid-forming power converters can be controlled in both the (α, β) stationary reference frame and the (d,q) rotating frame synchronous at the microgrid reference frequency (f^*).
Therefore, for the DC/AC power converter the following four control strategies are considered:

1. Single loop voltage control using a Proportional Integral (PI) controller;
2. Dual loop voltage control using Proportional Integral (PI) controllers;
3. Single loop voltage control using a Proportional Resonant (PRES) controller;
4. Dual loop voltage control using Proportional Resonant (PRES) controllers.

They will be compared in terms of load step response and of output current total harmonics distortions (THD).

2.0.1 Goals

In the next paragraphs, the system under study will be presented. Then, the control of the DC/DC and of the DC/AC power converters will be illustrated. In particular, the DC/AC will be controlled as Grid-Forming power converter. The tuning of the controllers will be described. Then, the effectiveness of the control will be validated by connecting active and passive loads. Moreover, all the topics covered in the study will be presented in details showing the results obtained using simulations in PLECS. These topics are:

- the influence of dead time and its compensation;
- comparison of the results obtained by using two different modulation techniques such as the Balanced Envelope Modulation PWM (PWM BEM) and the Discontinuous Pulse Width Modulation (DPWM);
- the choice of the best type of output converter filter for the system under study.

All the procedures and results obtained will be presented in detail. Finally, the experimental implementation steps and results will be shown and compared with the simulated ones.

2.1 System Under Study

The system under study is composed of a three phase two-levels inverter interfaced to the AC grid with an output LC filter as outlined in the scheme in Fig. 2.3. The inverter is supplied by a DC/DC converter, which regulates the DC-link voltage in order to act as an ideal voltage generator.

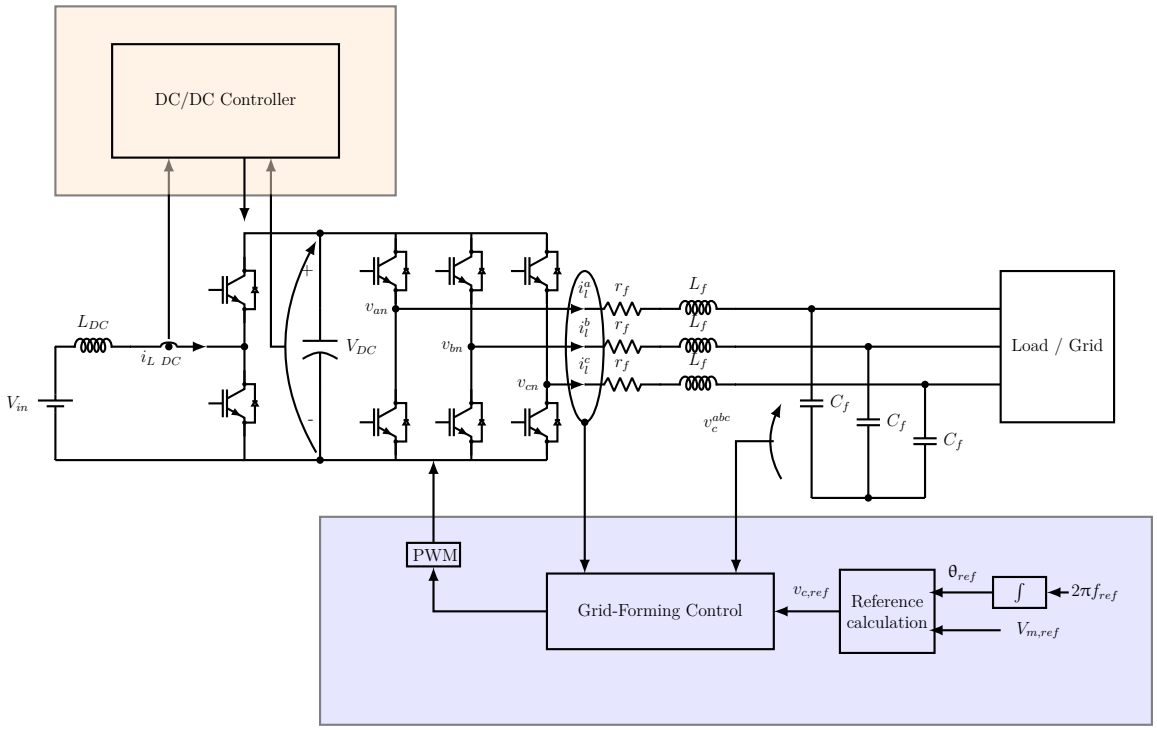


Figure 2.3: Equivalent circuit of the system in study.

2.2 DC/DC Power Converter

In this system the DC/DC converter behaves like a boost converter and a Cascaded Control Loop (CCL) is adopted to control it.
The CCL is composed of an external DC voltage loop and an inner DC current loop. The external one regulates the output voltage on the DC-link and it sets the reference current to the inner one, as shown in Fig. 2.4.

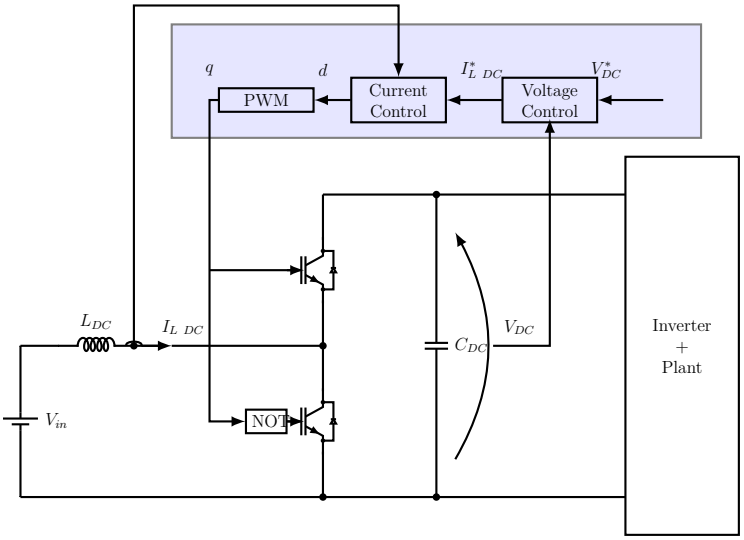


Figure 2.4: DC-side converter circuit Cascaded Control.

The structure of the CCL is outlined in the block scheme shown in Fig. 2.11.

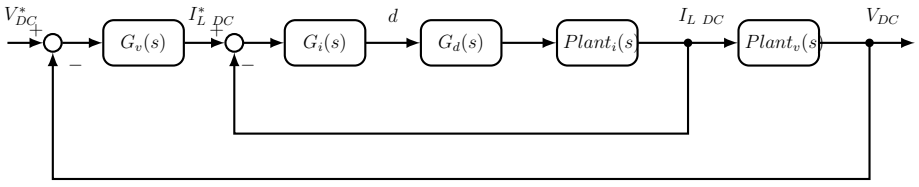


Figure 2.5: Cascaded control loop generic block scheme.

where

- $G_v(s)$ is the voltage loop controller transfer function;

- $G_i(s)$ is the current loop controller transfer function;
- $G_d(s)$ is the power converter delay transfer function;
- $Plant_i(s)$ is the transfer function of the current loop plant;
- $Plant_v(s)$ is the transfer function of the voltage loop plant;

Two PI regulators are used for both the voltage and the current controllers. They are tuned following these 3 steps:

1. design the inner current loop considering the output voltage V_{DC} constant, imposed by a ideal DC-source;
2. design the external voltage loop assuming an ideal inner current loop (it is justified by the bandwidth difference between the two control loops). Moreover, a constant load is considered simulated by and ideal current DC-source;
3. Test the complete designed control.

Tuning of the DC Current Regulator

The system considered is shown in Fig. 2.6.

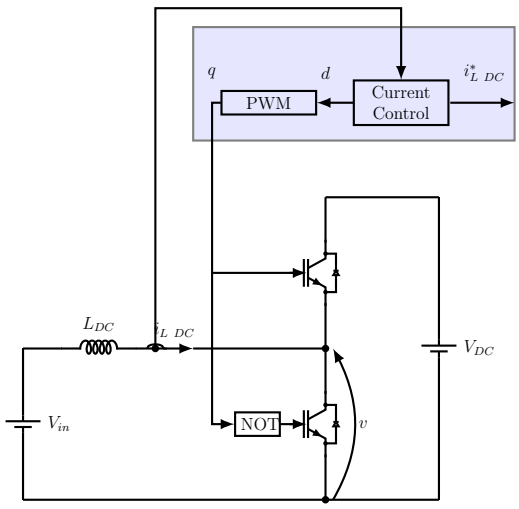


Figure 2.6: DC/DC converter circuit used to size the inner current loop.

The generic block scheme is outlined in the Fig. 2.7.

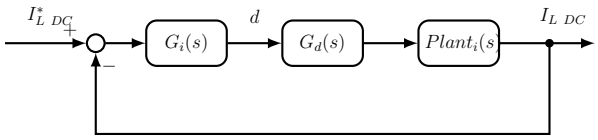


Figure 2.7: CCL: inner current loop block scheme.

where,

- $G_i(s) = k_p + \frac{k_i}{s}$ is the DC Current PI Regulator;
- the $G_d(s)$ transfer function represent the converter delay response. This delay is due to the fact that a digital control is used and. $G_d(s)$ is expressed by (2.1).

$$G_d = e^{-s\tau_d} \tag{2.1}$$

  assuming $\tau_d = 1.5 \cdot T_s$.

- $Plant_i(s) = \frac{I_{L,DC}}{d}$ is the plant of the current control loop. d represents the duty cycle generated to control in the right way.

In order to determine the correct formulation of $Plant_i(s)$, the second Kirchoff's law is applied in the circuit in Fig. 2.6 (2.2).

$$\frac{v - V_{in}}{s \cdot L_{DC}} = i_{L,DC} \tag{2.2}$$

where v depends on the duty cycle d value according to (2.3):

$$v = d \cdot V_{DC} \tag{2.3}$$

So, combining (2.2) and (2.3) the (2.4) is obtained:

$$i_{DC} = \frac{d \cdot V_{DC}}{s \cdot L_{DC}} + \frac{V_{in}}{s \cdot L_{DC}} \tag{2.4}$$

The first terms represents the plant of the current loop and the second term is the external disturbance. So,

$$Plant_i(s) = \frac{I_{L,DC}}{d} = \frac{V_{DC}}{s \cdot L_{DC}} \tag{2.5}$$

Now, it is possible to represent in detail the block diagram of the current loop, as shown in Fig. 2.8.

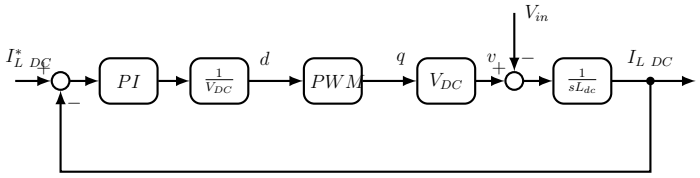


Figure 2.8: CCL: inner current loop block scheme in detail.

The Open Loop Transfer Function $T_{OL}(s)$ is calculated:

$$T_{OL}(s) = G_i(s) \cdot G_d(s) \cdot Plant_i(s) \tag{2.6}$$

After that, the Closed Loop Transfer Function $T_{CL}(s)$ is retrieved:

$$T_{CL}(s) = \frac{T_{OL}(s)}{1 + T_{OL}(s)} \tag{2.7}$$

The proportional and the integral gain are evaluated:

$$k_{p,i\ DC} = \omega_{b,\ i\ DC} \cdot L_{DC} \tag{2.8a}$$

$$k_{i,i\ DC} = 0.1 \cdot \omega_{b,\ i\ DC} \cdot k_{p,i\ DC} \tag{2.8b}$$

where

- $\omega_{b,\ i\ DC}$ is the current bandwidth;
- $k_{p,i\ DC}$, $k_{i,i\ DC}$ are the proportional and integral gain of the PI regulator.

Using the Software MATLAB the open and closed DC current loop transfer function bode diagrams are calculated and plotted. For the application, the power converter switching frequency is 10 kHz. Therefore, a frequency of $1000Hz$ is chosen for the current bandwidth according to:

$$f_{b,\ i\ DC} = \frac{f_{sw}}{10} = 1000Hz \tag{2.9}$$

The bode plots are reported in Fig. 2.9

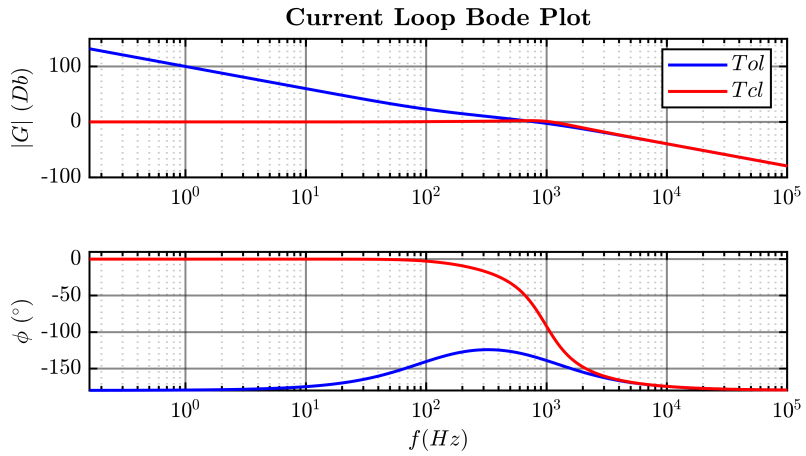


Figure 2.9: DC current open and closed loop bode plot.

Tuning of the DC Voltage Regulator

The system in study is illustrated in Fig. 2.10 .

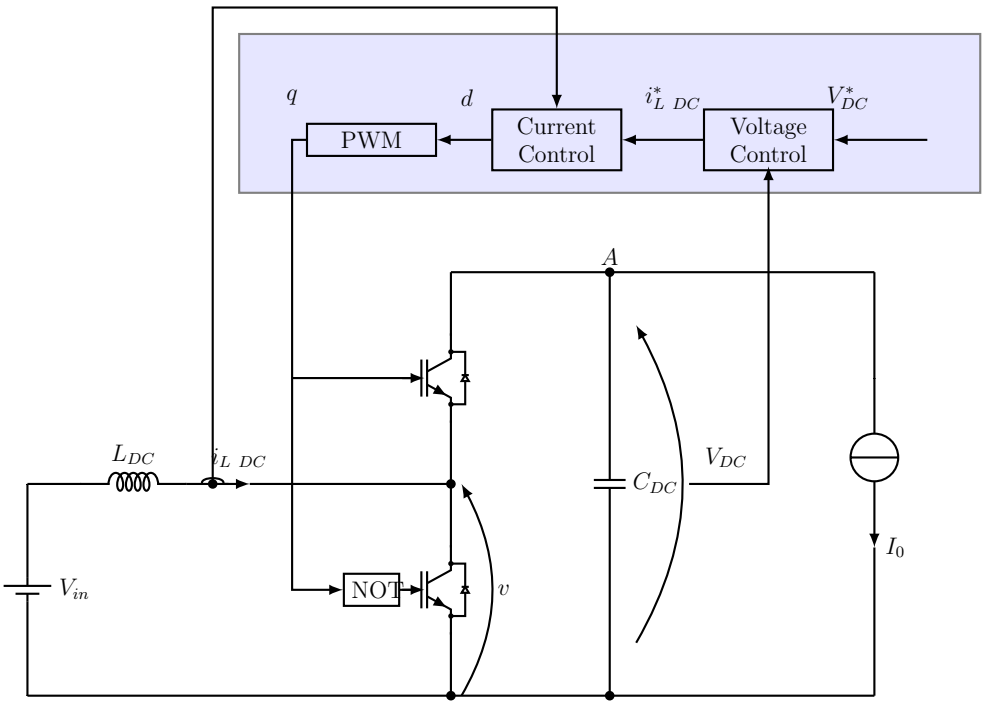


Figure 2.10: DC/DC converter circuit used to size the external voltage loop.

The complete Cascaded Loop Control block scheme shown in Fig. 2.11.

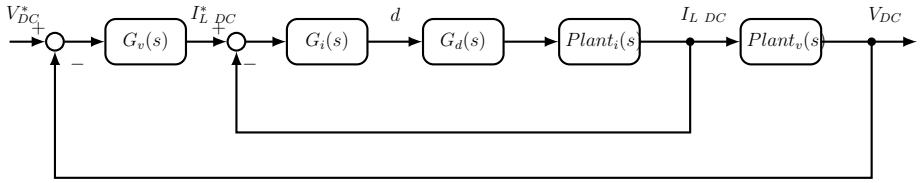


Figure 2.11: Cascaded control loop generic block scheme.

Assuming a voltage bandwidth of $20Hz$, the difference between the voltage and the current loop bandwidth is bigger than one frequency decade. So, the inner current loop has a faster response. Therefore, in the frequency range where the voltage will work, the closed loop transfer function of the inner current loop can be considered equal to one (Fig. 2.12).

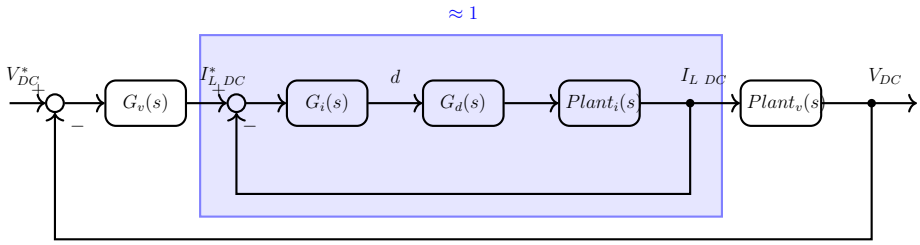


Figure 2.12: CCL block scheme.

So,

$$I^*_{L\ DC} \approx I_{L\ DC} \tag{2.10}$$

The block diagram simplified by this assumption is shown in Fig. 2.13.

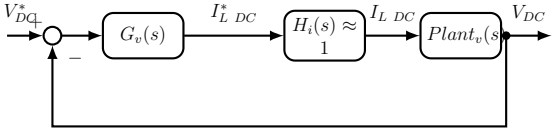


Figure 2.13: CCL: voltage loop block scheme.

where

- $G_v(s) = k_p + \frac{k_i}{s}$ is the DC Voltage PI Regulator;
- $Plant_v(s) = \frac{V_{DC}}{I_{L\ DC}}$ is the plant of the voltage control loop.

Applying the first law of Kirchoff at the node "A" in Fig. 2.10:

$$C_{DC}\frac{dv_c}{dt} = I_{LDC} - I_{LOAD} \tag{2.11}$$

So, passing to the Laplace domain:

$$v_c = \frac{1}{sC_{DC}} \cdot (I_{LDC} - I_{LOAD}) \tag{2.12}$$

The first terms in (2.12) represents the plant of the voltage loop and the second term is the external disturbance. So,

$$Plant_v(s) = \frac{V_{DC}}{I_{L\ DC}} = \frac{1}{s \cdot C_{DC}} \tag{2.13}$$

Using the (2.14) the open and closed loop transfer function are evaluated and the bode plot are using MATLAB (Fig. 2.14).

$$G_{OL}(s) = G_v(s) \cdot 1 \cdot Plant_v(s) \tag{2.14a}$$

$$G_{CL}(s) = \frac{G_{OL}(s)}{1 + G_{OL}(s)} \tag{2.14b}$$

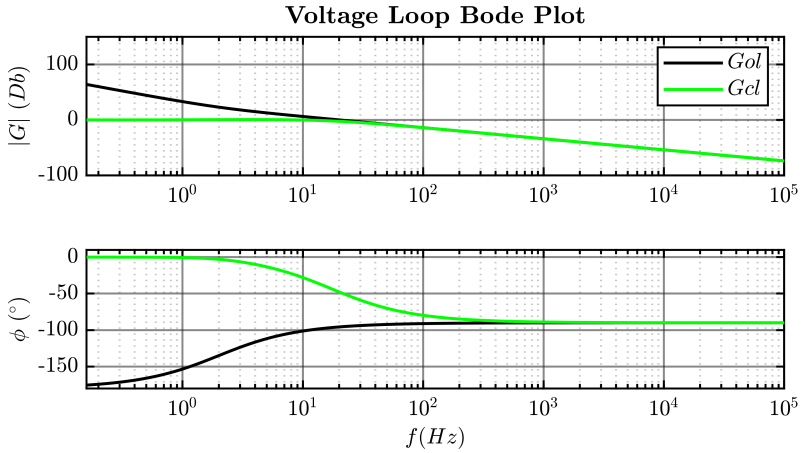


Figure 2.14: DC voltage open and closed loop bode plot.

The Fig. 2.15 summarizes the CCL bode plots of the transfer functions of the two voltage and current loops.

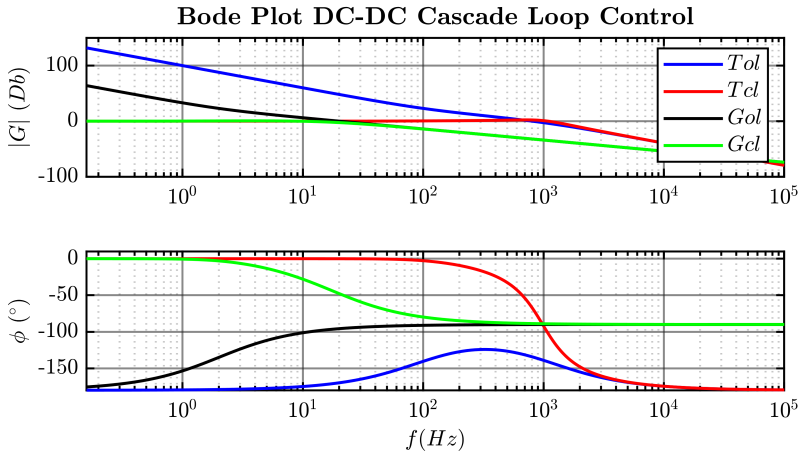


Figure 2.15: Cascaded Loop Control: Voltage and Current open and closed loop bode plot.

The control designed for the DC/DC converter is stable according to the Bode criterion.

2.2.1 PLECS Simulations Results of the DC/DC Cascaded Loop Control

In this section are reported the results obtained simulating in PLECS the Cascaded Loop Control where the DC/DC converter is supplying a resistive load.

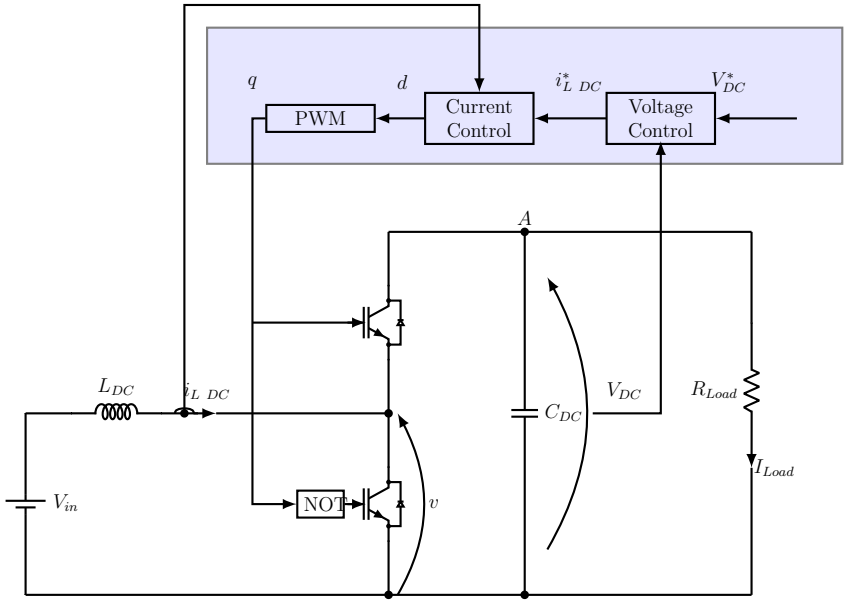


Figure 2.16: Cascaded Loop Control DC/DC converter electric circuit.

The Cascaded Loop Control is tested and the results obtained are shown in Fig. 2.17. The DC current and DC voltage measured follow the respective references, so it is possible to say that the Cascaded loop control works well on the DC/DC Converter.

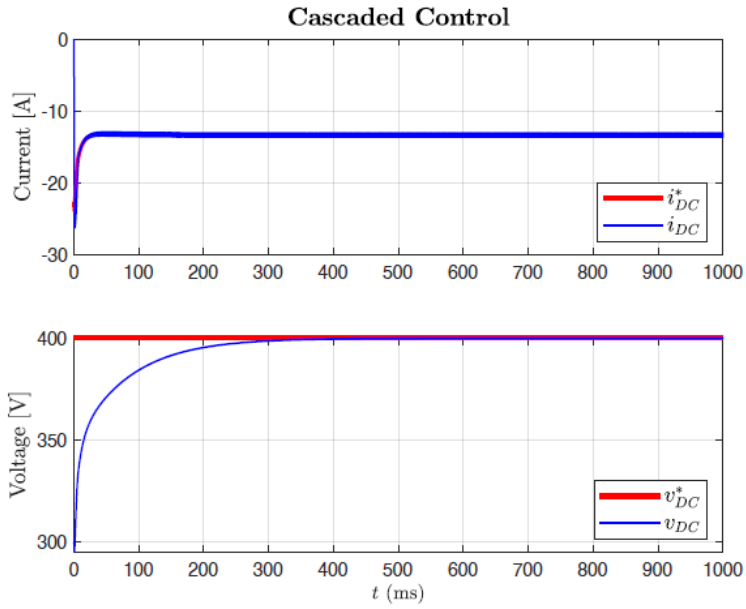


Figure 2.17: Cascaded Loop Control Voltage and Current simulation results.

2.3 DC/AC Power Converter

Assuming that the DC/DC converter works well, it is possible to consider the DC voltage, that supplies the DC/AC converter, is acting as an ideal voltage source. Under this hypothesis, the system under study become (Fig. 2.18):

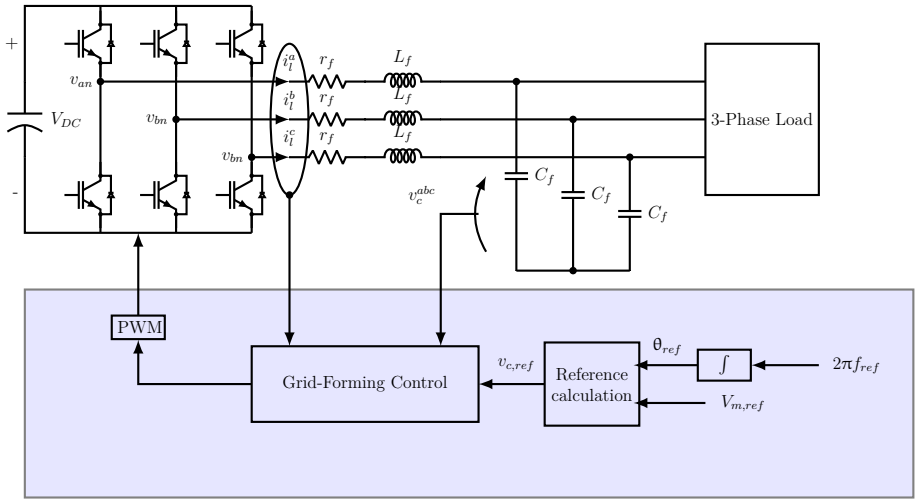


Figure 2.18: Equivalent three phase circuit of the system under study.

The equivalent single phase circuit is shown in Fig. 2.19.

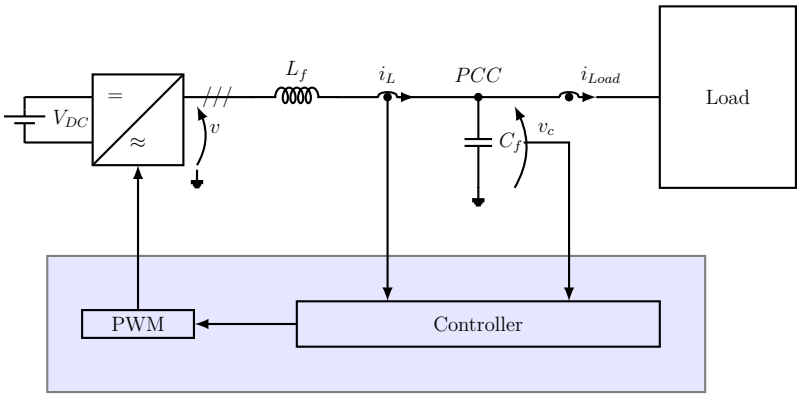


Figure 2.19: Equivalent single phase circuit of the system under study.

In order to be able to create a block scheme of the system, the relationships

among the following electrical quantities must be found:

- output converter voltage v;
- PCC voltage v_c;
- output converter current i_L;
- current absorbed by loads connected to the PCC ($i_{PCC} = i_{Load}$).

These relationships are ([29]):

$$Z_{0l} = -\frac{v_C}{i_{Load}} = \frac{Z_{lf}}{1 + Z_{lf} \cdot Y_{cf}} \tag{2.15}$$

$$G_{uv} = \frac{v_C}{v} = \frac{1}{1 + Z_{lf} \cdot Y_{cf}} \tag{2.16}$$

$$G_{ii} = \frac{i_L}{i_{Load}} = \frac{1}{1 + Z_{lf} \cdot Y_{cf}} \tag{2.17}$$

$$G_{ui} = \frac{i_L}{v} = \frac{Y_{cf}}{1 + Z_{lf} \cdot Y_{cf}} \tag{2.18}$$

where

$$Z_{lf} = r_F + sL_F \tag{2.19}$$

$$Y_{cf} = \frac{1}{r_C + \frac{1}{sC_F}} \tag{2.20}$$

r_F is the resistor in series with the filter inductance and r_C is the damping resistor of the output filter.

The block scheme in Fig. 2.20 represents the plant:

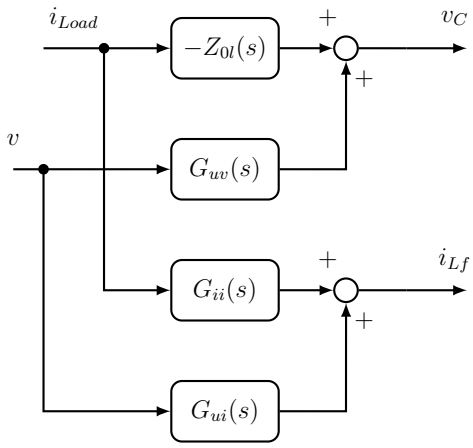


Figure 2.20: Plant equivalent block diagram.

2.3.1 Single Loop Voltage Control

The Single Loop Voltage Control is one of the strategies used to control a VSI as grid-forming converter. It consists of a closed voltage loop that control the PCC voltage.
In Fig. 2.21 and Fig. 2.22 are respectively shown the three phase and the equivalent single phase circuits of the system.

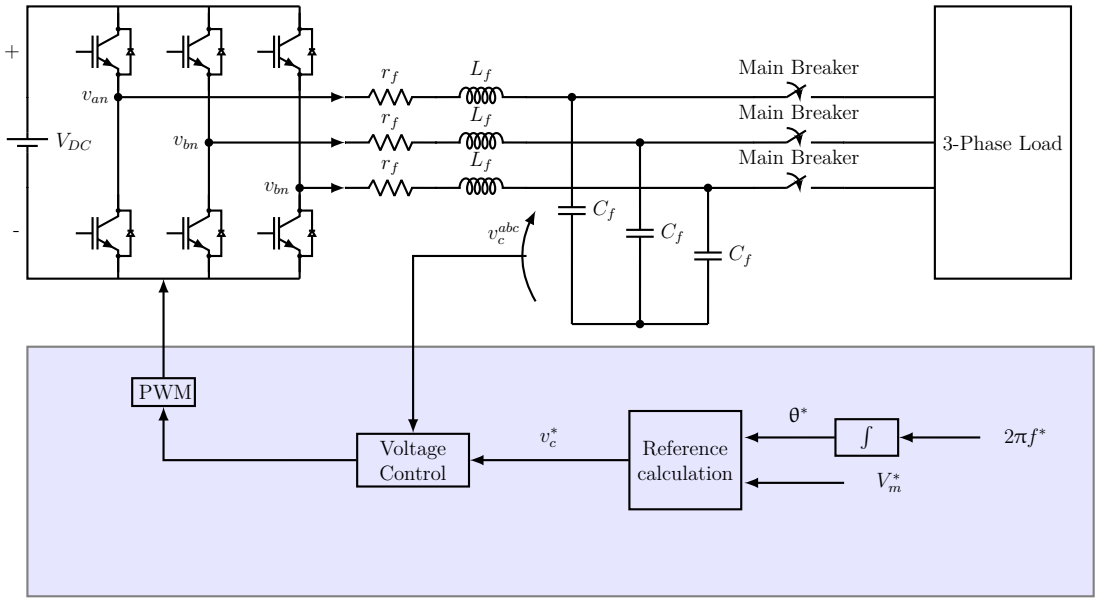


Figure 2.21: Single Loop Voltage Control: equivalent three phase circuit.

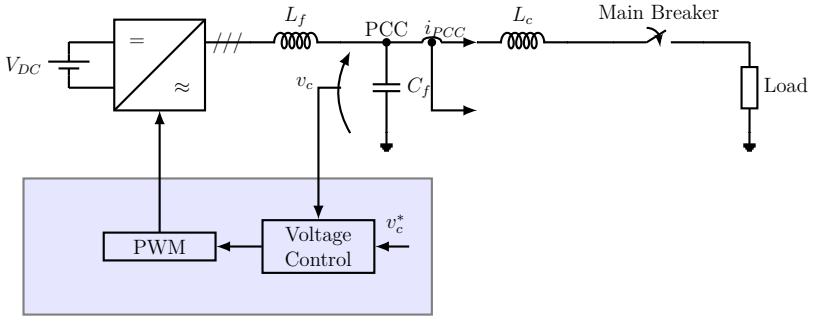


Figure 2.22: Single Loop Voltage Control: equivalent single phase circuit.

The voltage at the output filter capacitor is measured and compared to the reference one. The error between these two quantities is sent to the regulator. The PWM BEM technique is used.
The control is implemented both in the (α, β) stationary reference frame, adopting a PRES regulator, and in the (d,q) rotating frame synchronous with the reference frequency (f^*), using a PI regulator.
The block scheme of the Single Loop Voltage control is elaborated in Fig. 2.23.

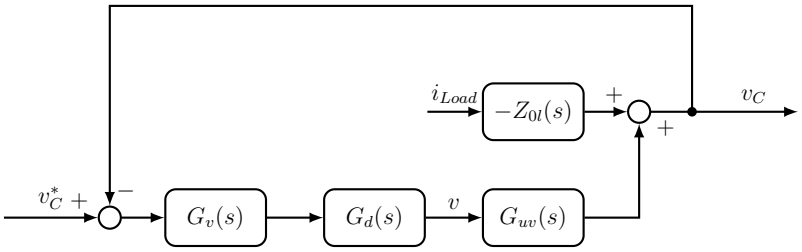


Figure 2.23: Block Scheme Single Loop.

where G_{uv} is expressed in (2.16) and its Bode Diagram is shown in Fig. 2.24.

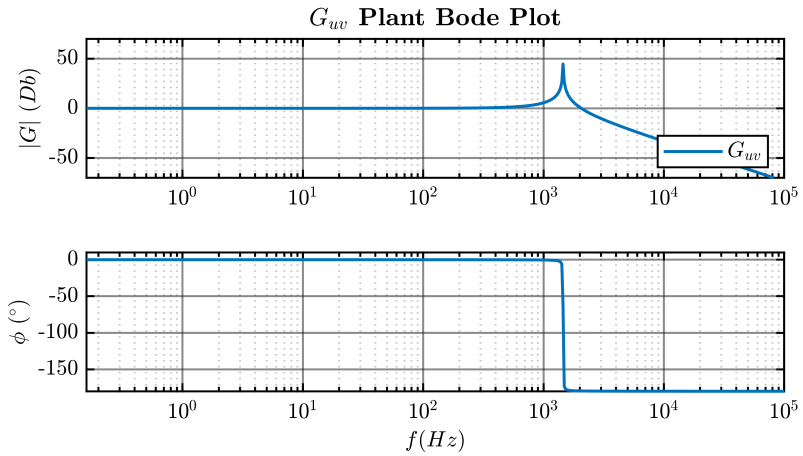


Figure 2.24: G_{uv} Bode Plot.

As it is visible from Fig. 2.24, the G_{uv} transfer function present an hight peak at the resonance frequency of the output filter. Because this situation can affect the control stability, a damping resistor is inserted in series to the output filter

capacitor. The choice of the best output filter damping topology will be described in section 2.4.
It is important to say that a digital control is used, so the converter delay response must be take into account. In fact, the $G_d(s)$ transfer function represent this delay and it is expressed by (2.1). From Fig. 2.23 it is possible to retrieved the open Loop (G_{OL}) and the closed Loop (G_{CL}) transfer functions:

$$G_{OL}(s) = G_v(s) \cdot G_d(s) \cdot G_{uv}(s) \tag{2.21}$$

$$G_{CL}(s) = \frac{G_{OL}(s)}{1 + G_{OL}(s)} \tag{2.22}$$

Moreover, the disturbance to output open loop transfer function is:

$$G_{OL\ Load}(s) = \frac{v}{i_{Load}} = \frac{-Z_{0l}(s)}{1 + G_{OL}(s)} \tag{2.23}$$

Tuning Of The Voltage PI Regulator

As in [18] and [17], the proportional and integral gains are chosen for the PI regulator.

According to the Bode criterion, the control is stable if the system has positive gain and phase margins and, according to the Nyquist stability criterion, the magnitude of the open loop transfer function must be less than one when its phase is at $-180° \pm 360°n$ (n integer):

$$|G(j\omega_{cf})_{OL}| < 1 \tag{2.24}$$

where ω_{cf} is the crossover frequency. The magnitude of the open loop transfer function crosses the 0 dB axis at this frequency in the bode plot.
Equation 2.24 guarantees a positive gain margin. Instead, a positive phase margin is ensured if the crossover frequency is limited according to (2.25), as it is suggested in [29].

$$\frac{\omega_s}{8} < \omega_{cf} < \frac{\omega_s}{6} \tag{2.25}$$

where ω_s is related to the converter switching frequency.
For this application, $\omega_{cf} = 2\pi\frac{f_s}{7}$ is chosen.

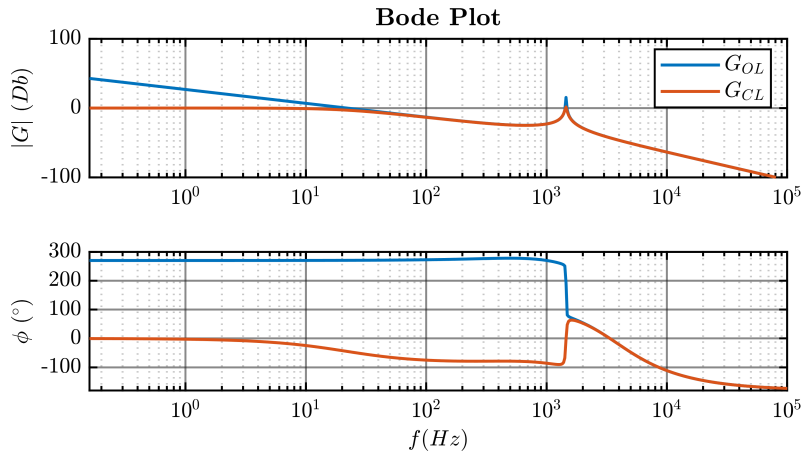


Figure 2.25: Open and Closed Loop Bode Plots, PI controllers used.

In order to size the PI regulator, the first step was to calculate the necessary $k_{p,v}$. To do it, at first, only a proportional regulator $G_v = k_{p,v}$ was considered and the converter delay effect was neglected. With these hypotheses the G_{OL} become:

$$G_{OL}(s) = \frac{1 + s^2}{\omega_r^2} \tag{2.26}$$

where w_r is the output filter resonance frequency:

$$w_r = \sqrt{\frac{1}{L_F \cdot C_F}} \tag{2.27}$$

Applying (2.24), $k_{p,v}$ is calculated using (2.28).

$$k_{p,v} = \frac{|\omega_r^2 - \omega_{cf}^2|}{\omega_r^2} \cdot 0.9 \tag{2.28}$$

The coefficient "0.9" is used in order to respect the relationship in (2.24).
After that, the integral gain $k_{i,v}$ is designed in order to avoid steady state error ((2.29)).

$$k_{i,v} = \omega_{cf} \cdot 0.5 \cdot k_{p,v} \tag{2.29}$$

Finally, using MATLAB the bode plots of the open and closed loop transfer functions are obtained (Fig. 2.25). According to the Bode criterion, the control is stable if the system is working with no load. However, in order to verify if the control is able to reject any additive disturbance, a load step is applied to the the system.

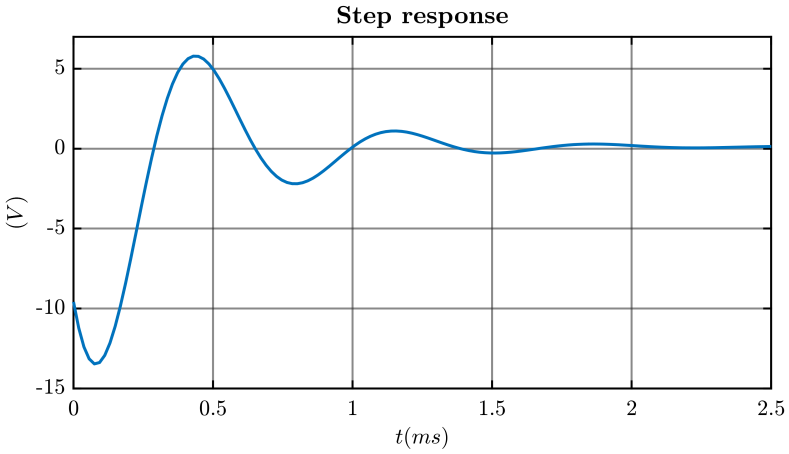


Figure 2.26: Single Loop Voltage Control using PI controllers: additive disturbance step response.

To do this, the MATLAB *step* function is used. Fig. 2.26. It is visible that there is an initial transient which at the end tends to zero. This means that the control designed in this way is able to reject any disturbance. In this example, it is considered that the load absorbs the 8% of the converter rated power (S_n).

Tuning Of The Voltage PRES Regulator

The Proportional Resonant Controller (PRES) is composed by two parts:

1. Proportional part;
2. Integral part.

The proportional part provides a signal output which is proportional to the error between the reference quantity and the measured quantity. Instead, the integral part is a second order generalized integrator. It is a double integrator that achieves an infinite gain at a certain frequency, called resonance frequency and almost no attenuation exists outside this frequency. In fact, it acts as a notch filter in a little range of frequency around the resonant one [26]. The PRES transfer function is shown in (2.30):

$$PRES(s) = k_p + k_i \cdot \frac{s}{s^2 + \omega_0^2} \tag{2.30}$$

where

k_p is the PRES controller proportional gain;

k_i is the PRES controller integral gain;

ω_0 is the PRES controller resonant frequency, that is tuned on the microgrid reference frequency f^*.

The following consideration are retrieved based on the [11], [26] and [8]:

- PRES controllers are similar to common PI controllers. The difference between them is that PRES controllers will only integrate frequencies very closed to the resonance one and will not introduce stationary error or phase shift.
- in PRES controllers k_p setting is based on the desired cross-over angular frequency (as in a PI regulator);
- in PRES controllers k_i setting is based on the desired transient response and on the specific phase margin required. The integral gain k_i determines the filter selectivity: higher value of k_i leads to lower filter selectivity faster settling time;

For this application, the PRES proportional and integral gains are chosen equal to the PI ones because:

- The same k_p guarantees the same bandwidth for both the regulator type;
- The same k_i ensures the same transient response at the fundamental frequency.

Furthermore, in grid applications the grid frequency is typically constant, but still it is allowed to vary with 1%. So, the choice of the integral gain k_i must take into account this aspect too.
The PRES proportional and integral gains are calculated by (2.28) and (2.29) for the single loop voltage control implemented in the (α, β) stationary reference frame. In Fig. 2.27 and Fig. 2.28 are respectively shown the open and closed loop Bode plots and the response of the control to the addictive disturbance step. Finally, it is possible to conclude that the control is stable according to the Bode criterion and it is also able to reject disturbances.

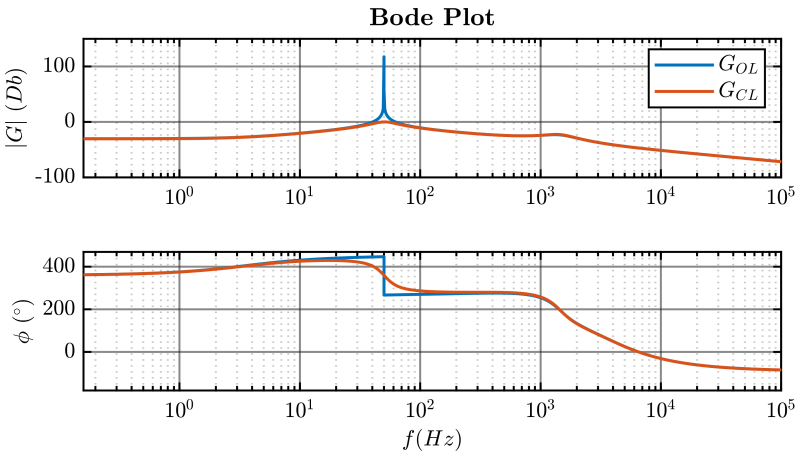


Figure 2.27: Open and Closed Loop Bode Plots, PRES controllers used.

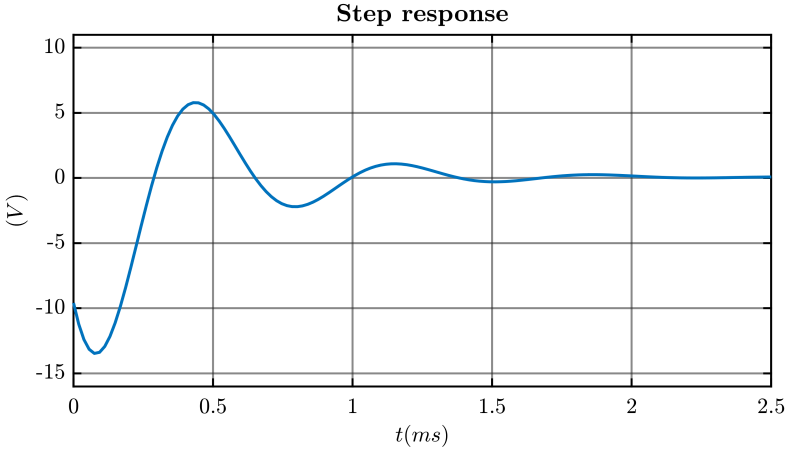


Figure 2.28: Single Loop Voltage Control using PRES controllers: additive disturbance step response.

2.3.2 Single Loop Voltage Control: Validation in PLECS Simulation

The effectiveness and robustness of the two methods studied to implement the Single Loop Voltage control has been validated in simulation PLECS, modelling the system presented in Fig. 2.21. The control was written using a *C-script* block, which is a specific element in the PLECS library that allows the user to code in C. As it was anticipated in the subsection 2.3.1, the control was implemented and

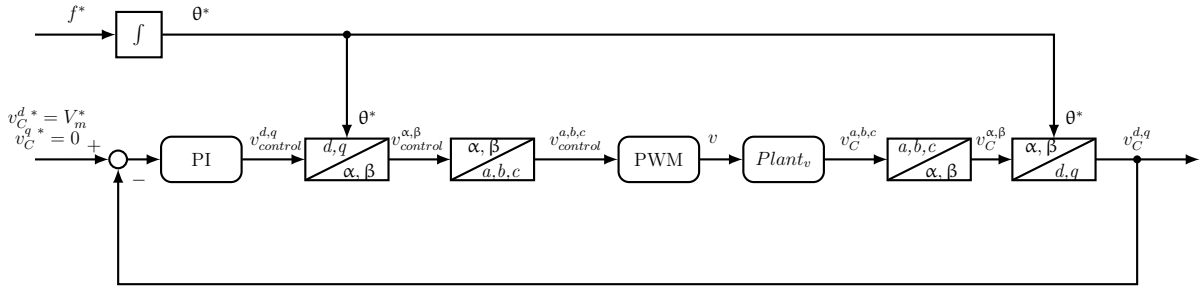


Figure 2.29: Single Loop Voltage Control in (d,q) synchronous rotating frame.

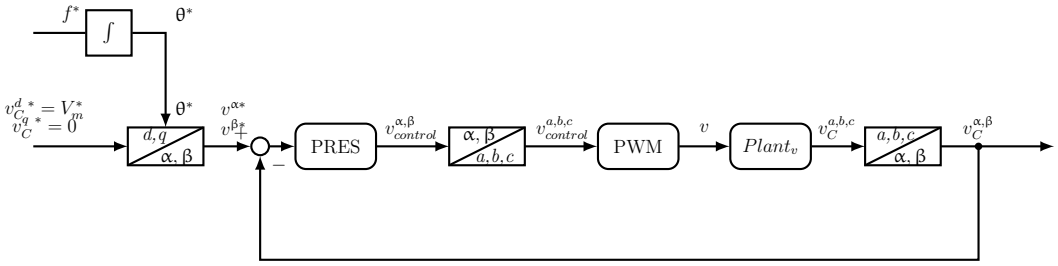


Figure 2.30: Single Loop Voltage Control in $(\alpha,\ \beta)$ stationary reference frame.

simulated in:

1. (d,q) synchronous rotating frame, using a PI regulator (Fig. 2.29);

2. (α, β) stationary reference frame, adopting a PRES regulator (Fig. 2.30).

The α, β-transformation, also called Clarke transformation, is used to pass from the (a,b,c) three phase reference frame to (α, β) stationary reference frame. Then, a rotation is applied to move from (α, β) stationary reference frame to the (d,q) synchronous rotating frame [8]. The two block schemes in Fig. 2.29 and in Fig. 2.30 are retrieved according to the previous considerations: In both the situations, the external reference voltage (V_m^*) is synchronized with the d-axis and the reference q-axis voltage is set to 0. Moreover, the rotation angle used is generated by integrating the set reference frequency.
Fig. 2.31 shows the PLECS simulation results of the single loop voltage control implemented in (d,q) axis using PI controllers. However, in Fig. 2.32 are reported

the waveforms obtained running the control simulation implemented in (α, β) axis using PI controllers.
The same step load is applied in both the simulations. It is chosen a resistive load that absorbs 8% of the converter rated power (S_n) according to the following equations:

$$P_{load} = x\% \cdot S_n \tag{2.31}$$

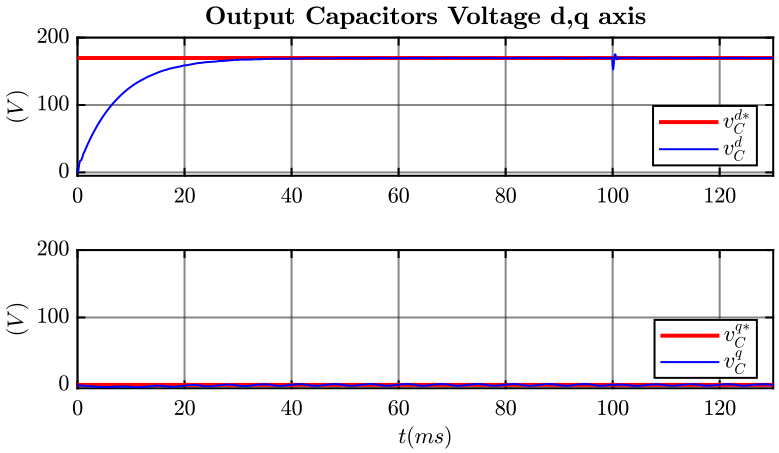
$$R_{load} = \frac{3 \cdot V_{C\ ph}}{P_{load}} \tag{2.32}$$

In the end, the two control strategies of the single loop voltage control give good results: the control is stable and is insensitive to disturbances in both the situations.
At the moment of the step load, the voltage drops. It is discovered that this drop also depends on the type of damping system used in the output filter. This topic will be discussed later in section 2.4.
The total harmonic distortion (THD) is used in order to compare the two strategies. From the literature it is known that the THD is a measurement of the harmonic distortion present in a signal and is defined as the ratio of the sum of the powers of all harmonic components to the power of the fundamental frequency. The THD is evaluated in the PCC output three phase current, that is the current absorbed by the load. The results of this analysis are summerized in Table 2.1.

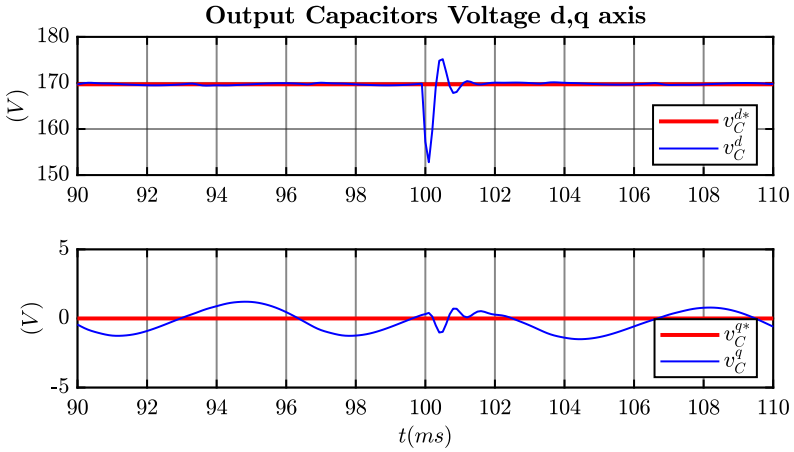
Single Loop Voltage Control - THD		
PI	**PRES**	**Percentage Difference**
$\approx 3.450\%$	$\approx 3.451\%$	$\approx 0.1\%$

Table 2.1: i_{PCC} THD Single Loop Voltage Control.

It is clear that both control strategies work well and with almost the same performances. Cause the THD is less than 5%, according to the normative, the signal is considered sinusoidal.

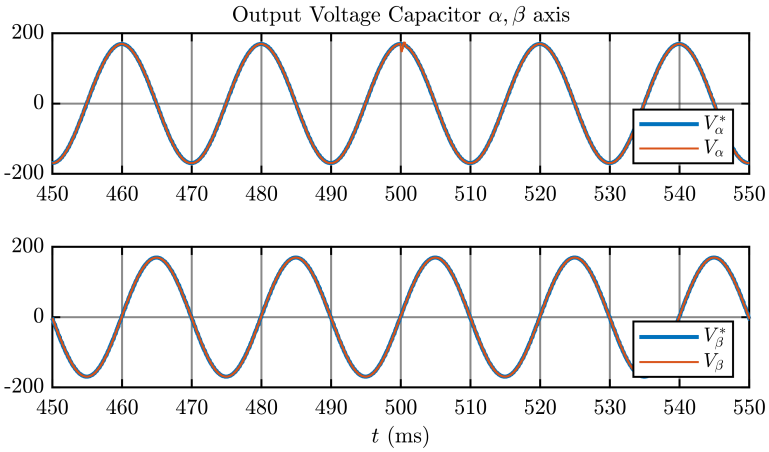


(a) Reference and measured voltage.

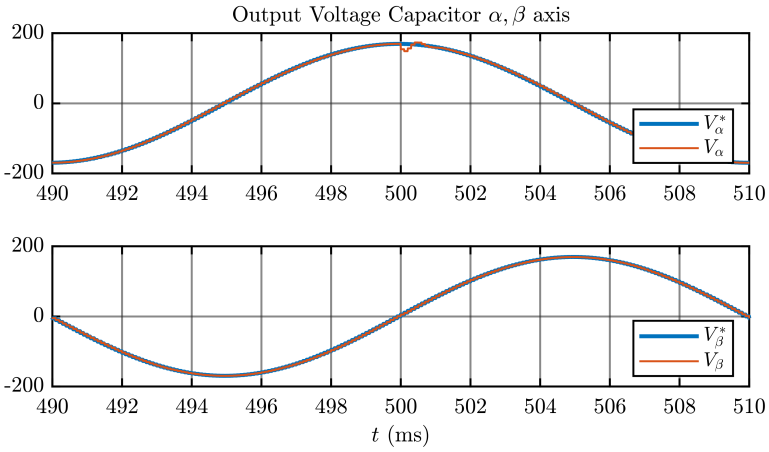


(b) Magnification at the load step.

Figure 2.31: Single loop voltage control implemented in (d,q) axis using PI controllers: PLECS simulation results.



(a) Reference and measured voltage.



(b) Magnification at the load step.

Figure 2.32: Single loop voltage control implemented in (α, β) axis using PI controllers: PLECS simulation results.

2.3.3 Dual Loop Control

The Dual Loop Control is another strategy used to control a VSI as grid-forming converter. It is a *nested* control and it consists of an inner current loop that controls the output converter current according to the reference setted by an external loop which regulates the voltage at the PCC. In Fig. 2.33 and Fig. 2.34a are respectively shown the three phase and the equivalent single phase circuits of the system.

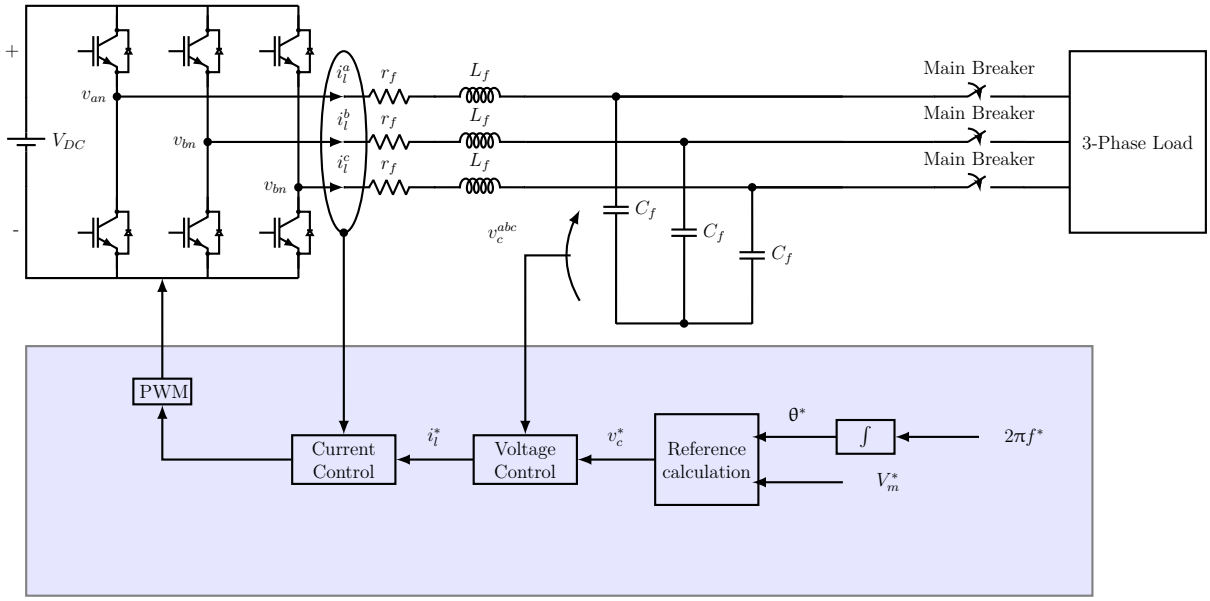
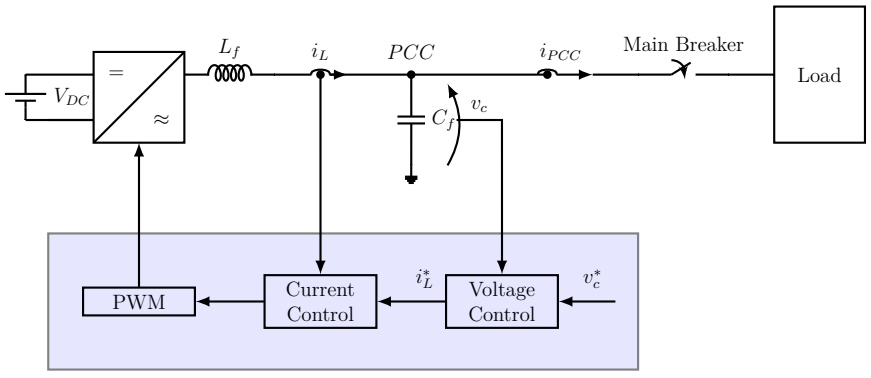


Figure 2.33: Dual Loop Control: equivalent three phase circuit



(a) Dual Loop Control: equivalent single phase circuit.

The output filter capacitor voltage is measured and compared to the reference one. The error between these two quantities is sent to the voltage regulator, which set the reference current for the inner current loop. The PWM modulation technique is used.

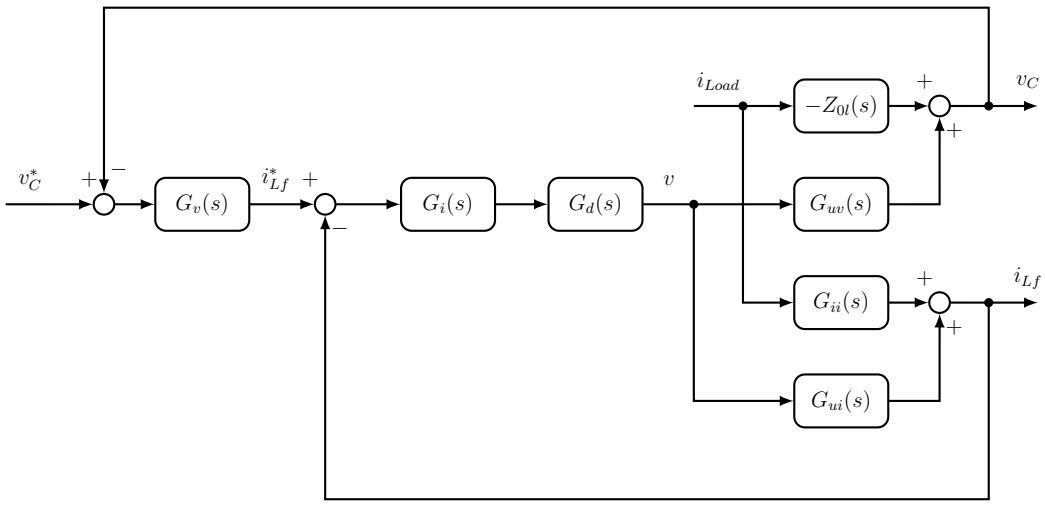


Figure 2.35: Block Scheme Dual Loop.

Also in this case, the control strategy will be implemented in

1. (α, β) stationary reference frame, adopting a PRES regulator;
2. (d,q) rotating frame synchronous with the reference frequency (f^*), using a PI regulator.

It is important to remember that a digital control is used, so the converter delay response must be take into account. In fact, the $G_d(s)$ transfer function represent this delay and it is expressed by (2.1).
These step were followed in order to size the control:

1. design the inner current loop regulator in both (d,q) and (α, β) frames;
2. design the external voltage loop regulator in both (d,q) and (α, β) frames;
3. implement and validate the dual loop control using a complete PLECS simulation, which combines the two loops properly.

2.3.4 Dual Loop Control: Inner Current Loop

The current loop is highlight in the Fig. 2.36.

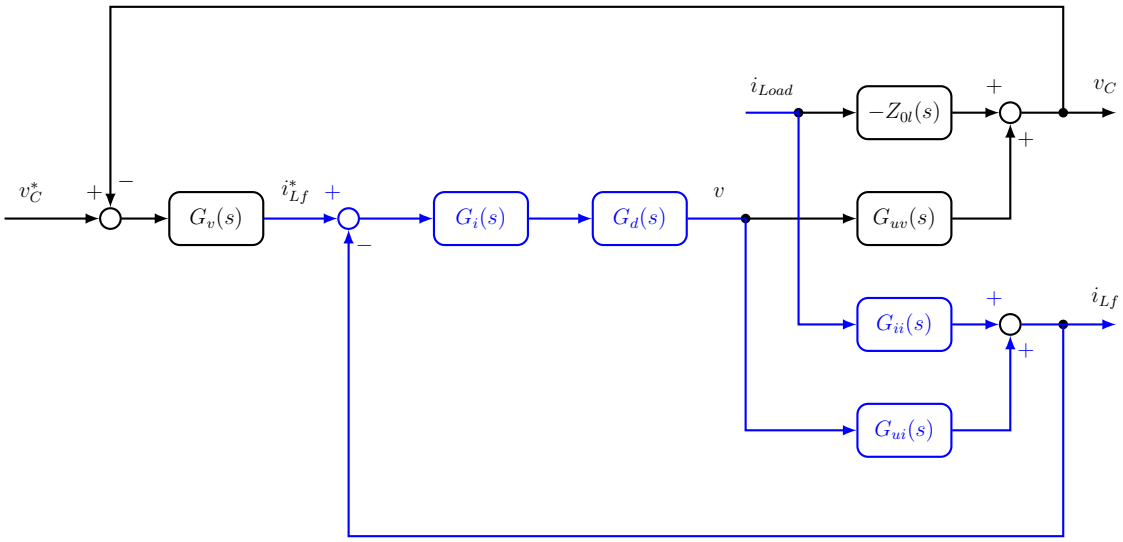


Figure 2.36: Highlight current control loop - Dual Loop Control Scheme.

So, in order to design the regulator it is necessary to consider the block scheme in Fig. 2.37.

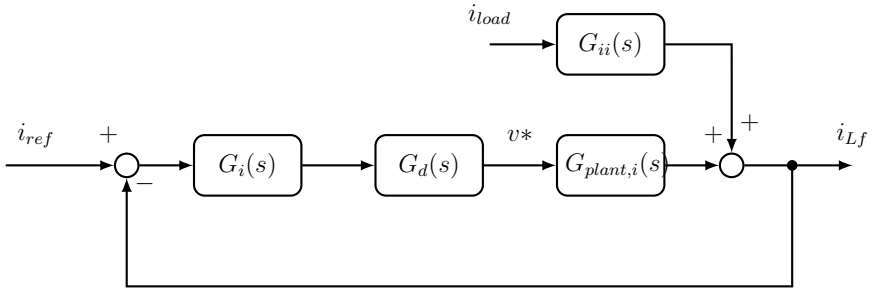


Figure 2.37: Dual loop control: inner current loop block scheme.

where

$G_i(s)$ is the current regulator;

$G_d(s)$ is the power converter delay;

$G_{plant,i}(s)$ is the plant of the current loop transfer function.

From the literature $G_{plant,i}(s) = G_{ui}$ ((2.16)). However, considering that the PCC voltage is controlled by the external voltage loop, it is possible to say that the rms current absorbed by the output filter capacitors is always constant. So, it is possible to simplify the current loop plant as in (2.33).

$$G_{plant,i}(s) = \frac{1}{Z_{lf}} \tag{2.33}$$

The open and the closed loop transfer functions are:

$$T_{OL}(s) = G_i(s) \cdot G_d(s) \cdot G_{plant,\ i} \tag{2.34}$$

$$T_{CL}(s) = \frac{T_{OL}(s)}{1 + T_{OL}(s)} \tag{2.35}$$

Tuning Of The Current PI Regulator

To implement the control in (d,q) axis two PI regulators are chosen for the current and voltage control. Bode and Nyquist criteria must be respected to build a stable control. So, first $k_{p,\ i}$ is chosen to obtain a current bandwidth ($f_{b,\ i}$) of 1000 Hz, and then $k_{i,\ i}$ is retrieved to obtain zero steady state error. Therefore, it is chosen:

$k_{p,\ i\ PI} = w_{b,\ i} \cdot L_F$

$k_{i,\ i\ PI} = f_{b,\ i} \cdot 0.1 \cdot k_{p,\ i\ PI}$

The bode plots of the open and closed loop transfer functions are shown in Fig. 2.38.

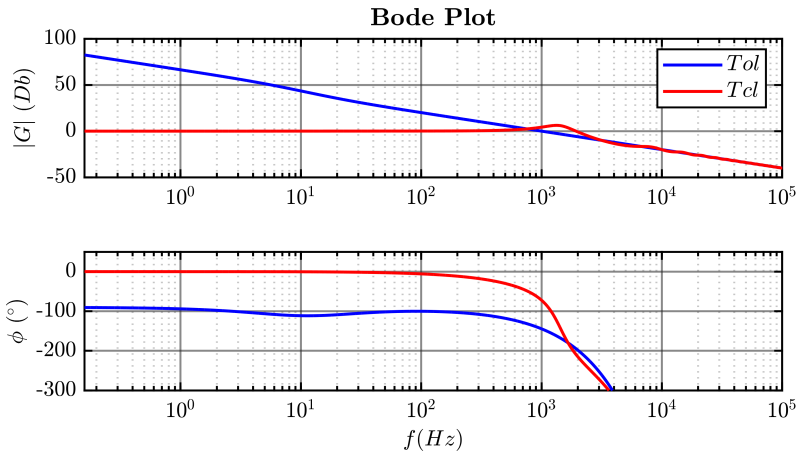


Figure 2.38: Open and Closed Loop Bode Plots, PI controllers used.

Moreover, to test the ability of the control to reject disturbances, the MATLAB *step* function is applied to the $G_{ii}(s)$ transfer function ((2.17)). The result obtained is illustrated in Fig. 2.39.

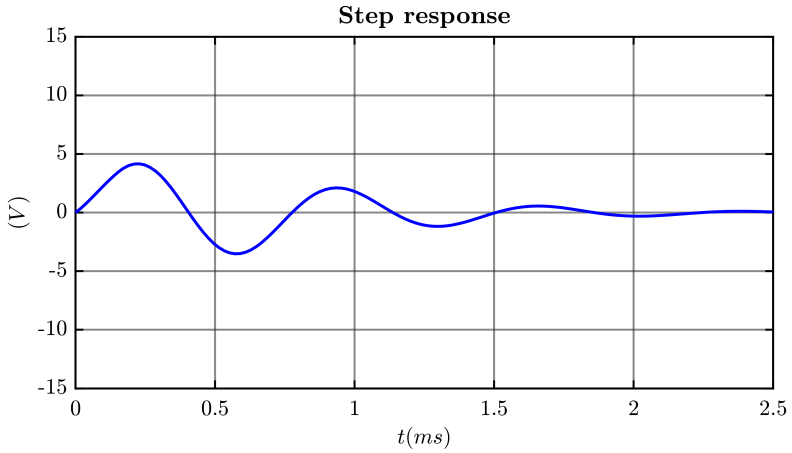


Figure 2.39: Step load response.

Designing the PI regulator in this way it is possible to say that the current control for the dual loop strategy is stable and robust.

Tuning Of The Current PRES Regulator

A PRES regulator is adopted when the dual loop voltage control is implemented in α, β axis. To obtain the same dynamic results the proportional and integral gains of the PRES regulator are chosen with the same values adopted respectively for the proportional and integral gains for the PI controller. So,

$$k_{p,\ i\ PRES} = k_{p,\ i\ PI}$$

$$k_{i,\ i\ PRES} = k_{i,\ i\ PI}$$

The bode plots open and closed loop transfer function obtained using MATLAB are reported in Fig. 2.40.

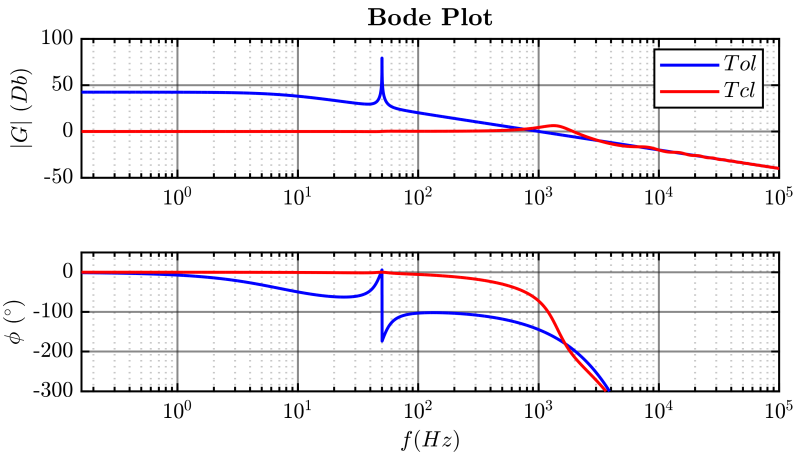


Figure 2.40: Open and Closed Loop Bode Plots, PRES controllers used.

Applying the MATLAB *step* function to the $G_{ii}(s)$ the result obtained is shown in Fig. 2.41.

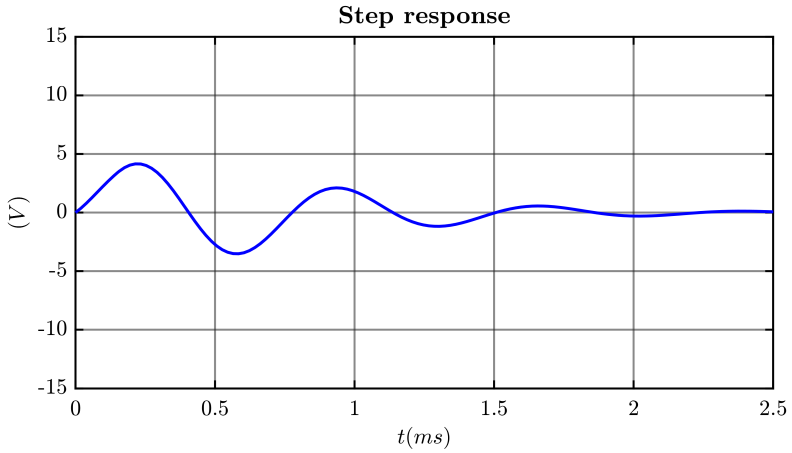


Figure 2.41: Step load response.

The control is stable and it is able to reject disturbances.

2.3.5 Dual Loop Control: External Voltage Loop

In the Dual Loop Control strategy the external voltage loop regulates the PCC voltage and set the reference for the inner current loop. The voltage loop is highlighted in Fig. 2.42.

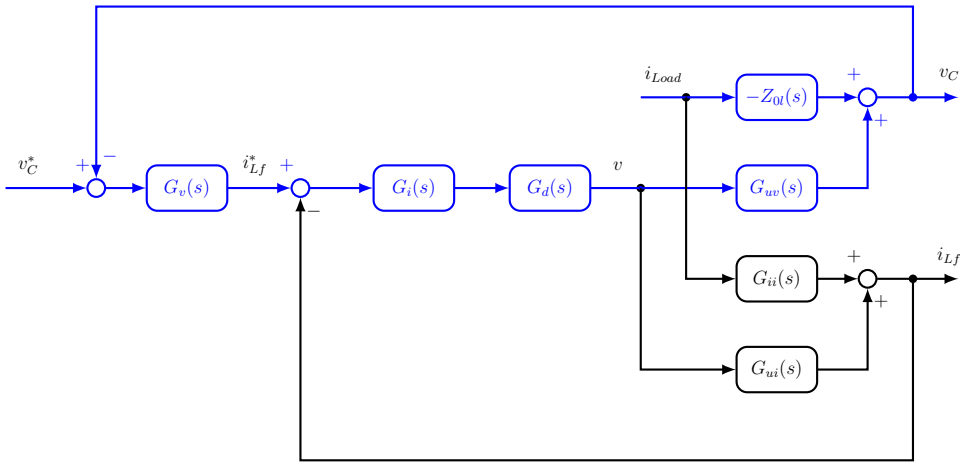


Figure 2.42: Highlight voltage control loop - Dual Loop Control Scheme.

In order to design the voltage regulator the block scheme in Fig. 2.43 is used as reference to retrieved the open ($G_{OL}(s)$) and closed ($G_{CL}(s)$) loop transfer functions.

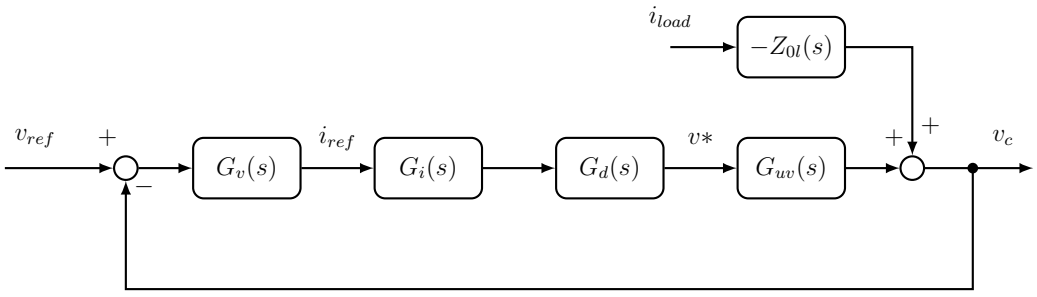


Figure 2.43: Dual loop control: external voltage loop block scheme.

Therefore,

$$G_{OL}(s) = G_v \cdot G_i(s) \cdot G_d(s) \cdot G_{uv}(s) \tag{2.36}$$

$$G_{CL}(s) = \frac{G_{OL}(s)}{1 + G_{OL}(s)} \tag{2.37}$$

Tuning Of The Voltage PI Regulator

A PI regulator is adopted when the dual loop voltage control strategy is implemented in (d,q) synchronous rotating frame. According to the Nyquist stability criterion, the magnitude of the open loop transfer function must be less than one when its phase is at $-180° \pm 360°n$ (n integer):

$$|G(j\omega_{cf})_{OL}| < 1 \tag{2.38}$$

where ω_{cf} is the voltage loop crossover frequency. (2.38) guarantees a positive gain margin. Instead, as suggested in [29], a positive phase margin is also ensured if the crossover frequency is limited according to the (2.39)

$$\omega_{cf} < \frac{\omega_s}{6} \tag{2.39}$$

where ω_s is related to the converter switching frequency.
For this application, $\omega_{cf} = 2\pi\frac{f_s}{8}$ is chosen. Considering that the expression of $G_{OL}(s \to j\omega)$ is:

$$G_{OL}(j\omega) = \frac{k_{p,\,v}}{1 - L_f \cdot C_f \cdot \omega^2} \cdot (k_{p,\,i} + \frac{k_{i,\,i}}{j\omega}) \tag{2.40}$$

where $k_{p,\,i}$ and $k_{i,\,i}$ are the proportional and integral gains of the current loop PI regulator Equation 2.3.4. Therefore, combining (2.40) and (2.38) the (2.41) is obtained:

$$|G_{ol}(j\omega_{cf})| = |\frac{k_{p,\,v} \cdot \omega_r^2}{\omega_r^2 - \omega_{cf}^2} \cdot (k_{p,\,i} + \frac{k_{i,\,i}}{j\omega_{cf}})| = 1 \tag{2.41}$$

Managing the (2.41), the $k_{p,\,v}$ is retrieved:

$$k_{p,\,v\;PI} = \frac{(\omega_r^2 - \omega_{cf}^2) \cdot \omega_{cf}}{\omega_r^2 \cdot (k_{p,\,i\;PI} \cdot \omega_{cf} + k_{i,\,i\;PI})} \tag{2.42}$$

After that, the integral gain is chosen to obtain a steady state error.

$$\omega_z = \omega_{cf} \cdot 0.5 \tag{2.43}$$

$$k_{i,\,v\;PI} = \omega_z \cdot k_{p,\,v\;PI} \tag{2.44}$$

The bode plots of the voltage open and closed loop transfer functions are shown in Fig. 2.44.

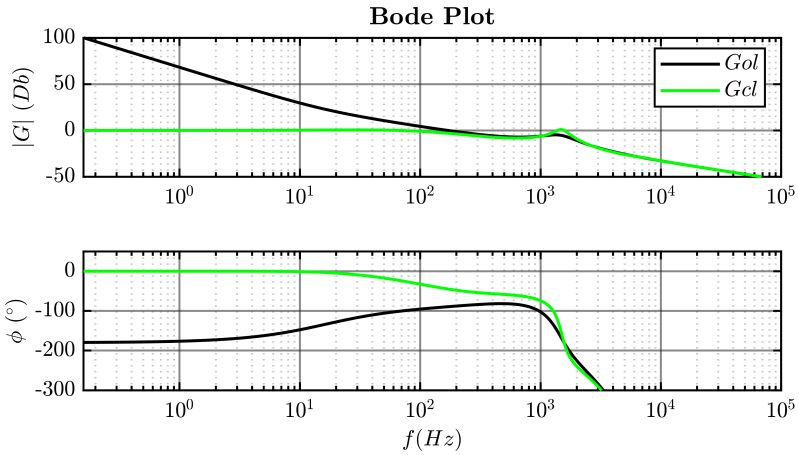


Figure 2.44: Open and Closed Loop Bode Plots, PI controllers used.

After that, the response to the disturbances of the loop is evaluated applying the MATLAB *step* function to the $G_{ii}(s)$ expressed in (2.17). The results is reported in Fig. 2.45.

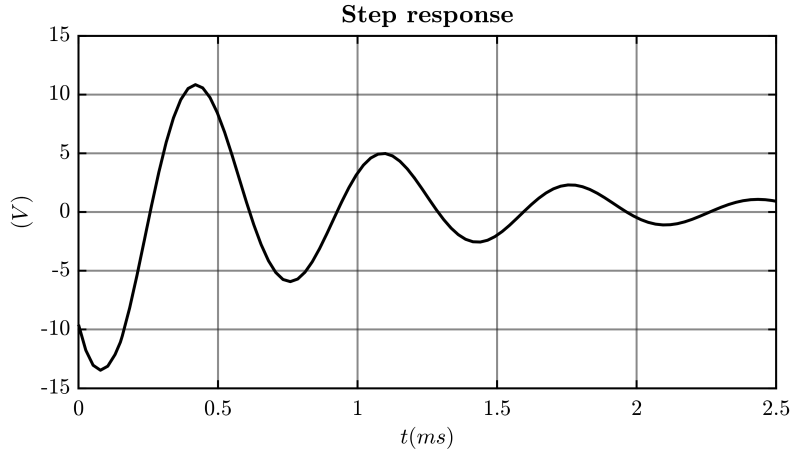


Figure 2.45: Step load response.

The voltage loop control designed in this way results stable and able to reject the disturbances. In Fig. 2.46 the bode plots of both the current loop and the voltage loop involving PI regulators are reported.

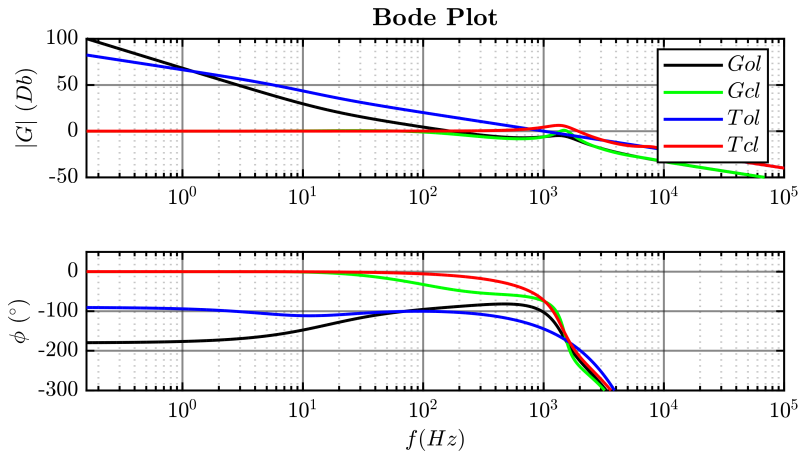


Figure 2.46: Dual loop control open and closed loop bode plots, PI controllers used.

Tuning Of The Voltage PRES Regulator

The external voltage loop of the dual loop control strategy implemented in α, β stationary reference frame involves a PRES regulator.
In order to obtain the same behaviour and dynamic response of the external voltage loop implemented in *d,q*, the proportional and the integral gains of the PRES regulator are equal to ones used for the PI voltage controller. So, $k_{p,\ v\ PRES}$ and $k_{i,\ v\ PRES}$ are expressed by (2.45) and (2.46).

$$k_{p,\ v\ PRES} = \frac{(\omega_r^2 - \omega_{cf}^2) \cdot \omega_{cf}}{\omega_r^2 \cdot (k_{p,\ i\ PRES} \cdot \omega_{cf} + k_{i,\ i\ PRES})} \tag{2.45}$$

$$k_{i,\ v\ PRES} = \omega_z \cdot k_{p,\ v\ PRES} \tag{2.46}$$

The bode plots and the *step* function response are reported in Fig. 2.47 and Fig. 2.48

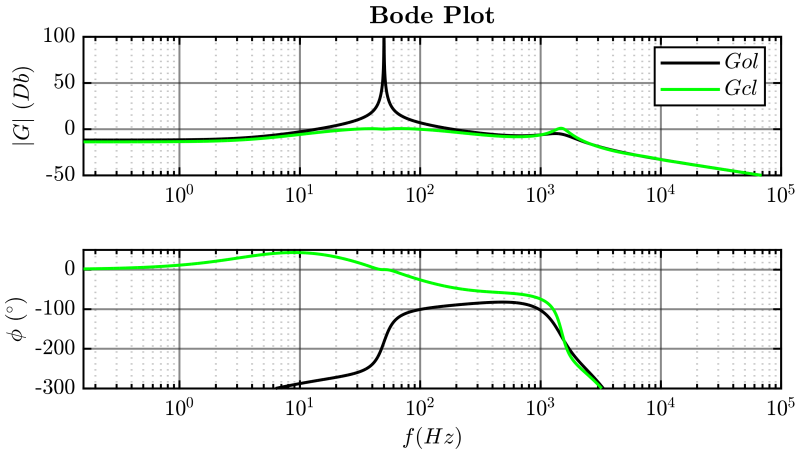


Figure 2.47: Open and Closed Loop Bode Plots, PRES controllers used.

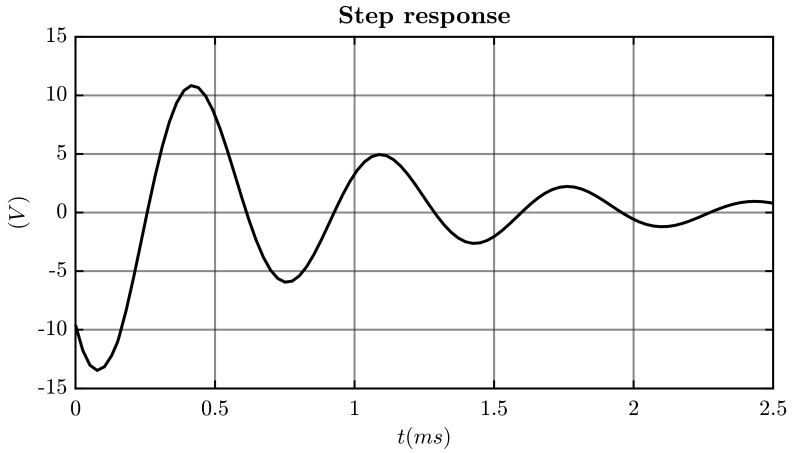


Figure 2.48: Step load response.

The control is stable also in this situation.
In Fig. 2.49 the bode plots of both the current loop and the voltage loop involving PRES regulators are illustrated.

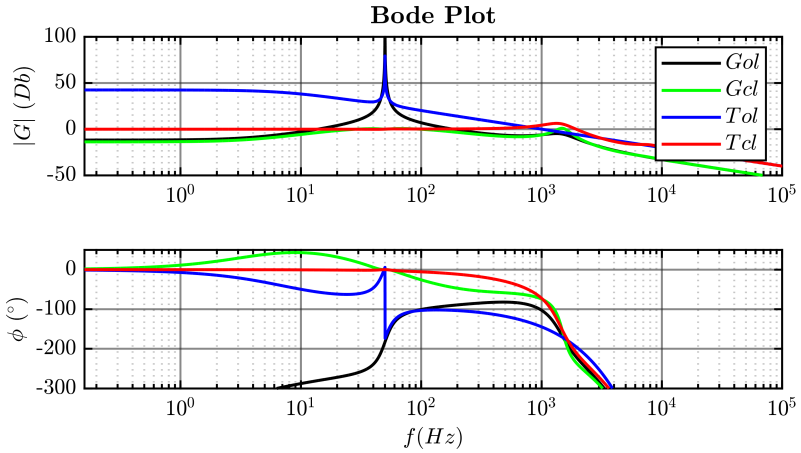


Figure 2.49: Dual loop control open and closed loop bode plots, PI controllers used.

2.3.6 Dual Loop Control: Validation in Simulation PLECS

The two control strategies used to implement the dual loop control were validated in simulation PLECS. To do it, the system in Fig. 2.33 is modelled. As it was anticipated in subsection 2.3.3, the dual loop control was implemented and simulated in:

1. (d,q) synchronous rotating frame, using a PI regulators (Fig. 2.50);

2. (α, β) stationary reference frame, adopting a PRES regulators (Fig. 2.51).

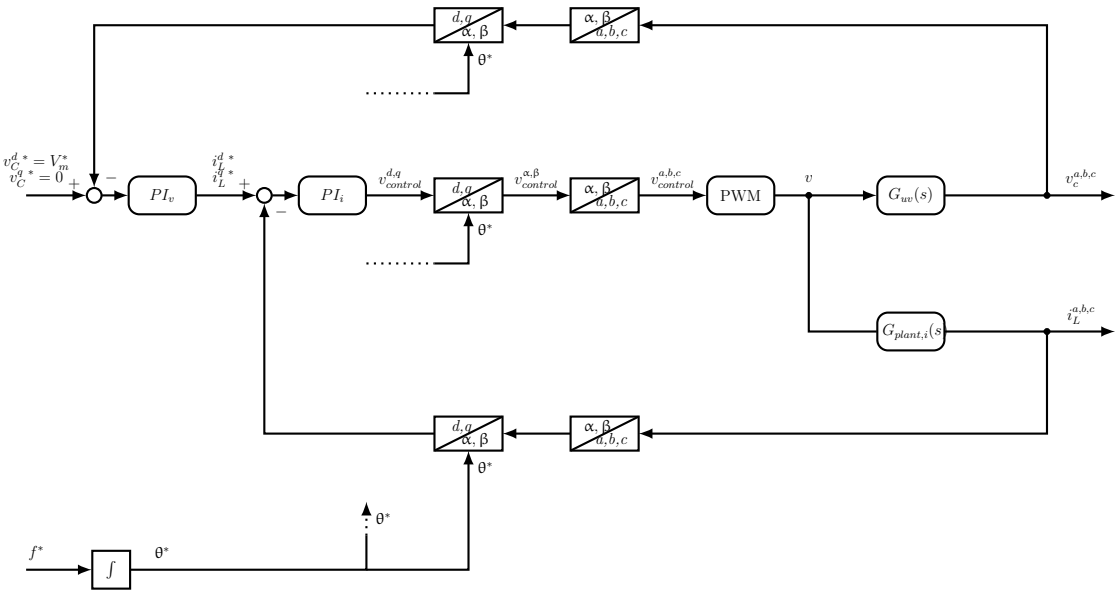


Figure 2.50: Dual Loop Control in (d,q) synchronous rotating frame.

In both the situations, the external reference voltage (V_m^*) is synchronized with the d-axis and the reference q-axis voltage is set to 0. Moreover, the rotation angle used in Park transformation derives from the integration of the set reference frequency. A step load is simulated at $0.5s$ in both cases.The load absorbs the 8% of the converter rated power ((2.31) - (2.32)). In Fig. 2.52 and in Fig. 2.54 are presented the voltage and current waveforms obtained with PLECS dual loop simulation implemented in (d,q) synchronous rotating frame. In particular, the reference and

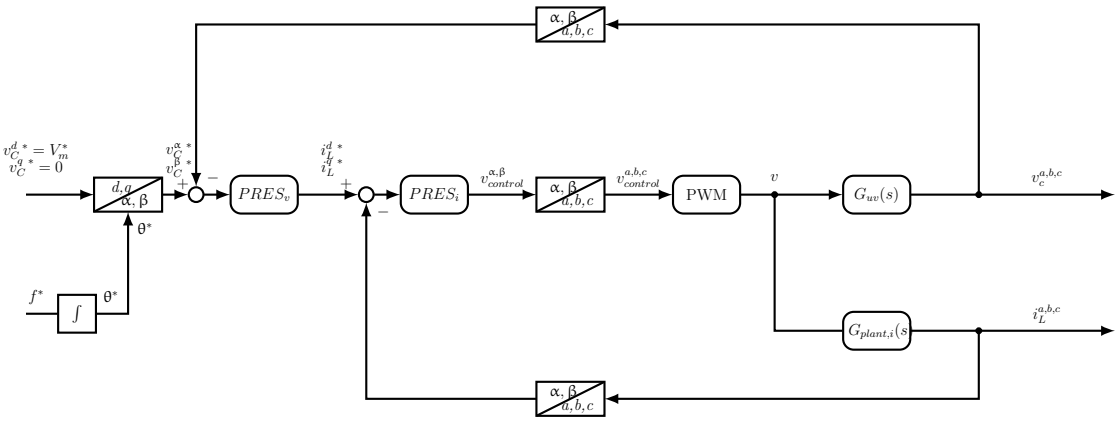


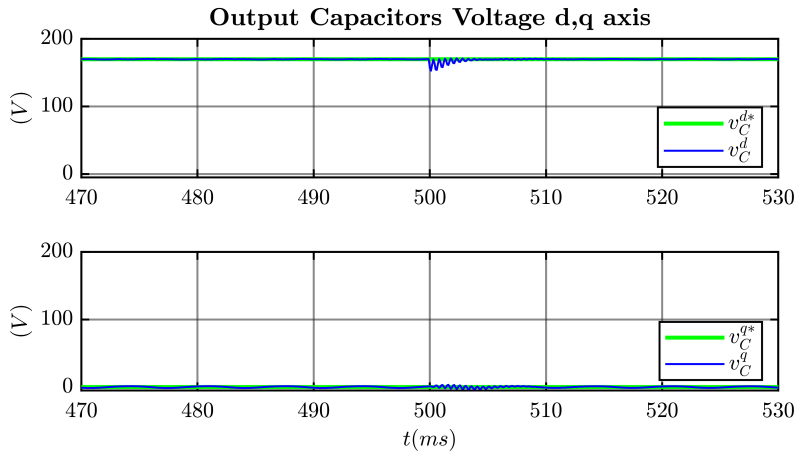
Figure 2.51: Dual Loop Control in (α, β) stationary reference frame.

measured PCC voltage is plotted in Fig. 2.52. Instead, the reference and measured output converter current is reported in Fig. 2.54.
In Fig. 2.53 and in Fig. 2.55 the results obtained implementing the dual loop control in (α, β) stationary reference frame are shown.
Both the control strategies used to implement the dual loop control give good results in simulations. The two ways are compared in terms of THD. In fact, the THD of the three phase current absorbed by the load is calculated in both the situation and the results are compared in order to determine which type of implementation gives better results. The synthesis of this analysis is reported in (2.2).

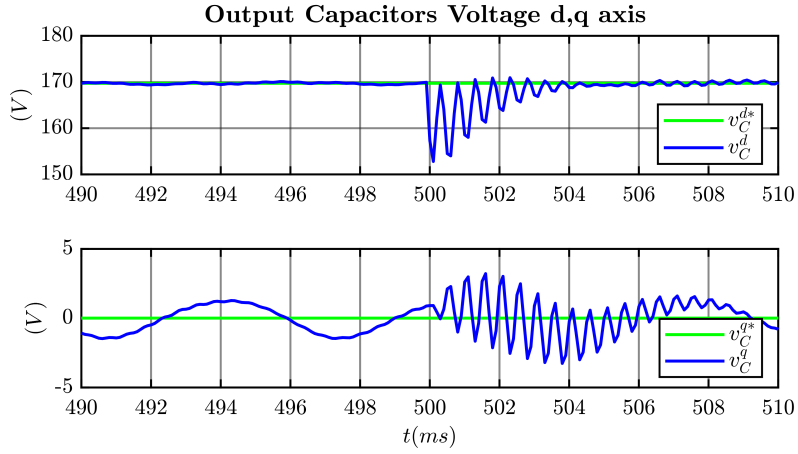
Dual Loop Control - THD		
PI	**PRES**	**Percentage Difference**
$\approx 3.461\%$	$\approx 3.468\%$	$\approx 0.715\%$

Table 2.2: i_{PCC} THD Dual Loop Control.

It is clear that both control strategies work well and with almost the same performances. Cause the THD is less than 5%, according to the normative, the signal is considered sinusoidal.

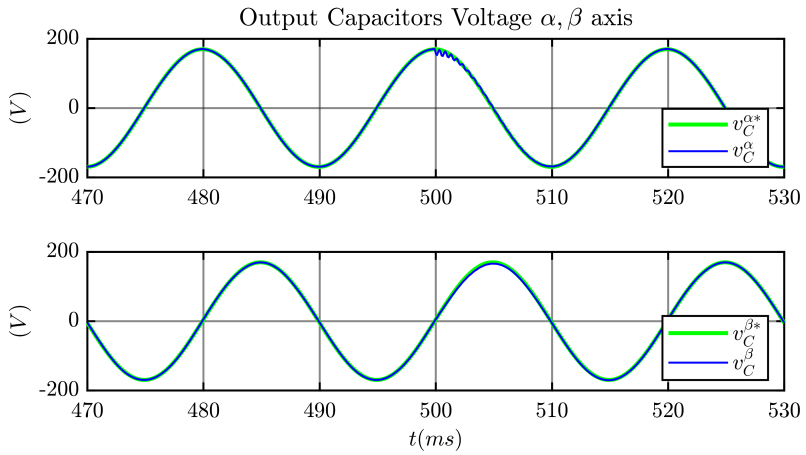


(a) Reference and measured voltage.

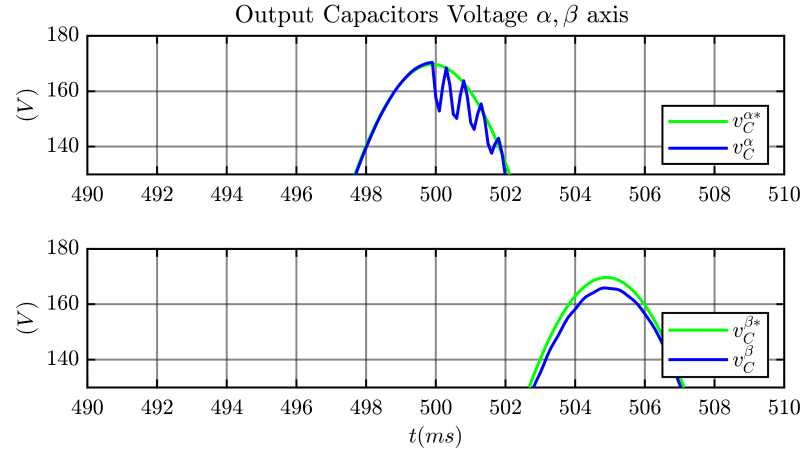


(b) Reference and measured voltage waveforms magnification at the load step.

Figure 2.52: Dual loop control implemented in d,q synchronous rotating frame using PI controllers: reference and measured voltage waveforms, PLECS simulation results.



(a) Reference and measured voltage.



(b) Reference and measured voltage waveforms magnification at the load step.

Figure 2.53: Dual loop control implemented in (α, β) stationary reference frame using PRES controllers: reference and measured voltage waveforms, PLECS simulation results.

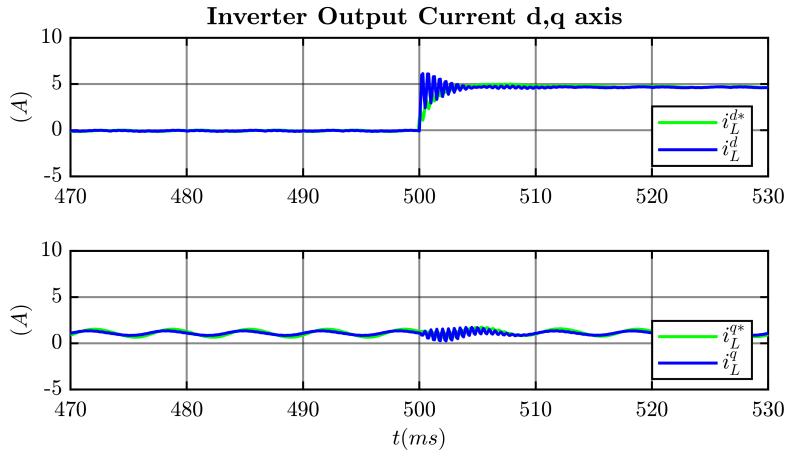


Figure 2.54: Dual loop control implemented in d,q synchronous rotating frame using PI controllers: reference and measured current waveforms, PLECS simulation results.

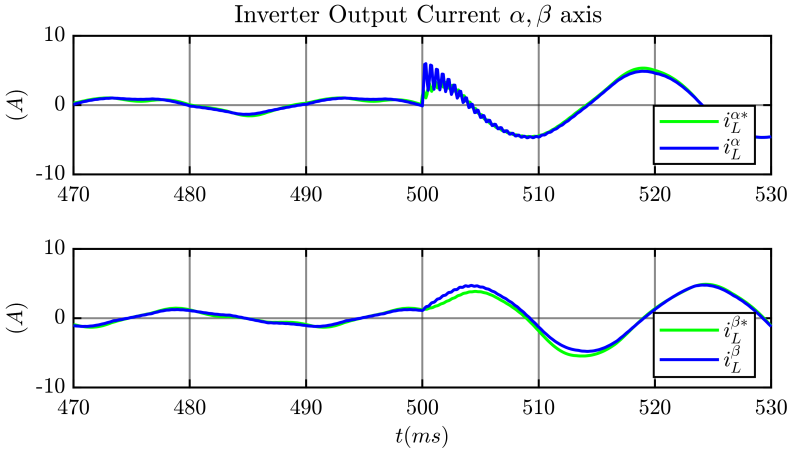


Figure 2.55: Dual loop control implemented in (α, β) stationary reference frame using PRES controllers: reference and measured current waveforms, PLECS simulation results.

2.4 Comparison of Output LC Filter Damping Topologies

In this section, the analysis of different output LC filter damping topologies will be presented. The study is conducted considering the DC/AC converter controlled as grid-forming using the Single Voltage Loop strategy (Fig. 2.22). The G_{UV} transfer function ((2.16)) expresses the relationship between the output converter voltage (v) and the output filter voltage (v_c). Fig. 2.23 shows the Bode plot of G_{uv}. At the resonance frequency the G_{UV} has a high peak in the graph of the module Bode plot. This peak makes the control implemented on the system unstable because during the calibration of the regulators it does not allow to satisfy the requests of positive gain margin and phase margin. For this reason, it is necessary to introduce a damping element in order to suppress the resonant peak. Three different passive damping topologies are analysed:

- **CASE A:** a damping resistor (R_D) in series with the filter capacitor (C_F);
- **CASE B:** a damping circuit composed by a capacitor (C_D) in series with resistor (R_D) placed in parallel with the filter capacitor (C_F);
- **CASE C:** a damping resistor (R_D) in parallel with the filter capacitor (C_F);

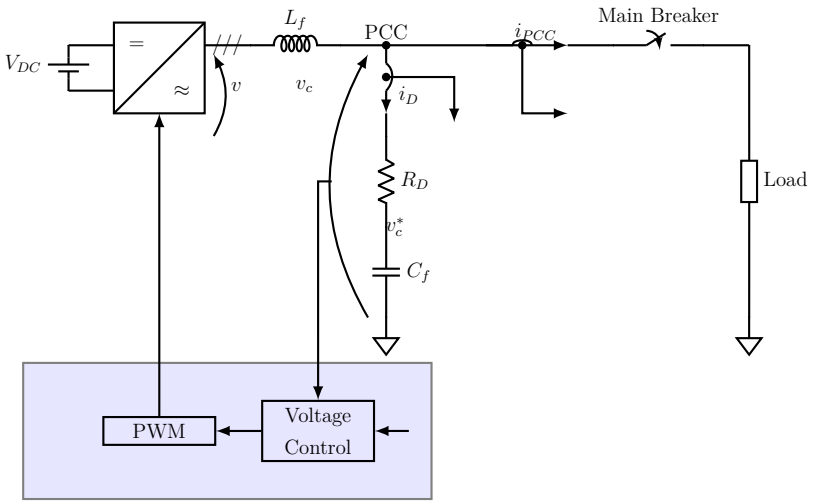


Figure 2.56: **CASE A:** L-C output filter with R_d damping resistor in series.

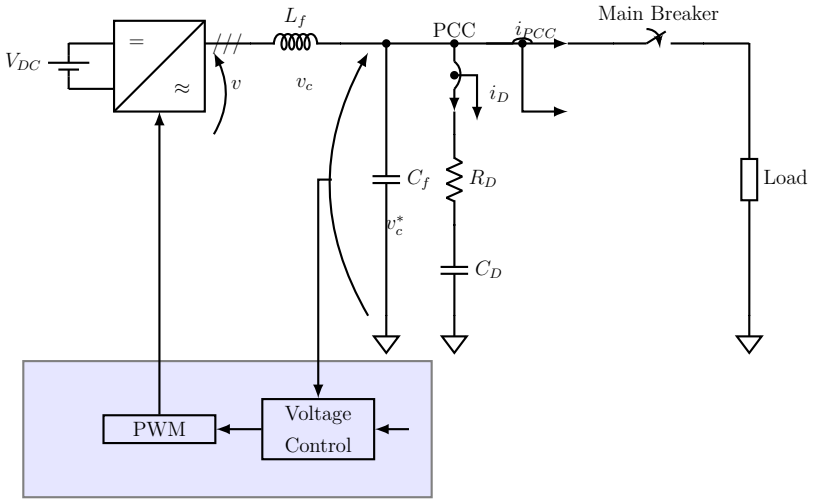


Figure 2.57: **CASE B:** L-C output filter with R_D and C_D damping resistor in parallel.

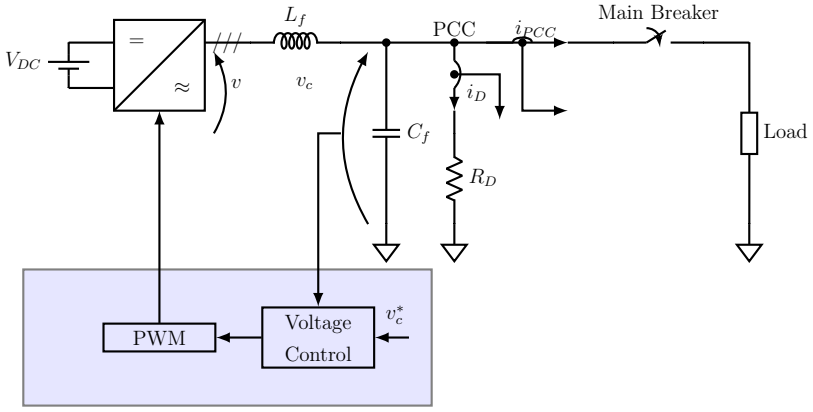


Figure 2.58: **CASE C:** L-C output filter with R_d damping resistor in parallel.

The analysis began by taking case A as a reference. However, in order to obtain less active power losses due to the damping resistor, the structures of case B and C were also analysed. In fact, the study involves the following step:

1. Design R_D in case A in order to suppress the resonant peak and build a stable control;

2. Design the damping circuit parameters of the other two solutions trying have the same peak attenuation assumed in case A;

3. Optimize the parameters considering the active power losses obtained in the damping circuit;

4. The three designed solutions are finally compared in terms of THD, harmonic attenuation, active power losses in the damping circuit and voltage drop recorded during load steps.

So, in Fig. 2.59 are shown the G_{UV} bode plots at different values of R_D chosen considering topology A. Moreover, the open and closed loop are calculated using (2.21) and (2.22) in order to understand the effect of the choice of R_D on their Bode plot and therefore the stability of the system. The open and closed loop Bode plots are presented in Fig. 2.60.

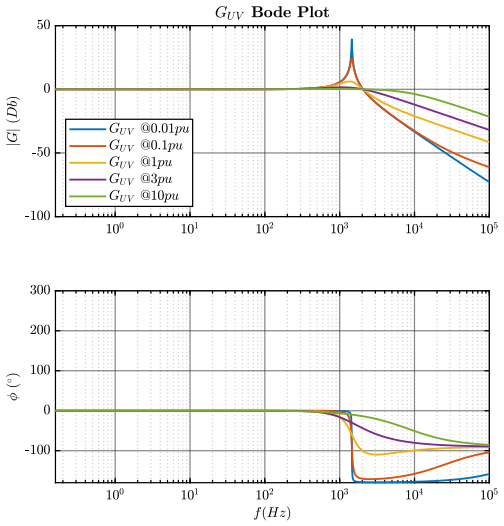
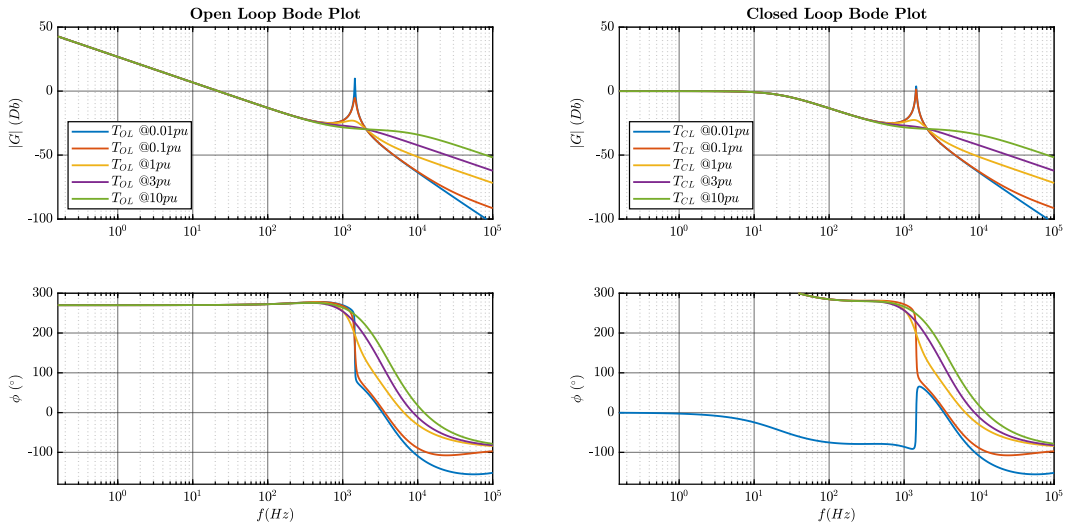


Figure 2.59: G_{UV} transfer function bode plots obtained using the case A damping circuit at different values of R_D chosen.



(a) Open loop bode plot.

(b) Closed loop bode plot.

Figure 2.60: Open and closed loop bode plots obtained using the case A damping circuit at different values of R_D chosen.

It is visible that the damping of the resonance peak increases with R_D. The resonance peak introduces an high phase lag that make unstable the control. However, if R_D is chosen too high, the high frequency attenuation decreases. So, for the application a value of $R_D = 1pu$ is chosen.
In case C, the parallel resistor R_D is chosen equal to $3.5pu$ to obtain the same peak attenuation of the case A.
Moving on case B, in order to keep the same value of the resonant frequency of the case A, this relationship must be satisfied:

$$C_{F,A} = C_D + C_{F,B} \tag{2.47}$$

So, the formula used to evaluate the resonant frequency in case A is:

$$w_{R,B} = \frac{1}{\sqrt{L_F \cdot (C_D + C_{F,B})}} \tag{2.48}$$

The damping resistor R_D has the role to attenuate the resonance peak. However, it is discovered that the resonance peak is more damped the higher is R_D, but the resonant frequency increase according to R_D value as well. In fact, in Fig. 2.61 are illustrated the G_{UV} bode plots in the three cases of damping circuit choosing:

- $R_{D,A} = 10pu$;
- $R_{D,B} = 1pu$;
- $R_{D,C} = 3.5pu$;

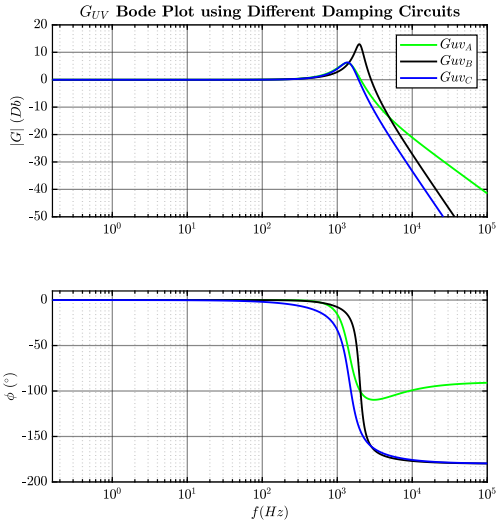


Figure 2.61: G_{UV} transfer function bode plots obtained for the three different damping circuit cases.

Fig. 2.61 shows the difference between the resonant frequency of the case B and the one of the case A and C. However, the resonant peak attenuation is almost the same in the three configuration.

Conducting an analysis on the power losses obtained in the case B with respect to the one obtained in the case A, the choice $R_{D,B} = 10pu$ are not accepted. In fact, the advantage of using the damping typology of case B compared to the case A is to obtain lower active power losses on the damping circuit since the current is divided between the two branches in parallel. However, in this situation the power losses recorded in the case B using a resistor $R_{D,B} = 10pu$ are approximately 250% higher than those recorded in case B.

For this reason, $R_{D,B}$ is chosen equal to $1pu$. However, a less damped peak is accepted, but its value does not influence the system stability. In this case, the power losses are reduced and become less than the reference case. Fig. 2.62 and

Fig. **??** the Bode plots of the G_{UV} and of the open and closed loop transfer functions adopting the following value of the damping circuit parameter in the three cases:

$$C_{D,B} = C_{F,B} = \frac{C_{F,A}}{2} \tag{2.49}$$

$$R_{D,B} = 1pu \tag{2.50}$$

$$R_{D,A} = 1pu \tag{2.51}$$

$$R_{D,C} = 3.5pu \tag{2.52}$$

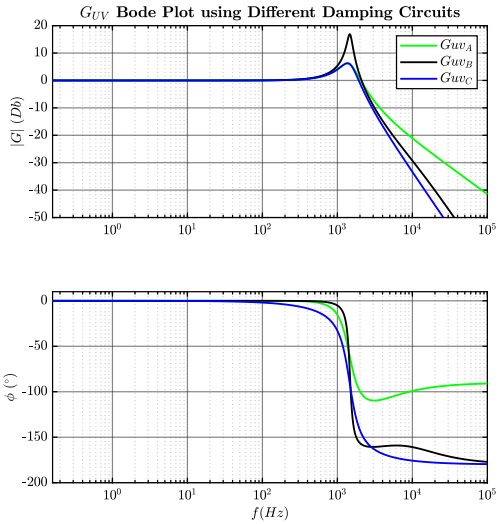
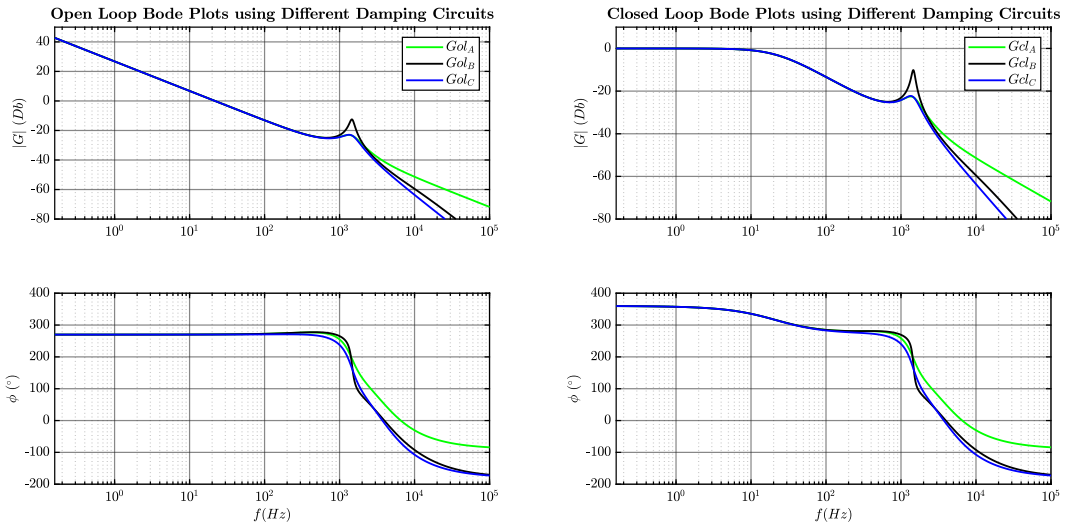


Figure 2.62: G_{UV} transfer function bode plots obtained for the three different damping circuit cases.



(a) Open loop bode plot.

(b) Closed loop bode plot.

Figure 2.63: Open and closed loop transfer functions bode plots obtained for the three different damping circuit cases.

As shown in [9] the active power losses in the damping circuits are evaluated at the fundamental frequency $f_0 = 50Hz$ and at the switching frequency $f_{sw} = 10kHz$ following these steps:

1. First the relationship $\frac{i_D}{v_C}(s)$ between the current flowing in the damping circuit (i_D) and the output filter voltage (v_C) is retrieved;
2. Then, the relationship $\frac{v_C}{v}(s)$ between the output filter voltage (v_C) and the output converter voltage (v) is calculated;
3. Then, (2.53) and (2.54) are applied to calculate the active power losses. It is important to note that $v_C(s \to j2\pi f_0) = 1pu$ and $v_C(s \to j2\pi f_{sw}) = func(v)$.

$$P_{d,\ 50Hz} = Real(V_C \cdot I_D^*)_{50\ Hz} \tag{2.53}$$

$$P_{d,\ 10kHz} = Real(V_C \cdot I_D^*)_{10\ kHz} \tag{2.54}$$

4. Finally, the total losses are evaluated:

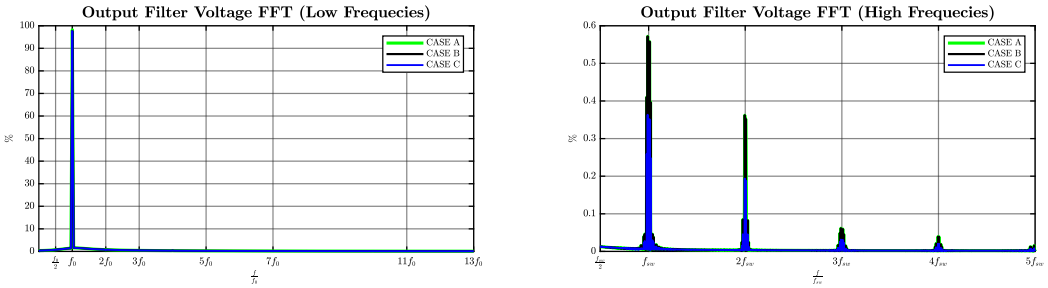
$$P_{d,\ tot} = P_{d,\ 50Hz} + P_{d,\ 10kHz} \tag{2.55}$$

The results obtained are shown in the table below:

Active Power Losses in the damping circuit			
	CASE A	**CASE B**	**CASE C**
$P_{d,\ 50Hz}$	$1.32 \cdot 10^{-2}\%$	$3.30 \cdot 10^{-3}\%$	9.52%
$P_{d,\ 10kHz}$	$2.40 \cdot 10^{-1}\%$	$3.16 \cdot 10^{-2}\%$	$4.41 \cdot 10^{-3}\%$
$P_{d,\ tot}$	$2.60 \cdot 10^{-1}\%$	$3.49 \cdot 10^{-2}\%$	$9.53 \cdot 10\%$

Table 2.3: Active power losses in damping circuit for the three configuration considered.

As expected, case B has less active power losses in the damping circuit compared to the other two configurations studied.
The three damping topologies shown in Fig. **??** were simulated in PLECS. The phase voltage of the output filter capacitor was measured and using the *Fast Fourier transformation* (FFT) algorithm it was possible to represent the measured signal in frequency domain. The results obtained were compared and are shown in Fig. 2.64.



(a) Phase Voltage FFT at low frequencies. (b) Phase Voltage FFT at high frequency.

Figure 2.64: Comparison of the phase voltage FFT obtained in the three simulations involving the different damping circuits.

Although in all 3 cases there are no significant low frequency harmonics, circuits A and B have the same percentage of high frequency harmonics while circuit C has a lower percentage than them.
However, the difference between the three circuits is not relevant. To clarify the

situation, the output filter capacitors phase voltage THD and the PCC current THD were calculated. The results are reported in (2.4).

Total Harmonics Distortion (THD)		
	$THD_{\ vc}$	$THD_{\ i\ PCC}$
CASE A	0.907%	2.97%
CASE B	0.907%	0.62%
CASE C	0.907%	0.97%

Table 2.4: THD.

Finally, in PLECS a step load was simulated and the voltage drops recorded at the loading time were compared in the three configuration. This load absorbs 25% of the rated power of the converter.
This analysis was made to evaluate the influence of the different types of damping circuit on control in transient behaviour.
In order to judge the results, the *IEC 62040-3 standard* for uninterruptible power source (UPS) has been taken as a reference.
An (UPS) is an electrical apparatus that provides emergency power to a load when the input power source or mains power fails. The converter under analysis is controlled as grid-forming and must operate in a microgrid which is electrically disconnected from the main grid. Therefore, high quality in dynamics is required because the converter is controlled as grid-forming and has the task of setting the reference voltage and frequency for all the converters connected to the PCC. In Fig. 2.65 are reported the voltage limits accepted during a transient due to a step load connection. This curve is extracted from the *IEC 62040-3 standard*.

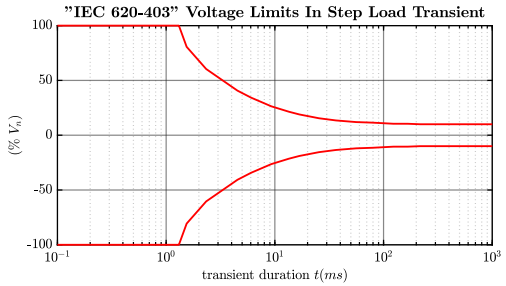


Figure 2.65: *IEC 62040-3 standard* transient voltage limitation.

In Fig. 2.66 are presented the voltage drops obtained in transient during the simulations compared to the limits imposed by *IEC 62040-3 standard*. This graph is obtained computing the PCC measured voltage recorded after being normalized to the reference voltage peak in PLECS simulations and the limits required by the standard.

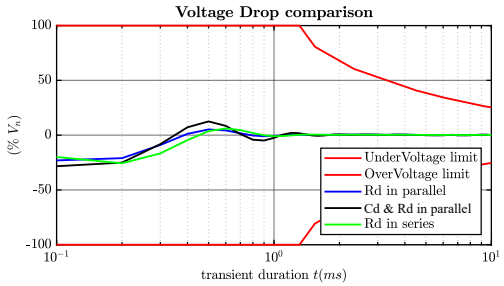
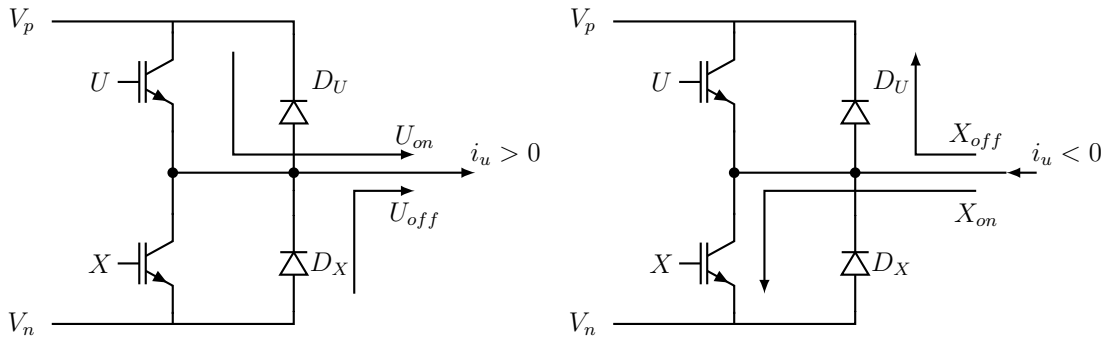


Figure 2.66: Voltage drops obtained in transient compared to the limits imposed by *IEC 62040-3 standard*.

In conclusion, the damping circuit in case B is the one with the least losses but also the lowest voltage peak recorded at the loading time. Instead, the circuit of case C has the most significant losses, but the best transient behaviour. To find a compromise the damping circuit of case A was adopted.

2.5 Effect and Compensation of Dead-Times

In this section, the effect on the output converter voltage due to the introduction of dead-times in the system are analysed and it is explained how to compensate them. Fig. 2.67 shows the leg of one phase of the PWM inverter.



(a) Case $i_u > 0$.

(b) Case $i_u < 0$.

Figure 2.67: One phase leg of a PWM inverter ([5]).

If the two switches of the same leg conduct in the same moment, a short circuit on DC-link occurs. Generally, in inverter modulated with PWM technique a time delay must be inserted in switching signals to prevent this short circuit on the DC-link. This time delay ensures a safe operation for the inverter, but it introduces a distortion on the output voltage of the converter. This effect is due to the fact that during these intervals of time there is a loss of control because both switches cease to conduct. However, since the inductive characteristic of the output converter circuit, during the dead-time intervals, the current flows through freewheeling diodes according to the current direction. So, in the delay time ranges, the direction of the current flow determines the output voltage. The situation is well explained in Fig. 2.68 where is illustrated that the voltage deviation due to the time delay opposes the current flow direction.

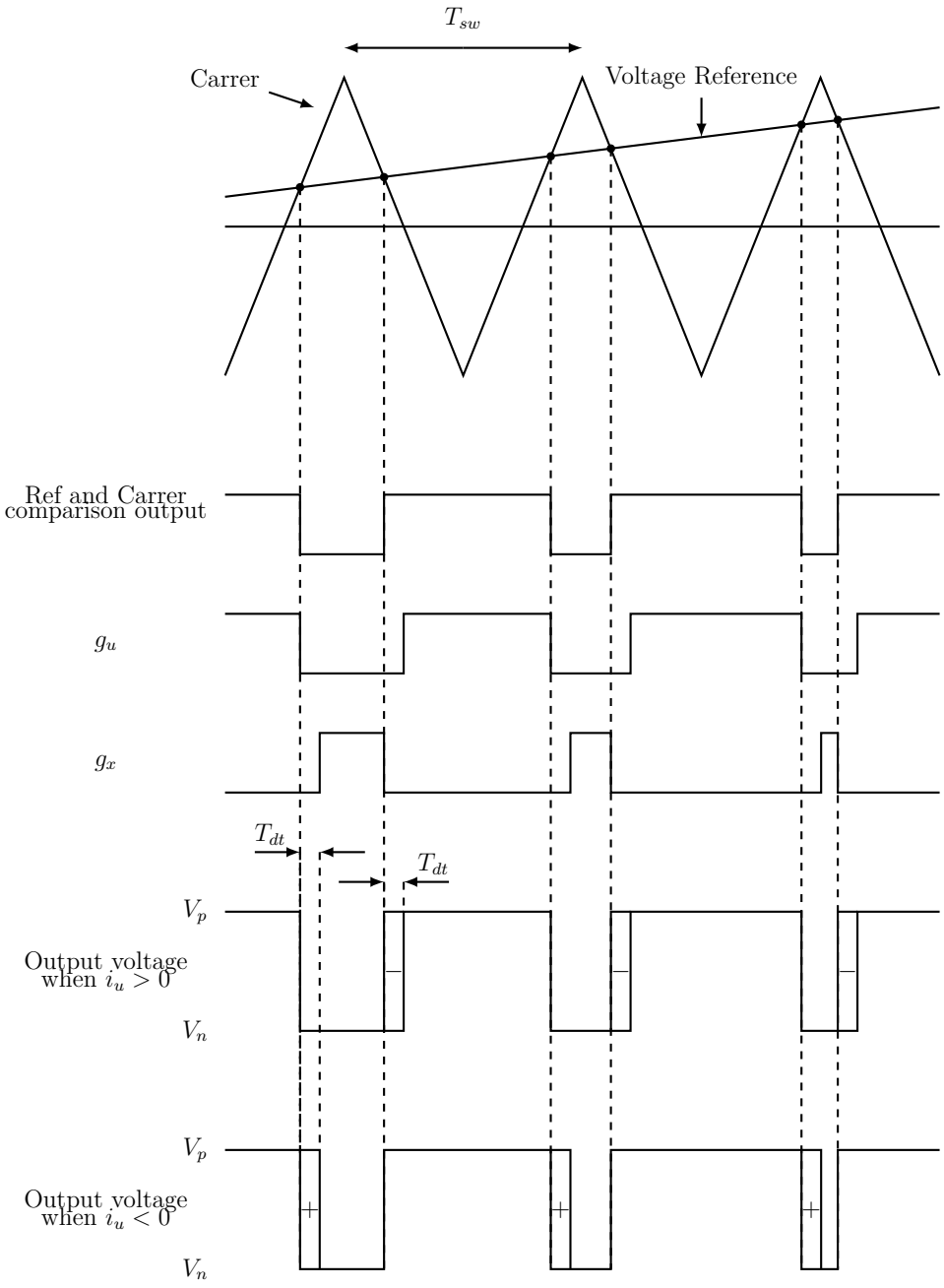


Figure 2.68: Dead-time: Relationship between the output converter current and the output voltage. [5]

The switching signals for the PWM modulator of the system under study will be modified to ensure that no short circuit could be generated on DC-link. The grid-forming converter will be controlled using the dual loop control strategy. The effect

will be evaluated calculating the THD for the output voltage signal and comparing this result with the one obtained in the simulation without implementation of dead-time (ideal case). After that, an appropriate dead-time compensation algorithm will be provided to try to improve the results by approaching the ideal situation.
In the ideal case, the switches on the same leg of the inverter are commanded in the opposite way: when the first receives command 1, at the same time, the other receives command 0 (Fig. 2.69).

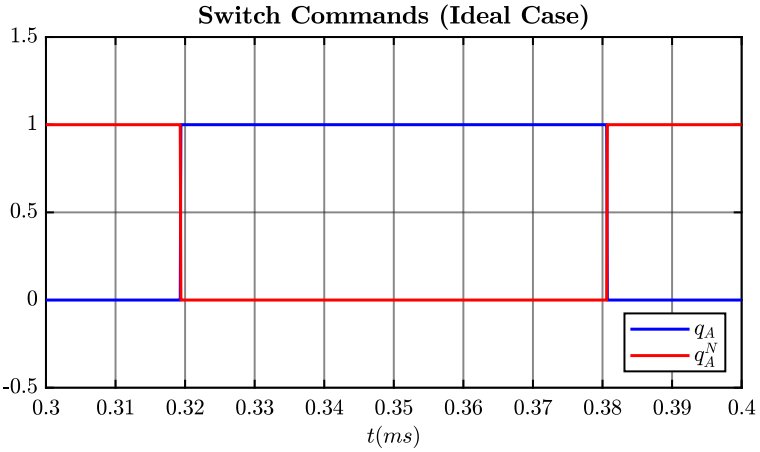


Figure 2.69: Switching command of one of the inverter leg in ideal case.

In this case, the change of the switch state is supposed instantaneous, but in real case this is not. In fact, to avoid generation of short-circuit on the DC-link, the command 1 for the switches is delayed of the dead-time (Fig. 2.70).

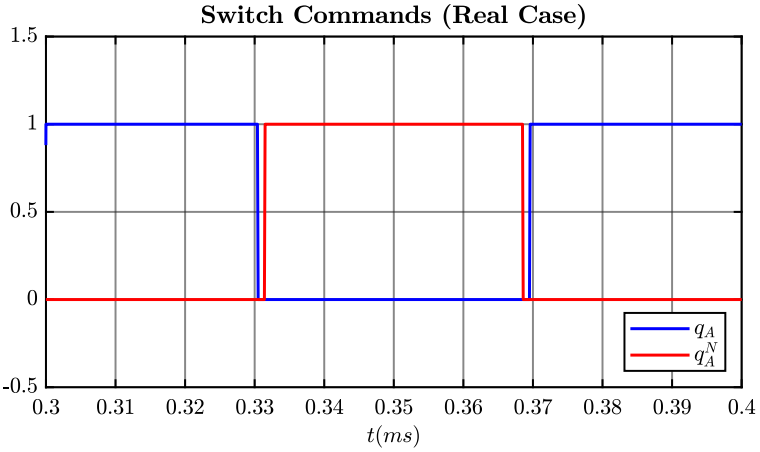


Figure 2.70: Switching command of one of the inverter leg in real case with introduction of dead-time intervals.

The voltage THD at the PCC is calculated in ideal and real case simulations by implementing grid-forming control using the dual loop control strategy. A dead-time of $1\mu s$ is considered for the system. The results obtained are summarized in the Table 2.5:

$\mathbf{v_{PCC}}$ **THD**	
Ideal case	**Real case**
No Dead-time added	**Dead-time added**
$\approx 3.74\%$	$\approx 4.35\%$

Table 2.5: v_{PCC} ideal and real case.

According to [5], the effect that the introduction of dead-time causes on the output voltage is a deviation on the voltage waveforms (ΔV), which can be represent by(2.56).

$$\Delta V = n_p \cdot (T_D \cdot V_{DC}) \cdot f_0 = f_{sw} \cdot T_D \cdot V_{DC} \tag{2.56}$$

where

T_D is the length in seconds of a dead-time interval;

V_{DC} is the amplitude of the dead-time interval;

f_{sw} is the switching frequency;

f_0 is the operating frequency;

$n_p = \frac{f_{sw}}{f_0}$ is the number of pulses per period caused by the dead-time. It depends on the switching frequency of the power converter;

The distortion of the output voltage caused by the dead-time leads to the increase of the THD value in confront of the one obtained in the ideal case (Table 2.5). A dead-time compensation algorithm is implemented to correct the effect caused by

delaying the switching commands. This algorithm is proposed in [5] . It consists on these two steps:

1. the output converter current polarity is detected. To avoid false detections, an hysteresis threshold is adopted;

2. ΔV quantity is added or removed to the $v_{control}$ signals before they are sent to the PWM block, according to the current polarity. In this way the effect of dead time, which is opposite to current polarity, is compensated.

A block scheme of the dead-time compensation algorithm used is shown in Fig. 2.71.

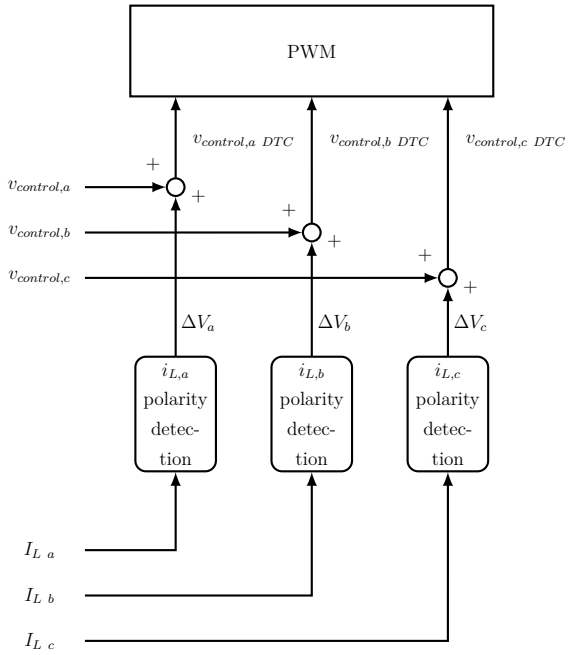


Figure 2.71: Dead Time Compensation Algorithm.

The C code is presented in Appendix in section 7.1. The determination of the current polarity is at the basis of this compensation algorithm, therefore it is necessary to avoid that the current ripple provokes mistakes. To do that for the hysteresis control is chosen a range of $\pm 3\% I_n$ (I_n: nominal current).
The PCC voltage THD is calculated simulating in PLECS the real case provided with the compensation algorithm. The result obtained is 3.9%.

Finally, comparing this value with the ones in Table 2.5, it is possible to say that the compensation algorithm works well and decreases the voltage THD value, bringing it very close to the one obtained in the ideal case.

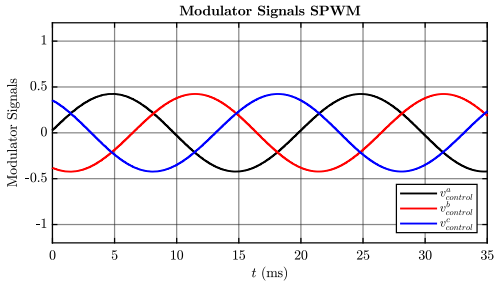
2.6 Modulation Techniques Comparison

Balanced Envelope Modulation PWM (PWM BEM) technique is used to implement the dual loop control strategy for the power converter under study. The performance of this technique is compared to the simulation in which the dual loop control is implemented using the Discontinuous Pulse Width Modulation (DPWM) technique. The PWM BEM and the DPWM techniques are two evolution of the classical Sinusoidal PWM (SPWM) in which modulator signals are sinusoidal waveforms.
Although with the SPWM technique it is possible to obtain a sinusoidal reference signal which will then be compared with the high frequency carrier to generate the leg commands, it does not allow to maximize the exploitation of DC-link voltage and also it has significant switching losses ([7]). These are two disadvantages of the classic SPWM, therefore other modulation techniques have been proposed with the aim of:

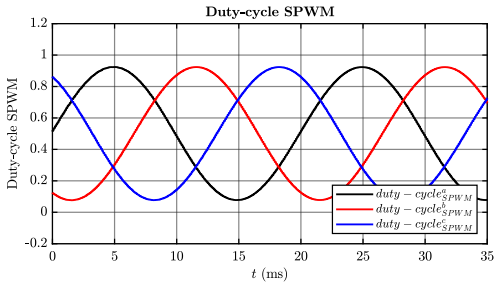
- extending the linear operating region to permit a better exploitation of the DC-link voltage;

- reducing the switching losses to improve the efficiency of the power converter.

Both PWM BEM and DPWM techniques improve the linear working region of 15%, that means the DC-link is exploited better. Moreover, involving DPWM technique the switching power losses are reduced of $\frac{1}{3}$ respected to the value obtained using SPWM or PWM BEM techniques. Fig. 2.72 shows the modulating signals and the duty cycle waveforms characterizing SPWM technique.



(a) Modulating signals.

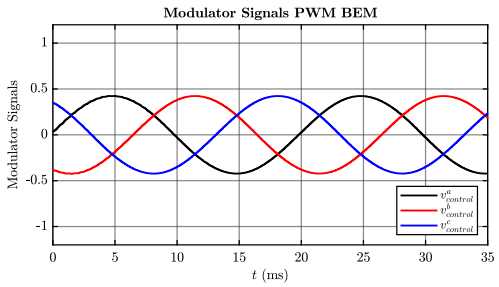


(b) Duty cycle waveforms.

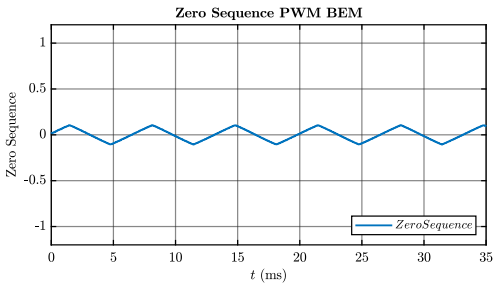
Figure 2.72: SPWM technique modulating signals and duty cycle waveforms.

As it is well explained in [31], PWM BEM is obtained adding a common-mode third-harmonic signal into the reference waveforms of each phase. This third-harmonic component does not influence the line-to-line fundamental output voltage because the common mode signal is equally introduced in all the three phases. The peak of the resulting modulating signal is lower, but has an higher RMS value respect to the one obtained for the SPWM. Figure 2.73 shows the modulating signals and the duty cycle waveforms for PWM BEM technique.

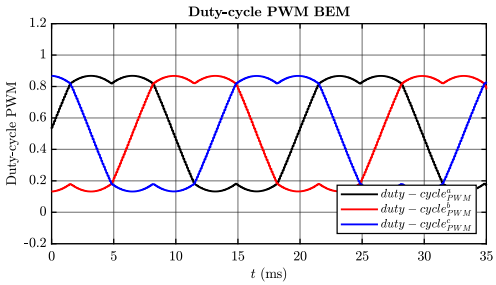
According to [31], there are different type of DPWM and the one involved for this application is the 60° discontinuous simmetrical modulation, called "DPWM1". In this technique, each phase leg is clamped to $\pm V_{DC}$ for 60° symmetrically around the fundamental voltage's positive and negative peaks. So, the line-to line voltage results symmetric. This is not for the other DPWM implementation strategy. The modulating signals and the duty cycle generated by DPWM1 are shown in Fig. 2.74.



(a) Modulating signals.

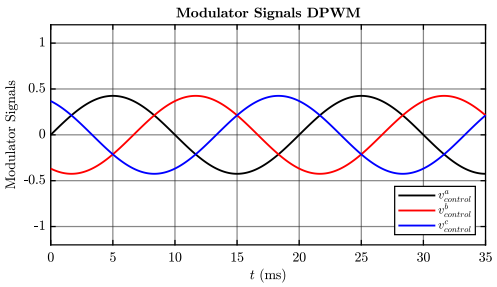


(b) Zero sequence.

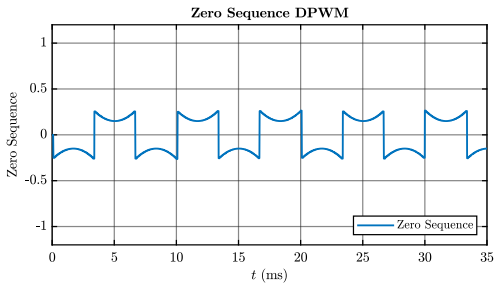


(c) Duty cycle waveforms.

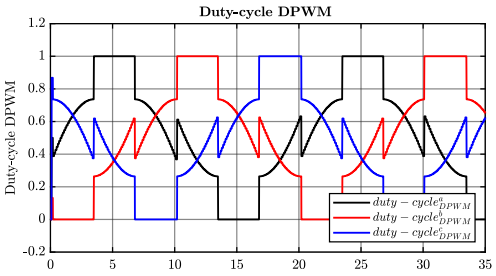
Figure 2.73: PWM BEM technique modulating signals and duty cycle waveforms.



(a) Modulating signals.



(b) Zero sequence.



(c) Duty cycle waveforms.

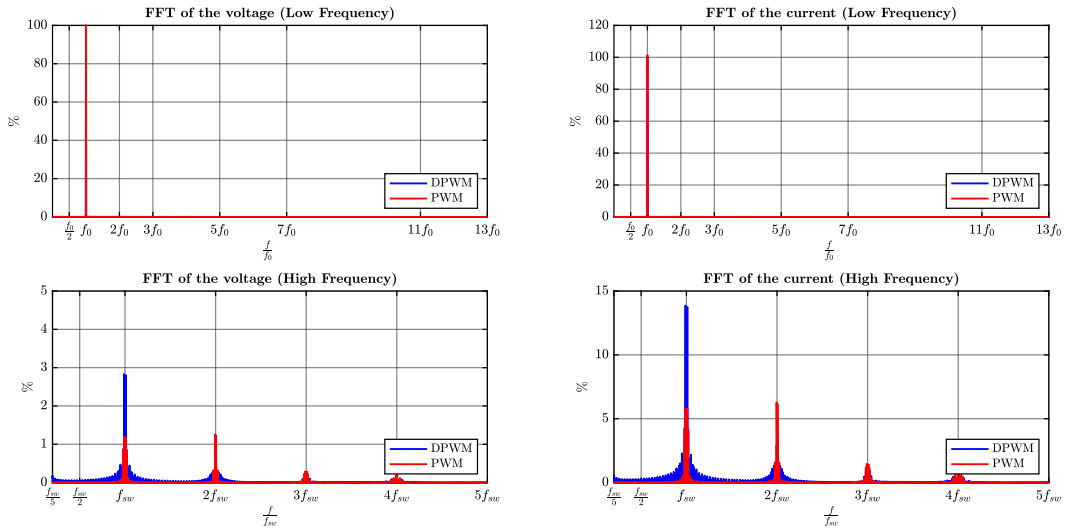
Figure 2.74: DPWM technique modulating signals and duty cycle waveforms.

Thanks to the time regions where the phase legs are clamped to $\pm V_{DC}$, the DPWM mode has the advantages of eliminating one switching transition in each half carrier interval, reducing the switching losses in comparison of the the other modulation techniques.

As written before, both PWM BEM and DPWM techniques are implemented in two different PLECS simulations which simulate a grid-forming power converter controlled using the dual loop control strategy. Then, the two modulation techniques are compared evaluating the harmonic distortion caused in the output converter voltage and current and calculating the power dissipated on the converter, doing a thermal analysis.

The C code used to implement PWM BEM and DPWM technique are reported in the Appendix in section 7.2 and section 7.3.

A Fourier analysis is conducted applying the Fast Fourier transformation (FFT) algorithm at the voltage and current signal. The results obtained are shown in Fig. 2.75.



(a) Phase Voltage FFT.

(b) Output converter current FFT.

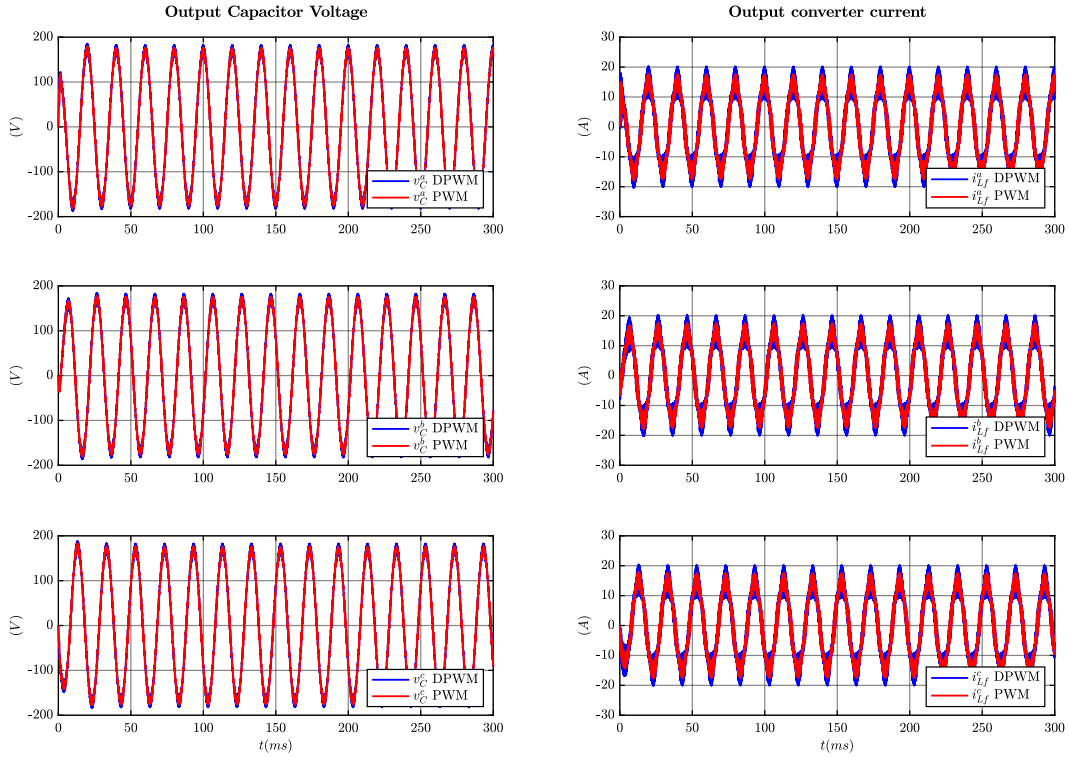
Figure 2.75: Comparison of the phase voltage and output converter current FFT obtained in the two PLECS simulations.

The harmonic distortion is evaluated calculating the THD for the current and voltage signals. A resistive load that absorbs an active power of $25\%S_n$ is considered. Table 2.6 summarize the results obtained.

Total Harmonics Distortion (THD)		
	$\mathbf{v_C}$ **THD**	$\mathbf{i_{Lf}}$ **THD**
PWM BEM	3.7%	14.6%
DPWM	6.0%	22.7%

Table 2.6: Phase voltage and output converter current THD.

In Fig. 2.76 are shown the three phase voltage measured at PCC and the three phase output current of the converter. The results obtained in the two PLECS simulations using the two described modulation techniques are illustrated.



(a) Three Phase Voltage at PCC. (b) Three phase output converter current.

Figure 2.76: Comparison of three phase voltage at PCC and output converter current measured during the two simulations using PWM BEM and DPWM technique.

To complete the study, a thermal analysis is conducted using PLECS. In fact, a thermal circuit is simulated inserting a block, which emulated a heat-sink, on the power converter switches. After that, a thermal resistance is connected between the heat-sink and the ambient. The scheme built in PLECS is shown in Fig. 2.77. The thermal environment is drawn in blue.

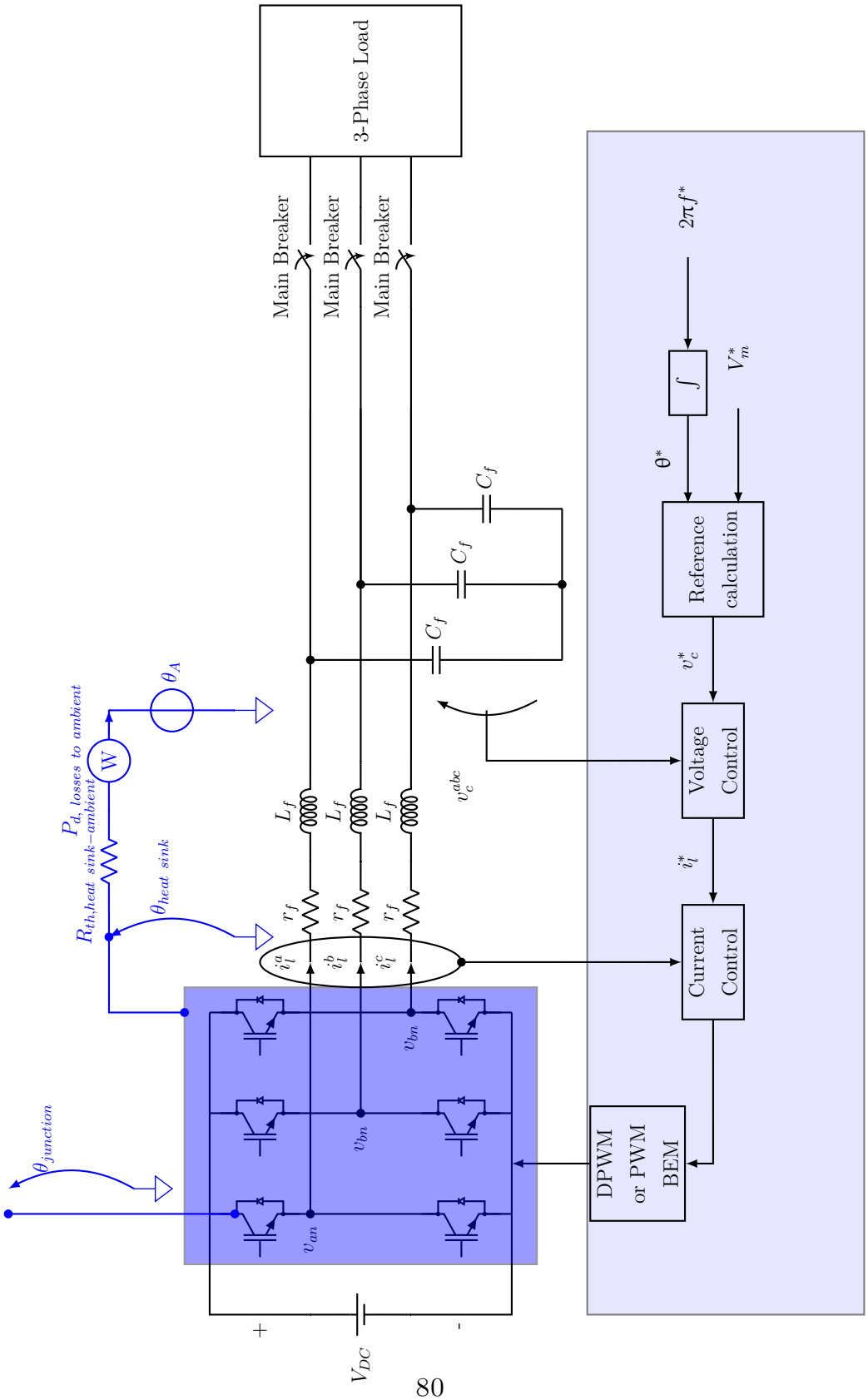

3-Phase Load
Main Breaker
Main Breaker
Main Breaker
C_f
C_f
C_f
L_f
L_f
L_f
r_f
r_f
r_f
i_l^a
i_l^b
i_l^c
v_c^{abc}
θ_A
$P_{d,\ losses\ to\ ambient}$
W
$R_{th,heat\ sink-ambient}$
$\theta_{heat\ sink}$
$\theta_{junction}$
v_{an}
v_{bn}
v_{bn}
+
-
V_{DC}
$2\pi f^*$
$\int$
θ^*
V_m^*
Reference calculation
v_c^*
Voltage Control
i_l^*
Current Control
DPWM or PWM BEM

Figure 2.77: Model of the thermal circuit used in PLECS simulations.

The *FS150R07N3E4 Infineon* three phase two level IGBT inverter module is chosen for the application and its datasheet is reported in [12]. The IGBT and the free-wheeling Diode have been thermally characterized using the "Thermal Description" tool of PLECS. In fact, it is possible to report the thermal characteristics and data read on the datasheet of the inverter module used and, in this way, define the thermal behaviour of it in the simulations.
An equivalent thermal circuit can be drawn in detail (Fig. 2.78).

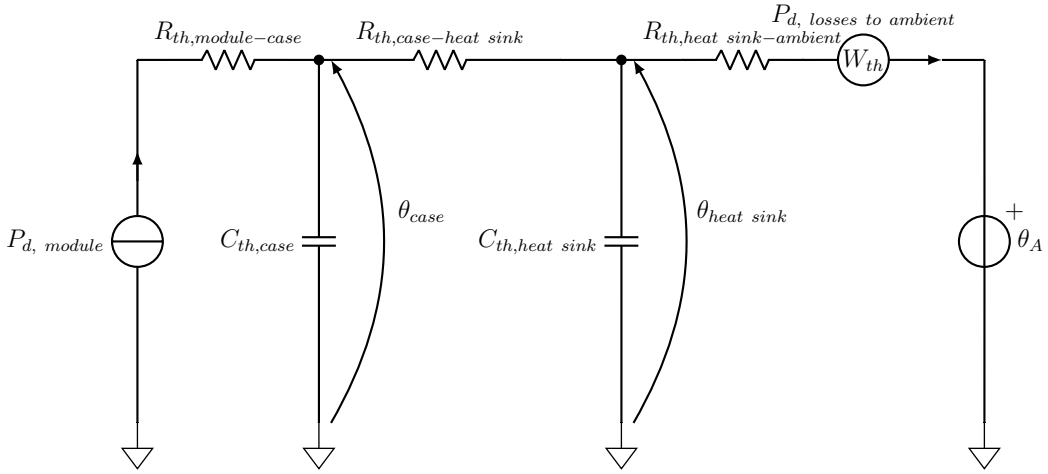


Figure 2.78: Equivalent Thermal Circuit.

where

- $P_{d,\ module}$ is the total power dissipated by the switches simulated inserting an equivalent current generator. This value is the sum of conduction and switching losses of all IGBTs and Diodes;
- θ_A is the ambient temperature. Considering that the ambient has an infinite thermal capacitance, the ambient temperature is supposed to be constant and equal to $25°C$;
- $R_{th,module-case}$ is the thermal resistance between the IGBT module and its case;

- $C_{th,\ case}$ is the thermal capacitance of the case. It represents the amount of heat that the case is able to store.

- $R_{th,\ case-heat\ sink}$ is the thermal resistance between the case and the heat-sink;

- $R_{th,\ heat\ sink-ambient}$ is the thermal resistance between the heat-sink and the ambient;

- $C_{th,\ heat\ sink}$ is the thermal capacitance of the case. It gives and idea of the amount of heat that can be accumulated by the heat-sink.

- W_{th} is the device to measure the total power losses to the ambient by the thermal circuit. It is simulated as an equivalent ammeter;

- θ_{case} is the temperature that the case reaches;

- $\theta_{heat\ sink}$ is the temperature that the heat sink achieves.

For this analysis, the thermal effect of the case have been neglected and the thermal resistance of the heat sink is supposed to be zero.
According to these hypothesis, the heat-sink absorbs the power losses dissipated by components within it's boundaries (see Fig. 2.77).
In PLECS, the *Probe* block enables you to monitor various quantities in a circuit. So, multiple *Probe* blocks are inserted to monitor the thermal behaviour of diodes, IGBTs and heat-sink. From these blocks, the following informations have been retrieved:

- the temperature that Diodes and IGBTs junctions reach during the thermal transient;

- the conduction and switching losses of diodes and IGBTs.

- the measured heat-sink temperature during the thermal transient;

Finally, the results obtained by the thermal analysis in the two simulations are summarize in Table 2.7.

Thermal Analysis		
	PWM BEM	**DPWM**
$\frac{P_{conduction}}{S_n}$ %	0.152%	0.154%
$\frac{P_{switching}}{S_n}$ %	0.380%	0.241%
$\frac{P_{losses\ to\ ambient}}{S_n}$	0.54%	0.41%
$T_{junction}$	41.2°	38.7°
$T_{heat\ sink}$	40.7°	37.9°

Table 2.7: Thermal analysis results.

S_n is the nominal power of the power converter. The results are calculated assuming that every IGBT have the same thermal behaviour of other IGBTs as well as a every diode has the same thermal behaviour of other diodes. Therefore, $P_{conduction}$ and $P_{switching}$ are calculated with (2.57) and (2.58).

$$P_{conduction} = 6 \cdot (P_{conduction,IGBT} + P_{conduction,Diode}) \tag{2.57}$$

$$P_{switching} = 6 \cdot (P_{switching,IGBT} + P_{switching,Diode}) \tag{2.58}$$

Finally, the total power losses to ambient $P_{losses\ to\ ambient}$ are evaluated using (2.59).

$$P_{losses\ to\ ambient} = P_{switching,IGBT} + P_{switching,Diode} \tag{2.59}$$

In the end, it is possible to conclude that modulating with the DPWM technique is advantageous from the thermal point of view. In fact, although conduction losses are almost the same, switching losses are reduced of 0.30% respect to the case in which the PWM BEM is used.
Moreover, also the junction and the heat-sink temperature reached a the end of the thermal transient are reduced.

However, as it is visible from Fig. 2.75, in DPWM technique the regions in which the phase leg switches are clamped to the upper or lower voltage level cause a distortion in the output voltage waveform introducing low frequency harmonics.
For that reason, the THD values of voltage and current result higher in the simulation involving the DPWM technique compare to the one using the PWM BEM.

2.7 Creation of the Grid-Forming Code

In this paragraph is presented the grid-forming code implemented in C that it is used to control the two levels three phase inverter of the experimental setup. First of all, a top-down approach is adopted and its main steps are provided in the Fig. 2.79.

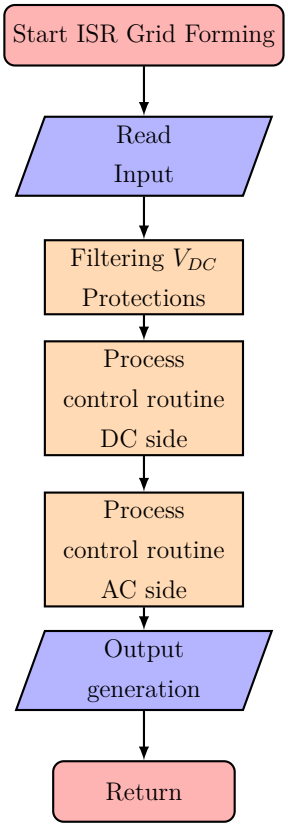


Figure 2.79: General flow chart grid-forming.

Every time that the interrupt service Routine (ISR) is called, the grid-forming code is executed. At the beginning, the input are read and the protection are activated if it is necessary. Before proceeding, the bus voltage (V_{DC}) is filtered. After that, the control routines of the DC side and of the AC side are processed. Finally, the output are generated.

Both the control routines are divided into states that permit to execute only a part of the code at every interrupt call. It gives the chance to develop the control algorithms in the time depending on the state of the inverter.
The process of the two control routines in the AC side and in the DC side are presented in details in the Fig. 2.80 and Fig. 2.81.

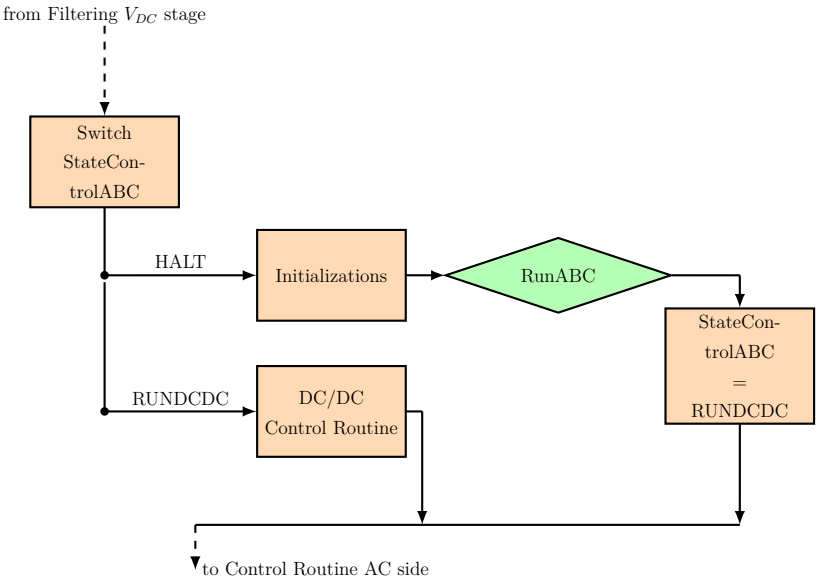


Figure 2.80: Control Routine DC side

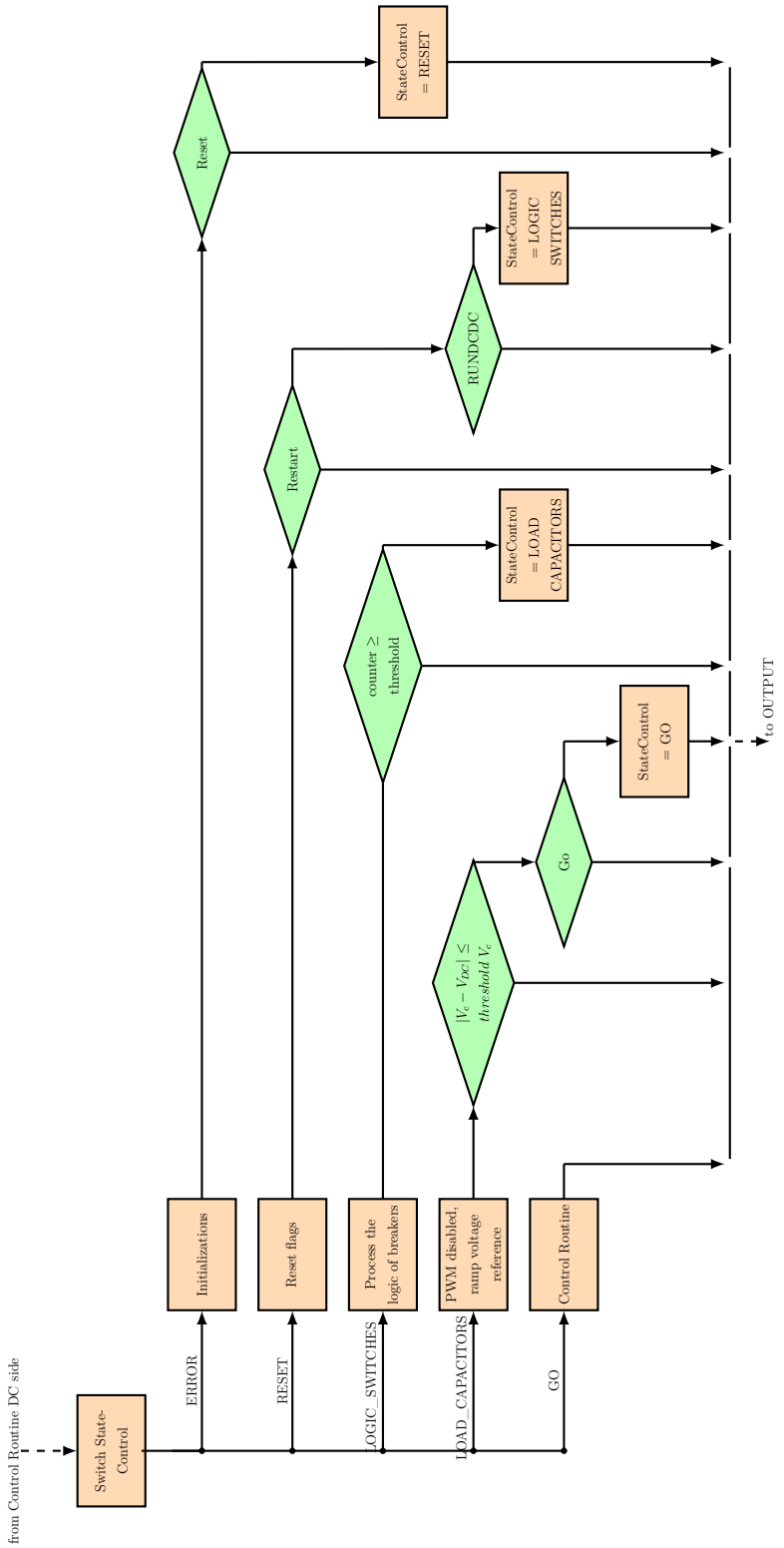

from Control Routine DC side
Switch State-Control
ERROR
RESET
LOGIC_SWITCHES
LOAD_CAPACITORS
GO
Initializations
Reset flags
Process the logic of breakers
PWM disabled, ramp voltage reference
Control Routine
Reset
Restart
RUNCDC
counter ≥ threshold
$|V_c - V_{DC}| \leq threshold\ V_c$
Go
StateControl = RESET
StateControl = LOGIC SWITCHES
StateControl = LOAD CAPACITORS
StateControl = GO
to OUTPUT

Figure 2.81: Control Routine AC side

Starting from the control routine in the DC side shown in Fig. 2.80 a detailed explanation of the the states is presented. The variable that determines the states is "StateControlABC". So, these states are:

1. **HALT**, in which the variables, such as the references and the integral parts of the regulator are reset and the PWM is disabled. When the push button "RUNABC" is pressed and the analog to digital converter is ready, the *"StateControlABC"* variable switches to the next state;

2. **RUNDCDC**, where the PWM is enabled and the cascade loop control of the DC side is implemented using a ramp reference voltage for the external control loop.

Moving on the control routine in the AC side shown in Fig. 2.81, the variable that controls the state is called *"StateControl"* and these are:

1. **ERROR**, which is the safety state of the DC/AC inverter. In fact, if a protection is activated, a flag is switched on and the *"StateControl"* variable is set to *"ERROR"*. Here, the PWM is disabled and the references and the integral part of the regulators are setted to zero. Moreover, all the breakers are opened.
   When the push button *"Reset"* is pressed and the analog to digital converter is ready, the *"StateControl"* variable switch to the *"RESET"* state;

2. **RESET**, where protection flags are reset. When the push button *"Restart"* is pressed the control switch to the next state.

3. **LOGIC_OF_SWITCHES**, that are a series of three states processed one after the other. They implement the logic of the two breakers: the RST main breaker and the soft-start breaker. Explaining better, it is necessary to close the soft-start relé, then open it and finally close the RST main breaker subsequently in different steps.
   When this logic is completed the *"StateControl"* variable is set to *"LOAD_CAPACITORS"*

4. **LOAD_CAPACITORS**, where the PWM at AC side is enabled and the output filter capacitors are charged imposing a ramp reference voltage. The system is controlled using a Single Loop voltage control with a PI regulator. When the difference between the voltage measured on the capacitors and the reference voltage imposed is lower than a fixed and accepted threshold, the control switch to the *"SOFTSTART"* state if the push button *"Go"* is pressed.

5. **SOFTSTART**, in which the soft-start of the load procedure is implemented. After a fixed time, the control switch to final state

6. **GO**, where the real control routine of DC/AC inverter is implemented. Imposing the value of the variable *"ControlType"* it is possible to set the control to test (Single Loop voltage control or Dual Loop). Moreover, it is possible to choose the regulator (PI or PRES) used in the control scheme varying the value of the variable *"RegulatorType"*.

The C code of the complete control routine is presented section 7.4.

2.8 Complete Model

Before proceeding to implement the grid-forming control to real inverter, a complete model of the system was simulated in PLECS implementing the C-code described in section 2.7 for the DC/AC converter. The complete model simulated is shown in Fig. 2.3. The DC/DC converter is controlled using the cascaded control loop strategy as described in section 2.2. The four different grid-forming strategy illustrated in section 2.3 were simulated. They were:

1. Single loop voltage control using PI regulator;
2. Single loop voltage control using PRES regulator;
3. Dual loop control using PI regulators;
4. Dual loop control using PRES regulators.

The connection of different type of load was simulated:

- Resistive Load;
- Inductive Load;
- Diode Rectifier used as active load;

Fig. 2.82, Fig. 2.83, Fig. 2.84 and Fig. 2.85 shows simulations results obtained using PLECS.

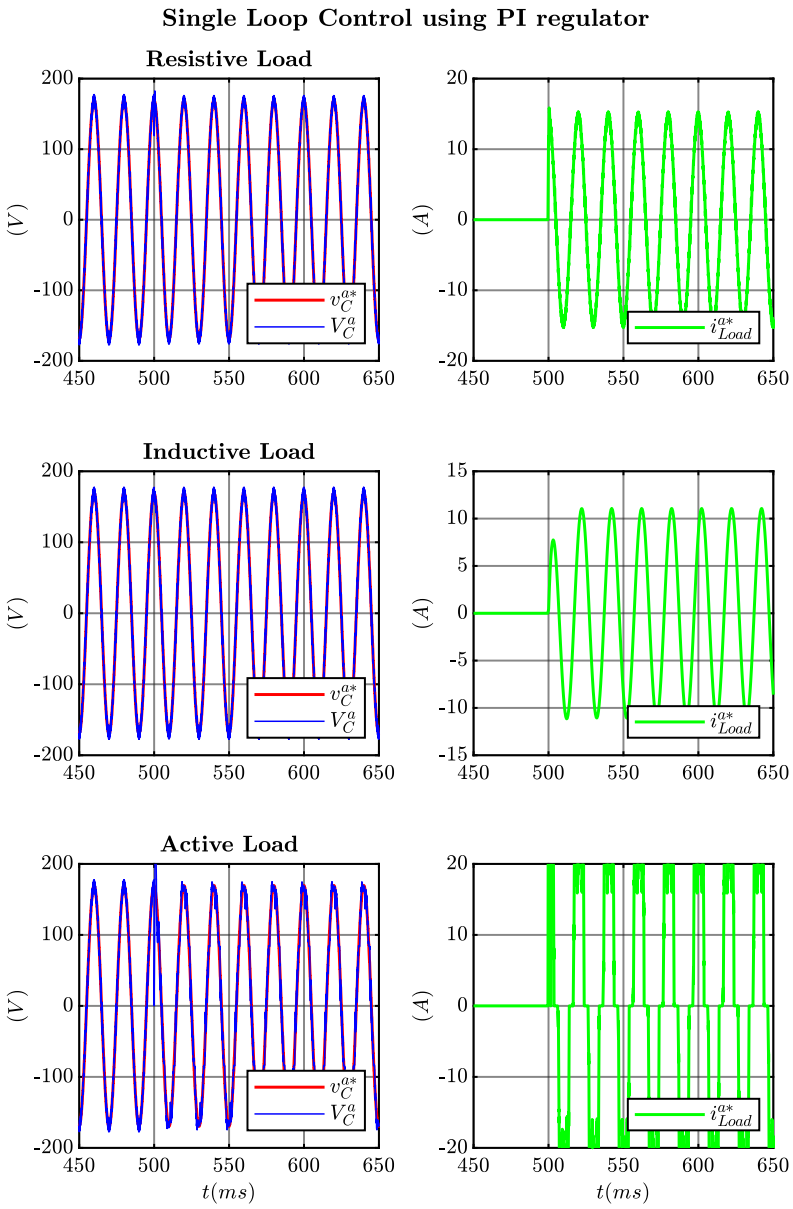


Figure 2.82: Simulation results obtained controlling the power converter with the single loop voltage control strategy using a PI regulator.

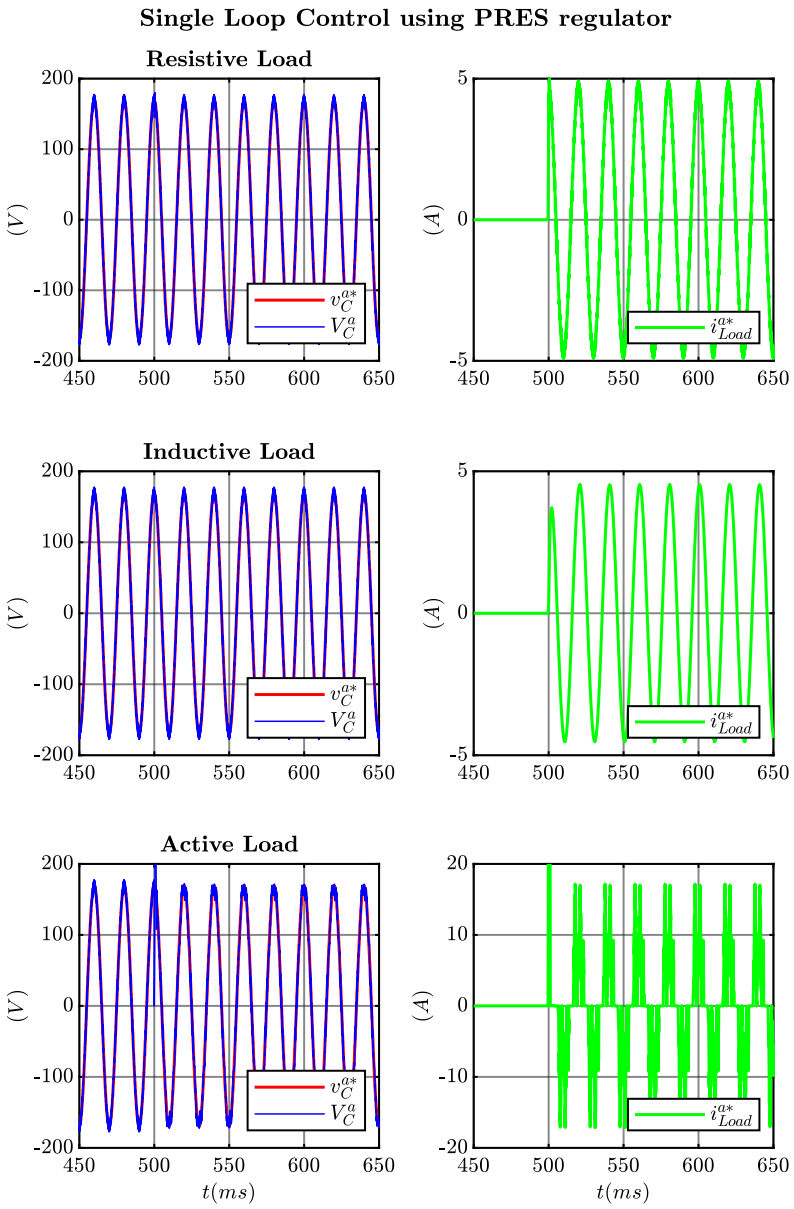


Figure 2.83: Simulation results obtained controlling the power converter with the single loop voltage control strategy using a PRES regulator.

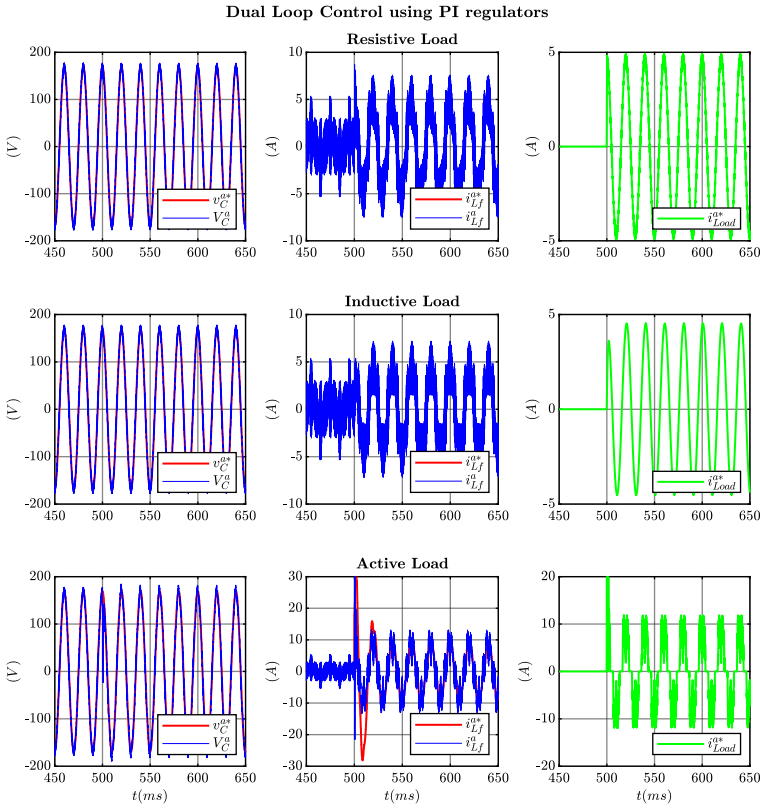


Figure 2.84: Simulation results obtained controlling the power converter with the dual loop control strategy using PI regulators.

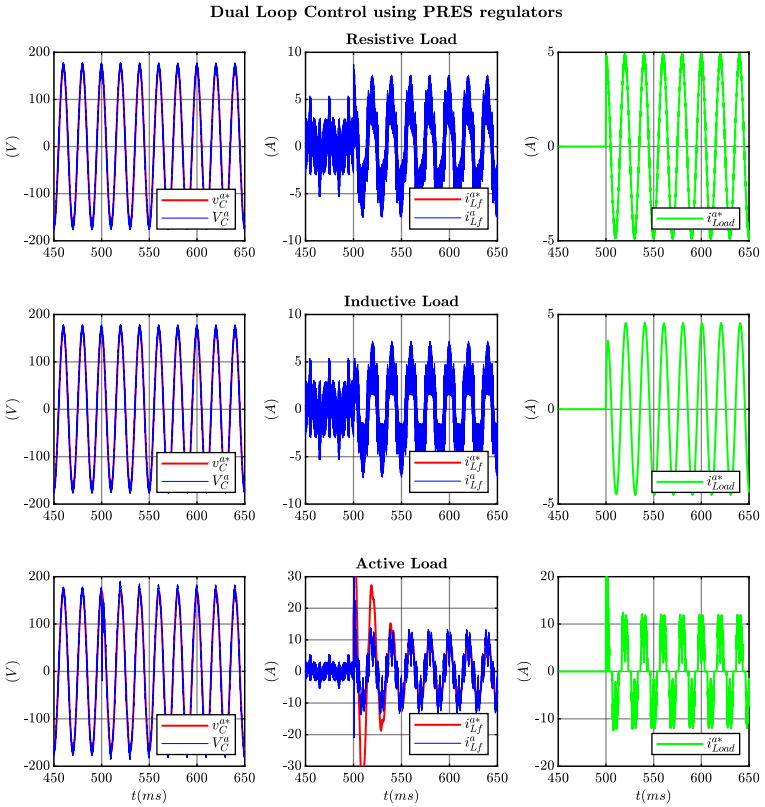


Figure 2.85: Simulation results obtained controlling the power converter with the dual loop control strategy using PRES regulators.

The simulations gave the expected results, so, it is possible to conclude that the C code is correct and it is possible to implement it into the real setup.

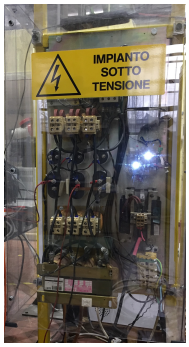
2.9 Description of the Experimental Setup

The experimental setup is composed by the following instruments:

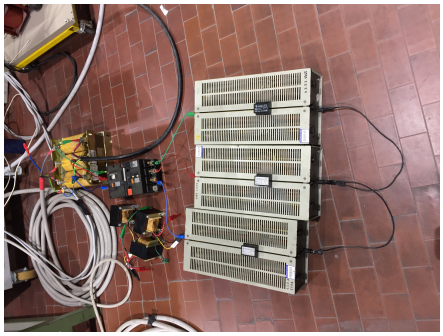
- the three phase two level inverter shown in Fig. 2.86a.
- a DC voltage source that sets the input voltage V_{in} (Fig. 2.86b).
- output converter filter inductors and capacitances (Fig. 2.86c).
- resistors and inductors used as passive load (Fig. 2.86d).
- a dSpace platform where the control strategy is implemented for the inverter (Fig. 2.86e).
- dSpace Control Desk, which is a debugging software that permits the user to communicate directly to dSpace. In fact, a debugging window has been programmed to be able to set voltage reference value ad to give GO and the RESET command to the control. This window is shown in Fig. 2.87
- Simulink and Matlab, where the model of the system and the C code is written. Simulink directly communicates with dSpace.
- Pump for the cooling circuit (Fig. 2.86f).

(a) Three phase two level inverter used in the experimental setup.

(b) DC voltage source.



(c) Picture of the output LC filter.



(d) Resistors and Inductors load.

(e) dSpace platform.

(f) Cooling circuit pump.

Figure 2.86: Experimental Setup Components.

The debug window designed in ControlDesk is illustrated in the following picture Fig. 2.87:

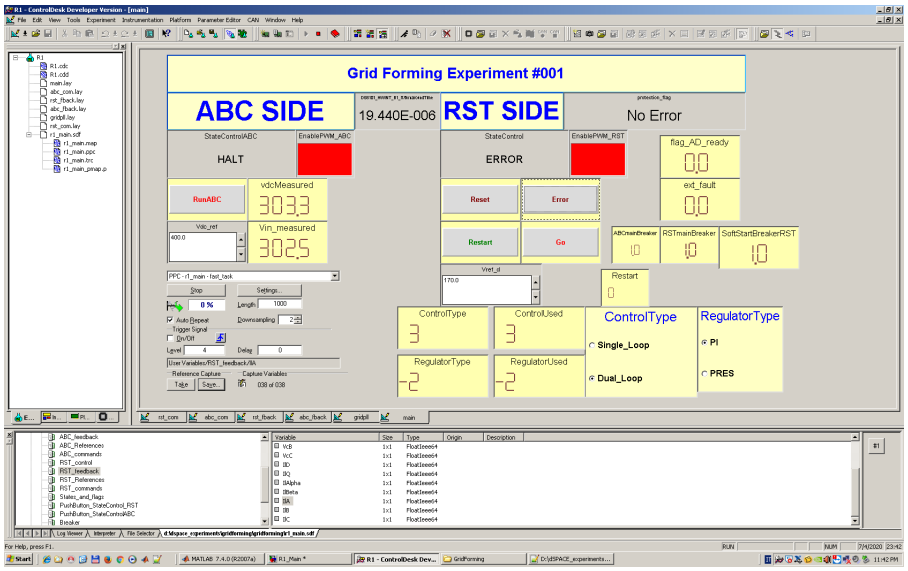


Figure 2.87: Debugging window in ControlDesk.

In the window there are two push buttons used to manage the State of the control routine of the DC/DC and the DC/AC converters. The DC and AC voltage references are setted using two *dSpace NumericInput Control windows*. The yellow windows are *dSpace Display Control* that permits the user to read the measured values. There are also two important boxes that allow the user to set with which control strategy and with which type of regulator the control must operate. In addition, there are two *dSpace Message Control* windows that shows the state of the PWM Enable variables. If the modulation is activated, these two windows become green.

A scheme of the test bench described is illustrated in Fig. 2.88.

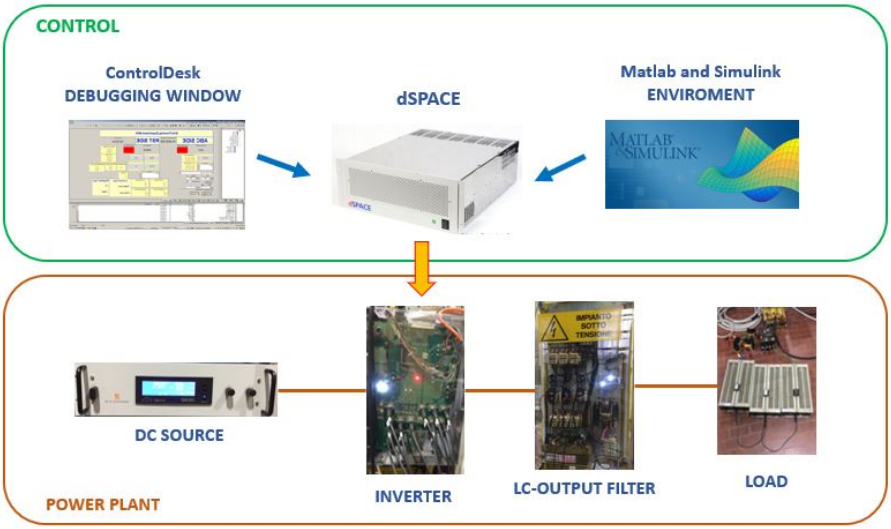


Figure 2.88: Block scheme of the experimental setup used to validate the grid-forming control algorithm.

Finally, the values of the main parameters of the components used are summarized in Table 2.8:

Experimental Setup	
Parameter	**Value**
S_n	15 kVA
f_{sw}	10 kHz
V_n	$120 \cdot \sqrt{3}$ V
L_F	545µH
C_F	22µF
V_{in}	300 V

Table 2.8: Inverter and Output filter parameter value of the experimental setup.

2.10 Experimental Tests and Results

Several experimental tests were conducted to validate the control designed and described in this chapter. In fact, the single loop and the dual loop control were tested using both PI and PRES regulators. Two types of passive loads and an active load were connected to the PCC to verify the correct behaviour of the control implemented:

- A pure resistive load that absorbs $P_{Load} = 8\%S_n$, which corresponds to $R_{load} = 36\Omega$;
- An inductive load that absorbs

$$\begin{cases} P_{Load} = 8\%S_n \longleftrightarrow R_{load} = 36\Omega \\ Q_{Load} = 27\%S_n \longleftrightarrow L_{load} = 32\mu H \end{cases} \tag{2.60}$$

In Fig. 2.89, Fig. 2.90, Fig. 2.91 and in Fig. 2.92 are shown the results recorded by ControlDesk during the tests when first a resistive load and then an inductive load is connected to PCC. For the tests involving PI regulators, the variable shown are plotted in *d,q* axis, while in the tests involving PRES regulators, the variable are plotted in α, β axis.

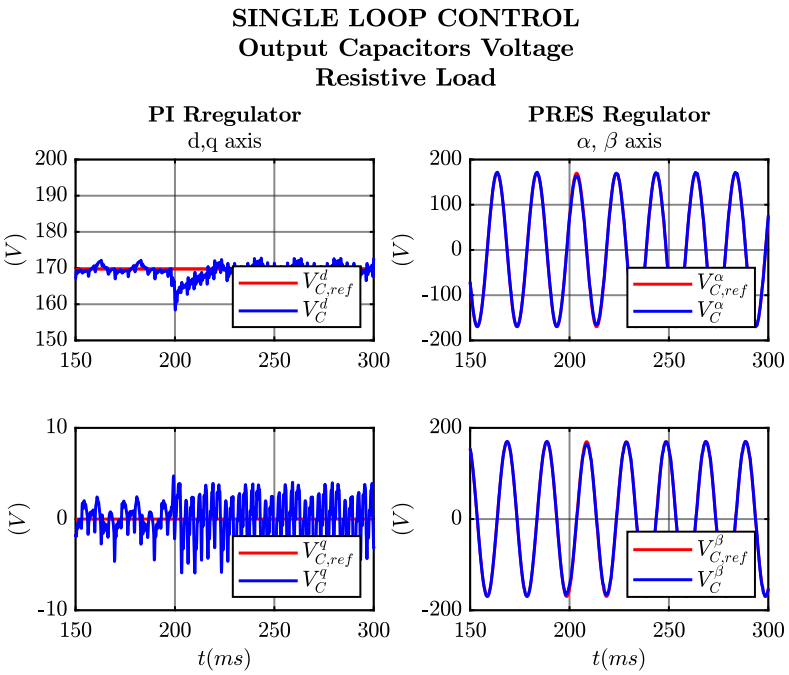


Figure 2.89: Experimental results acquired using ControlDesk environment connecting a resistive load.

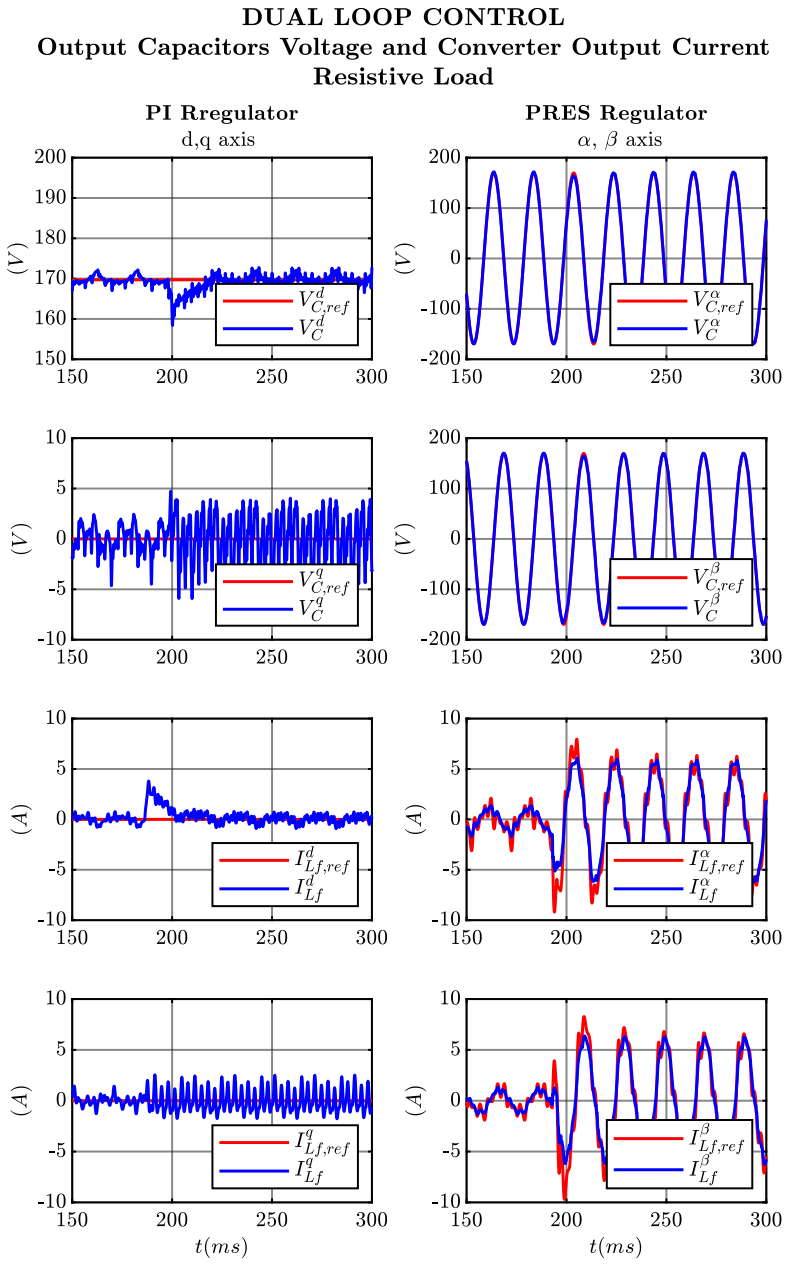


Figure 2.90: Experimental results acquired using ControlDesk environment connecting a resistive load.

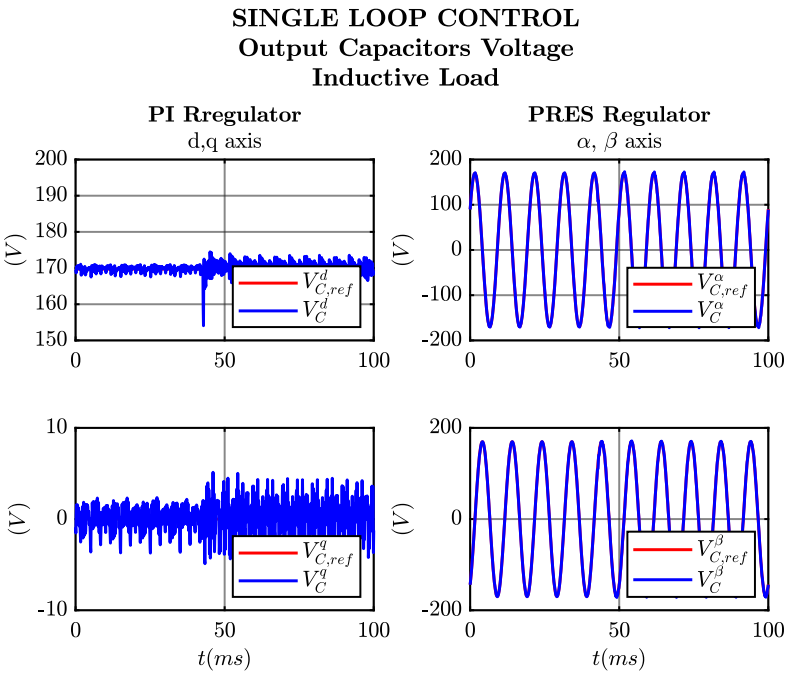


Figure 2.91: Experimental results acquired using ControlDesk environment connecting a inductive load.

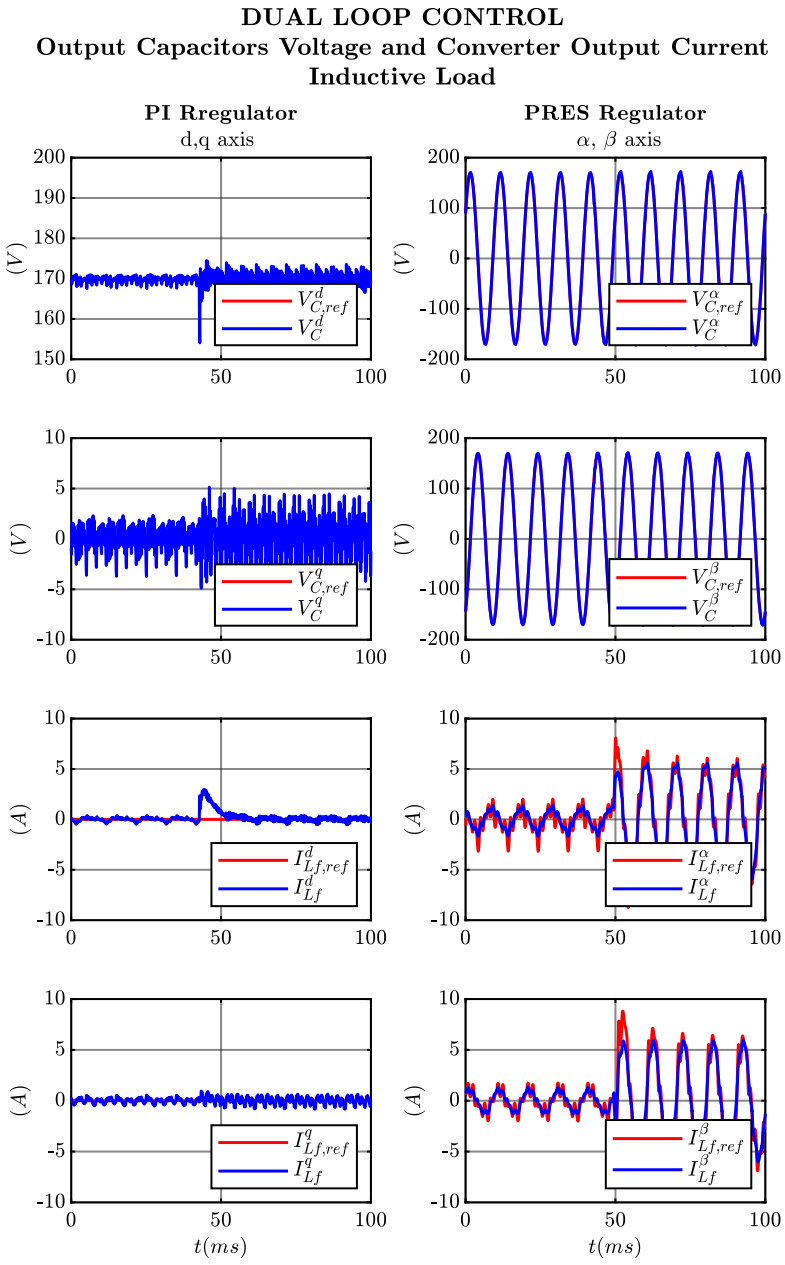
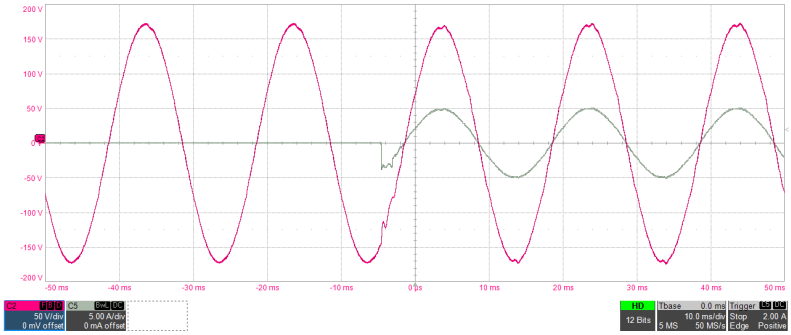
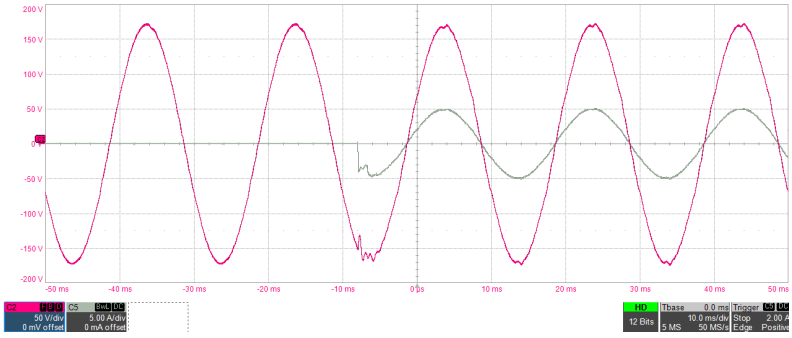


Figure 2.92: Experimental results acquired using ControlDesk environment connecting a inductive load.

It is possible to confirm that the regulators are well designed because the feedback signals of both the voltage and the current loop track the reference ones. The adoption of the dual loop control strategy in spite of the single loop, gives the chance to limit the current in a situation in which this is necessary, for example, if a fault occurs in the microgrid. However, in normal operating condition both the control strategies are good. In Fig. 2.93, Fig. 2.94, Fig. 2.95 and Fig. 2.96 are shown one voltage and one current phase measured during the tests using passive loads.



(a) PRES regulator.



(b) PI regulator.

Figure 2.93: One output converter phase current and output capacitors phase voltage measured while implementing the single loop control strategy when a resistive load is connected to the PCC. C2: v_a; C5: i_a.

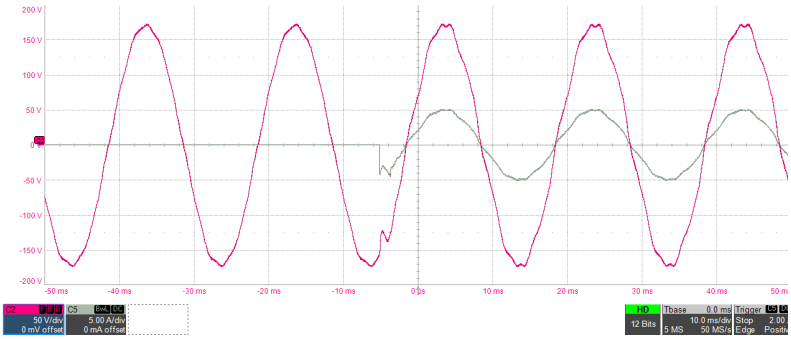
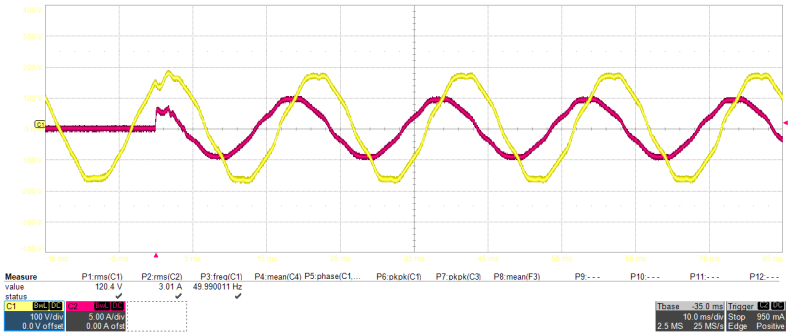
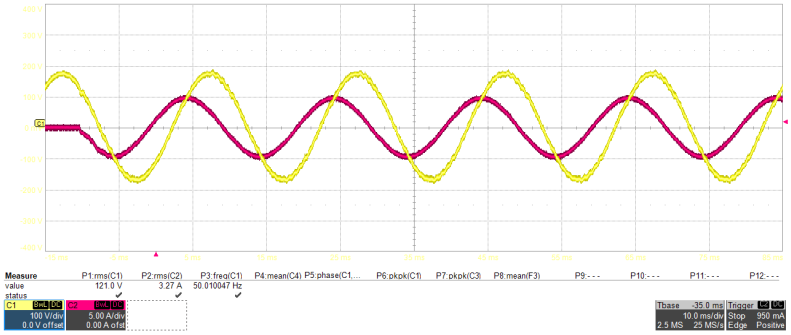


Figure 2.94: One output converter phase current and output capacitors phase voltage measured while implementing the dual loop control strategy using PI regulators and connecting the resistive load at the PCC. C2: v_a; C5: i_a.



(a) PRES regulator.



(b) PI regulator.

Figure 2.95: One output converter phase current and output capacitors phase voltage measured while implementing the dual loop control strategy when the inductive load is connected to the PCC. C1: v_a; C2: i_a.

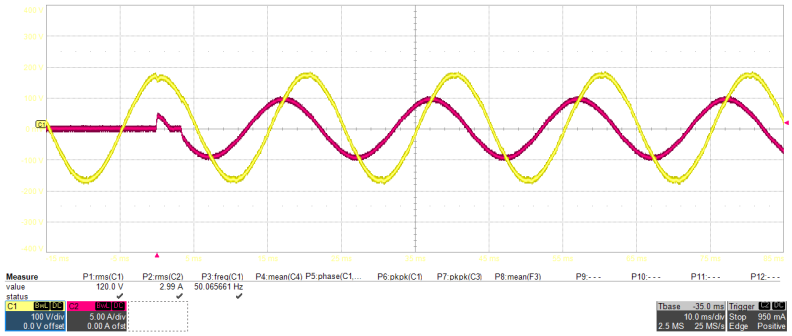


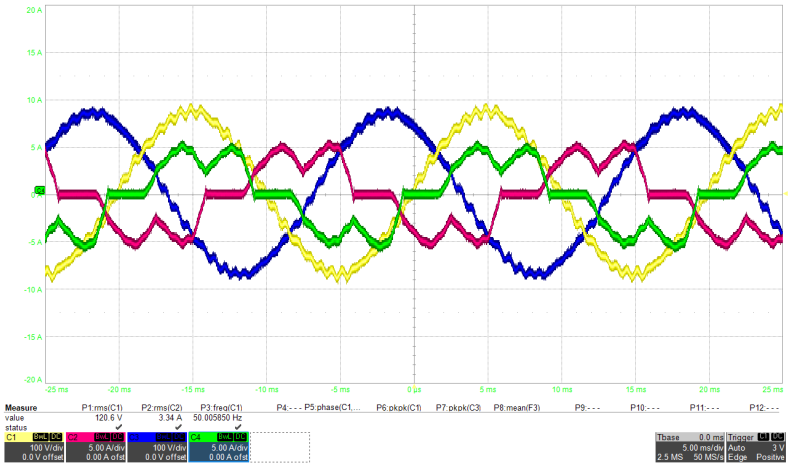
Figure 2.96: One output converter phase current and output capacitors phase voltage measured while implementing the single loop control strategy using PI regulators and connecting the inductive load at the PCC. C1: v_a; C2: i_a.

In Fig. 2.95a is shown that the voltage waveforms has a distortion. This is due to the fact that there is a little percentage of 5^{th} and 7^{th} harmonics. In order to solve this problem, a resonant filter must to be tuned to filter them. Moreover, it is visible in all the acquisitions a little distortion due to the dead-time effect, that is not compensated. Despite of these observations, the control in general gave good results.

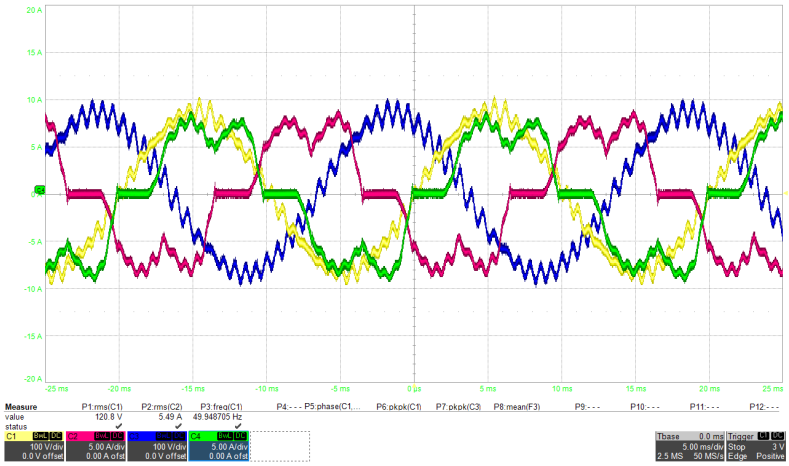
In the following figures the results obtained connecting the diode rectifier at the PCC as active load are reported. The diode rectifier is supplying a pure resistive load. Two different situations are tested:

- CASE A: Low load for the diode rectifier ($P_{Load} = 8\%S_n$).
- CASE B: High load for the diode rectifier ($P_{Load} = 40\%S_n$).

Since the output converter LC filter is not damped, in case B a significant distortion is detected. This is due to the fact that the value of load resistance is too low and is not enough to properly damp the components related to the resonant frequency. An FFT of the signal is calculated to better understand the distortion. The results obtained implementing the dual loop control strategy using the diode rectifier as active load is reported in Fig. 2.97, while the ones recorded controlling with the single loop strategy are shown in Fig. 2.99.



(a) CASE A: Low load for the diode rectifier.



(b) CASE B: High load for the diode rectifier.

Figure 2.97: Waveforms recorded controlling the converter with the dual loop control strategy using PI regulators. The diode rectifier is connected to the PCC as active load . C1: v_a; C2: i_a; C4: i_b.

So, the voltage phase FFT of Fig. 2.97b is calculated:

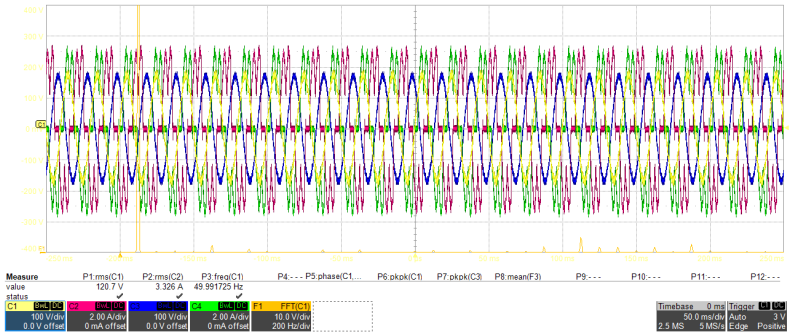
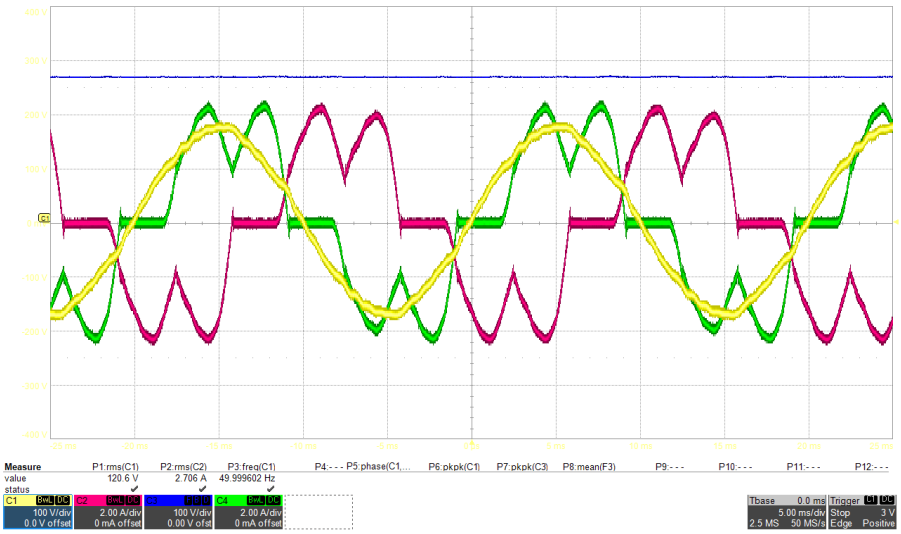
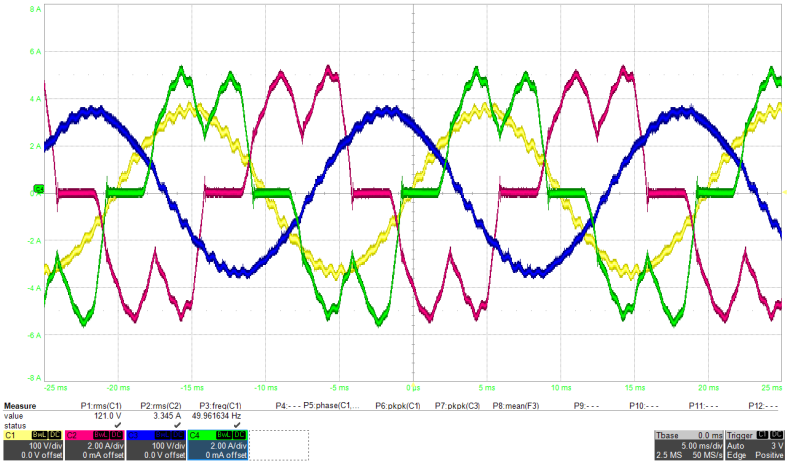


Figure 2.98: Phase voltage FFT. v_a; F1: fft_a.



(a) CASE A: Low load for the diode rectifier.



(b) CASE B: High load for the diode rectifier.

Figure 2.99: Waveforms recorded controlling the converter with the single loop control strategy using PRES regulator. The diode rectifier is connected to the PCC as active load. C1: v_a; C2: i_a; C4: i_b.

Also the voltage phase FFT of Fig. 2.99b is calculated:

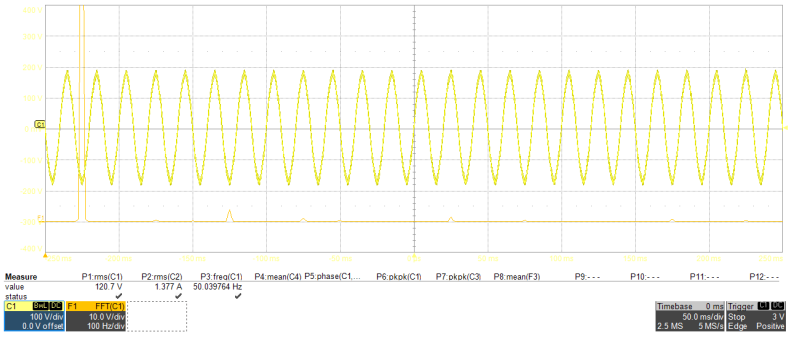


Figure 2.100: Phase voltage FFT. C1: v_a; F1: fft_a.

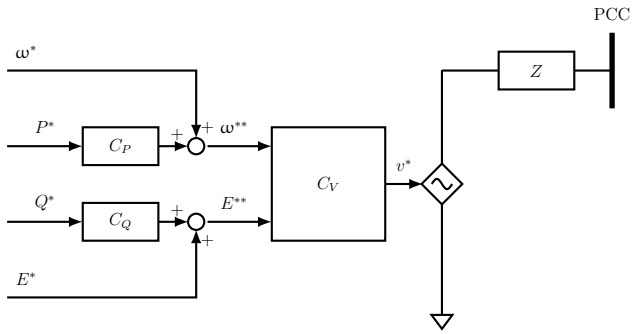
The high frequency components detected using the FFT algorithm in figure Fig. 2.98 and Fig. 2.100 are related to the resonant frequency as expected. This harmonic peaks can be efficiently decreased if a damping circuit is added to the LC output converter filter. One of the solutions described in section 2.4 can give satisfactory results.

In the end, it is possible to say that the grid-forming control implemented for the inverter was validated. To improve this control algorithm, a compensation of the dead time can me implemented using the analysis done in section 2.5. Moreover, also additional resonant filters can be implemented to attenuate the effect of the 5^{th} and 7^{th} harmonics. Overall, the grid-forming control implement gave satisfactory results during the experimental test and, therefore, the simulations done were validated.

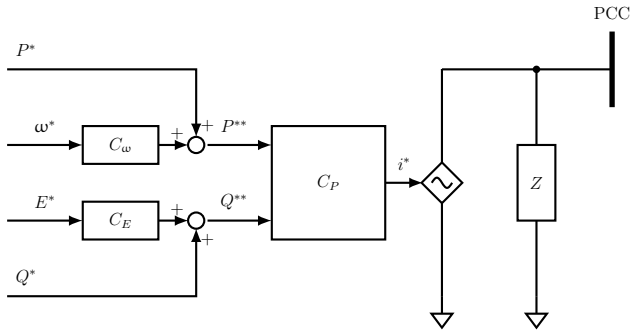
Chapter 3

Grid-Supporting Power Converters

The Grid-Supporting (GS) technique is one of the strategies used to control a power converter that is working connected to a microgrid or to the grid. GS power converters participate in the regulation of the AC grid voltage amplitude E^* and frequency ω^* by controlling the active and reactive power delivered to the grid [14]. A GS power converter can be controlled in closed loop to act as an AC voltage source or as an AC current source. Figure 3.1 shows the equivalent circuit of GS power converters.



(a) Grid-Supporting power converter controlled as an ideal voltage source.



(b) Grid-Supporting power converter controlled as an ideal current source.

Figure 3.1: Grid-Supporting Power Converters.

For this study, it is considered a GS power converter, controlled as an ideal AC voltage source, connected to the main grid through an inductive impedance. In this chapter, the active power loop is sized and implemented in PLECS simulation. An equivalent circuit can be drawn (Fig. 3.2).

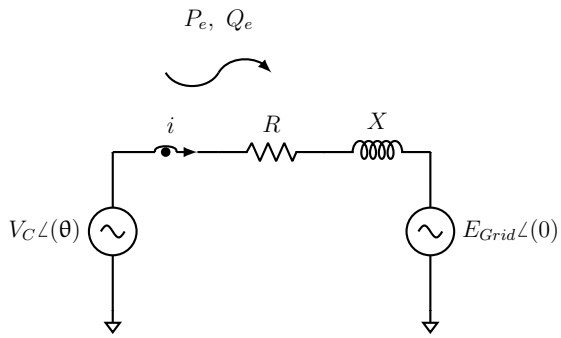


Figure 3.2: Active and reactive power transfer between a Grid-Supporting power converter and the main grid.

The relationship between the active and reactive powers transferred from the GS power converter to the grid and the PCC reference voltage and frequency are reported in (3.1) and (3.2).

$$P_e = \frac{E_{Grid} \cdot V_C \cdot sin\theta}{X} \tag{3.1}$$

$$Q_e = \frac{V_C^2 - E_{Grid} \cdot V_C \cdot cos\theta}{X} \tag{3.2}$$

where

- E_{Grid} is the rms voltage value of the grid;
- V_C is the rms voltage value on the PCC;
- θ corresponds to the phase-angle difference between the two voltages

The phasor diagram is proposed in Fig. 3.3.

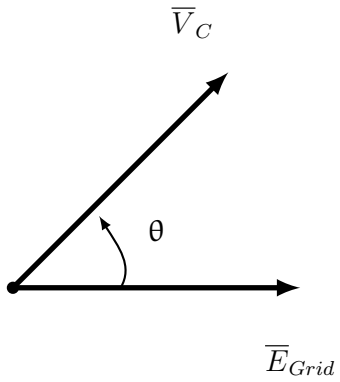


Figure 3.3: Phasor diagram of the equivalent circuit of Fig. 3.2.

So, the amount of active power delivered to the grid is connected to the phase-angle between the two voltage phasors while the amount of reactive power is related to the magnitude of voltages at the two terminals. This concept comes from the SM-grid interaction model, well known in power systems [24]. Here it is applied to power control. So, the power exchange is regulated adjusting the PCC voltage magnitude (V_C) and phase shift (θ).

The active power control implemented is the *Power Synchronization Control Loop* (PSL) described in [32, 23, 30]. PSL involves an external active power loop that sets the reference angle used for the transformation and an inner dual loop control for a VSI.

Fig. 3.4 shows the equivalent three phase circuit of the system controlled with the active power loop.

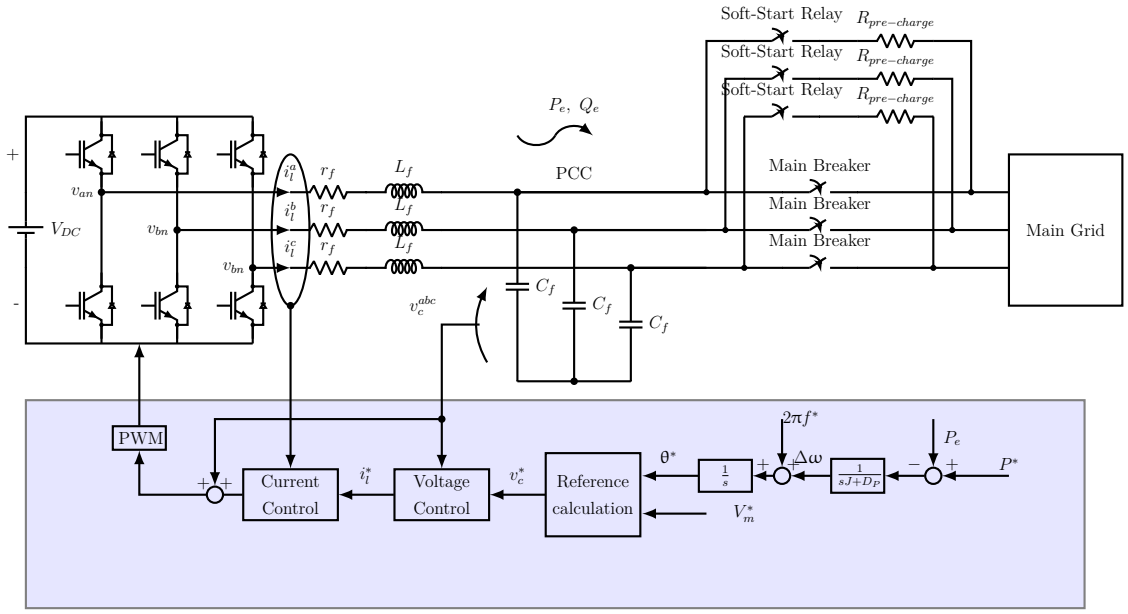


Figure 3.4: Power Synchronization Loop.

An equivalent single phase circuit is elaborated and it is illustrated in Fig. 3.5.

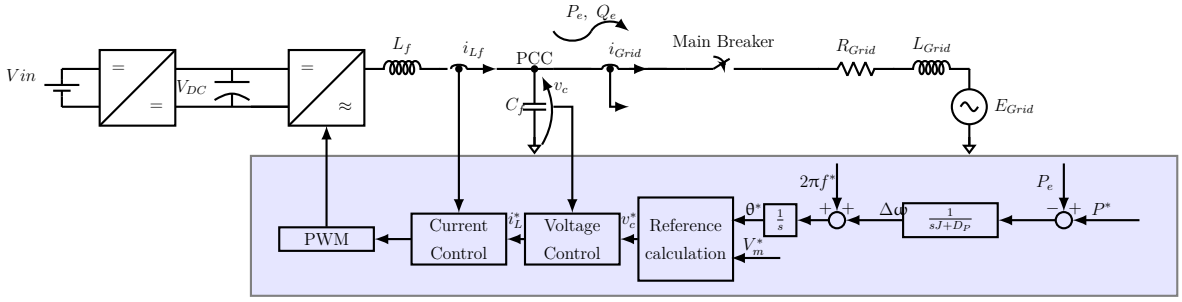


Figure 3.5: Power synchronization loop complete equivalent single phase circuit.

The inner dual loop control strategy is described in subsection 2.3.3. In Equation 2.3.4 and Equation 2.3.5 is shown the tuning of the two PI regulators used for the voltage and current loops.
Focusing on the active power control loop the block scheme is elaborated and reported in Fig. 3.6.

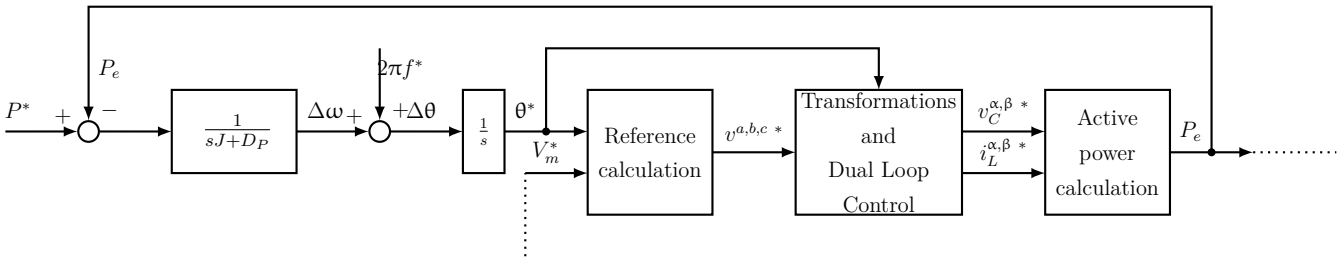


Figure 3.6: Active power control block scheme in detail.

As it is possible to see from the figure above, in order to transform the active power error ($\epsilon_P = P^* - P_e$) into an angle reference (θ^*) two integrator blocks are involved. The first block represents the active power regulator $G_P(s)$, whose transfer function is written in (3.3):

$$G_P(s) = \frac{1}{s \cdot J + D_p} \tag{3.3}$$

where

- J is a "virtual inertia". Its value is related to the inertia constant H and to the mechanical starting time T_M. It is "virtual" because the system uses the concept of power-synchronization mechanism between synchronous machines, but does not involves any of them. However, the nomenclature is the same.
- D_p is a damping constant factor designed to stabilize the active power loop.

The second block is a simple integrator which calculates the angle reference for the inner voltage loop control.
In [32], the transfer function $J_{P\theta}$ between the transferred active power and the phase angle shift is obtained as:

$$J_{P\theta}(s) = \frac{a_0 s^2 + a_1 s + a_2}{(R + sL)^2 + (\omega_1 L)^2} \tag{3.4}$$

where,

$$a_0 = \frac{L}{\omega_1}(E_{Grid,0} \cdot V_{C,0} \cdot cos\theta_0 - V_{C,0}^2) \tag{3.5}$$

$$a_1 = \frac{R}{\omega_1}(E_{Grid,0} \cdot V_{C,0} \cdot cos\theta_0 - V_{C,0}^2) \tag{3.6}$$

$$a_2 = \omega_1 L \cdot E_{Grid,0} \cdot V_{C,0} \cdot cos\theta_0 - R \cdot E_{Grid,0} \cdot V_{C,0} \cdot sin\theta_0 \tag{3.7}$$

$$\theta_0 = \pm arcos(\frac{V_{C,0}}{2E_{Grid,0}}) \tag{3.8}$$

The subscript 0 indicates the operating point in nominal conditions.

3.1 Tuning of the Power Synchronization Loop Regulator

For a synchronous machine the value of inertia J is correlated to the value of inertia constant H and to the mechanical starting time T_M according to (3.9), (3.10) and (3.11):

$$J = \frac{2H}{\omega_0^2} \cdot S_n \tag{3.9}$$

$$H = \frac{J \cdot \omega_0^2}{2S_n} \tag{3.10}$$

$$T_M = 2 \cdot H \tag{3.11}$$

where $w_0 = 2\pi f_0$ and $f_0 = 50Hz$.
For the application a inertia constant of $H = 4s$ is chosen. So, using (3.9) the value of J is found.

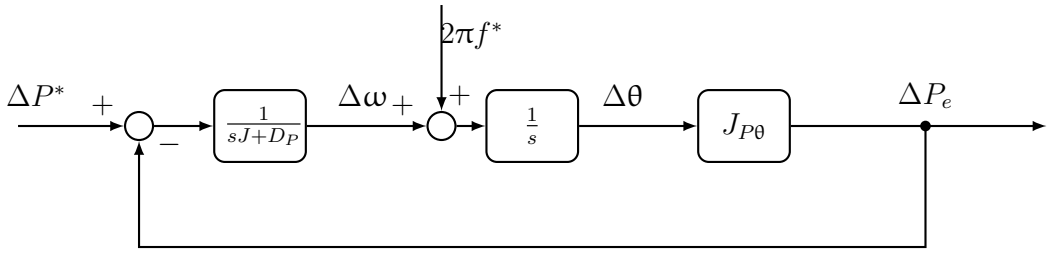


Figure 3.7: Active power control loop.

After that, referring to Fig. 3.7, the open and close loop transfer function are calculated using (3.12) and (3.13).

$$P_{OL} = \frac{1}{s \cdot J + D_p} \cdot \frac{1}{s} \cdot J_{P\theta}(s) \tag{3.12}$$

$$P_{CL} = \frac{1}{1 + P_{OL} \cdot 1} \tag{3.13}$$

Imposing a power loop bandwidth (ω_P) of $2\pi f_{crossover,PSL}$, the D_P value is found to stabilize the power loop according to the Bode criterion. So,

$$P_{OL(s)} = \frac{1}{s \cdot J + D_p} \cdot \frac{1}{s} \cdot J_{P\theta}(s) \tag{3.14}$$

$$= \frac{1}{s^2 \cdot J + s \cdot D_p} \cdot J_{P\theta}(s) \tag{3.15}$$

If $s \to j\omega_P$, $P_{OL}(s \to j\omega_P)$ become:

$$P_{OL(j\omega_P)} = \frac{1}{-\omega_P^2 \cdot J + j \cdot \omega_P \cdot D_p} \cdot J_{P\theta}(j\omega_P) \tag{3.16}$$

Considering that at the crossover frequency the magnitude of the bode plot of open loop transfer function crosses the $0dB$, can be written:

$$|P_{OL(j\omega_P)}| = 1 \tag{3.17}$$

$$|\frac{1}{j \cdot \omega_P \cdot D_p - \omega_P^2 \cdot J}| = \frac{1}{|J_{P\theta}(j\omega_P)|} \tag{3.18}$$

$$D_p = \sqrt{\frac{(|J_{P\theta}(j\omega_P)|^2 - (-\omega_P^2 \cdot J)^2)}{\omega_P^2}} \tag{3.19}$$

Choosing $f_{crossover,PSL} = 5Hz$, the bode plots of the open and close loop transfer function are evaluated in MATLAB and reported in Fig. 3.8.

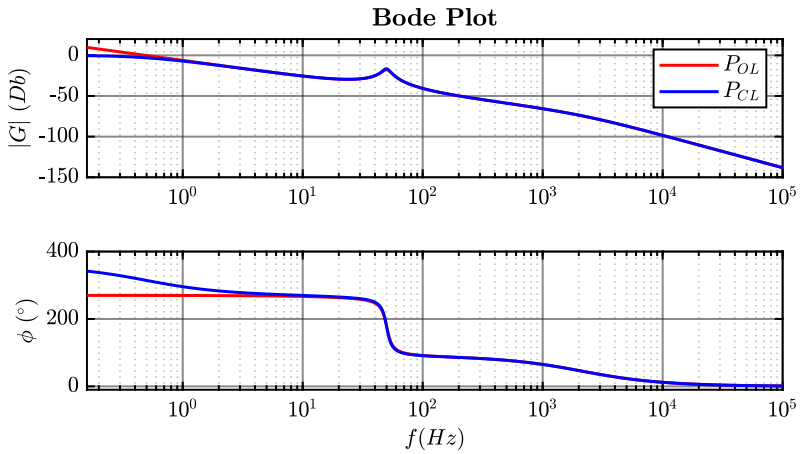


Figure 3.8: Power synchronization loop bode plots.

3.1.1 Power Synchronization Control Loop: Validation in PLECS Simulation

Then, the PSL is implemented in simulation PLECS for the system shown in Fig. 3.4. The simulation is divided in four steps:

1. a first safety state for the power converter is considered, in which the modulation is disabled.

2. the grid is disconnected and the inverter is controlled as Grid-Forming using the dual loop control strategy.

3. the power converter must be connected to the main grid, but it is important to ensure accurate voltage and frequency synchronization prior to the grid connection, in order to avoid large inrush currents. So, a synchronization state is provided in the control algorithm.

4. The power converter is connected to the grid and it is controlled as Grid-Supporting. From this moment, an active power loop works to regulate the active power exchanged between the two parts connected.

Each step corresponds to a specific "State" in the simulation. During the first active state in which the power converter is controlled using the grid-forming control

strategy, the phase shift θ of the PCC voltage is different from the one of the grid θ_{Grid}.

During the synchronization period, the power converter is controlled with the PSL control strategy imposing $P^* = 0$. In fact, if the reference power of the PSL is imposed to be zero, the control will act on the generated θ^* by moving it (Fig. 3.6) in order to synchronize it with the phase of the grid θ_{Grid}. When the exchanged power reaches the zero reference value, it will mean that synchronization has taken place (see (3.1)) and the main switch can be closed without problems.

The power P_e exchanged with grid is calculated using the component written in α, β stationary reference frame of the PCC voltage and the current flowing to PCC:

$$p = \frac{3}{2} \cdot (v_\alpha \cdot i_\alpha + v_\beta \cdot i_\beta) \tag{3.20}$$

Since during the synchronization process the inverter measured power will be 0 (grid disconnected), so, to ensure the virtual synchronization, a virtual power P_e is calculated. The synchronization is performed using a virtual (fictitious) impedance as in Fig. 3.9. This impedance emulates the real one that connects the PCC to the grid. However, in the virtual impedance does not flow current. This is a method used only to be able to have an estimation of the power exchanged P_e during the synchronization phase.

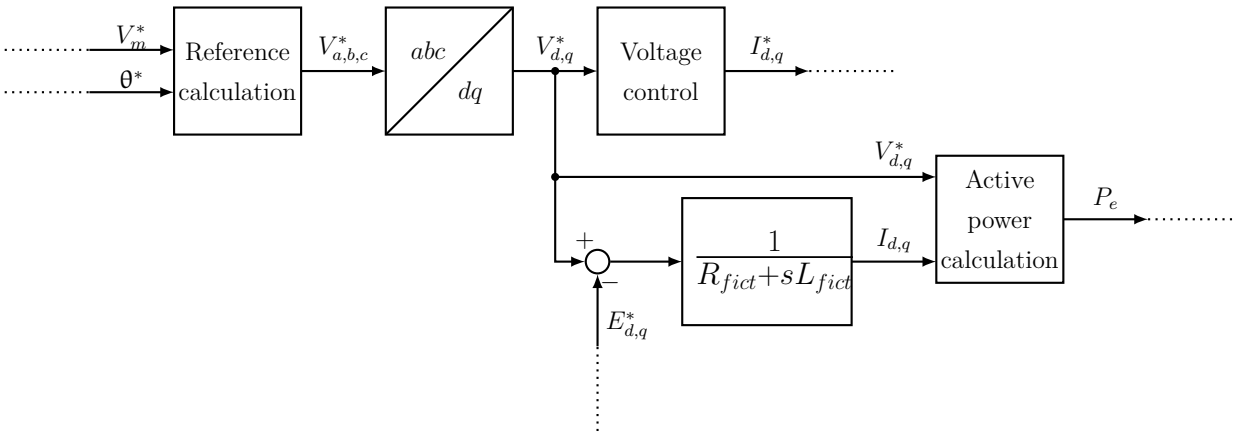
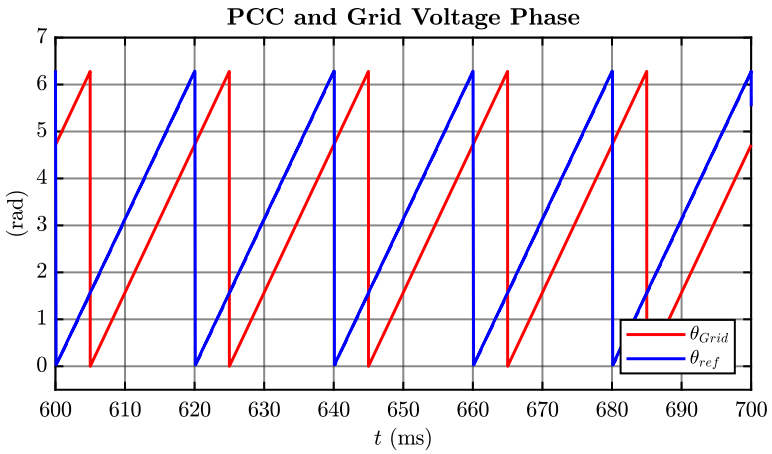
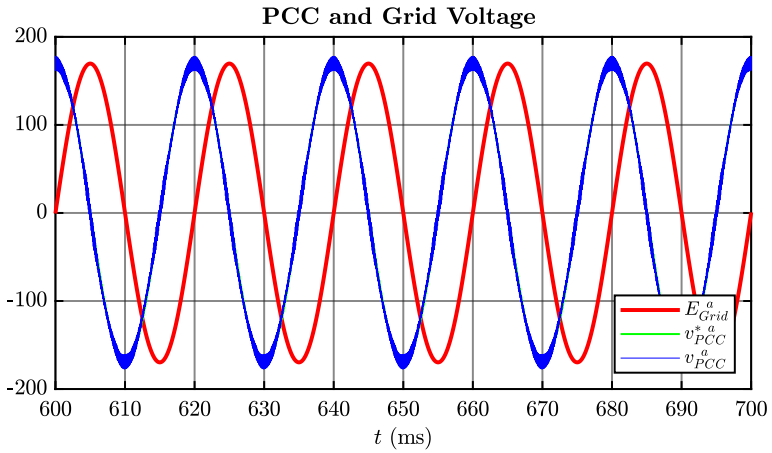


Figure 3.9: Active power calculation during the synchronization transient.

With this method no PLL is needed for synchronization [23].
So, focusing on the first active state, in which the power converter is controlled with the grid-forming control strategy, it is possible to see in Fig. 3.10 that the grid voltage and the PCC voltage have the same module but they are not in phase.



(a) PCC and grid voltage phase.



(b) PCC and grid voltage.

Figure 3.10: Results obtained in PLECS simulation during the state in which the power converter is controlled with the grid-forming strategy.

Fig. 3.11 shows the power reference and the virtual power calculated during the synchronization process. As written before, the power reference is set to zero in order to permit to the power synchronization algorithm to synchronize the voltage phase at the PCC with the grid voltage phase. During this range of time the power exchanged recorded is virtual; in fact a fictitious impedance, which simulate the real one between the PCC and the grid, is used to calculate the power exchanged

to permit the correct functionality of the PSL algorithm.

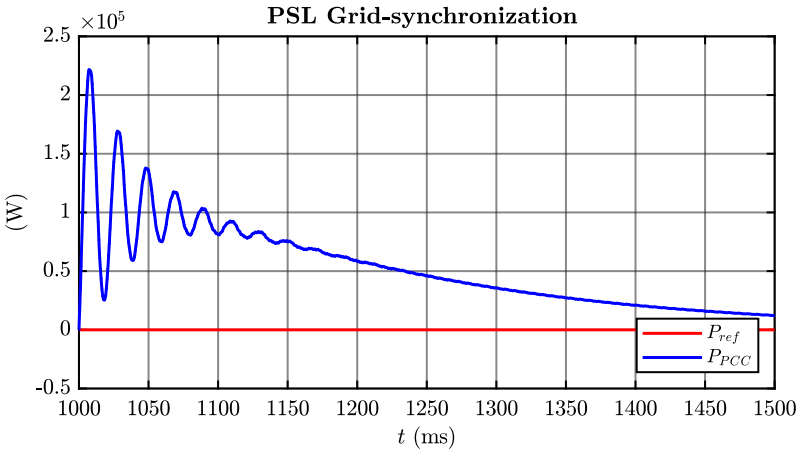


Figure 3.11: Active power reference and virtual power recorded during the synchronization process.

In Fig. 3.12 are shown the PCC and grid voltage phase during the synchronization process.

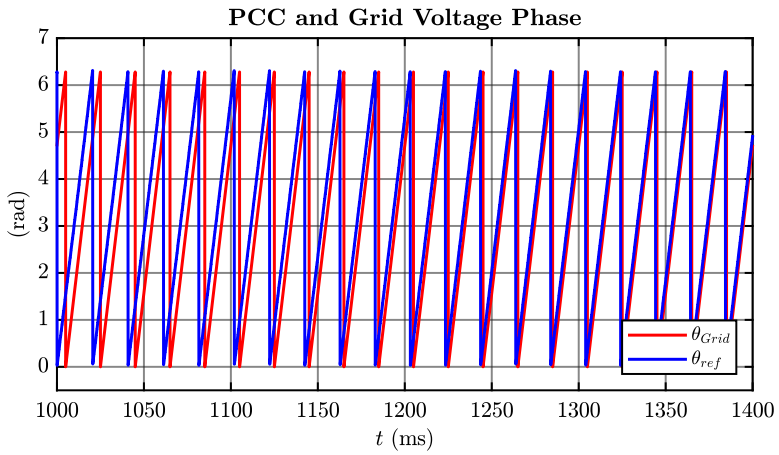


Figure 3.12: PCC and grid voltage phase during the synchronization process.

It is visible the synchronization process during which the θ^* tends to θ^{Grid}, so the phase shift between them tends to zero and also the power transferred. In the simulation the moment in which the synchronization is completed corresponds to the moment in which the active power flowing is around zero. In fact, if the two voltage vectors are synchronized, according to (3.1), θ tends to zero and so the active power transferred tends to zero.
Fig. 3.13 shows the PCC voltage synchronizes with grid voltage.

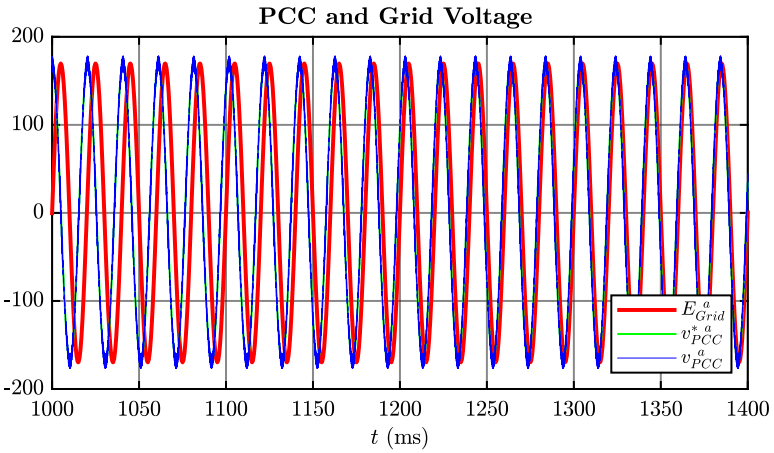


Figure 3.13: PCC and grid voltage obtained during the synchronization process.

When the synchronization process is done, the grid connecting switch is closed and the power converter starts to be controlled with the grid-supporting strategy described. Fig. 3.14 illustrates the final state in which the PSL algorithm is working. In this state, an active power reference is given and the control tracks the reference with 0 steady state error: the real transferred power P_e reaches the reference value.

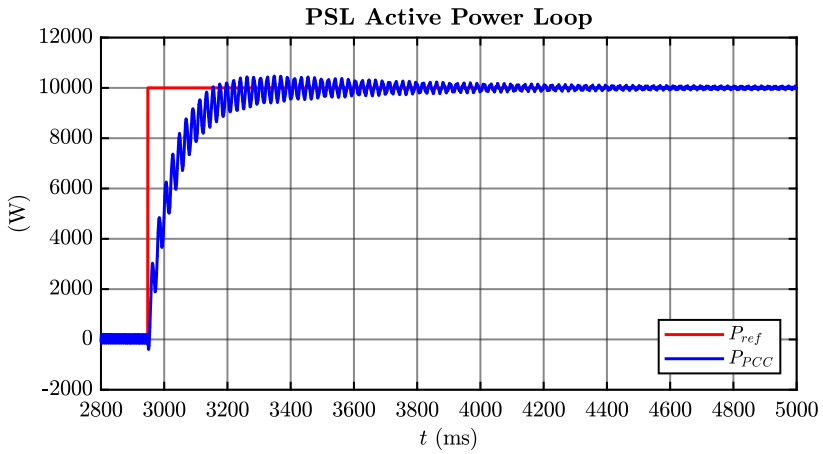


Figure 3.14: Active power results obtained in PLECS simulations during the state in which the power converter is controlled as grid-supporting implementing the PSL algorithm.

During this range of time the voltage at the PCC is perfectly synchronized with the grid voltage (Fig. 3.15).

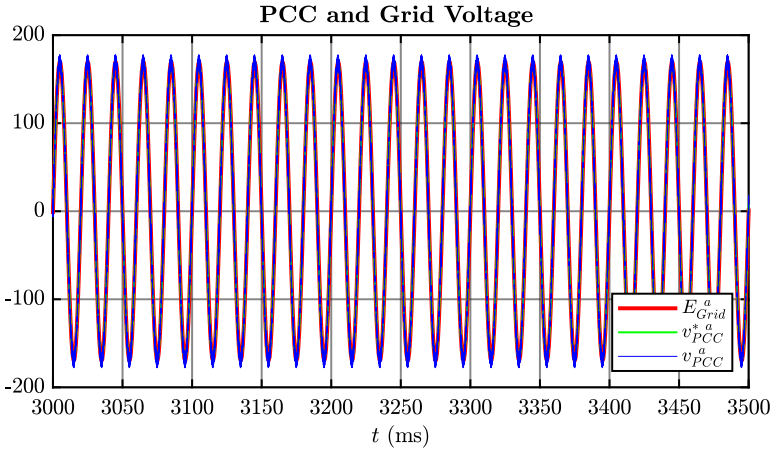


Figure 3.15: PCC and grid voltage during the final state.

Finally, this paragraph simulates the behaviour of a converter that works initially connected to a microgrid isolated and is subsequently connected to the main grid. Particular attention is given to the synchronization process, which is essential to avoid large inrush current. The synchronization procedure does not make use of any PLL, but the active power control of the converter is implemented giving a reference power of $0W$. It has been demonstrated that the PSL implemented gives good results in simulation PLECS.
So, this control can be used for future experimental test in which a Grid-Forming power converter working in a microgrid switches to be controlled as a Grid-Supporting power converter after the connection with the main grid.

Chapter 4

Mechanical Design

In this chapter, the mechanical design of the two identical support structures will be presented. Each of these structures will host a complete converter unit. This converter will be able to work connected to the grid or it can be used for motor control, depending on the purposes.
The idea is to use one of these two converters to build another test bench to test the grid-forming control strategy described in chapter 2.
The control algorithm of the converter unit will be managed by a rapid control prototyping system called *"dSpace"*.
A field-programmable gate array (FPGA) board will have the task of redirecting the commands coming from dSpace to the power component and also send informations to dSpace about the status of the system.
This FPGA will be mounted on a printed circuit board (PCB). More details about these boards will be provided in chapter 5.
The goal was to make the mechanical design of the entire support structure by creating its 3D assembly and producing 2D drawings with informations and dimensions to send to the manufacturing companies. The work was done using SolidWorks software.
To get an idea of the system that will be mounted, its electric circuit is shown in Fig. 4.1.

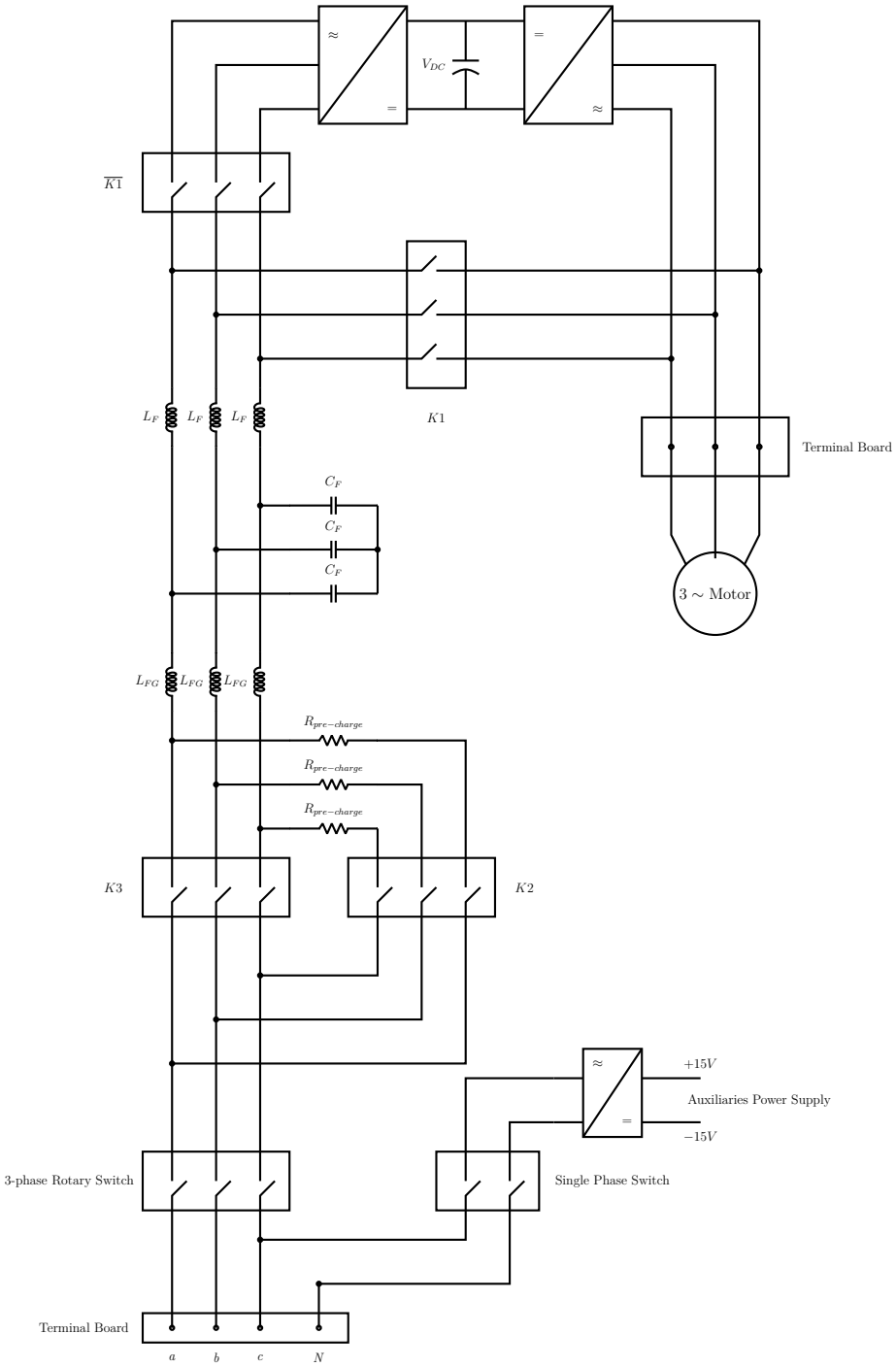


Figure 4.1: Equivalent electric circuit of the system that will be mounted on both structures.

The system presents the back to back converter equipped with an LCL-output filter.

There are contactors that will permit to connect the power converter directly to the grid and also to supply an induction motor.
The first part of the mechanical design involves the creation of the 3D models of each component that will be mounted on the structure according to mechanical informations read on their datasheets.
After that, an unique assembly is elaborated and a picture is shown in Fig. 4.2.

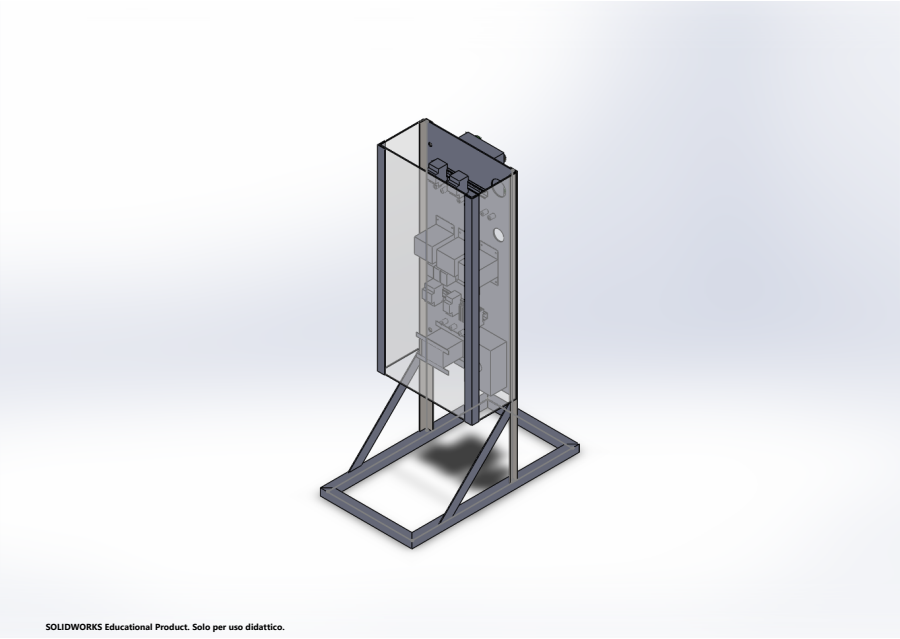
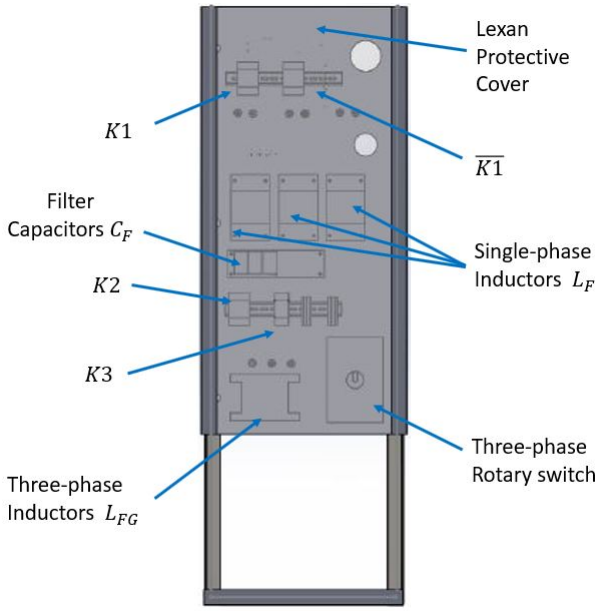
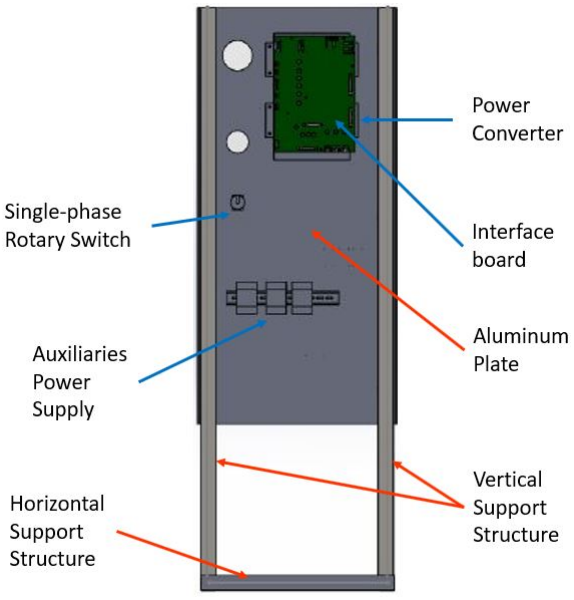


Figure 4.2: 3D assembly of the panel.

Fig. 4.3 presents the front and the rear view of the 3D assembly with a description of the various components.



(a) Frontal view of the complete assembly.



(b) Rear view of the complete assembly.

Figure 4.3: Prospective views of the complete assembly.

Basically, the vertical support consists of a rectangular structure of L-shaped iron profiles on which an aluminium plate is fixed with bolts. All the power components, that will be covered by a Lexan cover, will be mounted on the front of this plate, while the signal ones will be positioned on the rear side.
The vertical structure is mounted on a rectangular base of square section beams and is supported by two 45° iron rods. This rectangular base will be mounted on 4 wheels which will facilitate the movement of the structure.
The fixing holes of all the components that will be mounted have been designed on the aluminium plate according to the mechanical informations given by the datasheets.
After that, the entire support structure has been isolated and the 2D drawings have been produced and sent to the manufacturing company. They were provided with the various component dimensions and informations regarding the materials and the type of holes desired.
Fig. 4.4 shows only the support structure that has been commissioned.

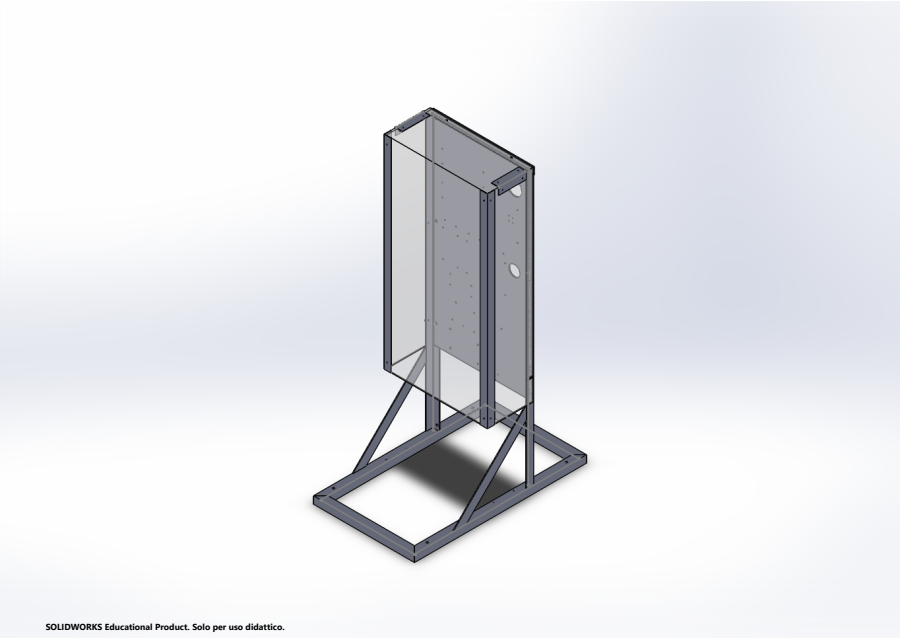
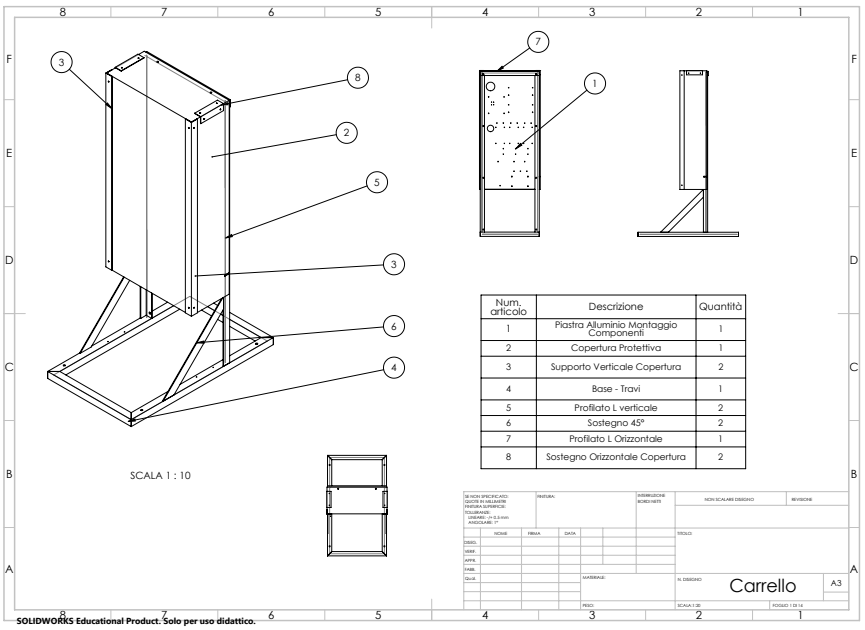


Figure 4.4: 3D assembly of the support structure of the panel.

Fig. 4.5 shows the first page of the 2D drawing elaborated. The complete document in insert section 7.5.



Num. articolo	Descrizione	Quantità
1	Piastra Alluminio Montaggio Componenti	1
2	Copertura Protettiva	1
3	Supporto Verticale Copertura	2
4	Base - Travi	1
5	Profilato L verticale	2
6	Sostegno 45°	2
7	Profilato L Orizzontale	1
8	Sostegno Orizzontale Copertura	2

Figure 4.5: 2D drawing.

Chapter 5

Interface FPGA Boards and Experimental Validation of the New Setup

In this chapter the work carried out on printed circuit boards (PCB) and FPGA will be presented. In fact, there are two identical printed circuits, which as anticipated in chapter 4, will be mounted on two different structures and will be equipped with a FPGA board. These structures are provided with a converter unit and the control is implemented on dSpace. The function of FPGA board is to interface the control of the converter unit and the power components. In fact, it will take care of receiving commands from dSpace, processing them and redirecting them to the various components thorough specific pins. In addition, it will also take care of reporting any system conditions to dSpace, such as the detection of a fault case. So, FPGA boards represents the interface between the signal part and the power part of the entire system.

In this application, the FPGA module used is the *Trenz Electronic TE0725* integrating a Xilinx Artix-7 (35-100T) and 32 MByte Flash memory for configuration and operation [28].

To permit the correct exchange of informations between dSpace and the FPGA boards, a protocol of communication must be defined. It will be presented in the

following sections.
Therefore, in this chapter the work done on these boards will be described step by steps. It includes:

- Board assembly;
- Preliminary Test on the principle components (Leds, Relays, digital communications pins, ADC e DAC device);
- Final project to interface dSpace to the power part (power converter, breakers etc...)

5.1 Assembly of Printed Circuits

The boards were designed by the PEIC research team of the Politecnico di Torino. Once all the components were received, they were welded onto the boards following the schematic of the project [22].
Two welding phases have been addressed: reflow soldering of most components and manual tin soldering of the remaining components.
In the reflow soldering phase, a layer of solder paste was applied to the surface of the printed circuits. Subsequently, most of the components were positioned to their contact pads. After that, the entire assembly were subjected to controlled heat in a convection oven.
The goal of the reflow process is to allow the solder paste to reach the eutectic temperature at which the particular solder alloy undergoes a phase change to a liquid or molten state, demonstrating properties of adhesion.
Fig. 5.1 shows a photo of the convection oven used to fix the component on one of the two PCBs.

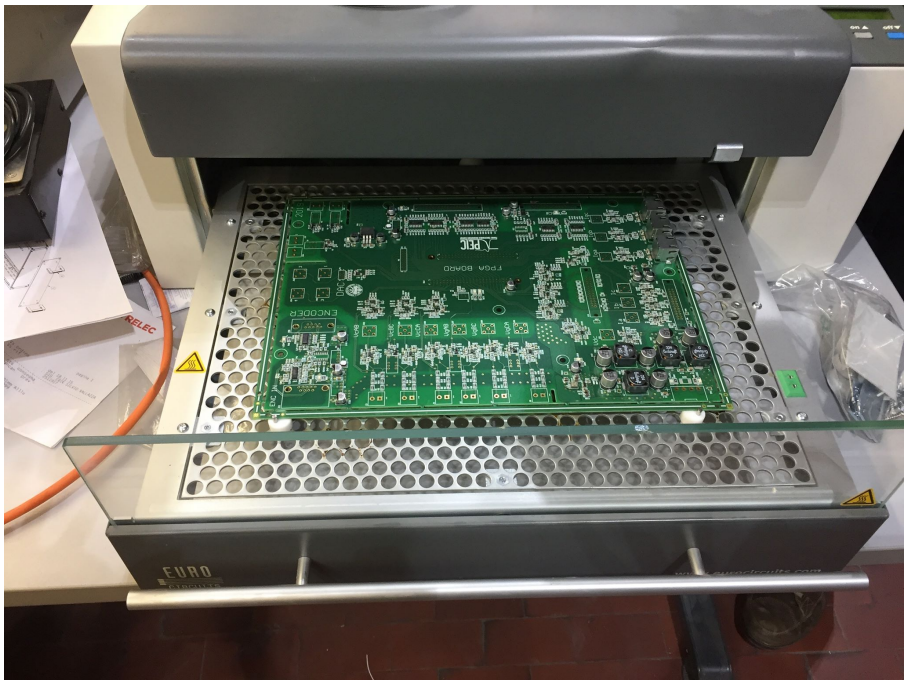


Figure 5.1: Reflow soldering convection oven.

A software called *eC−Reflow−Pilot* is used to set the temperature profile of the oven during the heat process. The profile is shown in Fig. 5.2.

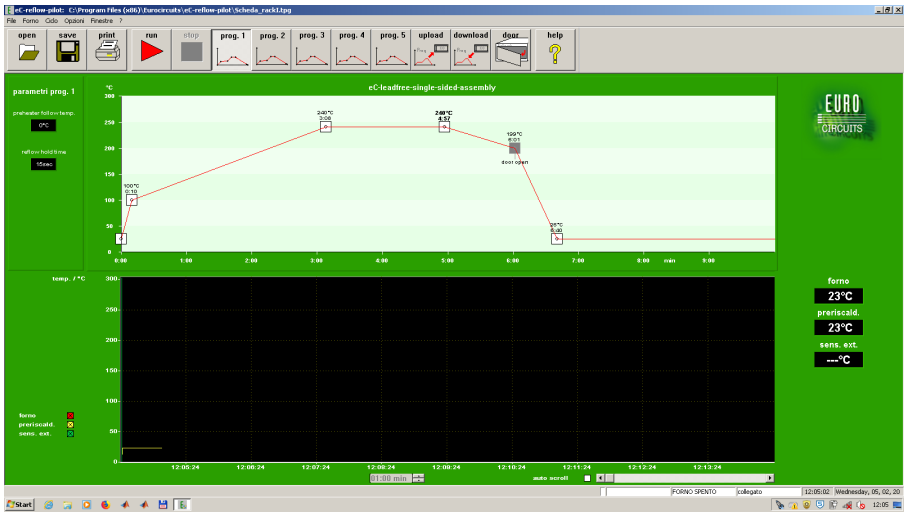


Figure 5.2: Temperature profile set for the heat process of the oven.

After the boards were cooled down, the remaining components were soldered manually. Fig. 5.3 shows a picture of the complete PCB board provided with a general description of the various component on it.

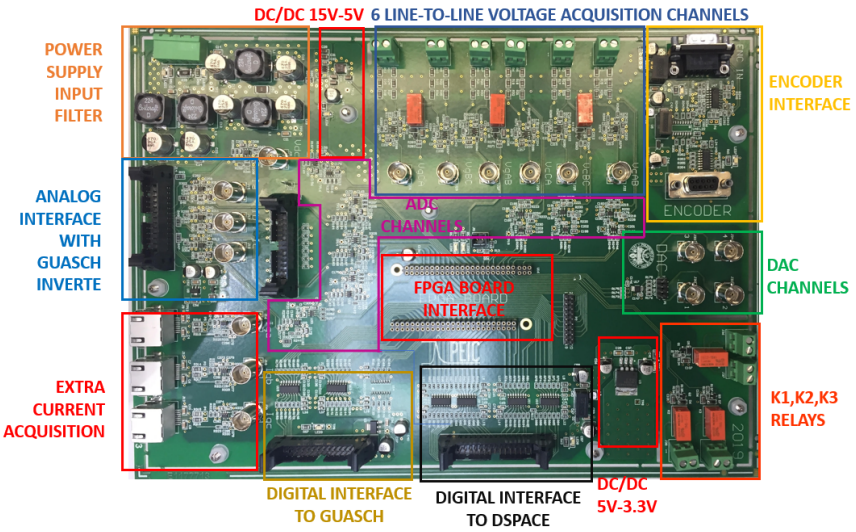


Figure 5.3: PCB.

In [22] is reported the interfaced board schematic where the pins layout of the

various components and the connection schemes between them are presented.

As it is possible to see, the board features:

- a power supply system and input filters;
- connectors to be interfaced with the power converter;
- analog conditioning circuits;
- local analog to digital converters(ADC);
- digital to analog converter (DAC);
- relays;
- leds;
- encoder interface;
- isolated digital communication with dSpace (12 bits in + 12 bits out) for dSpace communication (input and output);
- FPGA connectors;
- 20 header digital pins;

5.2 Test Bench Description

The test bench is composed by:

- PCB and FPGA board;
- A *Power Supply* to feed the board;
- An *oscilloscope* to visualize waveforms;
- *dSpace* platform in which is implemented the power converter control using Simulink;
- *ControlDesk*, which is a software used for debugging;

- *Vivado*, a software to program FPGA board;
- *Logic 1.2.18*, a software to visualize the information sent to the headers pins of the board.

A functional block diagram of the connection of the system is depicted in Fig. 5.4).

Figure 5.4: Test bench description.

The debugging software called *ControlDesk* is used to set the commands to send to the FPGA through dSpace pins. It is also used to set the reference value for the control system and also to be able to see the behaviour of the variable involved in the control.
ControlDesk is connected to a *Simulink* model that represents the system. The control code is written in a S-function.
DSpace reads this S-function implementing the control and redirecting the signals to the FPGA using a dedicated 12 bits line.
So, the FPGA receives informations from dSpace, processes them and send the answer to dSpace using a dedicated 12 bits digital line.

5.3 Preliminary Tests on FPGA Board

Once the assembly of the printed circuit board has been completed, the FPGA was connected and preliminary tests have been carried out to verify that all the components were working.

5.3.1 Power Supply Test

Obviously, the first test to do was to check the power supplies.
The board was powered by $\pm 15V$ direct voltage source.
Following the indications on the schematic in [22], all the points where it was

expected to record the potentials were verified using an oscilloscope. This process was facilitated thanks to the identification of particular test points (TP), inserted specifically during the design of the board to verify the functionality of the power supply report. Fig. 5.5 shows a picture of the test bench.

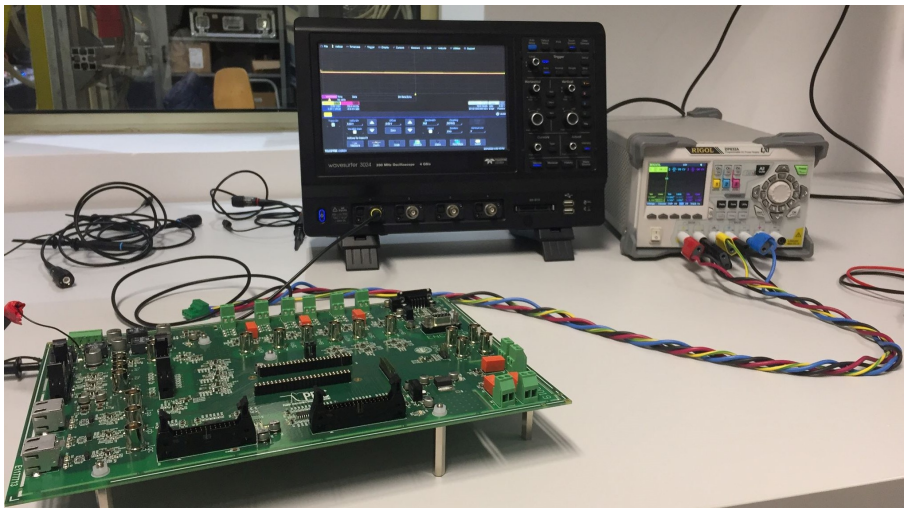


Figure 5.5: Power supply test bench.

5.3.2 Relays, Leds and Digital Output Pins Test

Each board is equipped with 6 relays: three are used for the power connection (Main relays, Pre-charge relays and V_{DC}, Grid-connection relay) and three are adopted for the characterization phase of the offsets for signal acquisition. These last three relays are commanded by the same signal.
In addition, there are also two LEDs on the board which will be used to encode visual information.
Moreover, the card is also equipped with 12 pins of digital outputs which can be used for communication to dSpace.
Relays, leds and digital output pins were tested programming the FPGA board using the software *Vivado*.
The block diagram of the project created for this test is shown in Fig. 5.6.

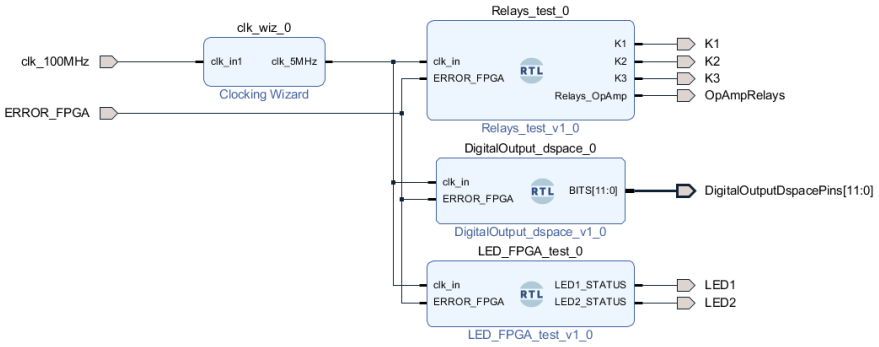


Figure 5.6: Test of relays, leds and digital output block diagram.

The clocking wizard block is responsible for generating a $5MHz$ clock output having $100MHz$ clock in input. This input clock is the one generated by the oscillator mounted on the board. Therefore, the clocking wizard block is a clock divider that allows to create in output different clocks at different frequencies according to necessities. The $5MHz$ clock will be sent to the management blocks of the relays, leds and digital outputs and will have the function of synchronizing the processes described in each of these blocks. As in Fig. 5.6, all the component management blocks have a signal called "ERROR FPGA". According to the logic implemented on the board, if this signal is high, then it communicates an error status to all the blocks and stops all processes, making the board safe. The error status is visually recognised because only one of the two LEDs is always on. The moment in which a reset fault signal arrives, the board exits the protection state and begins to perform the expected test.
The test consists in:

- commanding relays to open and close in sequence at a distance of one second.
- blinking LEDs at a frequency of 1 Hz;
- transferring a $2.5MHz$ square wave to each digital output pin. These waveforms will be visualized through an oscilloscope to verify proper functionality.

The behavioural simulation results are shown in Fig. 5.7.
Note that the $1Hz$ frequency of the blinking of the leds and the closing/opening of the relays has been increased to $1MHz$ during the behavioural simulation. This is to avoid long processing times of the simulation itself.

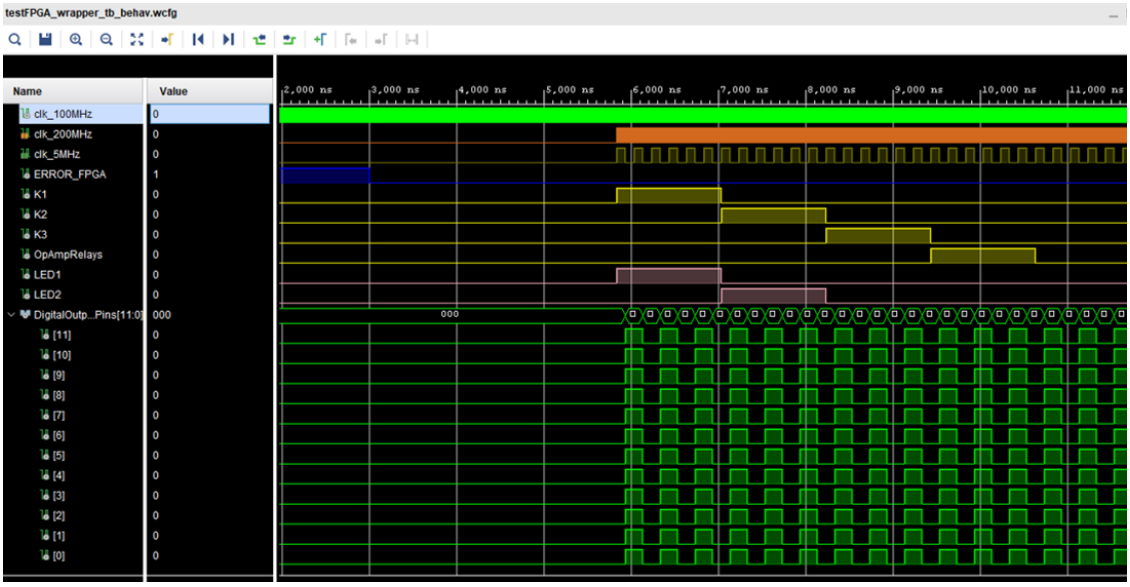


Figure 5.7: Test of relays, leds and digital output results obtained in behavioural simulations.

After the synthesis and implementation phase have been successfully completed, this project was subsequently loaded onto the FPGA board, implementing the correct blinking times of the LEDs and the opening/closing of the relays.
The experimental test gave the expected results. So, these components work well. In Fig. 5.8 is reported the waveform recorded of one of the digital output pins tested. All the others had the same behaviour.

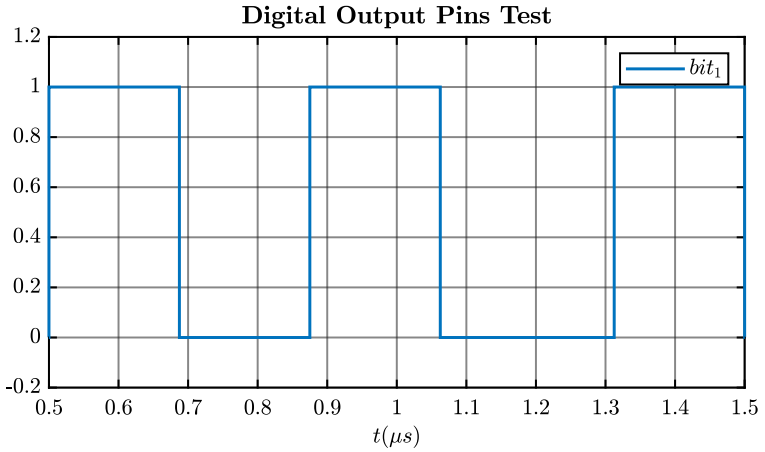


Figure 5.8: Digital output results obtained in experimental test.

5.3.3 Digital to Analog Converter Device Test

The board is equipped by a Digital to Analog Converter (DAC) device. It is *DAC124S085* 12-Bit. As, it can be read from its datasheet [27] it has 4 input data channel and it use a Serial Peripheral Interface (SPI) communication protocol. Some basic feature of this communication protocol are reported in .
A project is created to test the DAC device. The block diagram is presented in Fig. 5.9.

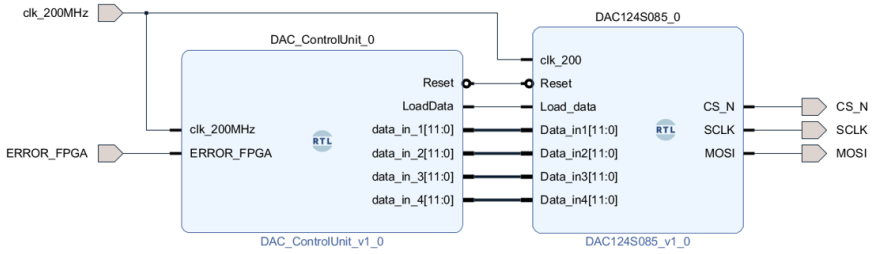


Figure 5.9: DAC test block diagram.

So, as it is shown in the figure, the DAC group is composed by two blocks synchronized by a $200MHz$ clock. These are:

- DAC Control Unit;
- DAC 124S085;

The DAC Control Unit generates a digitized sawtooth wave which represents the data that has to be acquired by the DAC channels. It also generates a $200kHz$ signal called "LOAD DATA". Every time this signal is lead high, the data are copied on the channel of the DAC device.
When DAC 124S085 block receives the "LOAD DATA" signal, it starts the transmission by lowering the bit CS. Then, a $20MHz$ SCLK is generated and every falling edge of it, a bit is copied to the MOSI line.

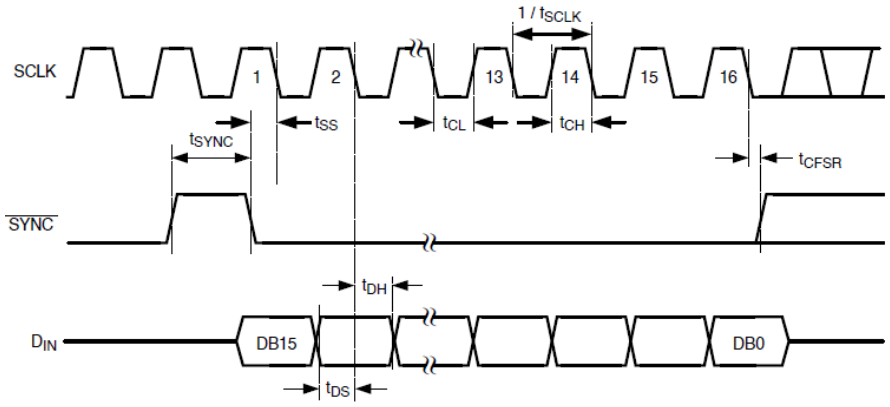


Figure 5.10: DAC serial timing diagram. Image taken from the datasheet [27].

The data of the 4 channels of the DAC are transmitted in sequence thanks to a special encoding. In fact, on the datasheet it is clarified that 16 bits are read from the input register and are transcribed on the MOSI. The four most significant bits decode the address of the channel from which they come and also the acquisition method. See the diagram shown in Fig. 5.11

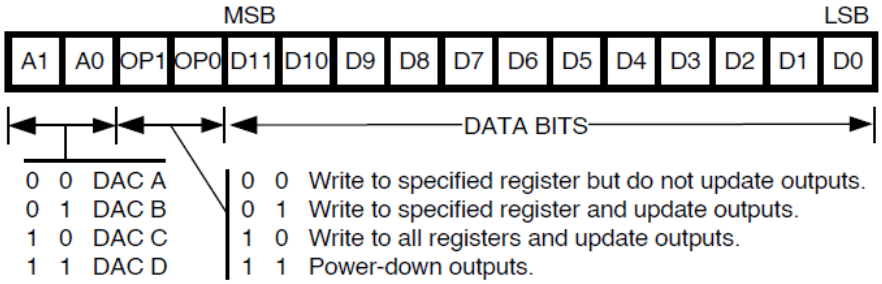
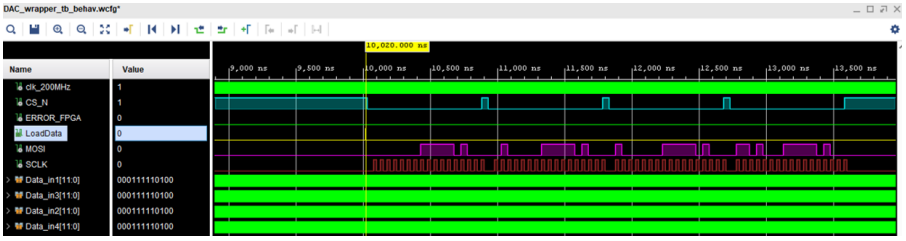
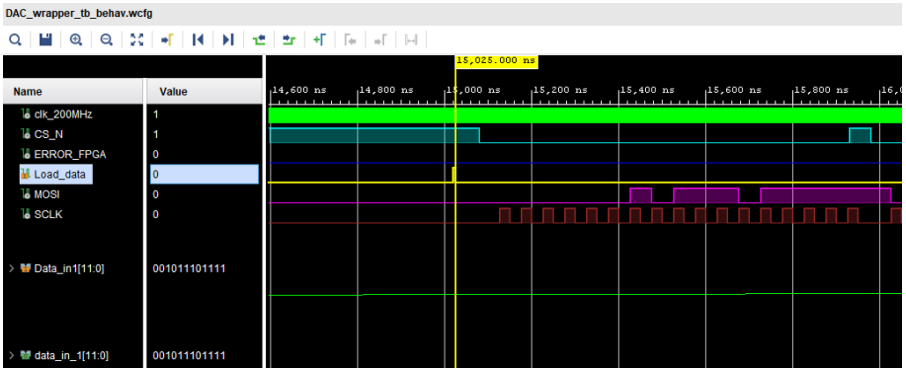


Figure 5.11: DAC Input Register Contents. Image taken from the datasheet [27].

The behavioural simulation is launched and the results are shown in Fig. 5.12.



(a) Complete DAC behavioural simulation results.



(b) Magnification of the first transmission results of DAC behavioural simulation.

Figure 5.12: DAC behavioural simulation results.

From the figure above, it can be seen the correct behaviour of the block described. In fact, when the chip selected is lowered, in correspondence with the falling edges of the SCLK, it is possible to check that the bits transmitted by the "DATA IN" vector are copied sequentially on the MOSI line. The "DATA IN" vector also contains the 4 bits referring to the encoding sequence described before.

5.3.4 Analog to Digital Converter Device Test

An analog to digital converter (ADC) is a system that converts an analog signal into a digital signal. The PCB is provided of an ADC device. It is the *AD7276* with 12 bit of resolution. The component is fed by a single $3.3V$ power supply and features throughput rates of up to 3 MSPS [4]. The device will be used for acquisitions of signal, such as measured currents or measured voltages.
A VHDL project was created to test this component using Vivado. The block diagram is shown in Fig. 5.13.

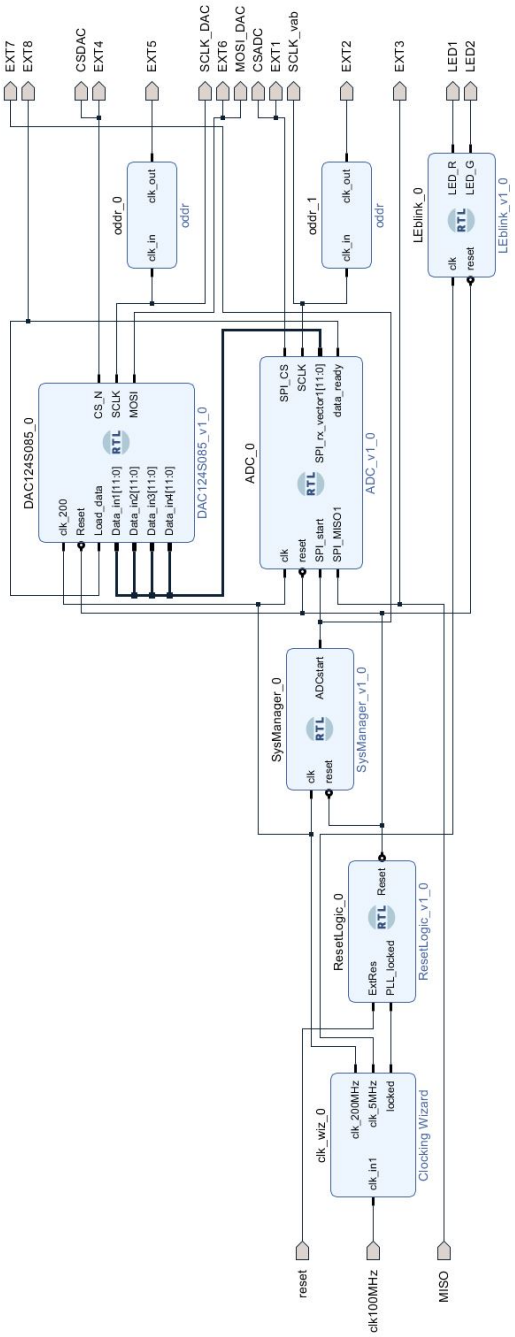


Figure 5.13: ADC test block diagram.

As it is visible in Fig. 5.13, the project is provided with the following macro-blocks:

- clocking wizard, that generated the clocks needed for synchronized processes of the other blocks;

- ADC manager, which gives in output a $1MHz$ signal called "trigger start" used to lead the acquisition process of the ADC block;

- ADC block, that starts the acquisition of the data when receives the "trigger start" signal. It reads the bits from the MISO line one by one, saving them in a response buffer vector. The process is synchronized with the serial clock SCLK;

- DAC block, that receives on the four channels the data from the ADC blocks and then transmitted it in output in the MOSI line. An oscilloscope is used to visualize the output waveforms. Because the data is acquired by the ADC block and then is given in output from the DAC device, the waveforms on the oscilloscope must represent the same data given in input to the ADC block on the MISO line, considering the appropriate scale factors. So, the DAC component is used to verify the correct functionality of the ADC device;

- Reset State Management. This macro-block is composed by two logic blocks. It manages the reset signal. In fact, if an error occur, the data transmissions are interrupted, and the system is put in a safe state. To exit from this state a reset signal is necessary. The reset signal is produced in output if the user push the emergency push button or if the clocking wizard block is still not generating the output clock, which are essential for the good functionality of the logic implemented for the FPGA board.

Therefore, when the "trigger start" signal is received, the ADC block lowers the bit CS and starts to generate the SCLK.
According to the datasheet [4], every rising edge of SCLK a bit is read from the MISO line and it is inserted in a 16-bit buffer vector.
In fact, on page 18 of the datasheet it can be read that "for the AD7276, a minimum of 14 serial clock cycles are required to complete the conversion and access the complete conversion result".
Therefore, it was decided that the block would generate 16 hits of SCLK at $48MHz$.
At the end of the transmission, the response buffer is filled as shown in Fig. 5.14.

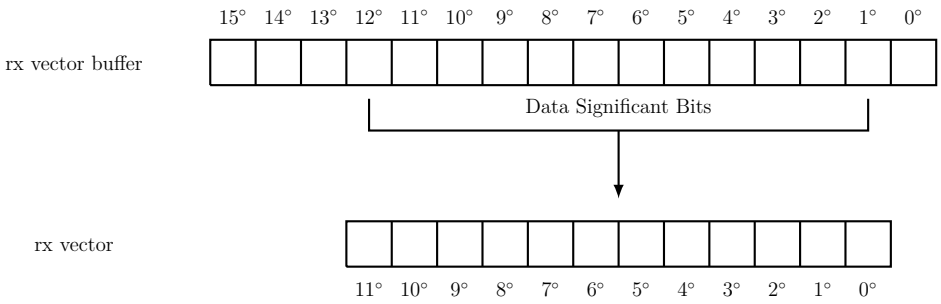
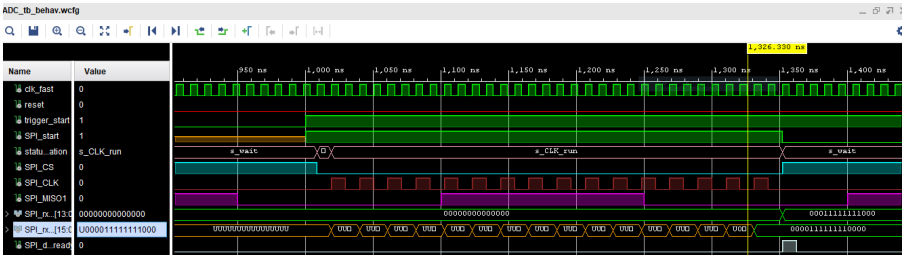
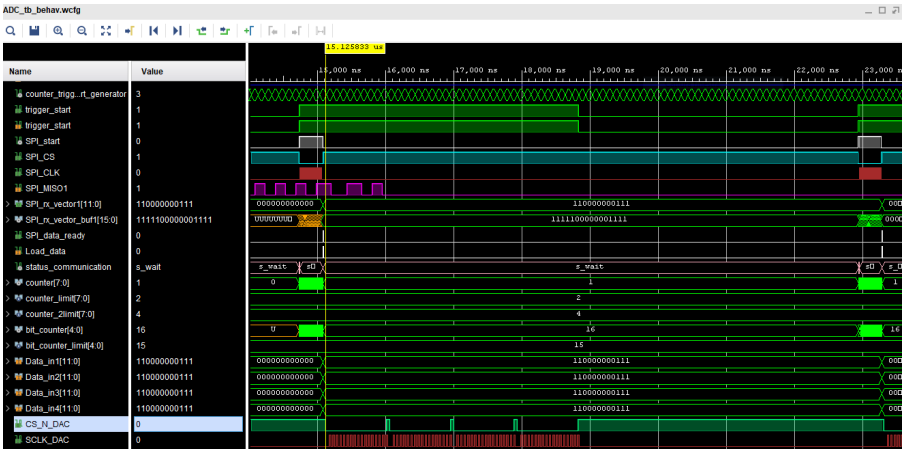


Figure 5.14: ADC Data Output Vector.

The real data transmitted is contained only in 12 bits of the 16 filled in the response buffer and, at the end of transmission, the 12 significant bits will be copied to a 12 bit output vector.
A flag called "data ready" signals the end of the transmission.
This signal is acquired by the DAC block which begins reading from its acquisition channels.
Note that the acquisition of the data from ADC block must be completed in order to begin the transcription of the it on DAC block the MOSI line.
A behavioural simulation was launched and the results are shown in Fig. 5.15.



(a) ADC acquisition process in behavioural simulation results.



(b) Complete ADC and DAC transmission in behavioural simulation results test.

Figure 5.15: ADC Behavioural Simulation results.

Fig. 5.15a shows one complete acquisition of ADC block. It can be seen that when the signal "trigger start" arrives, the acquisition process begins and every rising edge of SCLK a bit is read from the MISO line. After 16 hits of SCLK, the transmission is completed and the data 12 bits are copied into an output vector, which is send to the input channels of DAC device. In that moment, also the flag of "Data Ready" is activated. Fig. 5.15b shows the operation of the DAC block: it receives the data for its four channels and then it transmits them sequentially on the MOSI line.
After the validation in simulation, the code was synthetized and implemented. The same test was then performed on the actual hardware.
Using the waveform generator a $50Hz$ sinusoidal signal was sent to one of the acquisition channel of the board. The data was written on the MISO line of the

ADC. After that, according to the scheme in Fig. 5.13, it was sent to a DAC channel to be visualised on the oscilloscope. In Fig. 5.16 in pink is reported the input waveform of the ADC converter and in green the output waveform of the DAC. As it can be seen, the two waveforms match (given the proper scaling), therefore validating the analog acquisitions.

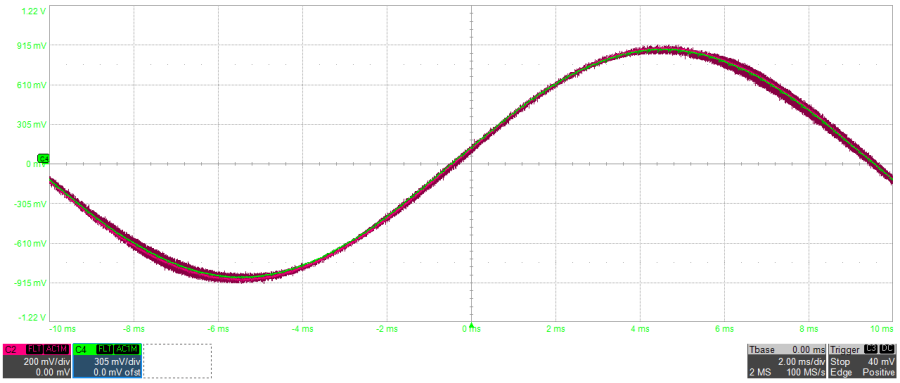
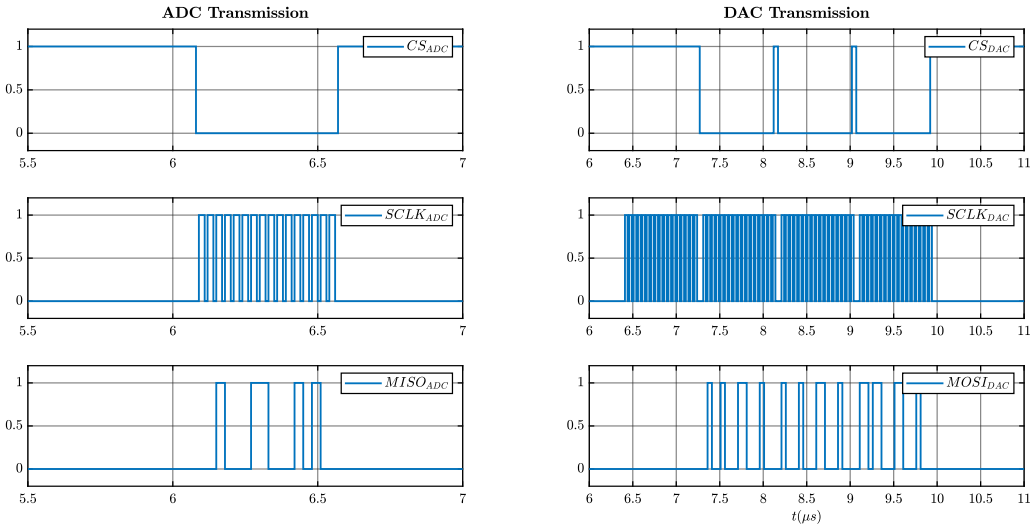


Figure 5.16: ADC and DAC experimental results from oscilloscope.

In Fig. 5.17 is presented one of ADC and DAC transmissions recorded during the test. Both transmissions comply with the datasheet specifications and given working parameters (frequency...).



(a) ADC transmission experimental results. (b) DAC transmission experimental results.

Figure 5.17: ADC and DAC experimental results obtained.

5.4 FPGA Board

The role of the board equipped with FPGA is to interface the control to the real component. Basically, the FPGA:

- must receive informations from dSpace, from the acquisition system on the board and from the inverter sensors;
- check that there are no faults. If a fault is detected, it has to ensure a safe state of the system;
- must execute the commands coming from dSpace and redirect the gate signals to the converter.

It has been decided to build a block that implement a final state machine to allow the FPGA to manage the different situations in a clear way.
A final complete project was created in Vivado and its block diagram is shown in Fig. 5.18.

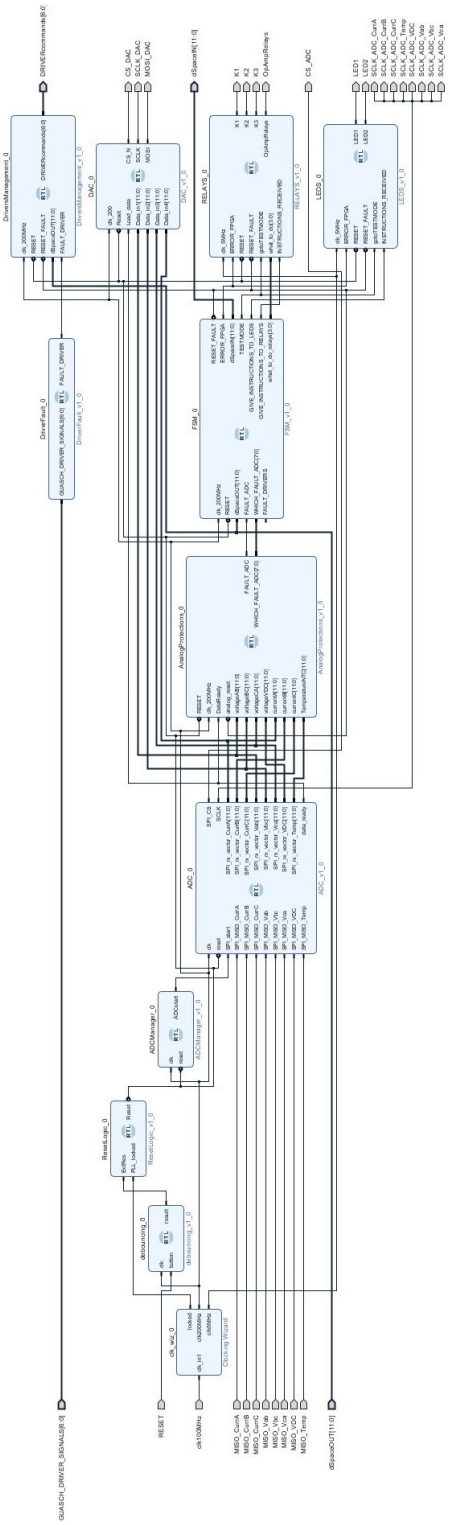


Figure 5.18: Final Project block diagram.

The diagram presents several blocks. They are:

- **clocking wizard**, which receives the input clock from on board oscillator and generates two different output clocks to synchronize the various processes.

- **ResetLogic block**, which reset all the blocks when the user presses the push button or if the system clock are still not generated by the clocking wizard block.

- **FSM block**, that is the core of the project and implements the final state machine.

- **ADC block**, which reads the measured data from its 8 input channels.

- **Analog Protections block**, that receives the data from the ADC and verifies if there is no anomaly in the measurements. In fact, it checks if the current, the voltage and the temperature measured values do not exceed the limits. If it occurs, an error flag is sent to FSM.

- **Driver management and Fault driver blocks**, which receive the gate drivers of the inverter and give informations to the FSM if a fault occur in one of them.

- **Relays and Leds blocks**, that receive the commands from the FSM redirecting them appropriately to the leds and relays on the board.

Here a detailed functional description of the main blocks will be given. Since the ADC and DAC clocks were already explained in subsection 5.3.3 and subsection 5.3.4, they will be here omitted. However, it is needed to say that in the ADC block must transmit the data that come from the acquisition system to the Analog protection blocks. The sampled analog quantities are the three line-to-line voltages measured at the PCC and the three phase output converter current. Also a measurement of the DC-link voltage is performed. Instead, the DAC is used to visualize 4 of these measurement using an oscilloscope.

5.4.1 Analog Protections

As written before, since the board is equipped with an acquisition system, it is reasonable to implement a block that checks the acquired measurements. A magnification of the block is illustrated in Fig. 5.19.

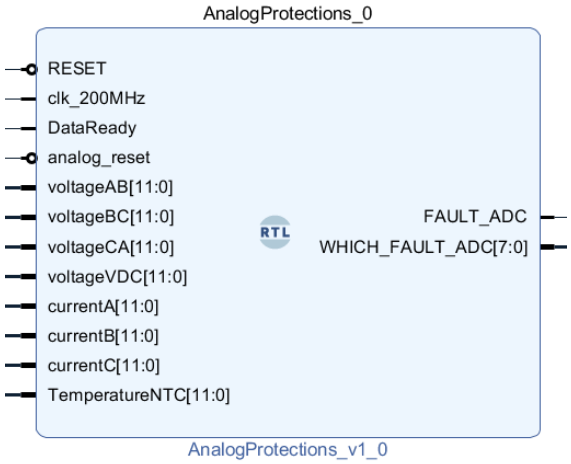


Figure 5.19: Analog protections block.

This block receives the data acquired by the ADC and verifies that they don't exceed their limit values. The logic implemented requires that if one of the signals exceed the threshold for three times in a row, an error message is sent to the FSM block and the FPGA is lead to the error state.

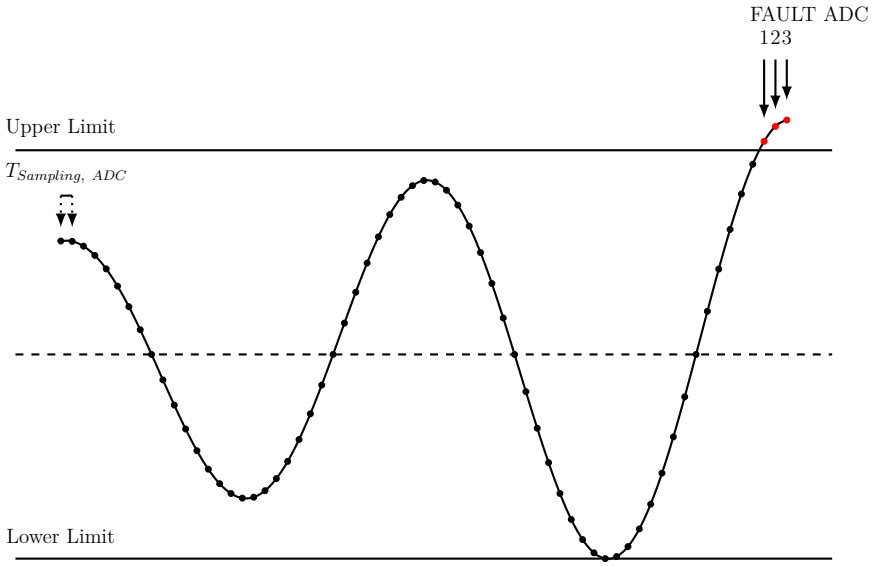


Figure 5.20: Example of detection of an anomalous value from the block of analog protections.

The VHDL code to implement this block is written in section 7.8.
For each measurement channel, a flag is created that indicates if a abnormal value is detected for three times in a row or not. If an error occurs the block sets the bit of "FAULT ADC" high and in the vector called "WHICH FAULT ADC" is written which type of anomaly is detected (overvoltage /overcurrent...).
The block functionality was validated programming the FPGA board with a simpler scheme outlined in Fig. 5.21.

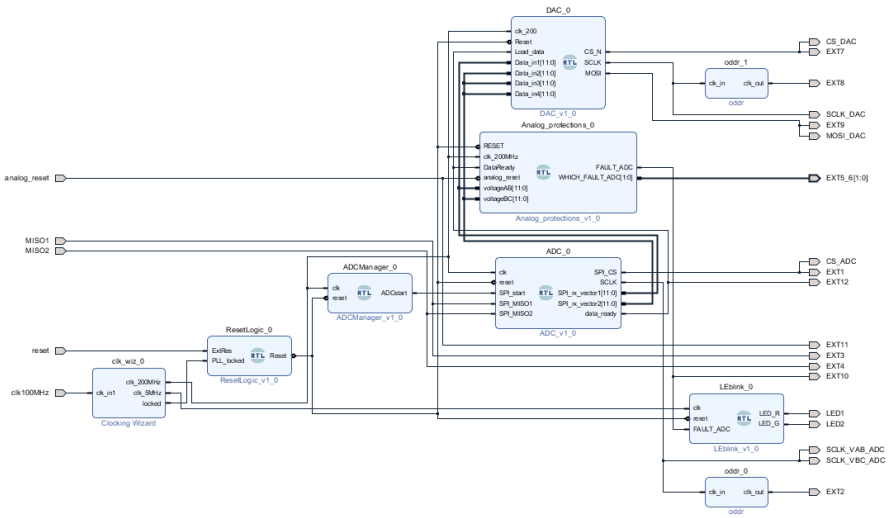
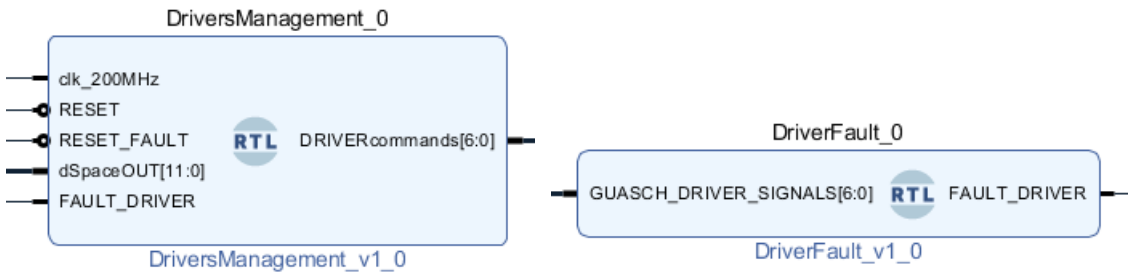


Figure 5.21: Analog protection test block diagram.

The simplest program provides this block only of two measurements. Two signal were created using a waveform generator. They were sent to the acquisition system of the board, that transmits the conditioned signals to the ADC input channel. After that, the signals were also redirect to the DAC so they can be visualized using an oscilloscope. When a signal exceeds the limit imposed, the FPGA indicates the error state and lead the converter to a safe state.

5.4.2 Management of the Driver Signals

This converter can signal the status of the gate drivers. Fault occurring to the switches (e.g. short circuit) are immediately reported to the FPGA, which stops the operation of the system and reaches a safe state. If a fault is detected in a driver, the block called "DriversFault" (Fig. 5.22b) sends to the FSM an error message.
As it can be read in the inverter datasheet [10], the seven fault signals are active low.
If no fault is detected, the block "DriversManagement" (Fig. 5.22a) redirects the gate commands received by the control implemented in dSpace to the gate drivers.
The two blocks used in the project to manage the driver signals are reported in Fig. 5.22.



(a) Driver Management block. (b) Drivers Fault block.

Figure 5.22: Drivers Management and Fault blocks.

The signals coming from the supervisor system of the inverter are written in "GUASCH DRIVER SIGNALS" vector, while the commands of the gate generated by dSpace controller are written in the first seven bits of the "dSpaceOUT" vector (see subsection 5.4.3).
The VHDL code of these two blocks are reported in subsection 7.9.1 and subsection 7.9.2.

5.4.3 DSpace Communication Protocol

The control algorithm of the inverter is implemented on the dSpace platform. DSpace communicates with FPGA board using 12 bits and also the FPGA sends to dSpace informations using other 12 bits. So, it is necessary to assign to each bit a specific meaning in the communication protocol. The vector that transmits data from dSpace to the FPGA is called "dSpaceOUT", while the vector of bits that transmits informations from FPGA to dSpace is called "dSpaceIN". In Table 5.1

DSpace Communication Protocol			
dSpaceOUT bits	**meaning**	**dSpaceIN bits**	**meaning**
0	T1	0	Overcurrent phase a
1	T2	1	Overcurrent phase b
2	T3	2	Overcurrent phase c
3	T4	3	Overvoltage DC
4	T5	4	Overvoltage ab
5	T6	5	Overvoltage bc
6	T7	6	Overvoltage ca
7	EnablePWM	7	Overtemperature
8	Awake	8	Fault Drivers
9	commands(0)	9	EnablePWM
10	commands(1)	10	Emergency Push Button
11	commands(2)	11	Errorstate

Table 5.1: DSpace Communication Protocol.

So, according to Table 5.1 the first seven bits of dSpaceOUT vector give the commands to redirect to the gate drivers [10]. Moreover, dSpace informs FPGA about the state of the EnablePWM signal, that indicates if the modulation is enabled or not. Furthermore, the Awake signal is sent to FPGA. This is a square wave signal which indicates that the dSpace board is powered on. In fact, if dSpace is off this signal is not generated and the FPGA must keep the system in an error state.

Finally, the last three bits are combined to create $2^3 = 8$ different commands to send to the FPGA. These commands are summarized in the following table:

DSpace Commands to FPGA				
BIT 9	**BIT 10**	**BIT 11**	**DSpace Commands**	**meaning**
0	0	0	Ext Fault	Reset commands to exit from the errorstate.
0	0	1	Test Mode	Execute the test of the relays and leds.
0	1	0	Error	Leads the FPGA to the error state.
1	0	0	K1	Change the state of the relay K1.
0	1	1	K2	Change the state of the relay K2.
1	0	1	K3	Change the state of the relay K3.
1	1	0	OpAmpRelays	Change the state of the relays used to calculate the offset value of the acquisition system.
1	1	1	GOGO	Leads the FPGA to the operative state.

Table 5.2: DSpace Commands to FPGA board.

To better understand which relays is commanded, see the equivalent circuit illustrated in Fig. 4.1 or the schematic of the board [22]. Moreover, the state of the finite state machine will be described in detail in subsection 5.4.4.

5.4.4 Finite State Machine

The FSM block implements the finite state machine managing the state of the FPGA board. It is the core of all the project because it interfaces together each block of the system. The final state machine is organized into four states. These are:

1. ERROR;
2. RESET STATE;
3. TEST MODE STATE;
4. GO.

In Fig. 5.23 is shown a magnification of the FSM block.

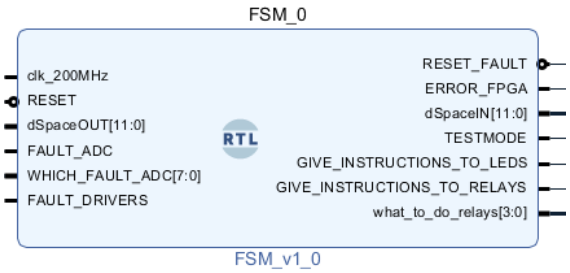


Figure 5.23: FSM block.

The block receives in input:

- the clock at $200MHz$, which synchronizes the process of instructions.
- the global RESET signal which comes from the *ResetLogic* block. This signal leads the final state machine to the reset state.
- commands from dSpace written in *dSpaceOUT* vector (see subsection 5.4.3).
- Error flag from the *AnalogProtections* and *DriversFault* blocks.
- the vector *WHICH FAULT ADC* (subsection 5.4.1).

The FSM block processes these informations and it gives in output:

- "ERROR FPGA" flag, which is high when a fault is detected. This signal is sent to all the other block to command them into the ERROR state.
- "RESET FAULT" signal, that is used to reset the condition of FPGA when a fault occurs.
- "TESTMODE" flag, which commands to execute a TEST of the relays and leds functionality.
- "GIVE INSTRUCTION TO LEDS" and "GIVE INSTRUCTION TO RELAYS". If they are set high, the LEDS and RELAYS block can execute the instructions written in their operative state. In fact, in this case, the leds blink and the relays execute the instructions given by dSpace written in "what to do relays" vector.

It is necessary to describe the behaviour of the *FSM* block.
First the FSM translates the command bits received by dSpace and checks if the global RESET flag is arrived.
After that, the FSM checks that no error message is arrived from the other block o from dSpace.
The default state is the "RESET STATE", which is indicated visually by the simultaneous flashing of the two LEDs. In this state, the FSM block leads the FPGA in a safe operation, opening all the relays and other power switches.
If an error message arrives from the *Analog Protections* or *Driver fault* block or if the dSpace platform is switched off the *ERROR FLAG* is activated (Fig. 5.24).

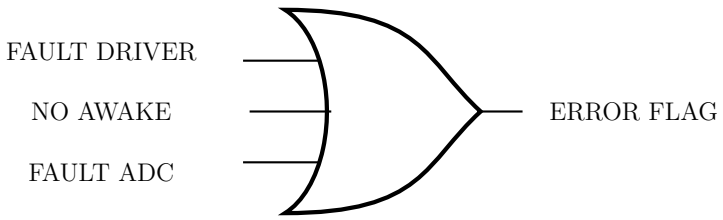


Figure 5.24: Error Flag Management.

If this flag is high, the FSM goes to "ERROR STATE", which is visually recognizable by the fact that one of the two LEDs remains on and the other remains off.

In this state, the relays are open and the drivers are not commanded. Moreover, the FSM sends to dSpace an error message indicating the error code. In this way it is easier to identify and solve the problem more quickly.
Instead, if no error is detected, if the user wants to make a test to verify the correct functionality of the system, from ControlDesk they must write the correct sequence of the last three dSpaceOUT vector (see subsection 5.4.3). In this way, the FSM is lead to the TEST MODE STATE processing the test routine described in subsection 5.3.2.
Otherwise, if the user wants to start the control routine, they have to write the correct sequence to send the message of "GO" to the FPGA. If the FSM receives this message and no error is detected, the FSM goes to its operative state called "GO", in which the instruction flag to RELAYS and LEDS block are sent and the commands that comes from dSpace are redirected to the relays. In this state, also the gate drivers are commanded.
So, in this state the FPGA performs its task of interface between control and power components. A flow chart of the logic implemented in this blocks it shown in Fig. 5.25.

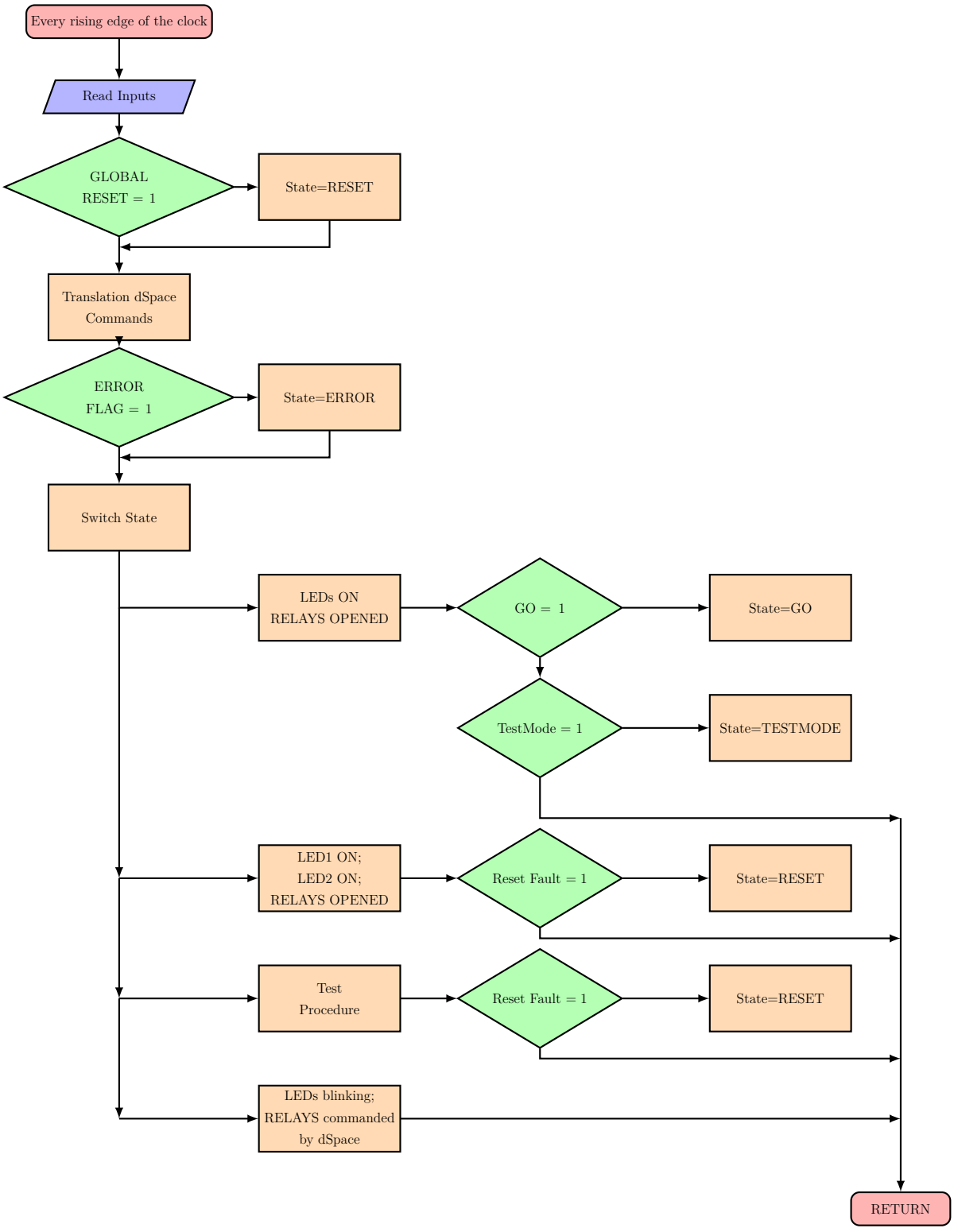
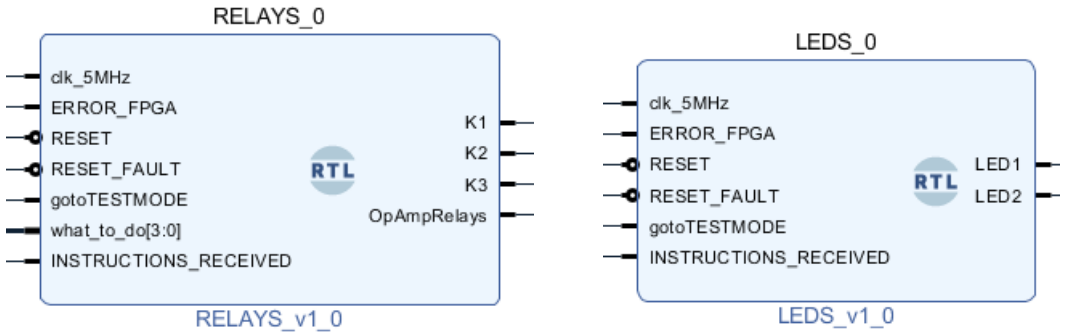


Figure 5.25: Flowchart of the logic implemented for FSM block.

The VHDL code of the FSM block is reported in section 7.10.

5.4.5 LEDS and RELAYS Blocks

The magnifications of two LEDS and RELAYS blocks are reported in Fig. 5.26.



(a) Relays block. (b) LEDS block.

Figure 5.26: LEDS and RELAYS blocks.

The logic implemented is the same for these two blocks. In fact, their logic is divided in 4 states:

- ERROR;
- RESET STATE;
- TEST MODE STATE;
- DO.

Each state corresponds to one of the state of the FSM block (subsection 5.4.4). Basically, Leds are used as visual markers for FSM states. So,

- in the "ERROR" state only one of two leds remains on and all the relays are open, the ERROR state is ;
- during the "RESET STATE" the relays are open and the leds remains both on;

- in the "TEST MODE STATE" the same test procedure described in subsection 5.3.2 is implemented;
- during the "DO" state, the leds blinks and the relays execute the commands that comes from dSpace and are redirected to them by the FSM block.

The VHDL code of the two blocks are written in section 7.11 and section 7.12.

5.4.6 DSpace Communication Pins Test and Validation of Finite State Machine Logic in Real Setup

On the PCB boards there are 12 pins dedicated for the acquisition of informations from dSpace and other 12 pins where FPGA writes informations to send to dSpace. These pins forms the communication system between FPGA and dSpace and they were tested to ensure the correct functionality. The communication protocol is described in subsection 5.4.3.

Therefore, to test the correct functionality of the communication system and to make sure that the FSM block correctly interpretates and manages the informations, a simplify project was created in Vivado. The block diagram is shown in Fig. 5.27

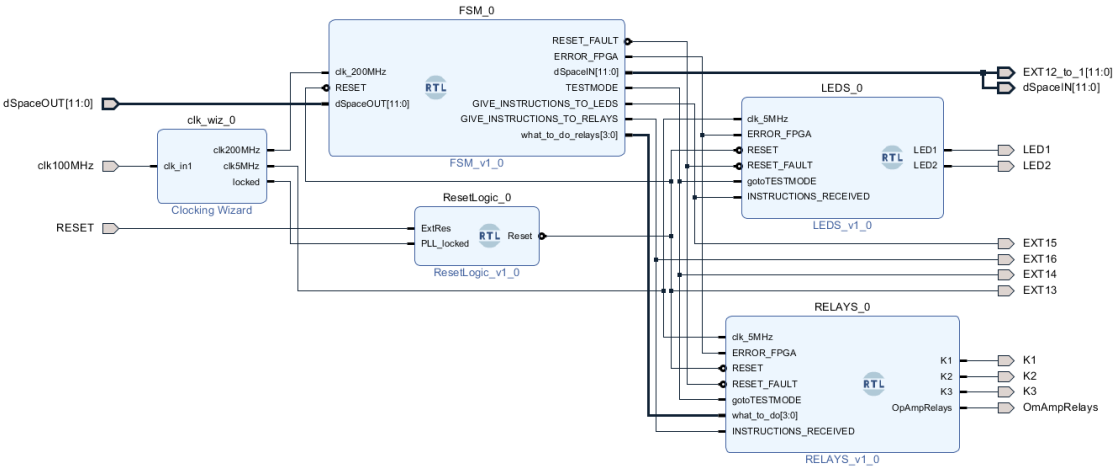


Figure 5.27: DSpace Communication test block diagram.

Comparing the block diagram in Fig. 5.27 and the one in Fig. 5.18, the project is simplified because not all the blocks are involved. However, as said before, it was chosen to test first the correct functionality of FSM block and of the dSpace communication pins before proceeding with the entire complete final project.

So, first a test bench was created in which was simulated the data transmission from dSpace. The FPGA correctly interpreted the commands sent by dSpace. Fig. 5.28 shows the results obtained from the behavioural simulation.

Figure 5.28: DSpace Communication behavioural simulation results.

From the results obtained during simulations, it is possible to say that the logic implemented works as expected.
After that, the synthesis and the implementation processes were performed. So, in the end the logic was loaded into FPGA boards and functionality of the FSM block and of the dSpace Communication system were tested in the setup. The experimental tests have given the expected results.

Chapter 6

Conclusions

This thesis focused on the study of converters for microgrid. Both grid-forming and grid-supporting control strategies have been analysed. The grid-forming control is a low-level control that sets the voltage and frequency reference values at PCC of the microgrid. Two different grid-forming control strategies were studied: the single loop voltage control and the dual loop control. The power converter that was considered is a three-phase two level inverter. The control was implemented both in the (α, β) stationary reference frame, adopting a PRES regulator, and in the (d,q) rotating frame synchronous with the reference frequency (f^*), using a PI regulator. In addition, other analyses were made simulating in PLECS the system behaviour, controlling the inverter unit with dual loop control strategy implemented in rotating reference frame using PI regulators: the comparison between the results obtained using the PWM BEM and the DPWM techniques, the effect of the implementation of the dead-time compensation algorithm and the comparison between different damping circuit topologies for the LC-output converter filter.
Then, these control structures were implemented in C code. The system behaviour was first simulated in PLECS and then it was implemented on a real setup. Experimental tests validated the simulated behavior. The experimental tests validated both control strategies with RL loads and active loads (diode rectifier). Subsequently, a grid-supporting control was designed and simulated in PLECS. It implements an external power loop that sets the reference frequency at the PCC according to the active power requirement at the PCC setted by a supervisory

control layer of the microgrid. Subsequently, a converter prototype was designed. This design involved: the mechanical structure (Solidworks), the firmware implementation on an interface FPGA board and the converter unit assembly. First, all the components of the interface board were soldered and the FPGA board programmed using the *Vivado* environment. Then, the grid-forming control strategy was implemented on a dSPACE rapid control prototyping system and the interface board was programmed to redirect the commands from dSpace to the converter unit. It was necessary to define a communication protocol between dSpace and the FPGA in order to interface the two parts. Finally, experimental tests were made to validate the entire system functionality.

Chapter 7

Appendix

7.1 Dead-Time Compensation Algorithm C Code

```
...

DeltaV.a = TDT*FSW*Vdc;
DeltaV.b = TDT*FSW*Vdc;
DeltaV.c = TDT*FSW*Vdc;

v_controlABC_DTC = v_controlABC;

 //Hysteresis Control

if (IlABC.a > I_hysteresis_high) v_controlABC_DTC.a = v_controlABC.a + DeltaV.a ;
if (IlABC.a < I_hysteresis_low) v_controlABC_DTC.a = v_controlABC.a - DeltaV.a ;


if (IlABC.b > I_hysteresis_high) v_controlABC_DTC.b = v_controlABC.b + DeltaV.b;
if (IlABC.b < I_hysteresis_low)    v_controlABC_DTC.b = v_controlABC.b - DeltaV.b;

```

```
18 if (IlABC.c > I_hysteresis_high)  v_controlABC_DTC.c = v_controlABC.c
     + DeltaV.c;
19 if (IlABC.c < I_hysteresis_low)   v_controlABC_DTC.c = v_controlABC.c
     - DeltaV.c;
20
21 ...
```

Listing 7.1: Dead Time Compensation Algorithm C Code.

7.2 PWM BEM C Code

```
1  void PWMduty(Xabc vsabc_ref, float vdc, Xabc* duty_abc) {
2
3    float tmp1, tmp2, tmp3, vdc_inv, vzs;
4
5    if (vdc > 0) vdc_inv = 1.0f/vdc;  // avoid division by zero
6
7    // zero-sequence detection:
8    // tmp1 = max,  tmp2 = min
9    // tmp3 is the phase in between: tmp3 = -(max + min)
10
11   tmp1 = vsabc_ref.a; // max
12   tmp2 = vsabc_ref.b; // min
13
14   if (vsabc_ref.a < vsabc_ref.b) {
15     tmp1 = vsabc_ref.b;    // max
16     tmp2 = vsabc_ref.a;    // min
17   }
18
19   if (tmp1 < vsabc_ref.c) tmp3 = tmp1;
20   else {
21       if (tmp2>vsabc_ref.c) tmp3 = tmp2;
22       else  tmp3 = vsabc_ref.c;
23      }
24
25   vzs = tmp3*0.5f;  // zero sequence voltage
26
27   // Duty-cycles, with zero-sequence
```

```
28   duty_abc->a = 0.5f + (vsabc_ref.a +vzs )*vdc_inv;
29   duty_abc->b = 0.5f + (vsabc_ref.b + vzs )*vdc_inv;
30   duty_abc->c = 0.5f + (vsabc_ref.c +vzs )*vdc_inv;
31
32    //duty cycles saturation
33     if ( Duty->a > MAXDUTY)   Duty->a = MAXDUTY;
34     if ( Duty->b > MAXDUTY)   Duty->b = MAXDUTY;
35     if ( Duty->c > MAXDUTY)   Duty->c = MAXDUTY;
36
37     if ( Duty->a < MINDUTY)   Duty->a = MINDUTY;
38     if ( Duty->b < MINDUTY)   Duty->b = MINDUTY;
39     if ( Duty->c < MINDUTY)   Duty->c = MINDUTY;
40
41 }
```

Listing 7.2: PWM BEM C code.

7.3 DPWM C Code

```
1  void DPWMCompute(Xabc ABC, Xabc *Duty, float Vdc , float *
       pwm_zero_sequence)
2
3  {
4
5      float a,b,c,x1,y1,pwm_zero_seq,pwm_zero_seq_mod;
6
7      x1 = 1.0/(Vdc);
8
9      a = ABC.a*x1;
10     b = ABC.b*x1;
11     c = ABC.c*x1;
12
13     y1 = a;
14
15     if(sgn(a)==sgn(b)||sgn(a)==sgn(c))
16     {
17         y1 = b;
18         if(sgn(b)==sgn(a)||sgn(b)==sgn(c))
```

```
            y1 = c;
    }

    pwm_zero_seq = sgn(y1)-2.0*y1;

    Duty->a = 0.5+a+0.5*pwm_zero_seq;
    Duty->b = 0.5+b+0.5*pwm_zero_seq;
  Duty->c = 0.5+c+0.5*pwm_zero_seq;

    *pwm_zero_sequence = pwm_zero_seq;

    //duty cycles saturation
    if ( Duty->a > MAXDUTY)   Duty->a = MAXDUTY;
    if ( Duty->b > MAXDUTY)   Duty->b = MAXDUTY;
    if ( Duty->c > MAXDUTY)   Duty->c = MAXDUTY;

    if ( Duty->a < MINDUTY)   Duty->a = MINDUTY;
    if ( Duty->b < MINDUTY)   Duty->b = MINDUTY;
    if ( Duty->c < MINDUTY)   Duty->c = MINDUTY;

}
```

Listing 7.3: DPWM1 C code.

7.4 Grid-Forming Control C Code

```
 ...
//Clarke Transformation of measured AC quantities
  _clarke(IlABC,IlAB);
  _clarkeLL(VcABC,VcAB);

  //thetaref calculation for RST side
  thetarefcalculcation(&thetaref, &wref, &SinCos_thetaref);

  /*=========================================================*/
  /*          Direct rotation transformation (alphabeta-dq)  */
  /*=========================================================*/
  _rot(VcAB,SinCos_thetaref,VcDQ);
```

```
13
14
15 /*
   ==========================================================================
   */
16 /*                        Control Code  lato ABC (DC/DC)
         */
17 /*
   ==========================================================================
   */
18    switch(StateControlABC)
19 {
20       case HALT:
21       Vdc_ref = 400.0;
22           EnablePWM_ABC=1.0;  //PWM disabled for the DC/DC power
   converter
23       Vdc_var.intg = 0.0;
24       Ildc_var.intg = 0.0;
25       if (flag_AD_ready<0.5)
26       {
27         if (RunABC>0.5)
28           {
29           VrampDCDC = Vdc_measured;
30           DeltaV_DCDC = 100.0*TS;
31           StateControlABC=RUNDCDC;
32           }
33       }
34       break;
35        case RUNDCDC:
36           EnablePWM_ABC=0.0; //PWM enable for the DC/DC power
   converter
37
38       Vend_rampDCDC = Vdc_ref;
39       VrampDCDC+=DeltaV_DCDC;
40
41       if(VrampDCDC>=Vend_rampDCDC)
42         VrampDCDC=Vend_rampDCDC;
43
44       if(Vdc_measured<0.1)
```

```
45          Vdc_measured=0.1;
46
47      //voltage loop
48        Vdc_var.ref=VrampDCDC;
49              Vdc_var.fbk=Vdc_measured;
50        Vdc_par.kp = kp_v_DC;
51        Vdc_par.ki = ki_v_DC;
52              Vdc_par.lim = Ildc_lim;
53              PIReg(&Vdc_par, &Vdc_var);
54              Ildc_ref=-Vdc_var.out;
55
56      //current loop
57        Ildc_var.ref=Ildc_ref;
58              Ildc_var.fbk=Ildc_measured;
59              Ildc_var.ffw=Vin_measured;
60        Ildc_par.kp = kp_i_DC;
61        Ildc_par.ki = ki_i_DC;
62              Ildc_par.lim=Vdc_measured;
63              PIReg(&Ildc_par, &Ildc_var);
64              v_control_DC=Ildc_var.out;
65
66              duty_cycle_DC=v_control_DC/Vdc_measured;
67
68              if(duty_cycle_DC>MAXDUTY) duty_cycle_DC=MAXDUTY;
69              if(duty_cycle_DC<MINDUTY) duty_cycle_DC=MINDUTY;
70        break;
71  }
72
73  /*
      ================================================================================
      */
74  /*                      Control Code  DC/AC power converter
                        */
75  /*
      ================================================================================
      */
76
77  switch(StateControl)
78  {
```

```
79      case ERROR:
80        VrefDQ.d=0.0;
81        VrefDQ.q=0.0;
82
83        IrefDQ.d=0.0;
84        IrefDQ.q=0.0;
85
86        //Reset Integral parts
87        Vd_var.intg = 0.0;
88        Vq_var.intg = 0.0;
89        Id_var.intg = 0.0;
90        Iq_var.intg = 0.0;
91
92        Valpha_var.intg = 0.0;
93        Vbeta_var.intg = 0.0;
94
95        Vramp=0.0;
96
97        EnablePWM_RST=1.0;  //disable PWM
98
99        //open all breakers
100       RST_SoftStart  =1.0;
101       RST_MainBreaker=1.0;
102       if (flag_AD_ready<0.5)
103       {
104         if (Reset>0.5 )
105           StateControl=RESET;
106       }
107     break;
108      case RESET:
109       EnablePWM_RST=1.0;
110             protection_flag=NO_ERROR;
111       if (Restart>0.5)
112       {
113         Vramp=0.0;
114         Vend_ramp=Vref;
115         DeltaV=100.0*TS;
116         StateControl=19.0;
117
```

```
118       }
119         break;
120     case 19:
121     RST_SoftStart   =0.0;
122     counter_Soft_Start = 0;
123     StateControl=20.0;
124     break;
125     case 20:
126     RST_SoftStart   =0.0;
127     if(counter_Soft_Start >= 10000){
128     RST_SoftStart   =1.0;
129     StateControl=21.0;
130     counter_Soft_Start = 0;
131     }
132     else
133       counter_Soft_Start++;
134     break;
135     case 21:
136     RST_MainBreaker=0.0;
137     if(counter_Soft_Start >= 10000){
138       counter_Soft_Start = 0;
139       StateControl=LOADCAPACITORS;
140     }
141     else
142       counter_Soft_Start++;
143     break;
144
145       case LOADCAPACITORS:
146           EnablePWM_RST=0.0;    //PWM enable for the DC/AC converter
147
148     //Ramp reference generation
149     Vramp+=DeltaV;
150
151     if(Vramp>=Vend_ramp)
152       Vramp=Vend_ramp;
153
154     VrefDQ.d=Vramp;
155     VrefDQ.q=0.0;
156
```

```
157     /*=========================================================================*/
158     /*                Single Loop Voltage control                              */
159     /*=========================================================================*/
160
161     //d-axis voltage control loop
162     Vd_var.ref=VrefDQ.d;
163     Vd_var.fbk=VcDQ.d;
164     Vd_par.kp=kp_v_SL;
165     Vd_par.ki=ki_v_SL;
166     Vd_var.ffw=0.0;
167     Vd_par.lim=SQRT1OVER3*Vdc_measured;
168     PIReg(&Vd_par, &Vd_var);
169     v_controlDQ_RST.d=Vd_var.out;
170
171     //q-axis voltage control loop
172     Vq_var.ref=VrefDQ.q;
173     Vq_var.fbk=VcDQ.q;
174     Vq_par.ki=ki_v_SL;
175     Vq_par.kp=kp_v_SL;
176     Vq_var.ffw=0.0;
177     Vq_par.lim=SQRT1OVER3*Vdc_measured;
178     PIReg(&Vq_par, &Vq_var);
179     v_controlDQ_RST.q=Vq_var.out;
180
181     //Transformations
182     _invrot(v_controlDQ_RST,SinCos_thetaref,v_controlAB_RST);
183     _invrot(VrefDQ,SinCos_thetaref,VrefAB);
184
185     if(fabs(VcDQ.d-Vend_ramp)<TOLV)
186     {
187         if(Go>0.5)
188         {
189          // close the softstart relé
190           RST_SoftStart=0.0;
191           counter_Soft_Start=0;
192           Regulator=Regulatortype;
193           Control=Controltype;
194           StateControl=SOFTSTART;
```

```
195         }
196       }
197     break;
198       case SOFTSTART:
199         //Ramp voltage reference generation
200      Vramp+=DeltaV;
201
202      if(Vramp>Vend_ramp)
203         Vramp=Vend_ramp;
204
205      VrefDQ.d=Vramp;
206      VrefDQ.q=0.0;
207
208       _invrot(VrefDQ,SinCos_thetaref,VrefAB);
209
210          if(Regulator<=0)    //PI Regulator
211      {
212         /*=========================================================================
    */
213         /*                Single Loop Voltage control
    */
214         /*=========================================================================
    */
215         //d-axis voltage control loop
216         Vd_var.ref=VrefDQ.d;
217         Vd_var.fbk=VcDQ.d;
218         Vd_par.kp=kp_v_SL;
219         Vd_par.ki=ki_v_SL;
220         Vd_var.ffw=0.0;
221         Vd_par.lim=SQRT1OVER3*Vdc_measured;
222         PIReg(&Vd_par, &Vd_var);
223         v_controlDQ_RST.d=Vd_var.out;
224
225         //q-axis voltage control loop
226         Vq_var.ref=VrefDQ.q;
227         Vq_var.fbk=VcDQ.q;
228         Vq_par.kp=kp_v_SL;
229         Vq_par.ki=ki_v_SL;
230         Vq_var.ffw=0.0;
```

```
231       Vq_par.lim=SQRT1OVER3*Vdc_measured;
232       PIReg(&Vq_par, &Vq_var);
233       v_controlDQ_RST.q=Vq_var.out;
234
235       //Transformation
236       _invrot(v_controlDQ_RST,SinCos_thetaref,v_controlAB_RST);
237         }
238     else //PRES Regulator
239     {
240       //alpha-axis voltage control loop
241             Valpha_var.ref=VrefAB.alpha;
242             Valpha_var.fbk=VcAB.alpha;
243       Valpha_par.kp=kp_v_SL;
244       Valpha_par.ki=ki_v_SL;
245             Valpha_par.lim=SQRT1OVER3*Vdc_measured;
246             PRReg(&Valpha_par, &Valpha_var,wref);
247             v_controlAB_RST.alpha=Valpha_var.out;
248
249       //beta-axis voltage control loop
250             Vbeta_var.ref=VrefAB.beta;
251             Vbeta_var.fbk=VcAB.beta;
252       Vbeta_par.kp=kp_v_SL;
253       Vbeta_par.ki=ki_v_SL;
254             Vbeta_par.lim=SQRT1OVER3*Vdc_measured;
255             PRReg(&Vbeta_par, &Vbeta_var,wref);
256             v_controlAB_RST.beta=Vbeta_var.out;
257     }
258       //Transformations
259       _invclarke(v_controlAB_RST,v_controlABC_RST);
260       _invclarke(VrefAB,VrefABC);
261
262        if(counter_Soft_Start>=TIME_SOFT_START)
263        {
264
265            RST_SoftStart=1.0;      //open SoftStart relays
266          RST_MainBreaker=0.0;  //close Main Breakers
267
268          if(Control>=1)  //Dual LOOP
269            {
```

```
270           if(Regulator <=0)
271           {
272           Vd_var.intg = IlDQ.d;
273           Vq_var.intg = IlDQ.q;
274           Id_var.intg = 0.0;
275           Iq_var.intg = 0.0;
276           }
277            else
278           {
279             Valpha_var.intg= IlAB.alpha;
280             Vbeta_var.intg= IlAB.beta;
281             Ialpha_var.intg = 0.0;
282             Ibeta_var.intg = 0.0;
283             }
284         }
285       else{  //Single Loop
286           if(Regulator>=1)
287           {
288           Valpha_var.intg= VcAB.alpha;
289           Vbeta_var.intg= VcAB.beta;
290           }
291         }
292            StateControl=GO;
293       }
294
295       counter_Soft_Start++;
296   break;
297     case GO:
298    VrefDQ.d=Vref;
299    VrefDQ.q=0.0;
300    //Reference Transformation
301    _invrot(VrefDQ,SinCos_thetaref,VrefAB);
302
303  if(Regulator <=0){
304  /*
     ==========================================================================
     */
305    /*                        PROPORTIONAL-INTEGRAL CONTROLLER
                                                                    */
```

```
306     /*
        ================================================================================
        */
307
308       if(Control<=0)  //Single Loop Voltage Control
309       {
310     //d-axis voltage control loop
311       Vd_var.ref=VrefDQ.d;
312       Vd_var.fbk=VcDQ.d;
313       Vd_par.kp=kp_v_SL;
314       Vd_par.ki=ki_v_SL;
315       Vd_var.ffw=0.0;
316       Vd_par.lim=SQRT1OVER3*Vdc_measured;
317       PIReg(&Vd_par, &Vd_var);
318       v_controlDQ_RST.d=Vd_var.out;
319
320     //q-axis voltage control loop
321       Vq_var.ref=VrefDQ.q;
322       Vq_var.fbk=VcDQ.q;
323       Vq_par.kp=kp_v_SL;
324       Vq_par.ki=ki_v_SL;
325       Vq_var.ffw=0.0;
326       Vq_par.lim=SQRT1OVER3*Vdc_measured;
327       PIReg(&Vq_par, &Vq_var);
328       v_controlDQ_RST.q=Vq_var.out;
329
330     //Trasformazioni
331       _invrot(v_controlDQ_RST,SinCos_thetaref,v_controlAB_RST);
332       _invrot(VrefDQ,SinCos_thetaref,VrefAB);
333       }
334       else     //Dual Loop Voltage Control
335       {
336     //d-axis voltage control loop
337             Vd_var.ref=VrefDQ.d;
338       Vd_var.fbk=VcDQ.d;
339       Vd_par.kp=kp_v_DL;
340       Vd_par.ki=ki_v_DL;
341             Vd_var.ffw=0.0;
342           Vd_par.lim=Vlim;
```

```
343             PIReg(&Vd_par, &Vd_var);
344               IrefDQ.d=Vd_var.out;
345
346
347     //q-axis voltage control loop
348               Vq_var.ref=VrefDQ.q;
349               Vq_var.fbk=VcDQ.q;
350       Vq_par.kp=kp_v_DL;
351       Vq_par.ki=ki_v_DL;
352             Vq_var.ffw=0.0;
353               Vq_par.lim=Vlim;
354               PIReg(&Vq_par, &Vq_var);
355               IrefDQ.q=Vq_var.out;
356
357     //d-axis current control loop
358               Id_var.ref=IrefDQ.d;
359               Id_var.fbk=IlDQ.d;
360       Id_par.kp=kp_i_DL;
361       Id_par.ki=ki_i_DL;
362               Id_var.ffw=VcDQ.d;   // FEEDFORWARD
363               Id_par.lim=SQRT1OVER3*Vdc_measured;
364               PIReg(&Id_par, &Id_var);
365               v_controlDQ_RST.d=Id_var.out;
366
367     //q-axis current control loop
368               Iq_var.ref=IrefDQ.q;
369               Iq_var.fbk=IlDQ.q;
370               Iq_var.ffw=VcDQ.q; // FEEDFORWARD
371       Iq_par.kp=kp_i_DL;
372       Iq_par.ki=ki_i_DL;
373               Iq_par.lim=SQRT1OVER3*Vdc_measured;
374               PIReg(&Iq_par, &Iq_var);
375               v_controlDQ_RST.q=Iq_var.out;
376
377           //Transformations
378     _invrot(IrefDQ,SinCos_thetaref,IrefAB);
379     _invrot(v_controlDQ_RST,SinCos_thetaref,v_controlAB_RST);
380       }
381   }
```

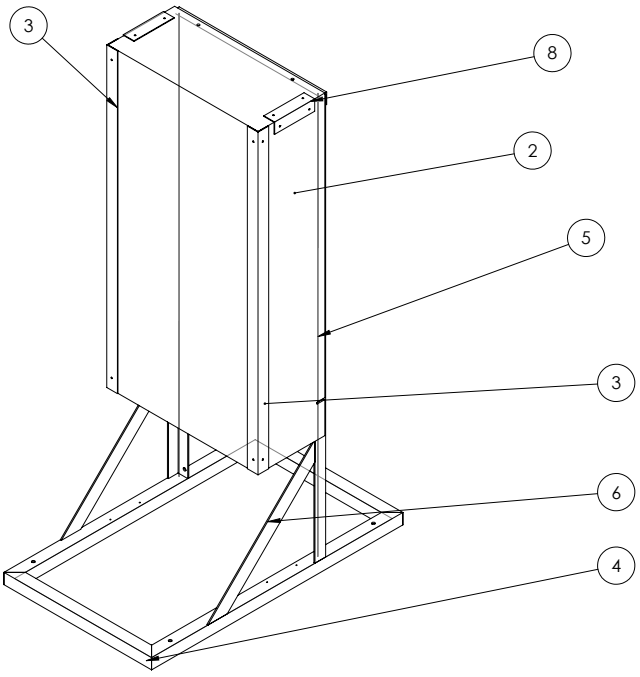
```
382   else{
383       /*
      ==========================================================================
      */
384 /*                          PROPORTIONAL-RESONANT CONTROLLER -
              */
385       /*
      ==========================================================================
      */
386
387           if(Control<=0)  //Single Loop Voltage Control
388           {
389       Valpha_par.kp=kp_v_SL;      //d  kp tensione SL
390       Vbeta_par.kp=kp_v_SL;      //q  kp tensione SL
391
392       Valpha_par.ki=ki_v_SL;      //d  ki tensione SL
393       Vbeta_par.ki=ki_v_SL;      //q  ki tensione SL
394
395       //alpha-axis voltage control loop
396             Valpha_var.ref=VrefAB.alpha;
397             Valpha_var.fbk=VcAB.alpha;
398             Valpha_par.lim=SQRT1OVER3*Vdc_measured;
399             PRReg(&Valpha_par, &Valpha_var,wref);
400             v_controlAB_RST.alpha=Valpha_var.out;
401
402       //beta-axis voltage control loop
403             Vbeta_var.ref=VrefAB.beta;
404             Vbeta_var.fbk=VcAB.beta;
405             Vbeta_par.lim=SQRT1OVER3*Vdc_measured;
406             PRReg(&Vbeta_par, &Vbeta_var,wref);
407             v_controlAB_RST.beta=Vbeta_var.out;
408        }
409        else{ //Dual Loop Voltage Control
410
411       Valpha_par.kp=kp_v_DL;
412       Valpha_par.ki=ki_v_DL;
413       Vbeta_par.kp=kp_v_DL;
414       Vbeta_par.ki=ki_v_DL;
415
```

```
416         Ialpha_par.kp=kp_i_DL;
417         Ialpha_par.ki=ki_i_DL;
418         Ibeta_par.kp=kp_i_DL;
419         Ibeta_par.ki=ki_i_DL;
420
421
422         //alpha-axis voltage control loop
423                 Valpha_var.ref=VrefAB.alpha;
424                 Valpha_var.fbk=VcAB.alpha;
425                 Valpha_par.lim=Vlim;
426         Valpha_var.ffw=0.0;
427                 PRReg(&Valpha_par, &Valpha_var,wref);
428                 IrefAB.alpha=Valpha_var.out;
429
430
431         //beta-axis voltage control loop
432                 Vbeta_var.ref=VrefAB.beta;
433                 Vbeta_var.fbk=VcAB.beta;
434                 Vbeta_par.lim=Vlim;
435         Vbeta_var.ffw=0.0;
436                 PRReg(&Vbeta_par, &Vbeta_var,wref);
437                 IrefAB.beta=Vbeta_var.out;
438
439         //alpha-axis current control loop
440                 Ialpha_var.ref=IrefAB.alpha;
441                 Ialpha_var.fbk=IlAB.alpha;
442                 Ialpha_var.ffw = VcAB.alpha;
443                 Ialpha_par.lim=SQRT1OVER3*Vdc_measured;
444                 PRReg(&Ialpha_par, &Ialpha_var, wref);
445                 v_controlAB_RST.alpha=Ialpha_var.out;
446
447         //beta-axis current control loop
448                 Ibeta_var.ref=IrefAB.beta;
449                 Ibeta_var.fbk=IlAB.beta;
450                 Ibeta_var.ffw = VcAB.beta;
451                 Ibeta_par.lim=SQRT1OVER3*Vdc_measured;
452                 PRReg(&Ibeta_par, &Ibeta_var, wref);
453                 v_controlAB_RST.beta=Ibeta_var.out;
454     }
```

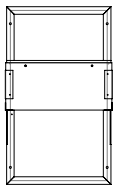
```
455     }
456       break;
457 }
458   //Transformation
459   _invclarke(VrefAB,VrefABC);
460   _invclarke(IrefAB,IrefABC);
461   _invclarke(v_controlAB_RST,v_controlABC_RST);
462   //duty cycle calculation
463   PWMduty(v_controlABC_RST, Vdc_measured, &duty_cycleABC_RST);
464 }
465 ...
```

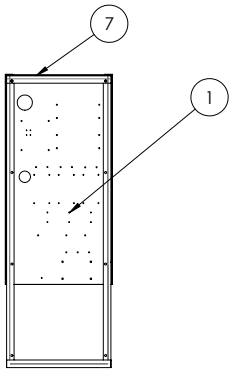
Listing 7.4: Grid-Forming C code.

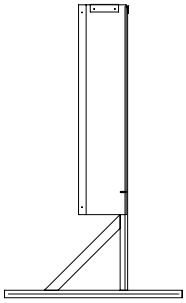
7.5 Mechanical Design 2D Drawing



SCALA 1 : 10







Num. articolo	Descrizione	Quantità
1	Piastra Alluminio Montaggio Componenti	1
2	Copertura Protettiva	1
3	Supporto Verticale Copertura	2
4	Base - Travi	1
5	Profilato L verticale	2
6	Sostegno 45°	2
7	Profilato L Orizzontale	1
8	Sostegno Orizzontale Copertura	2

SE NON SPECIFICATO:
QUOTE IN MILLIMETRI
FINITURA SUPERFICIE:
TOLLERANZE:
LINEARE: +/- 0.5 mm
ANGOLARE: 1°

FINITURA:

INTERRUZIONE BORDI NETTI

NON SCALARE DISEGNO

REVISIONE

	NOME	FIRMA	DATA
DISEG.			
VERIF.			
APPR.			
FABB.			
Qual.			

MATERIALE:

PESO:

TITOLO:

N. DISEGNO

Carrello

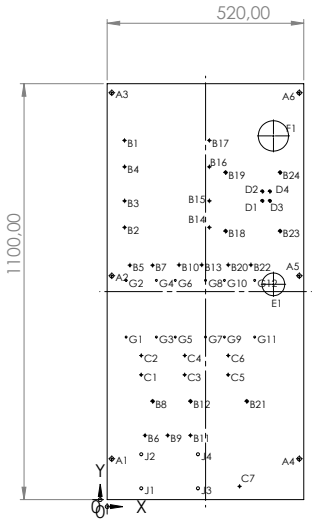
A3

SCALA:1:20

FOGLIO 1 DI 14

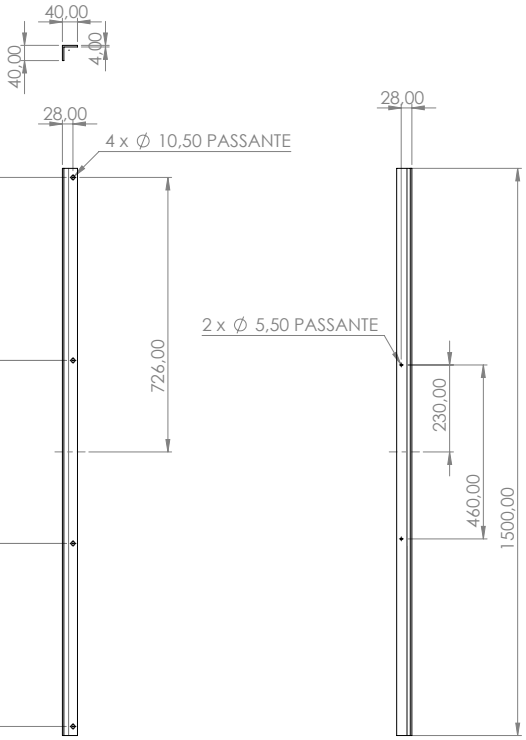
Scheda	Loc X	Loc Y	Dimensione
A1	12	108	Ø 11,00 PASSANTE
A2	12	592	Ø 11,00 PASSANTE
A3	12	1076	Ø 11,00 PASSANTE
A4	508	108	Ø 11,00 PASSANTE
A5	508	592	Ø 11,00 PASSANTE
A6	508	1076	Ø 11,00 PASSANTE
B1	45,73	950	Ø 5,50 PASSANTE
B2	46	720	Ø 5,50 PASSANTE
B3	46	790	Ø 5,50 PASSANTE
B4	46	880	Ø 5,50 PASSANTE
B5	60	620	Ø 5,50 PASSANTE
B6	100	170	Ø 5,50 PASSANTE
B7	120	619,81	Ø 5,50 PASSANTE
B8	120	260	Ø 5,50 PASSANTE
B9	160	170	Ø 5,50 PASSANTE
B10	190	620,76	Ø 5,50 PASSANTE
B11	220	170	Ø 5,50 PASSANTE
B12	220	260	Ø 5,50 PASSANTE
B13	250	620,76	Ø 5,50 PASSANTE
B14	270	720	Ø 5,50 PASSANTE
B15	270	790	Ø 5,50 PASSANTE
B16	270	880	Ø 5,50 PASSANTE
B17	270	950	Ø 5,50 PASSANTE
B18	313	710	Ø 5,50 PASSANTE
B19	313	865	Ø 5,50 PASSANTE
B20	320	620,76	Ø 5,50 PASSANTE
B21	369	260	Ø 5,50 PASSANTE
B22	380	620,76	Ø 5,50 PASSANTE
B23	457	710	Ø 5,50 PASSANTE
B24	457	865	Ø 5,50 PASSANTE
C1	89,74	330	Ø 4,50 PASSANTE
C2	90	381	Ø 4,50 PASSANTE
C3	204,74	330	Ø 4,50 PASSANTE
C4	205	381	Ø 4,50 PASSANTE
C5	319,74	330	Ø 4,50 PASSANTE
C6	320	381	Ø 4,50 PASSANTE
C7	350	35	Ø 4,50 PASSANTE
D1	410	790	Ø 3,40 PASSANTE
D2	410	815	Ø 3,40 PASSANTE
D3	430	790	Ø 3,40 PASSANTE
D4	430	815	Ø 3,40 PASSANTE
E1	439,53	569,36	Ø 60,00 PASSANTE
F1	439,5	961,87	Ø 80,00 PASSANTE
G1	50	430	Ø 5,00 PASSANTE
G2	50	580	Ø 5,00 PASSANTE
G3	130	430	Ø 5,00 PASSANTE
G4	130	580	Ø 5,00 PASSANTE
G5	180	430	Ø 5,00 PASSANTE
G6	180	580	Ø 5,00 PASSANTE
G7	260	430	Ø 5,00 PASSANTE
G8	260	580	Ø 5,00 PASSANTE
G9	310	430	Ø 5,00 PASSANTE
G10	310	580	Ø 5,00 PASSANTE
G11	390	430	Ø 5,00 PASSANTE
G12	390	580	Ø 5,00 PASSANTE
J1	90	30	Ø 8,00 PASSANTE
J2	90	120	Ø 8,00 PASSANTE
J3	240	30	Ø 8,00 PASSANTE
J4	240	120	Ø 8,00 PASSANTE

Materiale di partenza:
Piastra di Alluminio di spessore 5 mm



SE NON SPECIFICATO: QUOTE IN MILLIMETRI FINITURA SUPERFICIE: TOLLERANZE: LINEARE: +/- 0.5 mm ANGOLARE: 1°	FINITURA:		INTERRUZIONE BORDI NETTI				NON SCALARE DISEGNO	REVISIONE
	NOME	FIRMA	DATA				TITOLO: 1.Piastra Alluminio	
DISEG.								
VERIF.								
APPR.								
FABB.								
Qual.						MATERIALE: Alluminio	N. DISEGNO Carrello	A3
						PESO:	SCALA:1:10	FOGLIO 2 DI 14

Materiale di partenza:
Profilato di ferro ad L
40x40x4mm



SE NON SPECIFICATO: QUOTE IN MILLIMETRI FINITURA SUPERFICIE: TOLLERANZE: LINEARE: -/+ 0,5 mm ANGOLARE: 1°	FINITURA:	INTERRUZIONE BORDI NETTI	NON SCALARE DISEGNO / REVISIONE

	NOME	FIRMA	DATA
DISEG.			
VERIF.			
APPR.			
FABB.			
Qual.			

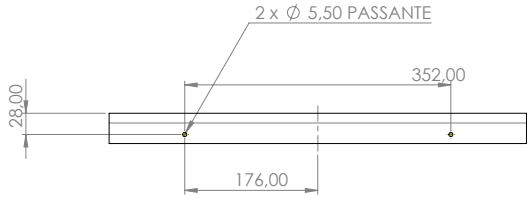
TITOLO: 5. Profilato L Verticale

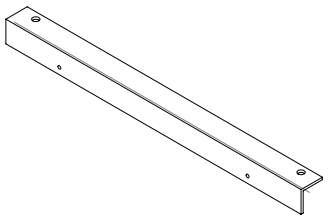
MATERIALE: Profilato L di ferro 40x40x4mm

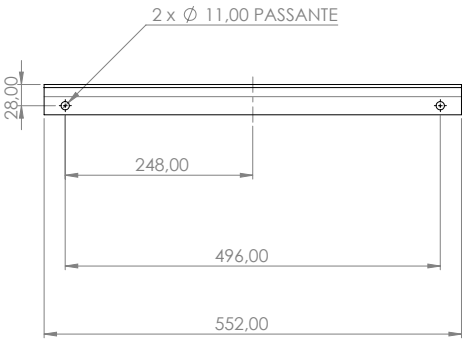
N. DISEGNO: Carrello — A3

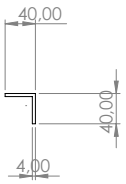
PESO: — SCALA:1:10 — FOGLIO 3 DI 14

Materiale di partenza:
Profilato di ferro ad L
40x40x4 mm



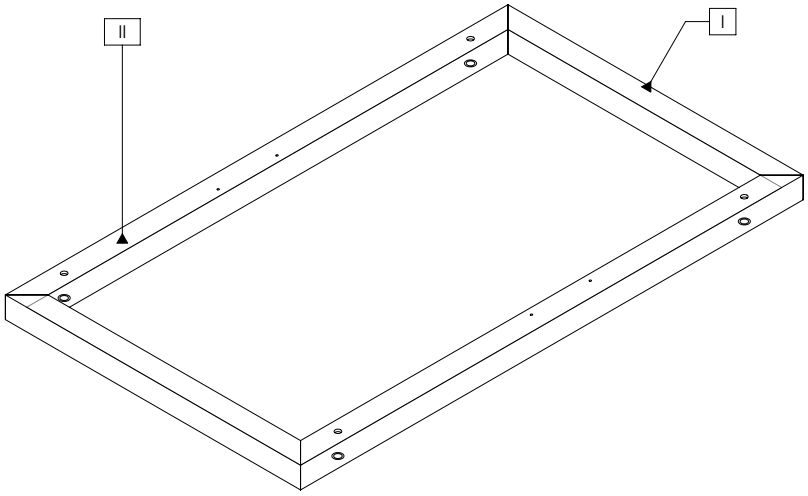






SE NON SPECIFICATO: QUOTE IN MILLIMETRI FINITURA SUPERFICIE: TOLLERANZE: LINEARE: -/+ 0,5 mm ANGOLARE: 1°	FINITURA:		INTERRUZIONE BORDI NETTI	NON SCALARE DISEGNO	REVISIONE
	NOME	FIRMA	DATA	TITOLO: 7. Profilato L Orizzontale	
DISEG.					
VERIF.					
APPR.					
FABB.					
Qual.			MATERIALE: Profilato ad L di ferro	N. DISEGNO Carrello	A3
			PESO:	SCALA:1:5	FOGLIO 4 DI 14

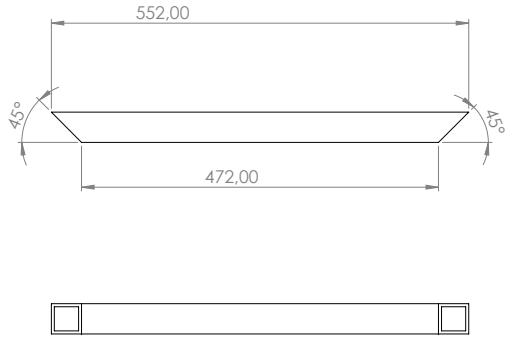
Materiale di partenza;
Profilato di ferro quadro
cavo 40x40x4mm

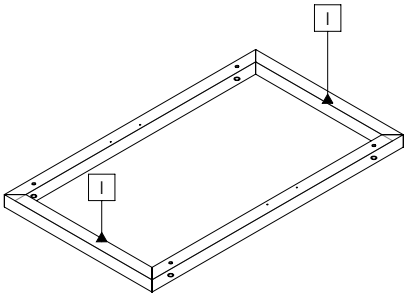


Num. Articolo	Descrizione	Quantità
I	Base - Profilato 1	2
II	Base - Profilato 2	2

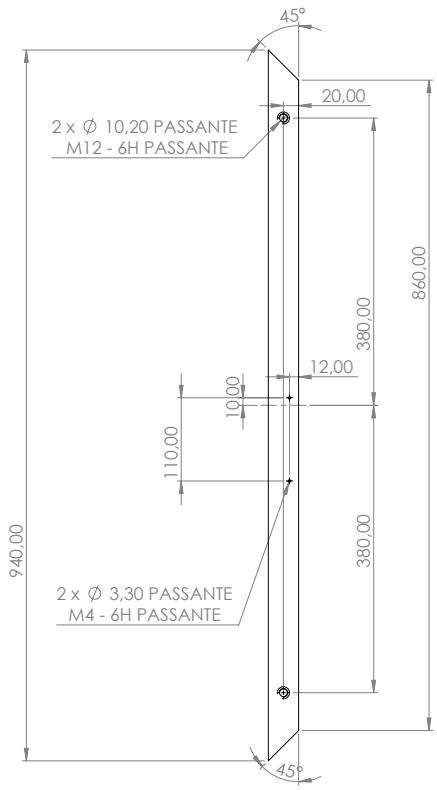
SE NON SPECIFICATO: QUOTE IN MILLIMETRI FINITURA SUPERFICIE: TOLLERANZE: LINEARE: +/- 0.5 mm ANGOLARE: 1°	FINITURA:		INTERRUZIONE BORDI NETTI	NON SCALARE DISEGNO	REVISIONE
	NOME	FIRMA	DATA	TITOLO: 4.Base	
DISEG.					
VERIF.					
APPR.					
FABB.					
Qual.			MATERIALE: Profilato quadro di ferro	N. DISEGNO Carrello	A3
			PESO:	SCALA:1:10	FOGLIO 5 DI 14

Materiale di partenza; Profilato
di ferro quadro cavo
40x40x4mm

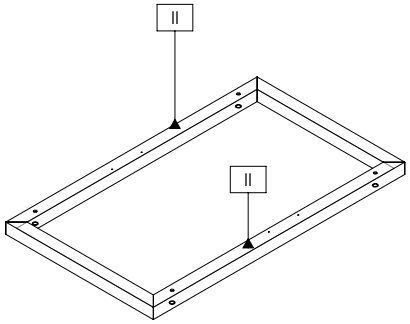




SE NON SPECIFICATO: QUOTE IN MILLIMETRI FINITURA SUPERFICIE: TOLLERANZE: LINEARE: +/- 0.5 mm ANGOLARE: 1°			FINITURA:			INTERRUZIONE BORDI NETTI	NON SCALARE DISEGNO	REVISIONE
	NOME	FIRMA	DATA				TITOLO: I. Base - Profilato 1	
DISEG.								
VERIF.								
APPR.								
FABB.								
Qual.								
				MATERIALE: Ferro			N. DISEGNO Carrello	A3
				PESO:			SCALA:1:10	FOGLIO 6 DI 14

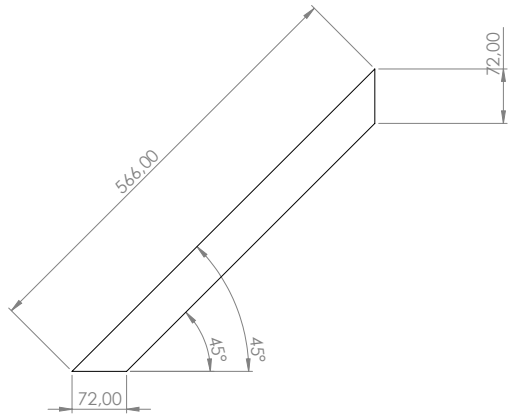


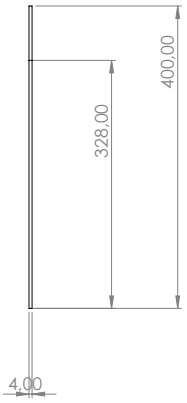
Materiale di partenza;
Profilato di ferro quadro cavo
40x40x4mm

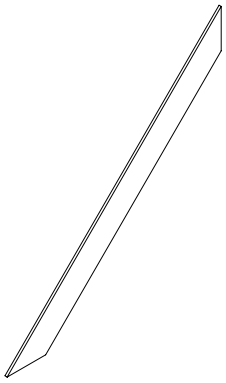


SE NON SPECIFICATO: QUOTE IN MILLIMETRI FINITURA SUPERFICIE: TOLLERANZE: LINEARE: -/+ 0.5 mm ANGOLARE: 1°	FINITURA:	INTERRUZIONE BORDI NETTI	NON SCALARE DISEGNO / REVISIONE
	NOME / FIRMA / DATA		TITOLO: II. Base - Profilato 2
DISEG.			
VERIF.			
APPR.			
FABB.			
Qual.		MATERIALE: Ferro	N. DISEGNO: Carrello — A3
		PESO:	SCALA:1:10 — FOGLIO 7 DI 14

Materiale di partenza:
travi di ferro sezione 40 mm x 4 mm

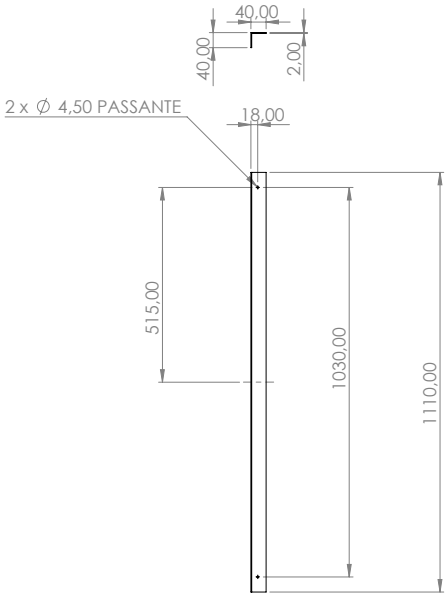




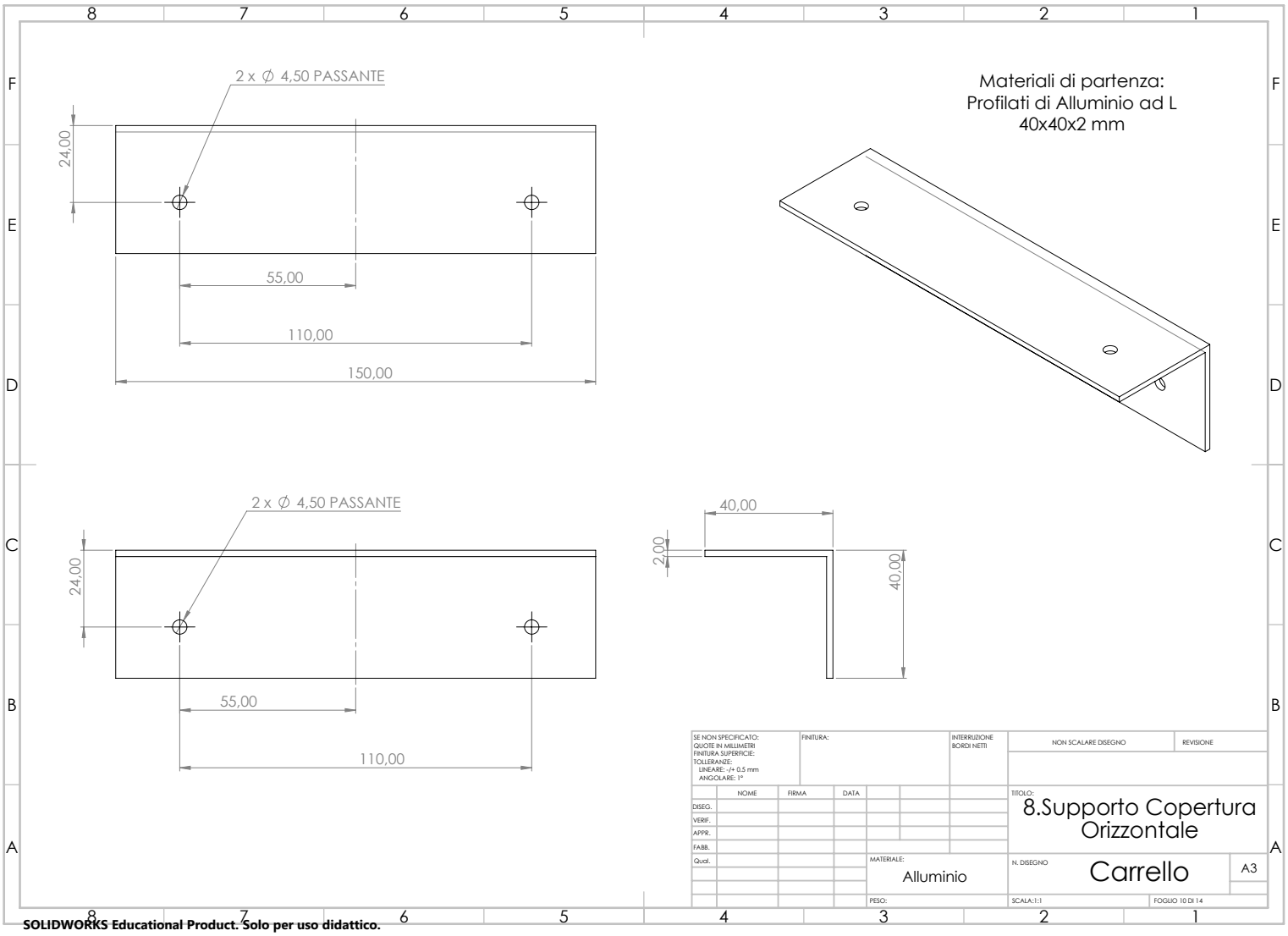


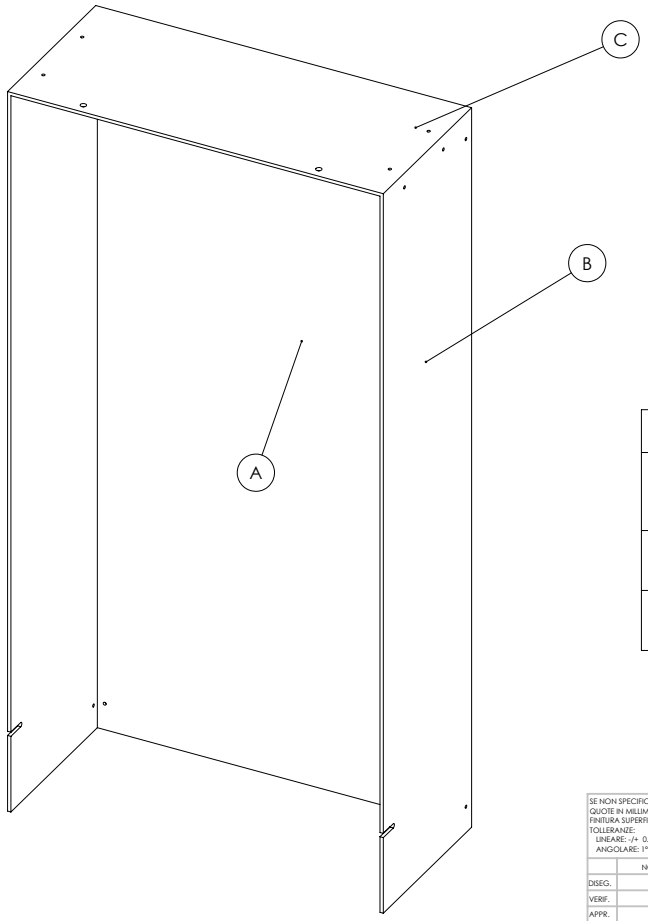
SE NON SPECIFICATO: QUOTE IN MILLIMETRI FINITURA SUPERFICIE: TOLLERANZE: LINEARE: +/- 0.5 mm ANGOLARE: 1°	FINITURA:		INTERRUZIONE BORDI NETTI	NON SCALARE DISEGNO	REVISIONE
	NOME	FIRMA	DATA	TITOLO: 6. Asta Sostegno	
DISEG.					
VERIF.					
APPR.					
FABB.					
Qual.					
				N. DISEGNO Carrello	A3
			MATERIALE: Ferro		
			PESO:	SCALA:1:5	FOGLIO 8 DI 14

Materiali di partenza:
Profilati di Alluminio ad L
40x40x2 mm



SE NON SPECIFICATO: QUOTE IN MILLIMETRI FINITURA SUPERFICIE: TOLLERANZE: LINEARE: +/- 0.5mm ANGOLARE: 1°		FINITURA:		INTERRUZIONE BORDI NETTI	NON SCALARE DISEGNO	REVISIONE	
	NOME	FIRMA	DATA				
DISEG.					3. Supporto VerticaleCopertura		
VERIF.							
APPR.							
FABB.							
Qual.				MATERIALE: Alluminio	N. DISEGNO Carrello		A3
				PESO:	SCALA:1:10	FOGLIO 9 DI 14	





Materiale di Partenza:
Lastra di Lexan di spessore 5 mm

Num.articolo	Descrizione	Quantità
A	Piastra Frontale Copertura Lexan	1
B	PiastraLaterale Copertura Lexan	2
C	Piastra Superiore Copertura Lexan	1

SE NON SPECIFICATO:
QUOTE IN MILLIMETRI
FINITURA SUPERFICIE:
TOLLERANZE:
LINEARE: -/+ 0.5 mm
ANGOLARE: 1°

FINITURA:

INTERRUZIONE BORDI NETTI

NON SCALARE DISEGNO | REVISIONE

	NOME	FIRMA	DATA
DISEG.			
VERIF.			
APPR.			
FABB.			
Qual.			

MATERIALE: Lexan

PESO:

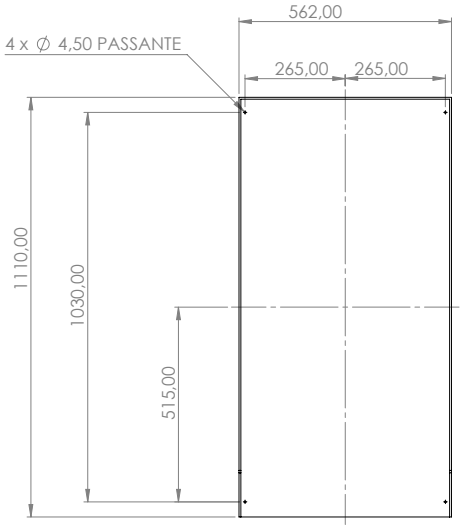
TITOLO: 2. Copertura Protettiva

N. DISEGNO: Carrello

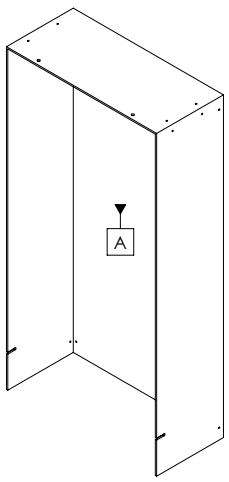
A3

SCALA:1:10

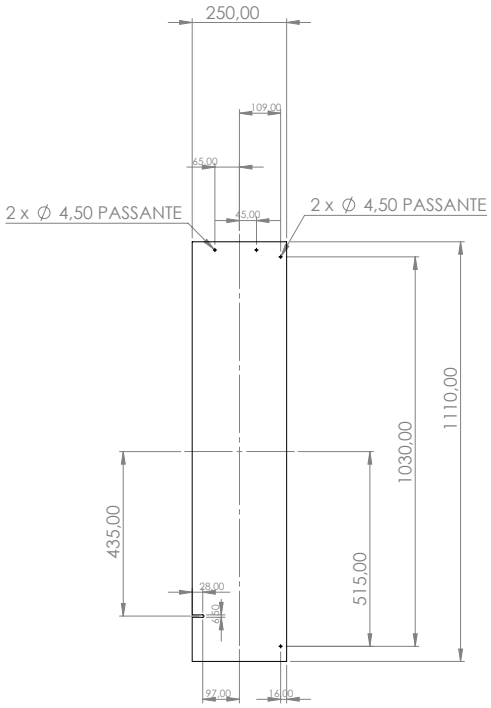
FOGLIO 11 DI 14



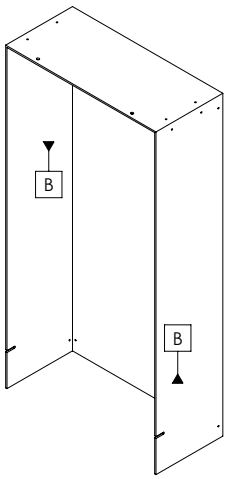
Materiale di Partenza:
Lastra di Lexan di spessore 5 mm



SE NON SPECIFICATO: QUOTE IN MILLIMETRI FINITURA SUPERFICIE: TOLLERANZE: LINEARE: +/- 0.5 mm ANGOLARE: 1°	FINITURA:			INTERRUZIONE BORDI NETTI	NON SCALARE DISEGNO / REVISIONE
	NOME	FIRMA	DATA		TITOLO: A. Piastra Frontale Copertura Lexan
DISEG.					
VERIF.					
APPR.					
FABB.					
Qual.				MATERIALE: Lexan	N. DISEGNO: Carrello — A3
				PESO:	SCALA:1:10 — FOGLIO 12 DI 14

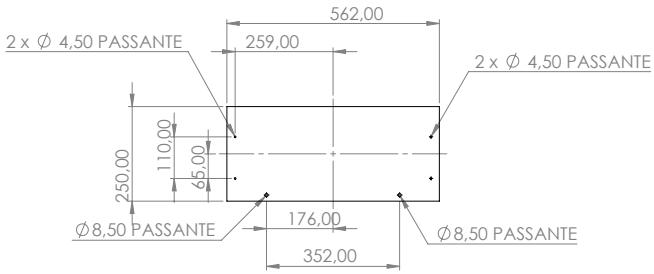


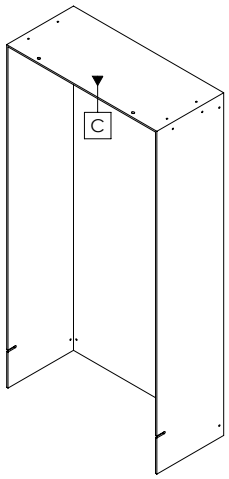
Materiale di Partenza:
Lastra di Lexan di spessore 5 mm



SE NON SPECIFICATO: QUOTE IN MILLIMETRI FINITURA SUPERFICIE: TOLLERANZE: LINEARE: +/- 0.5 mm ANGOLARE: 1°	FINITURA:	INTERRUZIONE BORDI NETTI	NON SCALARE DISEGNO / REVISIONE
	NOME / FIRMA / DATA		TITOLO: B.Piastra Laterale Copertura Lexan
DISEG.			
VERIF.			
APPR.			
FABB.			
Qual.		MATERIALE: Lexan	N. DISEGNO: Carrello — A3
		PESO:	SCALA:1:10 — FOGLIO 13 DI 14

Materiale di Partenza:
Lastra di Lexan di spessore 5 mm





SE NON SPECIFICATO: QUOTE IN MILLIMETRI FINITURA SUPERFICIE: TOLLERANZE: LINEARE: -/+ 0.5 mm ANGOLARE: 1°	FINITURA:		INTERRUZIONE BORDI NETTI	NON SCALARE DISEGNO	REVISIONE
	NOME	FIRMA	DATA	TITOLO: C. Piastra Superiore Copeertura Lexan	
DISEG.					
VERIF.					
APPR.					
FABB.					
Qual.				MATERIALE: Lexan	N. DISEGNO Carrello A3
				PESO:	SCALA:1:10 FOGLIO 14 DI 14

7.6 Vivado and FPGA Programming Workflow

Modern Xilinx FPGA boards are programmed by the Vivado Enviroment. In this project, VHDL was chosen to write the FPGA code. It must be noted that VHDL is an hardware description language and not a programming language. The main difference between an hardware description language and a programming language is that instructions in software programming (C,C++...) are sequentially executed while VHDL instructions in FPGA programming are mostly executed in parallel. The FPGA programming workflow is shown in Fig. 7.1.

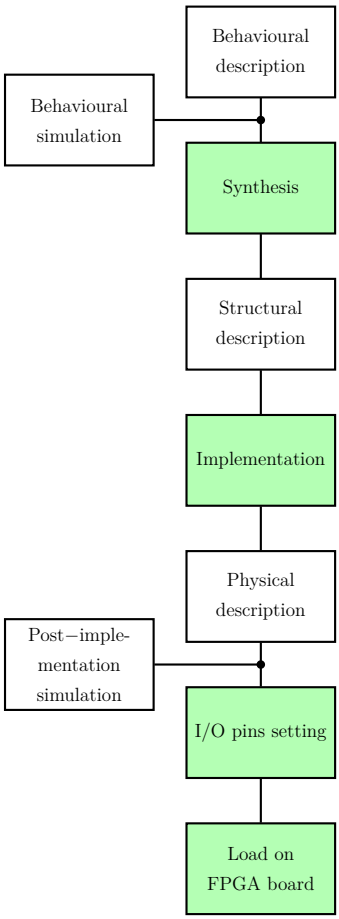


Figure 7.1: FPGA programming workflow.

So,

1. First, a behavioural description of what the FPGA must do is written, and a behavioural simulation is launched;
2. Then, the behavioural description is translated into a structural description. This process is called synthesis;
3. If the process of synthesis succeeds, a physical description is elaborated by the implementation procedure and a post-implantation simulation can be launched;
4. If the implementation process and post-implantation simulation gives good results without any error, the program can be loaded into the FPGA board through an USB cable using a JTAG communication protocol.

7.7 Serial Peripheral Interface (SPI) Communication Protocol

The Serial Peripheral Interface (SPI) communication protocol is a synchronous serial communication used in embedded systems.
SPI devices communicate using a master-slave architecture with a single master and multiple slave-devices. The master is in charge of the timing of the communication. Fig. 7.2 show a diagram of the connection:

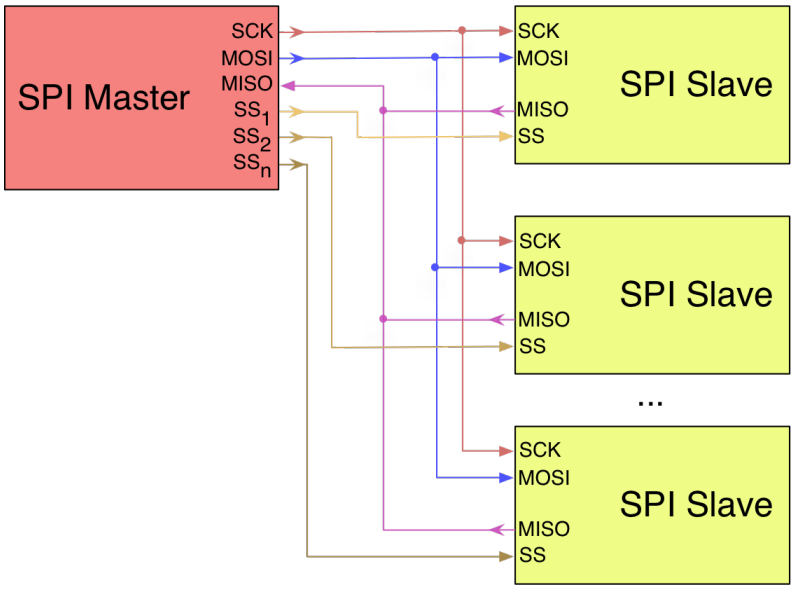


Figure 7.2: SPI communication generic scheme [3].

The SPI bus specifies four logic signals:

- SCLK: Serial Clock. It is a clock generated by the master that permits to synchronize the communication;
- MOSI: Master Output Slave Input. This bus contain the data send from the master;
- MISO: Master Input Slave Output. It contains the data output from slave;
- CS: Chip Select, also called Slave Select (SS). It is a signal send from the master to activate the slave, which has to receive or send data. It is often active low.

The communication starts when the CS bit is activated and the SCLK starts. Every rising and/or falling edge of SCLK a bit is sent from the Master to the Slave through the MOSI bus and a bit is received from the Slave through the MISO bus. SPI is "full duplex" because it has separate send (MOSI) and receive (MISO) lines. Therefore, in certain situations, it is possible to transmit and receive data at the same time.
A simple example is shown in Fig. 7.3

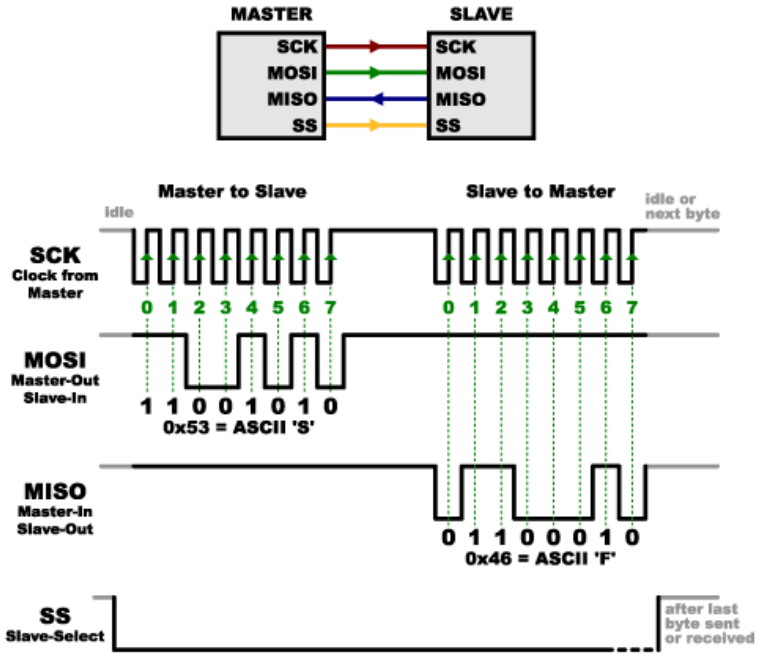


Figure 7.3: SPI communication example [1].

7.8 Analog Protections VHDL Code

```
...
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
--use IEEE.STD_LOGIC_ARITH.ALL;
--use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity AnalogProtections is
 generic  (N          : integer:=12; --N bit resolution
              crrTH: integer := 1675;
        voltACTH: integer := 1675;
              voltDCTH: integer := 1675;
        TempTH: integer := 1675;
              RisDiv2            : integer:=2048   ;
        n_campioni_max :integer :=2 );
  Port (     RESET :in STD_LOGIC;
```

```
17              clk_200MHz :in STD_LOGIC;
18              DataReady :in STD_LOGIC;
19              analog_reset :in STD_LOGIC;
20              voltageAB      : in  STD_LOGIC_VECTOR(N-1 downto 0);
21              voltageBC      : in  STD_LOGIC_VECTOR(N-1 downto 0);
22        voltageCA      : in  STD_LOGIC_VECTOR(N-1 downto 0);
23        voltageVDC       : in  STD_LOGIC_VECTOR(N-1 downto 0);
24        currentA       : in  STD_LOGIC_VECTOR(N-1 downto 0);
25        currentB       : in  STD_LOGIC_VECTOR(N-1 downto 0);
26        currentC       : in  STD_LOGIC_VECTOR(N-1 downto 0);
27        TemperatureNTC       : in  STD_LOGIC_VECTOR(N-1 downto 0);
28              FAULT_ADC: out STD_LOGIC;
29              WHICH_FAULT_ADC: out  STD_LOGIC_VECTOR(8-1 downto 0));
30 end AnalogProtections;
31
32 architecture Behavioral of AnalogProtections is
33
34
35
36 signal DataReceived : STD_LOGIC:='0';
37 signal CrrA : STD_LOGIC :='0';
38 signal CrrB : STD_LOGIC :='0';
39 signal CrrC : STD_LOGIC :='0';
40 signal Vab : STD_LOGIC :='0';
41 signal Vbc : STD_LOGIC :='0';
42 signal Vca : STD_LOGIC :='0';
43 signal VDC : STD_LOGIC :=  '0';
44 signal Temp : STD_LOGIC :=  '0';
45
46 signal cnt_CurrentA: integer := 0;
47 signal cnt_CurrentB: integer := 0;
48 signal cnt_CurrentC: integer := 0;
49 signal cnt_Vab: integer := 0;
50 signal cnt_Vbc: integer := 0;
51 signal cnt_Vca: integer := 0;
52 signal cnt_VDC: integer := 0;
53 signal cnt_Temp: integer := 0;
54
```

```
55 constant currMax : unsigned(N-1 downto 0) := to_unsigned(crrTH +
       RisDiv2, N);
56 constant currMin : unsigned(N-1 downto 0) := to_unsigned(RisDiv2 -
       crrTH, N);
57
58 constant voltMax : unsigned(N-1 downto 0) := to_unsigned(voltACTH +
       RisDiv2, N);
59 constant voltMin : unsigned(N-1 downto 0) := to_unsigned(RisDiv2 -
       voltACTH, N);
60
61 constant voltDCMax : unsigned(N-1 downto 0) := to_unsigned(voltDCTH, N
       );
62
63 constant TempMax : unsigned(N-1 downto 0) := to_unsigned(TempTH, N);
64
65 begin
66
67 process(clk_200MHz) --,DataReady
68 begin
69 IF ( rising_edge(clk_200MHz))THEN
70   IF (RESET ='0') then
71       IF(analog_reset = '0') then
72         CrrA <= '0';
73     CrrB <= '0';
74     CrrC <= '0';
75     VDC <= '0';
76     Vab <= '0';
77     Vbc <= '0';
78     Vca <= '0';
79     Temp <= '0';
80     cnt_CurrentA <= 0;
81     cnt_CurrentB <= 0;
82     cnt_CurrentC <= 0;
83     cnt_VDC <= 0;
84     cnt_Vab <= 0;
85     cnt_Vbc <= 0;
86     cnt_Vca <= 0;
87     cnt_Temp <= 0;
88     elsif(DataReady ='1') then
```

```
89          if(unsigned(currentA)>currMax OR unsigned(currentA)<
    unsigned(currMin)) then
90              cnt_CurrentA <=  cnt_CurrentA +1;
91                  if(cnt_CurrentA >=n_campioni_max)then
92                      CrrA <= '1';
93                  end if;
94          else
95              cnt_CurrentA <= 0;
96          end if;
97          if(unsigned(currentB)>currMax OR unsigned(currentB)<
    unsigned(currMin)) then
98              cnt_CurrentB <=  cnt_CurrentB +1;
99                  if(cnt_CurrentB >=n_campioni_max)then
100                     CrrB <= '1';
101                 end if;
102         else
103             cnt_CurrentB <= 0;
104         end if;
105         if(unsigned(currentC)>currMax OR unsigned(currentC)<
    unsigned(currMin)) then
106             cnt_CurrentC <=  cnt_CurrentC +1;
107                 if(cnt_CurrentC >=n_campioni_max)then
108                     CrrC <= '1';
109                 end if;
110         else
111             cnt_CurrentC <= 0;
112     end if;
113     if(unsigned(voltageAB)>voltMax OR unsigned(voltageAB)<unsigned(
    voltMin)) then
114             cnt_Vab <=  cnt_Vab +1;
115                 if(cnt_Vab >=n_campioni_max)then
116                     Vab <= '1';
117                 end if;
118         else
119             cnt_Vab <= 0;
120         end if;
121     if(unsigned(voltageBC)>voltMax OR unsigned(voltageBC)<unsigned(
    voltMin)) then
122             cnt_Vbc <=  cnt_Vbc +1;
```

```
123                     if(cnt_Vbc >=n_campioni_max)then
124                         Vbc <= '1';
125                     end if;
126             else
127                 cnt_Vbc <= 0;
128             end if;
129     if(unsigned(voltageCA)>voltMax OR unsigned(voltageCA)<unsigned(
    voltMin)) then
130                 cnt_Vca <=  cnt_Vca +1;
131                     if(cnt_Vca >=n_campioni_max)then
132                         Vca <= '1';
133                     end if;
134             else
135                 cnt_Vca <= 0;
136             end if;
137              if(unsigned(voltageVDC )>voltDCMax) then
138                 cnt_VDC <=  cnt_VDC +1;
139                     if(cnt_VDC >=n_campioni_max)then
140                         VDC <= '1';
141                     end if;
142              else
143                 cnt_VDC <= 0;
144              end if;
145              if(unsigned(TemperatureNTC)>TempMax) then
146                 cnt_Temp <=  cnt_Temp +1;
147                     if(cnt_Temp >=n_campioni_max)then
148                         Temp <= '1';
149                     end if;
150              else
151                 cnt_Temp <= 0;
152              end if;
153     end if;
154   --WHICH_FAULT_ADC <= '0' & CrrA;
155 ELSE
156    CrrA <= '0';
157 CrrB <= '0';
158 CrrC <= '0';
159   VDC <= '0';
160 Vab <= '0';
```

```
161   Vbc <= '0';
162   Vca <= '0';
163   Temp <= '0';
164     cnt_CurrentA <= 0;
165   cnt_CurrentB <= 0;
166   cnt_CurrentC <= 0;
167     cnt_VDC <= 0;
168   cnt_Vab <= 0;
169   cnt_Vbc <= 0;
170   cnt_Vca <= 0;
171   cnt_Temp <= 0;
172     END IF;
173 END IF;
174 end process;
175 FAULT_ADC <=  VDC or CrrA or CrrB or CrrC or Vab or Vbc or Vca or Temp
      ;
176 WHICH_FAULT_ADC <= CrrA & CrrB & CrrC & Vab & Vbc & Vca & VDC & Temp;
177 end Behavioral;
178
179
180 ...
```

Listing 7.5: Analog Protections VHDL code.

7.9 Management of the Gate Signals VHDL Code

7.9.1 Driver Management Block VHDL Code

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity DriversManagement is
5 generic (   N       :             integer := 12;
6             T               :           integer := 7);
7
8  Port (
```

```
9               clk_200MHz  :in STD_LOGIC;
10                RESET  : in STD_LOGIC;
11                RESET_FAULT  : in STD_LOGIC;
12                dSpaceOUT  : in STD_LOGIC_VECTOR (N-1 downto 0);
13               FAULT_DRIVER: IN STD_LOGIC;
14                DRIVERcommands :  out STD_LOGIC_VECTOR (T-1 downto 0));
15 end DriversManagement;
16
17 architecture Behavioral of DriversManagement is
18 -- variabili dove memorizzare i bit letti da dSpace
19 signal dSpaceOUTbit0:STD_LOGIC:='0';
20 signal dSpaceOUTbit1:STD_LOGIC:='0';
21 signal dSpaceOUTbit2:STD_LOGIC:='0';
22 signal dSpaceOUTbit3:STD_LOGIC:='0';
23 signal dSpaceOUTbit4:STD_LOGIC:='0';
24 signal dSpaceOUTbit5:STD_LOGIC:='0';
25 signal dSpaceOUTbit6:STD_LOGIC:='0';
26 signal dSpaceOUTbit7:STD_LOGIC:='0';
27
28 signal T1:STD_LOGIC:='0';
29 signal T2:STD_LOGIC:='0';
30 signal T3:STD_LOGIC:='0';
31 signal T4:STD_LOGIC:='0';
32 signal T5:STD_LOGIC:='0';
33 signal T6:STD_LOGIC:='0';
34 signal T7:STD_LOGIC:='0';
35
36 begin
37 process(clk_200MHz)
38 begin
39     if(rising_edge(clk_200MHz)) then
40         --leggo da dSpace
41         dSpaceOUTbit0<= dSpaceOUT(0);
42         dSpaceOUTbit1<= dSpaceOUT(1);
43         dSpaceOUTbit2<= dSpaceOUT(2);
44         dSpaceOUTbit3<= dSpaceOUT(3);
45         dSpaceOUTbit4<= dSpaceOUT(4);
46         dSpaceOUTbit5<= dSpaceOUT(5);
47         dSpaceOUTbit6<= dSpaceOUT(6);
```

```
48          dSpaceOUTbit7<= dSpaceOUT(7); --EnablePWM
49        IF(RESET ='0')THEN
50           if(dSpaceOUTbit7 ='1' and RESET_FAULT ='1' and FAULT_DRIVER
    ='0')then
51              T1 <= dSpaceOUTbit0;
52              T2 <= dSpaceOUTbit1;
53              T3 <= dSpaceOUTbit2;
54              T4 <= dSpaceOUTbit3;
55              T5 <= dSpaceOUTbit4;
56              T6 <= dSpaceOUTbit5;
57              T7 <= dSpaceOUTbit6;
58            else
59              T1 <= '0';
60              T2 <= '0';
61              T3 <= '0';
62              T4 <= '0';
63              T5 <= '0';
64              T6 <= '0';
65              T7 <= '0';
66            end if;
67        ELSE
68          T1 <= '0';
69          T2 <= '0';
70          T3 <= '0';
71          T4 <= '0';
72          T5 <= '0';
73          T6 <= '0';
74          T7 <= '0';
75        END IF;
76     end if;
77 end process;
78
79 DRIVERcommands(T-1 downto 0) <= T7 & T6 & T5 & T4 & T3 & T2 & T1 ;
80
81 end Behavioral;
```

Listing 7.6: Driver Management block VHDL code.

7.9.2 Drivers Fault Block VHDL Code

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  -- Uncomment the following library declaration if using
5  -- arithmetic functions with Signed or Unsigned values
6  --use IEEE.NUMERIC_STD.ALL;
7
8  -- Uncomment the following library declaration if instantiating
9  -- any Xilinx leaf cells in this code.
10 --library UNISIM;
11 --use UNISIM.VComponents.all;
12
13 entity DriverFault is
14 generic (   N       :             integer := 12;
15           G                :             integer := 7);
16
17  Port (
18              GUASCH_DRIVER_SIGNALS : in STD_LOGIC_VECTOR (G-1 downto 0)
    ;
19              FAULT_DRIVER: out STD_LOGIC );
20 end DriverFault;
21
22 architecture Behavioral of DriverFault is
23
24 begin
25
26 FAULT_DRIVER<=(NOT GUASCH_DRIVER_SIGNALS (0)) OR (NOT
    GUASCH_DRIVER_SIGNALS (1)) OR (NOT GUASCH_DRIVER_SIGNALS (2)) OR (
    NOT GUASCH_DRIVER_SIGNALS (3)) OR (NOT GUASCH_DRIVER_SIGNALS (4))
    OR (NOT GUASCH_DRIVER_SIGNALS (5)) OR (NOT GUASCH_DRIVER_SIGNALS
    (6));
27
28 end Behavioral;
```

Listing 7.7: Drivers Fault block VHDL code.

7.10 Finite State Machine VHDL Code

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity FSM is
5  generic (   N       :              integer := 12;
6              R               :              integer := 4;
7              L               :              integer := 2);
8    Port (   clk_200MHz :in STD_LOGIC;
9             RESET : in STD_LOGIC;
10            dSpaceOUT : in STD_LOGIC_VECTOR (N-1 downto 0);
11            FAULT_ADC : in STD_LOGIC;
12            WHICH_FAULT_ADC : in STD_LOGIC_VECTOR(8-1 downto 0);
13            FAULT_DRIVERS : in STD_LOGIC;
14            --EMG_PushButton: in STD_LOGIC;
15            RESET_FAULT : out STD_LOGIC;
16            ERROR_FPGA : out STD_LOGIC;
17            dSpaceIN : out STD_LOGIC_VECTOR (N-1 downto 0);
18            TESTMODE : out STD_LOGIC;
19            GIVE_INSTRUCTIONS_TO_LEDS : out STD_LOGIC;
20            GIVE_INSTRUCTIONS_TO_RELAYS : out STD_LOGIC;
21            what_to_do_relays :out STD_LOGIC_VECTOR (R-1 downto 0)
22            );
23
24 end FSM;
25
26 architecture Behavioral of FSM is
27 TYPE State_type IS (ERROR, RESETSTATE, TESTMODESTATE, GO);
28 -- Define the states
29 SIGNAL State : State_Type := RESETSTATE;
30
31 signal ERRORSTATE: STD_LOGIC := '0';
32 signal RESET_FAULT_FLAG: STD_LOGIC  :='0';
33
34 signal dSpaceOUTbit0:STD_LOGIC:='0';
35 signal dSpaceOUTbit1:STD_LOGIC:='0';
36 signal dSpaceOUTbit2:STD_LOGIC:='0';
```

```
37 signal dSpaceOUTbit3:STD_LOGIC:='0';
38 signal dSpaceOUTbit4:STD_LOGIC:='0';
39 signal dSpaceOUTbit5:STD_LOGIC:='0';
40 signal dSpaceOUTbit6:STD_LOGIC:='0';
41 signal dSpaceOUTbit7:STD_LOGIC:='0';
42 signal dSpaceOUTbit8:STD_LOGIC:='0';
43 signal dSpaceOUTbit9:STD_LOGIC:='0';
44 signal dSpaceOUTbit10:STD_LOGIC:='0';
45 signal dSpaceOUTbit11:STD_LOGIC:='0';
46
47 signal dSpaceINbuffer: STD_LOGIC_VECTOR(N-1 downto 0) :="000000000000"
       ;
48
49 signal K1STATE:STD_LOGIC:='0';
50 signal K2STATE:STD_LOGIC:='0';
51 signal K3STATE:STD_LOGIC:='0';
52 signal OpAmpSTATE:STD_LOGIC:='0';
53 --signal K1stateMemory:STD_LOGIC:='0';
54
55 signal cmdK1:STD_LOGIC:='0';
56 signal cmdK2:STD_LOGIC:='0';
57 signal cmdK3:STD_LOGIC:='0';
58 signal cmdOpAmp:STD_LOGIC:='0';
59 signal cmdGO:STD_LOGIC:='0';
60 signal cmdEXTfault:STD_LOGIC:='0';
61 signal cmdERRORFPGAdSpace:STD_LOGIC:='0';
62 signal cmdTESTMODE: STD_LOGIC := '0';
63
64
65 signal flagK1 :std_logic :='0';
66 signal flagK2 :std_logic :='0';
67 signal flagK3 :std_logic :='0';
68 signal flagOpAmp :std_logic :='0';
69
70 signal counterAWAKE : integer := 0;
71 constant  counterAWAKEMAX: integer := 200000;
72 signal AWAKE :std_logic :='1';   --AWAKE signal flag
73
74 signal NOTSAFE : STD_LOGIC := '0';
```

```
75
76 begin
77
78 process(clk_200MHz)
79 begin
80 IF(RISING_EDGE(clk_200MHz)) THEN
81 --leggo da dSpace
82 dSpaceOUTbit0<= dSpaceOUT(0);
83 dSpaceOUTbit1<= dSpaceOUT(1);
84 dSpaceOUTbit2<= dSpaceOUT(2);
85 dSpaceOUTbit3<= dSpaceOUT(3);
86 dSpaceOUTbit4<= dSpaceOUT(4);
87 dSpaceOUTbit5<= dSpaceOUT(5);
88 dSpaceOUTbit6<= dSpaceOUT(6);
89 dSpaceOUTbit7<= dSpaceOUT(7);
90 dSpaceOUTbit8<= dSpaceOUT(8);
91 dSpaceOUTbit9<= dSpaceOUT(9);
92 dSpaceOUTbit10<= dSpaceOUT(10);
93 dSpaceOUTbit11<= dSpaceOUT(11);
94     if(RESET ='0') then
95          GIVE_INSTRUCTIONS_TO_LEDS <= '0';
96          GIVE_INSTRUCTIONS_TO_RELAYS <='0';
97          if(dSpaceOUTbit8 ='0') then
98               counterAWAKE <= counterAWAKE +1;
99               if(counterAWAKE = counterAWAKEMAX) then
100                 AWAKE <= '0';  --dSpace platform is off
101              end if;
102         else
103              counterAWAKE <= 0;
104         end if;
105    -- interpreto i comandi da dSpace bit9 - bit10 - bit11
106         if(dSpaceOUTbit9 ='0' AND dSpaceOUTbit10 = '1' AND
    dSpaceOUTbit11 = '1') then   --k1 command
107              cmdK1 <= '1';
108         else
109              cmdK1 <= '0';
110         end if;
111         if(dSpaceOUTbit9 ='1' AND dSpaceOUTbit10 = '0' AND
    dSpaceOUTbit11 = '0') then --k2 COMMAND
```

```
112             cmdK2 <= '1';
113         else
114             cmdK2 <= '0';
115         end if;
116         if(dSpaceOUTbit9 ='1' AND dSpaceOUTbit10 = '0' AND
    dSpaceOUTbit11 = '1') then    --k3 commands
117             cmdK3 <= '1';
118         else
119             cmdK3 <= '0';
120         end if;
121         if(dSpaceOUTbit9 ='1' AND dSpaceOUTbit10 = '1' AND
    dSpaceOUTbit11 = '0') then    --OpAmp command
122             cmdOpAmp <= '1';
123         else
124             cmdOpAmp  <= '0';
125         end if;
126         if(dSpaceOUTbit9 ='1' AND dSpaceOUTbit10 = '1' AND
    dSpaceOUTbit11 = '1') then    --go command
127             cmdGO <= '1';
128         else
129             cmdGO <= '0';
130         end if;
131         if(dSpaceOUTbit9 ='0' AND dSpaceOUTbit10 = '0' AND
    dSpaceOUTbit11 = '0') then -- ext FAULT command
132             cmdEXTfault <= '1';
133         else
134             cmdEXTfault <= '0';
135         end if;
136         if(dSpaceOUTbit9 ='0' AND dSpaceOUTbit10 = '0' AND
    dSpaceOUTbit11 = '1') then    -- testmode command
137             cmdTESTMODE <='1';
138         else
139             cmdTESTMODE <= '0';
140         end if;
141         if(dSpaceOUTbit9 ='0' AND dSpaceOUTbit10 = '1' AND
    dSpaceOUTbit11 = '0') then    --cmd eror_fpga_from_dSpace
142             cmdERRORFPGAdSpace <= '1';
143         else
144             cmdERRORFPGAdSpace <= '0';
```

```
145         end if;
146
147         NOTSAFE <= cmdERRORFPGAdSpace or (NOT AWAKE) OR FAULT_DRIVERS
    OR FAULT_ADC;
148       if( NOTSAFE = '1') then
149       State <= ERROR;
150     end if;
151
152     CASE State IS
153          WHEN ERROR =>
154             ERRORSTATE <= '1';
155           RESET_FAULT_FLAG <='0';
156           GIVE_INSTRUCTIONS_TO_LEDS <= '0';
157           GIVE_INSTRUCTIONS_TO_RELAYS <= '0';
158           TESTMODE <= '0';
159           if(cmdEXTfault = '1') then
160                 ERRORSTATE <= '0';
161                 RESET_FAULT_FLAG <='1';
162               State <= RESETSTATE;
163         end if;
164       WHEN RESETSTATE =>
165             ERRORSTATE <= '0';
166             RESET_FAULT_FLAG <='1';
167             TESTMODE <= '0';
168             GIVE_INSTRUCTIONS_TO_LEDS <= '0';
169             GIVE_INSTRUCTIONS_TO_RELAYS <= '0';
170             if(cmdGO = '1') then
171                 State <= GO;
172           end if;
173           if(cmdTESTMODE = '1') then
174                 State <= TESTMODESTATE;
175           end if;
176       WHEN TESTMODESTATE =>
177            TESTMODE <= '1';
178           if(cmdEXTfault = '1') then
179             State <= RESETSTATE;
180         end if;
181       WHEN GO =>
182                 GIVE_INSTRUCTIONS_TO_LEDS <= '1';
```

```
183                 GIVE_INSTRUCTIONS_TO_RELAYS <='1';
184                  --comando i relays
185                 --K1
186                 if(flagK1 ='0' AND cmdK1 = '1') then
187                     flagK1 <= '1';
188                     K1STATE <= NOT K1STATE;
189                 end if;
190                  if(cmdK1 ='0') then
191                     flagK1 <= '0';
192                  end if;
193                  --K2
194                 if(flagK2 ='0' AND cmdK2 = '1') then
195                      flagK2 <= '1';
196                      K2STATE <= NOT K2STATE;
197                  end if;
198                 if(cmdK2 ='0') then
199                    flagK2 <= '0';
200                  end if;
201             --K3
202             if(flagK3 ='0' AND cmdK3 = '1') then
203                    flagK3 <= '1';
204                    K3STATE <= NOT K3STATE;
205             end if;
206             if(cmdK3 ='0') then
207                    flagK3 <= '0';
208             end if;
209             --OpAmp
210             if(flagOpAmp ='0' AND cmdOpAmp = '1') then
211                    flagOpAmp <= '1';
212                    OpAmpSTATE <= NOT OpAmpSTATE;
213             end if;
214             if(cmdOpAmp ='0') then
215                    flagOpAmp <= '0';
216             end if;
217         END CASE;
218     else
219         GIVE_INSTRUCTIONS_TO_LEDS <= '0';
220         GIVE_INSTRUCTIONS_TO_RELAYS <='0';
221         ERRORSTATE <= '1';
```

```
222             State <= RESETSTATE;
223         end if;
224 dSpaceINbuffer(N-1 downto 0) <= WHICH_FAULT_ADC(0) & WHICH_FAULT_ADC
        (1) & WHICH_FAULT_ADC(2) & WHICH_FAULT_ADC(3) & WHICH_FAULT_ADC(4)
        & WHICH_FAULT_ADC(5) & WHICH_FAULT_ADC(6) & WHICH_FAULT_ADC(7) &
        FAULT_DRIVERS & dSpaceOUTbit7 & RESET  & ERRORSTATE;
225 END IF;
226 end process;
227  dSpaceIN(N-1 downto 0)<= dSpaceINbuffer(N-1 downto 0);
228 ERROR_FPGA <= ERRORSTATE;
229 RESET_FAULT <= RESET_FAULT_FLAG;
230 what_to_do_relays(0) <= K1STATE;
231 what_to_do_relays(1) <= K2STATE;
232 what_to_do_relays(2) <= K3STATE;
233 what_to_do_relays(3) <= OpAmpSTATE;
234 end Behavioral;
```

Listing 7.8: Finite State Machine (FSM) VHDL code.

7.11 Leds Block VHDL Code

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  entity LEDS is
7    Port (      clk_5MHz : in STD_LOGIC;
8                ERROR_FPGA: in STD_LOGIC;
9                RESET: in STD_LOGIC ;
10               RESET_FAULT: in STD_LOGIC ;
11               gotoTESTMODE: in  STD_LOGIC ;
12               INSTRUCTIONS_RECEIVED : in STD_LOGIC;
13               LED1 : out STD_LOGIC;
14               LED2 : out STD_LOGIC);
15 end LEDS;
16
17 architecture Behavioral of LEDS is
```

```
18 TYPE State_type IS (ERRORSTATE, RESETSTATE, TESTMODE, DO); -- Define
       the states
19 SIGNAL State : State_Type := RESETSTATE;
20
21 signal counterLED : integer := 0;
22 constant cntMaxLED: integer := 5000000;
23 constant cntMaxLEDBlink: integer := 25000;
24
25 signal LED1_State : STD_LOGIC := '0';
26 signal LED2_State : STD_LOGIC := '0';
27
28 signal flagTESTMODEleds :STD_LOGIC := '0';
29 signal leadTESTMODEstateleds :STD_LOGIC := '0';
30
31 begin
32
33
34 process(clk_5MHz)
35 begin
36 IF(rising_edge(clk_5MHz)) then
37   IF(RESET='0') then
38         if(ERROR_FPGA = '1') then
39             State <= ERRORSTATE;
40     end if;
41     CASE State IS
42          WHEN ERRORSTATE =>
43             LED1_State <= '1';
44             LED2_State <= '0';
45             if(RESET_FAULT = '1') then
46              State <= RESETSTATE;
47           end if;
48     WHEN RESETSTATE =>
49             LED1_STATE <= '1';
50             LED2_STATE<= '1';
51             counterLED <= 0;
52                     if( INSTRUCTIONS_RECEIVED ='1') then
53                       State <= DO;
54                     else
55                 if(gotoTESTMODE ='1' and flagTESTMODEleds ='0') then
```

```
56                      leadTESTMODEstateleds <='1';
57                      flagTESTMODEleds <='1';
58                    else
59                       leadTESTMODEstateleds <='0';
60                        if(gotoTESTMODE ='0') then
61                          flagTESTMODEleds <='0';
62                        end if;
63                    end if;
64                    if (leadTESTMODEstateleds ='1' ) then
65                       State <= TESTMODE;
66                    end if;
67              end if;
68    WHEN TESTMODE=>
69                if (counterLED = cntMaxLED) then
70                      LED1_State <= '1';
71                        LED2_State <= '0';
72                  elsif (counterLED = 2*cntMaxLED) then
73                              LED1_State <= '1';
74                            LED2_State <= '1';
75                          elsif(counterLED = 3*cntMaxLED) then
76                                  LED1_State <= '0';
77                                LED2_State <= '0';
78                    end if;
79                 counterLED <= counterLED +1;
80           if( RESET_FAULT ='1' AND counterLED >= 3*cntMaxLED) then
81                      State <= RESETSTATE;
82                  end if;
83    WHEN DO =>
84           LED1_State <= NOT LED1_State;
85           LED2_State <= NOT LED2_State;
86               if (counterLED = cntMaxLEDBlink) then
87                    counterLED <=0;
88                    LED1_State <= NOT LED1_State;
89                      LED2_State <= NOT LED2_State;
90                  else
91                       counterLED <= counterLED +1;
92                  end if;
93        WHEN OTHERS =>
94              State <= RESETSTATE;
```

```
95    END CASE;
96    ELSE
97     LED1_STATE <= '0';
98     LED2_STATE<= '0';
99     State <= RESETSTATE;
100   END IF;
101  END IF;
102 end process;
103
104  LED1 <= LED1_State;
105  LED2<= LED2_State;
106
107
108 end Behavioral;
```

Listing 7.9: LEDS block VHDL code.

7.12 Relays Block VHDL Code

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  entity RELAYS is
7    Port (    clk_5MHz : in STD_LOGIC;
8              ERROR_FPGA: in STD_LOGIC;
9              RESET: in STD_LOGIC ;
10             RESET_FAULT: in STD_LOGIC ;
11             gotoTESTMODE: in  STD_LOGIC ;
12             what_to_do : in STD_LOGIC_VECTOR (3 downto 0);
13             INSTRUCTIONS_RECEIVED :in STD_LOGIC;
14             K1 : out STD_LOGIC;
15             K2 : out STD_LOGIC;
16             K3 : out STD_LOGIC;
17             OpAmpRelays : out STD_LOGIC);
18 end RELAYS;
19
```

```
20 architecture Behavioral of RELAYS is
21 TYPE State_type IS (ERRORSTATE, RESETSTATE, TESTMODE, DO); -- Define
       the states
22 SIGNAL State : State_Type := RESETSTATE;
23
24 signal K1_State : STD_LOGIC := '0';
25 signal K2_State : STD_LOGIC := '0';
26 signal K3_State : STD_LOGIC := '0';
27 signal OpAmpRelays_State : STD_LOGIC := '0';
28 signal what_to_doMemory3:STD_LOGIC := '0';
29 signal what_to_doMemory2:STD_LOGIC := '0';
30 signal what_to_doMemory1:STD_LOGIC := '0';
31 signal what_to_doMemory0:STD_LOGIC := '0';
32
33 signal flagTESTMODE :STD_LOGIC := '0';
34 signal leadTESTMODEstate :STD_LOGIC := '0';
35
36 signal counter : integer := 0;
37 constant cntMaxRelays: integer := 5000000;
38 begin
39
40
41 process(clk_5MHz)
42
43
44 begin
45 IF(rising_EDGE(clk_5MHz)) THEN
46 what_to_doMemory3 <= what_to_do(3);
47 what_to_doMemory2 <= what_to_do(2);
48 what_to_doMemory1 <= what_to_do(1);
49 what_to_doMemory0 <= what_to_do(0);
50     IF(RESET ='0') THEN
51         if(ERROR_FPGA ='1')then
52               State <= ERRORSTATE;
53         end if;
54         CASE State IS
55           WHEN ERRORSTATE =>
56                K1_State <= '0';
57                K2_State <= '0';
```

```
58              K3_State <= '0';
59              OpAmpRelays_State <= '0';
60             if( RESET_FAULT ='1') then
61                State <= RESETSTATE;
62            end if;
63          WHEN RESETSTATE =>
64              K1_State <= '0';
65              K2_State <= '0';
66              K3_State <= '0';
67              OpAmpRelays_State <= '0';
68              counter <= 0;
69              if( INSTRUCTIONS_RECEIVED ='1') then
70                State <= DO;
71              else
72                   if(gotoTESTMODE ='1' and flagTESTMODE ='0') then
73                     leadTESTMODEstate <='1';
74                     flagTESTMODE <='1';
75                   else
76                     leadTESTMODEstate <='0';
77                     if(gotoTESTMODE ='0') then
78                         flagTESTMODE <='0';
79                     end if;
80                   end if;
81                   if (leadTESTMODEstate ='1' ) then
82                     State <= TESTMODE;
83                   end if;
84             end if;
85          WHEN TESTMODE =>
86              if (counter= cntMaxRelays) then
87                 K1_State <= '1';
88              elsif  (counter= 2*cntMaxRelays) then
89                 K2_State <= '1';
90              elsif  (counter= 3*cntMaxRelays) then
91                 K3_State <= '1';
92              elsif  (counter= 4*cntMaxRelays) then
93                  OpAmpRelays_State <= '1';
94              elsif (counter = 5*cntMaxRelays) then
95                 K1_State <= '0';
96              elsif  (counter= 6*cntMaxRelays) then
```

```
97                  K2_State <= '0';
98              elsif  (counter= 7*cntMaxRelays) then
99                  K3_State <= '0';
100             elsif (counter= 8*cntMaxRelays) then
101                 OpAmpRelays_State <= '0';
102             end if;
103             counter <= counter +1;
104           if( RESET_FAULT ='1' AND counter>= 8*cntMaxRelays) then
105                   State <= RESETSTATE;
106                end if;
107         WHEN DO =>
108                 K1_State <= what_to_doMemory0;
109                 K2_State <= what_to_doMemory1;
110                 K3_State <= what_to_doMemory2;
111                 OpAmpRelays_State <= what_to_doMemory3;
112             if( INSTRUCTIONS_RECEIVED ='0') then
113                 State <= RESETSTATE;
114             end if;
115       END CASE;
116    ELSE
117         K1_State <= '0';
118         K2_State <= '0';
119         K3_State <= '0';
120         OpAmpRelays_State <= '0';
121         State <= RESETSTATE;
122    END IF;
123 END IF;
124
125 end process;
126 K1<= K1_State;
127 K2<= K2_State;
128 K3<= K3_State;
129 OpAmpRelays <= OpAmpRelays_State;
130
131 end Behavioral;
```

Listing 7.10: RELAYS block VHDL code.

Bibliography

[1] Serial Peripheral Interface (SPI) - learn.sparkfun.com.

[2] Thomas Ackermann, Thibault Prevost, Vijay Vittal, Andrew J. Roscoe, Julia Matevosyan, and Nicholas Miller. Paving the Way: A Future Without Inertia Is Closer Than You Think. *IEEE Power and Energy Magazine*, 15(6):61–69, November 2017.

[3] AHTOH. SPI (Mastering STM32), February 2018. Library Catalog: blog.radiotech.kz.

[4] Analog Device. AD7276_7277_7278 datasheet, 2015.

[5] L. Ben-Brahim. The analysis and compensation of dead-time effects in three phase PWM inverters. In *IECON '98. Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society (Cat. No.98CH36200)*, volume 2, pages 792–797, Aachen, Germany, 1998. IEEE.

[6] Berkeley Lab. About Microgrids | Building Microgrid, 2019.

[7] Radu Bojoi. Lecture Notes of the course of Power Electronica, Politecnico di Torino. page 88, 2020.

[8] Simone Buso and Paolo Mattavelli. Digital Control in Power Electronics. *Synthesis Lectures on Power Electronics*, 1(1):1–158, January 2006.

[9] Parikshith Channegowda and Vinod John. Filter Optimization for Grid Interactive Voltage Source Inverters. *IEEE Transactions on Industrial Electronics*, 57(12):4106–4114, December 2010.

[10] Guasch componentes y electronica de potencia. MTL-CBI0040F12IXHE_i datasheet.

[11] Hanju Cha, Trung-Kien Vu, and Jae-Eon Kim. Design and control of

Proportional-Resonant controller based Photovoltaic power conditioning system. In *2009 IEEE Energy Conversion Congress and Exposition*, pages 2198–2205, San Jose, CA, September 2009. IEEE.

[12] Infineon. Infineon FS150R07N3E4 datasheet, 2013.

[13] IRENA. Global Energy Transformation: A Roadmap to 2050 (2019 Edition). page 52, 2019.

[14] Joan Rocabert, Alvaro Luna, Frede Blaabjerg, and Pedro Rodríguez. Control of Power Converters in AC Microgrids. *IEEE Transactions on Power Electronics*, 27(11):4734–4749, November 2012. Conference Name: IEEE Transactions on Power Electronics.

[15] Sajjad M. Kaviri, Majid Pahlevani, Praveen Jain, and Alireza Bakhshai. A review of AC microgrid control methods. In *2017 IEEE 8th International Symposium on Power Electronics for Distributed Generation Systems (PEDG)*, pages 1–8, Florianopolis, Brazil, April 2017. IEEE.

[16] Benjamin Kroposki, Brian Johnson, Yingchen Zhang, Vahan Gevorgian, Paul Denholm, Bri-Mathias Hodge, and Bryan Hannegan. Achieving a 100% Renewable Grid: Operating Electric Power Systems with Extremely High Levels of Variable Renewable Energy. *IEEE Power and Energy Magazine*, 15(2):61–73, March 2017.

[17] Xiaoqiang Li, Pengfeng Lin, Yi Tang, and Kai Wang. Stability Design of Single-Loop Voltage Control With Enhanced Dynamic for Voltage-Source Converters With a Low LC -Resonant-Frequency. *IEEE Transactions on Power Electronics*, 33(11):9937–9951, November 2018.

[18] David Lumbreras, Ernesto L. Barrios, Andoni Urtasun, Alfredo Ursua, Luis Marroyo, and Pablo Sanchis. On the Stability of Advanced Power Electronic Converters: The Generalized Bode Criterion. *IEEE Transactions on Power Electronics*, 34(9):9247–9262, September 2019.

[19] Julia Matevosyan, Vijay Vital, Jon O'Sullivan, Ryan Quint, Babak Badrzadeh, Thibault Prevost, Eckard Quitmann, Deepak Ramasubramanian, Helge Urdal, Sebastian Achilles, Jason MacDowell, and Shun Hsien Huang. Grid-Forming Inverters: Are They the Key for High Renewable Penetration? *IEEE Power and Energy Magazine*, 17(6):89–98, November 2019.

[20] Mario Ndreko, Sven Rüberg, and Wilhelm Winter. Grid Forming Control for Stable Power Systems with up to 100 % Inverter Based Generation: A Paradigm Scenario Using the IEEE 118-Bus System. page 6, 2018.

[21] Dinesh Pattabiraman, R. H. Lasseter., and T. M. Jahns. Comparison of Grid Following and Grid Forming Control for a High Inverter Penetration Power System. In *2018 IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5, Portland, OR, August 2018. IEEE.

[22] PEIC. Guasch Project PCB, 2020.

[23] Qing-Chang Zhong, Phi-Long Nguyen, Zhenyu Ma, and Wanxing Sheng. Self-Synchronized Synchronverters: Inverters Without a Dedicated Synchronization Unit. *IEEE Transactions on Power Electronics*, 29(2):617–630, February 2014.

[24] Remus Teodorescu, Marco Liserre, and Pedro Rodrŧguez. *Power System Stability And Control by Prabha Kundur.pdf*. 2011 edition, 1993.

[25] Pedro Rodriguez. Fundamentals of Grid Forming Converters. page 67, 2019.

[26] Remus Teodorescu, Frede Blaabjerg, and Marco Liserre. Proportional-Resonant Controllers. A New Breed of Controllers Suitable for Grid-Connected Voltage-Source Converters. *Journal of Electrical Engineering*, page 6, 2004.

[27] Texas Instruments. DAC124S085 datasheet, April 2016.

[28] Trenz Electronic. TRM-TE0725-03 datasheet, June 2018.

[29] Xiongfei Wang, Poh Chiang Loh, and Frede Blaabjerg. Stability Analysis and Controller Synthesis for Single-Loop Voltage-Controlled VSIs. *IEEE Transactions on Power Electronics*, 32(9):7394–7404, September 2017.

[30] Dongsheng Yang, Heng Wu, Xiongfei Wang, and Frede Blaabjerg. Suppression of synchronous resonance for VSGs. *The Journal of Engineering*, 2017(13):2574–2579, January 2017.

[31] YANGDI HE. *A comparison of modulation techniques and motor performance evaluation*. Master thesis, Chalmers University of technology, Sweden, 2018.

[32] Lidong Zhang, Lennart Harnefors, and Hans-Peter Nee. Power-Synchronization Control of Grid-Connected Voltage-Source Converters. *IEEE Transactions on Power Systems*, 25(2):809–820, May 2010.