POLITECNICO DI TORINO

Master of Science in Electronic Engineering

Master Thesis Dissertation

High-performance Digital Control of Power Converters

Supervisors
Prof. Salvatore Musumeci
Dr. Fabio Mandrile

Candidate
Samuele Fabbri

October 2022

Summary

During the implementation of a power converter digital control system, one of the usual main drawbacks is the required programming time which introduces a delay between the system design and simulation and the experimental phase. An effective way to minimize it is achieved by specific tools which automatically generate the needed code for the target platform, hence, for instance, either C code for a microcontroller or HDL code for an FPGA. In this thesis, the Simulink HDL Coder tool effectiveness is validated for the automatic generation of DC/DC and DC/AC power converters digital control systems VHDL codes. In particular, two different high-performance digital average current control systems are designed and simulated on Simulink. Then HDL Coder is used to automatically generate the corresponding VHDL code. At last, the control systems are implemented on an FPGA and their correct operation is experimentally tested on the corresponding power converter. After addressing the main digital average current control issues, a PWM modulator and a current controller are implemented for a four-quadrant DC/DC H-bridge converter. Then, the same system is extended for the current control of a DC/AC three-phase inverter, designing a carrier-based space vector PWM modulator. In both modulation cases, a multisampling double-update strategy is exploited. Moreover, a dSPACE rapid prototyping system is used to provide a visual user interface, where the main reference and control signals can be visualized. As a final application, the digital current control system is adapted to the torque control of a permanent magnet assisted synchronous reluctance motor.

Acknowledgements

This thesis represents the conclusion of my master's degree journey. I would like to thank my supervisors, Prof. Salvatore Musumeci and Dr. Fabio Mandrile, who have guided and helped me in this work. I would also like to thank all the people at the PEIC laboratory of the Politecnico di Torino who I had the pleasure to meet. A heartfelt thanks to my family that has always supported me in every possible manner during this journey. Finally, I would like to thank all my friends who have made this experience unforgettable.

To my family.

Contents

List of Tables

List of Figures

Chapter 1

Introduction to Digital Control - Average Current Mode Control

In each power electronics system, independently of the kind of power converter which is considered, the overall performance is affected by both the converter and the control system designs. The choice of the control strategy and its actual implementation are therefore fundamental in these systems. In particular, a good control system should mainly be able to provide a [1]:

- stable closed-loop system, to avoid diverging responses;
- very low, ideally null, error in steady state, in order to be able to perfectly track a reference;
- high dynamic response, to fastly modify the variable under control after reference variations.

From a first implementation perspective, the controller can be both analog or digital: even though the former one is the best for what concerns the system speed, the latter introduces a series of advantages that are hereby reported [2]:

- possibility to implement sophisticated control laws;
- correction of nonlinearities, parameter variations and construction tolerances by means of auto-tuning strategies;
- reprogrammability of the controller without the need of hardware modifications;
- absence of thermal drifts and ageing effects.

It is therefore evident how digital control offers important functionalities, when compared to an analog one, which can be essential in specific power electronic applications. For this reason, in the following sections a digital control strategy and its design are presented: first, a test case, namely a two-quadrant DC/DC converter, is analyzed. Secondly,

a review of digital current control is reported, in order to understand its main peculiarities and issues. Finally, the proposed control system, which includes a current controller based on a PI regulator and a pulse width modulation (PWM) technique, is presented.

1.1 Case Study: Two-Quadrant DC/DC Converter for Electric Motor Driving

In order to better understand the main features and characteristics of digital current control, it is useful to focus on a specific converter topology: for this reason, the two-quadrant DC/DC converter depicted in Figure 1.1 is considered and will be analyzed in this section. It is anyway important to point out that the same concepts can be applied and adapted also to other power converters. The system is composed of the following elements:

- input DC voltage source V_{in};
- two IGBTs, with free-wheeling diodes, implementing the switching leg;
- inductance L, with its ESR R;
- output DC voltage source E.

For example, taking into account the particular kind of LRE load, this can be the case of a converter connected to an electric motor [3][4], where the output voltage source represents the back-emf of the motor. Finally the following conditions, independently of the actual inductor current value and direction, are always guaranteed:

$$\begin{cases} V_{in} > E + R|i(t)| \\ E > R|i(t)| \end{cases} \tag{1.1}$$

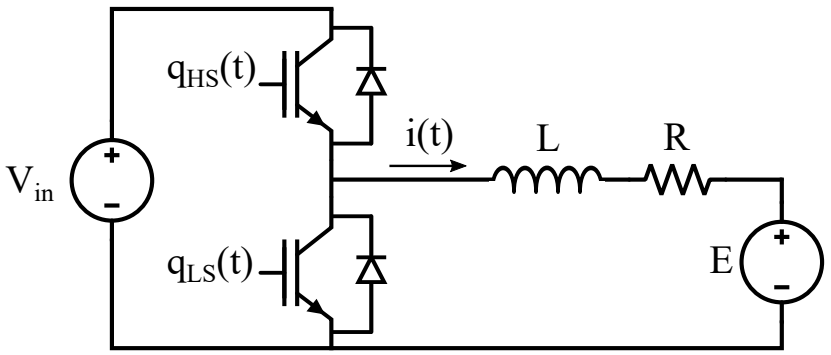


Figure 1.1: Two-quadrant DC/DC converter topology with LRE load.

Assuming the two IGBTs as ideal switches, their switching operation can be summarized by their switching functions [4], which are defined in (1.2) and (1.3).

$$q_{HS}(t) = \begin{cases} 1, \text{ when the high-side IGBT is on} \\ 0, \text{ when the high-side IGBT is off} \end{cases} \tag{1.2}$$

$$q_{LS}(t) = \begin{cases} 1, \text{ when the low-side IGBT is on} \\ 0, \text{ when the low-side IGBT is off} \end{cases} \tag{1.3}$$

It is important to highlight that these switching functions are complementary, as depicted in Figure 1.2 in order to avoid the simultaneous activation of the transistors which would lead to the short circuit of the input voltage source.

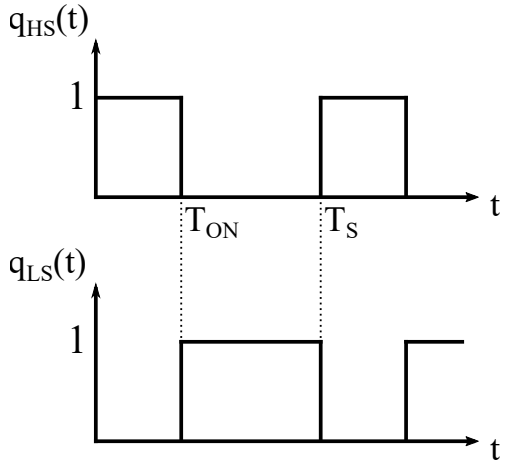


Figure 1.2: High-side and low-side IGBTs switching functions example.

Considering a switching period, the duty cycle of the converter can be defined as the ratio between the high-side IGBT on-time and the switching period, as reported in (1.4).

$$d = \frac{T_{ON}}{T_S} \tag{1.4}$$

Generally, the on-time of each transistor is not fixed but can be for example adjusted by the action of a controller. This means that the T_{ON} value, and consequently the duty cycle, is not constant. Due to this mechanism, the duty cycle of the converter becomes a function of time and is conveniently defined as the moving average of the high-side transistor switching function considering a switching period, as it is summarized in (1.5) [2].

$$d(t) = \frac{1}{T_S} \int_t^{t+T_S} q_{HS}(\tau) \mathrm{d}\tau \tag{1.5}$$

1.1.1 Converter Working Principle

The converter operation can be explained referring to [2]. In an ideal case, the two IGBTs act as ideal switches. If this assumption is valid, the equivalent model depicted in Figure 1.3 can be derived, noticing that when the high-side IGBT is turned on the voltage applied to the inductance, denoted in the figure as $v^*(t)$, is V_{in}, whereas when the low-side IGBT is active the applied voltage is null. Hence, the behaviour of $v^*(t)$ is that of a square wave and can be summarized in (1.6) considering the switching function of the high-side trasistor reported in (1.2).

$$v^*(t) = q_{HS}(t) \cdot V_{in} \tag{1.6}$$

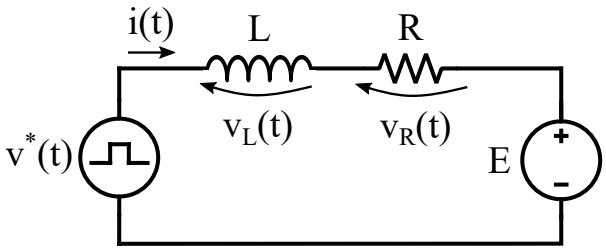


Figure 1.3: Two-quadrant DC/DC converter with LRE load large-signal equivalent model.

The total voltage $v_L(t)$ which is applied to the inductor is derived in (1.7):

$$v_L(t) = v^*(t) - v_R(t) - E = v^*(t) - Ri(t) - E \tag{1.7}$$

As can be seen, it depends both on the current i(t) flowing in the inductance and on the voltage $v^*(t)$ applied to the load. Recalling also (1.1), the following two cases can be obtained depending on the possible IGBTs states.

1. $v_L(t) = V_{in} - Ri(t) - E > 0$, which gives an increase of the inductor current, when the high-side IGBT is active and the low-side IGBT is turned off;

2. $v_L(t) = -Ri(t) - E < 0$, which gives a decrease of the inductor current, when the high-side IGBT is turned off and the low-side IGBT is active.

1.1.2 Converter and Load DC Equivalent Model

Considering the average values of the system quantities, the DC equivalent circuit represented in Figure 1.4 can be derived. In order to do so, the average value of $v^*(t)$ is computed in (1.8) considering the duty cycle to be constant and exploiting (1.4). Then, by looking at the equivalent circuit, the duty cycle value in steady-state condition can be

easily derived as shown in (1.9), which highlights the dependence between the average current flowing in the load and the corresponding duty cycle value.

$$V^* = \frac{1}{T_S}\int_0^{T_S} q_{HS}(t)V_{in}\,\mathrm{dt} = \frac{1}{T_S}V_{in}\int_0^{T_{ON}}\mathrm{dt} = \frac{T_{ON}}{T_S}V_{in} = dV_{in} \tag{1.8}$$

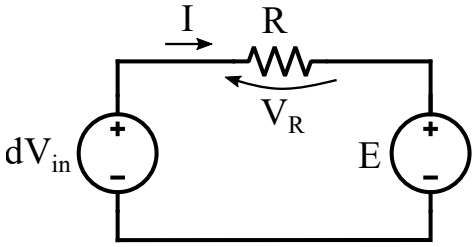


Figure 1.4: Two-quadrant DC/DC converter and LRE load DC equivalent model.

$$dV_{in} = V_R + E \leftrightarrow d = \frac{V_R + E}{V_{in}} = \frac{RI + E}{V_{in}} \tag{1.9}$$

As aforementioned, the duty cycle is in general a function of time. For instance, it can be modified as the result of a controller operation or if a different load average current is needed. For this reason, the DC equivalent model represented in Figure 1.4 is modified accordingly considering the moving averages in Figure 1.5, which underlines the dependence over time of both duty cycle and load average current. Exploiting (1.5), the moving average of $v^*(t)$ is computed in (1.10) which shows that a variation of the converter duty cycle directly implies a variation of the average voltage which is applied to the load.

$$V^*(t) = \frac{1}{T_S}\int_t^{t+T_S} q_{HS}(\tau)V_{in}\mathrm{d}\tau = \frac{1}{T_S}V_{in}\int_t^{t+T_S} q_{HS}(\tau)\mathrm{d}\tau = d(t)V_{in} \tag{1.10}$$

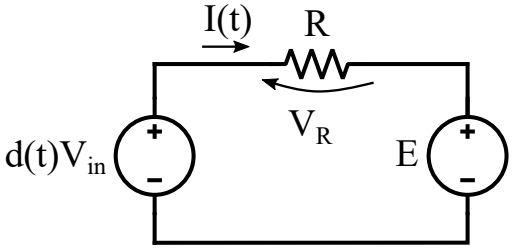


Figure 1.5: Two-quadrant DC/DC converter and LRE load DC equivalent model with moving averages.

1.1.3 Load Output Filtering Operation

An important consideration regards the filtering operation provided by the LR series at the output, as explained in [2]. Starting from Figure 1.3 and applying the Laplace transform, the equivalent model depicted in Figure 1.6 is obtained. This can be then exploited to easily compute the transfer function from the load input to the load current, as reported in (1.11).

$$\left.\frac{i(s)}{v^*(s)}\right|_{E(s)=0} = \frac{1}{sL+R} \tag{1.11}$$

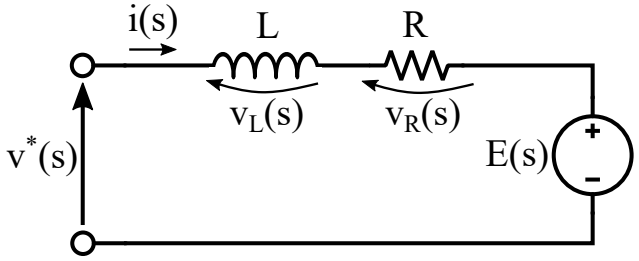


Figure 1.6: Two-quadrant DC/DC converter Laplace-domain equivalent model.

As can be seen, (1.11) presents a low-pass filtering behavior, leading to the fact that the load current average in the switching period is capable of tracking the load voltage average value variations. Anyway, since $v^*(t)$ is a square wave and the low-pass filtering operation is not ideal, the instantaneous output current also shows a ripple due to the residual presence of the switching frequency component, along with its harmonics, in the spectrum.

In order to derive the transfer function from duty cycle to load current, first the moving averages need to be considered for the variables in the model depicted in Figure 1.6. Then, the Laplace transform is applied to (1.10), giving (1.12).

$$V^*(s) = d(s)V_{in} \tag{1.12}$$

The equivalent model depicted in Figure 1.7 can be then immediately obtained. This is useful to easily compute the wanted transfer function in (1.13).

$$\left.\frac{I(s)}{d(s)}\right|_{E(s)=0} = \frac{V_{in}}{sL+R} \tag{1.13}$$

As shown, (1.13) presents again a low-pass filtering behavior. This proves that, in particular, the load current moving average tracks the duty cycle variations.

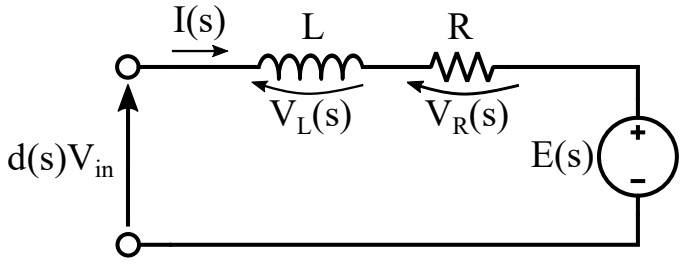


Figure 1.7: Two-quadrant DC/DC converter Laplace-domain equivalent model with moving averages.

1.1.4 Converter and Load Equivalent Discrete Model

For the design of the current controller and for simulation reasons that will be more clear in the following chapters, it is useful to derive the converter and load equivalent discrete model. Recalling (1.6) and (1.7), (1.14) is derived.

$$\begin{cases} v_L(t) = L\frac{\mathrm{d}i(t)}{\mathrm{d}t} \\ v^*(t) = q_{HS}(t)V_{in} \\ v_L(t) = v^*(t) - Ri(t) - E \end{cases} \tag{1.14}$$

Replacing the first and second equation in the third one and rearranging its terms, the expression reported in (1.15) is derived.

$$\frac{\mathrm{d}i(t)}{\mathrm{d}t} = \frac{-R}{L}i(t) + \frac{v^*(t) - E}{L} = \frac{-R}{L}i(t) + \frac{q_{HS}(t)V_{in} - E}{L} \tag{1.15}$$

The equation notation can be simplified by defining $\Delta v(t) = q_{HS}(t)V_{in} - E$, obtaining (1.16).

$$\frac{\mathrm{d}i(t)}{\mathrm{d}t} = \frac{-R}{L}i(t) + \frac{\Delta v(t)}{L} \tag{1.16}$$

Considering the equivalent model depicted in Figure 1.3, the two-quadrant DC/DC converter and load cascade can be described by the continuous time-invariant state-space equations reported in (1.17) [2][3][5][6].

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases} \tag{1.17}$$

All the state-space vectors and matrices, which are reported in (1.18), can be directly found by comparing (1.16) with the equations reported in (1.17). Notice that, for this

particular case, the state space vectors and matrices are reduced to one dimension.

$$\begin{cases} x(t) = [i(t)] \\ y(t) = [i(t)] \\ u(t) = [\Delta v(t)] \\ A = [\frac{-R}{L}] \\ B = [\frac{1}{L}] \\ C = [1] \\ D = [0] \end{cases} \tag{1.18}$$

The state-space model reported in (1.17) can be discretized assuming the input $u(t)$ to be constant for a sampling interval T_s, which is sufficiently small, obtaining (1.19)[2].

$$\begin{cases} x_{k+1} = A_d x_k + B_d u_k \\ y_k = C_d x_k + D_d u_k \end{cases} \tag{1.19}$$

where, in the considered case:

$$\begin{cases} A_d = e^{AT_s} \\ B_d = \int_0^{T_s} e^{At} \mathrm{d}t = A^{-1}(A_d - I)B \\ C_d = C \\ D_d = D \end{cases} \tag{1.20}$$

Therefore, for this particular system in which the continuous state space matrices and vectors are respectively simple constants and variables, (1.21) can be immediately derived, leading to the discrete equivalent model block scheme represented in Figure 1.8 exploiting also (1.19).

$$\begin{cases} A_d = e^{AT_s} = e^{\frac{-R}{L}T_s} \\ B_d = A^{-1}(A_d - I)B = \frac{-L}{R}(e^{\frac{-R}{L}T_s} - 1)\frac{1}{L} = \frac{1-e^{\frac{-R}{L}T_s}}{R} \\ C_d = C = 1 \\ D_d = D = 0 \end{cases} \tag{1.21}$$

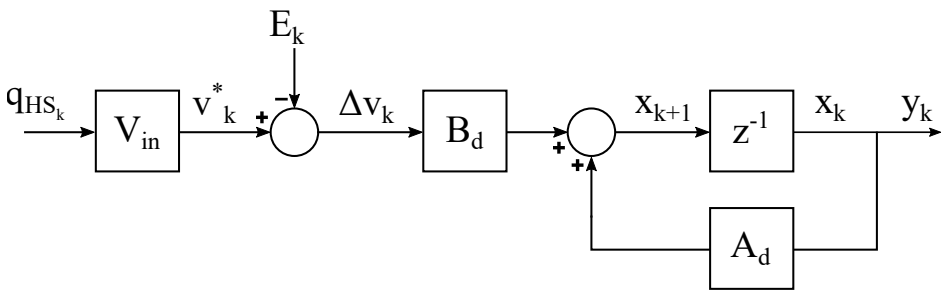


Figure 1.8: Two-quadrant DC/DC converter with LRE load discrete time block scheme.

1.2 Digital Current Control Theory

In order to understand the main issues and features of digital current control, in this section a review of the analog PWM modulator is provided. After that, the main characteristics of digital current control are presented, starting from the digital implementation of the PWM modulator and a comparison with the analog one. The analysis hereby reported is mainly inspired by [2]. To explain these concepts, the two-quadrant DC/DC converter with LRE load depicted in Figure 1.1 is assumed as a title of example. Notice that anyway all the concepts can be adapted also to other power converters. In the following discussions, the variable to be controlled is always the inductor current highlighted in the figure as i(t).

1.2.1 Analog PWM

As explained in [2], among the different modulation techniques that can be exploited for power converters, pulse width modulation (PWM) is certainly the most widely used. Its main advantages are the ease of implementation and the constant converter frequency operation, which allows easier output low-pass filters design with respect to other techniques, such as pulse frequency modulation for instance. The outputs of the PWM modulator are the switching functions of the converter transistors. In the analog case, it is basically composed of:

- a comparator, which compares a carrier signal, that can be either a trailing-edge or leading-edge sawtooth waveform, or a triangular waveform, with a control voltage signal given by the controller;
- a logic inverter, whose input is the output of the comparator.

Notice that, in a real case, gate drivers are needed to provide the correct voltage level to the transistors of the converter in order to properly turn them on.

In Figure 1.9, an example of analog PWM modulator block scheme and possible waveforms exploiting a triangular waveform carrier is depicted. In the figure, the output of the comparator provides the switching function of the high-side switch $q_{HS}(t)$, whereas through the inverter the switching function for the low-side transistor $q_{LS}(t)$ is generated. Furthermore, the presence of gate drivers, necessary in real applications, is highlighted. Notice that the variation of the control voltage $v_{contr}(t)$ is exaggerated in the figure to emphasize its effect on the pulses width: in practical cases, the bandwidth of $v_{contr}(t)$ is well below the switching frequency $1/T_S$, meaning that it slowly changes inside a switching period.

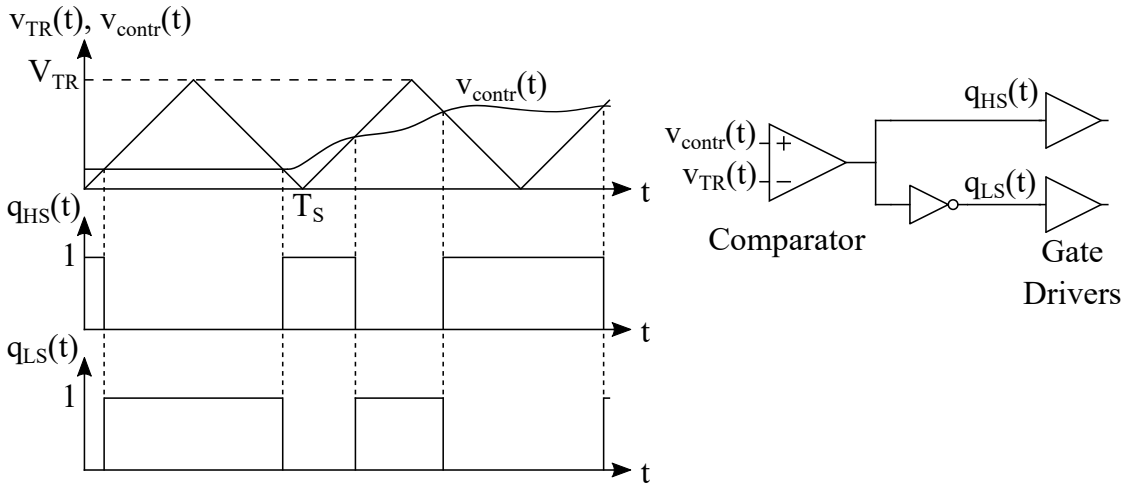


Figure 1.9: Analog PWM modulator: example waveforms and block scheme implementation.

In the case of an analog current control implementation, $v_{contr}(t)$ is properly generated by a current controller in order to make the controlled variable to follow a certain current reference signal. Since the control voltage is allowed to be modified in every time instant of the PWM modulator operation, when its amplitude changes the comparator is able to modify its output accordingly inside the same switching period T_S where the variation occurred. This means that the delay introduced by the PWM modulator between control voltage variations and converter duty cycle adjustments is the minimum one, ideally null, and can be therefore neglected.

A final important consideration regards the relationship between the duty cycle of the high-side switch and the control voltage signal. By looking at one modulation period, (1.22) can be found considering a constant control voltage signal V_{contr}, giving a constant duty cycle.

$$d = \frac{V_{contr}}{V_{TR}} \tag{1.22}$$

The result is still correct for a varying $v_{contr}(t)$ if its bandwidth is much lower than the switching frequency, which is usually the case, as aforementioned. This means that, under this consideration, the duty cycle will follow the control voltage signal evolution, carrying its informative content. This is a fundamental aspect, since as it was demonstrated in the previous section, the load current is able to track the duty cycle variations, which in turn follows the trajectory of the control voltage. As a result, by properly generating $v_{contr}(t)$, it is possible to correctly control the load current. Furthermore, another interesting feature appears when the amplitude of the triangular carrier waveform V_{TR} is unitary: in fact, in this case, the control voltage signal provided by the current controller is identical, in terms of value, to the wanted duty cycle.

1.2.2 Uniformly Sampled Digital PWM

As discussed in [2], the easiest way to implement a digital PWM modulator is to start from its analog implementation and replace each block with its corresponding digital one. For instance, referring to the example reported in Figure 1.9, a binary counter is exploited to generate the triangular waveform and relational operators can be deployed to implement the functionality of comparator and logic inverter. This can be easily done with a microcontroller, Digital Signal Processor (DSP) or Field Programmable Gate Array (FPGA). A block scheme deploying these modifications and some possible operation waveforms for the digital PWM modulator case are shown in Figure 1.10.

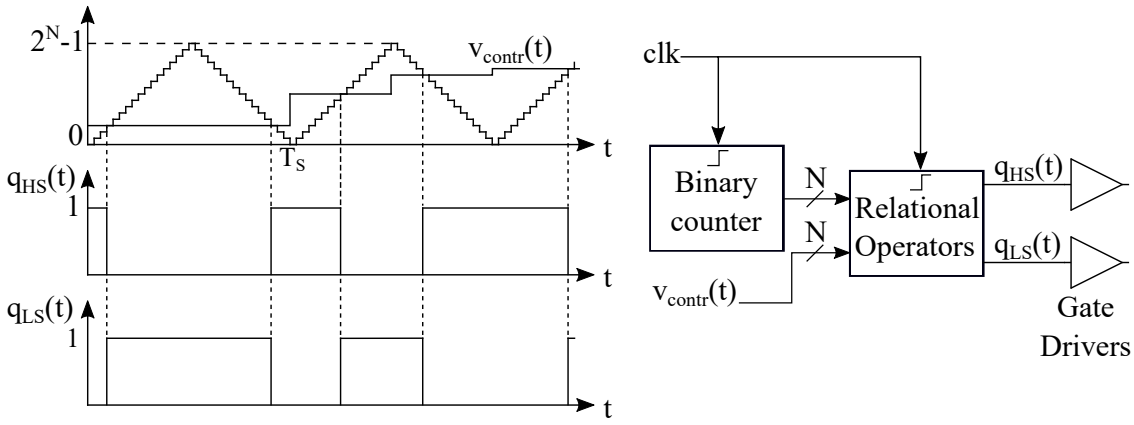


Figure 1.10: Digital PWM modulator: example waveforms and block scheme implementation.

As can be seen, the operation is almost the same as that of the analog case. Anyway, some differences need to be commented: first of all, being in digital domain, the triangular carrier and the control voltage can only assume discrete values, which basically bring a precision loss in the duty cycle value which is applied to the converter depending on the number of bits N on which they are represented. Furthermore, since the switching

frequency of the devices inside the converter depends on that of the triangular carrier, the number of bits N and the clock frequency have a strong impact on the performance: if the clock frequency is fixed, in order to have a certain switching frequency a specific number of bits for the triangular waveform is needed; this means that the number of bits for representing the output values could be limited, leading to a poor resolution. This last issue can be solved if for example a Phase-Locked Loop (PLL) is available, since it allows to increase the clock frequency and, consequently, to use a higher number of bits for the triangular carrier and control voltage representation.

Second of all, for reasons that will be motivated later, in the average current digital control case the update of the control voltage is allowed only in specific instants. These coincide, in the so called double-update mode, with the moments in which the maximum or minimum value of the triangular carrier is reached, as shown in Figure 1.10. Anyway, independently of the strategy which is exploited, the control voltage amplitude is not continuously updated.

Finally, a last important issue regards the delay: if infinite resolution is assumed for both the triangular carrier and the control voltage, the operation of the digital PWM resembles that of an analog one where the control voltage is kept constant until the following update instant. Notice that this problem cannot be neglected: in the analog case, the control voltage is allowed to vary in every time instant inside the switching period, hence with no time delay; on the other hand in the digital PWM case, as aforementioned, control voltage changes are permitted only at fixed instants inside the switching period, depending on the selected updating mode. This means that the control voltage variations due to system disturbances or due to a change in the reference signal is effectively applied with a certain delay, denoted as PWM modulator delay, which depends on the updating strategy. This is crucial since this implies an increase in the digital system response delay, leading to a lower phase margin and then to a lower control system bandwidth.

1.2.3 Sampling and Updating Strategies

When deploying a digital current controller, one of its fundamental aspects to be designed is a proper data acquisition path, which is basically composed of a current sensor, a signal conditioning circuit and an ADC. In particular, a uniformly sampled signal can be exactly reconstructed if the Shannon's sampling theorem is not violated, hence when the signal bandwidth is at most equal to half the sampling frequency of the ADC. Considering that the inductor current in the converter is affected by the high-frequency ripple, the sampling frequency should be sufficiently higher than the switching frequency in order to be able to correctly reconstruct the signal. In general, as will be explained in the following, the whole control system delay can be reduced by increasing the sampling frequency, leading therefore to a higher bandwidth.

As reported in [2][7], many sampling and updating strategies are possible. A conventional solution is to implement a single-sampling single-update (SSSU) or a double-sampling double-update (DSDU) mode, in which the sampling frequency is set equal to either the switching frequency or twice its value respectively. Notice that this always leads to a violation of the Shannon's theorem as explained above. However, this is not a drawback by means of synchronization: in fact, referring to the system in Figure 1.1, the average value of the inductor current can be automatically reconstructed, simplifying the overall control system design since low-pass filters are not needed to remove the ripple, if a precise sampling instant is chosen: in a steady-state situation, the current ripple is such that, if a switching period is considered, the initial and final value of the current are equal. This means that in the middle of the time interval where the inductor current slope is positive or negative, the value of the inductor current is exactly its average value. Notice that, in the considered type of converter, this is equivalent to say that the sampling instants should be in the middle of the time intervals in which the high-side or low-side IGBTs are respectively active. As shown in Figure 1.11, the synchronization condition in the steady-state situation corresponds to sampling the current when the triangular waveform reaches either its maximum or minimum value, in single-sampling mode, or both of them, in double-sampling mode.

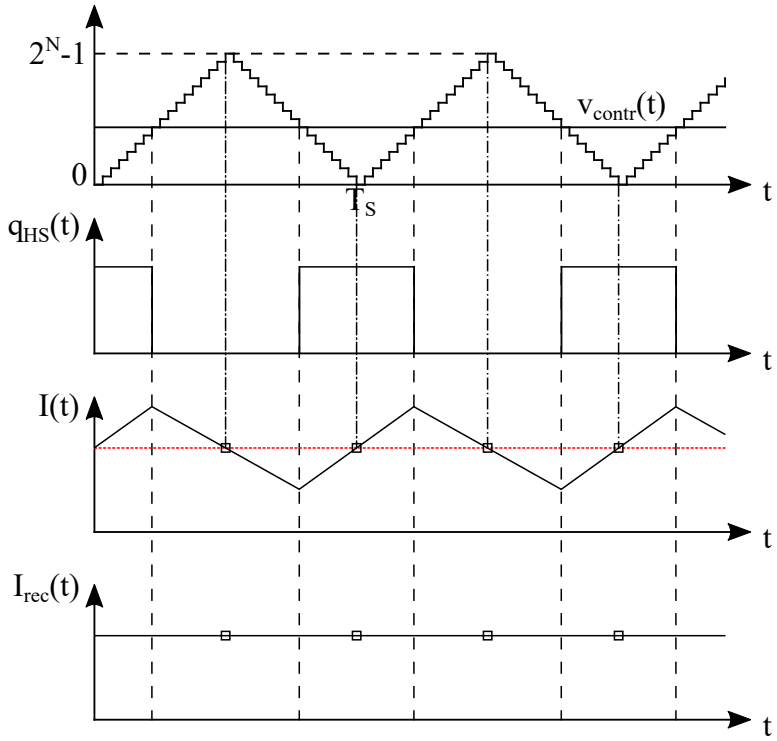


Figure 1.11: Example of controlled current and reconstructed average value with synchronized sampling and switching processes.

An important point which is stressed in [8] for what concerns the PWM modulator is that the comparison laws between the digital control voltage representation and the triangular carrier depend on the run-up and run-down phases of the latter one, in order to be able to provide symmetrical switching functions with respect to the sampling instant. This is a crucial aspect that is needed to guarantee that the sample exactly corresponds to the average inductor current value. To highlight this consideration, referring to the examples depicted in Figure 1.10 and Figure 1.11, the comparison laws which are needed to generate the switching functions $q_{HS}(t)$ and $q_{LS}(t)$ are summarized in Table 1.1, considering the control voltage as the first operand and the triangular carrier as the second one.

	Run-up phase	Run-down phase
$q_{HS}(t)$	$>$	$\geq$
$q_{LS}(t)$	$\leq$	$<$

Table 1.1: Digital PWM modulator comparison laws depending on run-up and run-down phases of the triangular carrier.

Nevertheless, aliasing effects can still be present whether the sampling instant is not precise or the sampling frequency is not exactly equal to once or twice the switching frequency. For this reason, an integral action is typically present in the current controller, in order to compensate any steady-state error that can be eventually present.

As aforementioned, the purpose of the current controller is to generate a proper control voltage $v_{contr}(t)$ starting from the comparison between the controlled variable and a reference signal. As opposed to the analog current controller, in the digital case its operation is triggered with a frequency which is at most equal to the sampling frequency, hence only when new sampled data is available. This is also due to the fact that, when the sampling trigger signal is received by the ADC, a certain amount of time elapses from the sampling event until the sampled data is available to the current controller. The ADC delay is given by two aspects:

1. The ADC operation is not instantaneous but requires a certain amount of time to complete the conversion;

2. Depending on the communication protocol, the sampled data is transferred to the current controller. In case of a serial communication protocol, the transmission time is strongly influenced by the ADC resolution, hence by the number of bits on which the sampled quantity is represented.

In addition to the ADC delay, the execution of the current controller introduces a computational delay which is necessary to determine the correct control voltage, and duty cycle value, corresponding to the sampled data. The presence of these two delay contributions

makes it impossible to sample the average inductor current and simultaneously update the control voltage with the corresponding value.

The update of the control voltage value can be allowed right after the end of the current controller execution; anyway, this may lead to a double-crossing of the triangular carrier, compromising the switching functions pulses symmetry with respect to the sampling instants, which is beneficial for the output spectrum, and incrementing the number of switching events that occur in a switching period, increasing the power dissipation. For these two reasons, it is useful to update the control voltage at the following sampling instant. In order to do so, the sum of the ADC delay and the computational time needs to be smaller than T_S and $T_S/2$ in the SSSU and DSDU modes respectively.

It is important to highlight that in the discussion only one ADC, for the controlled variable, has been considered; in general, more than one sampled variable is required for the current controller execution. Depending on the system specifications and on the available hardware, this may become a limiting factor for the sampling and updating strategy in certain applications.

Two summarizing examples of the SSSU mode where the sampling instant coincides with the maximum and the minimum value of the triangular carrier are depicted in Figure 1.12a and Figure 1.12b respectively, whereas a DSDU mode scheme is reported in Figure 1.13. The dashed arrows link each sampling instant with the instant in which the control voltage is updated with the corresponding value.

Without considering the ADC delay, the digital current controller computational time and the PWM modulator delay can be computed for the two strategies, as shown in [2] and [7]. In particular, they are reported in (1.23) and (1.24) for the SSSU and DSDU modes respectively, considering a triangular carrier. As can be immediately derived, the total control system delay is halved in the DSDU case with respect to the SSSU one, leading to a higher control system bandwidth. The result is still valid if the ADC delay is considered as long as the sum between the digital current controller computational time and the ADC delay is smaller than T_S, for the SSSU mode, and $T_S/2$, for the DSDU mode.

$$t_{SSSU} = t_{comp\_SSSU} + t_{PWM\_SSSU} = T_S + \frac{T_S}{2} = 1.5T_S \tag{1.23}$$

$$t_{DSDU} = t_{comp\_DSDU} + t_{PWM\_DSDU} = \frac{T_S}{2} + \frac{T_S}{4} = 0.75T_S \tag{1.24}$$

Another strategy which can be adopted is that of multisampling multiupdate (MSMU) [2][7]. In this solution, the inductor current is sampled N_{MS} times, where N_{MS} is an integer number higher than 2, within a switching period T_S and, consequently, the control variable is updated N_{MS} times in the same time interval. An example is reported in Figure

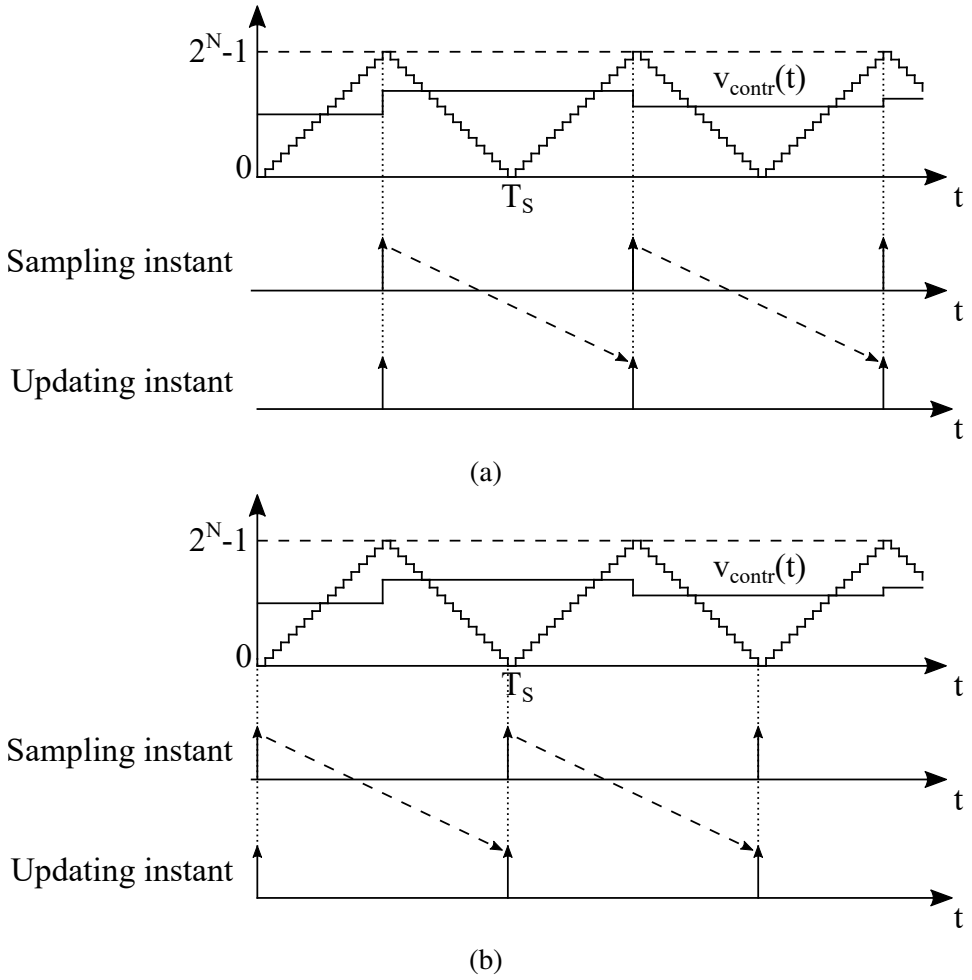


Figure 1.12: Single-sampling single-update mode synchronized with the maximum (a) and minimum (b) of the triangular carrier example.

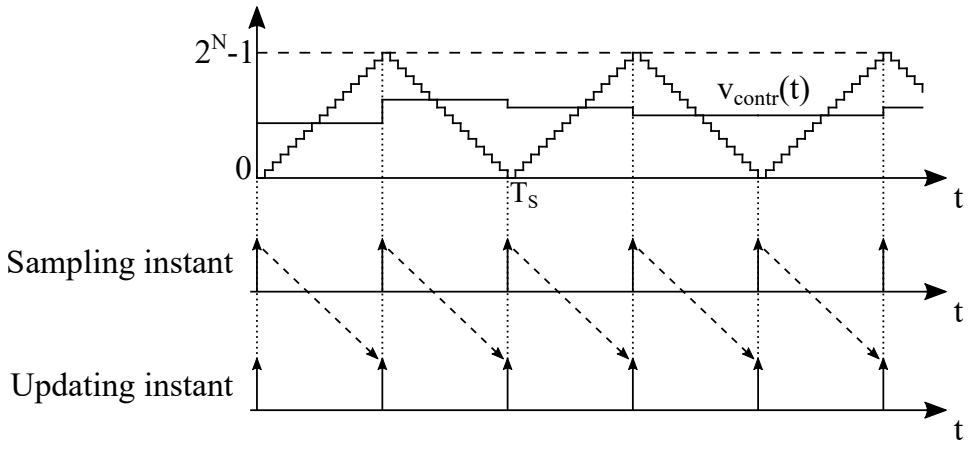


Figure 1.13: Double-sampling double-update mode example.

1.14, in which the dashed arrows link each sampling instant with the instant in which the control voltage is updated with the corresponding value. Notice that, in this case, the sum between ADC delay and current controller computational time needs to be smaller than T_S/N_{MS}.

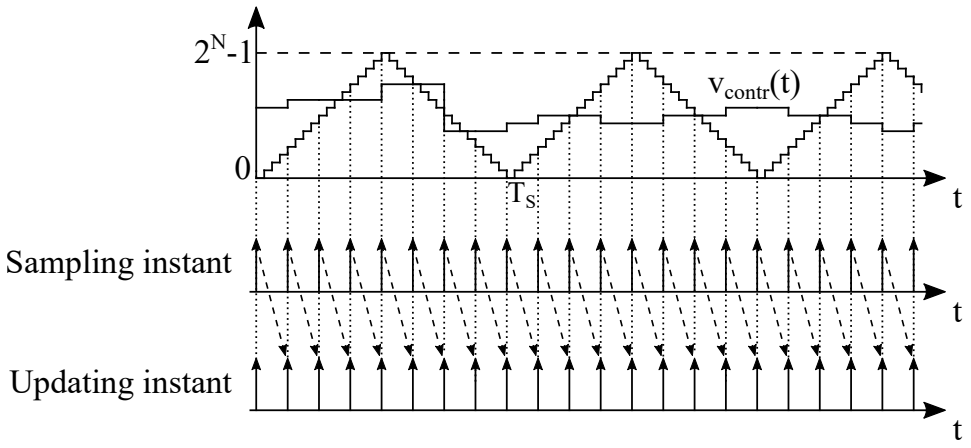


Figure 1.14: Multisampling multiupdate mode example.

As reported in [7], the total control system delay, considering the current controller computational time and the PWM modulator delay and neglecting the ADC delay, can be expressed in (1.25).

$$t_{MSMU} = t_{comp\_MSMU} + t_{PWM\_MSMU} = \frac{T_S}{N_{MS}} + \frac{T_S}{2N_{MS}} = \frac{1.5T_S}{N_{MS}} \tag{1.25}$$

Notice that this expression summarizes also the SSSU and DSDU strategies for N_{MS} equal to 1 and 2 respectively. By looking at the expression, the control system delay can be reduced in this case with respect to the previous ones by increasing the multisampling factor N_{MS} and, consequently, the sampling and updating frequency. This conclusion can still be drawn if the ADC delay is considered, as long as the sum between ADC delay and current controller computational time is smaller than T_S/N_{MS}. Ideally, the delay can also be null: in fact, as N_{MS} approaches infinity, the behaviour of the PWM modulator becomes that of the analog case, in which the delay can always be considered null, as previously explained.

However, this strategy presents a drawback: in order to obtain the inductor average current value, the switching ripple of the current needs to be filtered as opposed to the previously reported modes, needing then a more complicated system with respect to the other two cases. As explained in [7], the most effective way to achieve this operation is to implement a moving average filter (MAF) considering only the samples inside a switching period, even if this technique leads to an increase in the control system delay equal to $T_S/2$.

Finally, in the multiupdate strategy case the drawback of multiple intersections between control voltage and triangular carrier is present, as shown in the example depicted in Figure 1.14: as previously explained, this should be avoided in many applications. For these reasons, in the following chapters a multisampling double-update (MSDU) strategy will be implemented, since it does not present the multiple intersections problem and, as explained in [7], considering the moving average filter, the control system delay is comparable with that of the DSDU one for a high multisampling factor N_{MS}. The MSDU mode total control delay expression is reported in (1.26).

$$t_{MSMU} = t_{comp\_MSDU} + t_{PWM\_MSDU} + t_{MAF} = \frac{T_S}{N_{MS}} + \frac{T_S}{4} + \frac{T_S}{2} = 0.75T_S + \frac{T_S}{N_{MS}} \quad (1.26)$$

Anyway, this strategy introduces some important features with respect to the DSDU case: first, the synchronization is not required to obtain the current average value. Furthermore, the time distance from the moment in which the average current value is obtained to that in which the control voltage is updated is minimized. A MSDU strategy example is depicted in Figure 1.15. Notice that the dashed arrows link the sampling instant corresponding to the last sample on which the moving average is computed with the instant in which the control voltage is updated with the corresponding value.

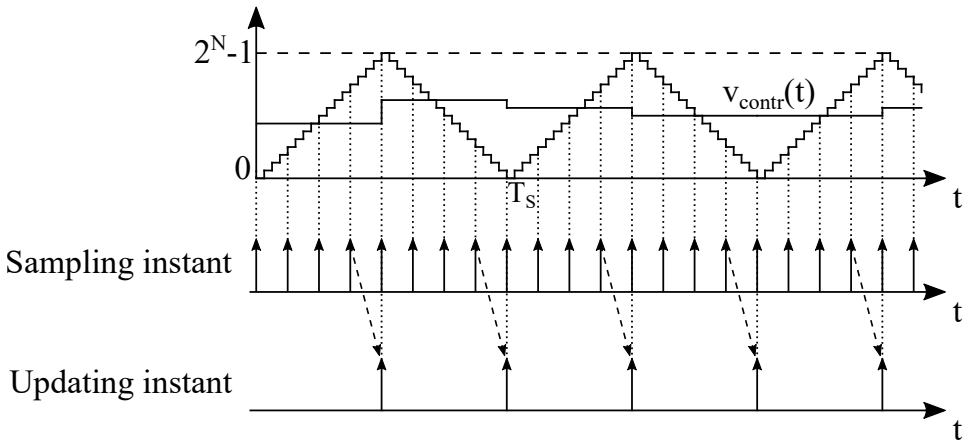


Figure 1.15: Multisampling double-update mode example.

1.3 Average Current Controller - PI Regulator and Control System

When a controller is designed, its parameters choices need to be referred to a specific power converter topology and to the system specifications. For this reason, considering the system reported in Figure 1.1, the following characteristics have been considered:

- input voltage source: V_{in} = 600 V;
- inductance: L = 4 mH;
- inductor ESR: R = 500 mΩ;
- output voltage source: E = 300 V;
- switching frequency: f_S = 10 kHz.
- control system bandwidth: $f_{BW} = \frac{f_{SW}}{20}$ = 500 Hz

Furthermore, when implementing a closed-loop system, one of its main features to be guaranteed is stability, usually designing a phase margin in the range [45 - 60]°, at least. On the other hand, it should have a large bandwidth in order to quickly detect variations in the reference signal. This is selected by designing the transfer function of the current controller. Finally, to provide a null steady-state error, an integral action must be provided, leading to the fact that the open-loop gain, considering the cascade of the current controller, the PWM modulator, the power converter and the load, needs to present a pole in the origin [2][3][4].

The block scheme of the overall closed-loop system is depicted in Figure 1.16 considering the moving average of the load current as the controlled variable. In order to design the current controller, it is necessary to analyze the PWM modulator and the converter and load transfer functions.

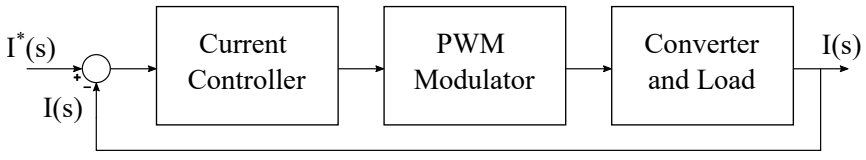


Figure 1.16: Overall closed-loop system block scheme.

As previously explained, the purpose of the PWM modulator is to provide the two IGBTs with the corresponding switching functions $q_{HS}(t)$ and $q_{LS}(t)$. If the moving averages are considered, starting from the average control voltage $V_{contr}(t)$ the output of

the PWM modulator is the converter duty cycle d(t). Its transfer function is reported in (1.27), where d(s) and V_{contr}(s) are respectively the Laplace transforms of the duty cycle and the average control voltage, and V_{TR} is the triangular carrier amplitude. As shown, it presents only a gain term which can be taken into account if a proper voltage scaling is introduced in the current controller.

$$\frac{d(s)}{V_{contr}(s)} = \frac{1}{V_{TR}} \tag{1.27}$$

Considering again the moving averages, the converter transfer function can be easily obtained in (1.28) by rearranging (1.12).

$$\frac{V^*(s)}{d(s)} = V_{in} \tag{1.28}$$

For what concerns the load, by performing a simple analysis of the Laplace-domain circuit depicted in Figure 1.6 and considering the moving averages values, (1.29) is obtained.

$$V^*(s) - E(s) = (sL + R)I(s) \tag{1.29}$$

As will be also more clear in the following discussion, the output voltage source acts as a disturbance for the average current control. For this reason, a feedfoward technique can be exploited to compensate for the dependence of the inductor current on the output voltage source; then the load transfer function, considering the moving averages, can be computed in (1.30), where a first-order low-pass filter operation with a single pole is shown.

$$\frac{I(s)}{V^*(s)} = \frac{1}{sL + R} \tag{1.30}$$

It follows that the current controller needs to introduce both a single zero and a pole in the origin, in order to obtain the wanted open-loop gain. Consequently, the current controller is composed of a PI regulator, whose transfer function is:

$$G_{PI}(s) = K_P + \frac{K_I}{s} = \frac{sK_P + K_I}{s} \tag{1.31}$$

In the following subsections the PI regulator is designed in continuous time and then discretized, addressing also the integral windup problem. Finally a discussion on transistors dead time is presented.

1.3.1 PI Regulator and Control Scheme Design

As explained in [2], in order to design the digital average current controller, it is useful to start from its design in continuous time and then proceed to its discretized implementation. In the following discussion, the control system delay, composed of

ADC delay, digital current controller computational time and PWM modulator delay, will be neglected. In Figure 1.7, the equivalent model of the system to be controlled considering the moving averages is depicted: referring to that figure, the main purpose of the control system is to provide the duty cycle d(s) which allows to obtain a specific load average current. Recalling (1.29), the block scheme represented in Figure 1.17 can be derived.

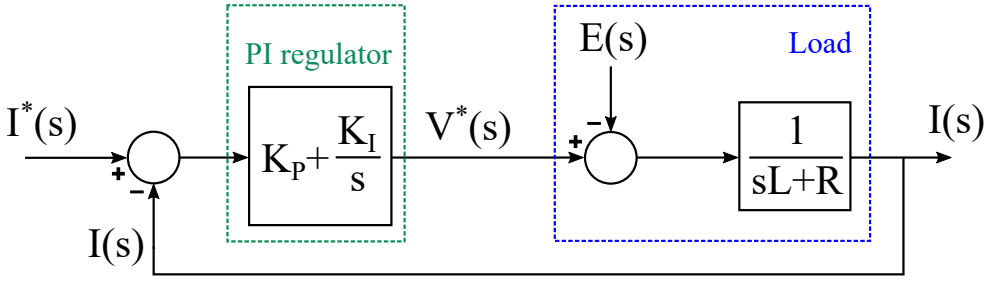


Figure 1.17: Closed-loop system block scheme.

As can be seen, in this case the load current depends on both the reference average current signal $I^*(s)$ and the output voltage source E(s). Since the latter term can be considered as a disturbance, as previously mentioned, a feedforward technique is implemented to delete its contribution, allowing the load current to depend only on the reference signal. The insertion of the feedforward and the new equivalent block scheme are reported in Figure 1.18a and Figure 1.18b respectively.

It is important now to notice that the overall system is composed by the cascade of current controller, PWM modulator, converter and load. In order to obtain the equivalent block scheme in Figure 1.18b, the action of PWM modulator and converter needs to be compensated inside the current controller. Since they are two simple gain terms, as shown in (1.27) and (1.28), a voltage scaling is accordingly included in the current controller. The overall control system block scheme is entirely represented, highlighting all the components that will be included in the current controller in the following discussion, in Figure 1.19. It is important to underline that the block scheme reported in Figure 1.19 is equivalent to that of Figure 1.18b and for this reason the latter one is exploited for the PI regulator design, considering a perturbation and linearization of the depicted quantities around the DC working point [5][6].

The open-loop gain of the system can be computed in (1.32) referring to Figure 1.19. As can be seen, a pole in the origin is present, as wanted.

$$T(s) = \frac{1}{s}\frac{sK_P + K_I}{sL + R} = \frac{1}{s}\frac{K_P}{L}\frac{s + \frac{K_I}{K_P}}{s + \frac{R}{L}} \tag{1.32}$$

In order to provide the above described features, an accurate choice of the PI regulator

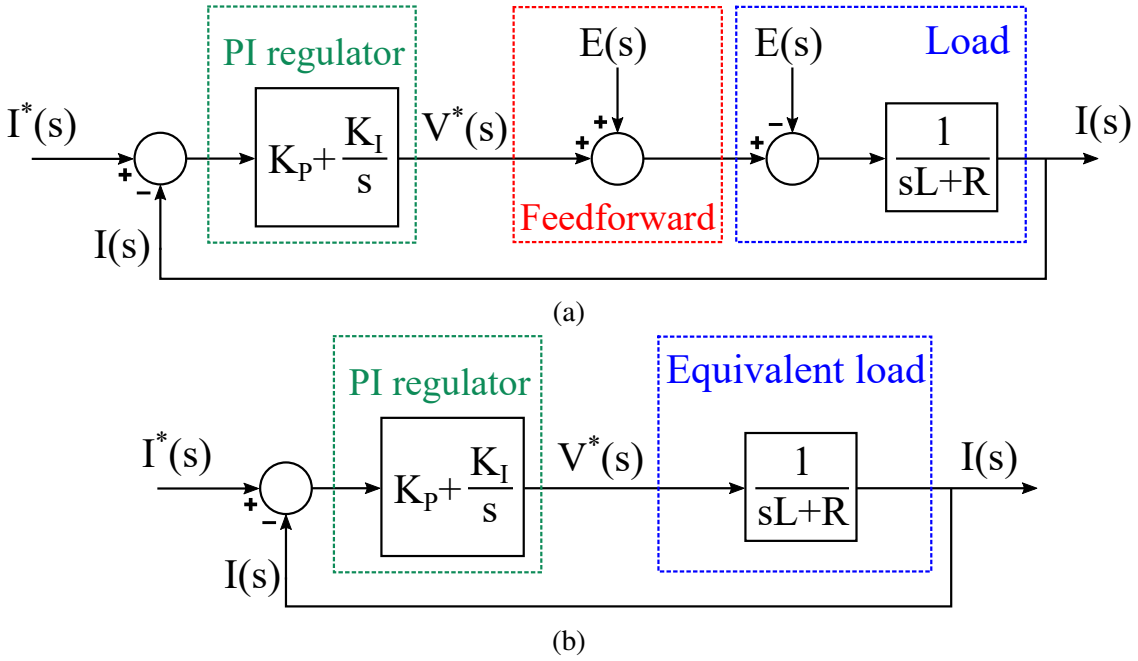


Figure 1.18: Closed-loop system block scheme with feedforward insertion (a) and new equivalent block scheme when applying this technique (b).

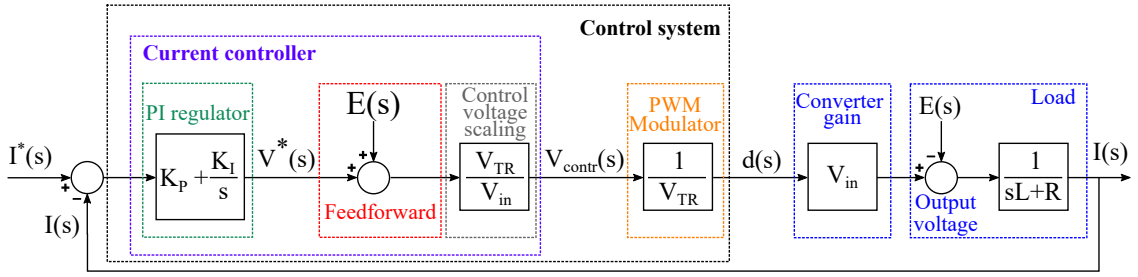


Figure 1.19: Closed-loop system block scheme highlighting the control system components.

parameters is needed. One possible solution is to apply the pole-zero cancellation method [4]; if this is the case, the open-loop gain becomes the one reported in (1.33), in which the open-loop gain phase is always $-90°$ and the bandwidth of the system is selected by the proportional term K_P.

$$T(s) = \frac{1}{s}\frac{K_P}{L} \tag{1.33}$$

The value of K_P can be found in (1.35) considering that the magnitude of the open-loop gain at the bandwidth frequency is unitary:

$$\begin{cases} |T(j\omega_{BW})| = \frac{1}{\omega_{BW}}\frac{K_P}{L} \\ |T(j\omega_{BW})| = 1 \end{cases} \tag{1.34}$$

$$K_P = \omega_{BW} L = 2\pi f_{BW} L \tag{1.35}$$

The integral term K_I can be immediately computed in (1.36) considering that in the pole-zero cancellation method the frequency position of both the pole and the zero is the same one.

$$\frac{K_I}{K_P} = \frac{R}{L} \leftrightarrow K_I = K_P \frac{R}{L} \tag{1.36}$$

The Bode diagrams of the equivalent load and the PI regulator transfer functions and of the open-loop gain are depicted in Figure 1.20 for the calculated K_P and K_I parameters given the system specifications: as expected, the phase is always equal to $-90°$ and the bandwidth of the whole system is set to 500 Hz, as highlighted in Figure 1.20b.

Another possible solution for the selection of the PI regulator parameters can be that of analyzing the open-loop gain expression and choose the K_P and K_I values in order to obtain a specific bandwidth and phase margin in the range [45 - 60]° exploiting (1.37) and (1.38) respectively, as explained in [2].

$$1 = |T(j\omega_{BW})| = \frac{1}{\omega_{BW}} \sqrt{\frac{(\omega_{BW} K_P)^2 + K_I^2}{(\omega_{BW} L)^2 + R^2}} \tag{1.37}$$

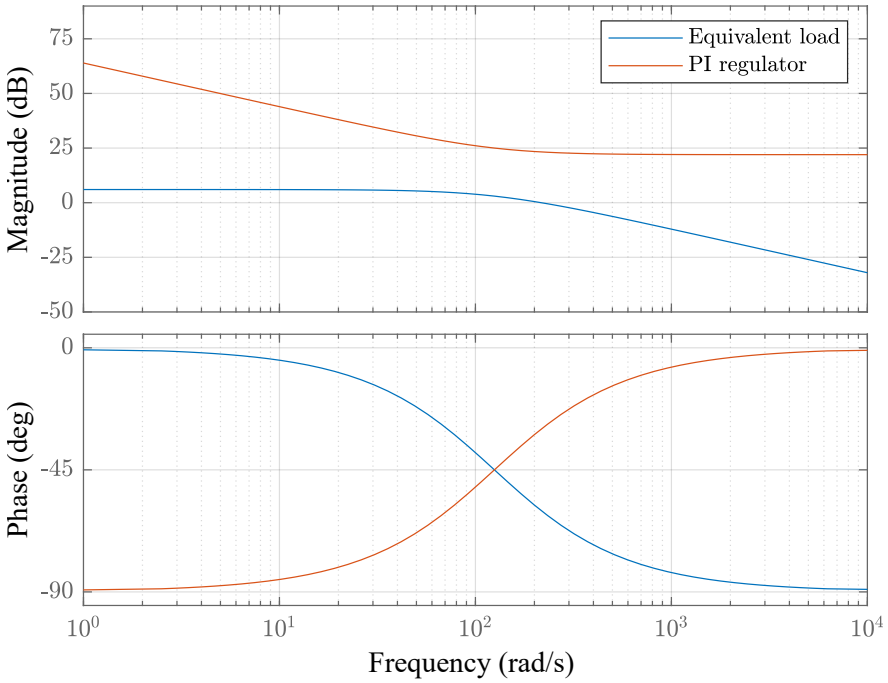
$$\phi = 180° - \angle T(j\omega_{BW}) = 180° - 90° + \arctan(\frac{\omega_{BW} K_P}{K_I}) - \arctan(\frac{\omega_{BW} L}{R}) \tag{1.38}$$

Both the two presented methodologies are valid as long as all the converter elements values are known. However, referring to the particular system under control, the inductor ESR is a parasitic component, which cannot be easily measured. This can be the case for many applications and, for this reason, the K_P and K_I values cannot be computed basing on these derivations. Furthermore, the damping ratio of the closed-loop system is not taken into account in these two strategies, even though it is a fundamental aspect in current controllers in order to have a better control on possible current overshoots after a change in the reference average current value [4][5].

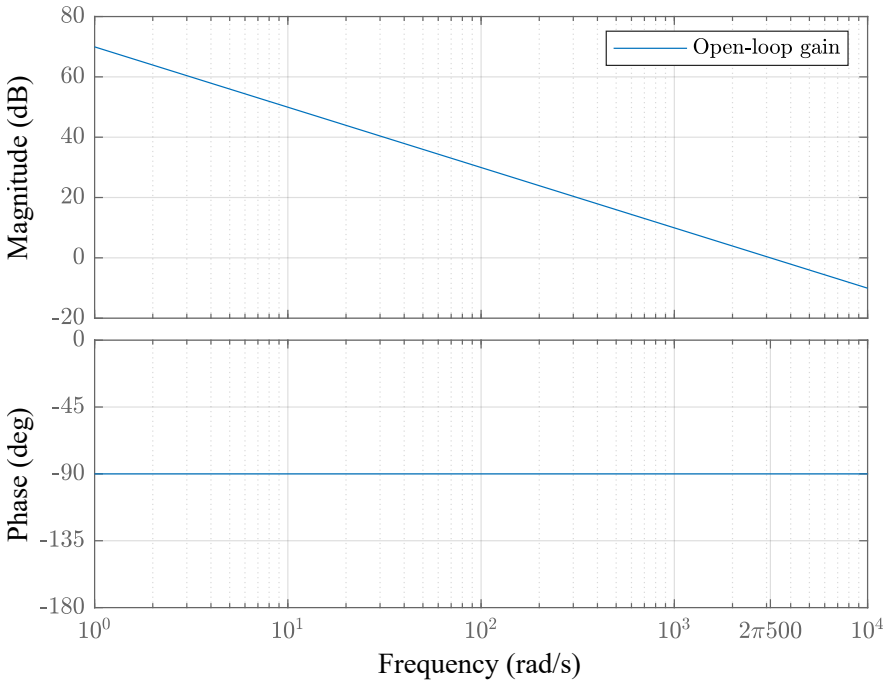
For these reasons, an approach based only on inductance and bandwidth values, considering also the damping ratio, will be exploited for the following current controller design [4]. In order to set a specific controller bandwidth, referring to the expression reported in (1.32), two assumptions need to be made: first, at the bandwidth frequency, $\omega_{BW} L \gg R$, which means that the inductor ESR can be neglected for the parameters choice. Secondly, the frequency of the zero is set as reported in (1.39).

$$\frac{K_I}{K_P} = \frac{\omega_{BW}}{5} \tag{1.39}$$

Under these hypotheses, the expression reported in (1.33) is correct and the proportional gain can be design exploiting (1.35). Moreover, the integral gain can be computed



(a)



(b)

Figure 1.20: Equivalent load and PI regulator transfer functions (a) and open-loop gain (b) Bode diagrams with pole-zero cancellation method.

by rearranging (1.39). The final values of the proportional and integral terms which are used for the design are then reported for clarity in (1.40):

$$\begin{cases} K_{\mathrm{P}} = \omega_{BW} L \\ K_{\mathrm{I}} = K_P \frac{\omega_{BW}}{5} \end{cases} \tag{1.40}$$

The equivalent load and the PI regulator transfer functions and the open-loop gain Bode diagrams are depicted in Figure 1.21 for the corresponding $\mathrm{K_P}$ and $\mathrm{K_I}$ values: as can be seen, the bandwidth of the whole system is set to 500 Hz with good approximation, as highlighted in Figure 1.20b. Furthermore, the phase margin is sufficiently higher than 60°.

Referring to the closed-loop gain, reported in (1.41), the second term structure is identical to that of a second-order closed-loop system [5]: the corresponding damping ratio is derived in (1.42)

$$G_{CL}(s) = \frac{\frac{1}{s}\frac{sK_P+K_I}{sL+R}}{1+\frac{1}{s}\frac{sK_P+K_I}{sL+R}} = \frac{sK_P}{s^2L+s(R+K_P)+K_I} + \frac{K_I}{s^2L+s(R+K_P)+K_I} \tag{1.41}$$

$$\zeta = \frac{R+K_P}{2\sqrt{K_I L}} \tag{1.42}$$

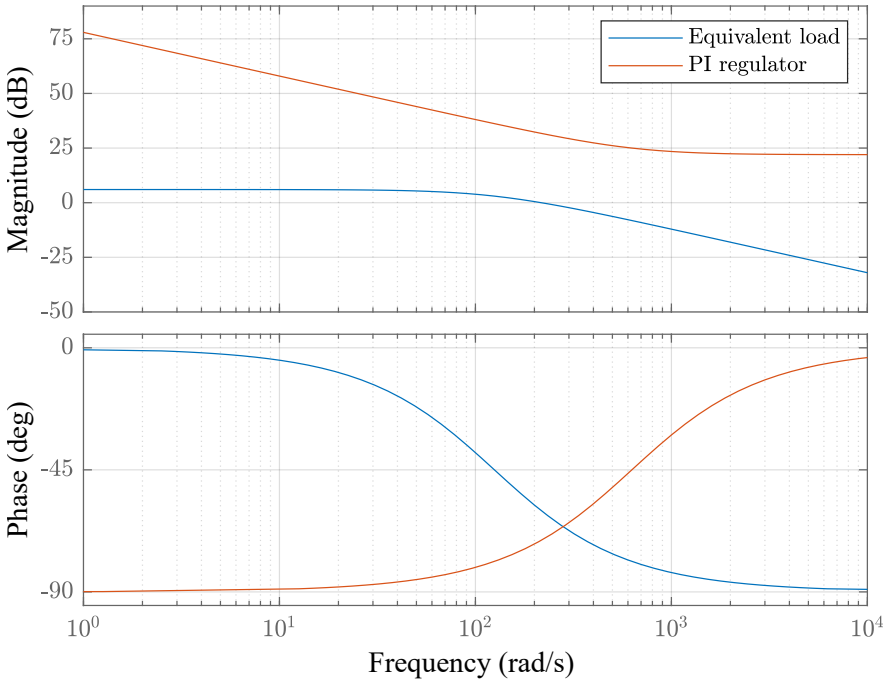
By substituting the proportional and integral gain expressions derived in (1.40) in (1.42) and neglecting the inductor ESR, the damping ratio can be computed in (1.43), which shows that these parameters choice corresponds to an overdamping situation, independently of the actual R value.

$$\zeta = \frac{K_P}{2\sqrt{K_I L}} = \frac{\omega_{BW} L}{2\sqrt{\omega_{BW} L \frac{\omega_{BW}}{5} L}} = \frac{\sqrt{5}}{2} > 1 \tag{1.43}$$

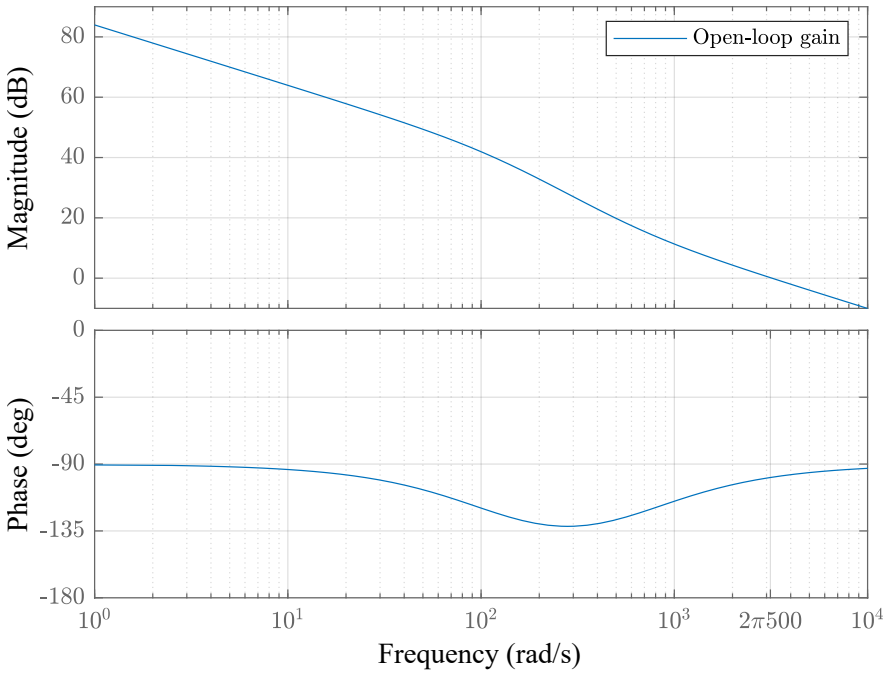
However, due to the presence of the first term in (1.41), a small overshoot will anyway be present after a step change in the reference average current signal.

1.3.2 Average Current Controller Discretization

The average current controller which has been presented in the previous section was designed in continuous time. In order to obtain its digital implementation, a proper discretization technique needs to be adopted for the PI regulator. Before proceeding, it is important to highlight that in the digital case, the controller introduces a delay which has been neglected in the continuous time design. Its main effect is a reduction of the phase of the open-loop gain. Anyway, as could be seen in Figure 1.21b, the phase margin obtained with the particular PI regulator parameters choice is well above 60°. For this reason, this drawback is not taken into consideration and the same design parameters are kept.



(a)



(b)

Figure 1.21: Equivalent load and PI regulator transfer functions (a) and open-loop gain (b) Bode diagrams with damping ratio based method.

As reported in [2], a simple way to discretize the PI regulator is to replace the continuous time integration with its numerical approximation. This can be done by means of an Euler or a trapezoidal integration method. Considering the Backward Euler integration method, the substitution to be performed to obtain the discrete model is:

$$s = \frac{z-1}{zT_s} \tag{1.44}$$

By substituting (1.44) in (1.31), it is possible to derive the PI regulator transfer function in the Z-domain as shown in (1.45).

$$G_{PI}(z) = K_P + K_I T_s \frac{z}{z-1} \tag{1.45}$$

The corresponding block scheme is reported in Figure 1.22, where the insertion of the feedforward voltage and the output scaling are highlighted. With respect to the continuous time scheme in Figure 1.19, the triangular carrier amplitude becomes $2^N - 1$ instead of V_{TR}, where N is the number of bits on which both the control voltage and the triangular carrier are represented and that is selected, along with the clock frequency, to obtain the wanted transistors switching frequency. Notice that the control voltage produced at the output is kept constant until the following updating instant, namely after a half switching period, so that the behaviour of $v_{contr}(t)$ illustrated in Figure 1.10 can be obtained. Finally, it is important to highlight that the term T_s refers to the period of the current controller operation which is, as mentioned, half of the switching period.

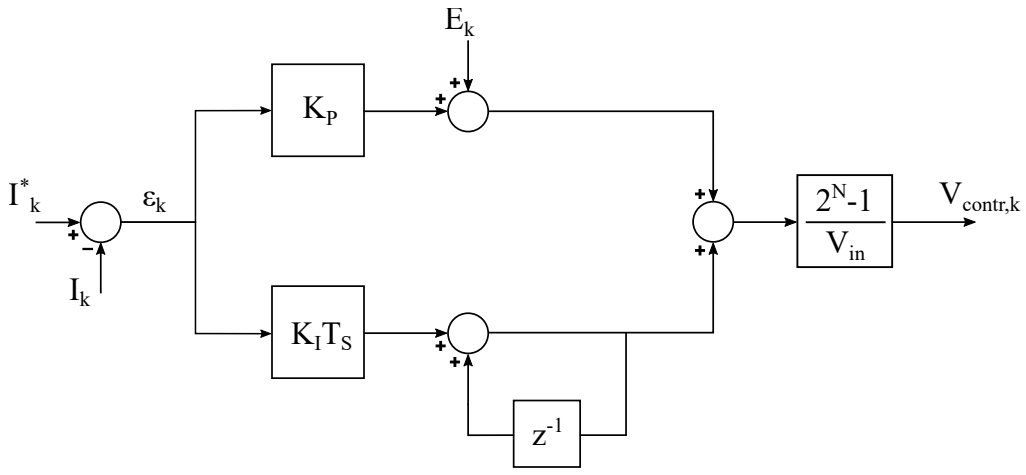


Figure 1.22: Discrete average current controller implementation.

1.3.3 Integral Anti-Windup Implementation

The integral part operation of the PI regulator basically consists in integrating over time the error between the load average current and the average current reference signal.

As explained in [2], in occurrence of large variations of the latter one, the difference between these two quantities can be not null for a large amount of time: this means that for the entire time interval duration until the load average current reaches the reference value, the integral part of the regulator accumulates the integral of the error. When this point is reached, the integral part state can be quite large, leading to, typically, a large overshoot, independently of the phase margin of the system.

This problem can be solved by limiting the integral part dynamics. In general, the output of the proposed PI regulator needs to be always limited within the power supply range, hence between 0 and V_{in}. For this reason, first the proportional part of the regulator is limited in order to never exceed these bounds. Denoting with ϵ the difference between the load average current and its reference value, and considering also the presence of the feedforward voltage, this means that:

$$0 \leq \epsilon K_P + E \leq V_{in} \tag{1.46}$$

After that, the integral part is saturated, depending on the proportional part value. In order not to make the integral state to largely increase, the following strategy is adopted:

1. The difference with positive sign between the proportional part ($\epsilon K_P + E$) and each one of the two bounds (0 and V_{in}) is computed;

2. The integral part is limited by imposing that its absolute value needs to be at most equal to the smallest computed difference.

In Figure 1.23, the core of the current controller block scheme of Figure 1.22 is depicted highlighting the presence of the two saturation blocks. For what concerns the integral part limits, two different scenarios are possible:

1.
$$V_{in} - (\epsilon K_P + E) \leq \epsilon K_P + E \rightarrow \begin{cases} Int\_max = V_{in} - (\epsilon K_P + E) \\ Int\_min = -V_{in} + (\epsilon K_P + E) \end{cases} \tag{1.47}$$

2.
$$V_{in} - (\epsilon K_P + E) > \epsilon K_P + E \rightarrow \begin{cases} Int\_max = \epsilon K_P + E \\ Int\_min = -(\epsilon K_P + E) \end{cases} \tag{1.48}$$

Notice that, in case of a large variation in the reference signal, the proportional part can be saturated to one of the bounds. This means that the integral part is imposed to be null: consequently, starting from the following cycle, the integral state is reset. Basically, the integration resumes only when the load current is close to the reference current value. In case of a variation of the reference signal which does not imply a saturation of the proportional part, the integration operation is not stopped, but is subject to a strong limitation which anyway avoids the windup phenomenon.

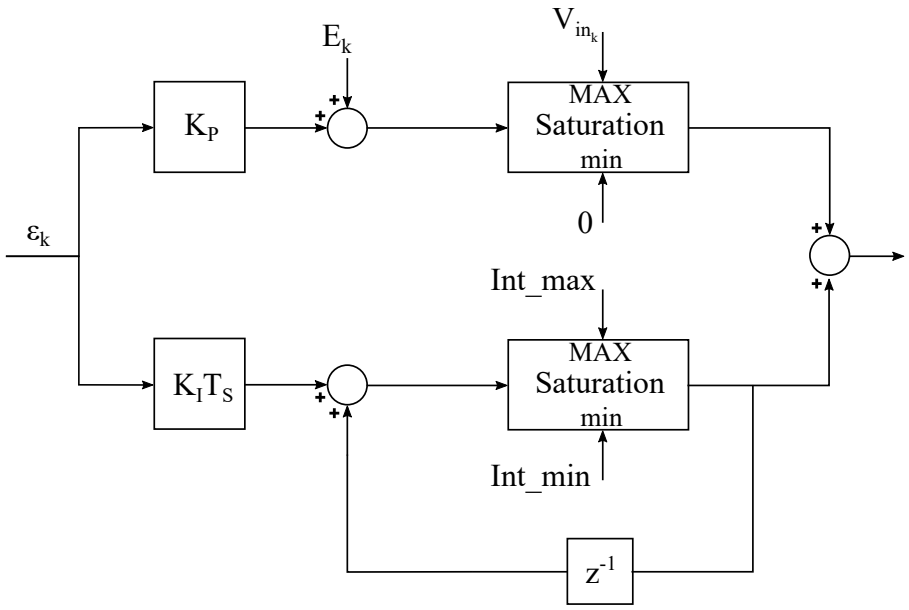


Figure 1.23: Detail of the current controller with anti-windup implementation.

1.3.4 Dead Time Discussion and Implementation

A final important issue which is discussed in [2] for what concerns the control system is the need of dead times in the transistors operation. In every power converter, short-circuits of the input voltage sources must be avoided since they lead to a huge increase in the current circulating in the circuit causing great damages in the devices. Referring to Figure 1.1, this means that the two IGBTs must not be active simultaneously.

In an ideal case, the switching action of the transistors is instantaneous: it is sufficient then to drive their gates with complementary signals to avoid this situation. In practical cases, the transistors require a certain commutation time to change their state, leading to the need of a more sophisticated driving strategy to avoid short-circuits. A simple solution to solve this drawback is the insertion of dead times in the transistors switching operation, that basically consists in a time interval in which the switches are both turned-off and the current circulates in one of the free-wheeling diodes, depending on the inductor current direction.

This technique can be implemented by providing two different control voltages, one for each transistor, equally spaced from the nominal one. In order to do so, referring to a single IGBT operation, it is important to notice that by shifting the control voltage, the instants when the transistor starts its commutation are accordingly shifted. For instance, if the control voltage value, considering that it is included in the range [0 - 2^N-1] and supposing that it is far from these bounds, is increased by 1, independently of the considered transistor the condition which implies the switch commutation is met

one clock period after the nominal one during the positive slope of the triangular carrier, whilst it is anticipated by one clock period during the negative slope phase. The opposite behaviour is obtained when decreasing the control voltage value. As a consequence, denoting the wanted dead time as t_d, if the control voltages of the high-side and low-side IGBTs are respectively decreased and increased by $\frac{t_d}{2} f_{clk}$, a time interval of duration t_d in which both the transistors are turned-off is obtained for both the slopes of the carrier. An example which shows the discussion above is reported in Figure 1.24.

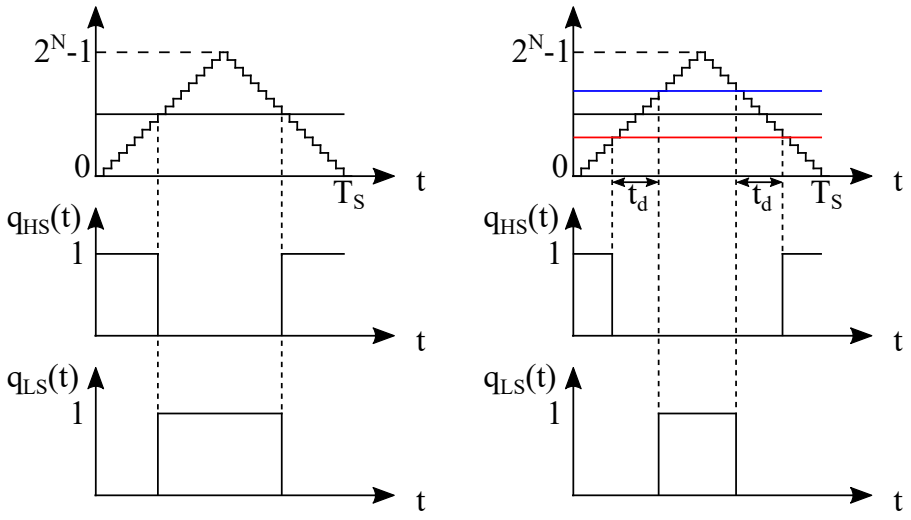


Figure 1.24: Digital PWM modulator with (on the right) and without (on the left) dead time insertion.

The main drawback of the insertion of a dead time in the switching operation is an error in the voltage which is applied to the inductor [2]: basically, each IGBT is active for a time interval of duration equal to the dead time less than the desired one in each switching period. Considering for instance the case in which the load current is positive, during the time intervals in which both the transistors are turned-off, the current flows in the low-side IGBT free-wheeling diode, which ideally imposes a null voltage on the inductor: therefore, the average voltage which is applied to the inductor is reduced, as a function of the dead time. The same thing happens when the current is negative, considering that during the dead time intervals the current always flows in the high-side IGBT free-wheeling diode, applying the input voltage source to the load, increasing the average voltage applied to it. The error in this voltage is then derived in (1.49)

$$\Delta V = -V_{in} \frac{t_d}{T_s} \mathrm{sgn}(I_L) \tag{1.49}$$

Anyway, as explained in [2], its effect can be seen as a square wave disturbance, whose frequency is that of the load current. If the reference current, and then the load current,

is a DC signal, then the integral action of the PI regulator is perfectly capable of compensating the voltage error induced by the dead time. In case of an AC reference signal, anyway the PI regulator action compensates the dead time disturbance effect.

Finally, it is important to highlight the fact that, implementing this technique for the dead time generation, the gate pulses are always symmetrical with respect to the peak and the minimum of the triangular carrier. This allows, in general, an improvement in the output spectrum of the current.

In the following chapter, some other pratical implementation considerations regarding the dead time generation will be addressed.

Chapter 2

Current Controller Implementation on Simulink

When deploying a digital controller, one of the usual main drawbacks is the time which is needed to produce the code that is used to program the device on which the controller is implemented. This problem can be solved by exploiting an automated code generation tool, reducing also the programming effort. The main result of this approach is that of minimizing the time from simulation to experimental validation. For this reason, in this chapter the current control system which was presented in the previous chapter, deploying the MSDU strategy, is implemented on Simulink [9]: in the following sections each current control system component implementation is presented. Notice that, even though not reported, all the components have also been simulated to ensure their correct operation.

After that, exploiting the Simulink HDL Coder [10], synthesizable VHDL code is generated for a Xilinx FPGA platform. In particular, as will be shown in the following chapter, every control system component is implemented and simulated on Vivado exploiting the generated VHDL code. Then, the overall current control system is implemented and simulated on Vivado exploiting the discrete time equivalent model of the converter and the load. Finally, the FPGA is programmed and the system is experimentally tested and evaluated. All the design process is summarized in the flowchart depicted in Figure 2.1.

The current controller, the averager, namely the block which performs the moving average computation, and the PWM modulator, composed of triangular carrier generator and comparator, will be implemented on Simulink by exploiting triggered subsystems. It is important to highlight an operating aspect of these blocks: when the trigger signal is detected, the outputs of the subsystem are immediately updated, since a register is present for each output. On the other hand, the inputs are not registered. This consideration is fundamental for the implementation of the current controller and the averager, as it will

be explained in their dedicated sections.

Finally, a MATLAB initialization script has been used for the parameters definition. Anyway, in the following discussions the values which have been designed are specified, in order to better clarify the explanation of each component of the control system.

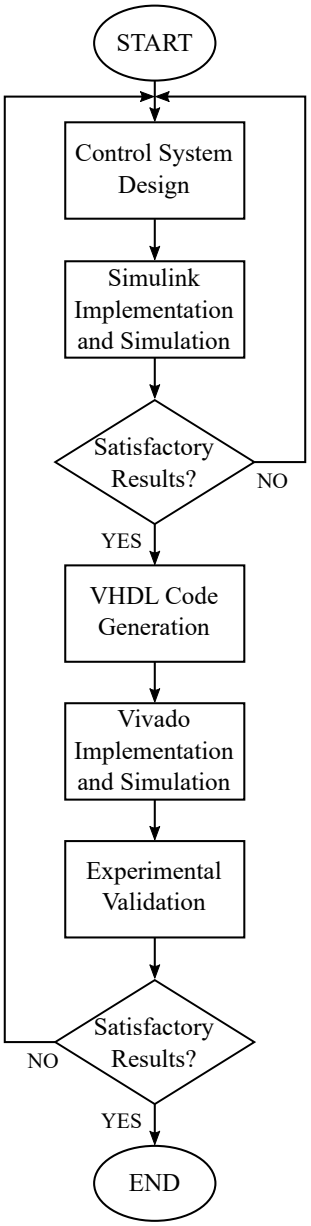


Figure 2.1: Design process flowchart.

2.1 PWM Modulator Block Scheme

The PWM modulator main purpose is to provide, given a specific input control voltage, the switching functions for the two IGBTs of the converter, besides the command signals which are needed for the operation of the current control system, as depicted in Figure 2.2, where the presence of comparator and triangular generator is underlined. In particular, according to the multisampling double-update strategy which was explained in the previous chapter, the following signals are generated:

- a sampling trigger signal in order to activate the ADC sampling operation;
- a control voltage updating signal to enable the control voltage changes in the instants which correspond to the maximum and minimum values of the triangular carrier;
- a controller execution signal to execute the current controller sufficiently before the control voltage updating instant.

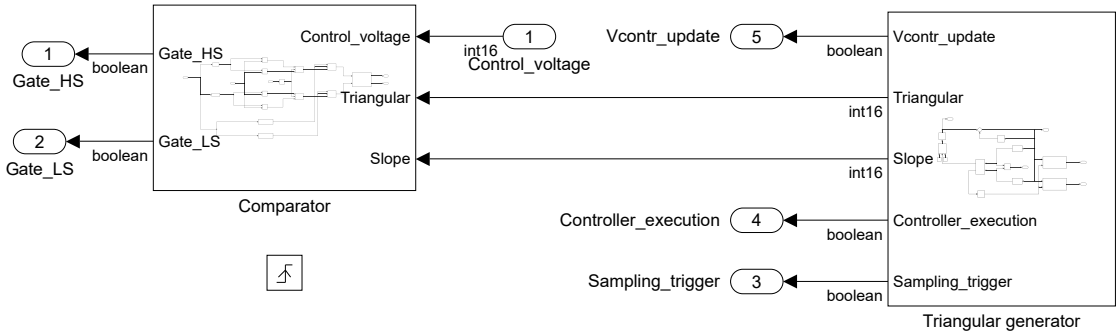


Figure 2.2: PWM modulator subsystem organization.

As explained in the previous chapter, the number of bits N of the triangular carrier and the clock frequency f_{clk} are crucial designs which select the switching frequency of the converter. For reasons that will be further explained in the following chapter, the system clock frequency is set to 80 MHz; in order to obtain a switching frequency which is around 10 kHz, a 12-bit resolution is selected for the triangular carrier and the control voltage. In fact, this allows to synthesize a switching frequency f_S approximately equal to 10 kHz, as computed in (2.1).

$$f_S = \frac{f_{clk}}{2 \cdot (2^N - 1)} \simeq 9.768\,\text{kHz} \tag{2.1}$$

Finally, it is important to highlight the arithmetic precision which has been used: even though both triangular carrier and control voltage can be unsigned integer numbers on 12 bits, the triangular carrier generator and the comparator mainly work on 16-bit integer numbers. This is done in order to operate with a standard type for what concerns the

PWM modulator and, besides, due to the fact that, as will be showed in the following discussions, the comparator needs to elaborate the control voltage for the dead time generation and can produce also negative numbers.

2.1.1 Triangular Generator Block Scheme

In order to better understand the triangular carrier generator operation, first the block scheme corresponding to the double-sampling double-update operation is reported. After that, the generation of the commands for the correct MSDU strategy is shown. In Figure 2.3, the block scheme corresponding to the DSDU strategy is shown. Since the triangular generator block is included inside the PWM modulator triggered block, its operation is in turn triggered by the clock frequency signal.

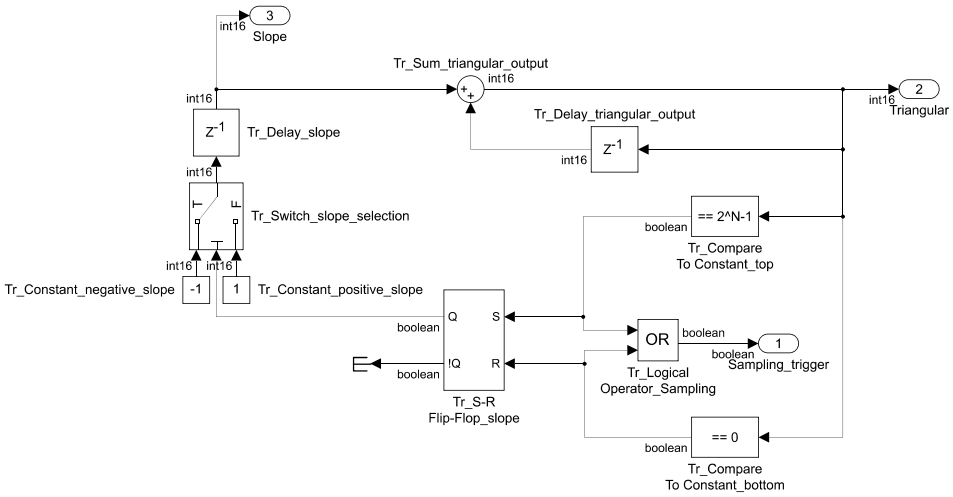


Figure 2.3: Triangular generator block scheme for the DSDU strategy.

As can be seen, a level-triggered SR Flip-Flop is exploited to switch the triangular carrier slope value for the following time step: when the peak of the triangular carrier is reached, hence 2^N-1, the slope becomes -1; on the other hand, when the carrier is equal to 0, the slope is set to 1. Notice that the output of the SR Flip-Flop is asynchronously updated depending on its truth table which is reported for clarity in Table 2.1.

S	R	Q	!Q
0	0	MEMORY	MEMORY
0	1	0	1
1	0	1	0
1	1	UNDEFINED	UNDEFINED

Table 2.1: SR Flip-Flop truth table.

An accumulator, reported in the top part, produces the correct triangular carrier value, based on its previous operation step value and on the correct slope. Finally, the value of the triangular carrier is checked in order to produce the sampling trigger signal in the instants corresponding to the maximum and the minimum of the carrier and change the SR Flip-Flop output correspondingly.

In the MSDU case, the triangular generator subsystem is modified as depicted in Figure 2.4. As can be seen, the maximum and minimum values of the triangular carrier are now exploited to generate the control voltage updating signal, whereas the block scheme reported in Figure 2.5 is deployed to generate the sampling trigger signal depending on the adopted multisampling strategy.

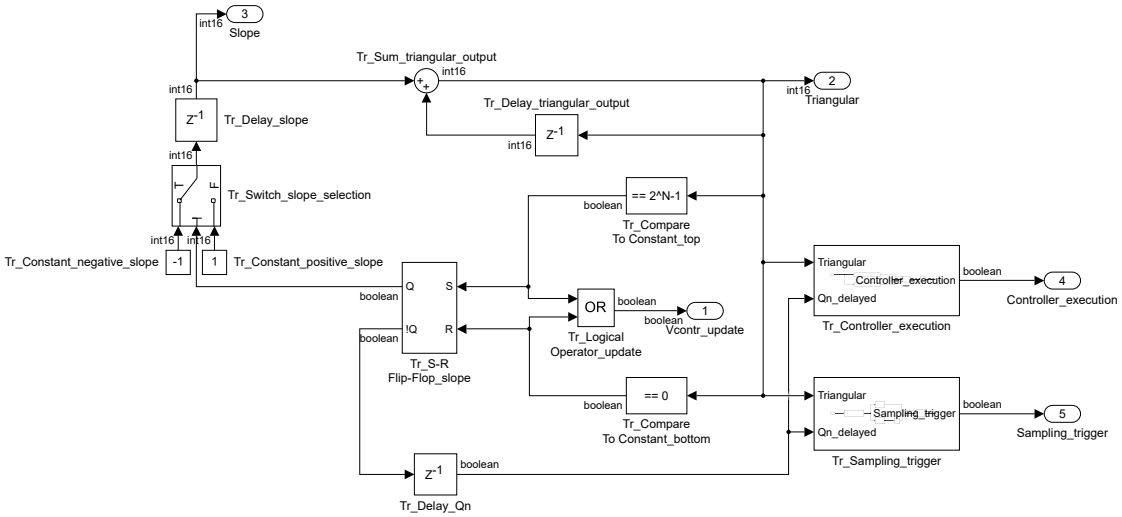


Figure 2.4: Triangular generator block scheme for the MSDU strategy.

As will be explained in the following chapter, the multisampling factor N_{MS}, considering a whole switching period, is set to 256; the period of the sampling trigger signal can be computed in (2.2), considering a single edge of the triangular carrier and the multisampling factor, as a function of the clock period T_{clk}.

$$T_{Sampling\_trigger} = T_{clk}\frac{2^N}{\frac{N_{MS}}{2}} = T_{clk}\frac{1}{f_{clk}}\frac{4096}{128} = 32T_{clk} \tag{2.2}$$

This means that the sampling trigger signal needs to be generated every 32 clock cycles. This can be done by exploiting the triangular carrier generator, since it is intrinsically a counter, considering its last five bits: during the positive and negative edges of the carrier, the sampling trigger is respectively produced every time all the five bits are equal to 1 and to 0. The distinction between the two carrier edges is obtained by looking at the previous step negated output of the SR Flip-Flop. This operation is described in Figure

2.5, where Tr_bits is defined as $\log_2 \frac{N_{MS}}{2}$.

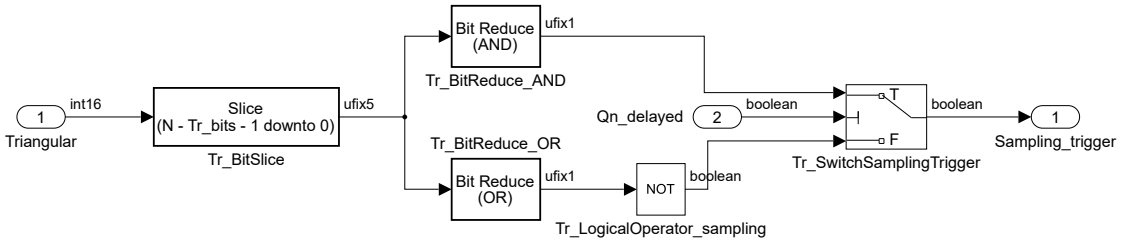


Figure 2.5: Triangular generator sampling trigger signal generator block scheme.

The subsystem which generates the controller execution signal is represented in Figure 2.6. The current controller execution needs to starts at an instant which is sufficiently before that of the control voltage update, as a function of its computational time. Besides, it needs to operate on the most recent available data, hence on the most recent computed moving average.

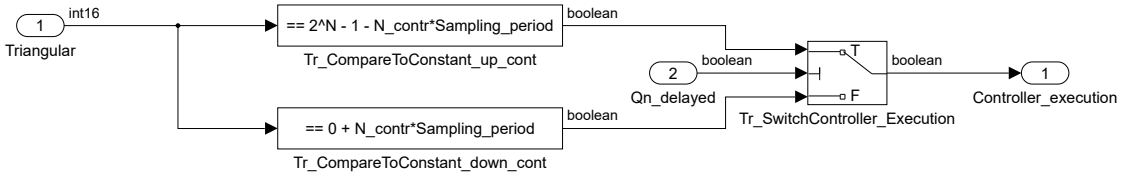


Figure 2.6: Triangular generator controller execution signal generator block scheme.

As will be more clear in the following discussions, in correspondence of the sampling trigger signal, the averager output is updated with the corresponding value, which is computed on the samples obtained up to the previous sampling trigger event. For this reason, the controller execution signal is generated one clock period after the last sampling instant which allows to terminate the controller operation before the control voltage updating instant. In particular the controller execution signal can be programmed from the initialization script through the N_contr value: this parameter indicates the sampling instant, starting from the updating instant and going backwards, after which the controller is executed.

In the implemented case, N_contr is set to 1; therefore the controller is operated one clock cycle after the last sampling trigger signal before that of the control voltage update. Referring to Figure 2.6, this is done by comparing the triangular carrier value with a constant which depends on the carrier edge: defining Sampling_period as 32, the comparison with the constant basically selects one of the sampling instants as a function of N_contr, depending on the corresponding carrier edge. As for the sampling trigger signal generation, the correct carrier edge is selected by exploiting the negated output

of the SR Flip-Flop. The last clock cycle shift is implemented at the top-level, as will be explained when the whole control system is presented. An example to clarify the commands generation is reported in Figure 2.7 considering N_contr equal to 1.

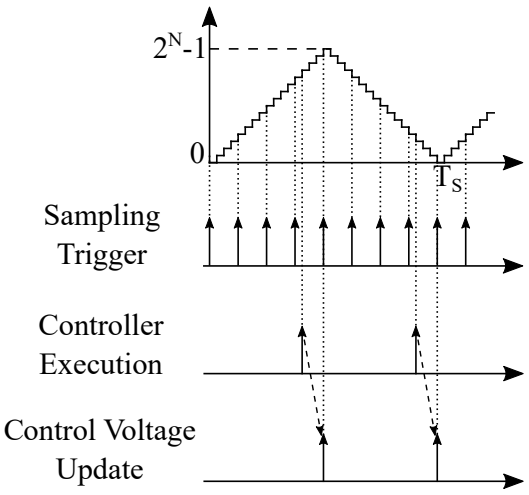


Figure 2.7: Triangular generator commands generation example.

2.1.2 Comparator Block Scheme

The block scheme of the implemented comparator is reported in Figure 2.8. As explained in the previous chapter, two different control voltages, one for each converter transistor, need to be generated in order to introduce a dead time in the IGBTs operation. This is done by modifying the control voltage produced by the current controller. In the implemented case, a 1 µs dead time has been deployed; its value in terms of clock cycles is computed in (2.3).

$$DT = 1\,\mu s \cdot 80\,\text{MHz} = 80 \tag{2.3}$$

Then, the high-side and low-side IGBT control voltages are respectively obtained by subtracting and adding DT/2 to the original control voltage, which leads to a situation as that reported for instance in Figure 1.24. After that, the generated control voltages for the two transistors are compared with the triangular carrier also considering its slope, as illustrated in Table 1.1.

An important point about this dead time introduction strategy needs to be discussed: considering the high-side IGBT, its corresponding control voltage is smaller, in terms of integer value, than that generated by the current controller. This means that, in particular, the high-side switch cannot be always active within a switching period, which is the situation of unitary converter duty cycle. The same thing happens for the low-side IGBT:

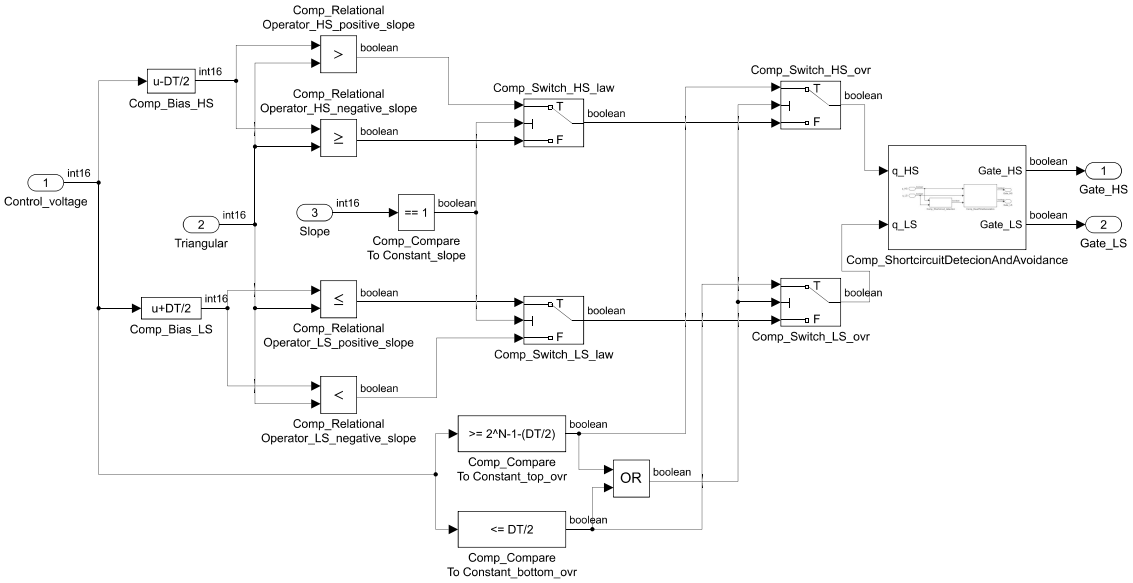


Figure 2.8: Comparator block scheme.

since its corresponding control voltage is always greater than the original one, it cannot be always active, which corresponds to a null converter duty cycle case. Depending on the current direction, during the dead time intervals one of the two freewheeling diodes turns on, making it possible to obtain a situation equivalent, in terms of applied load voltage, to that of unitary or null converter duty cycle. However, this effect depends on the current direction, as mentioned, and cannot therefore be achieved in every situation.

After a big reference signal variation, it can be useful anyway to provide these operating cases in order to quickly recover the steady-state operation. For this reason, the original control voltage value is compared with two constants: when it is bigger than or equal to (2^N-1-DT/2) or lower than or equal to DT/2, the high/side and low-side IGBTs switching functions are respectively set to 1 and 0, in the first case, and to 0 and 1 in the second one.

A crucial consideration regards the possibility of short circuits when the control voltage is updated due to the introduction of this overwriting operation. An example of this situation is reported in Figure 2.9; notice that this can happen in occurrence of both the sampling instants, depending on the overwriting operation.

A simple solution to this problem is provided by the block reported in Figure 2.10: a first subsystem, shown in Figure 2.11, compares the actual value of each switching function with the previous time step corresponding one; in case a simultaneous change in the switching functions, which would lead to a short circuit, is detected, the subsystem depicted in Figure 2.12 introduces the wanted dead time in the IGBTs switching operation

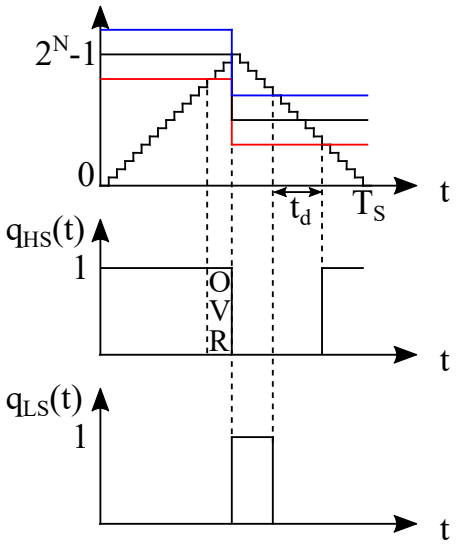


Figure 2.9: Short circuit at control voltage update instant due to overwriting operation example.

by exploiting a level-triggered SR Flip-Flop: in particular, when a short circuit is detected, both the switching functions are forced to be null. On the other hand, if no short circuit is detected, this subsystem is transparent and the switching functions are those determined as previously explained. Notice that this solution cannot replace the dead time generation achieved through the aforementioned two control voltages since it would compromise the pulses symmetry with respect to the updating instants.

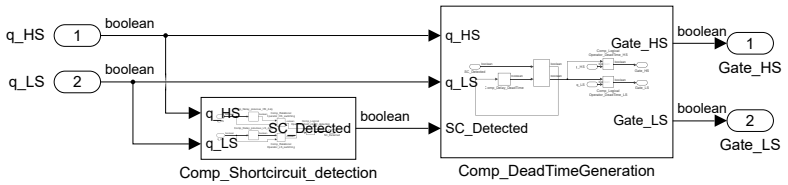


Figure 2.10: Comparator short circuit detection and avoidance subsystem block scheme.

2.2 Moving Average Block Scheme

In Figure 2.13, the block scheme of the triggered subsystem which computes the moving average is depicted. As will be explained in the following section, for the implemented current controller only the load current and the input voltage values are needed for its execution, thus only two ADCs are deployed. Referring for instance to Figure 1.19,

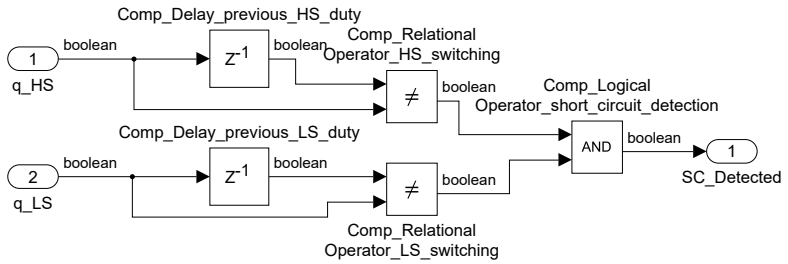


Figure 2.11: Comparator short circuit detection subsystem block scheme.

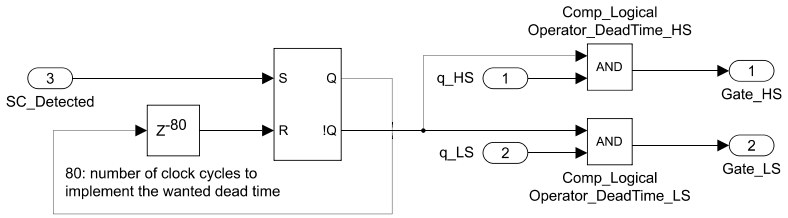


Figure 2.12: Comparator dead time generation subsystem block scheme.

also the output voltage source is exploited to implement the control system. Then, in general, the same structure for the moving average computation can be replicated if other quantities average values are needed.

The operation of the averager block can be explained as follows, considering that the ADCs are characterized by a 12-bit resolution: when the trigger signal is received, the data coming from the two ADCs is sampled. After that, a delay block, whose delay length is set to N_{MS}, hence 256 in this case, is deployed. An accumulator is implemented so that, at every time step, the new sampled data is added to the sum of the previous ones and the output of the N_{MS} length delay block is subtracted from the accumulator state, so that the sum of exactly 256 samples is obtained. After that, the average operation is accomplished by a bit right shift: by executing a right shift of (Tr_bits+1), namely 8 in this case, positions, the accumulator state value is divided by 256, and the correct average value considering a switching period is obtained.

Finally, it is important to underline a necessary change in the autogenerated VHDL code: since the averager is implemented with a triggered subsystem, its outputs are registered when the block trigger signal, which is generated when the samples are available, is received. In order to obtain the wanted behavior, it is necessary to modify the generated code in order to update the average output registers by exploiting the sampling trigger signal.

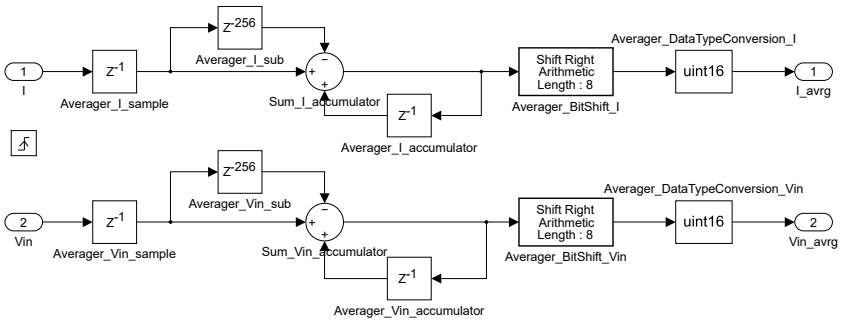


Figure 2.13: Averager block scheme.

2.3 Current Controller Block Scheme

Prior to the presentation of the current controller, it is important to highlight a characteristic of the experimentally tested converter. Due to the hardware availability, the exact two-quadrant DC/DC converter depicted in Figure 1.1 could not be tested. Anyway, an equivalent converter, namely an H-bridge converter [3][4], which is shown in Figure 2.14, has been exploited.

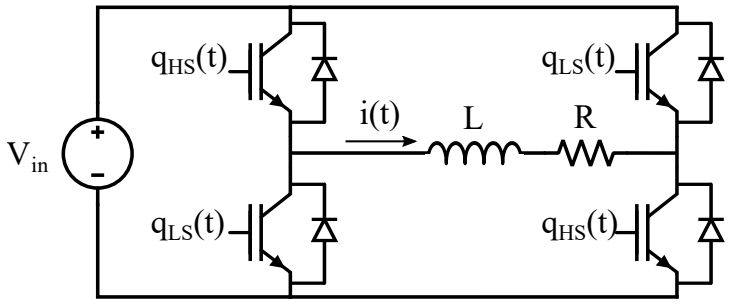


Figure 2.14: H-bridge converter topology with LR load.

As shown, with respect to the two-quadrant DC/DC converter reported in Figure 1.1, a second switching leg is inserted; in particular, its high-side IGBT switching function is the same as that of the first switching leg low-side IGBT, whereas its low-side IGBT one is the same as that of the first switching leg high-side transistor. As was done in the previous chapter, the behavior of the second switching leg can be then equivalently described as a square wave voltage, ranging from 0 to V_{in}, as a function of the first switching leg low-side IGBT switching function. Therefore, the H-bridge converter DC equivalent model can be immediately obtained and is represented in Figure 2.15.

As explained in the previous chapter, a feedforward technique has been considered in order to compensate the disturbance introduced by the output voltage source. In this

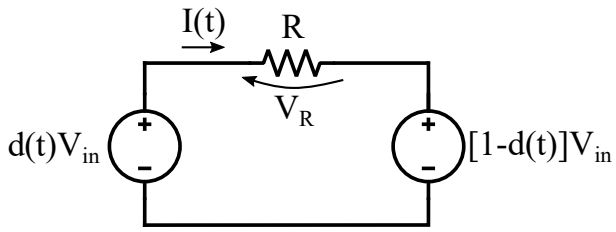


Figure 2.15: H-bridge converter and load DC equivalent model.

case, the second switching leg DC voltage behavior cannot be directly sampled. Besides, since it depends on the converter duty cycle, an algebraic loop would be present in the current controller and the control voltage could not be determined. For this reason, the feedforward voltage is obtained with a slight modification of the previously presented current controller.

By a simple mathematical rearrangement of the model in Figure 2.15, the converter DC equivalent model reported in Figure 2.16 is obtained.

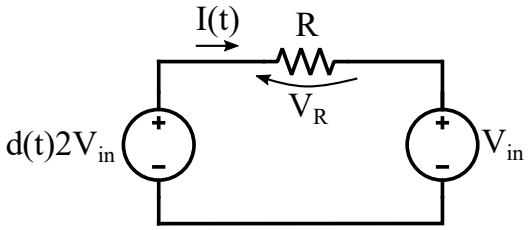


Figure 2.16: H-bridge converter and load rearranged DC equivalent model.

As can be seen, it is identical to the two-quadrant DC/DC converter DC equivalent model derived in Figure 1.5 in which the input voltage source value is set to $2V_{in}$ and the output voltage source E is equal to V_{in}. For this reason, the current controller which was designed in the previous chapter can be used to control the H-bridge converter as long as some modifications are provided: in particular, two different solutions are hereby discussed. The first solution regards the implementation of the current controller considering an equivalent input source equal to $2V_{in}$: since the ADC samples the effective input voltage source, whose value is V_{in}, inside the current controller a multiplication by 2 needs to be introduced. The feedforward voltage is instead directly obtained by the input voltage source sampled data. Also, all the saturation operations which were discussed in the previous chapter, and derived in (1.46), (1.47) and (1.48), need to be referred to the equivalent input voltage source equal to $2V_{in}$.

The second solution is obtained by applying another mathematical elaboration, hence by dividing by 2 all the quantities represented in Figure 2.16. By doing so, the equivalent input voltage source becomes equal to V_{in}, hence the real one, whereas the output voltage source is $V_{in}/2$. This means that the input voltage ADC sample can be directly used inside the current controller and that the feedforward is implemented by halving the sampled value. The difference with respect to the two-quadrant DC/DC converter lies in the proportional and integral gains. For the mathematical elaboration to be valid on all the converter, also the value of the inductance needs to be divided by 2. Since the K_P and K_I are based, as previously explained, on the inductance value, in this solution also the value of these gains needs to be halved, in order to obtain the same control system bandwidth.

The discrete current controller block scheme depicted in Figure 1.22 and 1.23, implementing the first solution, is reported in Figure 2.17. In order to illustrate its operation, it is important to comment the ADCs conditioning circuit: considering that, as aforementioned, the ADCs are characterized by a 12-bit resolution, their output values are in the range [0 - 2^N-1]. Regarding the relation between the output values and the input voltage source and the load current values, the range [0 - 2^N-1] proportionally corresponds respectively to the ranges [0 - 750]V and [-40 - 40]A. Even though the input voltage and load current values on which the current controller operates come from the averager block, they are still 12-bit values with the same correspondence. For this reason, inside the current controller subsystem, the sampled input voltage needs to be properly scaled by 750/(2^N-1), defined as Vin_GAIN, whereas the load current needs to be both scaled by 80/(2^N-1), defined as I_GAIN, and decreased by 40, namely I_BIAS. The input voltage value is then left shifted by one position to obtain the $2V_{in}$ equivalent value.

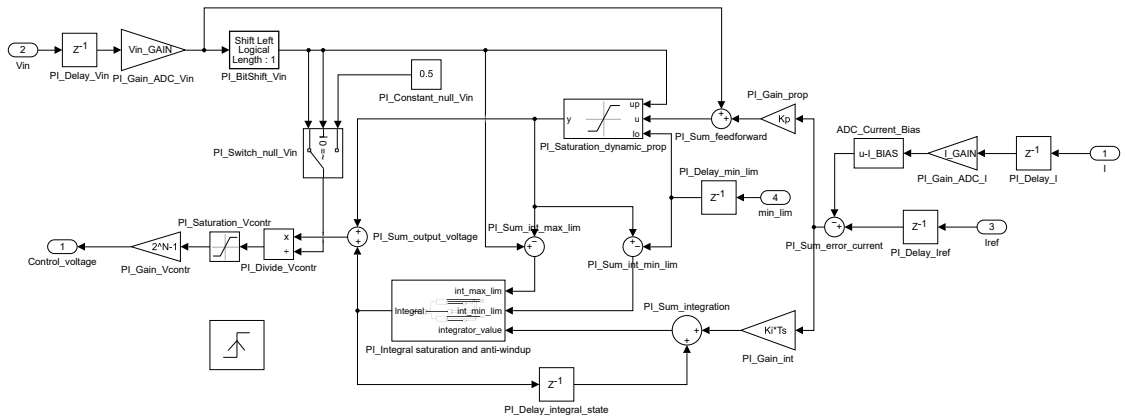


Figure 2.17: Current controller block scheme.

When the trigger signal is received, all the input quantities are sampled, in order to avoid variables changes during the controller execution. Starting from the right of the

figure, after computing the current error, the proportional part is obtained considering also the feedforward voltage and is saturated in the range [0 - $2V_{in}$]. The integral part limits are then derived, accordingly to the equivalent input source value, basing on the expressions reported in (1.47) and (1.48). The integral part is then computed and saturated as shown in Figure 2.18. The proportional and integral gains values were computed in (1.40); anyway, in the discretized current controller case, also the T_S term has to be considered. It is then derived in (2.4).

$$T_S = \frac{2^N - 1}{f_{clk}} \tag{2.4}$$

Finally, the control voltage is obtained by dividing the sum of proportional and integral parts by the equivalent input voltage value, saturating the result in the range [0 - 1], and scaling it by the triangular carrier amplitude, namely 2^N-1. Notice also the presence of a switch which avoids a division by 0 in case the input voltage source is not connected to the circuit.

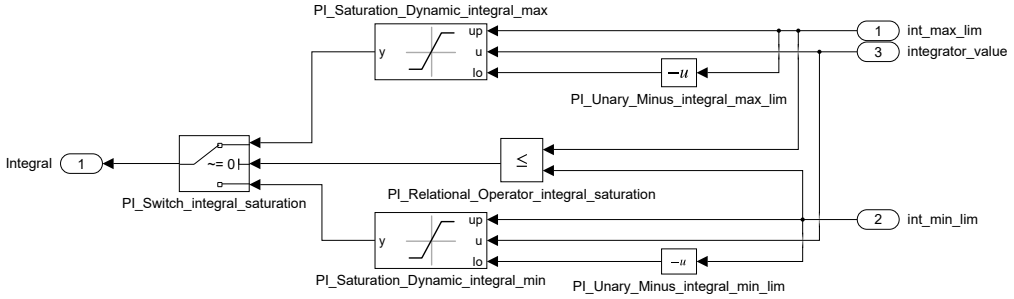


Figure 2.18: Current controller integral saturation and anti-windup subsystem block scheme.

It is important to highlight that the current controller quantities are fixed-point numbers. In order to speed up the design of every component, the Fixed-Point Tool, provided in Simulink, has been exploited. A final important notice regards the need of a modification of the VHDL code: in fact, the current controller output registers are updated when the controller execution trigger signal is received. As is done for the averager block, in order to obtain the wanted behavior, the generated code is modified since these registers need to be updated by using the control voltage update signal instead.

2.4 Average Current Control System Block Scheme

In Figure 2.19, the whole control system block scheme is reported. As can be seen, all the previously described triggered subsystem are present. Besides, a delay block is inserted to shift the controller execution signal, as aforementioned. Notice also that the averager trigger signal is delayed: this is to be sure that the averager operates on the

wanted quantities since, as will be explained in the following chapter, its trigger signal becomes high simultaneously with the change in the sampled data. For what concerns the average current reference signal, it is provided inside the Vivado environment in single precision: it is then converted in the corresponding fixed-point representation for the current controller correct execution.

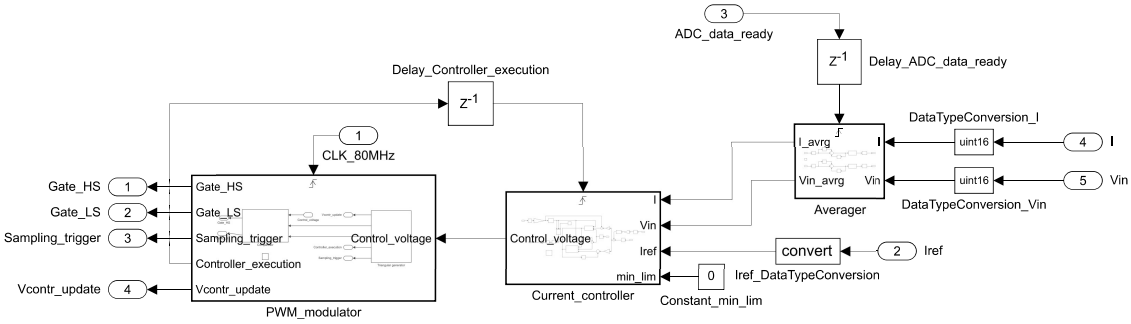


Figure 2.19: Control system block scheme.

2.5 Discrete Time Converter Equivalent Model

The discrete time equivalent model of the plant, hence the combination of converter and load, can be exploited to perform a simulation of the overall system in Vivado. In order to do so, the corresponding VHDL code needs to be generated. For this reason, the block scheme derived in Figure 1.8 is implemented in Simulink, as shown in Figure 2.20. Besides, recalling (1.49), the subsystem depicted in Figure 2.21 is inserted to model the dead time effect on the voltage which is provided to the load considering the switching behavior and the load current direction. Notice that, with respect to (1.49) and to the scheme reported in Figure 1.8, the equivalent input voltage source value has to be considered.

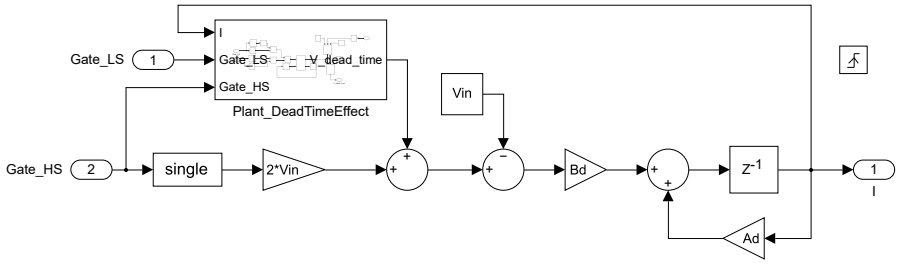


Figure 2.20: Plant block scheme.

At this point, it is useful to perform some control system simulations to verify that the discretized plant block scheme models with good approximation the converter and

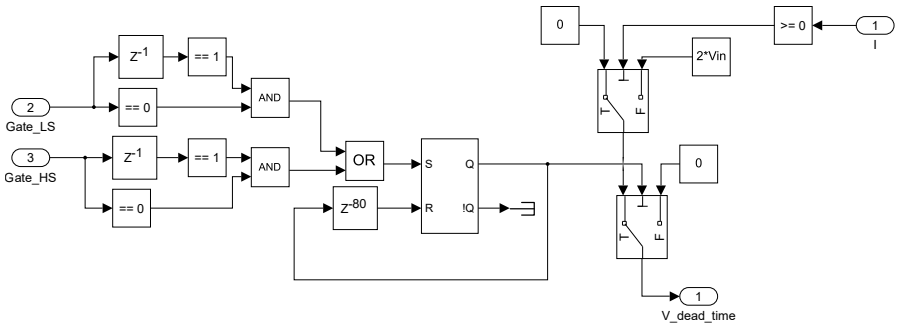


Figure 2.21: Plant dead time effect block scheme.

load real behavior and to show the expected outcome when implementing the control system on Vivado. Since it is not possible to deploy the MSDU strategy, due to the triggered subsystems output registers which lead to necessary changes in the generated VHDL code, the DSDU case has been simulated. In fact, the discretized plant model validation is independent of the actual designed control strategy; besides, as explained in the previous chapter, the behavior of these two strategies is similar.

The reference system is implemented in Figure 2.22 by exploiting PLECS Blockset for the plant realization, which is depicted in Figure 2.23. Then, the plant realized in PLECS is replaced by the discrete time plant equivalent model reported in Figure 2.20 and the delay blocks are configured in order to ensure the correct DSDU strategy operation, as depicted in Figure 2.24. As can be seen, in both the schemes the behavior of the ADCs is emulated by introducing proper gain and bias blocks. Furthermore, the average current reference signal consists of a step function. The simulation and step function parameters are:

- simulation time: 14 ms;
- initial current reference step value: 20 A;
- final current reference step value: -20 A;
- step time: 7 ms.

The current of the two plants is then measured and reported in Figure 2.25a: as can be seen, the two load currents waveforms are basically overlapped and their average value is equal, after the transient, to the reference one. Furthermore, the difference between them is computed in Figure 2.25b: even though a small difference is present, by noticing that its magnitude is always lower than 20 mA, hence relatively small with respect to the load average current values, and considering that a clock cycle delay is present in the two circuits operations, the plant discrete time equivalent model is validated for the simulation purpose that will be performed in the next chapter.

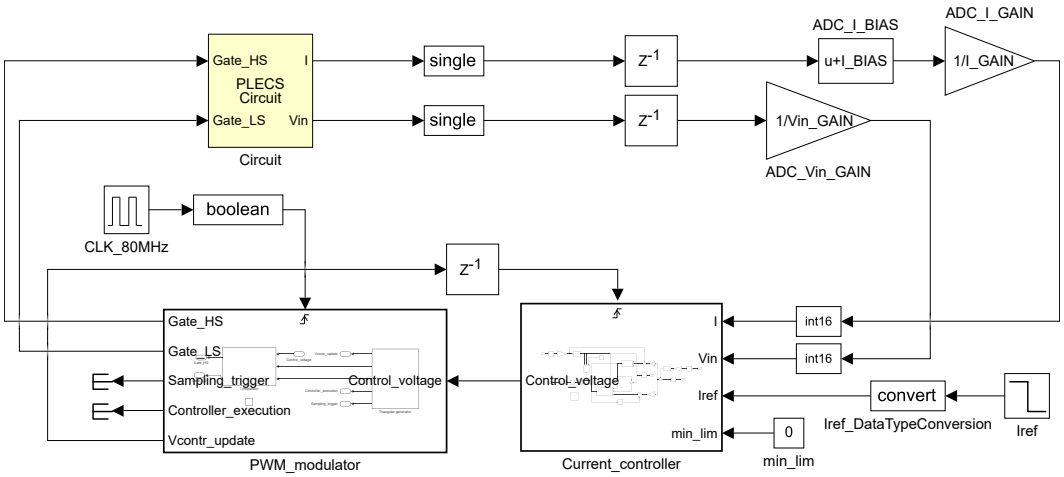


Figure 2.22: System with plant realized in PLECS block scheme.

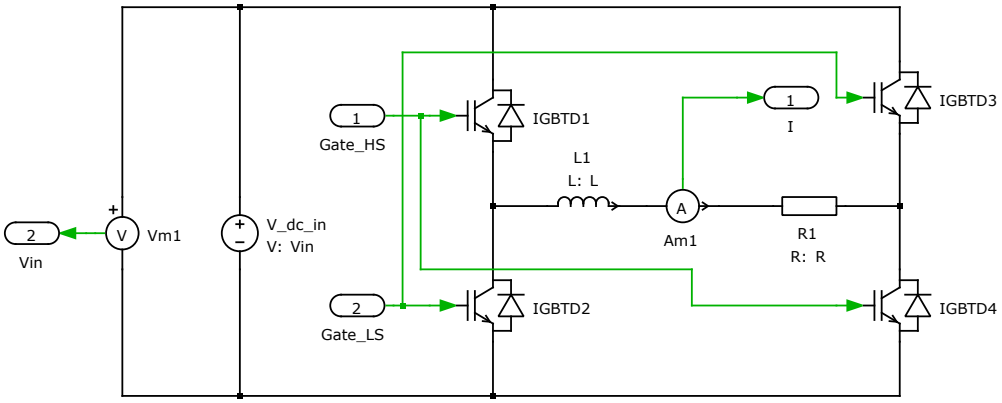


Figure 2.23: H-bridge converter topology realized in PLECS.

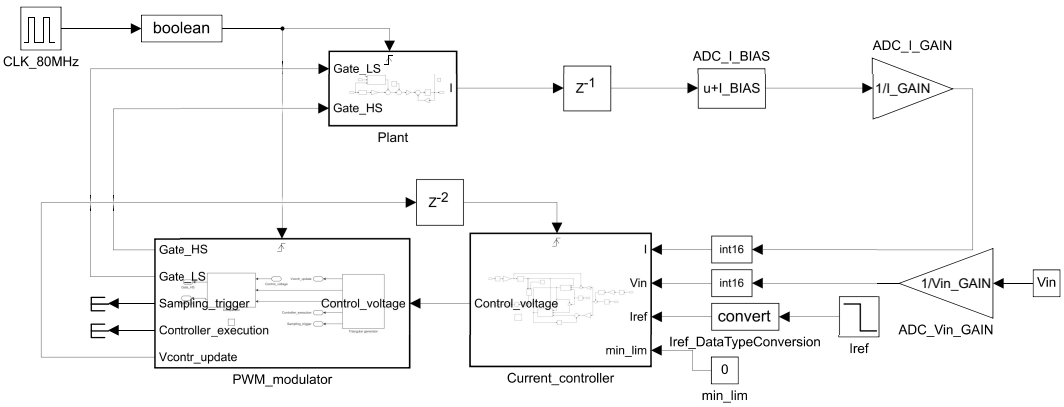
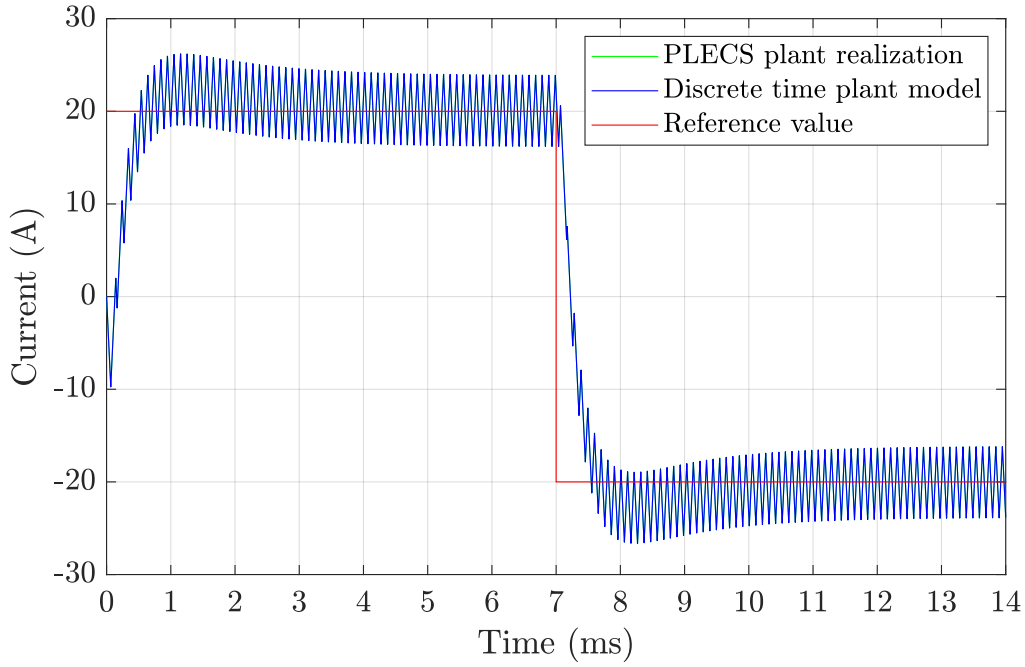
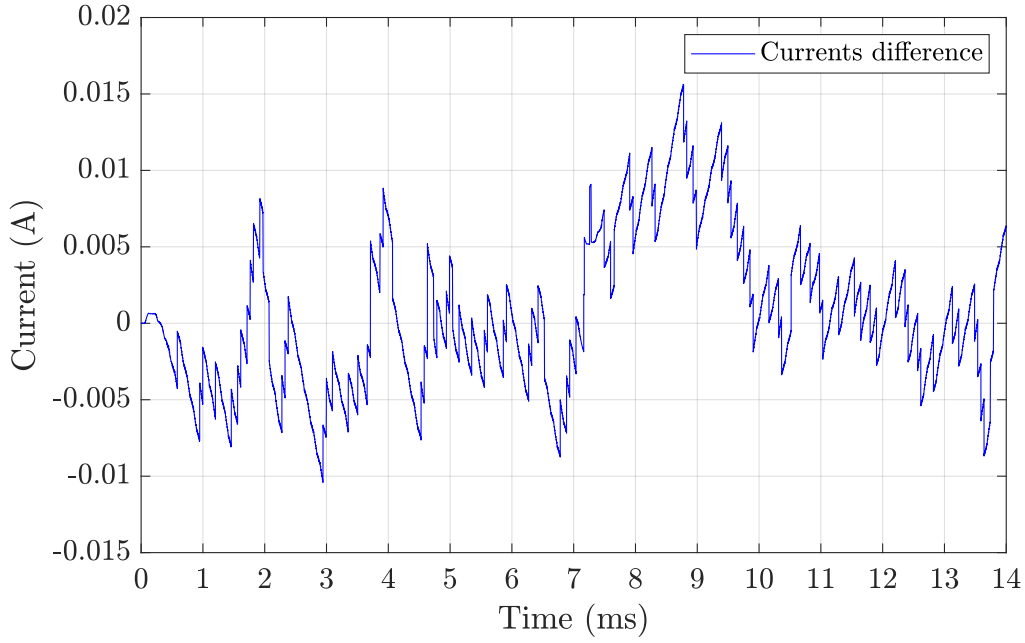


Figure 2.24: System with discrete time plant equivalent model block scheme.



(a)



(b)

Figure 2.25: Current scope simulation results: current in the two plants and reference value (a) and difference between the current waveforms (b).

2.6 Automatically Generated Code Modifications Summary

In the previous sections, it was mentioned that some changes in the produced VHDL need to be introduced in order to obtain the control system wanted behavior. For this reason, in this section the necessary modifications are listed for clarity: in particular, the current controller and the averager block generated codes need to be modified.

Starting from the averager block, as shown in Figure 2.19 its operation is triggered by the ADC_data_ready signal generated by the ADC manager, delayed by one clock cycle. As a result, also its outputs are updated in occurrence of this signal, Anyway, as previously described, the output update needs to be performed when the sampling trigger signal is produced. Therefore, this signal has to be added to the averager entity inputs list and the block output registers update need to be referred to the sampling trigger signal only.

For what concerns the current controller, as highlighted in Figure 2.19 its operation is triggered by the controller execution signal generated by the PWM modulator and delayed by one clock cycle, whereas its output update has to be triggered by the control voltage update signal. Similarly, the latter signal needs to be added to the current controller entity inputs list and the output registers update has to be performed in occurrence of the control voltage update signal instead of the controller execution one.

Chapter 3

Control System Implementation on Vivado and Simulations

In this chapter, the different control system components are implemented on a Xilinx FPGA module and experimentally validated. In particular, the device characteristics are hereby listed:

- Family: Artix-7 series;
- Device: XC7A35T;
- Package: CSG324;
- Speed: -2.

Starting from the components implementations on Simulink, presented in the previous chapter, the corresponding VHDL codes are automatically generated and are imported on the Xilinx Vivado Design Suite [11]. After that, the components are first simulated to ensure their correct behavior and are then implemented on the FPGA module, which is inserted on the printed circuit board (PCB) designed at the PEIC laboratory of the Politecnico di Torino reported in Figure 3.1. The main elements of the board are highlighted in the figure. For what concerns the experimental validation of the control system, only a few of these are used, apart from the FPGA board, the input power supply filter and the DC/DC converters:

- 2 ADCs among the many available, to sample the input voltage and the load current;
- The DAC, to visualize the sampled load current;
- The relays, to correctly configure the converter and the load as will be explained in the following;
- The analog and digital interfaces with the converter.

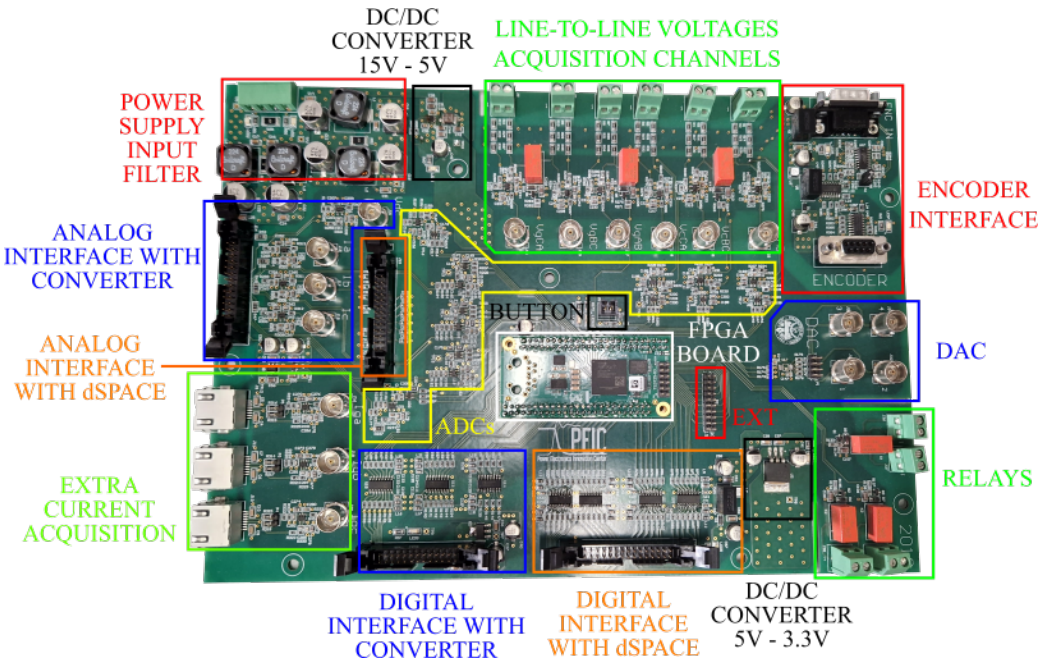


Figure 3.1: PCB organization.

In this chapter, the various reference signals and commands will be directly provided inside Vivado by means of the Virtual Input/Output (VIO) and Integrated Logic Analyzer (ILA) Intellectual Properties (IPs). In one of the following chapters, an SPI protocol will be implemented for the communication with dSPACE MicroLabBox, which will be exploited for the generation of these signals besides the visualization of the main converter quantities and state signals on a GUI. Therefore, also the digital interface with dSPACE will be used. Finally, the encoder interface will be exploited for the implementation of an electric motor torque control system in the last chapter.

3.1 Control System Experimental Validation Setup

The setup which is exploited for the experimental validation of each control system component is shown in Figure 3.2. In particular, it consists of:

- The PCB with the FPGA module;
- An oscilloscope;
- A power supply;
- A logic analyzer;
- A function generator.

In the figure, the FPGA programmer and an emergency push button, plugged in the connector denoted as BUTTON in Figure 3.1, are also present.

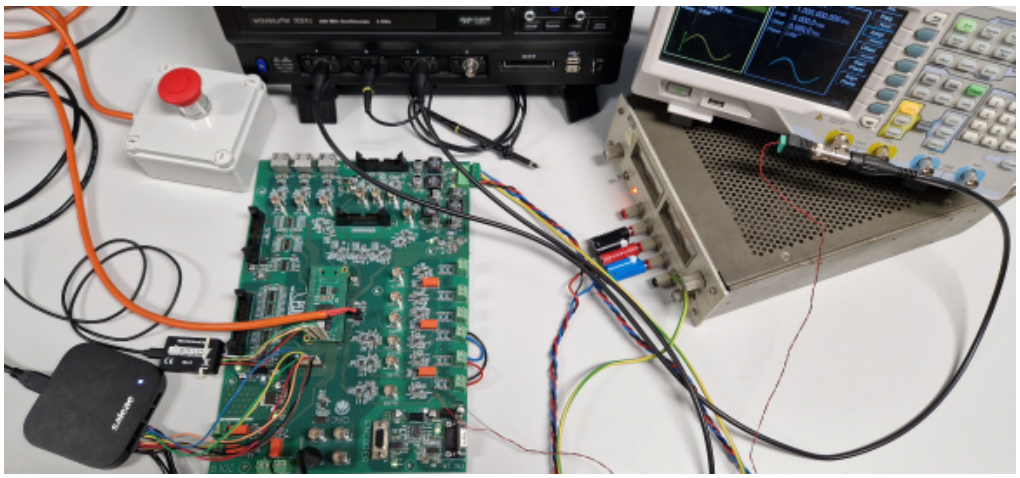


Figure 3.2: Setup for the control system components experimental validation.

After the completion of all the components preliminary tests, the whole control system is implemented on the FPGA and the board is connected, through the corresponding connectors highlighted in Figure 3.1, to the converter. This consists of an IGBTs power stack from Guasch, whose schematic, taken from its datasheet [12], is reported in Figure 3.3. Some important characteristics of the power stack are hereby reported:

- maximum DC voltage: 750 V;
- DC-link voltage measurable range: [0 - 750]V;
- IGBTs DC Collector current: 40 A;
- output current measurable range: [-40 - 40]A;
- maximum output RMS current per phase: 25 A, tested with a 10 kHz switching frequency and 50 Hz output frequency;
- minimum dead time: 1 µs.

In order to implement the H-bridge power converter presented in the previous chapters, some steps need to be executed: first, the relay needs to be left open, so that the input three-phase diode rectifier is not connected to the remaining part of the circuit. Secondly, a 600 V DC voltage is imposed across the capacitor. Thirdly, since the wanted converter is composed of two switching legs, only IGBTs T1, T2, T3 and T4 are driven, whereas the others are not used. Finally, the 4 mH load inductance is placed between the pins U and V which are highlighted in the figure. The actual inductor ESR is not measured; anyway,

as explained in the previous chapters, the control system is designed independently of its actual value.

The power stack provides also a temperature measurement, by means of an NTC sensor, and additional active low FAULT state signals, one for each IGBT, that indicate an error in the corresponding transistor driving voltage and therefore disables the driving operation, which can restart only after an active low signal, named EN_PWM in the following discussion, is provided, and the FAULT signals are reset at the high value. Notice that these state signals are registered and will be used for the design of a modules protection block in the following discussions.

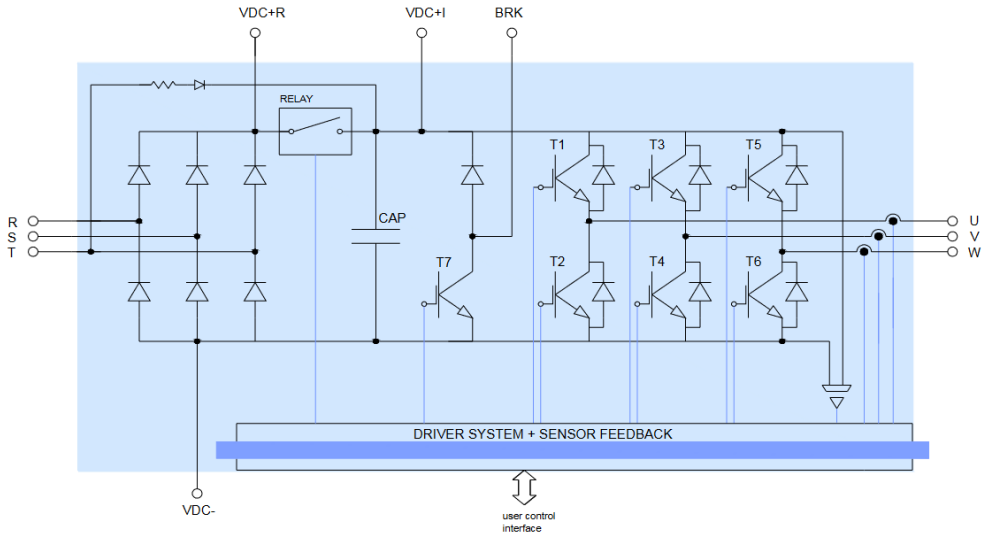


Figure 3.3: Guasch IGBTs power stack schematic from [12].

In the following sections, the various components of the control system will be implemented on the FPGA and experimentally tested. In order to clarify the various components placement inside the system, a block scheme which shows their connection is depicted in Figure 3.4. The implemented system operation can be easily summarized as follows: the load current and the input voltage values are sampled by exploiting two ADCs, whose operation is triggered by the sampling trigger signal generated by the PWM modulator as described in the previous chapter. The samples are directly fed to a DAC, whose conversion is triggered every time new samples are available. The moving average is applied to the two ADCs data samples and the averaged values are provided to the current controller. Consequently, the corresponding control voltage is computed and provided to the PWM modulator which generates the IGBTs switching functions, referring to Figure 2.14. Finally, a modules protection block is inserted: it receives the transistors switching functions, the two ADCs samples, a signal which is generated by the emergency button and the aforementioned FAULT signals. Basically, it is used to

enable the IGBTs driving signals only if a protection action is not necessary, as will be more clear in the following.

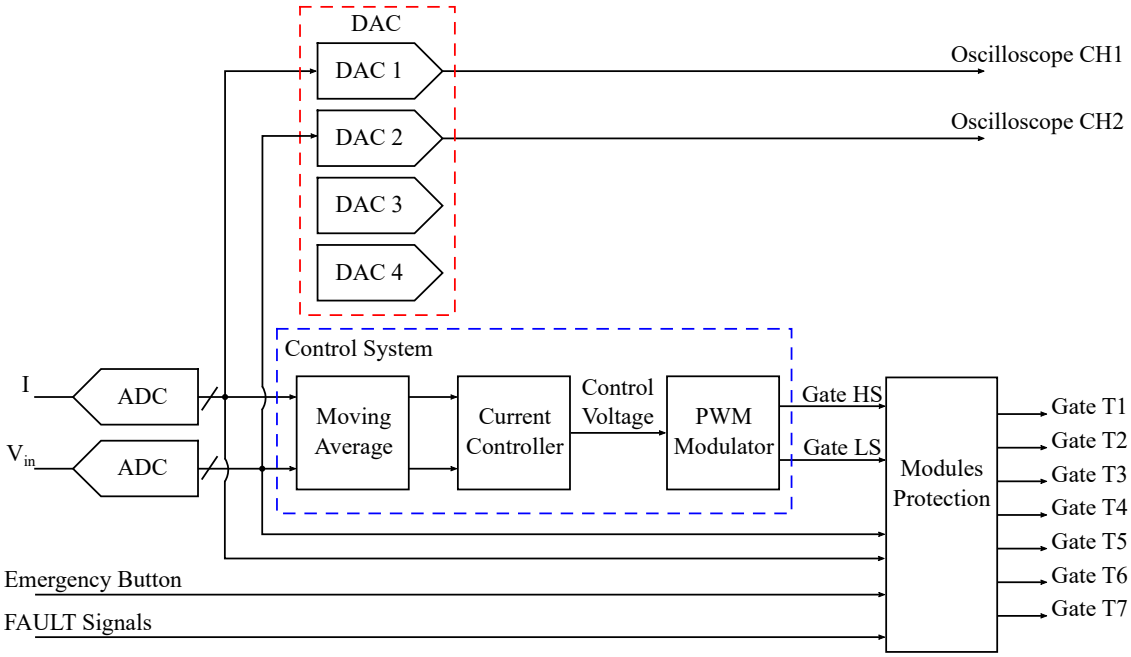


Figure 3.4: System components implemented on the FPGA.

3.2 ADC Implementation and Test

As explained in the previous chapter, for the control system execution the input voltage source and the load current need to be sampled. For this reason, one voltage sensor and one current sensor are exploited. As reported, the corresponding measurable ranges are [0 - 750]V and [-40 - 40]A. On the PCB, specific conditioning circuits are designed in order to adapt these ranges to the ADCs full-scale range (FSR). In particular, the B-grade AD7276 12-bit resolution ADCs from Analog Devices [13] are exploited: for performance reasons, the fastest Serial Peripheral Interface (SPI) protocol, among the two possible ones, is implemented and its timing diagram is reported in Figure 3.5. In particular, the Serial Clock (SCLK) signal allowed maximum frequency is 48 MHz and is generated only after the active low Chip Select ($\overline{CS}$) is driven low. In the considered case, the transmission length is 14 bits, hence 14 SCLK cycles are required. Then, a VHDL code is written in order to generate an apposite manager which, for each ADC, accurately produces the two signals and at the same time receives the Serial Data (SDATA) bits, composing the corresponding 12-bit data. Finally, a signal, named ADC_data_ready, is driven high for one system clock cycle when the conversion is completed and the sampled

data is available. Referring to Figure 3.5, the following important timing specifications need to be respected:

- minimum t_1 : 3 ns;
- minimum t_{QUIET}: 4 ns;
- minimum t_2: 6 ns.

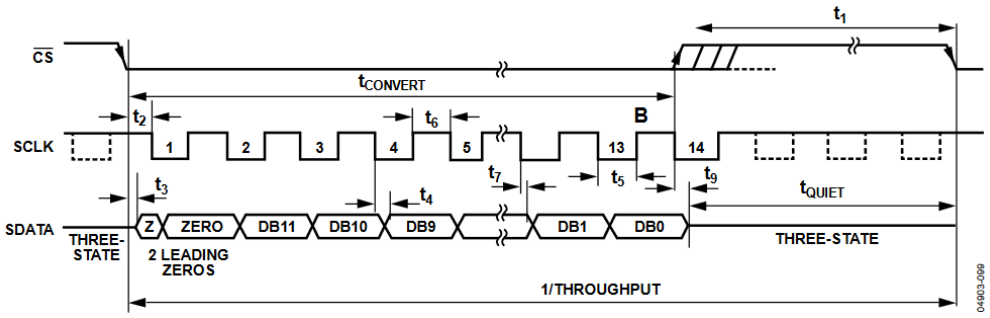


Figure 3.5: AD7276 SPI protocol timing diagram from [13].

The implemented ADC manager is composed of a Finite State Machine (FMS) whose state diagram is depicted in Figure 3.6. In particular, Figure 3.6a highlights the input signals which determine the next state depending on the current state, whereas Figure 3.6b reports the FSM outputs as functions of current state and inputs. The whole ADC manager scheme, depicted in Figure 3.7, highlights the presence of the various entity inputs and outputs, the FSM and the Execution Unit (EU). The latter is responsible for the correct signals timing specifications required by the SPI protocol besides the 12-bit word reconstruction. For this reason, it is composed of the following elements:

- a counter to set the correct SCLK signal frequency;
- a counter to compute the transitions of the SCLK signal, and consequently its cycles, to manage the transmission length;
- a counter to guarantee that the $\overline{CS}$ signal stays high for the minimum amount of time between two transmissions;
- a shift register, whose input is the SDATA signal, to receive and store all the 14 transmission bits;
- an output register, to provide the correct 12-bit data at the output;
- an edge-triggered D Flip-Flop, to generate the ADC_data_ready signal when the output data is available.

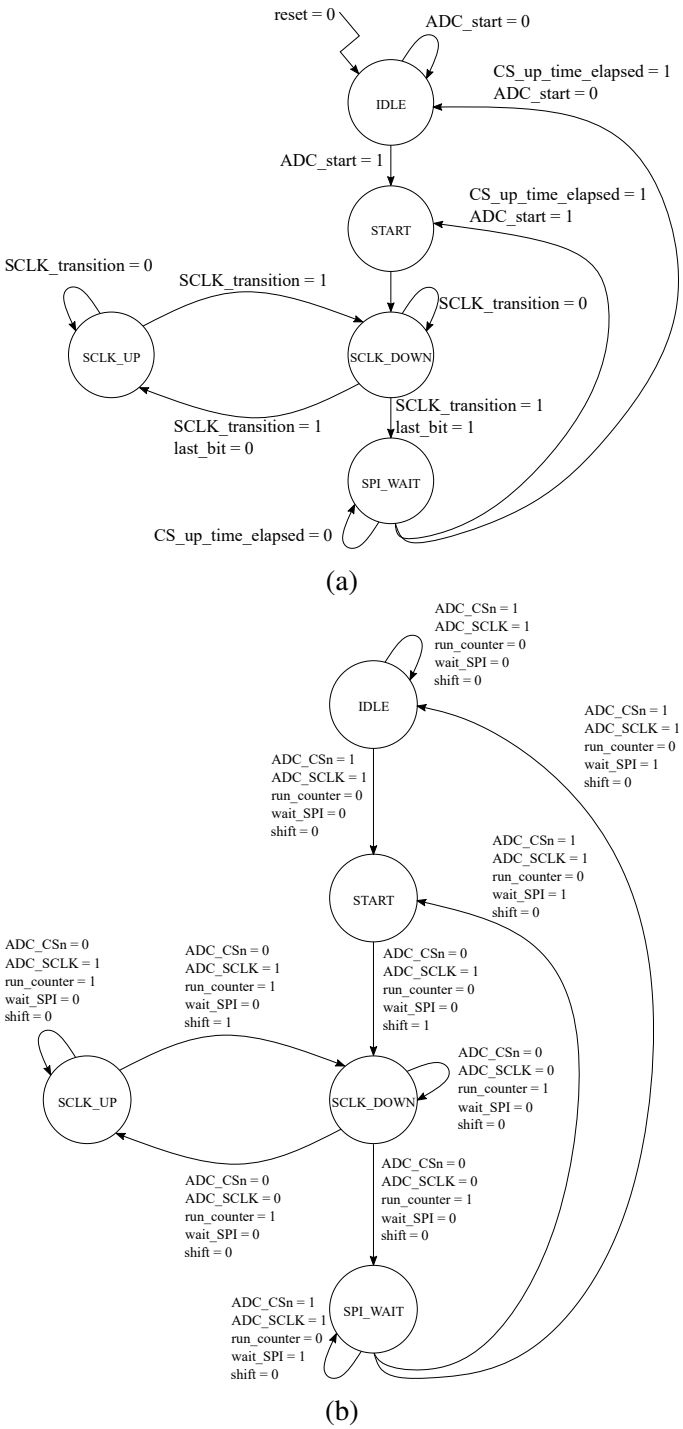


Figure 3.6: AD7276 manager FSM state updates (a) and outputs (b).

Depending on the FSM outputs, the blocks in the EU are accordingly enabled. On the other hand, the outputs of the different counters are exploited to trigger the FSM

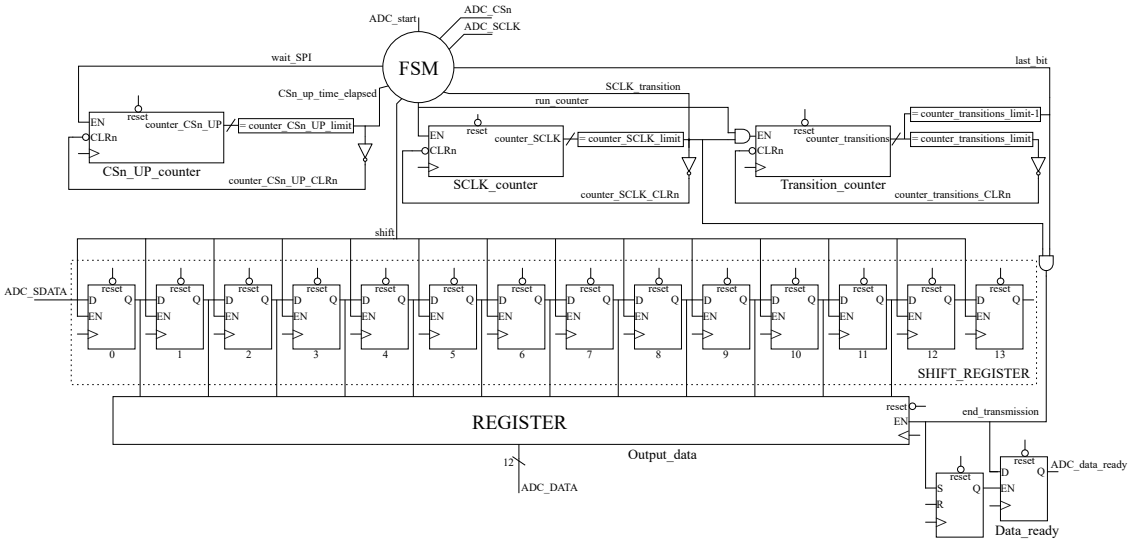


Figure 3.7: AD7276 manager entity organization.

state updates. In the figure, also an edge-triggered SR Flip-Flop is present: this is due to the fact that every ADC, when experimentally tested, generates a wrong first sampled data. For this reason, the inserted SR Flip-Flop goal is to neglect the first transmission, enabling the ADC_data_ready signal starting from the second conversion. For simplicity, the reset and clock signal lines are not indicated in the figure.

The ADC manager operation can be easily explained by noticing that, when the $\overline{CS}$ signal goes low, each transmission bit is provided by the ADC in correspondence of every SCLK falling-edge, therefore the shift register enable signal is provided on every SCLK rising-edge so that the input bit is correctly received. When all the 14 transmission bits are transferred, the output register is enabled to provide at the output the corresponding 12-bit data and the ADC_data_ready signal is driven high for one system clock cycle; at the same time, the $\overline{CS}$ signal is driven high for 12.5 ns.

In the figure, different programmable variables can be seen. In the following discussions, the hereby reported specific values are used:

- counter_SCLK_limit = 0;
- counter_transistions_limit = 27;
- counter_CSn_UP_limit = 0.

Considering a 80 MHz system clock frequency, these parameters values correspond to the generation of a 40 MHz SCLK signal, characterized by 14 cycles inside a transmission. The obtained t_1 and t_2 are both equal to 12.5 ns. Finally, it is important to underline that

a single ADC has been considered in Figure 3.7; anyway, since on the PCB, as already discussed, only the B-grade AD7276 ADCs are present, by deploying other shift and output registers and SDATA signal lines, the same manager can simultaneously handle more than one transmission at the same rate by providing the ADCs with the same SCLK and $\overline{CS}$ signals.

The ADC manager block design implemented in Vivado is reported in Figure 3.8. As shown, a Clocking Wizard IP generates the 80 MHz system clock frequency, whereas the VIO IP provides the ADC manager entity with the active low reset signal and with the always active ADC_start signal which triggers the conversion whenever possible. The ADC sampled data corresponds to the current flowing in the Guasch power stack node U, underlined in Figure 3.3, and is therefore expected to be approximately equal to 2047, since no current is flowing in the circuit. Then, the outputs of the manager can be visualized in Figure 3.9 by exploiting the ILA IP. In particular, in the figure, the $\overline{CS}$, SCLK, ADC_data_ready signals and the 12-bit data are reported in this order: as can be seen, the wanted operation is verified. Anyway, the obtained data does not exactly correspond to expected value: this is mainly due to the tolerances in the ADCs conditioning circuits which introduce an error in the sampled quantities. In the setup which is exploited for the control system experimental validation, this error magnitude is negligible for what concerns the sampled currents, whereas it is compensated for the input voltage one. For this reason, in the following discussions a proper multiplication factor will be considered in the Vin_GAIN term highlighted in Figure 2.17.

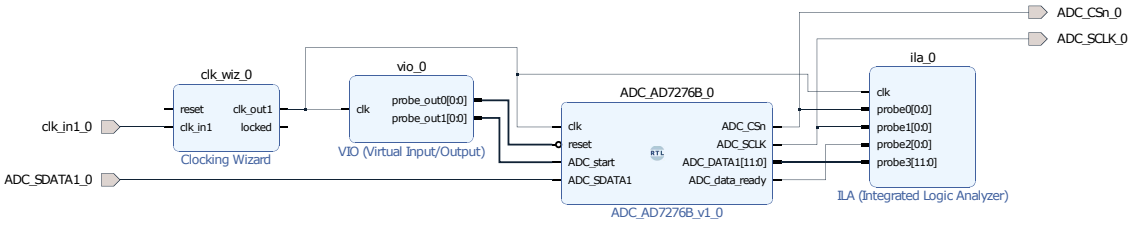


Figure 3.8: AD7276 manager block design in Vivado.

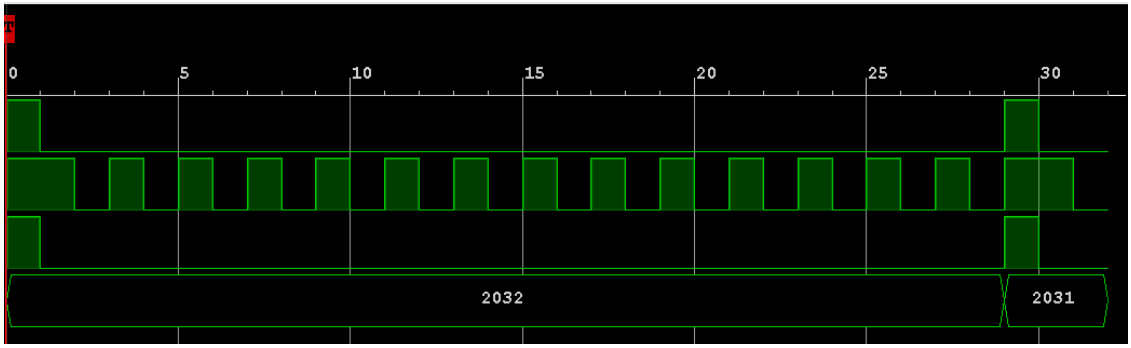


Figure 3.9: AD7276 manager operation waveforms.

It is important to compute the total conversion time in order to justify the previously reported multisampling factor N_{MS} choice. The maximum allowed time for the ADC conversion, hence the sampling trigger period, is computed in (3.1) recalling (2.2).

$$t_{ADC,MAX} = 32T_{clk} = 32\frac{1}{80\,\text{MHz}} = 400\,\text{ns} \tag{3.1}$$

The ADC conversion time can be in turn derived in (3.2) considering that the START state duration is equal to a system clock period and that the $\overline{CS}$ signal is brought high during the last SCLK signal rising-edge, as also highlighted in Figure 3.9. As computed, its value is below the maximum one and the chosen multisampling factor can be used.

$$t_{ADC} = t_{START} + 14T_{SCLK} = 12.5\,\text{ns} + 14 \cdot \frac{1}{40\,\text{MHz}} = 362.5\,\text{ns} < 400\,\text{ns} \tag{3.2}$$

Considering a given sampling instant, it is impossible, in the designed control system, to compute the average based on the most recent samples and to execute the current controller before the following sampling instant. For this reason, as mentioned in the previous chapter, the averager output data is updated on every sampling instant whereas the current controller is executed after the last sampling instant before the control voltage updating one. This strategy allows a whole sampling period for the current controller termination.

3.3 DAC Implementation and Test

On the PCB, the DAC124S085 12-bit DAC from Texas Instrument [14], which converts the input data to the corresponding value in the range [0 - 5]V, is present: it implements a 16-bit SPI transmission, whose timing diagram, taken from its datasheet, is depicted in Figure 3.10; referring to the figure, the following timing specifications need to be satisfied:

- maximum SCLK frequency (f_{SCLK}): 40 MHz;
- minimum t_{SYNC}: 10 ns;
- minimum t_{SS}: 10 ns.

Furthermore, on the board four BNC connectors, one for each DAC channel, are available and can be used to plot various signals waveforms on the oscilloscope.

The content of the 16-bit word, which is provided to the DAC through the serial data input DIN, is reported in Figure 3.11. As shown, apart from the 12-bit data which needs to be converted from digital to analog, the remaining 4 bits are exploited to select the address, hence one of the four channels responsible for the conversion, and to indicate the

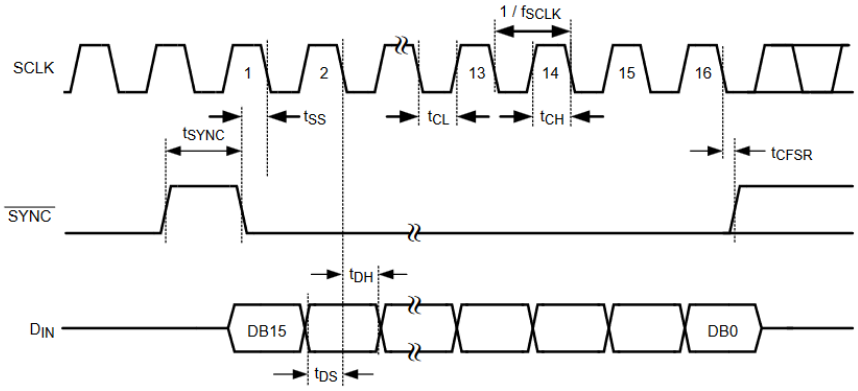


Figure 3.10: DAC124S085 SPI protocol timing diagram from [14].

operation code. For this reason, a VHDL code is written in order to implement a DAC manager which is responsible for the 16-bit word correct generation. In particular, when the data_ready signal is detected, it samples four 12-bit inputs, one for each channel, then attaches the corresponding addresses and operation codes; after that, it sends all the generated words to the DAC in four different transmissions, respecting the $\overline{SYNC}$ signal timing specification. The DAC outputs update is performed only after the fourth transmission has completed. Finally, notice that the DAC samples the serial data on the SCLK signal falling-edges: for this reason, the DAC manager sets every transmission bit value on every SCLK signal rising-edge.

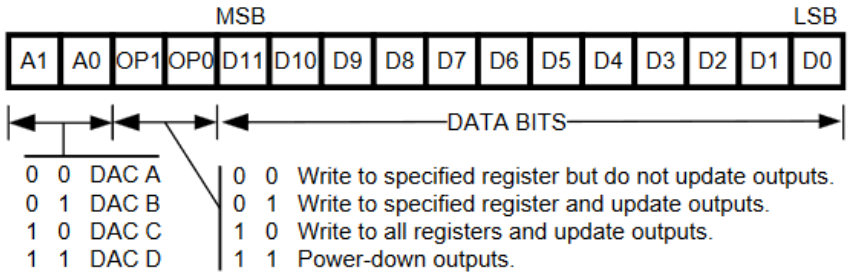


Figure 3.11: DAC124S085 register specifications from [14].

The DAC manager is composed of a Finite State Machine (FMS) whose state diagram is depicted in Figure 3.12. In particular, Figure 3.12a highlights the signals which determine the next state depending on the current state, whereas Figure 3.12b reports the FSM outputs corresponding to each current state and inputs. The whole DAC manager scheme is depicted in Figure 3.13, where the various entity inputs and outputs, the FSM and the EU are highlighted. Since it is simple to implement an even integer clock divider by exploiting a counter, the system clock frequency is set to 80 MHz, as aforementioned, in order to provide a 40 MHz SCLK signal through the operation of a specific counter.

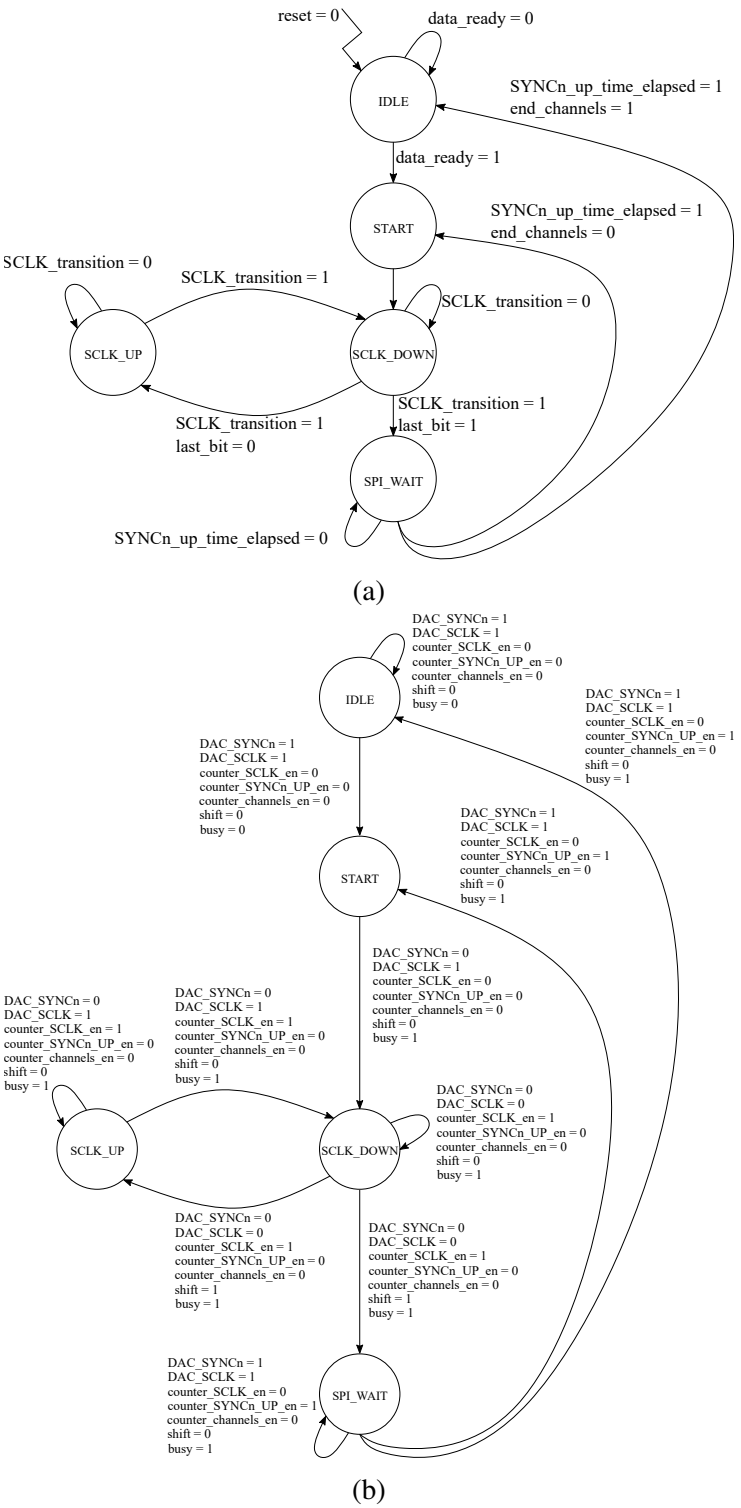


Figure 3.12: DAC124S085 manager FSM state updates (a) and outputs (b).

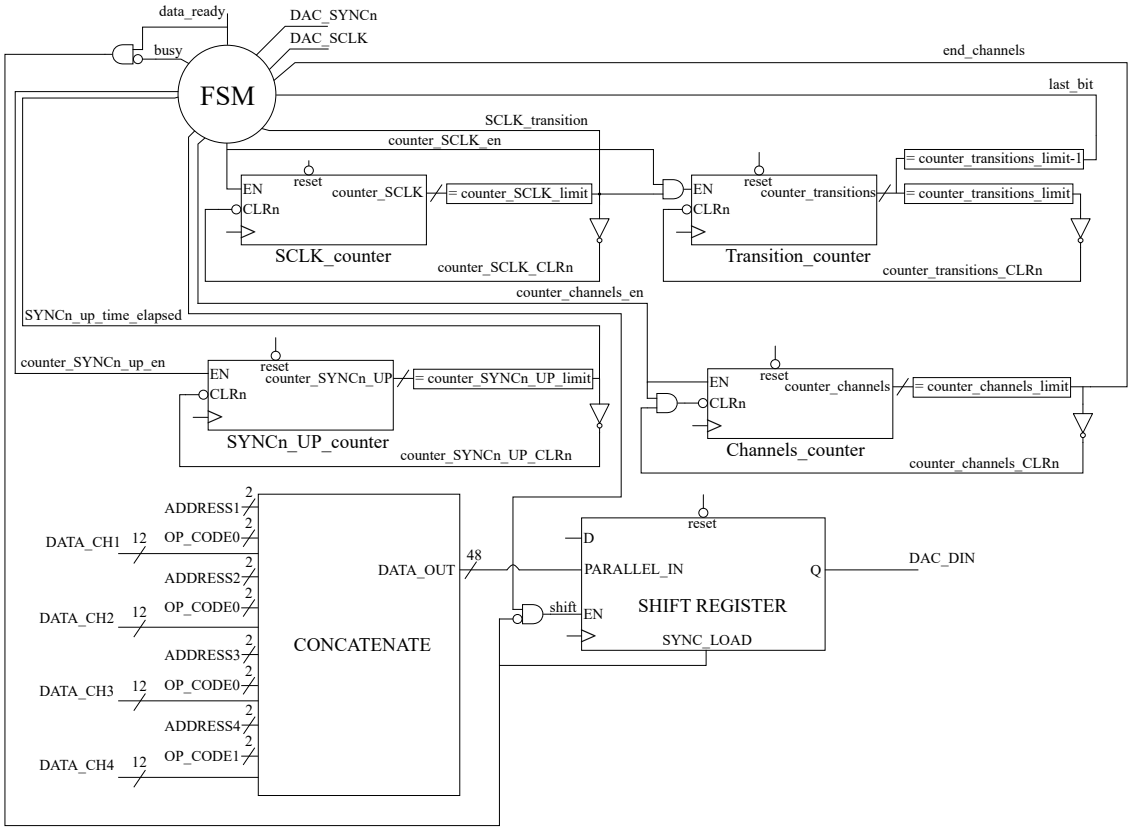


Figure 3.13: DAC124S085 manager entity organization.

With respect to the ADC case, the DAC manager presents similar blocks. In particular:

- a counter to set the correct SCLK signal frequency;
- a counter to compute the transitions of the SCLK signal, and consequently its cycles, to manage the transmission length;
- a counter to guarantee that the $\overline{SYNC}$ signal stays high for the minimum amount of time between two transmissions;
- a block which concatenates every channel 12-bit data with the corresponding address and operational code and then every 16-bit word with the other ones;
- a counter to keep track of the transmitted words;
- a shift register, with a synchronous parallel 48-bit input load and a serial output, Most Significant Bit (MSB) first.

The aforementioned sampling process is intrinsic to the shift register parallel loading: in fact, the signal which triggers the parallel loading operation is generated by the AND

operator between the data_ready signal and the logically negated busy signal. The latter is kept at the high value for all the four transmissions duration and therefore ensures that the state of the shift register is overwritten only when no transmission is ongoing. Also in this case, depending on the FSM outputs, the blocks in the EU are accordingly enabled and, on the other hand, the outputs of the different counters are exploited to trigger the FSM state updates.

In the following discussions, the programmable variables which are reported in the figure are set to the hereby listed values:

- counter_SCLK_limit = 0;
- counter_transistions_limit = 31;
- counter_SYNCn_UP_limit = 0;
- counter_channels_limit = 3.

These parameters values correspond to the generation of a 40 MHz SCLK signal, characterized by 16 cycles inside a transmission. Moreover, the obtained t_{SYNC} and t_{SS} are both equal to 12.5 ns and greater than the minimum required values.

The DAC manager block design implemented in Vivado is reported in Figure 3.14. As shown, the Clocking Wizard IP generates the 80 MHz system clock frequency, whereas the VIO IP provides the DAC manager entity with the active low reset signal, an always active data_ready signal to trigger the conversion whenever possible and the four channels respective 12-bit values to be converted. Then, the outputs of the manager can be visualized on the oscilloscope as depicted in Figure 3.15: in the figure legend, the 12-bit unsigned integer value data which are transmitted to the corresponding DAC channels are highlighted. As shown, the DAC correct operation is validated.

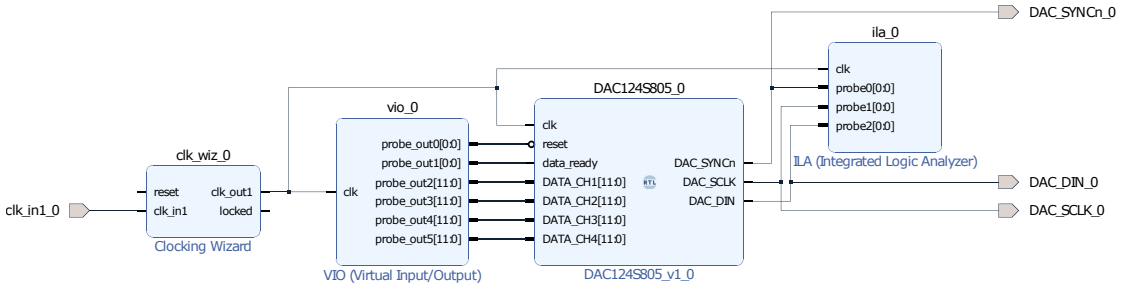


Figure 3.14: DAC124S085 manager block design in Vivado.

In order to further check the correct operation of ADC and DAC, the block design depicted in Figure 3.16 is implemented in Vivado: as shown, 4 ADCs are exploited in

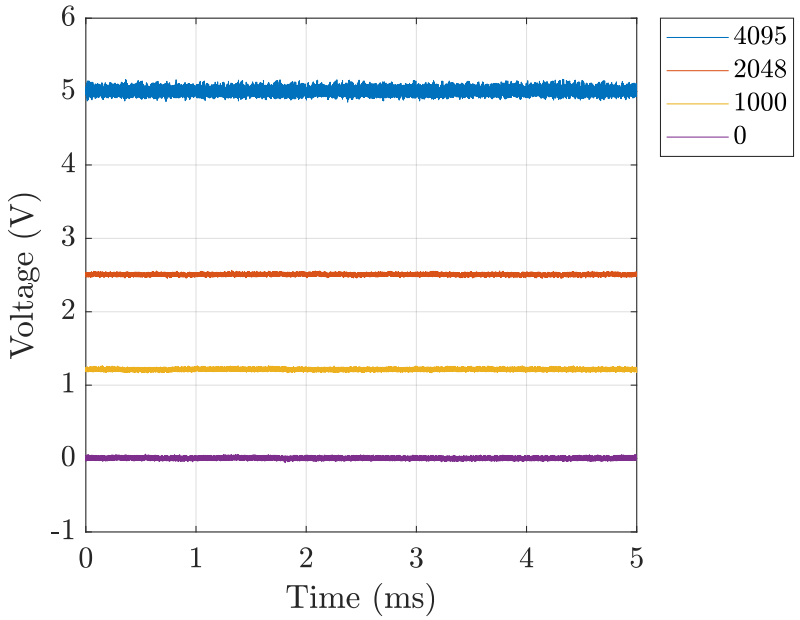


Figure 3.15: DAC124S085 oscilloscope waveforms.

this case and their sampled data are visualized on the oscilloscope by means of the four DAC channels. In particular, one of the ADCs is fed with a sinusoidal waveform by means of the function generator depicted in Figure 3.2, whereas the other ones sample the current flowing in the U, V, W nodes of the Guasch power stack represented in Figure 3.3 and, therefore, are expected to produce on the oscilloscope constant voltages whose value is around 2.5 V since no current is flowing in the converter. Anyway, as aforementioned, the actual obtained values are slightly different due to the tolerances of the ADC conditioning circuits. The result of the experiment is depicted in Figure 3.17: as can be seen, the obtained waveforms are the expected ones, demonstrating the two blocks correct behavior.

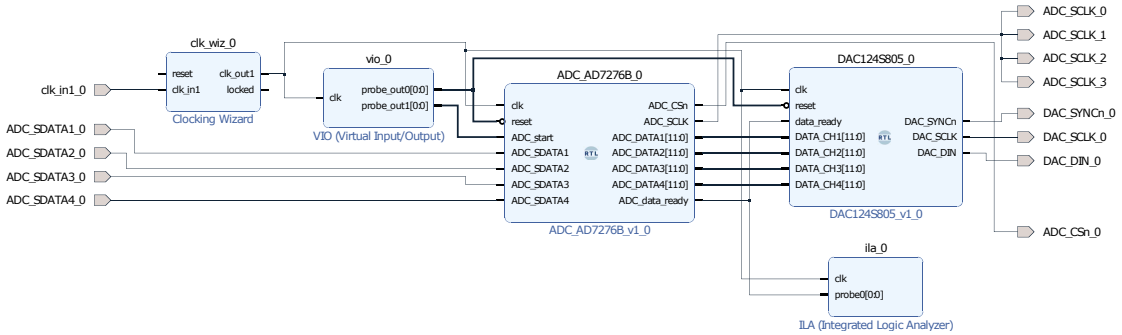


Figure 3.16: ADC7276 and DAC124S085 managers block design in Vivado.

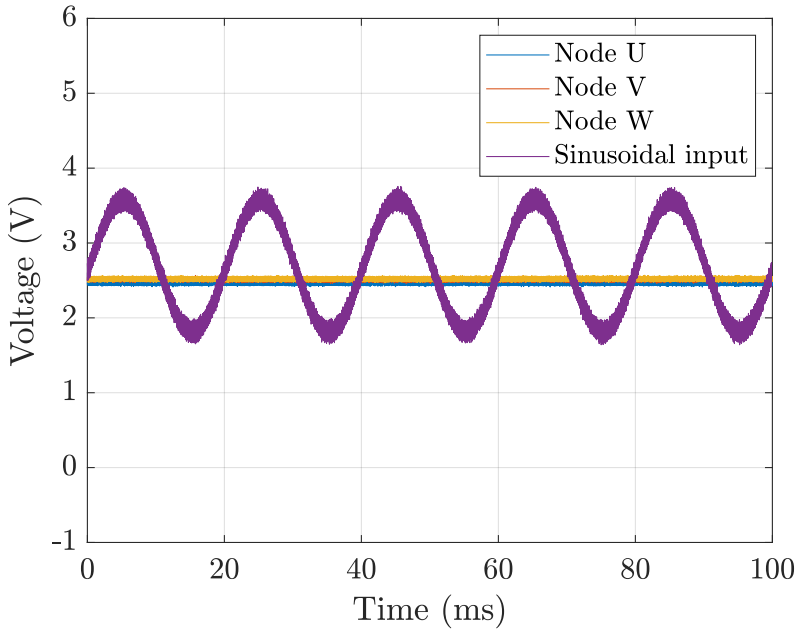


Figure 3.17: ADC7276 and DAC124S085 cascade oscilloscope waveform.

3.4 PWM Modulator Test

The PWM modulator block design deployed in Vivado is depicted in Figure 3.18. In particular, to validate its various features, the Logic Analyzer is exploited. In fact, all the PWM modulator outputs are connected to the EXT pin header. Some example operating waveforms are reported in Figure 3.19, considering a constant control voltage integer value equal to 2048, and the correspondence between each Logic Analyzer channel and PWM modulator output signal is summarized in Table 3.1.

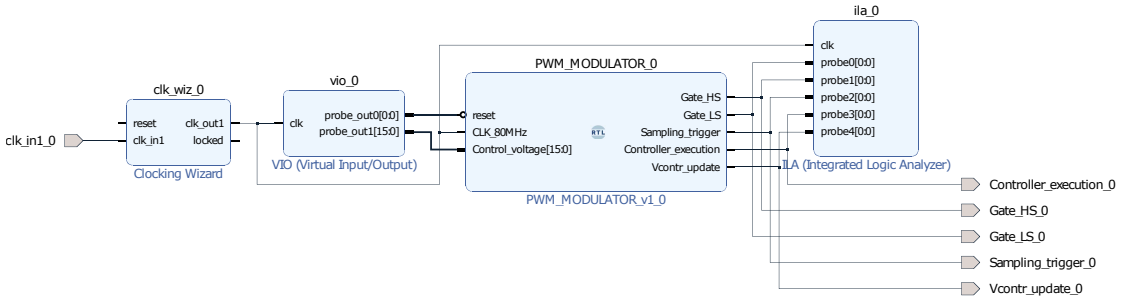


Figure 3.18: PWM modulator block design in Vivado.

As can be seen, the wanted dead time is inserted in the switching functions operation. Furthermore, the control voltage updating signal is placed in the middle points of the switching functions pulses, which coincide with the maximum and minimum values of the triangular carrier and, consequently, with the according sampling instants. Moreover,

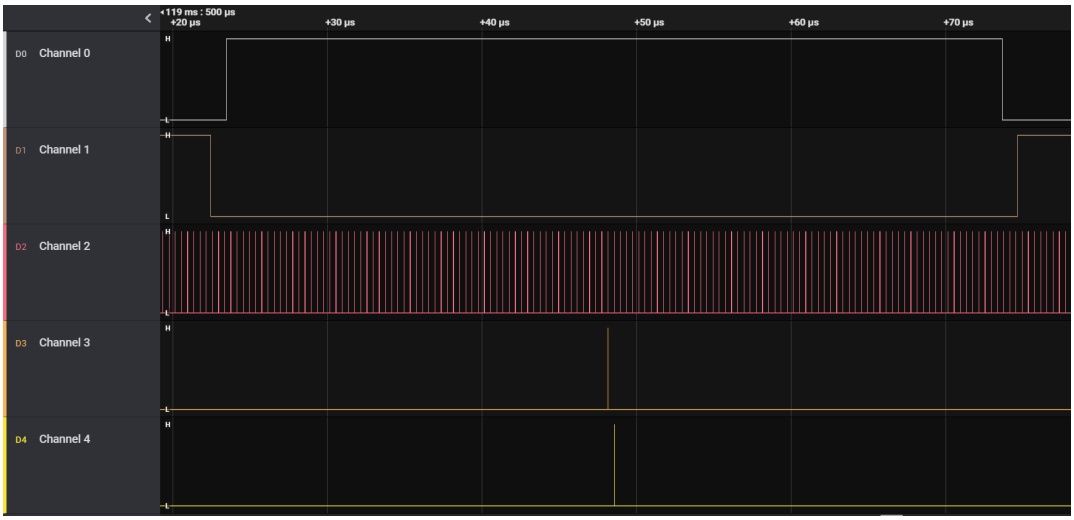


Figure 3.19: PWM modulator example waveforms obtained with the Logic Analyzer.

Logic Analyzer channels	PWM modulator outputs
Channel 0	Gate_HS
Channel 1	Gate_LS
Channel 2	Sampling_trigger
Channel 3	Controller_execution
Channel 4	Vcontr_update

Table 3.1: Logic Analyzer channels and PWM modulator outputs correspondence.

the controller execution signal is generated at the previous sampling instant with respect to the control voltage updating one. The additional clock cycle delay which was discussed in the previous chapter is, in fact, inserted at the control system level.

Finally, the correct overwriting and consequent necessary short circuit avoidance operations have been validated. An example is reported in Figure 3.20a, where the two switching functions and the control voltage update signal are reported when the control voltage value is equal to 40, hence to DT/2, and then changed: as shown, in the first part the high-side IGBT switching function is always set to its low value, whereas the low-side IGBT one is always high. When, instead, the control voltage is updated, the wanted dead time is provided in the switching functions operation avoiding the short circuit, as underlined in Figure 3.20b, which depicts a zoom on the control voltage variation region. Notice that only in this experimental validation the control voltage is not changed in correspondence of its updating signal, since it is provided through the VIO IP in Vivado. The correct operation is obtained when deploying also the current controller, which ensures that the control voltage is updated only in the correct instants.

The same result is similarly obtained for the case in which the control voltage value is set to 4055, hence equal to 2^N-1-DT/2: in that case, the high-side and low-side IGBTs switching functions are respectively always high and low until the control voltage value is changed. Anyway, also in this case the short circuit is correctly avoided by the insertion of the wanted dead time in the switching functions operations.

(a)

(b)

Figure 3.20: PWM modulator overwriting and short circuit avoidance operations when the control voltage is smaller than or equal to DT/2 example (a) and zoom on the dead time insertion in correspondence of the control voltage change (b).

3.5 Modules Protection Block Operation and Test

It is useful, when validating the whole control system operation, to insert a block in order to protect both the converter and the system in case control errors or failures in general occur. For this reason, the IGBTs gate driving signals are enabled only if no error is detected during the system execution: this is done by simply exploiting AND logic gates between the PWM modulator outputs and a specific signal, named PROTECTION, which is set to 0 whenever the driving system needs to be stopped, namely when one of the following conditions occurs:

- the load current exceeds given limits independently of its direction;
- the input voltage is higher than a given limit;
- the emergency button is pushed, producing an active high signal;
- one of the FAULT signals is driven low.

For what concerns the load current and the input voltage, the ADC sampled data is exploited: since it provides a 12-bit data in the corresponding range, the samples are compared with programmable 12-bit bounds accordingly set. In particular, the load current modulus is imposed to be always lower than 20 A, hence the corresponding following unsigned values are deployed for its bounds:

- load current maximum bound: 3071;
- load current minimum bound: 1025.

On the other hand, the maximum input voltage bound unsigned value is set to 4000, corresponding to a value which is higher than 700 V. For the bounds choice, it is important to consider that the deployed system implements a current control: an error in the control strategy has a strong, and dangerous, impact on the controlled current, as opposed to the input voltage, which is provided by a power supply.

In case a comparison operation results in a bound exceeding, a registered signal is imposed to 0. In a similar manner, when the emergency button is pushed, a registered signal is set to 0. The aforementioned PROTECTION signal is therefore obtained by means of an AND logic gate, whose operands are these two registered signals and all the FAULT signals. Notice that if one of the two registered signals is set to 0, the driving operation is immediately stopped and can only resume only when the active low reset signal is provided, whereas if a FAULT signal is driven low by the Guasch power stack, the driving operation is stopped until the EN_PWM signal is driven low for the required amount of time.

In the implemented case, the adopted multisampling strategy leads to a more effective protection block execution: in fact, the load current is monitored N_{MS} times inside a switching period, with respect to the 2 times characteristic of a double-sampling strategy for instance. This helps to quickly detect an overcurrent situation as opposed to other strategies.

Notice that the description above is referred to the deployed H-bridge converter case. This means that IGBTs T5, T6 and T7 are not driven by the PWM modulator outputs. Therefore, the protection block also provides an always null signal for these three transistors. Finally, it is important to highlight that the same protection block implementation can be easily extended also to other power converters and additional quantities can be monitored, as will be done in the following chapter.

The protection block corresponding VHDL code is implemented in Vivado and is validated. The deployed block design, which is characterized by the presence of other H-bridge converter control system blocks, is reported in Figure 3.21. In particular, the Clocking Wizard IP generates the 80 MHz system clock, whereas the VIO IP provides the active low reset signal and a constant control voltage value equal to 2048 to the PWM modulator, which is exploited to provide the gate driving signals to the protection block, whereas the Sampling_trigger signal is fed to the ADC manager in order to provide the ADC_start signal and accordingly trigger its sampling operation.

The Logic Analyzer is exploited to visualize the protection block outputs, namely all the gate driving signals, besides the PROTECTION signal, which are connected to the pin header denoted as EXT in Figure 3.1. The correspondence between the Logic Analyzer digital channels and the protection block output signals is reported in Table 3.2, whilst some operation waveforms are depicted in Figure 3.22. As shown, the T5, T6 an T7 IGBTs driving signals are always null, as wanted, whereas the switching functions of the other transistors are correctly provided accordingly to the set control voltage value. When the PROTECTION signal is driven low, all the protection block outputs are consequently set to 0, as expected: in the case shown in the figure, this is obtained by exploiting the function generator to provide the protection block with a value which exceeds the current bounds. However, the same waveforms are obtained also for the other conditions.

3.6 Averager Test

The averager correct operation is evaluated by providing its inputs with constant values and by checking the averaged outputs after N_{MS} execution cycles. For this reason, the averager block scheme depicted in Figure 2.13 is slightly modified in order to provide an active high signal, named Average_computed, after N_{MS} averager block executions, indicating that the average values have been computed.

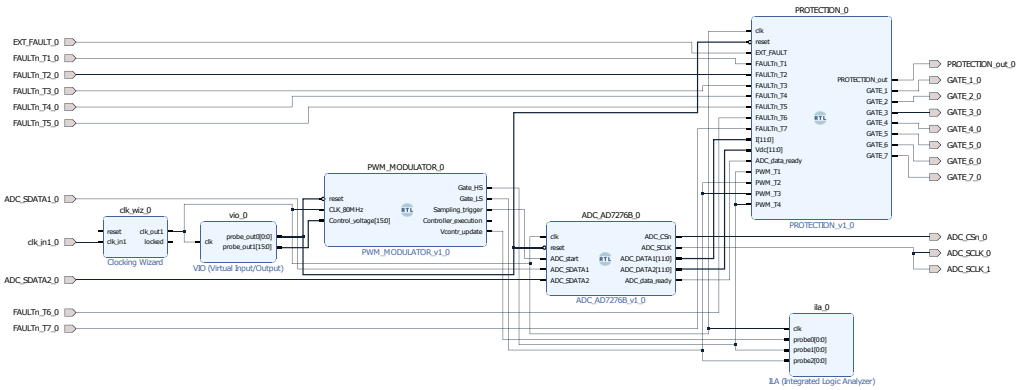


Figure 3.21: Protection block design in Vivado.

Logic Analyzer channels	Protection block outputs
Channel 0	Gate_1
Channel 1	Gate_2
Channel 2	Gate_3
Channel 3	Gate_4
Channel 4	Gate_5
Channel 5	Gate_6
Channel 6	Gate_7
Channel 7	PROTECTION

Table 3.2: Logic Analyzer channels and protection block outputs correspondence.

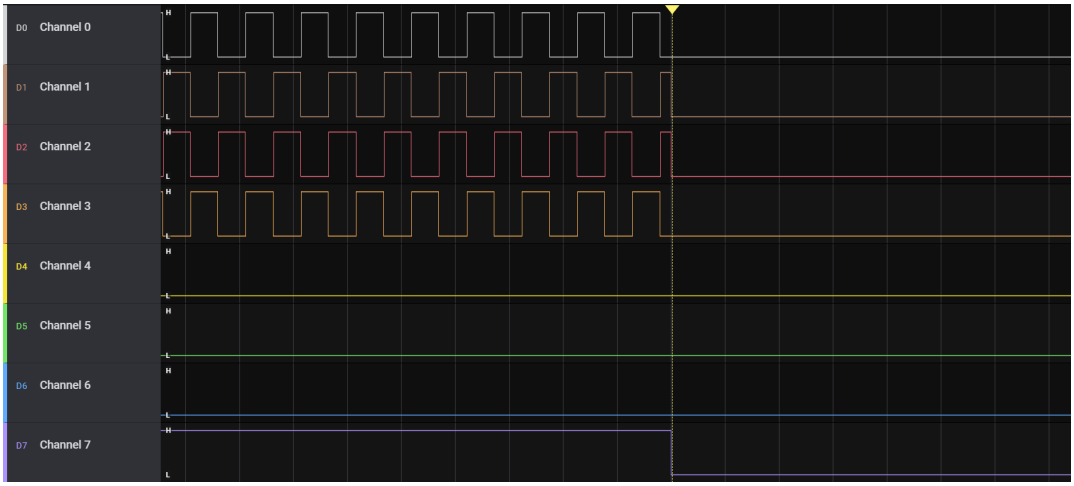


Figure 3.22: Protection block Logic Analyzer waveforms.

The block design implemented in Vivado is reported in Figure 3.23: the PWM modulator Sampling_trigger signal is exploited, only for this validation purpose, to provide the averager block with the ADC_data_ready signal, whereas the constant current and voltage values are generated by means of the VIO IP and the averager outputs are visualized through the ILA IP.

An example of the averager operation is depicted in Figure 3.24, where the constant current and voltage integer values are set to 4095 and 1000 respectively. In particular, the Sampling_trigger signal, the current and voltage computed average values and the Average_computed signal are shown in the figure in this precise order. As highlighted, the correct average values are computed after N_{MS} block executions, as indicated by the appropriate signal.

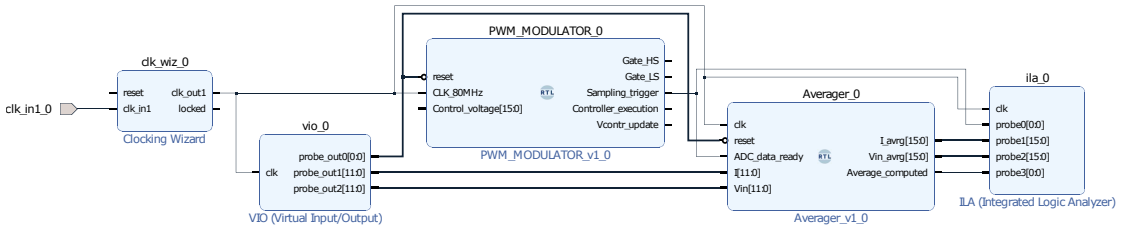


Figure 3.23: Averager block design implementation in Vivado.

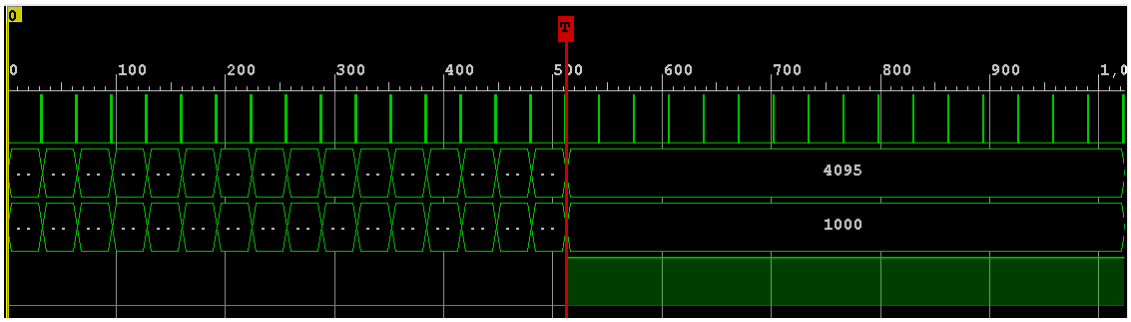


Figure 3.24: Averager block operation example.

Finally, notice that the averager block outputs change one clock cycle after the sampling trigger signal is received: this is, as wanted, due to the presence of the delay block placed on its trigger port as shown in the whole control system block scheme reported in Figure 2.19.

3.7 System Test with Discrete Time Plant Equivalent Model

In this section both the whole control system and the discrete time plant equivalent model VHDL code, derived from the block scheme reported in Figure 2.24 which implements a DSDU strategy, is imported in Vivado in order to perform a simulation to verify the system correct operation. For this reason, a testbench has been written to generate the proper reset and clock signals besides the reference current signal: in particular, the latter is initially set to 20 A to then be changed, after 7 ms, to −20 A, whereas the whole simulation is performed for 14 ms. Notice that these parameters correspond to the settings exploited for the Simulink simulations depicted in Figure 2.25, in order to be able to compare that simulation with the one performed in Vivado. The result of the latter is reported in Figure 3.25. As shown, the obtained load current behavior is similar to that depicted in Figure 2.25a and the control system correct operation is therefore verified.

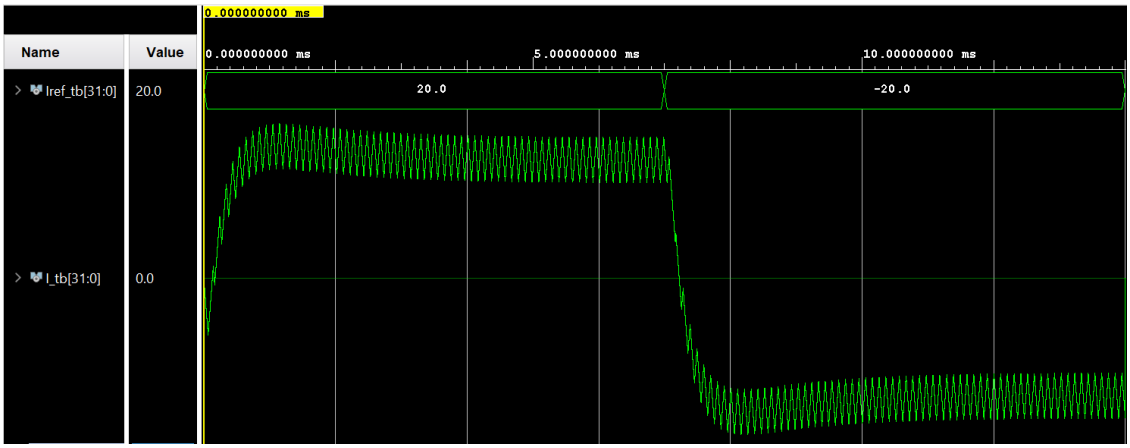


Figure 3.25: Control system and discrete time plant equivalent model simulation.

3.8 Control System Experimental Validation

The whole control system is experimentally validated in this section, exploiting the setup depicted in Figure 3.26. As shown, the PCB is connected to the Guasch power stack through the corresponding connectors underlined in Figure 3.1. A current probe is exploited to measure the load current which is showed on the oscilloscope. The power supply provides the wanted 600 V input voltage source. The inductive load cannot be seen in the figure, anyway it is connected to the switching legs as specified in the previous chapter: in particular, a specific control signal, named K1, configures a relay in order to connect the inductive load to the circuit. The default configuration for the Guasch power

stack relay, which is underlined in Figure 3.3, is open as wanted: for this reason, in the following discussion no specific control signal will be provided.

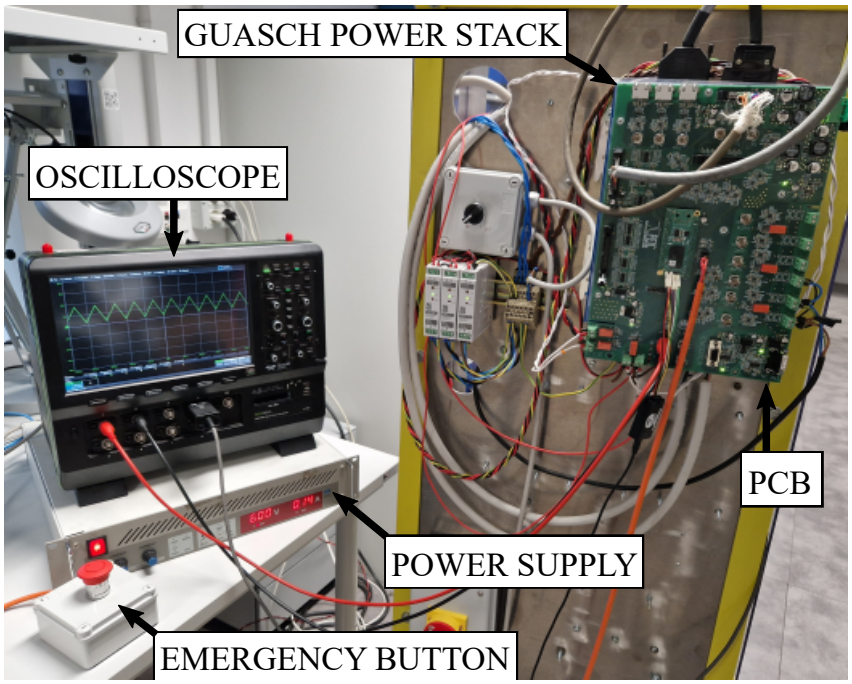


Figure 3.26: Control system experimental validation setup.

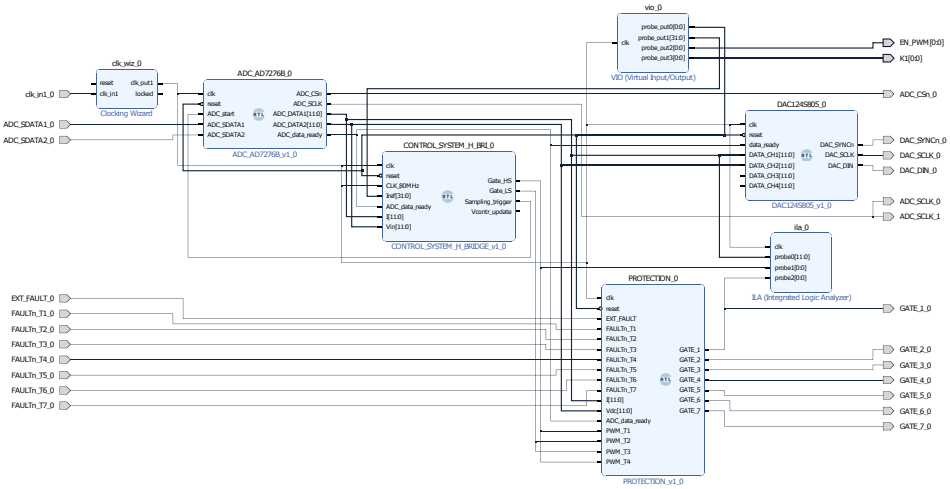


Figure 3.27: Control system block design implementation in Vivado.

The control system block design implemented in Vivado is reported in Figure 3.27. As shown, the main previously described components are present: the ADC and DAC

managers, the modules protection block and the control system block, composed of the averager, the current controller and the PWM modulator. The VIO IP provides the current reference value Iref, in single floating-point precision, besides the reset, K1 and EN_PWM signals. Various reference average current values have been tested; the corresponding load current waveforms are depicted in Figure 3.28, where the tested reference values are highlighted in the figure legend. The corresponding load current average values, computed by the oscilloscope, are reported in Table 3.3. As shown, the load current is controlled with good accuracy, as expected.

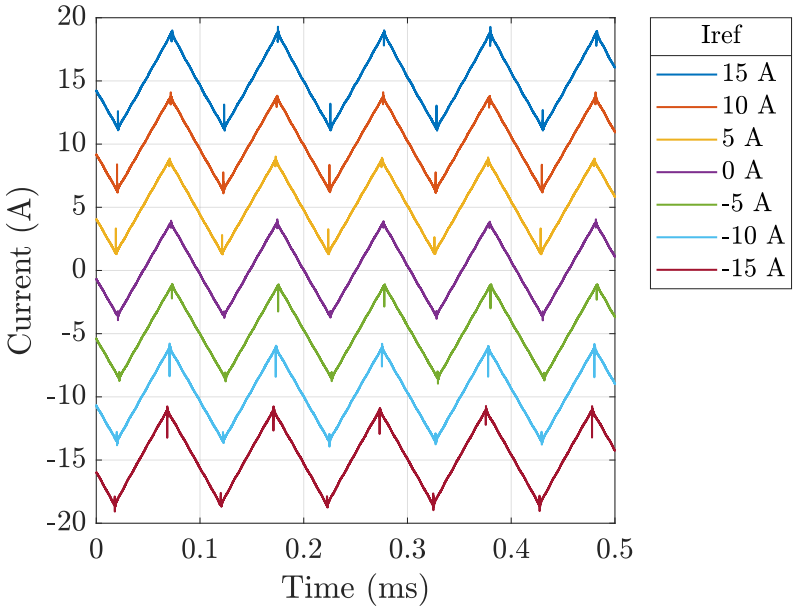


Figure 3.28: H-bridge converter load current waveforms.

Reference average current value	Load current average value
15 A	14.965 A
10 A	10.013 A
5 A	5.048 A
0 A	0.087 A
−5 A	−4.851 A
−10 A	−9.817 A
−15 A	−14.766 A

Table 3.3: Reference and measured load current average values comparison.

Chapter 4

Three-Phase Inverter Average Current Control Implementation

In this chapter, the previously designed average current control system is extended to a three-wire three-phase inverter with a star connected inductive load, whose topology is depicted in Figure 4.1. After a quick review of the converter operation, the Clarke's and Park's transfoms are presented, since they can be exploited to obtain a more effective control system. For this reason, their block schemes are implemented and simulated on Simulink and experimentally tested by deploying them on the FPGA, importing the automatically generated VHDL code. After that, they are inserted in the previously presented control system, which is adapted to the converter specifications and operation. Moreover, a carrier-based space vector PWM modulation is implemented to increase the system performance. Finally, the whole control system is implemented on the FPGA and experimentally validated by testing it on the three-phase inverter realized with the Guasch power stack reported in Figure 3.3. For this reason, in Figure 4.1 the corresponding transistors which are driven are highlighted.

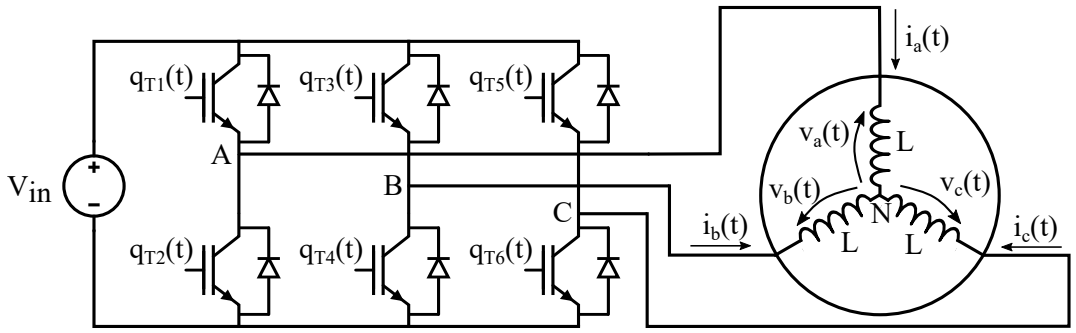


Figure 4.1: Three-phase inverter topology.

4.1 Three-phase Inverter Operation

In this section, the operation of the three-phase inverter depicted in Figure 4.1 is reviewed referring to a sinusoidal-PWM (SPWM), in which each switching leg is modulated independently from the others. As explained in [1][3][4], the PWM modulator operation is identical to that reported in Chapter 1 for the two-quadrant DC/DC converter, comparing in this case the triangular carrier with a set of three-phase sinusoidal waveforms, hence with same frequency and amplitude but with 120° phase shift with each other. For this reason, the control voltage for each phase needs to be computed appropriately, as will be explained in this section referring to [2][3][4].

First, it is useful to rearrange the input voltage source as reported in Figure 4.2 where a virtual ground is highlighted. In the following discussion, $v_{A0}(t)$, $v_{B0}(t)$ and $v_{C0}(t)$ will be referred as line-to-ground voltages, whereas $v_a(t)$, $v_b(t)$ and $v_c(t)$ are named phase voltages.

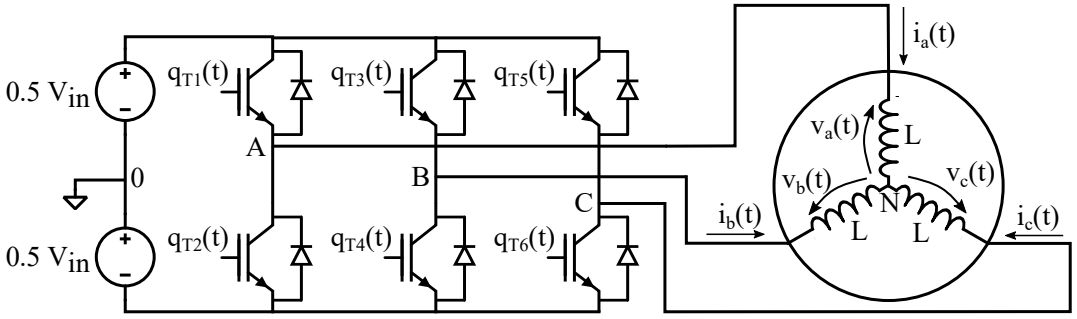


Figure 4.2: Three-phase inverter topology with virtual ground.

Each switching leg operation can be modeled, referring to the virtual ground, as a square wave: taking as an example phase A and considering complementary switching functions in the same leg, its line-to-ground voltage can be expressed as:

$$v_{A0}(t) = \frac{V_{in}}{2} q_{T1}(t) - \frac{V_{in}}{2} q_{T2}(t) = \frac{V_{in}}{2} q_{T1}(t) - \frac{V_{in}}{2}(1 - q_{T1}(t)) = V_{in} q_{T1}(t) - \frac{V_{in}}{2} \quad (4.1)$$

By considering the moving average quantities in (4.1) and recalling (1.5), (4.2) can be derived, where the duty cycle of the phase $d_A(t)$ is underlined.

$$V_{A0}(t) = \frac{1}{T_S} \int_t^{t+T_S} (V_{in} q_{T1}(t) - \frac{V_{in}}{2}) \mathrm{d}\tau = V_{in} d_A(t) - \frac{V_{in}}{2} = V_{in}(d_A(t) - \frac{1}{2}) \quad (4.2)$$

Recalling (1.22) and that the relation is still valid for a slow varying control voltage, expression (4.3) can be derived. This is a fundamental aspect, since by imposing that each phase control voltage is a sinusoidal signal whose frequency is much lower than

the switching frequency, each line-to-ground voltage moving average will be a sinusoidal signal at the same frequency.

$$V_{A0}(t) = V_{in}(\frac{v_{contr,A}(t)}{V_{TR}} - \frac{1}{2}) \tag{4.3}$$

By requiring $V_{A0}(t)$ to be a sinusoidal voltage at the fundamental angular frequency ω with a given amplitude $\hat{V}$, (4.3) can be rearranged to find the control voltage expression for the switching leg A in (4.4).

$$v_{contr,A}(t) = [\frac{\hat{V}}{V_{in}} sin(\omega t) + \frac{1}{2}]V_{TR} \tag{4.4}$$

Similar derivations can be made also for the other phases, considering the according switching functions and defining the corresponding duty cycles and control voltages. In particular, by requiring the moving average line-to-ground voltages to be three-phase sinusoidal waveform, as expressed in (4.5), (4.6) is obtained.

$$\begin{cases} V_{A0}(t) = \hat{V} sin(\omega t) \\ V_{B0}(t) = \hat{V} sin(\omega t - \frac{2\pi}{3}) \\ V_{C0}(t) = \hat{V} sin(\omega t + \frac{2\pi}{3}) \end{cases} \tag{4.5}$$

$$\begin{cases} v_{contr,A}(t) = (\frac{V_{A0}(t)}{V_{in}} + \frac{1}{2})V_{TR} = [\frac{\hat{V}}{V_{in}} sin(\omega t) + \frac{1}{2}]V_{TR} \\ v_{contr,B}(t) = (\frac{V_{B0}(t)}{V_{in}} + \frac{1}{2})V_{TR} = [\frac{\hat{V}}{V_{in}} sin(\omega t - \frac{2\pi}{3}) + \frac{1}{2}]V_{TR} \\ v_{contr,C}(t) = (\frac{V_{C0}(t)}{V_{in}} + \frac{1}{2})V_{TR} = [\frac{\hat{V}}{V_{in}} sin(\omega t + \frac{2\pi}{3}) + \frac{1}{2}]V_{TR} \end{cases} \tag{4.6}$$

A fundamental aspect can be found by noticing that, in the three-wire case, the following equation holds:

$$i_a(t) + i_b(t) + i_c(t) = 0 \tag{4.7}$$

In fact, under this constraint the sum of the three phase voltages $v_a(t)$, $v_b(t)$ and $v_c(t)$ is always null, as shown in (4.8). Consequently, (4.9) can be derived.

$$v_a(t)+v_b(t)+v_c(t) = L\frac{\mathrm{d}i_a(t)}{\mathrm{dt}}+L\frac{\mathrm{d}i_b(t)}{\mathrm{dt}}+L\frac{\mathrm{d}i_c(t)}{\mathrm{dt}} = L\frac{\mathrm{d}(i_a(t)+i_b(t)+i_c(t))}{\mathrm{dt}} = 0 \tag{4.8}$$

$$v_{A0}(t)+v_{B0}(t)+v_{C0}(t) = v_a(t)+v_{N0}(t)+v_b(t)+v_{N0}(t)+v_c(t)+v_{N0}(t) = 3v_{N0}(t) \tag{4.9}$$

After that, the moving averages quantities can be considered in (4.9), obtaining (4.10).

$$V_{A0}(t) + V_{B0}(t) + V_{C0}(t) = 3V_{N0}(t) \tag{4.10}$$

Notice that the left part of the equation, recalling (4.5), consists of the sum of three sinusoidal voltages with same amplitude and 120° phase shift, whose result is always 0.

As a consequence, also $V_{N0}(t)$ is always null, and the neutral can be considered connected to the virtual ground. This is a crucial aspect, since the phase voltages moving averages are equal to the corresponding line-to-ground voltages in the three-wire case, as shown in (4.11). Furthermore, by imposing the control voltages as derived in (4.6), the obtained phase voltages moving averages describe the wanted three-phase system.

$$\begin{cases} V_a(t) = V_{A0}(t) - V_{N0}(t) = V_{A0}(t) = \hat{V} sin(\omega t) \\ V_b(t) = V_{B0}(t) - V_{N0}(t) = V_{B0}(t) = \hat{V} sin(\omega t - \frac{2\pi}{3}) \\ V_c(t) = V_{C0}(t) - V_{N0}(t) = V_{C0}(t) = \hat{V} sin(\omega t + \frac{2\pi}{3}) \end{cases} \tag{4.11}$$

The same conclusions can be drawn also for instance for LRE loads, as long as the inductances and resistances values are identical in every phase and that the sum of the three back-emf voltages is always null [3]. Anyway, the load has been considered to be composed of inductors only since this is the system which will be experimentally tested at the end of this chapter.

From a control system point of view, it is useful to notice that if the three control voltages are properly generated, the three-phase voltages are correctly applied to the load. Furthermore, since the neutral point is connected to the virtual ground, as aforementioned, each phase can be modeled as shown in Figure 4.3, where phase A, an LRE load and moving averages quantities are considered.

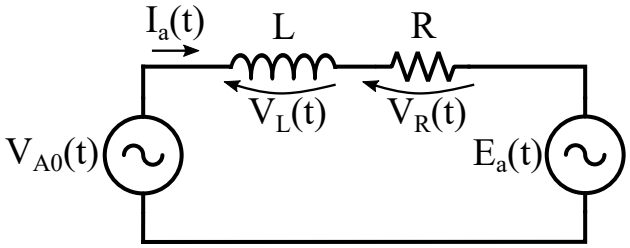


Figure 4.3: Three-phase inverter phase equivalent model considering moving averages.

As can be seen, this is similar to the model depicted in Figure 1.3 if the moving averages are considered: in this case, the voltage which is applied to the load and the back-emf are sinusoidal voltages. As a result, all the considerations about the two-quadrant DC/DC converter are still valid in this case for each phase, as long as the proper adjustments are introduced. In particular, considering that each phase voltage is a sinusoidal waveform and that a low-pass filtering operation is introduced by the load, the current in each phase is respectively a sinusoidal waveform at the fundamental frequency. As a consequence, starting from the three-phase voltages, three-phase currents are obtained in the star connected load.

As aforementioned, the equivalent model depicted in Figure 4.3 is valid for each phase: for what concerns the average current control system implementation, this means that three different PI regulators, one for each phase, need to be deployed. Furthermore, with respect to the average current controller presented in Chapter 1, the following considerations need to be taken into account: first, for each PI regulator, the K_P and K_I expressions derived in (1.40) are still valid and have to be referred to the phase inductance value. Secondly, by noticing that each PI regulator output is the line-to-ground voltage, whose value is in the range $[\frac{-V_{in}}{2}$ - $\frac{V_{in}}{2}]$, the proportional and integral parts, considering also a feedforward technique in general, need to be limited and saturated by accordingly adapting (1.46), (1.47) and (1.48). Finally, a different strategy has to be adopted for the control voltage generation: since the output of the PI regulator is the line-to-ground voltage, as computed in (4.6) it has to be divided by V_{in}, then 0.5 needs to be summed and in the end the triangular carrier amplitude is multiplied to the result. Each control voltage is then compared to the triangular carrier by means of three comparators, one for each phase, inside the PWM modulator block, to generate the transistors switching functions [3][4].

In the described case, three PI regulators are needed; anyway, a simplified and better performing control system can be implemented by noticing that the three phases are not independent from each other by means of (4.7) and by exploiting Clarke's and Park's transforms.

4.2 Clarke's and Park's Transforms

As explained in [2][15], in a three-phase electric system each phase is self-consistent from a mathematical point of view and can be modeled separately from the other ones, as was done in the previous section. Anyway, due to the three-wire physical constraint reported in (4.7), an equivalent two-phase system can be obtained. From a control point of view, this means that only two PI regulators can be deployed instead of three. This simplification is achieved by applying the Clarke's transform to a set of three-phase variables, transforming the abc three-phase reference frame into the $\alpha\beta$ one. This leads to a two-phase equivalent description of the three-phase system. In this case, the transformation matrix is reported in (4.12) in the amplitude invariant case.

$$\begin{bmatrix} x_\alpha \\ x_\beta \end{bmatrix} = \frac{2}{3}\begin{bmatrix} 1 & \frac{-1}{2} & \frac{-1}{2} \\ 0 & \frac{\sqrt{3}}{2} & \frac{-\sqrt{3}}{2} \end{bmatrix}\begin{bmatrix} x_a \\ x_b \\ x_c \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & \frac{-1}{3} & \frac{-1}{3} \\ 0 & \frac{\sqrt{3}}{3} & \frac{-\sqrt{3}}{3} \end{bmatrix}\begin{bmatrix} x_a \\ x_b \\ x_c \end{bmatrix} \tag{4.12}$$

In particular, by recalling (4.7), it can be further simplified as shown in (4.13).

$$\begin{cases} x_\alpha = x_a \\ x_\beta = \frac{1}{\sqrt{3}}x_b - \frac{1}{\sqrt{3}}x_c \end{cases} \tag{4.13}$$

The inverse Clarke's transform is derived in (4.14) by reversing (4.13) and recalling (4.7).

$$\begin{cases} x_a = x_\alpha \\ x_b = \frac{\sqrt{3}}{2} x_\beta - \frac{1}{2} x_\alpha \\ x_c = -\frac{\sqrt{3}}{2} x_\beta - \frac{1}{2} x_\alpha \end{cases} \tag{4.14}$$

An improvement in the average current controller performance can be achieved by exploiting the rotating reference frame dq instead of the fixed $\alpha\beta$ one. As explained in [2], by applying the Park's transform, the dq reference frame rotates around the $\alpha\beta$ one at a constant angular frequency ω: when applying the Clarke's transform to a set of three-phase signals, a rotating vector in the reference frame $\alpha\beta$ is obtained with a frequency equal to that of the three-phase variables. Exploiting the Park's transform, the dq frame angular speed can be set equal to that of the rotating vector obtained by means of the Clarke's transform: as a result, by doing so in this reference frame the sinusoidal signals are seen as constant. This is crucial when deploying a PI regulator, since it is able to provide a null steady-state error only with constant reference signals. By exploiting these two transforms, this aspect can be achieved.

To summarize, the Clarke's and Park's transform are implemented in the three-phase inverter control system since they lead to a null steady-state error and to a reduction of the needed PI regulators: basically, considering moving averages quantities, the direct transforms are applied to the three-phase currents obtaining the equivalent $I_d(t)$ and $I_q(t)$ values. These are the inputs of two PI regulators, which compare these values with the corresponding reference and compute the $V^*_d(t)$ and $V^*_q(t)$ two-phase equivalent voltages corresponding to $V^*_a(t)$, $V^*_b(t)$ and $V^*_c(t)$, which are obtained by applying the inverse transforms. The direct and inverse Park's transformation matrices are respectively reported in (4.15) and (4.16), where $\theta = \omega t$ is the angle between d and α reference axes. Furthermore, in the three-phase inverter case, the phase voltages fundamental frequency, and consequently that of the three-phase currents, can be set by properly generating θ.

$$\begin{bmatrix} x_d \\ x_d \end{bmatrix} = \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x_\alpha \\ x_\beta \end{bmatrix} \tag{4.15}$$

$$\begin{bmatrix} x_\alpha \\ x_\beta \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x_d \\ x_q \end{bmatrix} \tag{4.16}$$

4.3 Sine and Cosine Functions Generator Implementation and Test

In order to deploy the direct and indirect Park's transform, proper sine and cosine functions at the fundamental frequency need to be provided. For this reason, a specific subsystem is implemented in Simulink, whose block scheme is reported in Figure 4.4.

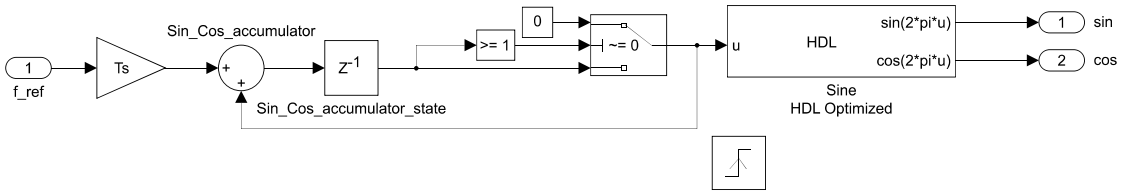


Figure 4.4: Sine and cosine generator block scheme.

Only in this preliminary test phase it is deployed as a triggered subsystem: when the whole control system is implemented, it is inserted as a simple subsystem inside the current controller. Anyway, to emulate the correct operation, its execution will be triggered by the controller execution signal produced by the PWM modulator during the experimental test. Referring to the figure, the Sine HDL Optimized block produces sine and cosine waveforms in a programmable fraction length fixed point representation, as functions of the input angle normalized with respect to 2π.

In order to do so, the wanted fundamental frequency is given as input to the subsystem and is multiplied by the execution period, which is computed in (2.4), and is added to an accumulator. The Sine HDL Optimized block input is the accumulator state, which is set at 0 every time it is greater than or equal to 1, to avoid overflows situations. In an ideal case, after a given number of execution steps which depends on the input fundamental frequency and on the execution period, the accumulator state becomes exactly equal to 1; anyway, due to both numerical precision errors and dependence on input frequency and execution period values, this is not generally the case. This leads, at the end of every sine and cosine waveforms periods, to a distortion which can be neglected as long as the input frequency is much lower than the subsystem execution one, besides the impossibility of precisely generate the wanted fundamental frequency.

In Figure 4.5, the top-level view of the sine and cosine generator block scheme is shown to highlight the presence of bias and gain blocks to adapt the [-1 - 1] range of the two generated waveforms to the [0 - 4095] acceptable by the DAC, in order to visualize them on the oscilloscope in the range [0 - 5]V during the experimental phase.

The Simulink HDL Coder tool is then exploited to generate the corresponding VHDL code which is imported in Vivado. The block design represented in Figure 4.6 is implemented to experimentally test the sine and cosine generator correct operation: in particular, the Clocking Wizard IP produces the 80 MHz system clock frequency, whereas the PWM modulator provides the sine and cosine generator with the appropriate trigger signal. The DAC operation is triggered by the control voltage update signal generated by the PWM modulator. Finally, the VIO IP produces the system reset signal and the fundamental frequency value. The generated waveforms are then visualized on the oscilloscope: three different tests have been performed, namely by setting a 50 Hz, 60 Hz and

a 100 Hz reference frequency. The experiment result is reported in Figure 4.7. As can be seen, the wanted waveforms are correctly generated.

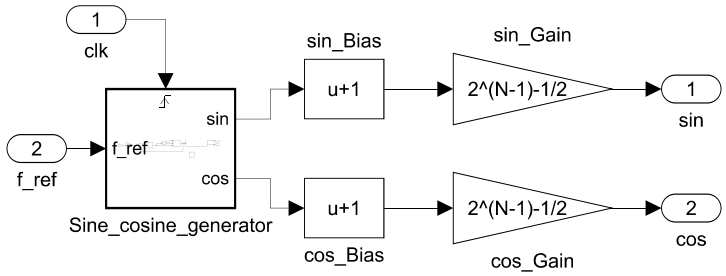


Figure 4.5: Sine and cosine generator block scheme top-level view.

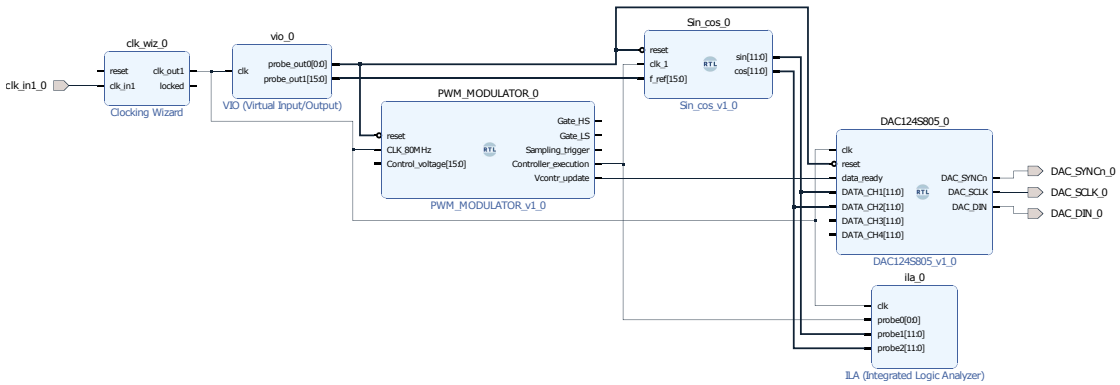
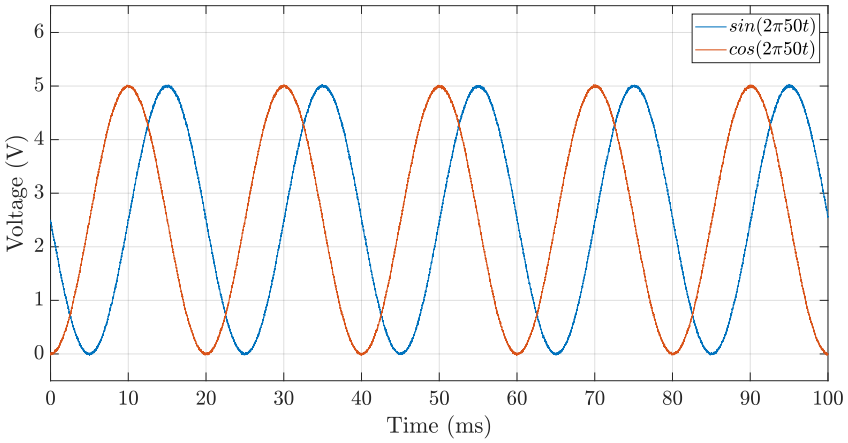


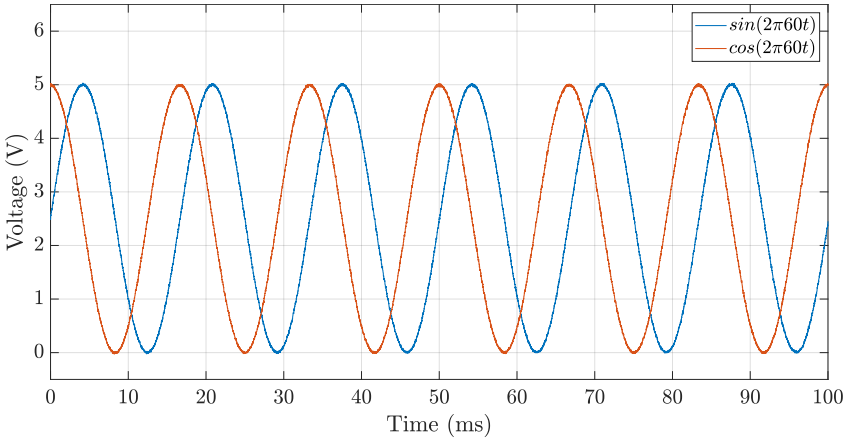
Figure 4.6: Sine and cosine generator block design in Vivado.

4.4 Clarke's and Park's Transforms Implementation and Test

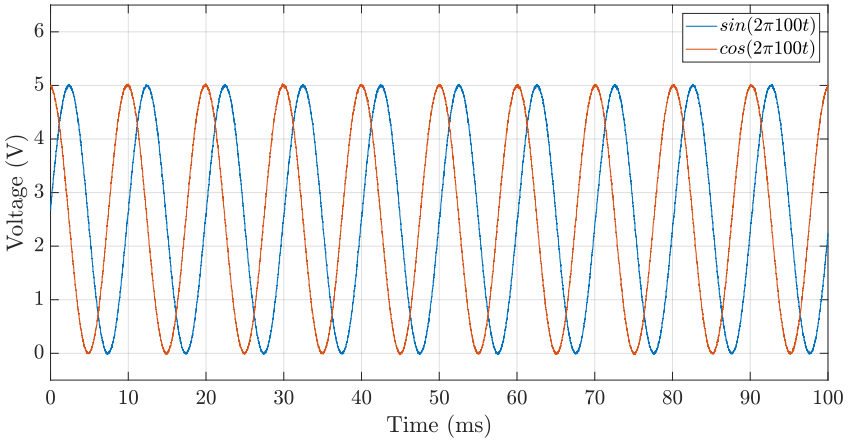
Prior to the insertion of the Clarke's and Park's transforms inside the three-phase inverter average current controller, their correct operation is checked. In order to do so, first the inverse transforms are implemented: a triggered subsystem, composed of the inverse transforms implementations and the sine and cosine generator subsystem, is deployed in Simulink, as depicted in Figure 4.8. In particular, the sine and cosine generator block scheme is that reported in Figure 4.4, whereas the two subsystems which implement the Clarke's and Park's inverse transforms, recalling expressions (4.14) and (4.16) respectively, are depicted in Figure 4.9 and Figure 4.10.



(a)



(b)



(c)

Figure 4.7: Sine and cosine generator oscilloscope waveforms with reference frequency set to (a) 50 Hz, (b) 60 Hz and 100 Hz (c).

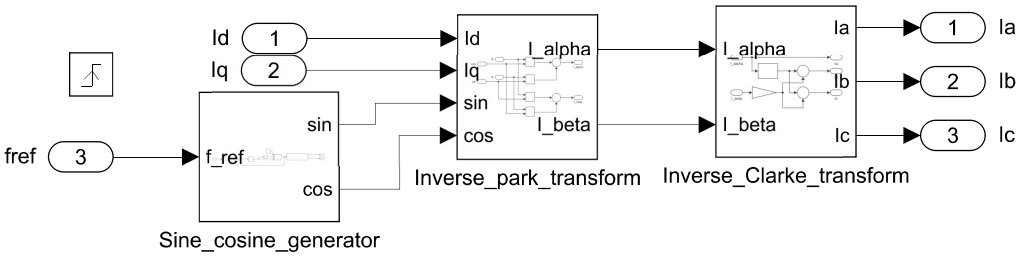


Figure 4.8: Inverse Clarke's and Park's transforms block scheme in Simulink.

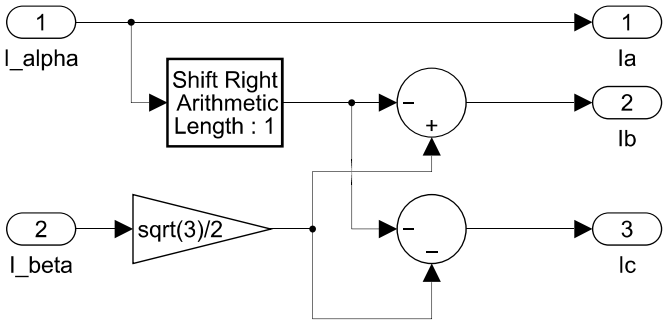


Figure 4.9: Inverse Clarke's transform block scheme in Simulink.

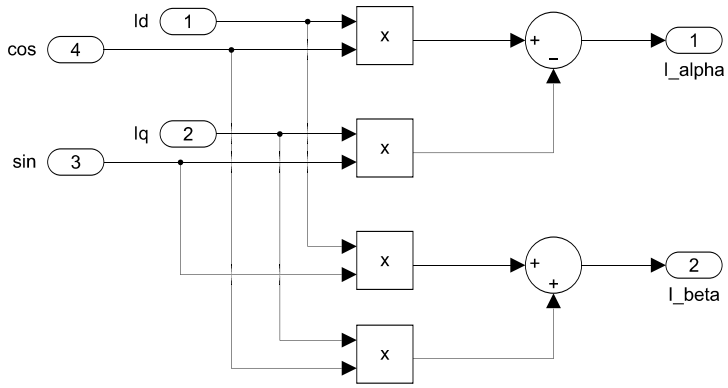


Figure 4.10: Inverse Park's transform block scheme in Simulink.

The top-level view is reported in Figure 4.11 to highlight the presence of blocks needed to correctly plot the generated waveforms on the oscilloscope through the DAC: in particular, the expected outcome is to obtain three-phase sinusoidal waveforms at the given reference frequency whose amplitude depends on the dq-frame reference currents values, provided in single floating point precision. If I_q is kept null, the amplitude of the three waveforms needs to be equal to the I_d value, as can be derived by recalling (4.14) and (4.16).

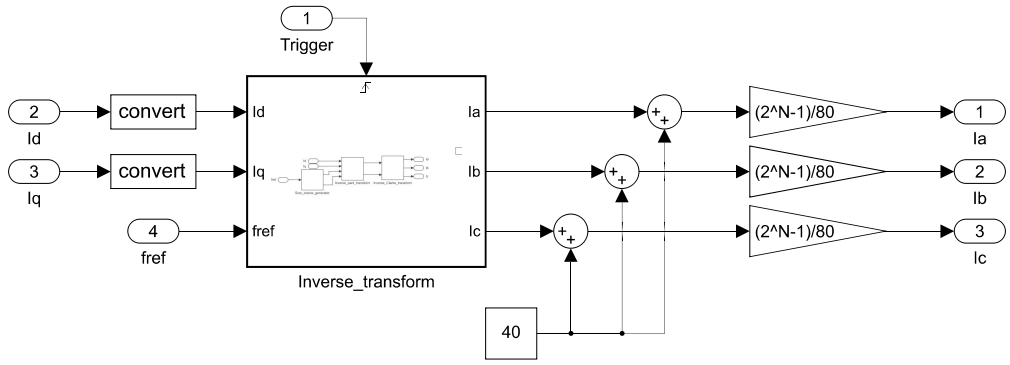


Figure 4.11: Inverse transforms top-level view block scheme in Simulink.

The VHDL code is then automatically generated and imported in Vivado to deploy the block design depicted in Figure 4.12: as shown, the PWM modulator block is exploited to produce the controller execution trigger signal to correctly generate the sine and cosine waveforms. The dq-frame reference current and the reference frequency values are provided by the VIO IP. An example of experimentally obtained waveforms is reported in Figure 4.13, where the reference frequency, I_q and I_d values are respectively set to 50 Hz, 0 A and 10 A. In particular, the oscilloscope data has been converted accordingly in order to show the measurements in terms of current instead of voltage. As shown, the expected behavior is obtained.

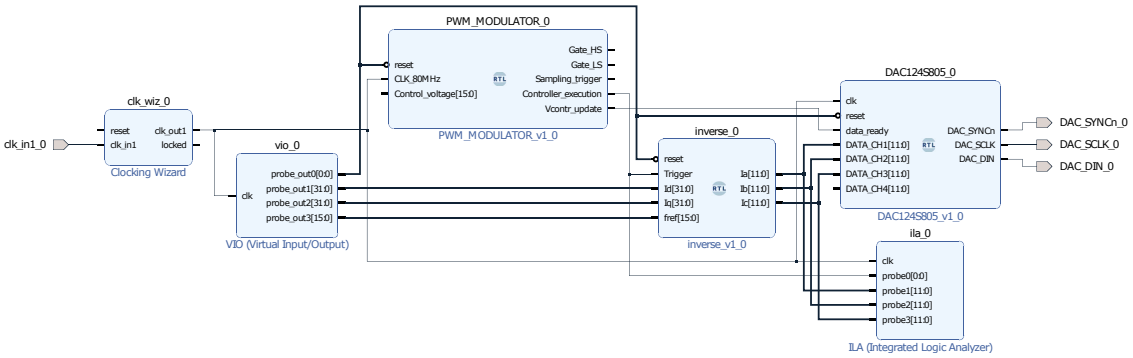


Figure 4.12: Inverse Clarke's and Park's transforms block design in Vivado.

The direct Clarke's and Park's transforms correct operation is validated by exploiting the inverse transforms previously designed, as shown by the block scheme reported in Figure 4.14: the idea is to provide specific reference values at the input and obtain the same values at the output. The implementation of the direct Clarke's and Park's transforms is based on (4.13) and (4.15), as shown in Figure 4.15 and Figure 4.16 respectively. As before, a triggered subsystem is exploited, as highlighted in Figure 4.17, to properly generate the sine and cosine functions depending on the input reference frequency.

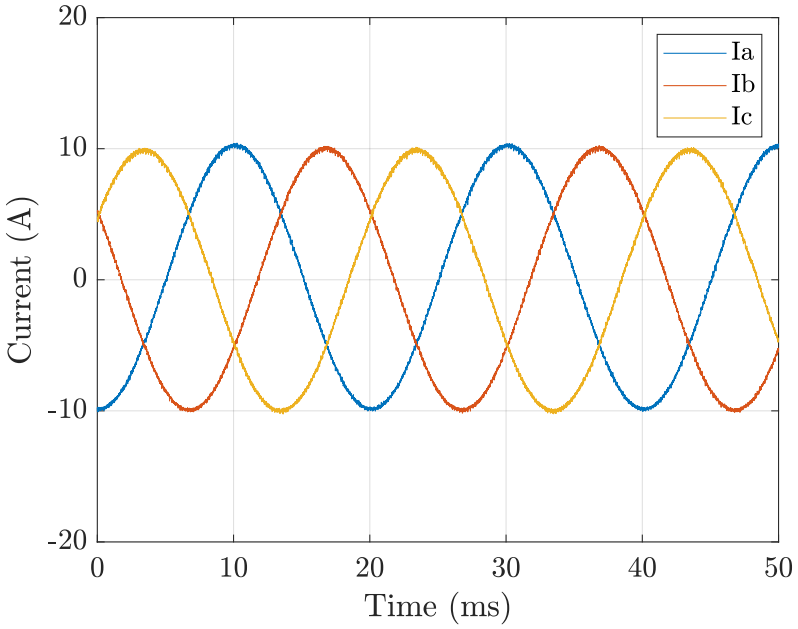


Figure 4.13: Inverse Clarke's and Park's transforms oscilloscope waveforms.

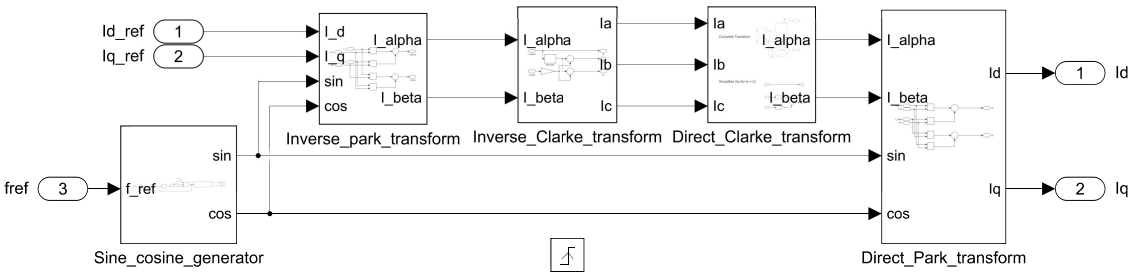


Figure 4.14: Inverse and direct Clarke's and Park's transforms block scheme in Simulink.

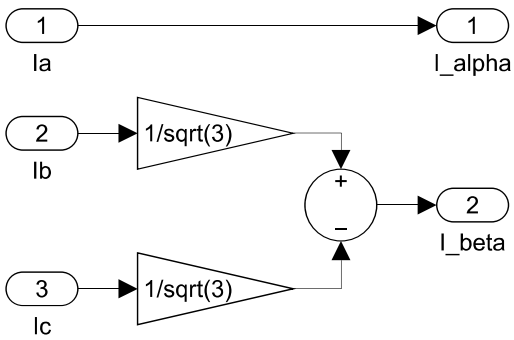


Figure 4.15: Direct Clarke's transform block scheme in Simulink.

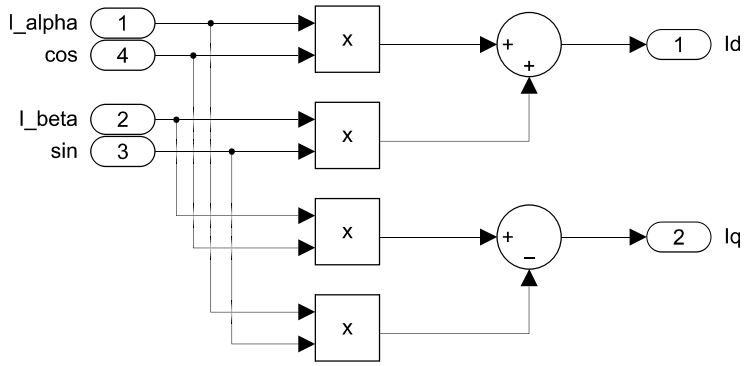


Figure 4.16: Direct Park's transform block scheme in Simulink.

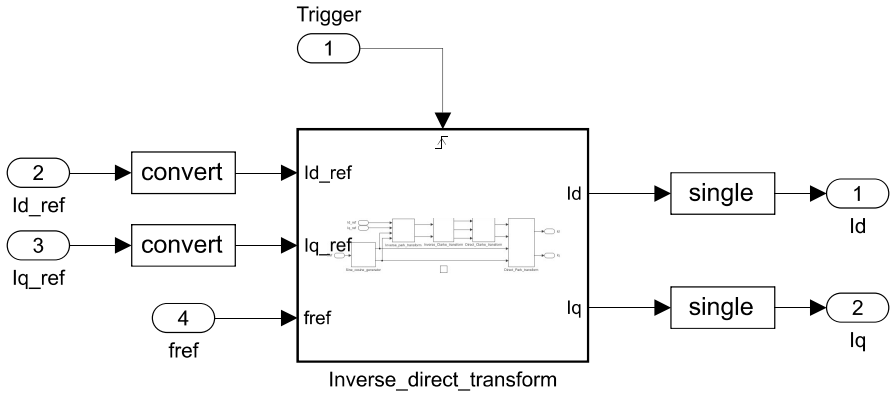


Figure 4.17: Inverse and direct Clarke's and Park's transforms top-level view block scheme in Simulink.

The automatically generated VHDL code is then imported in Vivado to deploy the block design depicted in Figure 4.18. As can be seen, the Clocking Wizard IP provides the 80 MHz frequency system clock, the PWM modulator block generates the controller execution signal to trigger the transforms operation, whereas the VIO IP is exploited to provide the dq-frame reference signals in single floating point precision and the reference frequency values. An example of the obtained values visualized on the ILA is reported in Figure 4.19, where the Controller_execution signal, the dq-frame reference values and the Id and Iq values at the output of the block are depicted in this order, exploiting the hexadecimal representation. In particular, fref, Iq_ref and Id_ref are respectively set to 50 Hz, 0 A and 5 A. As can be noticed, the obtained values are identical to the corresponding reference, proving the direct transforms implementation correct operation.

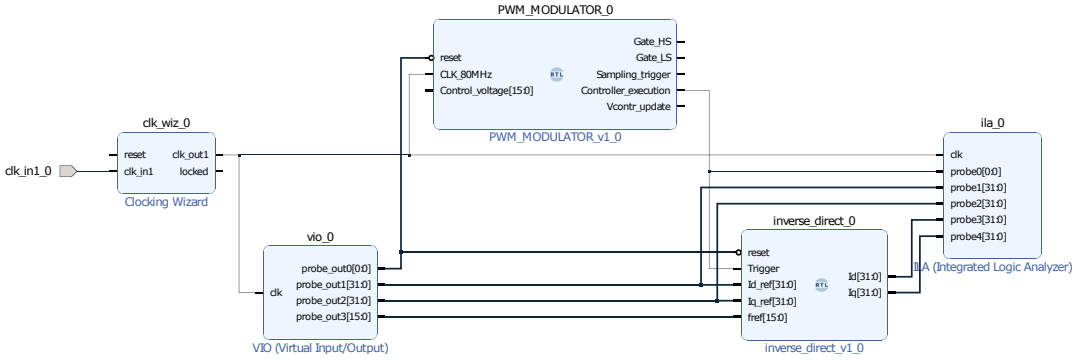


Figure 4.18: Inverse and direct Clarke's and Park's transforms block design in Vivado.

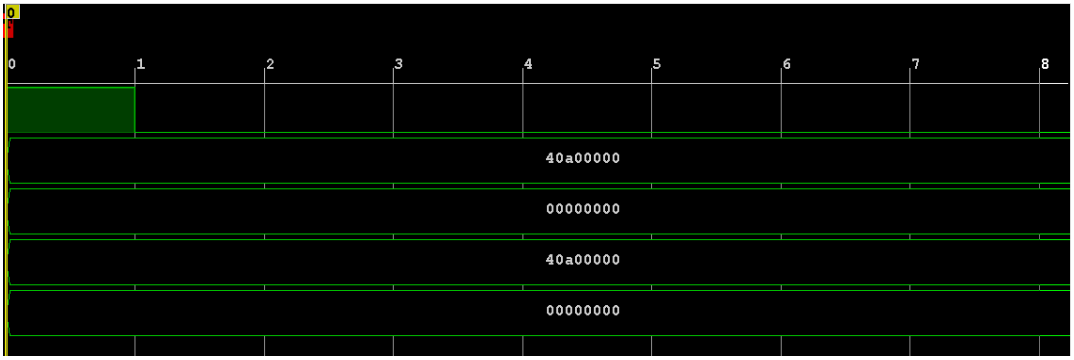


Figure 4.19: Inverse and direct Clarke's and Park's transforms ILA values.

4.5 Implemented PWM Modulation Technique

As explained in [1][4] many PWM techniques are possible to modulate a DC/AC converter. In the three-wire three-phase inverter case, the performance can be increased, with respect to the conventional techniques, by injecting particular zero-sequence signals in the control voltages. One of these methods, which increases the linear region without compromising the obtained output spectrum, is the min-max injection one. To explain its operation, it is useful to notice that each line-to-ground voltage is equal to the corresponding phase voltage. Consequently, the output of each PI regulator is the phase voltage which needs to be imposed to the load in order to obtain the wanted behavior.

This technique consists of modifying the control voltages exploited by each comparator inside the PWM modulator to produce the transistors switching functions, by simply adding to each PI regulator output the average value between the minimum and the maximum values among the three phase voltages, with negative sign, as defined in (4.17), where $V_a^*(t)$, $V_b^*(t)$ and $V_c^*(t)$ indicate the outputs of the three PI regulators, hence the phase voltages moving averages values which need to be applied to the load.

$$V_{N0}(t) = -\frac{\min[V_a^*(t), V_b^*(t), V_c^*(t)] + \max[V_a^*(t), V_b^*(t), V_c^*(t)]}{2} \tag{4.17}$$

This leads to a modification of the control voltages expressions which are reported in (4.6), as shown in (4.18).

$$\begin{cases} v_{contr,A}(t) = (\frac{V_a^*(t)+V_{N0}(t)}{V_{in}} + \frac{1}{2})V_{TR} \\ v_{contr,B}(t) = (\frac{V_b^*(t)+V_{N0}(t)}{V_{in}} + \frac{1}{2})V_{TR} \\ v_{contr,C}(t) = (\frac{V_c^*(t)+V_{N0}(t)}{V_{in}} + \frac{1}{2})V_{TR} \end{cases} \tag{4.18}$$

Considering the control system implementation without Clarke's and Park's transforms, the outputs of the three PI regulators are the corresponding phase voltages, as aforementioned. Similarly, by applying the direct Clarke's and Park's transforms, an equivalent two-phase system is obtained and only two PI regulators are needed: consequently, their outputs are the wanted phase voltages average values in the equivalent two-phase system described in the dq-reference frame. The wanted values can be then obtained by applying the inverse transforms; anyway, the division by V_{in} can be also performed in the dq-reference frame, and the inverse transforms can be applied after these operations. From a mathematical point of view this is equivalent to the previous description. The implemented current controller block scheme is summarized in Figure 4.20.

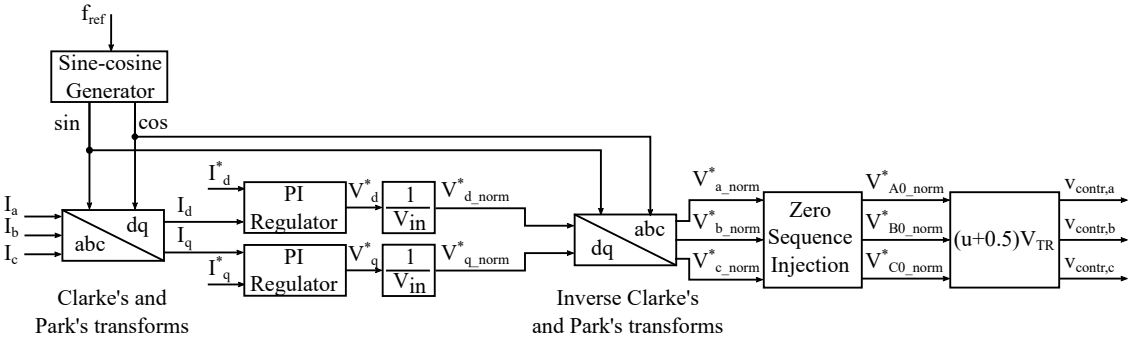


Figure 4.20: Three-phase inverter current controller block scheme.

4.6 Three-Phase Inverter Current Control System Implementation and Test

In this section, the three-phase inverter current control system is implemented. The whole block scheme deployed in Simulink is depicted in Figure 4.21. As can be seen, the same components used for the H-bridge converter control system, reported in Figure 2.19, are present also in this case, appropriately adapted for the three-phase inverter

converter. In particular, the averager block is extended to compute the two additional currents moving averages, as depicted in Figure 4.22. Moreover, the comparator block depicted in Figure 2.8 is replicated three times inside the PWM modulator, one for each corresponding phase, as shown in Figure 4.23.

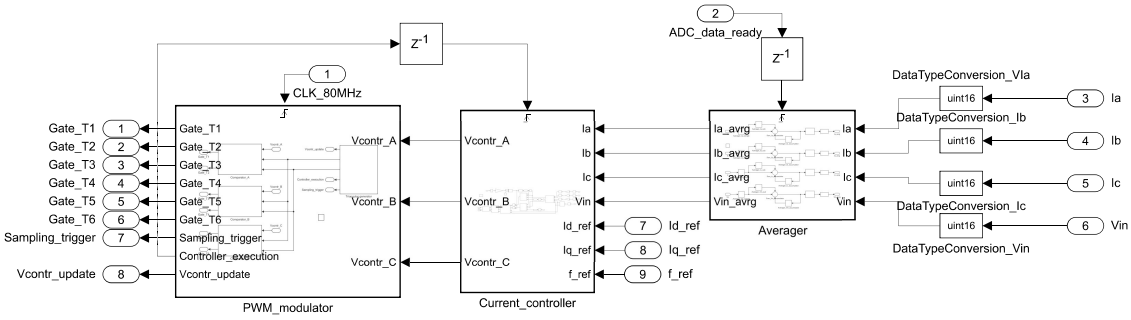


Figure 4.21: Three-phase inverter control system block scheme.

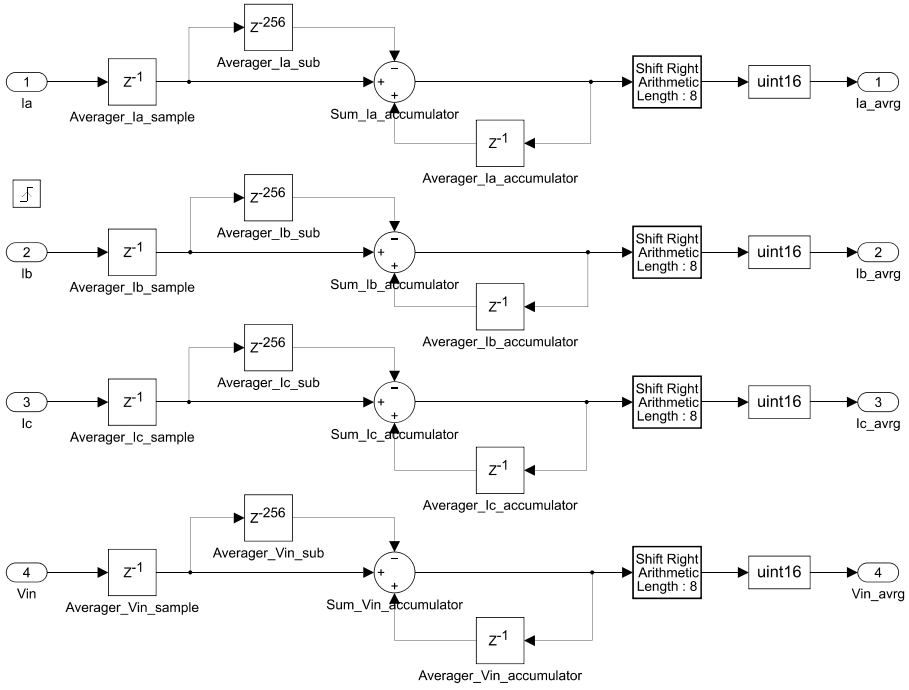


Figure 4.22: Three-phase inverter averager block scheme.

For what concerns the current controller, its block scheme is represented in Figure 4.24. As can be seen, the previously described blocks which implement the direct and indirect Clarke's and Park's transforms are present, besides the sine and cosine generator. Furthermore, the blocks which perform the division by V_{in} are placed before the inverse

transform block, as aforementioned. The zero-sequence injection block is implemented, according to the expression reported in (4.17), as shown in Figure 4.25. Finally, the correct control voltages are computed based on the expression derived in (4.18), as shown in Figure 4.26. With respect to the current controller depicted in Figure 2.17, a few additional differences are present: first, the PI regulators proportional and integral parts are limited according to expressions (4.19) and (4.20), where ϵ denotes the error between dq-frame reference values and direct transforms blocks output values and E indicates the feedforward voltage in the dq-frame.

$$-\frac{V_{in}}{2} \leq \epsilon K_P + E \leq \frac{V_{in}}{2} \tag{4.19}$$

$$\begin{cases} Int\_max = \frac{V_{in}}{2} - |(\epsilon K_P + E)| \\ Int\_min = -\frac{V_{in}}{2} + |(\epsilon K_P + E)| \end{cases} \tag{4.20}$$

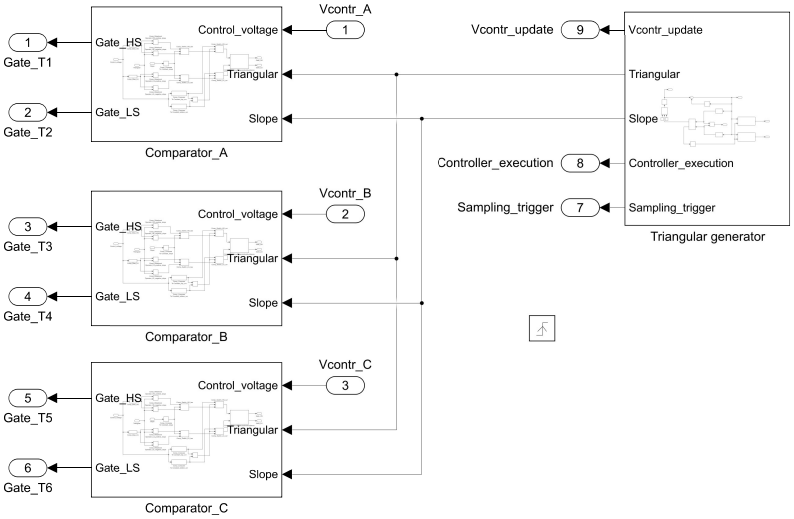


Figure 4.23: Three-phase inverter PWM modulator block scheme.

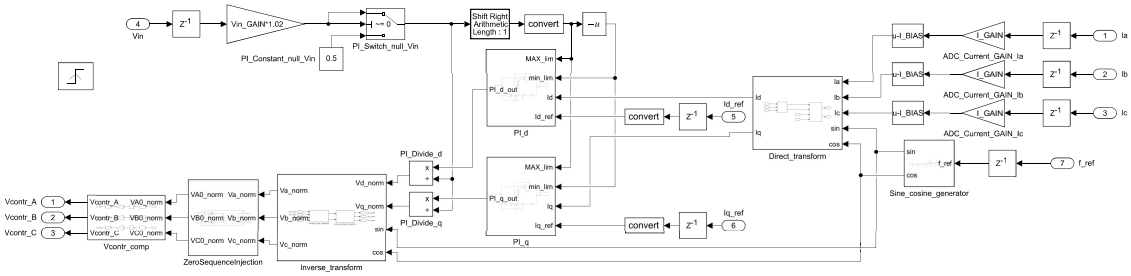


Figure 4.24: Three-phase inverter current controller block scheme in Simulink.

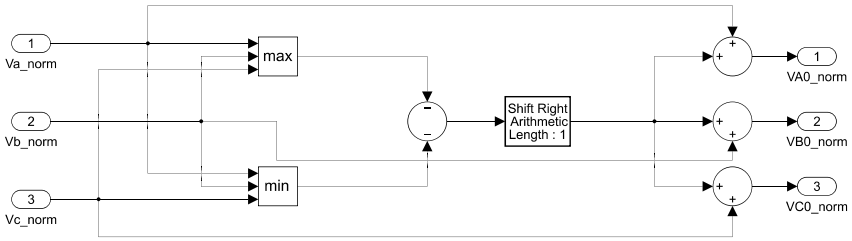


Figure 4.25: Zero-sequence injection block scheme.

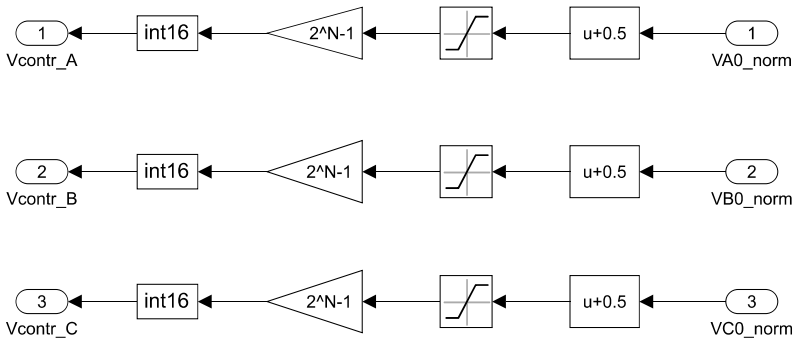


Figure 4.26: Control voltages generation block scheme.

Secondly, the correction term explained in Chapter 3 is highlighted in the input voltage gain term. Finally, the PI regulator block scheme is slightly modified, and simplified, as shown in Figure 4.27. As aforementioned, the experimental validation is carried out on an inductive load, hence without the presence of back-emf voltages. Nevertheless, since in general a feedforward technique can be implemented, the presence of the proper sum block is highlighted. The figure is referred to the implementation of the d-reference axis. Anyway, the same block scheme is deployed also for the q-axis. The design of the whole current controller is achieved by exploiting the Fixed-Point Tool, as was done in the previous chapters.

The VHDL code corresponding to the shown control system is imported in Vivado to implement the block design represented in Figure 4.28, after applying the same modifications in the code which were described in the previous chapters. As shown, the same components described in Figure 3.27 are present. In particular, the modules protection block is extended to check an overcurrent situation in all the three phases, with respect to the H-bridge converter implementation. Moreover, only transistor T7, referring to Figure 3.3, is not driven. This is not an issue, since it drives the braking leg, which is not used in this thesis. By means of the VIO IP, the system reset and the already described K1 and EN_PWM signals are generated. Furthermore, the reference frequency value and the dq-frame reference single floating point precision values are provided to the control system.

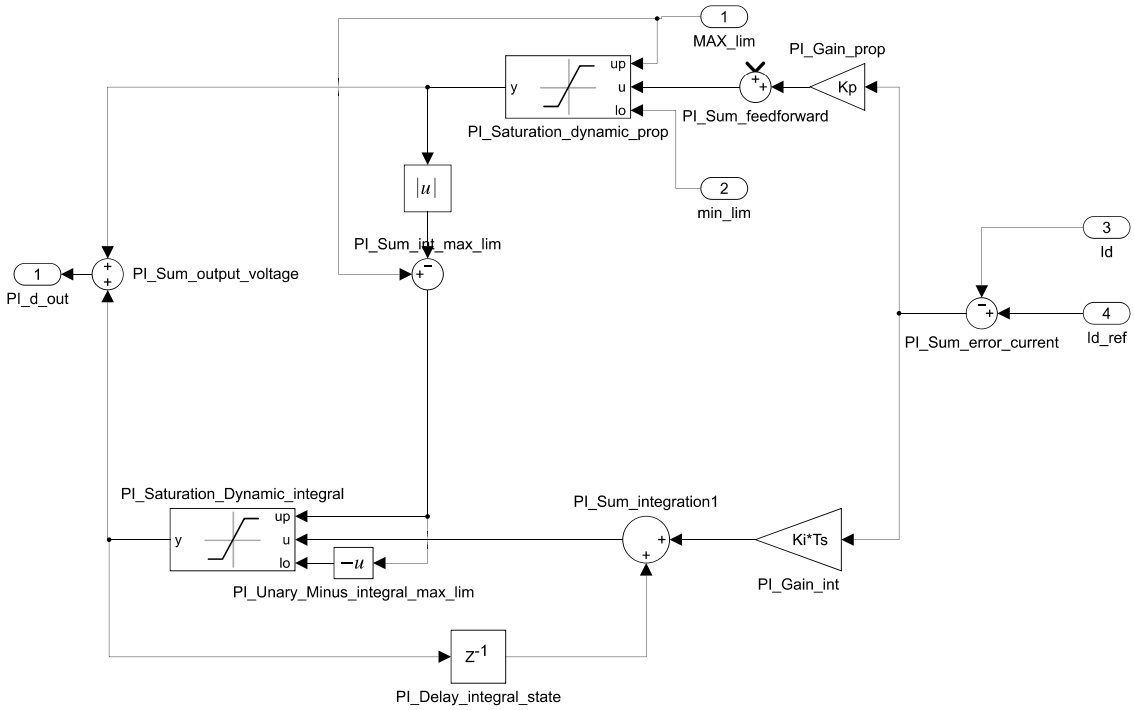


Figure 4.27: Three-phase inverter control system PI regulator block scheme.

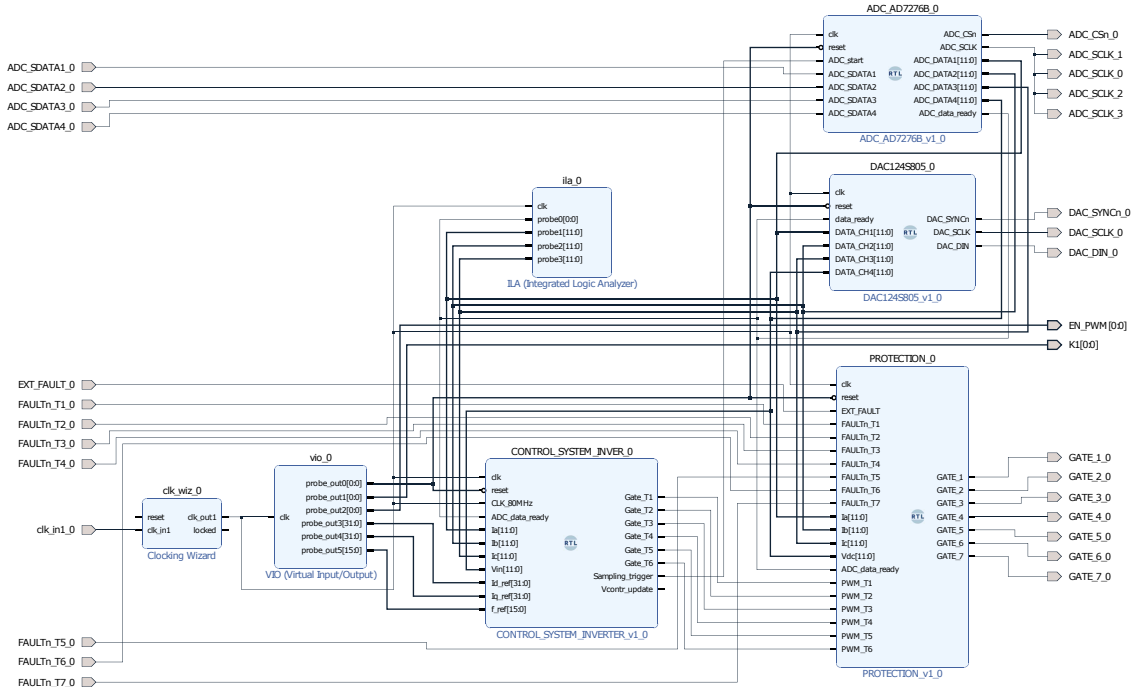


Figure 4.28: Three-phase inverter control system block design in Vivado.

Several experiments have been performed exploiting the setup depicted in Figure 4.29: this is identical to that used for the H-bridge control system validation, reported in Figure 3.26. In this case, a star connected inductive load is exploited: each phase inductance value is equal to 2 mH. For what concerns the design of the K_P and K_I parameters of each PI regulator, they are referred to this inductance value, considering a bandwidth frequency equal to 500 Hz, according to the expressions reported in (1.40).

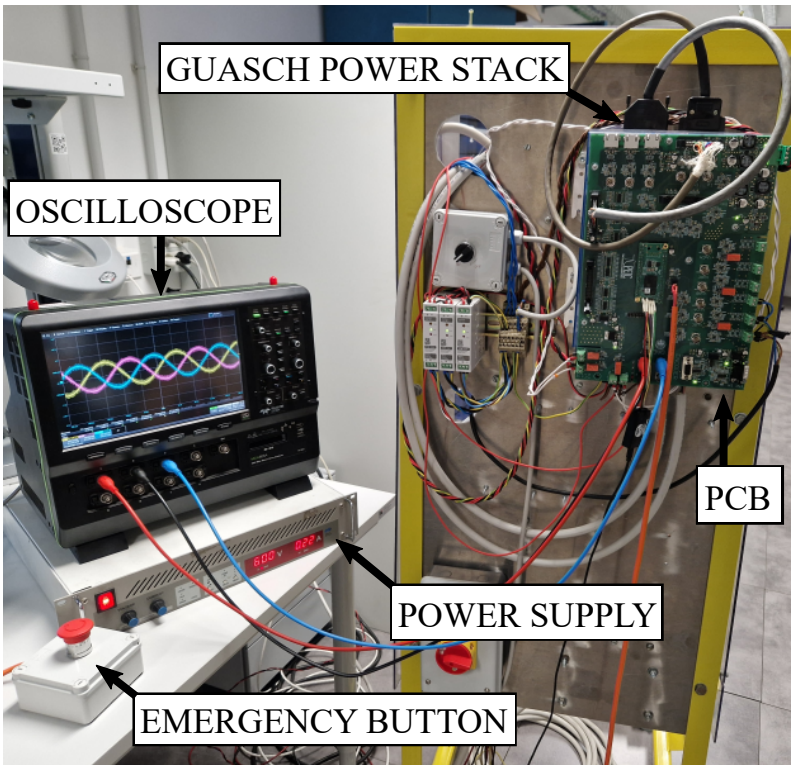
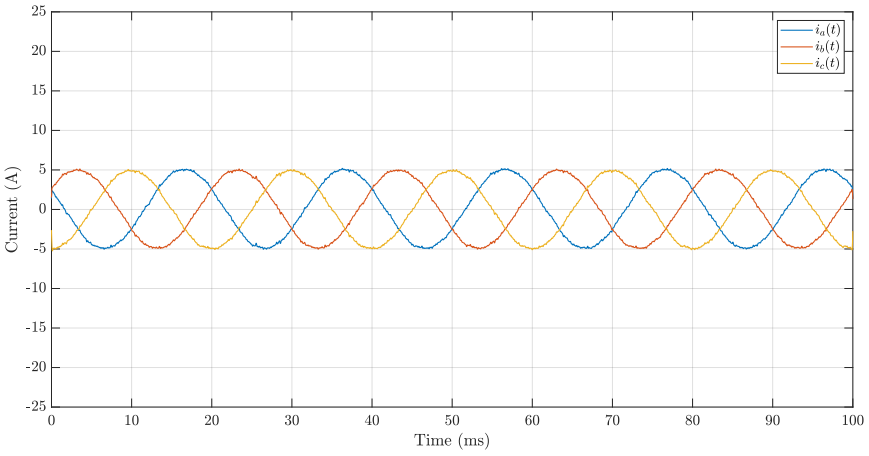
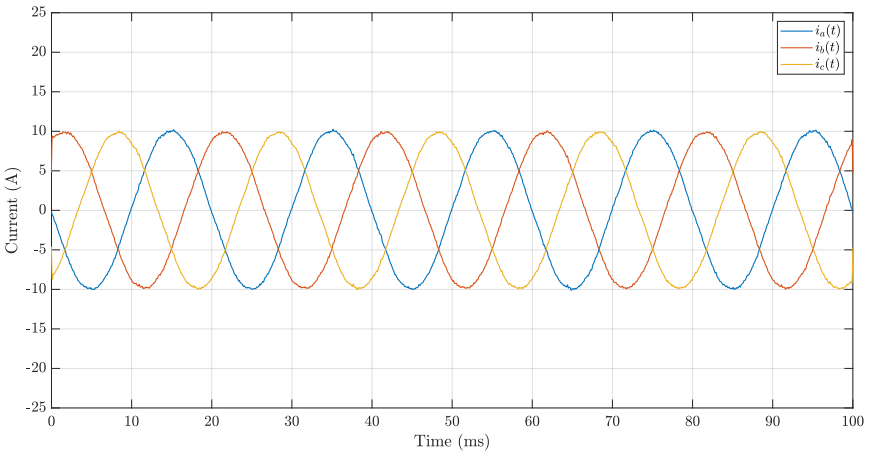


Figure 4.29: Three-phase inverter experimental setup.

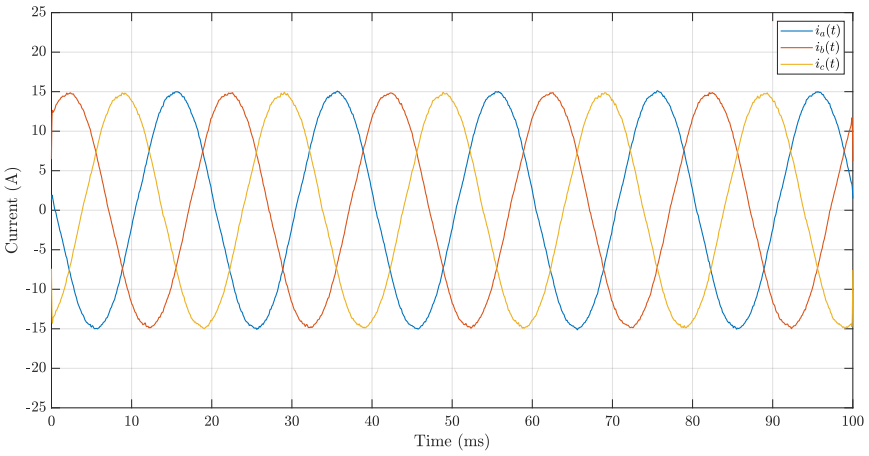
An example of the obtained waveforms, which are low-pass filtered to eliminate the overshoots caused by the converter switching operation, is depicted in Figure 4.30, where a 50 Hz reference frequency and a 0 A reference I_q value are imposed, whereas the reference I_d is set to 5 A, 10 A and 15 A. As can be seen, the control system correct operation is validated.



(a)



(b)



(c)

Figure 4.30: Three-phase inverter oscilloscope waveforms with a 50 Hz fundamental frequency, 0 A reference q-axis current and (a) 5 A, (b) 10 A and (c) 15 A reference d-axis current values.

Chapter 5

Implementation of a User Interface on a dSPACE Rapid Prototyping System

In the previous chapters the designed control systems have been validated by providing the necessary commands and reference values by means of the VIO IP inside Vivado, while simultaneously monitoring the wanted quantities on the oscilloscope by exploiting the DAC or by means of the ILA IP. This presents a huge restriction, since up to four quantities can be visualized on the oscilloscope through the DAC and the ILA IP is limited in terms of data samples which can be analyzed. Moreover, it can be convenient to design a visual interface to ease the generation of the various commands and references for the control system execution, which can be provided directly in the wanted format avoiding the data type conversion blocks highlighted in the previous chapters. For these reasons, in this chapter the dSPACE MicroLabBox prototyping system [16], which can be programmed through Simulink to execute specific tasks, is inserted in the developed three-phase inverter control system. In particular, several SPI communications are designed and deployed as will be explained in the following sections in order to exchange all the needed data between the FPGA module and the dSPACE system in both directions. Consequently, the digital interface with dSPACE connector highlighted in Figure 3.1 is exploited to implement the communication. Finally, visual interfaces are used to both provide the control system with the main commands and reference signals and to visualize the system main quantities.

5.1 SPI Protocol Design

In this section, the implemented SPI interface between the dSPACE system and the FPGA module is presented. Specifically, due to the dSPACE MicroLabBox specifications, up to four different SPI communication units can be simultaneously deployed: the dSPACE system units always act as masters whereas proper slaves have to be designed

in the implemented control system to manage the exchange of data. The same SPI protocol specifications, which can be defined by programming the dSPACE system through Simulink by deploying the appropriate blocks, are used for all the four units. In particular, the following main settings are chosen, considering an active low chip select signal:

- SPI clock (SCLK) signal: 2 MHz;
- clock polarity: low;
- clock phase: trailing;
- number of words: 2;
- bits per word: 32.

Basically, after the chip select is driven low, each master unit and each slave completes a 64-bit transmission, sampling the data line at each SCLK falling-edge. This is done by appropriately designing each slave by deploying two shift registers. In particular, each slave:

- when the chip select signal is driven low, loads the two 32-bit words to be transmitted in the first shift register;
- at every SCLK signal rising-edge, loads the MISO line with the first shift register most significant bit and shifts the content of the second shift register;
- at every SCLK signal falling-edge, loads the MOSI line bit to the second shift register least significant bit and shifts the content of the first shift register;
- when the chip select signal is driven high, outputs the second shift register state, hence the received two 32-bit words;

The transmissions are periodically triggered every 50 µs. It is important to highlight that the transmitted and received words are requested to be 32-bit unsigned integers by the deployed SPI communication blocks. For this reason, appropriate conversions need to be introduced in order to both correctly interpret the received data and properly provide the various references.

In this chapter, only three SPI master units are needed; the fourth one is anyway deployed since it will be exploited in the following chapter. In particular, on the first SPI channel, the master receives the outputs of the averager block depicted in Figure 4.22, hence the moving average values of the three-phase currents and of the input voltage source. A proper conversion needs to be introduced, considering the already described I_BIAS, I_GAIN, Vin_GAIN bias and gain terms, considering also the correction term, to correctly interpret them. Furthermore, it transmits the bounds used in the modules protection block to detect overcurrent and overvoltage situations. For this reason, starting

from single precision floating point numbers inserted by the user, similar gain and bias terms need to be introduced to apply the conversion into the corresponding 12-bit data, used by the modules protection block.

On the second SPI channel, the master receives the dq-frame currents computed by the current controller. Consequently, the received data is divided by a factor as a function of the fixed point representation used for these two quantities in the current controller. At the same time, it transmits the dq-frame reference currents values inserted by the user in floating point single precision after converting them in the proper fixed-point representation.

Finally, on the third SPI channel, the master receives the Guasch power stack temperature measurement, obtained by exploiting the NTC temperature sensor, and the FAULT signals values, besides some additional outputs from the modules protection block which indicate if a protection situation is detected and what caused it, namely the external emergency button, an overcurrent or overvoltage situation based on the ADC samples, or a FAULT signal which is driven low. The received 12-bit data corresponding to the temperature measurement needs to be properly converted by reversing the expression taken from [12] and reported for clarity in (5.1), where $R_{25} = 5\,\text{k}\Omega$, $B = 3375$ K and considering that the R_T measurement can be derived by considering that the ADC conditioning circuit designed on the board corresponding to the NTC sensor is depicted in Figure 5.1. The received 12-bit data corresponds to the $V(R_T)$ value. On the other side, the slave receives the reference frequency 16-bit unsigned integer value inserted by the user for the sine and cosine generator and the K1, EN_PWM and reset signals.

$$R_T = R_{25} e^{B\left(\frac{1}{T[K]} - \frac{1}{298.15K}\right)} \tag{5.1}$$

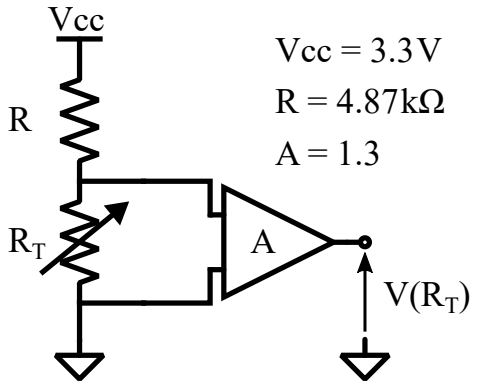


Figure 5.1: NTC temperature sensor ADC conditioning circuit.

5.2 FPGA Implementation and Experimental Validation

In this section, the correct dSPACE MicroLabBox system deployment is validated. In order to do this, an appropriate VHDL code is written to implement a block which instantiates one slave for each SPI channel, composes the respective words to be transmitted by each one of them and unpacks the received data into the corresponding quantities. The three-phase inverter control system depicted in Figure 4.21 is deployed again in this chapter, even though a few modifications need to be introduced: first, by looking at Figure 4.24, the data type conversion blocks for the dq-frame reference currents are deleted. Secondly, the dq-frame current values computed by the current controller and the three-phase currents and the input voltage average values computed by the averager block are added to the control system outputs. Notice that the automatically generated VHDL code needs to be modified accordingly also in this case, as was done in the previous chapters. Finally, referring to Figure 4.28, the modules protection block is adapted to receive as inputs the bounds for the overcurrent and overvoltage detection, besides adding the protection signals to the outputs as aforementioned.

The block design which is implemented in Vivado is depicted in Figure 5.2, where the previously described modifications with respect to that reported in Figure 4.28 can be noticed.

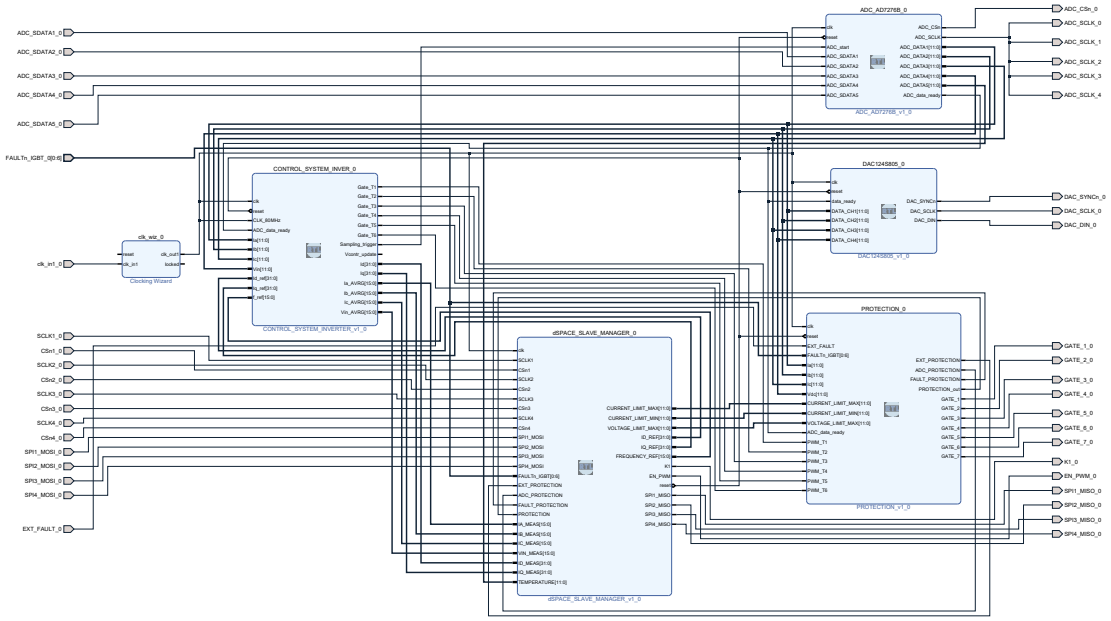


Figure 5.2: Three-phase inverter control system with dSPACE system deployment block design in Vivado.

For what concerns the experimental validation, the setup depicted in Figure 5.3 is exploited. As shown, it is identical to that depicted in Figure 4.29, apart from the presence

of the dSPACE MicroLabBox system which is connected to the board through the apposite connector, highlighted in Figure 3.1.

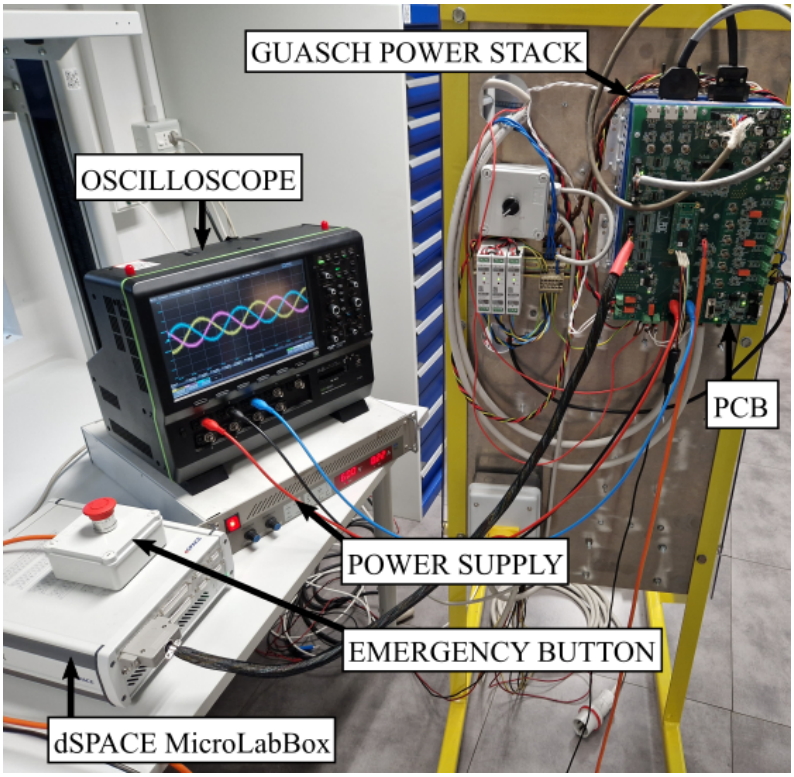


Figure 5.3: Three-phase inverter control system with dSPACE system deployment experimental setup.

The commands and reference signals are provided from the dSPACE system to the FPGA module by means of the visual interface shown in Figure 5.4. As highighted, the reset, K1, EN_PWM signals can be easily provided by the user besides the modules protection block current and voltage bounds and the reference frequency and dq-frame currents values for the current controller execution. Moreover, the temperature measured by the NTC sensor, the outputs of the modules protection block and the FAULT signals are shown. The meaning of the modules protection block outputs is here summarized:

- PROTECTION: indicates if a protection situation has occurred, disabling the gate driving signals;
- FAULT_PROTECTION: indicates if a FAULT signal is driven low;
- EXT_PROTECTION: shows if the external emergency button is pushed;
- ADC_PROTECTION: indicates if an overcurrent or an overshoot is detected.

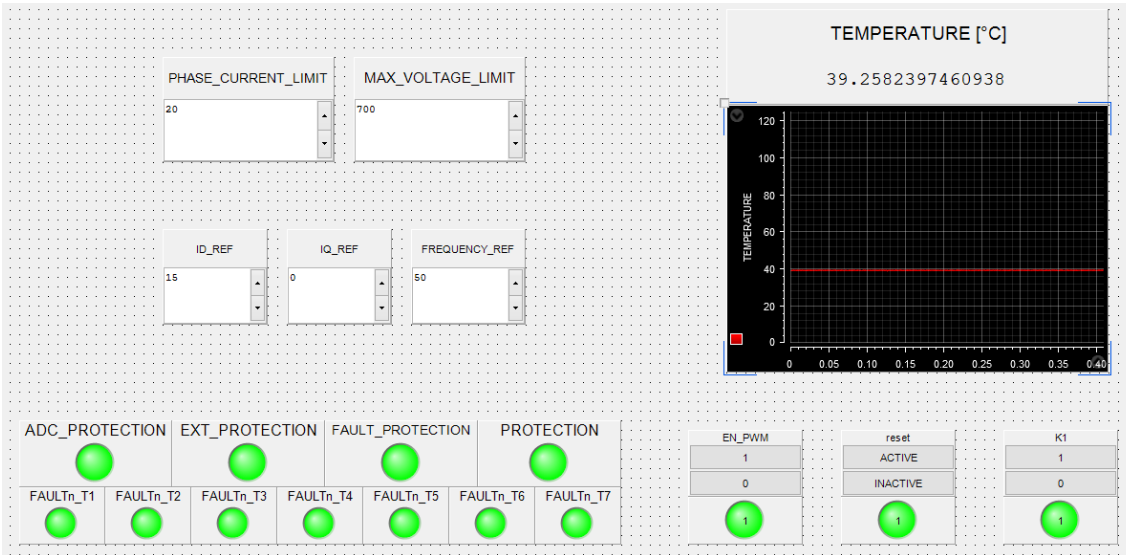


Figure 5.4: Commands and reference signals dSPACE system visual interface.

Furthermore, the three-phase currents, the input voltage value and the dq-frame computed currents are plotted in the interface depicted in Figure 5.5, which refers to the experiment in which the fundamental frequency and the dq-frame reference currents are respectively set to 50 Hz, 0 A and 15 A, as also underlined in Figure 5.4.

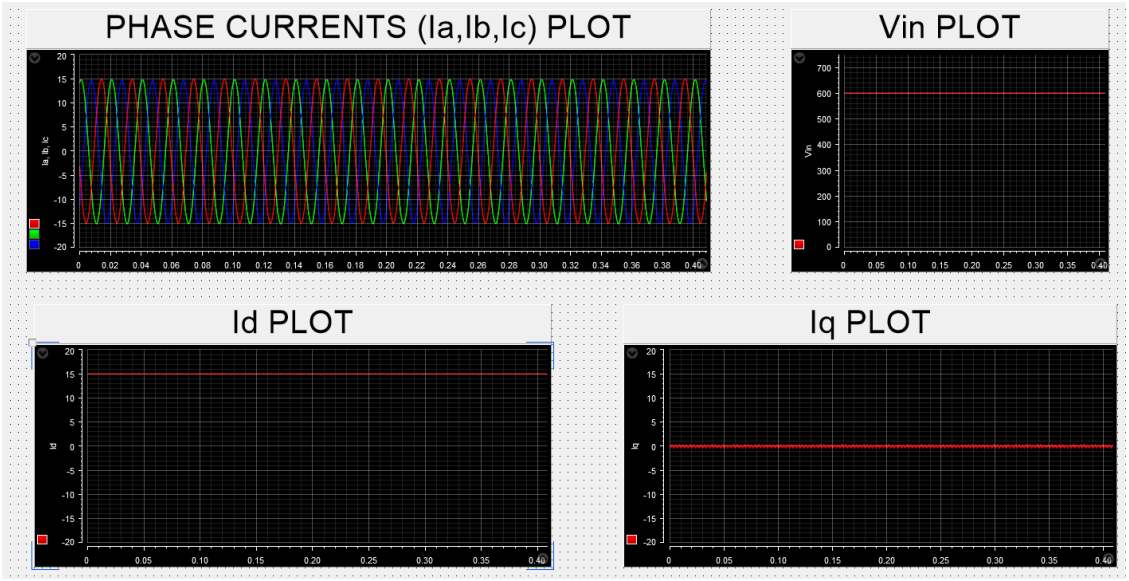


Figure 5.5: Plots dSPACE system visual interface.

Chapter 6

Electric Motor Torque Control Implementation

In this chapter, the previously designed three-phase inverter average current control system is adapted to the torque control of a permanent magnet assisted synchronous reluctance (PMASR) motor. By exploiting the dSPACE system designed in the previous chapter, a Maximum Torque Per Ampere (MTPA) look-up table is used to provide the control system implemented on the FPGA module with the dq-frame reference currents which generate the reference torque requested by the user. Furthermore, an external speed loop is deployed in order to implement also the motor speed control. After describing the main system modifications with respect to the previous chapters, presenting also an encoder, the experimental setup is described and the designed control system is validated.

6.1 System Description and Encoder Implementation

In this section, the torque control of a PMASR motor is implemented. Even though the purpose of this chapter is not that of reviewing the considered electric motor, a few characteristics are recalled in order to explain the implemented design. The main electric motor specifications which will be used in the following discussion are hereby listed:

- rated power: 2.2 kW;
- poles pairs: 2;
- rated speed: 1800 rpm;
- rated torque: 11.7 Nm.

The first test that will be performed in the following section regards the production of a wanted average torque. As explained in [17][18], the PMASR motor is capable of

producing an average torque whose expression is reported in (6.1), considering the dq-frame, where p indicates the number of poles pairs, i_d and i_q are the dq-frame currents, ϕ_m is the flux linkage of the magnets and L_d and L_q denote the inductances in the dq-frame.

$$T = \frac{3}{2} p(L_d - L_q) i_d i_q + \phi_m i_q \tag{6.1}$$

This expression is important to highlight that an average torque can be generated by applying proper dq-frame currents. For this reason, a Maximum Torque Per Ampere (MTPA) look-up table is exploited: basically, for every given wanted torque value, the corresponding dq-frame reference currents values are given. An interpolation technique is used for the values which are not included in the table.

It is important to highlight that the motor can be modeled as a star connected inductive load: for this reason, the previously designed three-phase inverter control system, depicted in Figure 4.21 and adapted for the communication with the dSPACE system, is deployed also in this chapter. In this case, the dq-frame reference currents values are provided by the MTPA look-up table, which is implemented inside the dSPACE system deployed in the previous chapter, and not directly by the user, that in turn indicates the requested average torque. Furthermore, a 500 V power supply will be used during the experimental phase.

Another important difference with respect to the previously designed three-phase inverter control system concerns the sine and cosine waveforms generator, whose block scheme is depicted in Figure 4.4. In fact, in this case the angle normalized with respect to 2π on which the Sine HDL Optimized block computes the sine and cosine values is directly computed from the electric motor operation and, consequently, the wanted frequency is not provided by the user. This is achieved by exploiting the encoder interface depicted in Figure 3.1: in fact, the PMASR motor is equipped with an appropriate encoder which can be exploited to compute the mechanical rotation angle.

Considering for simplicity a steady-state situation in which the motor speed is constant, three periodic signals are generated: two signals, defined as A and B, are two identical square waves with 90° phase shift, whose edges indicate an angle increment step $\Delta\theta$, as highlighted in Figure 6.1. A counter can be then exploited to compute the angular position of the motor by counting the edges of the A and B signals. An additional signal, named Z, presents an active low pulse every time the rotor completes a full revolution. It is important to notice that the A and B signals are sufficient to compute the motor revolutions; anyway, when the system is turned off, the electric motor can be freely rotated. Furthermore, the counter state is not memorized and it is set to 0 every time the system is activated. This basically makes it impossible to always associate the 0 counter state to the same rotor angular position. However, a crucial aspect for the designed system which will be explained in the following is to be always able to start the counting operation

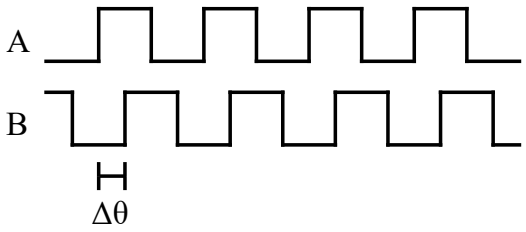


Figure 6.1: A and B signals steady-state operation example.

from the same position. This aspect can be achieved by exploiting the Z signal: in fact, when its low pulse is detected, the counter state is immediately reset to 0. This means that, apart from the first revolution, the counter state null value can be always associated to the same angular position. Notice that the steady-state condition is not necessarily requested to retrieve the angular position from the A and B signals: in fact, their edges can be counted in every condition and the angular position can be consequently computed.

A final important comment on these signals regards the number of pulses which are generated and, as a result, the $\Delta\theta$ value. Referring the exploited encoder, both A and B are trains of 1024 pulses. This means that the total number of edges of the two signals combined corresponding to a full motor revolution is 4096. Thus, a 12-bit counter is used and the obtained revolution increment step expressed in radians can be computed in (6.2).

$$\Delta\theta = \frac{2\pi}{4096} = \frac{\pi}{2048} \tag{6.2}$$

From the electrical point of view, it is crucial to notice that the electrical angle θ_{el} is obtained by multiplying the mechanical angle by the number of poles pairs [3]. In the implemented solution, this is taken into account by increasing the counter in correspondence of every A and B signals edge by the number of poles pairs instead of 1. The electrical angle is then fed to the current controller for the sine and cosine waveforms generation.

The implemented control system is summarized in Figure 6.2. The current controller implementation is that depicted in Figure 4.20, with the only exception of the sine and cosine generator block which receives θ_{el} as input instead of the reference frequency inserted by the user.

Finally, it is important to mention that the MTPA look-up table is referred to a situation in which the d-axis is aligned with the magnets flux linkage ϕ_m direction. For this reason, an offset is added to the counter output in order to associate the 0 counter state with the motor angular position corresponding to an alignment between the d-axis and the flux linkage vector.

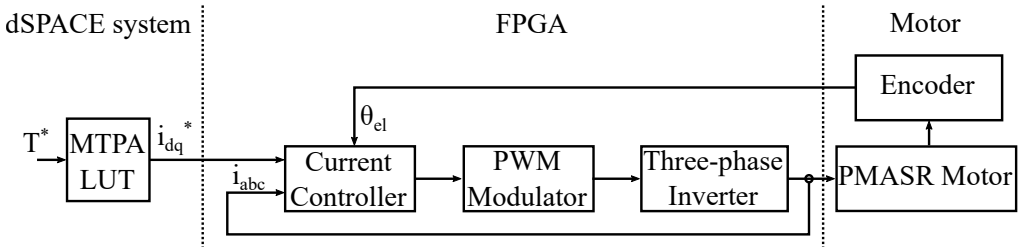


Figure 6.2: Torque control system block scheme.

6.2 Deployed Setup and Experimental Validation

In order to validate the torque control of the electric motor, the setup depicted in Figure 5.3 is accordingly modified: first, the power supply is replaced by another one which is able to provide more current and a 500 V input voltage. Secondly, the K1 signal is used to connect the PMASR motor to the U, V and W nodes highlighted in Figure 3.3 instead of the previously deployed star connected inductive load. Finally, the encoder interface underlined in Figure 3.1 is exploited to provide the A, B and Z signals to the FPGA module by linking the interface with the EXT pin header through the use of wires.

The block design exploited in Vivado is reported in Figure 6.3. As shown, it is basically identical to that depicted in Figure 5.2 even though few differences need to be commented: first, the EN_PWM signal provided from the dSPACE system is used to enable also the modules protection block modulation outputs, allowing to activate the control system without driving the transistors. By doing so, the ADCs and DAC operations and the dSPACE system communications can be activated. In particular, this is useful to retrieve the aforementioned offset value through dSPACE.

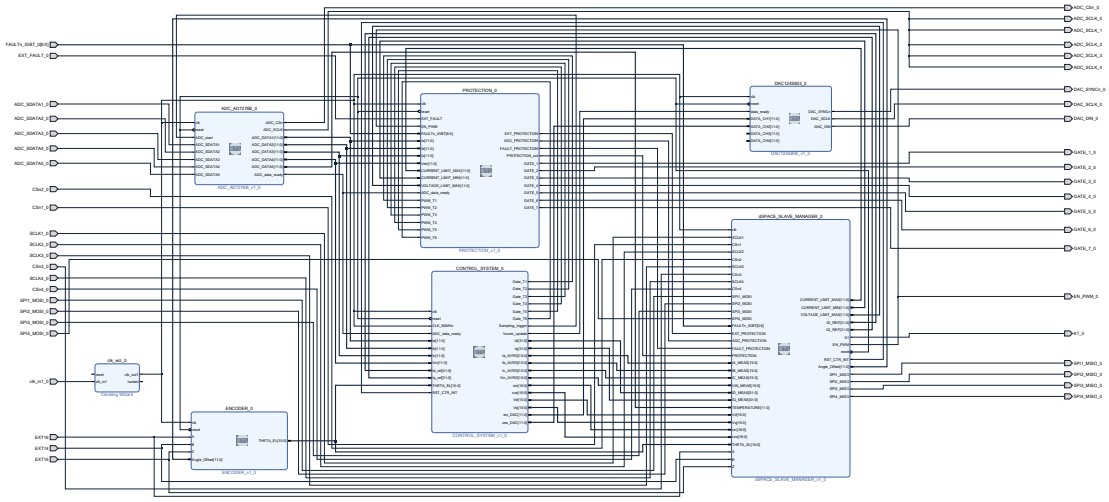


Figure 6.3: Torque control system block design in Vivado.

Secondly, due to this introduction, the integrators inside the current controller need to be provided with a dedicated reset signal which need to be produced after the modulation is enabled. This aspect is achieved by the RST_CTR_INT signal, which is provided by the dSPACE system: when it is activated, the integrators states are set to 0. Furthermore, the computed sine and cosine values and the dq-frame voltages are added to the control system outputs since they are transmitted to the dSPACE system.

Thirdly, the encoder block, which implements the previously described counter based on the A, B, Z signals, is deployed. Its output, namely the electrical angle θ_{el}, is given to the control system besides the dSPACE slaves manager.

Finally, the dSPACE slaves manager is accordingly modified: in particular, referring to the implementation presented in Chapter 5, the FPGA modules receives the RST_CTR_INT signal on the third channel and transmits the dq-frame voltages to the dSPACE system. Moreover, the fourth channel is exploited to transmit the offset value for the encoder to the FPGA, whilst the sine and cosine values, the electrical angle computed by the encoder and the A, B, Z signals are provided to the dSPACE system.

With respect to the block scheme depicted in Figure 6.2, an important additional comment regards the motor mechanical load. In the following experiments, the setup depicted in Figure 6.4 is exploited. It is composed of the PMASR motor, namely the Motor Under Test (MUT), which is connected to a Driving Machine (DM), hence another electric motor, by means of a shaft on which a torque meter is placed. The two electric motors are driven by two different three-phase inverters. The actual electric motors setup is reported in Figure 6.5.

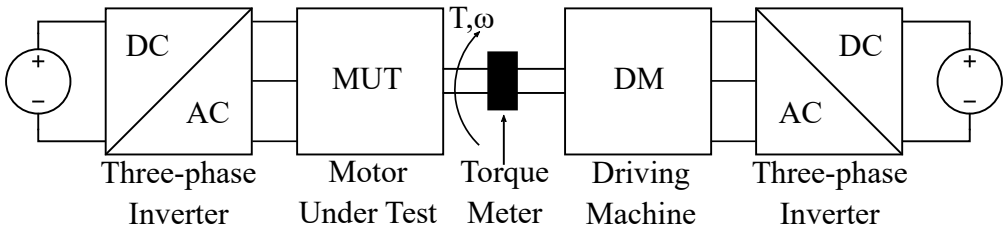


Figure 6.4: Motor Under Test and Driving Machine block scheme.

The DM can be set to impose either a certain rotational speed or a specific load torque. In a first experiment, the DM is exploited to impose a 500 rpm rotational speed. After that, the control system and the modulation are activated, the MTPA look-up table is used to provide the dq-frame reference currents values from the dSPACE system, the integral operations inside the current controller are enabled and the torque is measured by means of the torque meter. The performed test, whose result is depicted in Figure 6.6, consists of requesting a 10 Nm reference torque to the MUT and checking the correct regulation

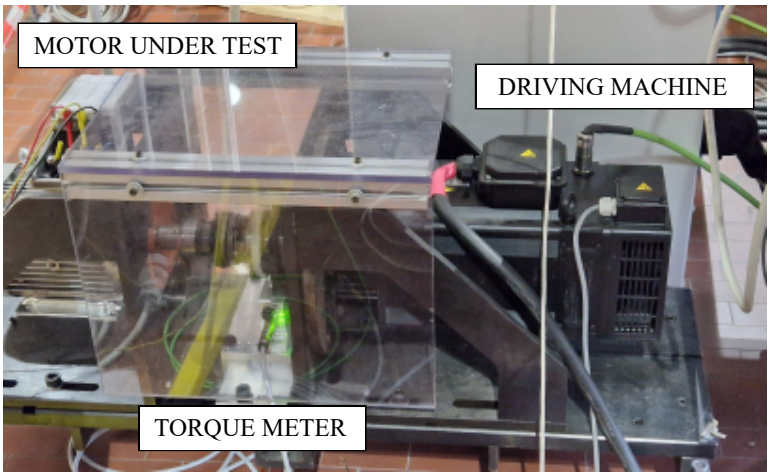


Figure 6.5: Motor Under Test and Driving Machine setup.

by looking at the torque meter measurement on the oscilloscope. Since this is strongly affected by noisy oscillations, as can be seen from the instantaneous signal plot, the torque average value is computed considering the analyzed time interval. As reported, the average torque value is approximately equal to the wanted one. Then, the system response due to a torque variation is highlighted in Figure 6.7, by applying a reference torque step from 0 Nm to 5 Nm. Since also in this case the measurement is affected by noisy oscillations, a moving average filter is applied and the resulting waveform is plotted. As can be seen, after the transient has elapsed, the wanted torque value is obtained.

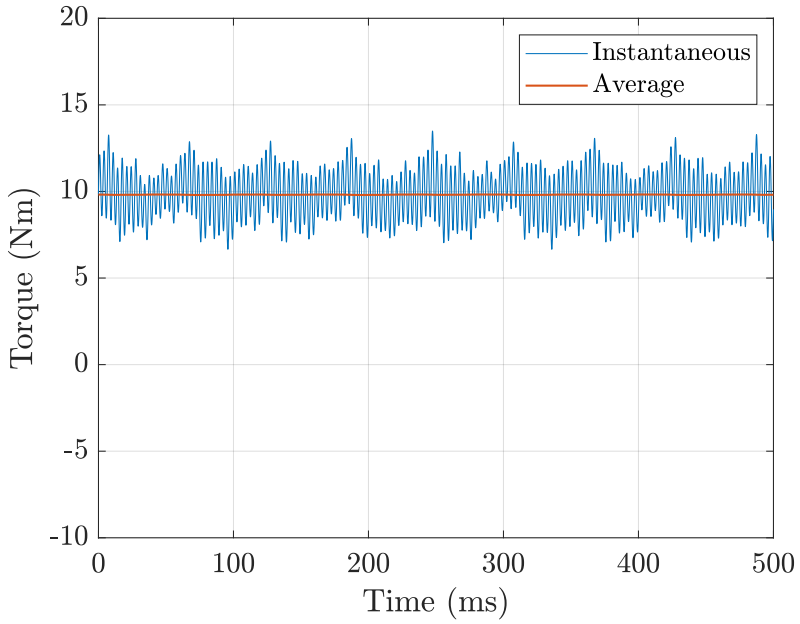


Figure 6.6: Torque regulation oscilloscope waveforms.

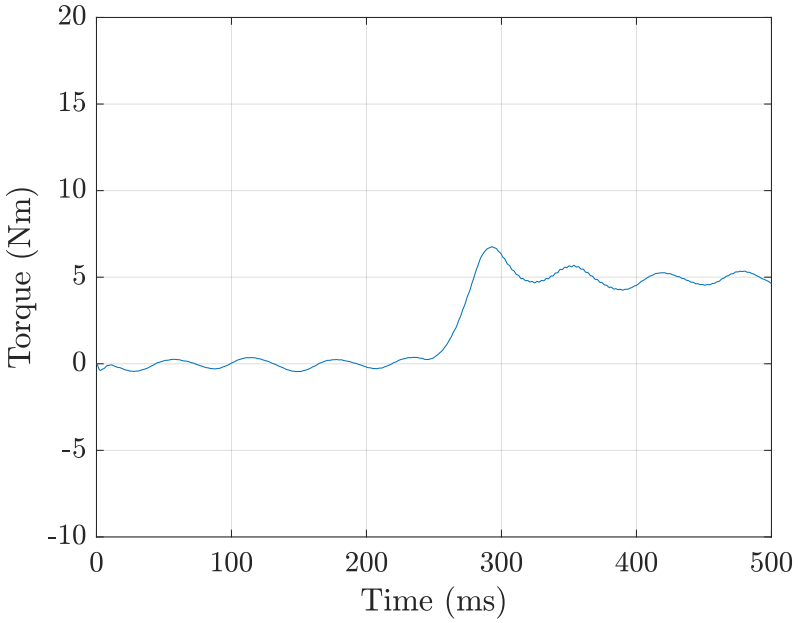


Figure 6.7: Torque regulation after a reference step plot.

After that, an external speed control loop is added, as shown in Figure 6.8: the speed of the motor is compared with a reference value and the error between them is fed to a PI regulator which produces the reference torque value. Thus, through the dSPACE system, the user provides the wanted reference speed in this case instead of the reference torque, which is regulated accordingly depending on the speed error. Since the inertia of the tested PMASR motor is not known, the K_P and K_I gains values are manually tuned in order to achieve the wanted regulation [3]. The speed of the motor, in the implemented case, is obtained by exploiting the mechanical angle.

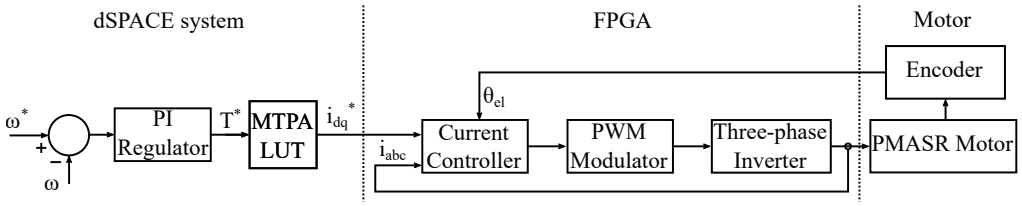


Figure 6.8: Torque control system with external speed loop block scheme.

A second experiment is then performed by configuring the DM to impose a load torque independently of the shaft rotational speed: starting from a null value, the reference speed is set to 500 rpm and then back to 0 rpm. The result of this test is reported in Figure 6.9: as shown, the motor speed is well regulated. After that, the reference speed is set to 500 rpm and the load torque is varied: this produces a change in the motor speed value. The

speed control reacts to this variation adjusting the reference torque value and the wanted speed is recovered. An example of this behavior is reported in Figure 6.10 where the response of the control system due to load torque variations is depicted.

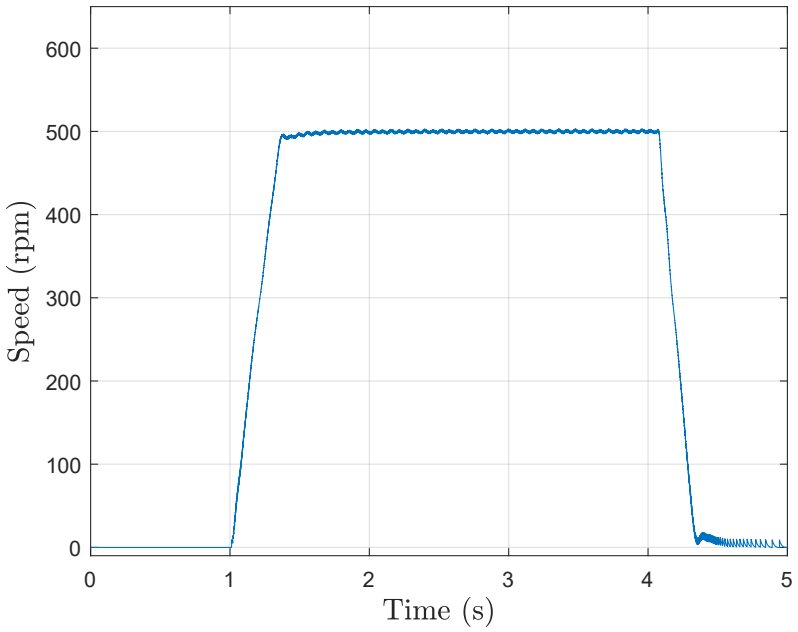


Figure 6.9: Speed regulation plot.

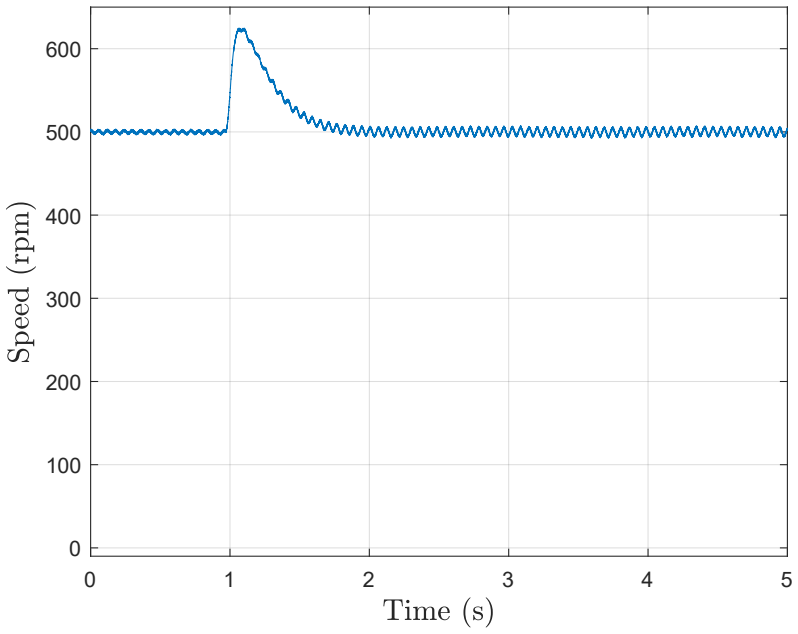


Figure 6.10: Speed regulation after a load torque variation plot.

Chapter 7

Conclusions

In this thesis, the Simulink HDL Coder automatic code generation tool effectiveness has been validated as a method to minimize the programming time which is necessary from the design and simulation of a power converter control system to the experimental validation. Different high-performance digital control systems have been designed and simulated on Simulink. After that, the corresponding VHDL code has been produced and used for the control system implementation on a Xilinx FPGA, to be afterwards experimentally tested. It is worth noticing that the automatically generated code needed to be manually modified in order to obtain the wanted control system behavior. Even though the introduced changes were few in number, this aspect compromises the system complete automatic code generation. Anyway, a script can be used to resolve this drawback. An average current digital control system has been designed for an H-bridge converter and then extended to a three-phase inverter. A dSPACE rapid prototyping system has been deployed in the system to ease and improve the experimental validation. Finally, the torque control of a permanent magnet assisted synchronous reluctance motor has been designed and an external speed control loop has been added. In particular, in this thesis work, I:

- designed several power converters control systems;
- simulated each component of the control systems;
- generated the VHDL code corresponding to each control system through Simulink HDL Coder and modified it to obtain the wanted operation;
- implemented each control system on the FPGA;
- wrote the VHDL codes for additional blocks which were inserted in the system for instance for protection reasons or to manage the acquisitions;
- experimentally validated every designed control system.

Since every implemented control system has been experimentally tested and the correct behavior has been always obtained, the Simulink HDL Coder tool effectiveness has been proved. For these reasons, it can be considered an excellent tool for every system automatic code generation.

Bibliography

[1] Ricardo P. Aguilera et al. "Chapter 2 - Basic Control Principles in Power Electronics: Analog and Digital Control Design". In: *Control of Power Electronic Converters and Systems*. Ed. by Frede Blaabjerg. Academic Press, 2018, pp. 31–68. ISBN: 978-0-12-805245-7. DOI: https://doi.org/10.1016/B978-0-12-805245-7.00002-0. URL: https://www.sciencedirect.com/science/article/pii/B9780128052457000020.

[2] Simone Buso and Paolo Mattavelli. *Digital Control in Power Electronics*. Morgan & Claypool, 2006. URL: https://ieeexplore.ieee.org/document/6813194.

[3] N. Mohan, T.M. Undeland, and W.P. Robbins. *Power Electronics: Converters, Applications, and Design*. 2nd ed. Wiley, 1995. ISBN: 978-0-471-58408-7.

[4] Muhammad H. Rashid. *Power Electronics Handbook*. 3rd ed. Elsevier Science, 2011. ISBN: 978-0-12-382037-2.

[5] R. W. Erickson and D. Maksimović. *Fundamentals of Power Electronics*. 2nd ed. Springer New York, NY, 2001. ISBN: 978-1-4757-0559-1.

[6] Luca Corradini et al. *Digital Control of High-Frequency Switched-Mode Power Converters*. 1st ed. Wiley-IEEE Press, 2015. ISBN: 978-1-118-93510-1.

[7] Shan He et al. "A Review of Multisampling Techniques in Power Electronics Applications". In: *IEEE Transactions on Power Electronics* 37.9 (2022), pp. 10514–10533. DOI: 10.1109/TPEL.2022.3169662.

[8] Simone Buso, Tommaso Caldognetto, and Danilo Iglesias Brandao. "Dead-Beat Current Controller for Voltage-Source Converters With Improved Large-Signal Response". In: *IEEE Transactions on Industry Applications* 52.2 (2016), pp. 1588–1596. DOI: 10.1109/TIA.2015.2488644.

[9] The MathWorks Inc. *Simulink - Simulation and Model-Based Design*. URL: https://it.mathworks.com/products/simulink.html.

[10] The MathWorks Inc. *HDL Coder - Matlab & Simulink*. URL: https://it.mathworks.com/products/hdl-coder.html.

[11] Xilinx. *Vivado Design Suite*. URL: https://www.xilinx.com/products/design-tools/vivado.html.

[12] Guasch componentes y electronica de potencia. *MTL-CBI10040N12IXFE Datasheet*. URL: https://www.e-guasch.com/onlinedocs/catalogue/datasheets/power%20stacks/power%20modules/mt%20series/MTL/MTL-CBI0040F12IXHE_i.pdf.

[13] Analog Devices. *3 MSPS, 12-/10-/8-Bit ADCs in 6-Lead TSOT*. AD7276/AD7277/AD7278 Datasheet. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/ad7276_7277_7278.pdf.

[14] Texas Instruments. *DAC124S085 12-Bit Micro Power Quad Digital-to-Analog Converter With Rail-to-Rail Output*. DAC124S085 Datasheet. URL: https://www.ti.com/lit/ds/symlink/dac124s085.pdf?ts=1651584000121&ref_url=https%253A%252F%252Fwww.google.com%252F.

[15] *Clarke & Park Transforms on the TMS320C2xx Application Report*. Literature Number: BPRA048. Texas Instruments. 1997. URL: https://www.ti.com/lit/an/bpra048/bpra048.pdf.

[16] dSPACE GmbH. *dSPACE MicroLabBox*. URL: https://www.dspace.com/en/pub/home/products/hw/microlabbox.cfm.

[17] Hamidreza Heidari et al. "A Comparison of the Vector Control of Synchronous Reluctance Motor and Permanent Magnet-Assisted Synchronous Reluctance Motor". In: *2021 XVIII International Scientific Technical Conference Alternating Current Electric Drives (ACED)*. 2021, pp. 1–6. DOI: 10.1109/ACED50605.2021.9462265.

[18] Minghu Yu. "Analysis of hybrid permanent magnet assisted synchronous reluctance motor for compressor". In: *2013 International Conference on Electrical Machines and Systems (ICEMS)*. 2013, pp. 1256–1259. DOI: 10.1109/ICEMS.2013.6713351.