UNIVERSITY OF SOUTHAMPTON

Modelling and Control of Grid-Tied NPC Inverters

by

Jorge Guzman Guemez

A thesis submitted for the degree of

Doctor of Philosophy

in the

Faculty of Engineering and Physical Sciences

Engineering Sciences

January 2020

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

ENGINEERING SCIENCES

Doctor of Philosophy

Modelling and Control of Grid-Tied NPC Inverters

by Jorge Guzman Guemez

Grid-tied inverters used for interfacing renewable energy sources and energy storage systems to the grid are set to be essential components in the future smart grid. This thesis focuses on the modelling and control of grid-tied neutral-point-clamped (NPC) inverters, including experimental implementation and validation.

Switched-state-space models of both single-phase and three-phase NPC inverters are developed. Output voltage and capacitor balancing is governed by the transistors of the inverter, which are the only control inputs available. The models allow for the switching nature of the inverter and its non-linear dynamics.

A single-phase NPC switched-state-space model is used to develop and assess the performance of an output voltage regulator, including a space-vector pulse-width modulation (SVPWM) technique that maintains the dc link capacitors balanced. The simulation results demonstrate the efficacy of the controller and the simulation technique.

Similarly, simulation studies were carried out to examine the performance of a three-phase NPC inverter using proportional-integral (PI) decoupled control of active and reactive power. The results of this work were used to provide a bench mark to assess the performance of a novel direct power control (DPC) scheme,

including capacitor balancing. Particular attention was given to power and vector analysis in both schemes.

The new DPC was evaluated experimentally using a three-phase NPC inverter driven by a field-programmable gate array (FPGA). The results show good agreement with the theoretical predictions and demonstrate the proposed DPC's performance. Technical documentation of the entire experimental set up is provided and hardware limitations such as resolution of voltage sensors, signal filtering, and switching speed of transistors are observed.

Contents

List of Figures

List of Tables

Declaration of Authorship

I, Jorge Guzman Guemez, declare that this thesis entitled *Modelling and Control of Grid-Tied NPC Inverters* and the work presented in it are my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;

2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;

3. Where I have consulted the published work of others, this is always clearly attributed;

4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;

5. I have acknowledged all main sources of help;

6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;

7. Either none of this work has been published before submission, or parts of this work have been published as shown in the list of publications.

Jorge Guzman Guemez

Signed:

Date:

List of Publications

- J. Guzman-Guemez, "Modelling and Control of a Single-Phase Three-Level NPC Inverter connected to the Grid," in *PhD Presentation Showcase of the United Kingdom Automatic Control Council*, London, UK, 2015.

- J. Guzman-Guemez, D. S. Laila, and S. M. Sharkh, "Modelado y Control de un Inversor Monofásico de Tres Niveles tipo NPC Conectado a la Red Eléctrica," in *Memorias del Simposio Becarios CONACyT en Europa*, Strasbourg, France, 2016.

- J. Guzman-Guemez, D. S. Laila, and S. M. Sharkh, "State-Space approach for Modelling and Control of a Single-Phase Three-Level NPC Inverter with SVPWM," in *2016 IEEE Power and Energy Society General Meeting*, Boston, MA, 2016.

- J. Guzman-Guemez, L. Michel, S. Perkins, A. Arvanitapoulos, D. S. Laila, N. Lophitis, and S. M. Sharkh, "A Modelling Performance Assessment of Si/SiC IGBT and MOSFET," in *To be published.*

- J. Guzman-Guemez, X. Yan, S. M. Sharkh, "Analytical comparison of DPC and PI Control for a Three-Phase Grid-Tied NPC Inverter" *To be published.*

- J. Guzman-Guemez, X. Yan, S. M. Sharkh, "A New DPC approach with capacitor balance for a Three-Phase Grid-Tied NPC Inverter" *To be published.*

Acknowledgements

I would like to thank Professor Suleiman Sharkh, my supervisor, for his unbelievable patience, wise advice and supervision. Thanks to Dr. Dina Laila who also provided me supervision and motivation for my research. Very much appreciated is the support of Dr. Xingda Yan, who dedicated time to give me guidance and significant technical help. Andy Westerman was key to the development of the prototype for which I thank him. I would also like to acknowledge Dr. Georgios Orfanoudakis for sharing insights while drafting my thesis.

I acknowledge my closest colleagues Jaime, Giuseppe, Lu, Mu, Robert, Ahmed and Teddy for our work and non-work discussions. Also thanks to those who dedicated me time for personal growth and friendship in this long, long journey at the University of Southampton: Manan, Arslan, Saba, Rafa, Fer, Muthu, Sarah, Muller, Thalía, Fer, Miguel, Gregorio, Lalo, Arron, Maddy, Rosie, Uğur, Giorgos, Tobi, Michelle and Marine.

Special thanks to my now wife, Kayleigh, who has been incredibly supportive in uncountable ways. Thanks to my dearest family Aby, Jaime, Jose and Jorge for their unconditional love, as well as the strength they provided me. Also thanks to the old school primos and my extended families for the additional strength.

Lastly, I acknowledge the support of the Consejo Nacional de Ciencia y Tecnología (CONACYT), who funded four years of this PhD. Moreover, I am grateful to Professor Mike Russell, Prof. Suleiman Sharkh and Dr. Denis Kramer who, directly or indirectly, funded others parts of my PhD.

Chapter 1

Introduction

The energy sector in general is facing an enormous challenge in today's world due mainly to the usage of fossil fuel and its associated combustion problems. These problems are primarily severe environmental pollution, which heavily contributes to climate change [1, 2], and the aggravation of human health issues [3]. This has triggered the start of a decarbonisation stage, of which the Paris Agreement plays a key role [4]. This is an agreement between nearly 200 countries; it aims to reduce greenhouse emissions that are responsible for climate change. Consequently, the energy sector is moving towards using cleaner and renewable sources, along with their inherent problems of intermittency and complex connectivity to the grid. This means more effort is put not only into the production of renewable energy, but also into its storage, transmission and conversion.

The inclusion of battery-powered cars as new electric power consumers (or loads) adds to the complexity of the supply and demand performance of the grid. However, it also represents a unique opportunity for the energy sector, because if seen by the grid as aggregated dispatchable storage (or energy sources), it represents significantly large packages of power available with nearly instant grid interaction, paving the way towards a smart grid. This scenario is also applicable to houses

and buildings and even utilities incorporating battery systems [5]. As these systems become more ubiquitous, users could potentially become buyers or sellers of electricity, according to their needs. Examples include Vehicle-to-Grid (V2G) in Figure 1.1, Vehicle-to-Home (V2H), Home-to-Grid (H2G) and Home-to-Vehicle (H2V) applications [6, 7, 8]. In this context, Direct Current (DC) to Alternating Current (AC) power converters, together with their power control and efficiency, become increasingly important for grid connection as availability of battery-stored power increases.

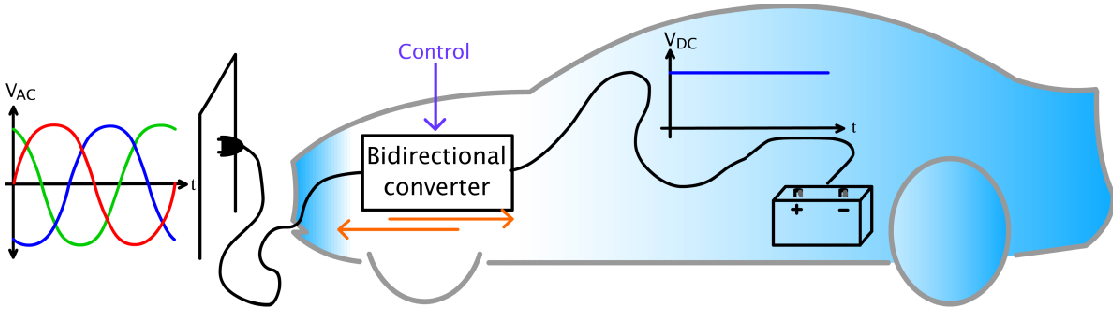


FIGURE 1.1: Converter application for vehicle charge/discharge.

The Neutral-Point-Clamped (NPC) type inverter has been proven to be effective at handling mid to high power (up to several megawatts) [9, 10] and is the most widely used for industrial applications such as wind turbines, motor drives, pumps, uninterruptible power supplies, among others [11, 12]. It features high efficiency and low total harmonic distortion, showing versatility using various controlling techniques [13]. However, analyses of power control and switching losses for grid connection applications are still developing for research and commercialisation purposes.

A way of improving the efficiency of inverters is by studying its dynamics. Due to the necessity to control power and current ripple, a model which allows these variables to be controlled is desirable. Various approaches to modulate and control NPC inverters in general have been reported in the literature, such as Space Vector Modulation (SVM), Carrier based modulation, PI Control, Direct Power Control (DPC) and Predictive Control. Most of these works are for machine applications,

but there are publications which explore these same techniques when the inverter is connected to the grid. However, there is still important analysis regarding these techniques which are tightly related to switching losses and control which need to be addressed.

This PhD research studies the operation of a grid-tied NPC converter, its power analysis and control. Models for single– and three–phase NPC inverters are developed. Techniques for the modulation and control are studied and compared, namely Space-Vector Pulse-Width-Modulation (SVPWM), PI Control and a novel DPC. Both simulations and experiments are used to aid the design and assess the performance of the models with the proposed modulation and control techniques.

1.1 Scope of the Research

Although some fundamental knowledge is outlined for inverter and rectifier modes of converters in this document, the NPC converter is only studied in inverter mode. Its model is detailed, its switching techniques identified and novel ones proposed. An FPGA-driven NPC inverter is constructed to provide an experimental platform for evaluating the proposed modulation and control techniques.

- *Modelling.* State-space based models are developed to simulate the performance of both single– and three–phase NPC inverters. This is not only useful for understanding system dynamics, but it is also a first step towards stability analysis.

- *Modulation and Control.* Firstly, SVPWM with capacitor balancing is proposed for a single–phase NPC inverter and output voltage regulation is addressed and validated with simulations. Secondly, an extensive analysis of grid interaction of a three-phase NPC inverter is carried out. The work is

focused on DPC and a PI controller, which are compared in terms of switching losses, current ripple, capacitor balancing and active and reactive power. Advantages of each technique are presented and discussed.

- *Experiment.* The modelling and control techniques are experimentally tested using an off-the-shelf three-phase NPC inverter (3L SKiiP 28 MLI07E3V1 Evaluation Inverter by Semikron) and using an FPGA board (Altera EP4CE1 15F29C7) for control. The results are compared with simulations in terms of current ripple and power control.

This thesis also includes extra work on the development of a model for new Si/SiC MOSFETs and IGBTs in Appendix A, the work was carried out at the beginning of the PhD, but was not continued following a change of supervisor and direction of the work.

1.2 Aim and Objectives

The aim of this work is to analyse, develop and implement modulation and control techniques for grid-tied NPC inverters, assessing their limitations and advantages with a power control focus. Additionally, the work aims to build an experimental platform for testing of these techniques.

1.2.1 Objectives

- Review inverters and grid-tied inverters. Become familiar with inverter dynamics, modulation, single– and three–phase types and comprehension of the Park and Clarke transformations to explore modulation and control of the models. Develop models for single– and three–phase NPC inverters compatible with different modulation techniques.

- Simulate a single-phase NPC inverter with voltage control. Validate the model and implement a new Space-Vector Pulse-With-Modulation in order to achieve output voltage regulation and capacitor balance. Test the strategy with a sudden change of load.
- Develop and implement a three-phase grid-tied NPC inverter with a novel power control. Design both DPC and PI Controllers with carrier-based modulation to provide power control and capacitor balancing. Compare and explain simulation and experimental results in terms of power control, current ripple and efficiency.

1.3 Contents of the Report

In Chapter 2 fundamental operation, modulation and control techniques for NPC inverters are introduced, and relevant works reported in literature are reviewed. Key concepts, terms and variables such as modulation, frequency and power control for grid connection are identified. Additionally, the modelling and of the single– and three–phase NPC inverters using a state-space approach is presented.

Chapter 3 addresses output voltage regulation for a single–phase NPC inverter by using a new SVPWM approach. Switching patterns using this technique are explained in detail and a proposed control system for voltage regulation while keeping the capacitors balance is put to test using simulation studies.

Next, Chapter 4 provides vector and power analyses of the three–phase NPC inverter when connected to the grid. It focuses on the PI decoupled control and DPC techniques. It presents a novel DPC table and provides observations on effective switching frequency and its relationship with losses.

Experimental work is reported in Chapter 5 and presents a 250 [W] three–phase off-the-shelf NPC inverter connected to the grid and controlled by an FPGA.

Interaction of hardware and software parts is detailed. Extensive analysis and results under typical operating conditions are presented and discussed.

Finally, Chapter 6 highlights the most important outcomes of this thesis and its implications, including proposals for future work.

1.4 Research Contributions

- Functional single– and three–phase models for the NPC inverter and validation of all its parts.
- Development and simulation of novel SVPWM modulation and control for a single–phase NPC inverter that is robust to load changes while keeping output voltage regulation and capacitor balance.
- Development of a new DPC approach capable of controlling active and reactive power very quickly while keeping the capacitors balanced. FPGA implementation of this control and performance results of the experimental prototype.
- A comparison between PI Control and the new DPC approach in terms of convergence speed, switching losses and current ripple.

Chapter 2

NPC Inverters: Structure, Modelling and Control

As mentioned earlier, due to the grid presenting an increased dependency on renewable sources, more effort will be needed to store, convert and transmit this energy when available. In fact, on top of the already ongoing Electric Vehicle (EV) commercialisation, storage systems connected to the grid for houses are just taking off. This is, without taking into account other storage systems potentially implemented by energy providers. Converter technology play a big role in many fields and battery-equipped grid-tied systems require quality converters for applications such as EVs, domestic and industrial energy storage, photovoltaic farms, among others. Hence, the models and dynamics of converters need to be understood more than ever in order to provide efficient, reliable and quality energy conversion to and from the grid.

The general purpose of power electronic converters is to convert input (or 'raw') power into a conditioned output power according to the application. Conditioning of power could vary greatly, depending on voltage and current amplitude, shape, frequency, harmonics, among others. AC–DC converters, or rectifiers, are fed by

an AC voltage and output a unipolar voltage (or DC). DC–AC converters, or inverters, are fed with a DC voltage and output a bipolar AC wave with a desired magnitude, frequency and phase angle. There are also bidirectional converters, capable of changing the power direction (interchanging input and output), performing as rectifiers or inverters, as required. This work focuses specifically on the NPC inverter (i.e. DC–to–AC mode only). Analysis of rectifier operation (AC–DC) and DC–DC converters is out of the scope of this thesis, as mentioned in the introduction.

Fundamentals of the operation of an NPC inverter and its grid-connected dynamics are introduced in this chapter. Additionally, the models for a single– and a three–phased NPC inverter are deducted.

2.1 The NPC Inverter

A conventional three-phase converter is generally formed of six transistors, two per leg and it can only switch between two voltage levels. The NPC inverter, a type of multilevel converter, is capable of switching between three voltage levels per leg. The three-level NPC was first introduced in [14] and was reported to handle double of kilo-volt-amperes of a conventional converter, by suitable design. After many years, the NPC type of converter has become the most widely used inverter in a variety of industrial applications, capable of operating up to 5 [MVA] alone or up to 27 [MVA] in a specific topology [13] and driving 6 [kV] motors [11]. Advantages of the three-level NPC converters in contrast with the two-level ones are higher ripple frequency (hence, smaller filter), increased efficiency and lower total harmonic distortion (THD) [9, 10]. Some applications of this converter include roll mills, windmills, pumps and ventilators, and can potentially work for many more such as Uninterruptible Power Supply (UPS) systems [12] or even as bidirectional Vehicle-to-Grid (V2G) converters [8].

As this thesis presents single– and three–phase topologies of the NPC inverter, an analysis per leg, X, is provided. For a single–phase, there is Leg A and Leg B, whereas for three–phase there is Leg A, Leg B and Leg C. Figure 2.1 depicts the architecture and operation of one leg of an NPC converter. A DC input voltage is represented as v_{DC}, which feeds two series connected capacitors, C_1 and C_2 in parallel with the leg. A leg has four switches numbered from 1 to 4. In this way, a switch can be identified as S_{xi} where $x = \{a, b, c\}$ respective to the leg and $i = \{1, 2, 3, 4\}$ respective to the transistor number from the top to the bottom. A transistor is assumed ideal (i.e. a perfect switch) and always holds a binary value so that $S_{xi} = \{0, 1\}$, where 1 and 0 identify a switch as on or off, respectively. Each leg also has a pair of series-connected diodes (D_1 and D_2) connected to the terminals of the two central transistors; the anode of the top diode D_1 is connected to the mid-point NP between the capacitors.

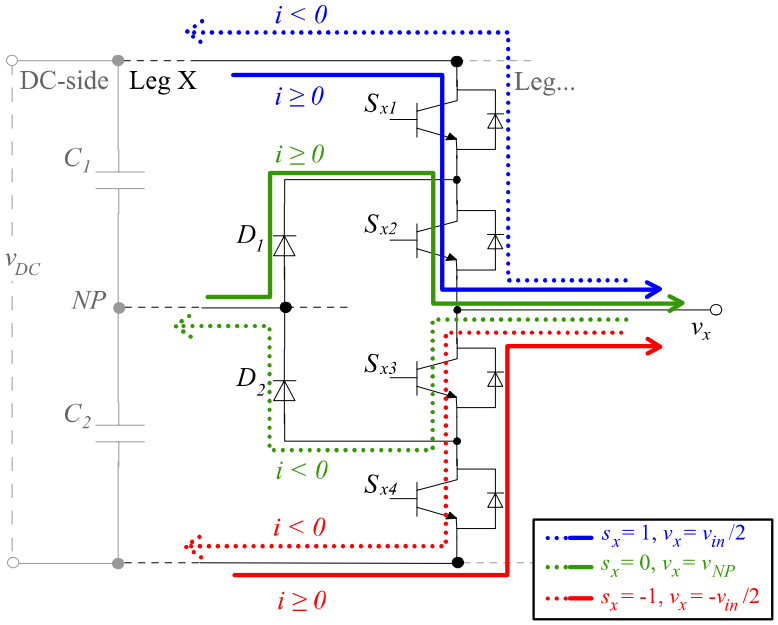


FIGURE 2.1: One leg of an NPC inverter.

A Leg X switching vector, formed of four transistors, is defined as

$$s_x = \begin{bmatrix} S_{x1} \\ S_{x2} \\ S_{x3} \\ S_{x4} \end{bmatrix} = [S_{x1}\ S_{x2}\ S_{x3}\ S_{x4}]^\top, \tag{2.1}$$

where $s_x = \{1, 0, -1\}$ governs the only allowed switching patterns for all S_{xi} as follows:

$$s_x = \begin{cases} 1 & \text{for } S_{x1} = S_{x2} = 1 \text{ and } S_{x3} = S_{x4} = 0, \\ 0 & \text{for } S_{x2} = S_{x3} = 1 \text{ and } S_{x1} = S_{x4} = 0, \\ -1 & \text{for } S_{x1} = S_{x2} = 0 \text{ and } S_{x3} = S_{x4} = 1. \end{cases} \tag{2.2}$$

Therefore, in a converter with m number of legs and n allowed switching patterns (or levels) per leg, the number of all possible operating states of an NPC inverter is given by n^m. This means that for a single–phase inverter the number of states are $3^2 = 9$, while for a three-phase inverter there are $3^3 = 27$ states. Further notation is defined: $s_{ab} := [s_a\ s_b]$ and $s_{abc} := [s_a\ s_b\ s_c]$, e.g.

$$s_{abc} = [1\ 0\ -1] \Rightarrow [s_a\ s_b\ s_c] = [1\ 0\ -1] \Rightarrow \begin{bmatrix} S_{a1} & S_{b1} & S_{c1} \\ S_{a2} & S_{b2} & S_{c2} \\ S_{a3} & S_{b3} & S_{c3} \\ S_{a4} & S_{b4} & S_{c4} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.3}$$

In Figure 2.1 the voltage v_x corresponds to that of the middle point of the leg. It can be deduced from equation (2.2) and the coloured lines of the same figure, that

the voltage output per leg is given by

$$v_x = \begin{cases} +v_{in}/2 & \text{for } s_x = 1, \\ 0 & \text{for } s_x = 0, \\ -v_{in}/2 & \text{for } s_x = -1, \end{cases} \tag{2.4}$$

where the solid and dotted coloured lines show the sign convention of the current flow. This means that voltage terminals v_a and v_b will be present in a single–phase inverter, whereas voltage terminals v_a, v_b, and v_c will be present in a three–phase inverter. A voltage difference between voltages v_a and v_b will be represented as v_{ab}, and between v_a and neutral point NP as v_{aNP}.

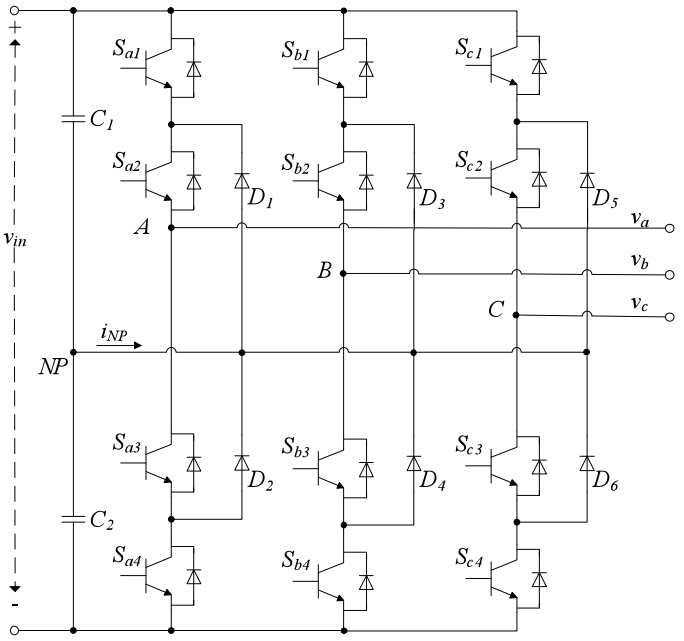


FIGURE 2.2: Circuit diagram of an NPC inverter.

To illustrate the description above, take a look at Figure 2.2, where a three-phase three-level NPC inverter is shown. A switching pattern $s_{abc} = [1\ 0\ -1]$ will result in voltages $v_{ab} = +v_{in}/2$, $v_{bc} = +v_{in}/2$, and $v_{ca} = -v_{in}$. By using these voltages, a sinusoidal wave could be produced by appropriately changing the switching patterns in each leg.

Now that the basic operation of the NPC inverter alone has been described, it is important to understand its dynamics when tied to the grid or other external voltage sources. As there are specific variables intended for monitoring and control, such as currents and voltages, it is necessary to define them (and their dependency with other variables) in models which include this coupling.

2.2 Modelling of NPC Inverters

State-space modelling is a very well known approach to describe dynamical systems by a set of differential equations, which relates the input variables to the output variables [15]. A general state-space model of a dynamical system is written as:

$$\dot{x}(t) = f(x, u, t), \tag{2.5}$$

$$y(t) = h(x, u, t), \tag{2.6}$$

with $x(t)$ states, $u(t)$ inputs and $y(t)$ outputs. The functions $f(x, u, t)$ and $h(x, u, t)$ are vector functions. While most systems in nature are nonlinear, it is quite common to linearize the dynamics. In this case, the state-space model can be written in matrix form as:

$$\dot{x}(t) = A(t)x(t) + B(t)u(t), \tag{2.7}$$

$$y(t) = C(t)x(t) + D(t)u(t), \tag{2.8}$$

where $A(t)$ is called the state matrix, $B(t)$ the input matrix, $C(t)$ the output matrix, and $D(t)$ the direct transmission matrix.

This input/output framework is particularly important as there is a large number of tools to analyse such models [16]. In other words, the state-space approach

in (2.5)–(2.6) has been chosen to model this electrical system for convenience in stability analysis and control design as shown in [17, 18, 19, 20].

The objective is to produce switched models suitable for the control methods studied in this work with a certain degree of steady-state accuracy. The models developed are based on the following hypotheses:

- Transistors are considered ideal switches with on-state ('1') or off-state ('0'). That is, during the on-state a transistor resistance and inductance are assumed to be zero, whereas in the off-state the transistor is considered as an infinitely large resistance.
- Passive elements (such as resistors, inductors, capacitors) are considered linear and time invariant.
- DC power supplies can provide infinite power.

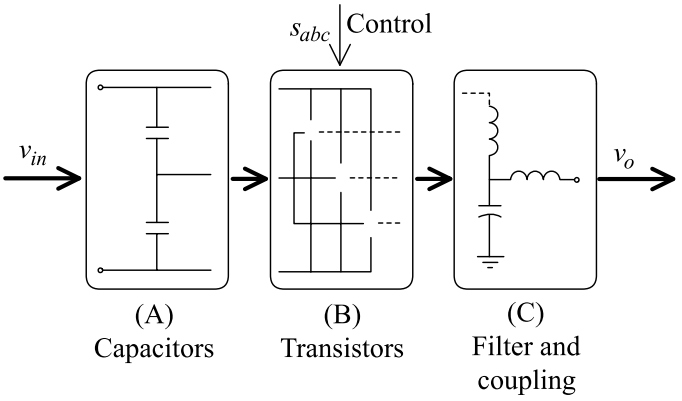


FIGURE 2.3: Block diagram for NPC inverter modelling.

The NPC inverter is composed of three main subsystems illustrated in Figure 2.3: the dc-link capacitors, the transistors and the filter. Thus, an NPC inverter can be represented as a hybrid system. The capacitors and transistors subsystems are modelled as switching dynamic systems, while the filter is modelled as a continuous-time dynamic system.

The following state-space models describe the dynamics of the NPC inverters when tied to another source (the grid in this case) in a structured manner and part by part. New models for a single– and three–phase three-level NPC inverter are derived to investigate the power dynamics from the DC voltage to the AC voltage of the grid.

2.2.1 Single–Phase Model with LC Filter

The circuit diagram of a grid-tied single-phase NPC inverter with an output LC filter is shown in Figure 2.4. An input voltage source is represented as v_s with its corresponding internal resistance R_s connected in parallel with two capacitors C_1 and C_2, with voltages v_1 and v_2 across them, respectively. The single-phase NPC inverter is expected to produce a fundamental sinusoidal wave between the points v_a and v_b. The neutral point is denoted as NP. Voltages between legs are denoted as v_{ab} for the voltage between points A and B, v_{aNP} for the voltage between points A and NP and so on. The filter comprises an inductor L_f with an associated resistor R_f and an ideal capacitor C_f (ignoring its equivalent series resistance). L_o and R_o represent the grid inductance and resistance, respectively. v_g is the grid voltage. The DC voltage of the capacitors C_1 and C_2 need to be converted so that ultimately v_o is a sinusoidal wave, or a regulated output.

As mentioned earlier, the capacitors' and transistors' subsystems are modelled as switching dynamics as it involves the switching mechanism. For a single-phase system, the three-level converter has two input terminals. In the general case (when v_{ab} is not in phase with v_o), using Kirchhoff's laws for $v_{ab} \geq 0$ the voltage supply is described by the equations

$$C_1 \frac{d}{dt} v_1 = i_{in} - i_f W_1, \tag{2.9}$$

$$C_2 \frac{d}{dt} v_2 = i_{in} - i_f W_2, \tag{2.10}$$

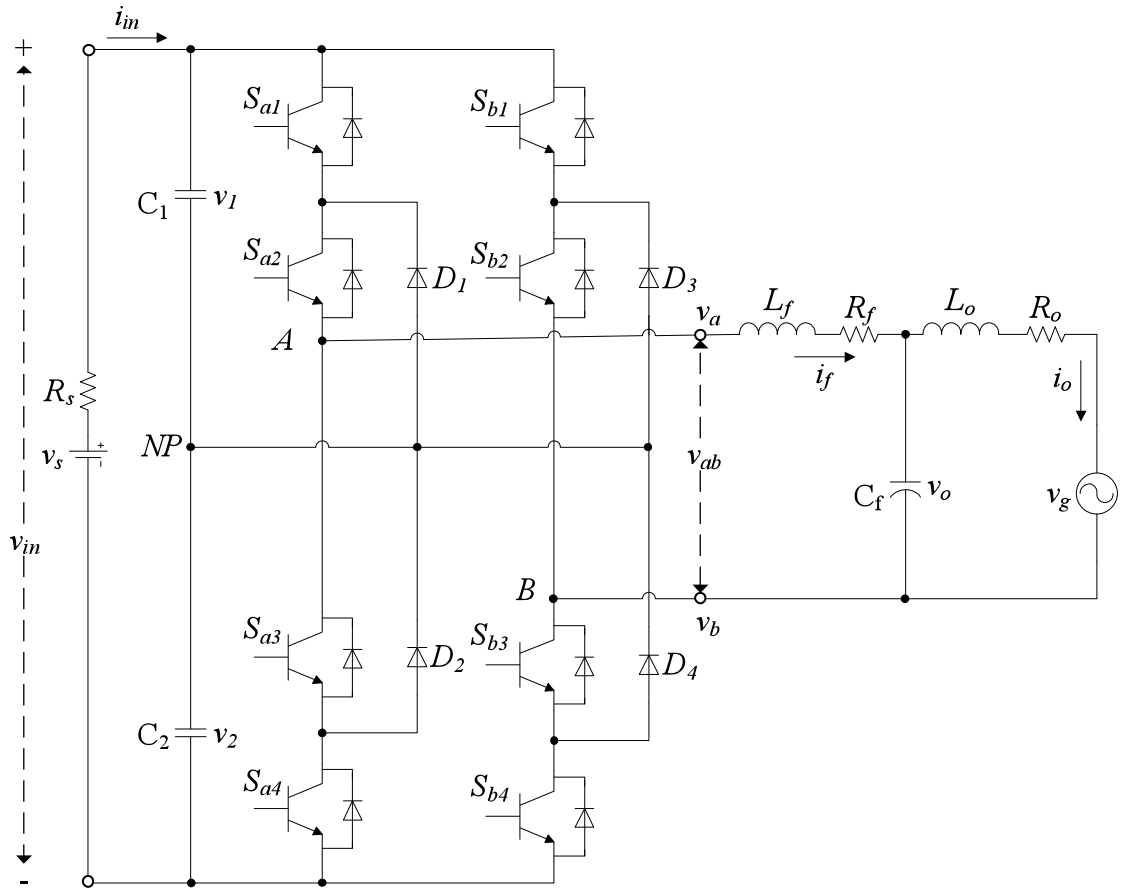


FIGURE 2.4: Circuit diagram of a single-phase three-level grid-tied NPC inverter.

and for $v_{ab} < 0$,

$$C_1 \frac{d}{dt} v_1 = i_{in} + i_f W_2, \tag{2.11}$$

$$C_2 \frac{d}{dt} v_2 = i_{in} + i_f W_1, \tag{2.12}$$

where the input voltage and input current are given by

$$v_{in} = v_1 + v_2, \tag{2.13}$$

$$i_{in} = -\frac{v_{in}}{R_s} + \frac{v_s}{R_s}, \tag{2.14}$$

respectively, and

$$W_1 = \begin{cases} 1 \text{ for } s_{ab} = \{[1,-1],[-1,1],[1,0],[0,1]\} \\ 0 \text{ for } s_{ab} = \{[0,0],[0,-1],[-1,0]\}, \end{cases} \tag{2.15}$$

$$W_2 = \begin{cases} 1 \text{ for } s_{ab} = \{[1,-1],[-1,1],[0,-1],[-1,0]\} \\ 0 \text{ for } s_{ab} = \{[0,0],[1,0],[0,1]\}. \end{cases} \tag{2.16}$$

Assuming the switches are ideal for the single-phase inverter, the transistor arrangement is represented as:

$$v_{ab} = \begin{cases} v_{in} & \text{for } s_{ab} = \{[1,-1]\}, \\ v_{in}/2 & \text{for } s_{ab} = \{[1,0],[0,-1]\}, \\ 0 & \text{for } s_{ab} = \{[0,0]\}, \\ -v_{in}/2 & \text{for } s_{ab} = \{[-1,0],[0,1]\}, \\ -v_{in} & \text{for } s_{ab} = \{[-1,1]\}. \end{cases} \tag{2.17}$$

As the filter subsystem does not involve switching, it is modelled as a continuous-time dynamic one. Using Kirchhoff's laws, the output filter is described as

$$L_f \frac{d}{dt} i_f = v_{ab} - i_f R_f - v_o, \tag{2.18}$$

$$L_o \frac{d}{dt} i_o = v_o - i_o R_o + v_g, \tag{2.19}$$

$$C_f \frac{d}{dt} v_o = i_f - i_o, \tag{2.20}$$

where all the parameters are shown in Figure 2.4. The combination of the switching subsystems (2.9)–(2.12), (2.17) and the continuous subsystem (2.18)–(2.20) define the state-space representation of the NPC inverter. Note that due to the

involvement of the switching mechanism, this model is a nonlinear state-space model as in (2.5). Further verification of accuracy is provided later in Chapter 3.

2.2.2 Three–Phase Model with RL Filter

The model of a three-phase NPC inverter with a simple *RL* filter is presented in Figure 2.5. On the DC-side, or capacitor part, we have the DC source v_s, its associated resistance R_s and the total DC input voltage v_{in} as well as the DC current flowing to the inverter i_{in}. The DC source feeds the top capacitor C_1 and the bottom capacitor C_2; these are connected in series through a Neutral-Point (NP) and have voltages of v_{C1} and v_{C2}, respectively. The transistor part keeps the same nomenclature used in the previous subsection; it only adds one more leg with the same transistor/diode arrangement. Here, three points of 'raw' AC voltage outputs of the converter are shown: A, B and C, forming v_{ca}, v_{cb} and v_{cc} respective of each leg or phase. In the filter and coupling part, each phase is connected to an inductor L_x and a resistor R_x, which then connects to the corresponding grid terminals v_{gx} with a current flow i_x, where $x = \{a, b, c\}$.

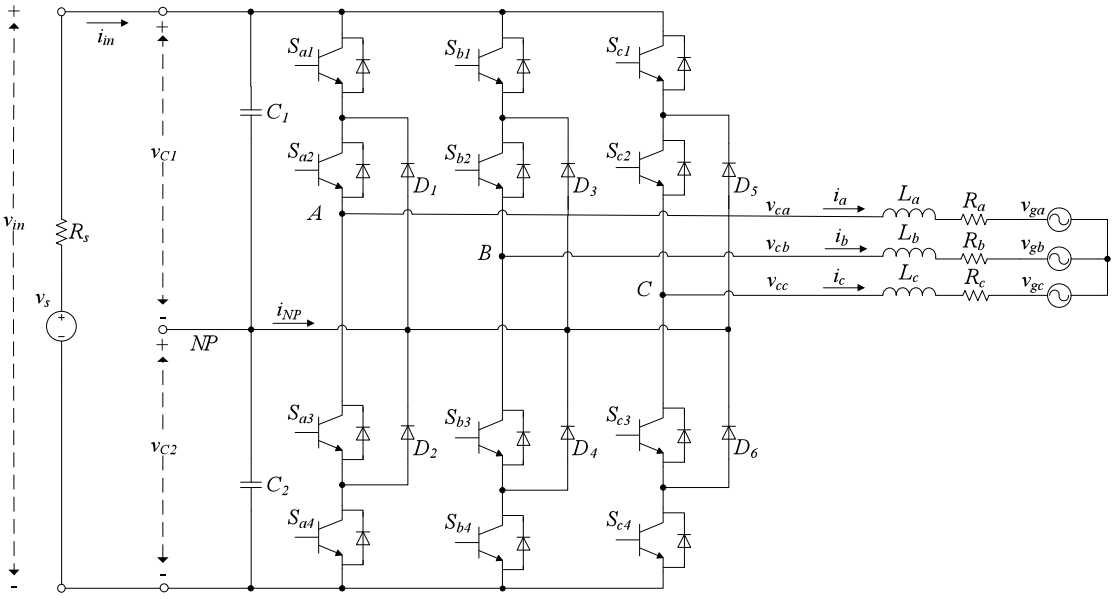


FIGURE 2.5: Circuit diagram of a three-phase three-level NPC inverter with RL filter.

The dynamics of the described system are now presented by parts, as shown in Figure 2.3. The capacitors' part is determined by

$$C_1 \frac{d}{dt} v_{C1} = i_{in} + i_{NP}, \tag{2.21}$$

$$C_2 \frac{d}{dt} v_{C2} = i_{in} - i_{NP}, \tag{2.22}$$

where

$$v_{in} = v_{C1} + v_{C2}, \tag{2.23}$$

$$i_{in} = -\frac{v_{in}}{R_s} + \frac{v_s}{R_s}, \tag{2.24}$$

and by defining $S_{sum} = |s_a| + |s_b| + |s_c|$ and $[S_{aa}\ S_{bb}\ S_{cc}] = [|s_a|\ |s_b|\ |s_c|]$ the current flowing in the i_{NP} point is governed by

$$i_{NP} = \begin{cases} -(S_{aa} i_a + S_{bb} i_b + S_{cc} i_c) & \text{for } S_{sum} = \{1\}, \\ -(S_{aa} - 1) i_a - (S_{bb} - 1) i_b - (S_{cc} - 1) i_c & \text{for } S_{sum} = \{2\}, \\ 0 & \text{for } S_{sum} = \{0, 3\}. \end{cases} \tag{2.25}$$

The switching part of the transistors can be developed in the same fashion as equations (2.15)–(2.17). On the continuous or filter side, the outputs are given by

$$L_x \frac{d}{dt} i_x = v_{cx} - i_x R_x - v_{gx}, \tag{2.26}$$

with

$$L_x = \begin{bmatrix} L_a & L_b & L_c \end{bmatrix}, \tag{2.27}$$

$$R_x = \begin{bmatrix} R_a & R_b & R_c \end{bmatrix}, \tag{2.28}$$

$$v_{cx} = \begin{bmatrix} v_{ca} & v_{cb} & v_{cc} \end{bmatrix}, \tag{2.29}$$

$$v_{gx} = \begin{bmatrix} v_{ga} & v_{gb} & v_{gc} \end{bmatrix}, \tag{2.30}$$

$$i_x = \begin{bmatrix} i_a & i_b & i_c \end{bmatrix}, \tag{2.31}$$

as shown in Figure 2.5. More details on how the involved variables (2.21)–(2.26) respond to changes of s_{abc} are given later in Chapter 4.

The above analyses present examples on how the dynamics of the grid-tied inverter can change according to its architecture and the filter choice or coupling configuration. However, a key feature that the two models have in common is a control input, which is a type of s vector formed by the available leg vectors s_x, i.e. s_{ab} for single phase or s_{abc} for three phase. This feature in the framework above provides a control signal into the effect of switching patterns and controllers of the system as a whole.

In fact, the main challenges of this inverter, as with other inverters, is to control the switching losses and to keep harmonics within appropriate levels, but with the additional challenge of balancing the capacitor's voltage. The purpose of the modulation and control of the inverter is to provide an appropriate output while managing these challenges.

2.3 Modulation and Control

Generally, modulation and control of inverters could be categorised as controlled-switching and uncontrolled-switching. Controlled-switching includes carrier based, Space Vector Modulation (SVM) [21] and Selective Harmonic Elimination (SHE) [22]. Uncontrolled-switching considers variable hysteresis band controllers, such as Direct Torque Control (DTC), and Direct Power Control (DPC) with its variations.

The uncontrolled rectifier mode of the NPC converter is enabled by turning off all transistors. However, this is rarely used as it results in uncontrolled dumping of energy into the capacitor or battery, which can be dangerous. Usually controlled rectification is used. Besides, the inverter mode is the one chosen in this thesis and these transistors require a switching pattern, or a modulation technique.

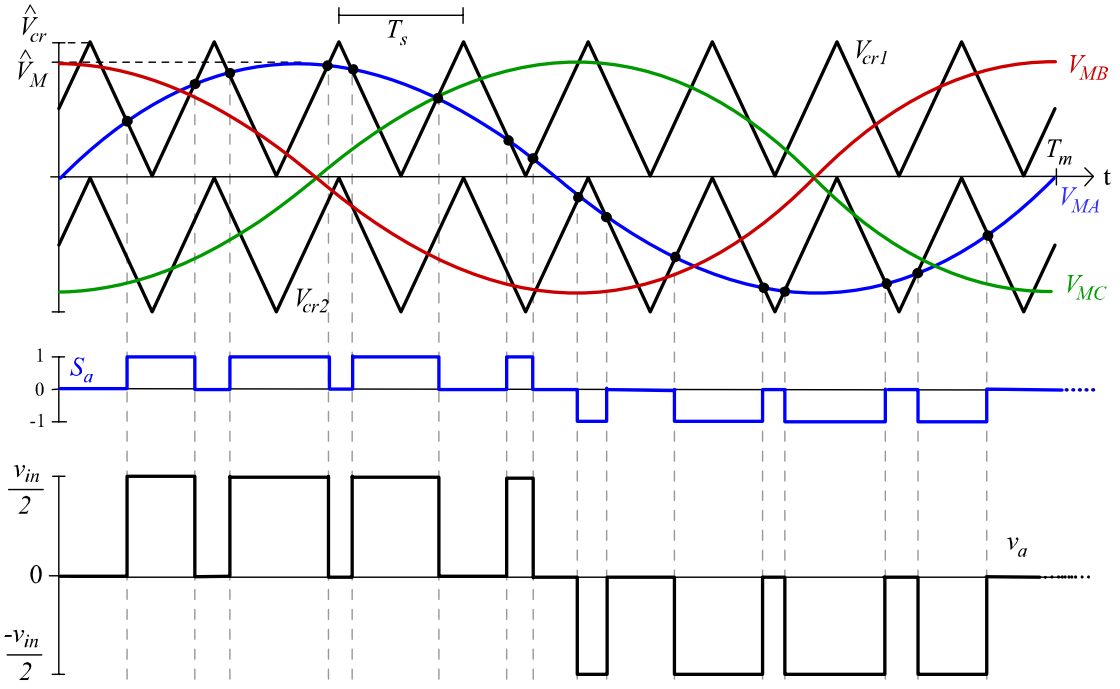


FIGURE 2.6: Illustrative SPWM

One of the carrier based techniques is sinusoidal pulse-width-modulation (SPWM), illustrated in Figure 2.6. This modulation technique uses two triangular carrier signals V_{cr1} and V_{cr2} with T_s periods, and a modulating signal V_{MX} per leg in a

T_m period. In a three-phase inverter these three modulating signals will be shifted by 120° from each other. In Figure 2.6, the black dots on V_{cr1} and V_{cr2} represent a switching point for V_{MA}, which can only cross one of the carrier signals at a time. The switching pattern s_a changes according to equation (2.2), so that

$$S_{a1} = 1 \text{ for } V_{cr1} < V_{MA}, \tag{2.32}$$

$$S_{a2} = 1 \text{ for } V_{cr2} < V_{MA}, \tag{2.33}$$

$$S_{a3} = 1 \text{ for } V_{cr1} > V_{MA}, \tag{2.34}$$

$$S_{a4} = 1 \text{ for } V_{cr2} > V_{MA}. \tag{2.35}$$

Let the peak-to-peak amplitude of any carrier signal (V_{cr1} or V_{cr2}) be $\hat{V}_{cr}$ and the peak-to-peak amplitude of V_{MX} be $2\hat{V}_M$ (where subscript $X = A, B, C$). Back to Figure 2.6, notice that the sum of the peak-to-peak amplitudes of the carrier signals $2\hat{V}_{cr}$ is slightly larger than the peak-to-peak amplitude of the modulating signals $2\hat{V}_{MX}$. The relationship between these two signals is called amplitude modulation ratio and is defined as

$$m_a = \frac{\hat{V}_{cr}}{\hat{V}_M}, \tag{2.36}$$

where the frequency of $\hat{V}_{cr}$ is generally kept constant due to available transistor type, processor speed or others. If over-modulation takes place (i.e. $m_a > 1$), it means that $\hat{V}_{MX} > \hat{V}_{cr}$ and causes an increase of the notches between the signals and the output voltage to have more harmonics [23]. Since harmonics are detrimental in grid-tied inverters, the inequality $m_a < 1$ has to be strictly satisfied.

By defining the switching frequency of V_{cr1} as $f_s = 1/T_s$ and the switching frequency of V_{MX} as $f_m = 1/T_m$, a frequency ratio is defined as

$$m_f = \frac{f_s}{f_m}. \tag{2.37}$$

This means that the required frequency of output voltage per leg is the grid frequency, $f_m = f_g = 50$ [Hz] and the control will be defined by the duty cycle at a fixed frequency, f_m and f_s will remain constant.

In Space-Vector-Modulation (SVM) and its variants [10], a reference voltage vector, or desired voltage vector V^*, is acquired or calculated. SVM uses a vectorised map of the switching patterns and output voltage in which vectors are selected according to a reference voltage, power and/or capacitor's balance requirements. Once this reference vector is located in a vector map such as the one in Figure 2.7, a set of neighbouring vectors (usually the three-nearest) are selected for every set period T. Different from SPWM in which a switching pattern changes twice per leg for every period of the carrier T_s, in SVM three changes occur every period T across all legs. The time in which each of these vectors is on within the period is measured by an algorithm such that the mean value of the output voltage is as close as possible to the desired voltage vector. For the illustrated case in Figure 2.7,

$$V^* \approx \bar{v} = \frac{1}{T}(v_1 t_1 + v_2 t_2 + v_7 t_3), \tag{2.38}$$

$$T = t_1 + t_2 + t_3, \tag{2.39}$$

or even some short vectors could be included such as V_{13} or V_{16} and V_{13} or V_{14}.

On the other hand, there are modulation and control strategies which do not have a controlled or fixed frequency, f_s. This is the case for Direct Torque Control (DTC) and some hybrids of them. The work reported in [24] was arguably the first one to introduce the Direct Torque Control (DTC), which was implemented in a two-level three-phase inverter to drive an induction motor. DTC has a background in machines, where torque needs to be controlled. It also uses a vector map such as in Figure 2.7, but it does not calculate vector averages. This is due to a hysteresis

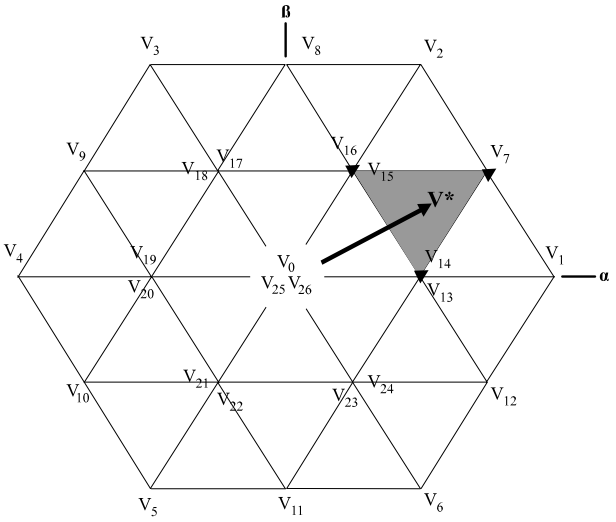


FIGURE 2.7: NPC vector map used in SVM

based algorithm which receives readings of flux, torque and flux angle in the dq or $\alpha - \beta$ frame (Figure 2.8) and outputs one of the vector values stored in a table, which keeps changing according to the flux and torque required. In this manner, each hysteresis block (flux, torque and flux angle) inputs a value to a table which together becomes a pointer to switching states in the table, which is dependent in these three variables. The output vector then, determines the switching pattern for each leg of the inverter.

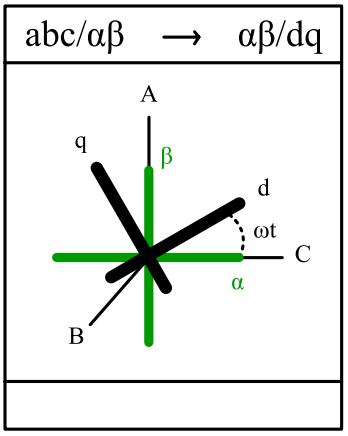


FIGURE 2.8: Illustration of abc-alpha-beta-dq transformations

DTC has been improved and modified since it was first released [25], evolving into Direct Power Control [26] and it is still developing today [27]. In an analogous fashion to DTC, DPC receives readings of active and reactive power and applies

the appropiate voltage vector. Direct Power Control (DPC) has been successfully implemented in many other ways such as grid voltage modulated (GVM) DPC to reduce current THD [28] or virtual flux based model predictive direct power control (VF-MPDPC) to improve robustness against mains harmonics [29]. These efforts are put towards making the NPC converter more efficient with decreased THD. More details are provided in Chapter 4.

As the switching timings can be selected by the user in order for an algorithm to calculate the vector switching in an ordered manner, SVM qualifies as a technique with controlled switching frequency. In DPC (and DTC) however, the output vector could change at any time when a threshold in a variable such as active power, reactive power (or torque and flux for DTC) or capacitors balance is reached, according to the switching table design. This means that although a desired value might be reached faster, it could be well exceeded or provoke another value to change faster. In other words, more switching can take place, and not precisely in an efficient manner. This is where the importance of the hysteresis and switching table design in DPC lies.

SVM also allows greater liberty when selecting vectors, making this technique to switch in a more ordered and calculated manner, so it could be used for implementations with very demanding or exigent requirements. Thus, SVM features better performance by reducing torque ripple and increasing efficiency, but due to their more complex algorithms, they are more complicated to implement and often require high computational frequencies [30]. Generally, DPC and similar techniques offer a simple and rapid real-time implementation, but they have an uncontrolled switching frequency. Hence, the literature has been focused in offering hybrid techniques that could potentially offer the best of both [31].

2.4 Power and Grid Connection

Naturally, inverters can be connected to any AC working motor, grid or device and thus the application will govern the requirements of the converter. For some inverter applications, the phase-angle synchronisation of the inverter's output voltage and current may not be a determining factor for its adequate operation, as shown in Figure 2.9. For example, a speed control for electric motors or an off-grid solar inverter.

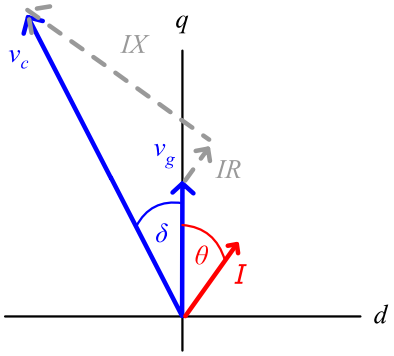


FIGURE 2.9: Voltage vectors in the *dq* frame

However, control of the voltage and current angle is key for adequate operation in other applications such as uninterruptible power supplies, HVDC power transmission and other grid-tied systems such as Vehicle-to-Grid (V2G) and Home-to-Grid (H2G). When the energy has to be transferred to the grid, such as in a vehicle discharging application, the main concern in the AC side is to transfer energy suitable for immediate consumption in the AC grid with minimum distortion and maximum efficiency according to relevant standards.

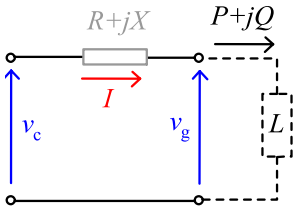


FIGURE 2.10: A simple electrical model of two voltage sources.

When an inverter is connected to the grid, it is the case that the phase of the voltage and current vectors have to be carefully aligned with the grid voltage vector in order to control the flow of power. According to circuit theory, when there are two sources as in Figure 2.10, the relation of the voltages is given by

$$v_c = v_g + (R + jX)I, \tag{2.40}$$

where R is the resistance, and jX is the reactance of the transmission line. The load complex power can also be expressed as $P+jQ$. Here is where the importance of circuit design (i.e. inductance and resistance values) can be appreciated. The magnitude of the voltage vectors depends on Q and according to [32], the angle of transmission δ depends on P.

This work presents an evaluation on advantages and drawbacks of DPC and carrier based techniques such as SVM and PI controllers. These switching techniques will be explored to keep interconnection with the grid healthy while keeping the inverter in operable conditions.

Chapter 3

SVPWM and Model Evaluation of a Single-Phase NPC Inverter

Single-phase three-level NPC inverters are useful for applications in residential areas, where a DC source is available, such as Photo Voltaic (PV) source or Vehicle-to-Home (V2H) systems. This chapter proposes a modeling approach, namely the state-space method, for this type of inverter with SVPWM. The state-space method is commonly used for modelling of dynamical systems. It is convenient for use in time domain analysis of the dynamics of an inverter system when the load changes and for the synthesis of advanced controllers. Since SVPWM is based on a vector approach, a state-space model and a control scheme is proposed, successfully keeping the capacitors' balance. Simulation results show the inverter operation, the capacitors' voltage response and the output voltage.

This Chapter was published in a conference paper [33], and was part of early work before change of supervisor.

3.1 Introduction

As energy consumers move towards home energy generation and storage systems, the demand for efficient and reliable DC-to-AC conversion increases. The same is true for energy providers at large scale, moving towards the smart grid. Low power DC–AC inverters with ratings of around 5 [kW] are usually enough for a home power system, converting DC to AC from PV (or solar) and battery sources. Higher power requirements supposes a larger scale of solar and battery capacity, with power of up to hundreds of megawatts for immediate grid consumption.

From the consumer perspective, an inverter in home energy systems could feed into grid to sell excess energy or provide energy at home. Moreover, vehicle batteries could also operate on-grid (V2G) or off-grid at the consumer's home (V2H). Similarly, the opposite energy conversion (AC-to-DC) is needed when charging batteries. If the consumer chooses, installation of an inverter is possible by using the most widely available single–phase infrastructure. Commercial single-phase inverters are usually PV inverters. *Fronius Primo* is a single-phase inverter rated at 3–8.2 [kW] with up to 98% efficiency, while its *UL* version operates a 4–15 [kW] range with 95% efficiency [34]. *Vincotech* manufactures three-level NPC modules (on a per phase basis) and its variants are used for applications of up to 2.4 [kV] dc and 1.8 [kA] [35].

Multilevel inverters cover a wide range of applications, from motor control, through HVDC transmission, to Uninterruptible Power Supplies (UPS) [12], including home and industrial energy systems. The NPC converter is a multilevel inverter, presenting a closer approximation to a sinusoidal voltage, which reducing harmonic distortion. Also, it has the capability to connect the earth-neutral of the grid to the middle of the DC-link (NP), thus eliminating common mode voltage and producing nearly zero leakage current [36]. It has been reported that *Danfoss* NPC inverters in the *TripleLynx series* can handle 10–15 [kW] and reach up to

98% efficiency. Other industrial NPC inverters are manufactured by *ABB* (*ACS 1000* with 0.3–5 [MVA]) and *Siemens* (*Sinamixs SM150* with 5–28 [MVA]) [13].

NPC inverters have been heavily investigated: modified topologies [37], incorporation of new devices [38]; and different control techniques have appeared in the search for higher quality output power in the most efficient manner. Space-Vector Pulse Width Modulation (SVPWM) and its variants have been shown to be very versatile for single-phase and three-phase converters due to the reduced number of commutations during switching patterns [39], which reduces switching losses of the inverter [40].

However, in the search for an illustrative and didactic approach, an state–space is chosen for its ease in dynamic system analysis and practicallity to implement advanced controllers. Much of the literature regarding power transfer to the grid is related to PV and wind turbine systems and it is often a three–phase type of NPC inverter, which requires Park and Clarke transformations for power calculations.

In this work [41], a full converter state-space model is developed and a sliding mode controller based on space vector modulation is presented. Active and reactive power is controlled without an external voltage controller involved in a three–phase NPC inverter with an L-filter. More works on three–phase with a similar state–space approach are found in [42] where load current harmonics are compensated, and few others in [43, 44].

For the single–phase inverter, the work in [18] proposes a state–space approach to study the dynamic effects of deadtime in a single–phase NPC rectifier. A deadtime is introduced to prevent dc-link short circuit. However, there is no load change (i.e. the load was assumed constant) in this work. The authors in [45] propose different controllers for a single–phase NPC inverter with an L-filter, but they do not analyse the dynamics of the different parts of the inverter.

This chapter focuses on a low power NPC inverter or consumer side by developing the theory behind the SVPWM and capacitor balancing and at the same time illustrating the dynamics of the different parts of the model. The state-space approach is a well known modelling tool and it has been used in several converter configurations. However, such approach has not been reported and implemented for a single–phase NPC inverter with an LC filter, which additionally, allows a simple control for capacitor balance.

The performance of the proposed scheme is tested against a standard circuit model under different capacitor conditions and load change. This work is restricted to the control of amplitude and phase of the output voltage and includes an insight for a power control extension.

3.2 Modulation Technique

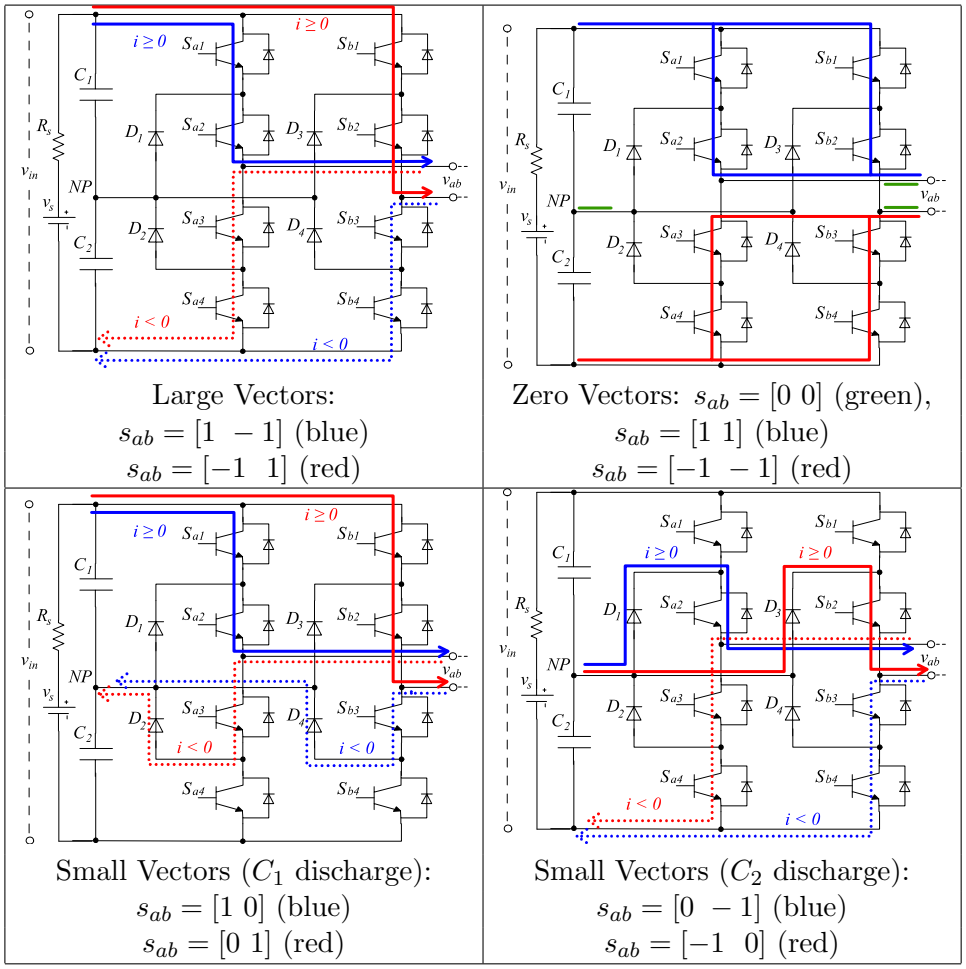
The NPC inverter presented in Figure 2.4 comprises three main subsystems as shown in section 2.2, where the control input is defined by

$$s_{ab} = \begin{bmatrix} s_a & s_b \end{bmatrix} = \begin{bmatrix} S_{a1} & S_{a2} & S_{a3} & S_{a4} \\ S_{b1} & S_{b2} & S_{b3} & S_{b4} \end{bmatrix}^{\top} . \tag{3.1}$$

If all allowed combinations for the two legs within $s_x \in \{1, 0, -1\}$ are used, this inverter provides a total number of operation states of $3^2 = 9$. To understand what occurs with all the operation states in Figure 2.4 for all allowed s_{ab} inputs, a set of figures in Table 3.1 illustrate the conduction path for each case. From the top left figure and clockwise, the control input is: using full DC voltage, producing no output voltage, using half-voltage discharging C_2, and using half-voltage discharging C_1, respectively.

Assuming the capacitors are balanced, Table 3.2 show all possible operation states by naming the vectors illustrated in Table 3.1. These vectors can in turn be classified into three vector types: zero vectors (ZV) where $v_{ab} = 0$, small vectors (SV) where $v_{ab} = \pm v_{in}/2$, and large vectors (LV) make $v_{ab} = \pm v_{in}$. Henceforth, the modulation technique is developed in the same state-space framework by using the base model in equations (2.9)–(2.20).

TABLE 3.1: Simplified operation modes for the single-phase NPC inverter.



3.2.1 Space-Vector PWM

Let a vector v_{ref} be the sinusoidal voltage reference to follow as shown in Figure 3.1(a) with $+v_p$ and $-v_p$ as the positive and negative peaks of the reference voltage, respectively. The output v_{ab} of the inverter is limited to the values 0, $\pm v_{in}/2$ and

TABLE 3.2: Operation states with corresponding idealized voltages (balanced capacitors) for a single-phase NPC inverter.

vector	s_{ab}	v_{aNP}	v_{bNP}	v_{ab}	SV	ZV	LV
v_{z+}	[1, 1]	$v_{in}/2$	$v_{in}/2$	0		■	
v_{1+}	[1, 0]	$v_{in}/2$	0	$v_{in}/2$	■		
v_{L+}	[1, -1]	$v_{in}/2$	$-v_{in}/2$	v_{in}			■
v_{1-}	[0, 1]	0	$v_{in}/2$	$-v_{in}/2$	■		
v_z	[0, 0]	0	0	0		■	
v_{2+}	[0, -1]	0	$-v_{in}/2$	$v_{in}/2$	■		
v_{L-}	[-1, 1]	$-v_{in}/2$	$v_{in}/2$	$-v_{in}$			■
v_{2-}	[-1, 0]	$-v_{in}/2$	0	$-v_{in}/2$	■		
v_{z-}	[-1, -1]	$-v_{in}/2$	$-v_{in}/2$	0		■	

$\pm v_{in}$, which are selected in pairs according to the region where v_{ref} is located. A vectorised view in Figure 3.1(b) shows v_{ref} moving along the vertical axis and the vectors (or control inputs) as points located at the height of the output voltage v_{ab} they provide. All mid-voltage points ($\pm v_{in}/2$) are in a shaded grey background and discharge one of the capacitors at a time, either C_1 for left vectors or C_2 for vectors on the right. Notice that the vectors also correspond to those in Tables 3.1–3.2.

Given that the maximum absolute value of v_{ab} is $|v_{in}|$, in order to produce an average of v_{ref} throughout a full cycle, the following Master Duty Cycle is defined:

$$DC_M = \left| \frac{v_{ref}}{v_{in}} \right|, \tag{3.2}$$

with the constraint

$$| v_p | < | v_{in} | . \tag{3.3}$$

This Master Duty Cycle is shown in Figure 3.1(c) and its value will govern the inverter modulation. However, each region can only switch between a specific set of vectors and needs to be modulated in terms of DC_M. Given an actual value of v_{ref} at any point in time, the converter must be designed to output an average of

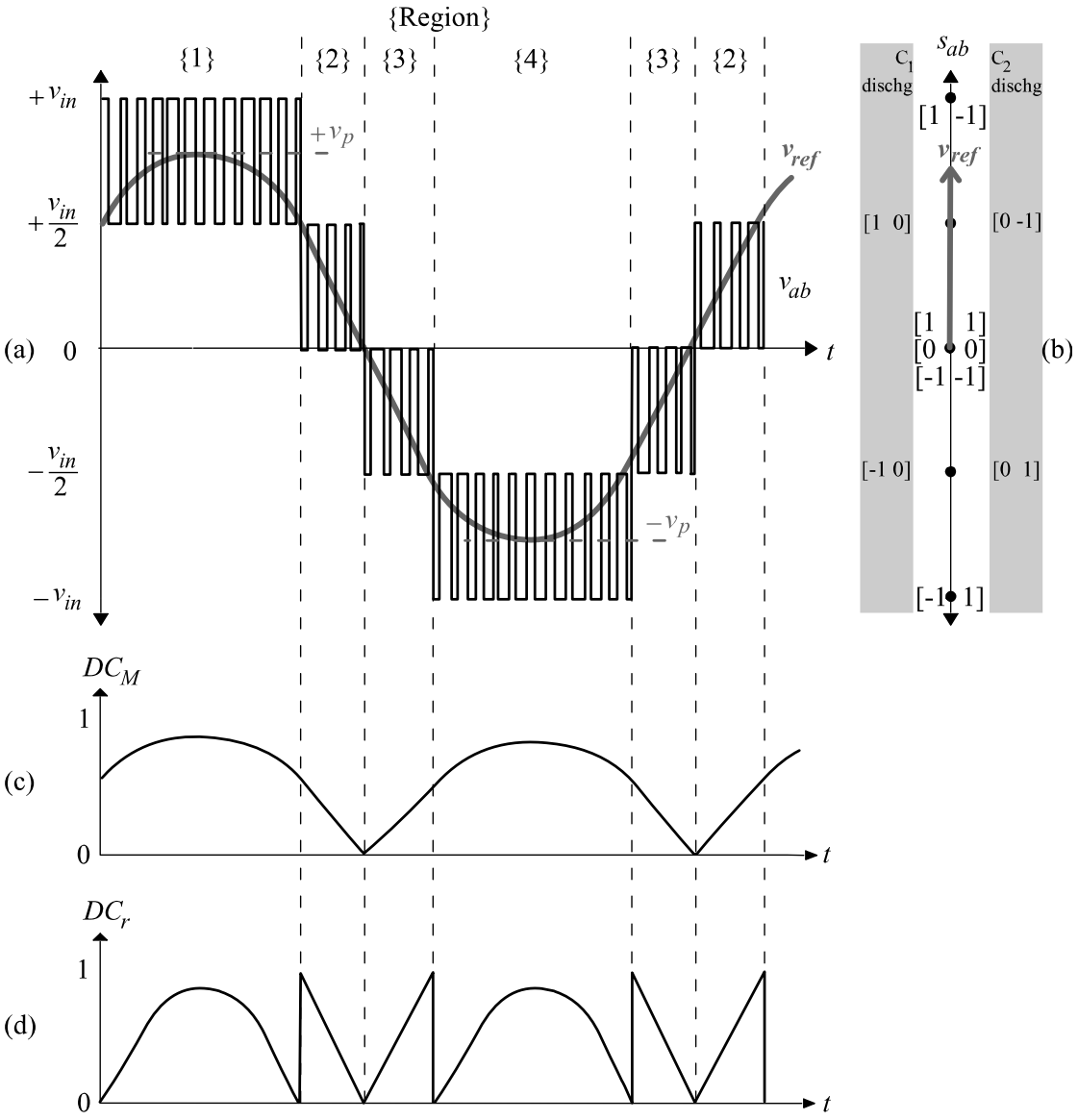


FIGURE 3.1: Representative operation of the SVPWM: (a) v_{ref} with corresponding switching pattern at constant v_{in}, (b) vectorised form, (c) Master Duty Cycle and (d) Regional Duty Cycle.

that value with the set of vectors available in the current region. In general,

$$\frac{v_x t_x + v_y t_y}{2} = v_{ref} T_s, \tag{3.4}$$

where the sampling time is defined as

$$T_s = t_x + t_y, \tag{3.5}$$

with v_x and v_y any pair of different vectors turned on for a time t_x and t_y, respectively and consecutively. These equations are illustrated in Figure 3.2(a) and need to be satisfied to make the average of v_{ab} as close as possible to v_{ref}.

For example, in region 1 of Figure 3.1(a), an average of v_{ref} is made out of voltage levels $+v_{in}$ and $+v_{in}/2$, which correspond to input vectors

$$s_{ab} = \{[1\ \ -1], [1\ 0], [0\ -1]\} = \{v_{L+}, v_{1+}, v_{2+}\}, \tag{3.6}$$

as shown in Tables 3.1–3.2. To make the average, vector v_{L+} will be active for some time and either vector v_{1+} or vector v_{2+} will be activated next, so that equation (3.4) becomes

$$\frac{v_{L+}t_x + v_{1+}t_y}{2} = v_{ref}T_s \quad \text{or} \quad \frac{v_{L+}t_x + v_{2+}t_y}{2} = v_{ref}T_s, \tag{3.7}$$

depending on which capacitor is more convenient to discharge. More specifically, transistors S_{a2} and S_{b3} are kept on in region 1, while in leg A S_{a1} and S_{a3} switch complimentarily (i.e. $S_{a1} = \overline{S_{a3}}$), and in leg B S_{b2} and S_{b4} switch complimentarily (i.e. $S_{b2} = \overline{S_{b4}}$). It could then be said, the duty cycle of region 1 is that of vector v_{L+}. In the same way, the duty cycles of regions 2, 3 and 4 are those of vectors $v_{1+} \cup v_{2+}$, $v_{1-} \cup v_{2-}$, and v_{L-}, respectively.

An alternative approach to deduct regional duty cycles is to take zero voltage as reference, and mathematically deduct them from equation (3.2) and the number of positive regions, $n_r = 2$. Therefore,

$$DC_r = n_r DC_M - 1 \tag{3.8}$$

is the duty cycle for regions 1 (or vector v_{L+}) and 4 (or vector v_{L-}), while

$$DC_r = n_r DC_M, \tag{3.9}$$

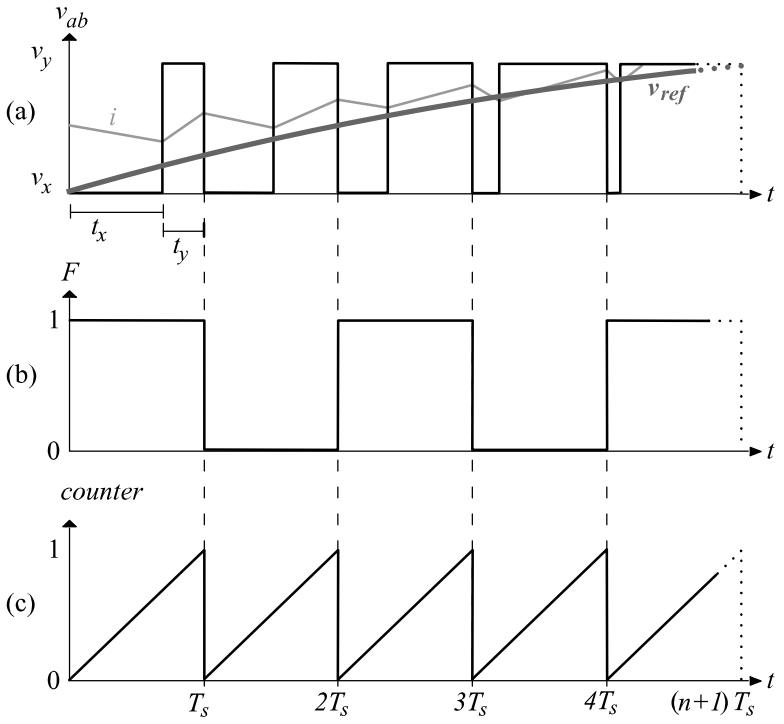


FIGURE 3.2: Fundamental sampling time for region 1.

is the duty cycle for regions 2 (or vectors $v_{1+} \cup v_{2+}$) and 3 (or vectors $v_{1-} \cup v_{2-}$). Thus, depending on the region of the reference voltage v_{ref} and the value of DC_M, only one regional duty cycle DC_r will operate at a time. This pattern is described in Figure 3.1(c)-(d), where DC_M goes from 0 towards 1 and back to 0 for each half cycle, while DC_r changes its value depending on the region.

Additionally, Figure 3.3 illustrates a block diagram of the SVPWM and its dependencies. It requires v_g and v_{in}, which enter a region selector and calculates the Master Duty Cycle DC_M. It also requires timing (or processing time) variables such as sampling time T_s and a capacitor balancing variable F, developed in the next subsection. These four inputs are then sent for Regional Duty Cycle (DC_r) calculation by solving equations (3.4) and (3.8)–(3.9). The output of the SVPWM block are the switching states s_{ab} and its associated v_{ab} and W values corresponding to equations (2.15)–(2.16), and (2.17), respectively.

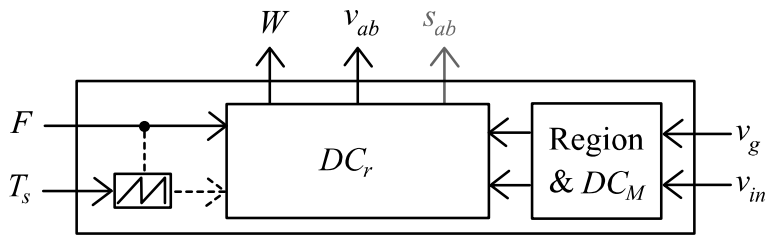


FIGURE 3.3: Block diagram of the SVPWM with F as a fixed pulse function.

3.2.2 Capacitor Balancing

The small vectors, which discharge a particular capacitor, play a key role for operating the inverter. For instance, let us suppose the real value of v_{ref} is operating between 0 and $+v_{in}/2$, which means that the inverter is operating in region 2. Thus, the inverter will keep adjusting its duty cycle while switching between the zero vector and a selected small vector. For the latter, two switching options can be taken, $s_{ab} = [1, 0]$ (discharge C_1) or $s_{ab} = [0, -1]$ (discharge C_2). If the switching pattern for this particular case contains the sequence [1,1] $\rightarrow$ [0,-1], two changes of switching states will be necessary, resulting not only in capacitors unbalance, but also in more switching losses. So, in order to maintain the balance of the capacitors as well as to reduce the number of commutations, the small vectors need to alternate, in this case between both [1,0] and [0,-1]. However, it may take several cycles (of selecting either of the two along with the zero vector) to bring capacitors back to balance, depending on the circumstances.

Assuming the capacitors are exactly the same and that the components of the inverter are ideal, the capacitor voltages can be balanced by alternating the small vectors in such a way that after a vector discharges C_1, a vector that discharges C_2 follows in the next possible option. Therefore, a control variable F is created to flag the small vector which discharges the capacitor that follows. Figure 3.2(b) shows $F = \{0, 1\}$ and it completes a cycle every $2T_s$ from the *counter* in Figure 3.2(c): '1' for $0 \leq t < T_s$ and '0' for $T_s \leq t < 2T_s$. This explains the variable F in previous Figure 3.3.

Table 3.3 is based on a T_s cycle and describes how a control input s_{ab} keeps the capacitor balanced in ideal conditions using F with vector multiplications. Taking region 1 as an example, $v_x = +v_{in}/2$ and $v_y = +v_{in}$ (see Figure 3.2). $F = 1$ during $0 \leq t < T_s$, therefore $s_{ab} = [1\ 0]$ (discharging C_1) during t_x and $s_{ab} = [1\ -1]$ during t_y. Then $F = 0$ during $T_s \leq t < 2T_s$, therefore $s_{ab} = [0\ -1]$ (discharging C_2) during t_x and $s_{ab} = [1\ -1]$ during t_y. Repeat until region 1 is over. Similarly for region 2, 3 and 4 as seen in Table 3.3. Generally, under ideal conditions and equal discharge of capacitors (exactly half the time for each during a v_g cycle), the control input is given by

$$s_{ab} = \begin{cases} v_x & \text{for } t_x \\ v_y(F) + v_y(1-F) & \text{for } t_y. \end{cases} \tag{3.10}$$

with the corresponding v_{ab} in equation (2.17).

TABLE 3.3: The timing of s_{ab}.

Region	s_{ab}	Time-slot
1	$\begin{bmatrix}1 & -1\end{bmatrix}$	t_y
	$\begin{bmatrix}1 & 0\end{bmatrix} F + \begin{bmatrix}0 & -1\end{bmatrix} (1-F)$	t_x
2	$\begin{bmatrix}1 & 0\end{bmatrix} F + \begin{bmatrix}0 & -1\end{bmatrix} (1-F)$	t_y
	$\begin{bmatrix}0 & 0\end{bmatrix}$	t_x
3	$\begin{bmatrix}0 & 0\end{bmatrix}$	t_x
	$\begin{bmatrix}-1 & 0\end{bmatrix} F + \begin{bmatrix}0 & 1\end{bmatrix} (1-F)$	t_y
4	$\begin{bmatrix}-1 & 0\end{bmatrix} F + \begin{bmatrix}0 & 1\end{bmatrix} (1-F)$	t_x
	$\begin{bmatrix}-1 & 1\end{bmatrix}$	t_y

However, these ideal conditions are not usually true in practice, as it is difficult to find components with ideal properties. For instance, hardly any two capacitors with the same specifications are exactly identical to each other. Therefore, assuming voltage measurement of the capacitors is available for each sampling period, a simple error variable is introduced and defined as

$$E_c = v_1 - v_2, \tag{3.11}$$

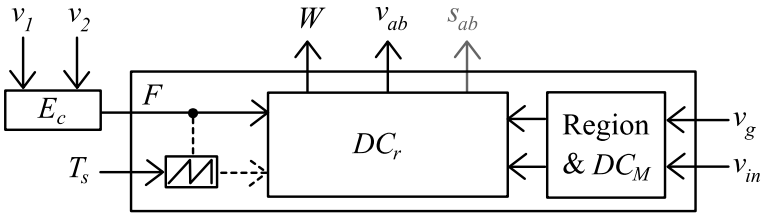


FIGURE 3.4: Block diagram of the SVPWM with F as a voltage difference in eq. (3.12)

.

and the control variable for capacitor balancing becomes

$$F = \begin{cases} 1 & \text{for } E_c \geq 0 \\ 0 & \text{for } E_c < 0. \end{cases} \tag{3.12}$$

As shown in Figure 3.4, now this flag F is used as a simple control variable to keep the capacitors in equilibrium and is directly associated with the timing of the SVPWM explained in the previous subsection. In other words, equation (3.10) still holds, but instead of having F driven by a fixed pulse signal of $2T_s$, it is driven by eq. (3.12).

3.3 Model Verification using Simulation

To verify and test the accuracy of the proposed state-space model representing the inverter, a state-space model (SSM) and a circuit model (CM) were built in *Matlab/Simulink* with Figure 2.4 as a base. The state-space model in equations (2.9)–(2.20) was implemented using a Simulink *S-function* (see *Mathworks* manual [46] for reference) in user-defined functions library. The circuit model, on the other hand, is component-based (as shown in Figure 2.4) and was built using *SimPowerSystems*.

The *S-function* in the SSM was set in continous-mode and is capable of processing several differential equations simultaneously and setting initial conditions. For

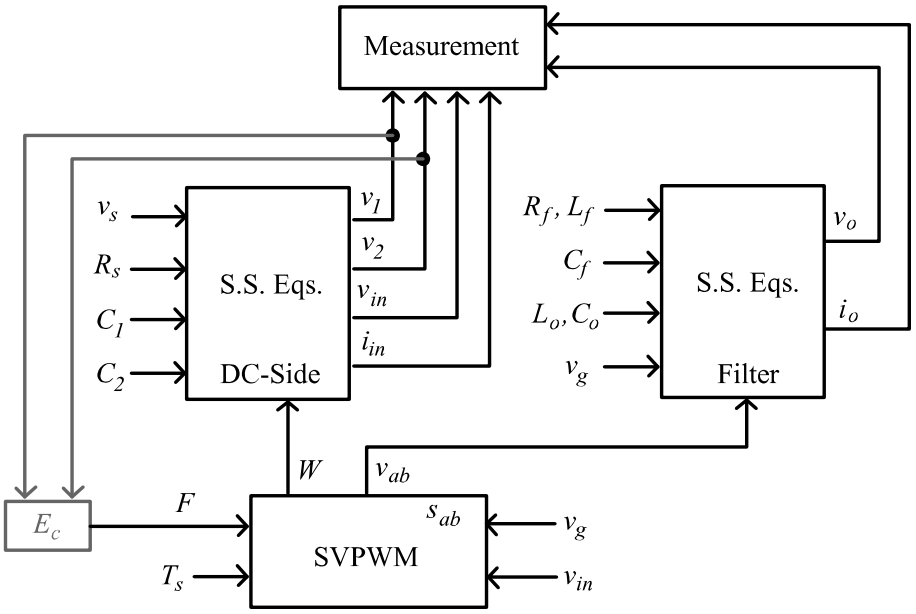


FIGURE 3.5: Block diagram of the state-space model (SSM).

simplicity, Figure 3.5 illustrates the SSM model in two parts: DC-side and AC-side. The DC-side on the left contains equations (2.9)–(2.14) embedded, requiring v_s, R_s, C_1, C_2, and W in order to output v_1, v_2, v_{in}, and i_{in}. The AC-side on the right contains equations (2.18)–(2.20) and requires the filter parameters (R_f, L_f, C_f, L_o, C_o) as well as v_{ab} in order to output v_o and i_o. Note that only v_{ab} and W are required from the SVPWM block (at the bottom) as they are the only values required to solve differential equations on the DC-side and AC-side respectively. The DC_r block, as explained earlier, solves the switched dynamics in a region basis. All outputs of the S.S. equations are available for measurement.

The circuit model is shown in Figure 3.6 and comprises a DC-side, an inverter and a filter. The DC-side has a voltage source v_s and associated resistance R_s connected to a pair of capacitors: C_1 at the top and C_2 at the bottom. The inverter is simply built in software as an arrangement of transistors and diodes as defined for a single–phase NPC inverter. On the filter side, the LCL configuration is shown with the associated resistances. The measurable outputs are exactly the same as the ones in the state-space model and the only input required from the SVPWM block is s_{ab}. The SVPWM algorithm in Section 3.2 was also built in

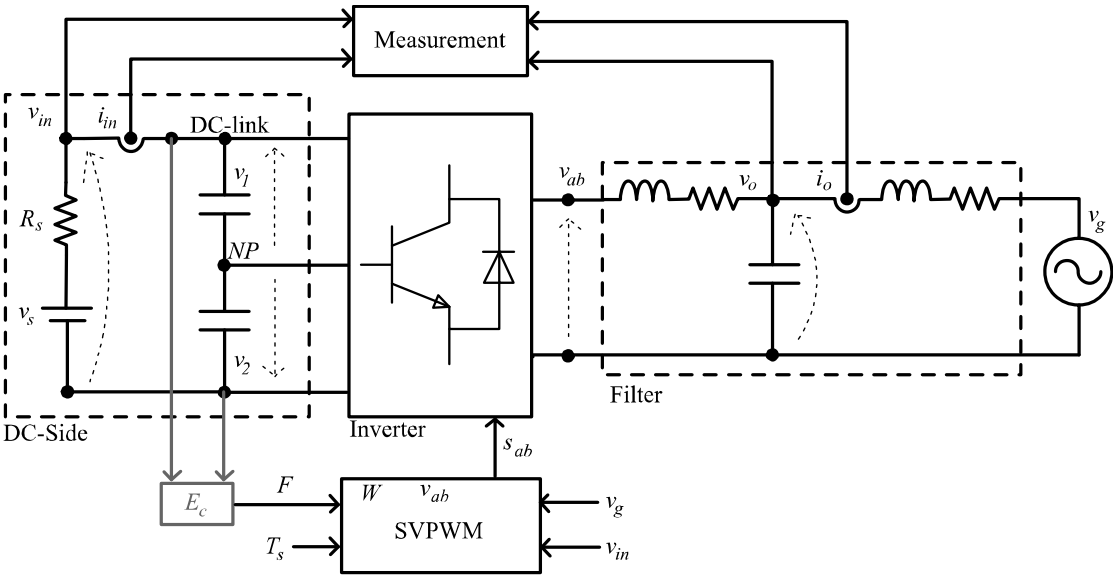


FIGURE 3.6: Block Diagram of the circuit model (CM).

Matlab/Simulink and was used in the two models with the same conditions, as shown in Figures 3.3–3.4.

In the following simulations, two scenarios are considered:

- *Balacing of ideal capacitors.* Assumes equal discharge of two same ideal capacitors where eq. (3.10) is fed by a fixed pulse signal F with a $2T_s$ cycle.
- *Balancing of non-ideal capacitors.* Assumes measurement availability of capacitors voltage in real-time where eq. (3.10) is fed by F, defined as the voltage difference in eq. (3.12).

Block diagrams in Figures 3.5 and 3.6 represent balancing of ideal capacitors in black and balancing of non-ideal capacitors in black and grey (which includes measurements of v_1 and v_2). Both models were implemented in different files with exactly the same SVPWM block and also both simulations were run with solver *ode8 (Dormand-Prince)* at 1×10^{-7} [s] fixed step size. The purpose of this small step size is to obtain high precision readings in order to detect differences between the models. Table 3.4 describes the parameters under which both models were run with the proposed modulation method.

TABLE 3.4: Inverter parameters.

Parameter	Value	Unit
v_s	60	[V]
R_s	1	[Ω]
C_1	390	[μF]
C_2	390	[μF]
R_f	1	[Ω]
L_f	1	[mH]
C_f	1	[nF]
R_o	68	[Ω]
L_o	15	[mH]
$v_g = V_{rms}$	$\frac{40}{\sqrt{2}}$	[V]
T_s	10	[kHz]
f	60	[Hz]

3.3.1 Balancing of Ideal Capacitors

Initial conditions of the capacitor voltages, v_1 and v_2, are 0 [V] and their response is shown in Figures 3.7–3.8, respectively. Capacitor voltage v_1 is measured from the neutral point NP to $+v_{in}$ while capacitor voltage v_2 is measured from $-v_{in}$ to the neutral point NP. The voltages are shown in red for SSM and in blue for CM (which usually overlaps the SSM red line). The capacitor voltages are reduced most at 4.5 [ms] and 13 [ms], which is when the sinusoidal output voltage v_o is at the peak. In the closer views located at the bottom of each of the figures, continuous increase and decrease of the voltages show the switching nature of the system. Figures 3.7–3.8 look identical due to both models assuming both capacitors are identical and have the same discharge response.

It is also observed in the same figures that capacitor voltages coincide in the first part of the cycle (i.e. $0 < t < 8$ [ms]), while in the second part ($8 < t < 16$ [ms]) seems to be non-synchronous. This could be caused by the dynamics of W interacting with the mathematical model compared to the CM. The CM includes a more detailed switch model with voltage drop, gate capacitance and more, that could explain the difference. However, the difference of the average

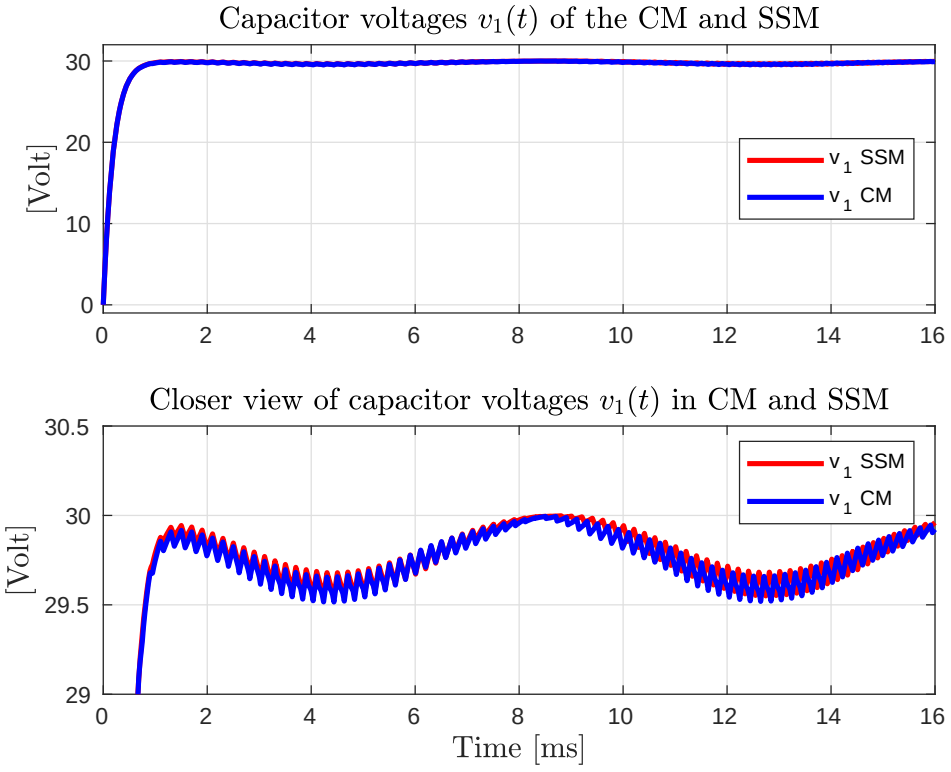


FIGURE 3.7: Capacitor balance v_1.

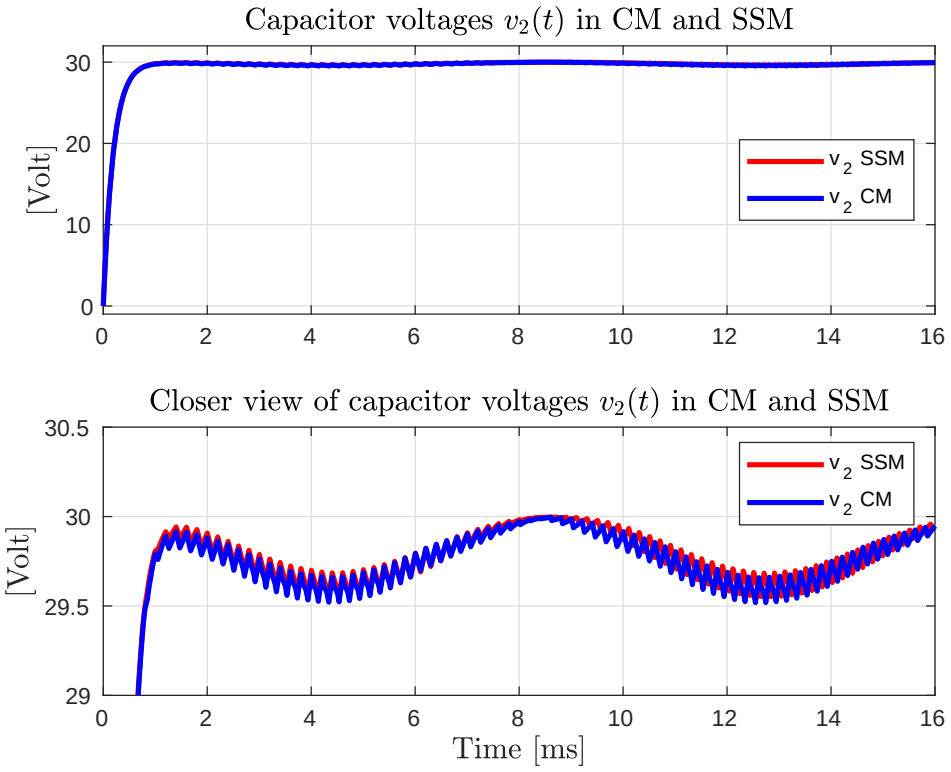


FIGURE 3.8: Capacitor balance v_2.

value of the capacitors voltage (v_1 or v_2) in SSM and CM is close to zero, and respond to the control algorithm in a balanced manner. Although these figures show the capacitors charging time to be approximately 1 [ms], the initial charging of the capacitors is not of particular interest, as normally there is some pre-charge mechanism in place to avoid this high initial current.

Figures 3.9–3.10 show the terminal voltage v_{in} in continuous line and current i_{in} in dashed line. The lines in red correspond to the SSM and the lines in blue correspond to the CM. v_{in} and i_{in} in these two figures present a similar wave pattern to the one shown in the capacitor voltages. Since v_{in} in both models are equal to $v_1 + v_2$, the input voltages present a similar wave pattern as v_1 and v_2. Every time there is a decrease (or increase) in input voltage v_{in}, the current i_{in} increases (or decreases). This pattern is caused by R_s and corresponds to the increasing and decreasing values of v_o in the next figure.

The close-up in Figure 3.10 shows a slight variation of v_{in} and i_{in} between both models. In fact, the SSM shows smoother waves of input voltage and current. This could be explained by the assumption of ideal conditions in SSM, such as a capacitor with no energy loss and transistors with no associated resistance or capacitance (i.e. ideal switches). These elements are modelled with higher accuracy in Simulink/Matlab using CM, however, the SSM offers a smooth operation even though it does not know about the non-idealities of various elements.

The comparisons for the output voltage v_o and the output current i_o obtained from the circuit and the state-space models are shown in Figure 3.11. The blue line corresponding to the results of CM mostly overlaps the red line corresponding to the SSM. The output current i_o in the SSM is nearly equal to that of CM. An average of v_o was obtained with a *Mean* block in *Simulink* at 100 [μs] and defined as $v_{g(avg)}$. v_o switches between 0 and 30 [V] when $0 \leq v_{o(avg)} < 30$ [V] and between 30 to 60 [V] when $30 \leq v_{o(avg)} < 60$ [V]. It is noticeable how v_o in the state-space

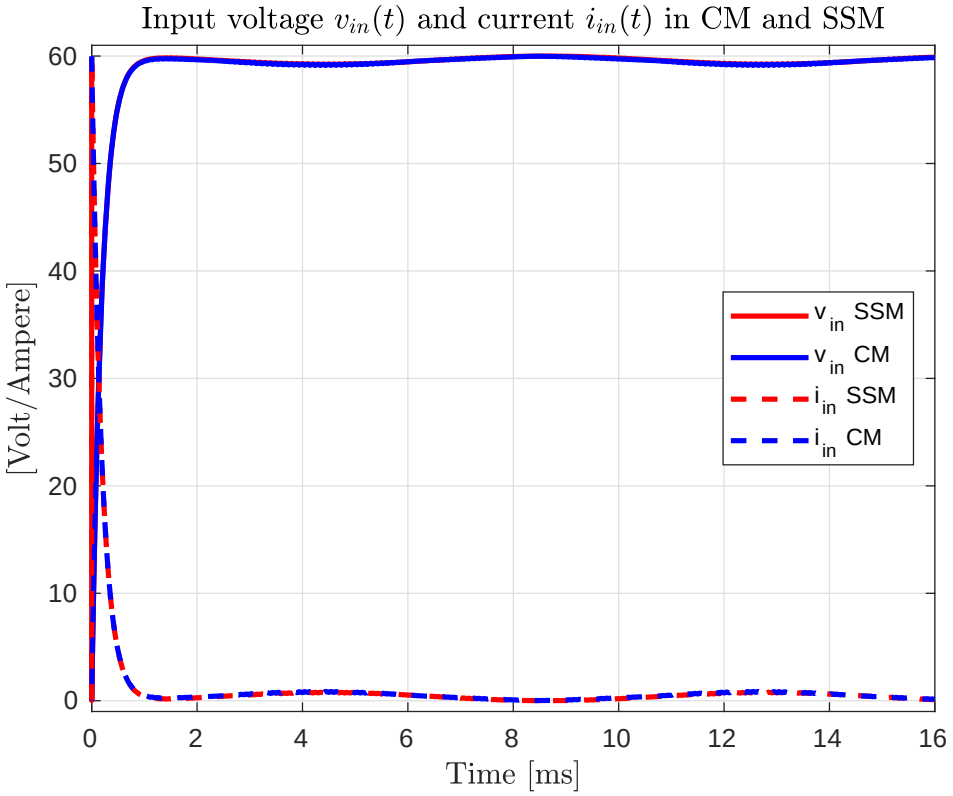


FIGURE 3.9: Terminal voltage v_{in} and current i_{in}.

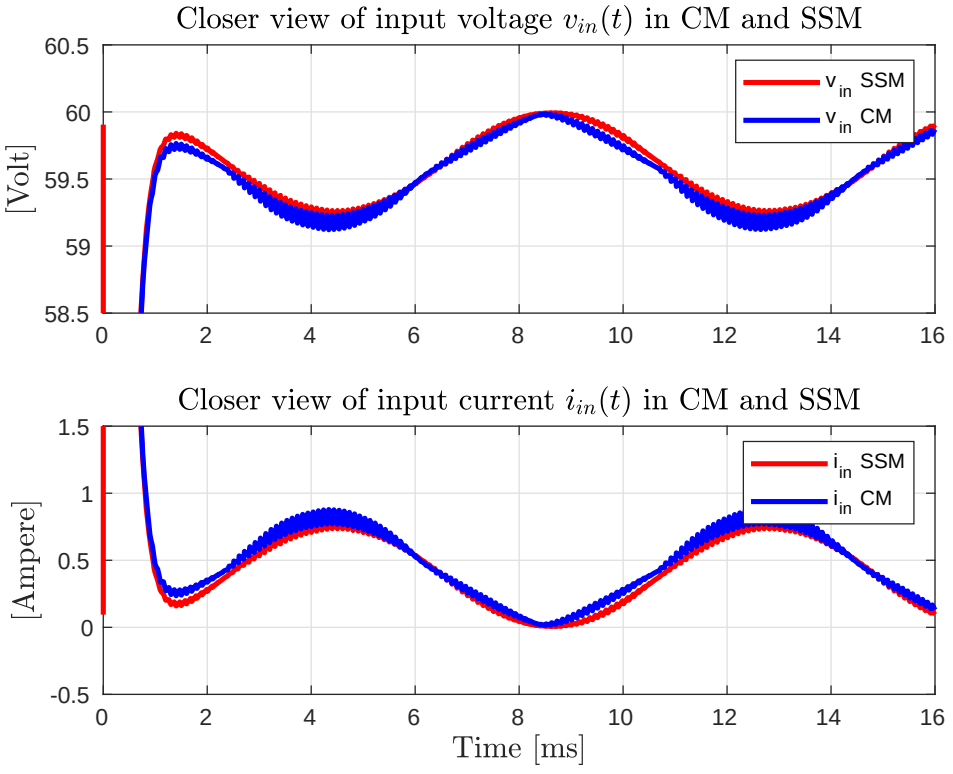


FIGURE 3.10: Closer view of terminal voltage v_{in} and current i_{in}.

model has higher peaks than the circuit model. Also v_o and i_o in the circuit model (CM) even starts smoother due to the internal characteristics of the switches. The three measurements i_o, v_o and $v_{o(avg)}$ present a sinusoidal wave corresponding to a grid frequency $f_g = 60$ [Hz].

A closer view is pictured in Figure 3.12 from 0 to 1 [ms]. The middle graph resembles Figure 3.1 due to a pattern in v_o increasing the time in 'high' value (30 [V] in this case) as time passes and $v_{o(avg)}$ increases. In other words, as the regional duty cycle DC_r in region 2 increases, v_o keeps 'high' longer. However, this graph is not exactly as the one in Figure 3.1 and the reason lies in the models used, particularly in the intrinsic features of the transistors. This will be discussed next.

It is appreciated that simulation results of the two models (Figure 3.12) are very similar to each other, but there are slight differences observed in both, particularly at the beginning. The output current i_o in SSM starts with high peak values and i_o in CM only reaches those peak values after about 10 cycles of f_s. It is found a similar performance in the output voltage v_o; in SSM it starts with high values and in CM it slowly reaches them. In the latter, it is also appreciated a more pronounced damping both, in the high voltage level and the low voltage level (0 [V] and 30 [V] in this time-window, respectively). This phenomena is due to the component-based model (CM) having more complete switching characteristics (such as resistance) while the mathematical model (SSM) considers the switches as ideal. The average output voltage $v_{o(avg)}$ in CM shows more damping, it is seen to be slightly behind the SSM and also paused for a short time when it crosses zero value. This has to do with how $v_{o(avg)}$ is resolved in SSM and the transistors in the CM unable to make a complete turn-on with such a small pulse of s_{ab}. Additionally, recalling the SSM, it obtains v_{ab} and W directly, completely ignoring the dynamics of switching devices.

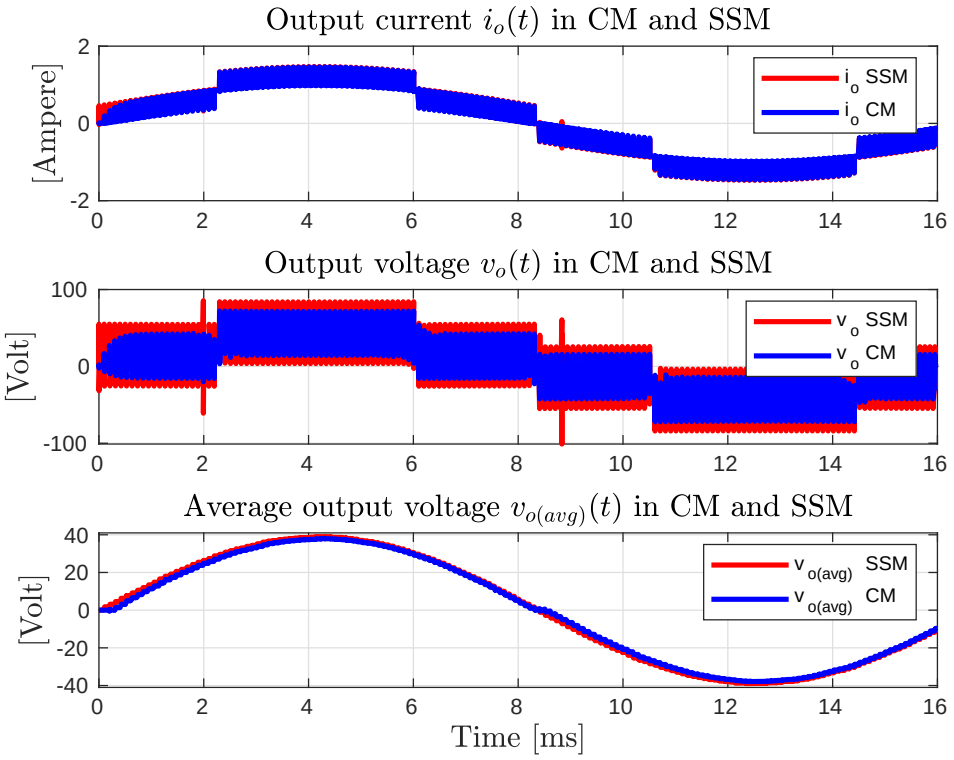


FIGURE 3.11: Output voltage v_o and current i_o.

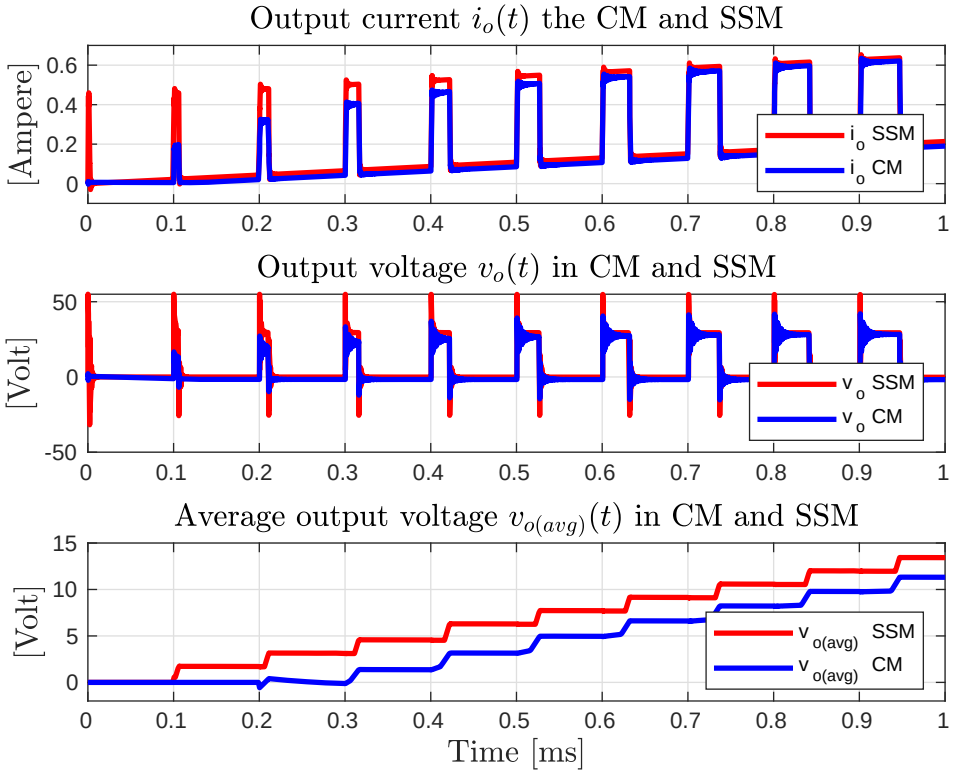


FIGURE 3.12: Closer view of output voltage v_o and current i_o.

3.3.2 Balancing of Non-ideal Capacitors

Previous Figures 3.7 – 3.11 demonstrate a very good agreement between the circuit model and the state-space model, proving the state-space model to be a valid representation of the single–phase NPC inverter proposed. However, the modelling is done assuming exact same ideal capacitors on the DC-side. This is usually not the case in practice, as it is difficult to find two components with ideal properties. For instance, hardly any two capacitors with the same specifications are exactly identical to each other. Therefore, assuming the measurement of capacitor voltages on the DC-side is available for each sampling period to keep the capacitors balance, F will only change value according to (3.11)-(3.12) in the SSM.

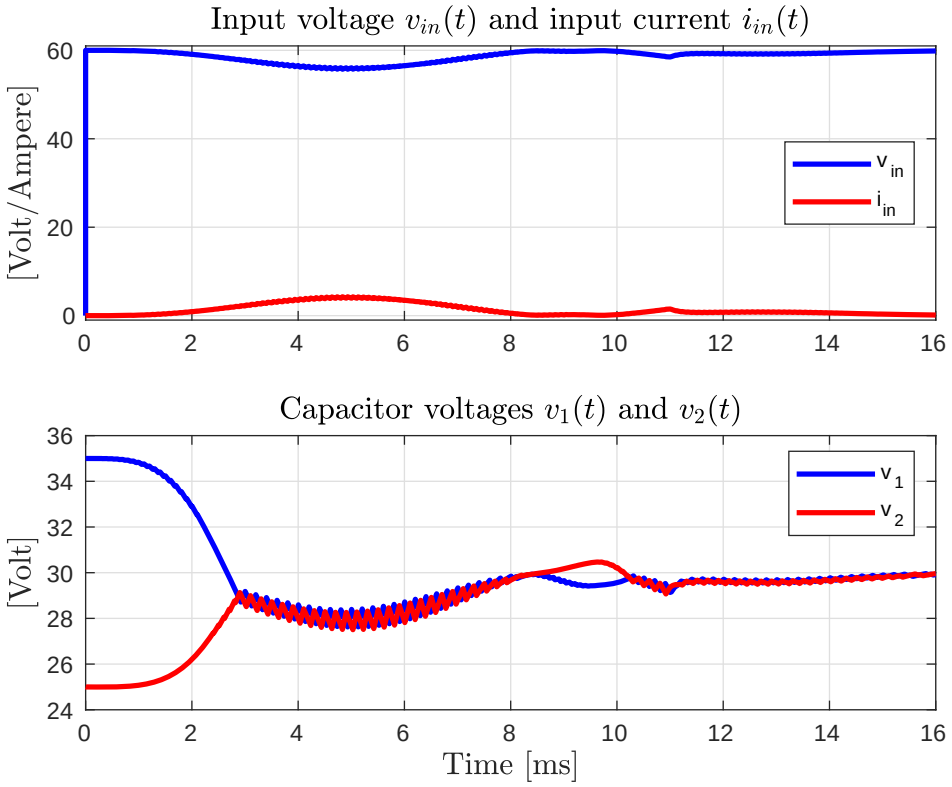


FIGURE 3.13: DC-side voltages and currents for a load change.

In order to test this simple control strategy, initial conditions for the capacitor voltages are set to be significantly different, with $v_1(0) = 35$ [V] and $v_2(0) = 25$ [V]. The values of resistance and inductance are changed in such a way that R_o

$= 10$ [Ω], $L_o = 15[mH]$ for $0 < t < 11$ [ms] and $R_o = 68$ [Ω], $L_o = 15[uH]$ for $11 < t < 16$ [ms].

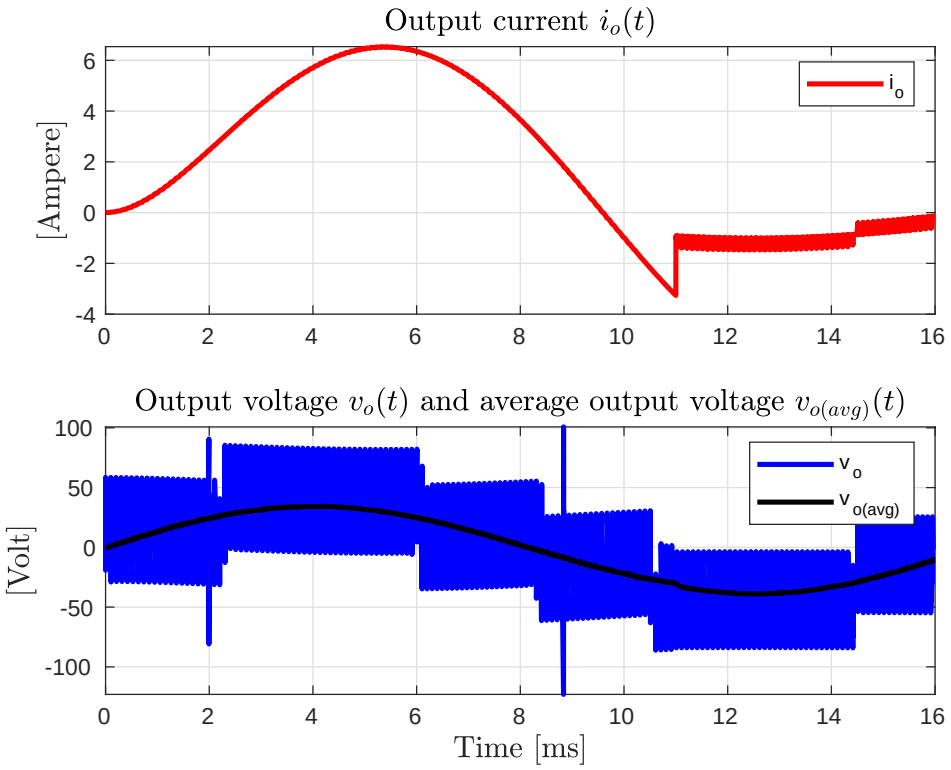


FIGURE 3.14: Output current i_o, and output voltage v_o for a load change.

It is demonstrated in Figures 3.13 and 3.14 how the capacitor voltages are balanced and the output voltage v_o is maintained in spite of both, different initial conditions of capacitors and changes in inductance and load at the output. These figures only show the SSM. On top of Figure 3.13, input voltage and current is shown in blue and red, respectively. They perform in a similar way to the previous results and when the R_oL_o change comes, a spike is noticeable, and the swing of the waveform is reduced. Capacitor voltages start with values of 35 and 25 [V] and then reach a symmetrical and strictly balanced performance just before 3 [ms]. Due to v_o reaching zero, v_{in} keeps maximum and i_{in} keeps minimum at about 8 [ms] and at the same time making v_1 and v_2 slightly unbalanced. These voltages reach balance and symmetrical performance again at 10 [ms]. When the R_oL_o disturbance is introduced at 11 [ms], the capacitor voltages are less stressed and

manage to stay balanced smoothly. The disturbance is also noted in i_o, which is significantly reduced. $v_{o(avg)}$ stays at its expected value.

The current i_o is shown at the top of Figure 3.14 with a large and smooth swing (reaching more than 6 [A] peak) before the change in R_oL_o. After this change, it shows a switching ripple with reduced current. As seen in the bottom of Figure 3.14, the voltage v_o switches in such a way that it averages the black line $v_{o(avg)}$ and the change of R_oL_o is also noticeable.

As a voltage controlled inverter, the algorithm is designed to follow a sinusoidal voltage and therefore v_o must keep its peak-to-peak value and its sinusoidal shape as smooth as possible. For this reason, when the R_oL_o change takes place the resistance increases and the output voltage v_o preserves its value while the output current i_o decreases its value, but preserves its sinusoidal nature. This is expected in accordance to the Ohm law $I = V/R$.

An observation on the balance of the capacitors' voltage is that, for either of the two sets of parameters, it is enough to implement equations (3.11)-(3.12) to achieve the required v_o as long as one of the capacitors is fully charged. For higher currents or more demanding applications it will be important to balance the capacitors first, and possibly to increase their size.

3.4 Grid Connection

Defining the inverter in this Chapter as a voltage controlled one, Figure 3.14 shows a phase difference between the averaged output voltage and the output current. Before there is a load change, $v_{o(avg)}$ crosses zero near 8 [ms], while i_o crosses zero near 10 [ms]. Once the load increases, this phase difference is reduced.

If it was a power controlled inverter shown in a similar figure and assuming a power factor of $PF = 1$, the grid current will be in phase with voltage when the inverter is operating in discharging mode. In charging mode (rectifier), the current and voltage phases will be a 180^o degree out of phase, assuming the positive current direction is that into the grid.

Therefore, as a power controlled inverter, particular attention should be paid to the phase of voltage and current through the grid, which means a circuit to measure the grid voltage would be required (e.g. PLL). If it is assumed R_oL_o is the grid impedance (such as filters, lines, transformer, etc.), the inverter and grid connection could be seen as in Figure 3.15.

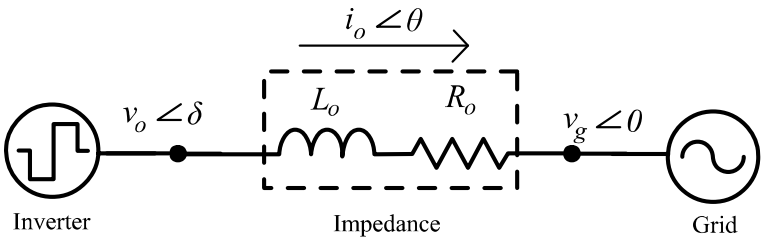


FIGURE 3.15: Power transfer to the grid.

Taking $v_g\angle 0$ as a reference in Figure 3.15, as the angle θ approaches zero, the active power will approach its maximum value. If we can manipulate v_o (or i_o otherwise), it means that we can give v_o the amplitude and phase needed with the SVPWM strategy above in order to obtain the desired power. In this case

$$P = v_o i_o \cos(\delta - \theta), \tag{3.13}$$

$$Q = v_o i_o \sin(\delta - \theta), \tag{3.14}$$

where P is the active power and Q the reactive power.

It has been proven that the amplitude and angle of v_o could be kept as desired even when R_oL_o values change. Therefore, the key to power control lies in making θ controllable (usually via $v_o\angle\delta$).

In other words, the SVPWM scheme with capacitor balancing described in this chapter could be enhanced to achieve power control, most visibly, by setting v_o with the right amplitude and phase. A more detailed vector analysis for power control is presented in the next chapter.

3.5 Summary

In this chapter, a state-space dynamic model of a grid-connected inverter is proposed. The model structure provides the building blocks for modelling the power transfer from a battery to the grid for UPS applications. Simulation results show the response of the inverter with a fixed and variable RL while keeping the capacitors balanced. Furthermore, the model shows the dynamics of the different parts of the inverter (voltages and currents on the DC-side and the AC-side) and how the model interacts with changing parameters. This provides a background about how the SVPWM and the control can be optimized to control power or to further reduce switching losses, or to provide more robustness to the system.

This Chapter presented a voltage controlled NPC inverter, where v_o needs to be kept with an specific angle and magnitude in spite of load changes. On the other hand, this change of load could be seen as a V2H prototype for low loads, where the voltage needs to be kept at a certain level when a high load is changed for a low load. In other words, the output voltage is kept regulated in spite of current changes (in absence of grid connection). However, not all load sizes can be handled since the inverter is limited mainly by its DC voltage source, the current capability in the transistor part, and naturally the size of the load.

If the inverter only handles an external load (i.e. with no other power source in a given power network) it has potential to operate off-grid to feed a home from

solar or battery for example. By incorporating a power control algorithm it could also interact with other power sources in a given network.

Due to *Simulink* providing a circuit model which keeps more of the intrinsic characteristics of the components (transistor resistance and capacitance, capacitor resistances, etc), and the switching nature of the system, the following work will keep an CM approach.

Chapter 4

Direct Power Control of a Grid-tied Three-Phase Inverter

In this Chapter a grid-tied three-phase inverter with an RL filter is evaluated with two different control schemes: PI with Decoupled Control and Direct Power Control (DPC). A vector analysis is presented for voltages, currents, their transformations, as well as for active and reactive power. The PI with decoupled control is described with enough detail to comprehend its dynamics, and simulations results are used to quantify its performance. More detail is provided in the DPC section, where a new switching table is presented and evaluated using simulations. Hysteresis control within the DPC is shown to directly affect the output voltage ripple of this technique. Differences between the two schemes are also discussed.

4.1 Introduction

While a single phase NPC inverter may be preferable for V2G applications due to the availability of the infrastructure, a three-phase inverter is also possible for this and other applications. A three-phase NPC inverter is useful in a wide

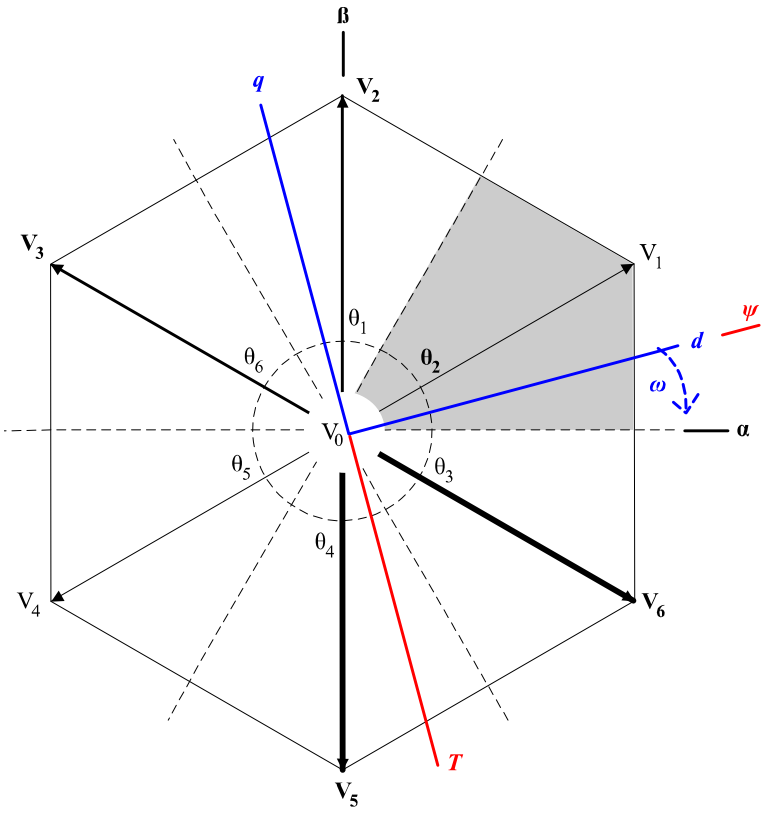


FIGURE 4.1: DTC. Vector map of a two-level inverter with rotating dq axis, flux ψ anchored to the d axis.

range of power transfer applications, either to power a fixed load or a variable load using energy stored in a battery or other DC source. One of the common applications is in the renewable energy sector, where a variable input voltage is available from wind turbines or solar panels and the inverter is required to deal with these variations to feed a load or the grid. For instance, the work in [47] uses a 13 [kW] hybrid converter comprising a boost DC/DC converter with an NPC inverter to transfer energy from a PV array to the grid. The paper in [48] deals with these variations by designing and controlling a 150 [kVA] back-to-back NPC converter interface, while the authors of [49] consider the same interface but rather than the design, they focus on fault detection to avoid distortion of the input current and torque vibration.

Direct Power Control is a strategy that is very similar to, and probably evolved from Direct Torque Control (DTC) that was introduced by Takahashi in 1986.

The latter, aimed at driving an induction motor (IM) using a 2-level three-phase inverter by defining space vectors in the $\alpha\beta$ plane (along with six regions), and selecting them using flux and torque hysteresis according to a pre-designed switching table.

A vector map (also known as vector diagram) of DTC is shown in Figure 4.1 and the way it works is as follows. The vector V_n with $n = 1, 2...6$ represents the output voltage of the inverter and a specific switching pattern. Output voltages corresponding to the vectors are mapped within six regions (θ_1–θ_6) in the dq-frame, limited by dashed lines in the figure. The flux vector ψ is anchored to the d-axis and is turning over the grey area θ_2 at ω speed. A virtual torque vector T (which in reality is based on the torque equation of the machine) is drawn at the negative of the q-axis, perpendicular to the flux. When the dq-frame turns anti-clockwise (ACW), vectors V_2 and V_3 alternate to provide the flux and torque required. This figure shows the dq frame turning clockwise (CW), where the selected vectors are V_5 and V_6. In any rotation, the vector that "pulls" ψ more in the direction of rotation (towards T direction), provides the greatest torque.

Figure 4.2 provides another perspective of DTC, where vectors V_n are selected within the flux ring or flux band. There exists a fixed value of ψ (in blue) rotating from the center (red dashed line) for which a hysteresis band is set $\Delta\psi$ (in blue). In the same figure, V_6 entered θ_2 until it reached the maximum allowed flux value and then V_5 was selected to decrease the flux. The current value of ψ is now at the minimum allowed and the vector selected next will be V_6 (dashed black).

In a nutshell: when more torque is required, a vector that moves faster towards T direction in the ring is selected; when less torque is required the vector that moves to the outer ring boundary is selected; when torque is between the hysteresis zero vectors are chosen. For the flux hysteresis, simply, a vector towards the inner boundary of the ring is selected when less flux is required, whereas a vector towards

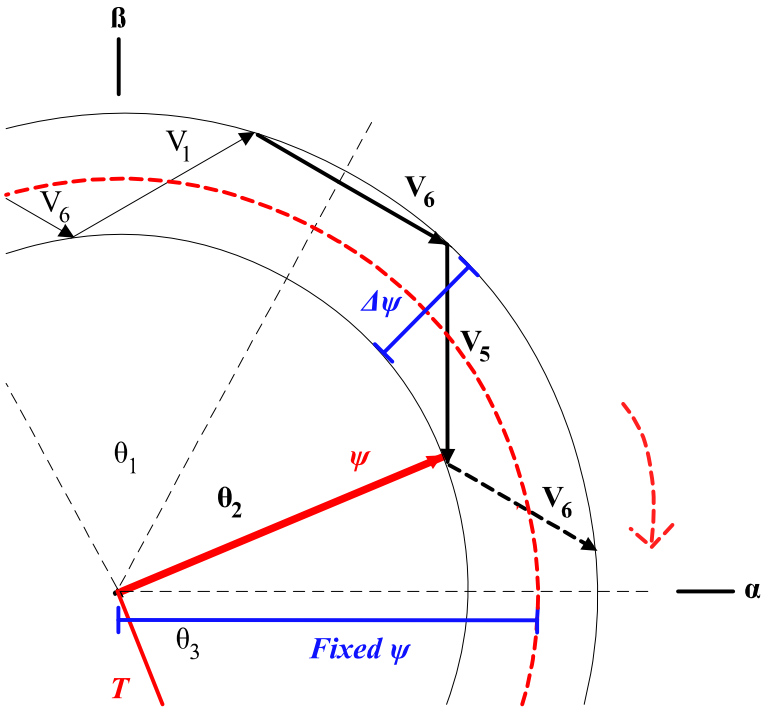


FIGURE 4.2: DTC. Ring-trail view of ψ rotating in the $\alpha\beta$ frame.

the outer boundary of the ring is chosen when more flux is required. This is the reasoning behind DTC as it was published back then. Eventually, the technique increased in sophistication and was fitted to other inverter topologies.

Direct Power Control is used for controlling the active and reactive power instead, but in a similar fashion [26]. In DTC, flux and torque are calculated and the right vectors are applied to increase or decrease them in order to reach the desired flux and torque (or to keep them between desired limits). In DPC, active and reactive power are calculated and the right vectors are applied to increase or decrease them in order to reach the desired active power P_{des}, and desired reactive power Q_{des}. The grid-tied three-phase NPC inverter, in which the power control is applied, is shown in Figure 4.3. Naturally, DPC can also be used for motor control purposes. First, the question of how the vectors need to be selected or how they move in $\alpha\beta$- and dq-frames to provide the desired power needs attention. More details are provided in the following sections.

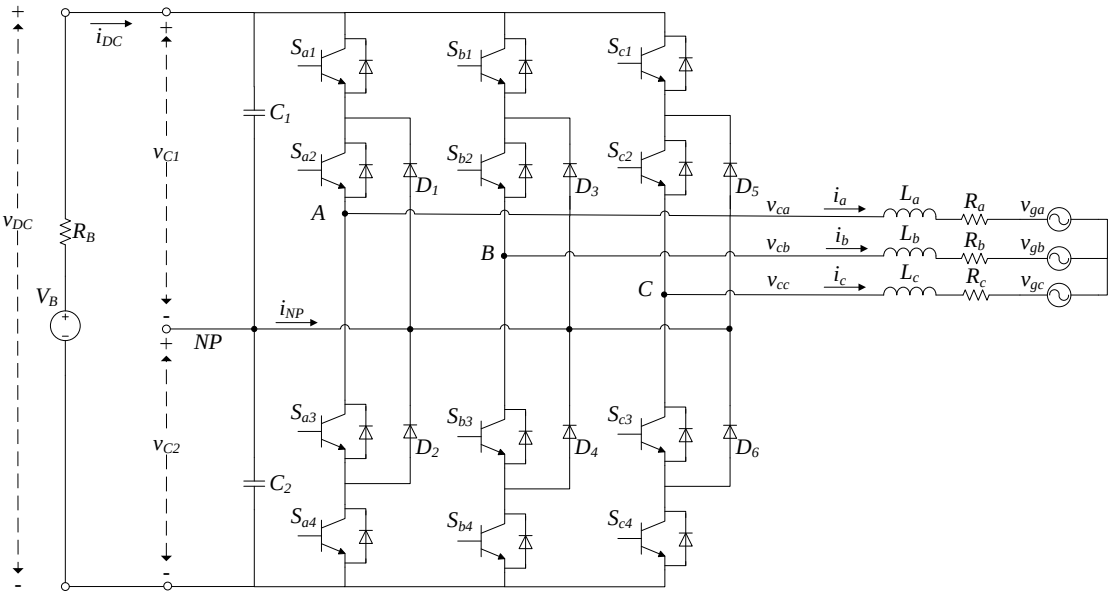


FIGURE 4.3: The three-phase three-level NPC inverter.

4.2 Vector Analysis and Transformations

In this section, circuit analysis and transformations are discussed, considering the capacitors part, transistors part and filter part. The latter could be configured differently, however, in this work the RL type has been chosen. Generally, the connection between the inverter and the grid could be seen as two voltage sources connected through an RL filter, as seen in Figure 4.4 and their relationship is

$$v_c = v_g + (R + jX)i, \tag{4.1}$$

with v_c the converter output voltage (before the RL filter) and v_g the grid voltage at frequency f_g. The key for power transfer from v_c to v_g is to measure v_g and reduce the phase shift θ between i and v_g, as shown in Figure 4.4 and 4.6. Therefore, a phase shift between the voltage and current from the inverter side is desired to make this happen.

Specifically, this power transfer takes place from a three-phase inverter to the three-phase terminals of the grid. v_c is controllable and v_g is not dependent on v_c.

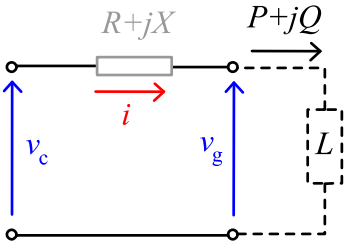


FIGURE 4.4: A simple electrical model of inverter and grid.

Let the three-phase grid voltage v_g be defined by

$$v_{ga} = V_g \cos(\theta_g), \tag{4.2}$$

$$v_{gb} = V_g \cos(\theta_g - \frac{2\pi}{3}), \tag{4.3}$$

$$v_{gc} = V_g \cos(\theta_g + \frac{2\pi}{3}), \tag{4.4}$$

where θ_g is the phase angle from a reference axis and V_g is the magnitude of the grid voltage vector. As the grid voltages are 120° apart from each other all the time during normal grid operation, these voltages can be transformed into the $\alpha-\beta$ frame (also called stationary frame or fixed reference frame) using the Clarke transformation (or $\alpha\beta$ transformation). Due to the different variations present in literature, the Clarke transformation used in this work is illustrated in Figure 4.5 and given by

$$\begin{bmatrix} u_\alpha \\ u_\beta \\ u_0 \end{bmatrix} = \begin{bmatrix} 2/3 & -1/3 & -1/3 \\ 0 & 1/\sqrt{3} & -1/\sqrt{3} \\ 1/3 & 1/3 & 1/3 \end{bmatrix} \begin{bmatrix} u_a \\ u_b \\ u_c \end{bmatrix}, \tag{4.5}$$

where u_x represents any unit (for voltage or current) in the respective frame. In the $\alpha-\beta$ frame, the grid voltage can be now represented with a vector with two components.

This transformation can be applied to any three-variable vector and for this particular case, it is used for measuring grid voltage $v_g \in \mathbb{R}^3$, current $i \in \mathbb{R}^3$ and

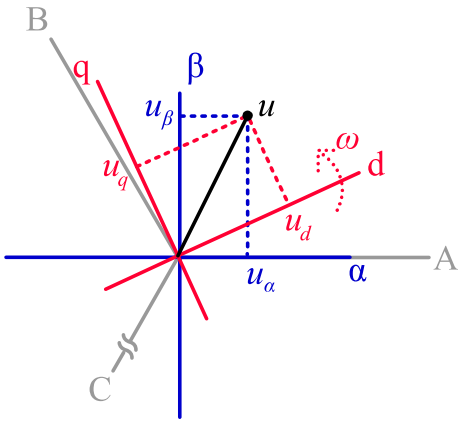


FIGURE 4.5: Transformations in different frames

calculating the converter voltage $v_c \in \mathbb{R}^3$. Now, these three vector sets can be read as $v_{g\alpha\beta}$, $i_{\alpha\beta}$ and $v_{c\alpha\beta}$, respectively. These three vectors will appear with its $\alpha\beta$ components in the $\alpha\beta$ frame of Figure 4.5. In the DPC approach later, it will help us to identify regions in this same plane.

DPC only requires the Clarke transformation, but it could well be implemented using the dq frame (or rotating reference frame) which rotates at constant speed ω in Figure 4.5. The PI with decoupled control also uses the dq frame, for which the following Park transformation (or dq transformation) is used:

$$\begin{bmatrix} u_d \\ u_q \\ u_0 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_\alpha \\ u_\beta \\ u_0 \end{bmatrix}. \tag{4.6}$$

Once the information of the current and voltage vectors is in the $\alpha\beta$ or the dq form, it is then processed by the control algorithm used. After this, it is often required to transform the resulting voltage output back to the abc frame. It becomes then necessary to define the inverse Park transformation as

$$\begin{bmatrix} u_\alpha \\ u_\beta \\ u_0 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_d \\ u_q \\ u_0 \end{bmatrix}, \tag{4.7}$$

and the inverse Clarke transformation as

$$\begin{bmatrix} u_a \\ u_b \\ u_c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ -1/2 & \sqrt{3}/2 & 1 \\ -1/2 & -\sqrt{3}/2 & 1 \end{bmatrix} \begin{bmatrix} u_\alpha \\ u_\beta \\ u_0 \end{bmatrix}. \tag{4.8}$$

Either in the $\alpha - \beta$ frame or the dq frame, three vectors can be appreciated: inverter voltage v_c, grid voltage v_g and current i. If we align the grid voltage v_g to the d axis as shown in Figure 4.6, we can define the angle with the inverter voltage v_c as δ. The angle between the current i and the grid voltage v_g is then defined as θ. As long as the grid voltage vector v_g is aligned to the d axis (i.e. $v_q = 0$), the power can be calculated either in the $\alpha - \beta$ frame or the dq frame as follows:

$$P = v_\alpha i_\alpha + v_\beta i_\beta (3/2) = v_d i_d (3/2) \tag{4.9}$$

$$Q = v_\beta i_\alpha - v_\alpha i_\beta (3/2) = -v_d i_q (3/2). \tag{4.10}$$

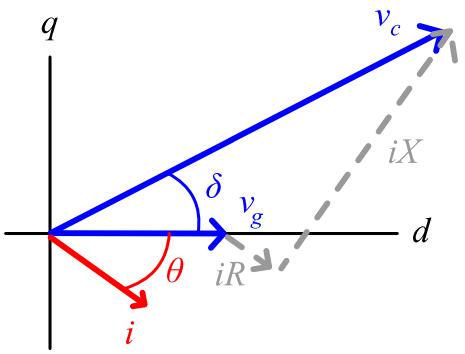


FIGURE 4.6: Frame arrangement with v_g, i and v_c.

The grid voltage is used as a reference for the transformations and the notations v_{gx}, v_{gxx} and v_{gxxx} correspond to the grid voltages in their respective frames, where $x = \{a, b, c, \alpha, \beta, d, q\}$. The same type of notation applies to the converter voltage v_c. As current is shared, the only subscripts are to specify the frame variable or variables. Due to the handling of two voltage vectors, and the slight variation of the grid vector frequency, a PLL is required for connection into the transformation blocks.

4.3 Phase Locked Loop

The electrical grid voltage expressions shown in (4.2)–(4.4), present values of magnitude, phase-angle and frequency. These electrical grid variables are not entirely constant as the grid is a real-time system with continuous connection and disconnection of loads and other disturbances, including faults. This causes grid distortions such as voltage unbalance, line notching, noise, frequency variation, among others. Therefore, in order to achieve successful connection of the power converter to the grid, grid synchronisation becomes fundamental. Quoting Remus Teodorescu (et. al) [36], grid synchronisation

> *[...] of a power converter is nothing but an instantaneous monitoring of the state of the grid to which the power converter is connected. [...] is an adaptive process by means of which an internal reference signal generated by the control algorithm of a grid-connected power converter is brought into line with a particular grid variable, usually the fundamental component of the grid voltage.*

This means that precise information of magnitude and particularly phase-angle of the grid voltages v_g are required to execute the transformations described in the previous section as well as to control active and reactive power injected into the grid.

There are several techniques to synchronise the power converter to the grid, which can be boldly classified in two: Fourier series/transform in the frequency domain, and Phase Locked Loop (PLL) in the time domain.

PLL is the most extended synchronisation method in engineering applications. Several versions of PLLs can be found in literature, according to the algorithm or application requirements. It is also possible to incorporate detection of other

harmonic components and other features using these methods. A detailed review of the different types of PLL is outside the scope of this document: a more detailed explanation on these can be found in [36].

Typically, a PLL is a closed-loop system formed of three blocks: a phase detector, a loop filter and a voltage controlled oscillator. The ultimate goal of a PLL is to track the phase of the electric grid as accurately and cleanly as possible. Due to the analysis of active and reactive power in this three-phase system relying on the $\alpha-\beta$ and dq transformations, it is convenient to use these transformations as tools to phase-lock to the grid voltages. This approach is not new and has been reported in literature for single–phase [50] and three–phase [51], [52] systems. Therefore, a basic PLL with a dq transformation is proposed in Figure 4.7, which is capable of keeping a constant and fast phase tracking.

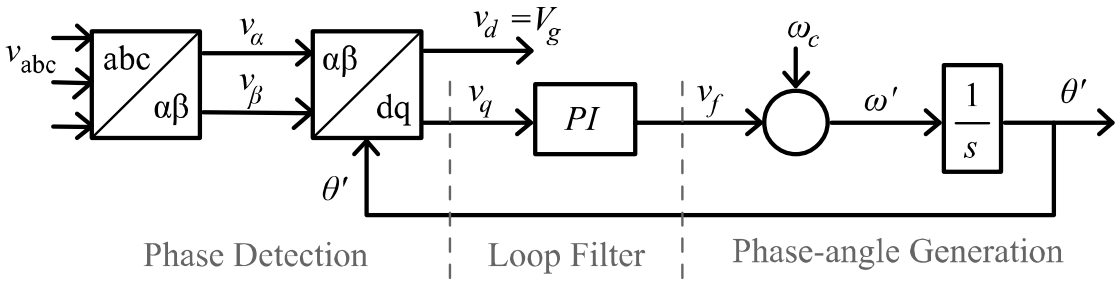


FIGURE 4.7: PLL with dq transform and Loop Filter on the Q axis.

Firstly, the dq transformation could be considered a phase detection block. v_{abc} values are read by the Clarke transformation block, which is equivalent to $u_{\alpha\beta}$ from equation (4.5). As shown in Figure 4.5, v_a is anchored to v_α and if a virtual vector u is assumed to be $v_{g\alpha\beta}$, it would be rotating anti-clockwise (ACW) with time and a variable angle θ_g in the $\alpha\beta$ frame. Now, by introducing a dq frame (or rotating reference frame) as in u_{dq} in equation (4.6), there will be a new set of dq coordinates for v_{gdq} rotating in the same direction with the angle θ on the dq frame. This means that the dq components of v_g can be calculated as follows:

$$v_{gdq} = T_{dq} T_{\alpha\beta} v_{gabc}, \tag{4.11}$$

where $T_{\alpha\beta}$ and T_{dq} are the $u_{\alpha\beta}$ and u_{dq} transformations shown in equations (4.5) and (4.6) respectively, applied to the grid voltage v_g. Expanding and resolving,

$$\begin{bmatrix} v_{gd} \\ v_{gq} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} 2/3 & -1/3 & -1/3 \\ 0 & 1/\sqrt{3} & -1/\sqrt{3} \end{bmatrix} \begin{bmatrix} \cos(\theta_g) \\ \cos(\theta_g - \frac{2\pi}{3}) \\ \cos(\theta_g + \frac{2\pi}{3}) \end{bmatrix} \quad (4.12)$$

$$\begin{bmatrix} v_{gd} \\ v_{gq} \end{bmatrix} = \begin{bmatrix} V_g \cos(\theta - \theta_g) \\ -V_g \sin(\theta - \theta_g) \end{bmatrix}. \quad (4.13)$$

This makes the angle difference $\theta - \theta_g$ constant, and therefore v_d and v_d constants as well. Moreover, back to Figures 4.5 and 4.6, v_d and v_q could govern the active and reactive power, respectively, which will be addressed in a later section.

Secondly, the v_{gq} component is fed into the loop filter, which in this case is a PI controller with the purpose of making $v_{gq} = 0$. The central frequency ω_c is input and the PI controller outputs the estimated frecuency ω' as seen in Figure 4.7. Thirdly, the frequency/phase angle generator takes the place. The integration

$$\theta' = \int \omega dt \quad (4.14)$$

allows to obtain the estimated phase angle.

The phase is then locked when $\theta - \theta_g = 0$ due to feedback of v_{gq} into the PI controller. This means that the PI control sets the voltage in the q-axis to zero. Therefore, on the one hand, the d-axis of the voltage provides the amplitude of the input voltage vector, and on the other hand, the phase-angle detected by the PLL will be in phase with the 'total' voltage vector. Another way of explaining the phase lock is by using the vector magnitude

$$V_g = \sqrt{v_{gd}^2 + v_{gq}^2}, \quad (4.15)$$

and finding that $V_g = v_{gd}$.

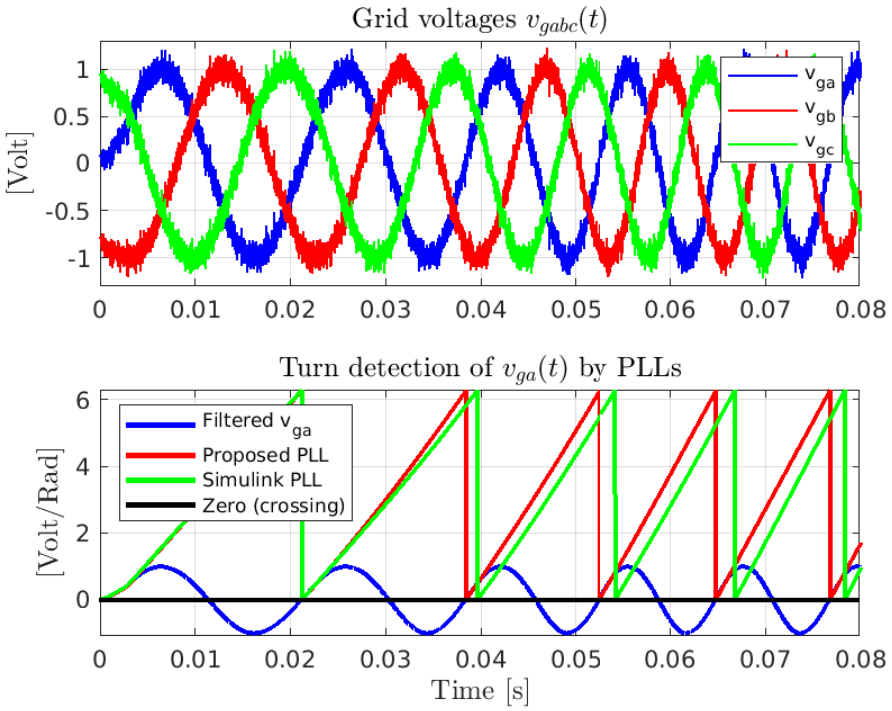


FIGURE 4.8: Phase detection of v_{ga} with different PLL systems.

As an illustrative example, Figure 4.8 presents three 1 [V] amplitude voltages comprising a grid voltage vector v_g, produced by a *Three-Phase Programmable Generator* block in Simulink/Matlab with added noise and starting at 50 [Hz]. A simulation test was devised to observe the speed of phase detection of two PLL systems when v_g increases frequency during $20 < t < 60$ [ms]. At the bottom of Figure 4.8 the filtered v_{ga} signal is shown crossing the zero value. The proposed PLL in Figure 4.7 detects every 'turn' of the vector v_{ga} in the $\alpha\beta$ frame, from zero to 2π [Rad]. Another PLL, *PLL (3ph)* block from Simulink, operates with intentionally deviated gains to illustrate a lack of precision. The Simulink PLL shows a delay of about 2.5 [ms], and although it will eventually detect the phase more precisely at a latter time, this delay could be detrimental for grid-connected applications.

In the example above, both PLL systems produce the required harmonic-free signal, but one is too slow to respond to a frequency variation. Naturally, by tunning the PI gains, the performance can be improved. However, this is to show that the

robustness of a PLL can be demonstrated by the speed and precision at which it can reject harmonics distortion and identify the sequence components of the grid voltages v_{gabc} (or voltage vector v_g).

In further development of this Chapter, it will be assumed a configuration of PLL as shown in Figure 4.7 and therefore, any current vector lying on the d-axis of a grid-synchronised dq frame will deliver active power into the grid, whereas any current vector on the q axis will deliver reactive power.

4.4 PI Decoupling Control

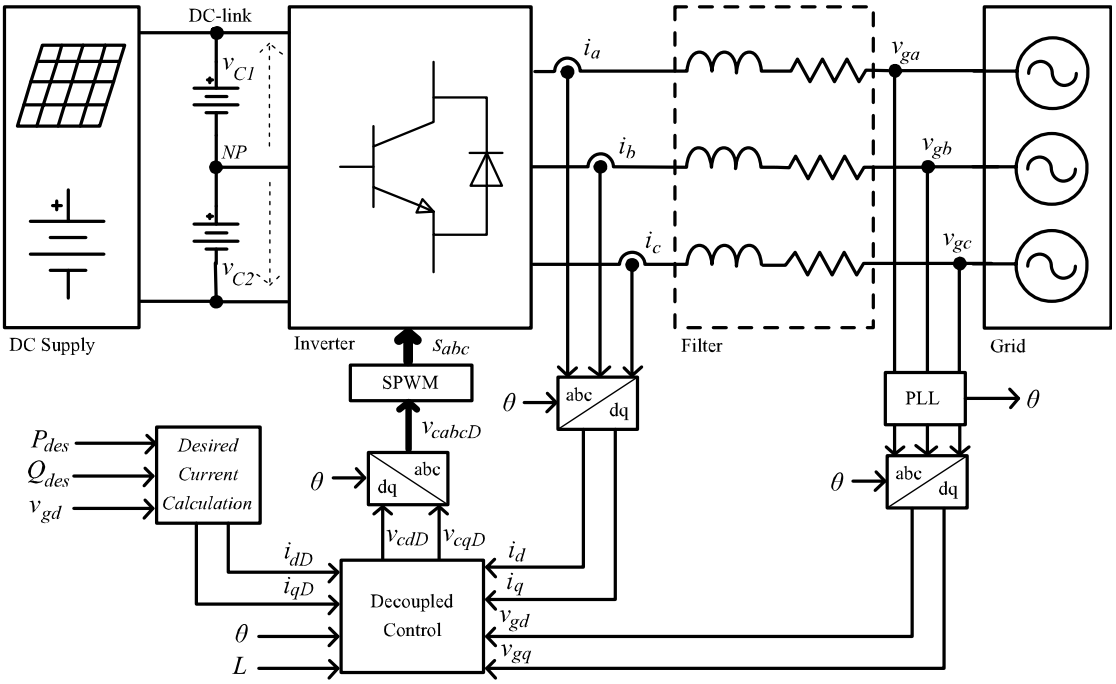


FIGURE 4.9: Illustrative diagram of PI Decoupled Control in NPC Inverter.

PI Decoupling is a controlled-switching frequency technique and is shown in Figure 4.9. A DC-voltage source, or a pair of DC-voltage sources feed the DC-side of the inverter through a DC-link. The inverter outputs PWM voltage and current, which then pass through a filter (in this case an RL filter) and then feeds into the grid. In order to input the correct switching pattern s_{abc} into the inverter, four steps take place:

- Acquisition and transformation: The voltage phase is detected, magnitude of grid voltages and currents are acquired and then transformed into dq frame. The PLL process v_{gdq} and feeds the $abc - dq$ and $dq - abc$ transformation blocks.

- Power control: The desired power, measurements and parameters are read and the desired currents and voltages are calculated.

- Inverse transformation: The desired voltages in dq frame are transformed into abc frame.

- Modulation: The magnitude of the desired voltages is determined and sinusoidal PWM is applied to obtain the switching pattern.

Firstly, the voltage phase θ is detected and three-phase voltages and currents, v_{gabc} and i_{abc}, are acquired. Using θ, the transformation abc–dq in equations (4.5) and (4.6) is performed to calculate v_{gdq} and i_{dq}. v_{gdq} is used by the PLL to process the loop filter and the phase-angle generator, and keeps feedback to the transformation abc–dq.

Secondly, the power control is developed in two parts:

1. *Desired Current Calculation.* The d-component of the grid voltage v_{gd} is input and the user specifies the Desired Active Power P_{des} and Desired Reactive Power Q_{des}. Recalling the power equations (4.9)–(4.10), this block calculates

$$i_{dD} = \frac{P_{des}}{v_d(3/2)}, \tag{4.16}$$

$$i_{qD} = \frac{-Q_{des}}{v_d(3/2)}, \tag{4.17}$$

where the pair i_{dD} and i_{qD} is the desired current in the dq frame, i.e. $i_{dqD}(P_{des}, Q_{des}, v_d)$. Once the desired current is calculated, the next block takes action.

2. *Decoupled Control.* This scheme has been very well reported in the literature using different names such as *feedforward decoupling control, current control* or similars, but the approach is the same in essence [53, 54, 55, 56]. It could also be said that it is a current regulator with PI control. It consists of a Proportional-Integral controller which uses a decoupling of the active and reactive power, P and Q respectively. In order to achieve this, a current dependent term has to be decoupled. Only certain values are allowed for PI gains due to the linked nature of the system dynamics. As shown in Figure 4.10 the decoupled control is governed by the following:

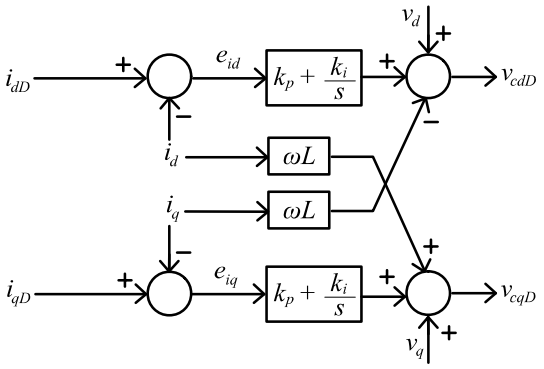


FIGURE 4.10: PI decoupled control

$$v_{cdD} = (k_p + k_i/s)e_{id} - \omega L i_q + v_{gd}, \tag{4.18}$$

$$v_{cqD} = (k_p + k_i/s)e_{iq} + \omega L i_d + v_{gq}, \tag{4.19}$$

where

$$e_{id} = i_{dD} - i_d, \tag{4.20}$$

$$e_{iq} = i_{qD} - i_q, \tag{4.21}$$

and within the dq frame nomenclature, v_{cdD} and v_{cqD} are the desired converter voltage components, e_{id} and e_{iq} are the current errors, and k_i and k_p are the integral and proportional gains, respectively. Having v_g aligned with the d-axis as explained in Section 4.2. According to the alignment of v_g, $v_{gq} = 0$. As a result, in equation (4.18), v_{cdD} is dependent on i_q and v_{cqD} is dependent on i_d, i.e. $v_{cdqD}(i_{dqD}, v_{dq}, i_{dq}, \omega, L)$. In Figure 4.10 it is appreciated that v_{cdD} depends mostly on the d-components, except for the term $-\omega L i_q$. Similarly, v_{cqD} depends mostly in q-components, except for the term $+\omega L i_d$. These exceptions define the dependency between v_{cdD} and v_{cqD}, which become controllable for some k_p and k_i.

The inverse transformation is the third step. The transformation $dq - abc$ is applied to the desired voltages v_{dqD} to obtain the desired abc voltages, v_{abcD}. Finally, to convert the desired (or reference) voltages into switching patterns, a modulation part takes place. v_{cabcD} works as the reference three-phase voltages. As the maximum voltage per leg is given by

$$v_{x(max)} \approx \pm \frac{v_{DC}}{2}, \tag{4.22}$$

the modulation vector is defined as

$$M_{abc} = \frac{2}{v_{DC}}[v_a \; v_b \; v_c] = [M_a \; M_b \; M_c], \tag{4.23}$$

where M_x is the duty cycle (positive or negative) for each leg and it defines the modulation ratio into the SPWM (as shown in section 2.3), which turn the twelve switches on and off. In this manner, we have two functions that trigger the switches, a continuous $M_{abc}(v_{abcD}, v_{DC}, f_g)$ and a switched $s_{abc}(f_{sw})$. Overmodulation depends on v_{DC} and P: if the modulation index exceeds unity, it leads to increased ripple in P and Q, and contributes to more triangle-shaped currents i_{abc}.

The speed at which the described algorithm is processed depends in the processor of the hardware unit. Even though the algorithm could process at several [MHz], the output of the algorithm will still be restricted by the maximum switching frequency of the transistors of the inverter.

4.4.1 Simulations

The PI Decoupling Control was implemented in Matlab/Simulink as can be seen in Figure 4.11 using a Fixed-step solver with fundamental sampling time of 1 [μs]. Initially, this solver configuration prevented state-space model simulations from crashing and later provided a degree of certainty when using unexplored Simulink blocks or codes, and when debugging errors across the different frequencies involved in the system. However, it has been noted that setting a variable-step solver has provided the same simulation results.

The NPC inverter is connected to two voltage sources with the same voltage value along with measuring blocks. This assumes the capacitors are balanced, which in a non-ideal scenario will require the use of a space-vector modulation approach. However, the assumption that the capacitors are balanced is made and SPWM is used in order to focus purely on the power control and to provide a benchmark to compare the DPC to.

The inverter in Figure 4.11 also has a 12-bits-gate input and the three phase A,B and C as outputs. These phases pass through a current and voltage measuring block (V-I m2), which measures i_{abc} and v_{cabc}. Then, after the filter, there's a second measurement block (Volt meas) which measure the grid voltage v_{gabc}. Notice that the ground of the voltage measuring blocks for the DC side of the inverter as well as 'V-I m2' is connected to the mid-point of the inverter, whereas the neutral point used for 'Volt meas' is the one connected to the grid neutral point ($node10$) on the AC side. The transformations of v_{cabc}, i_{abc} and v_{gabc} are shown

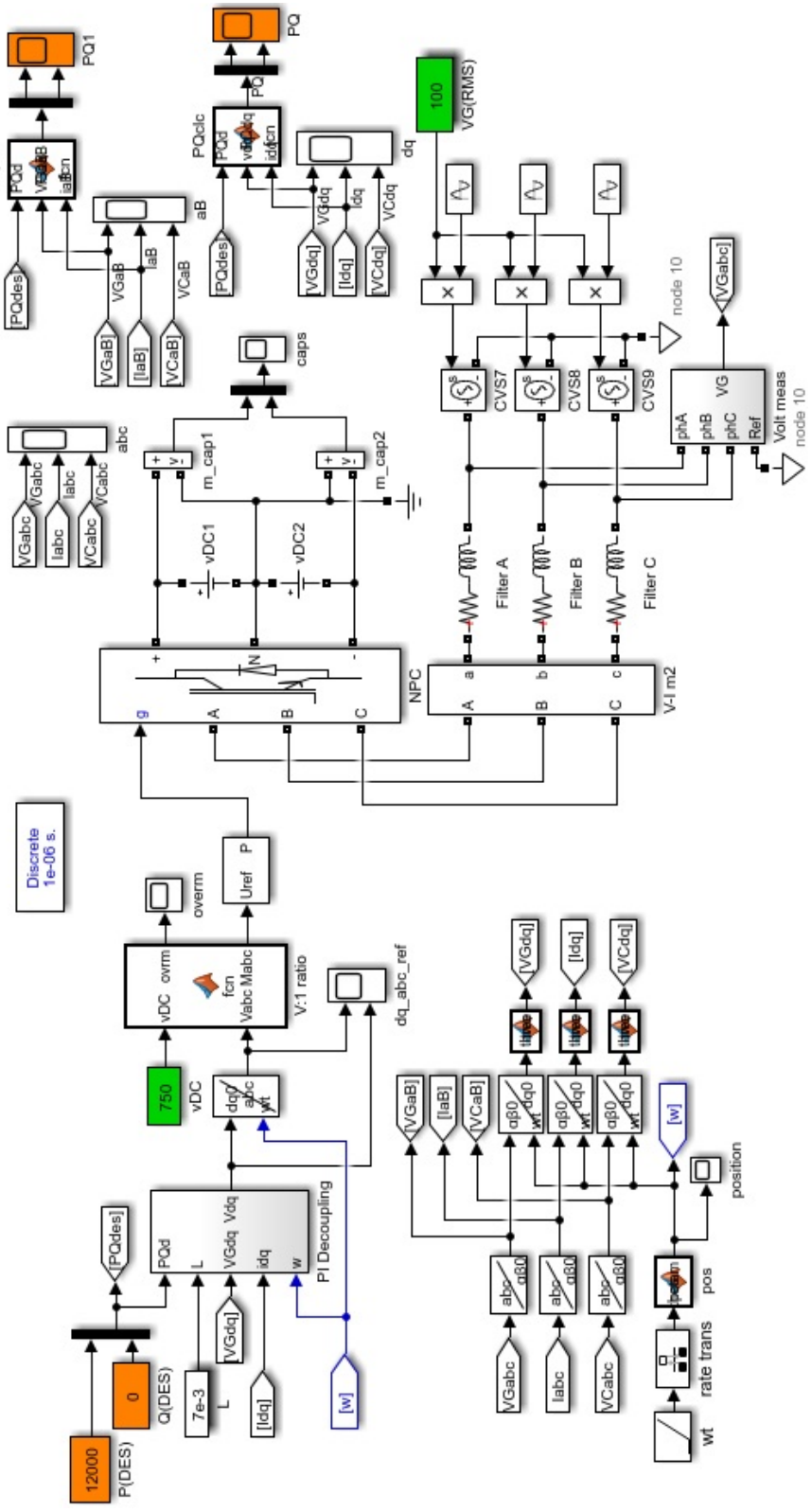


FIGURE 4.11: Simulink schematic of PI Decoupling Control

TABLE 4.1: Set of values for PI decoupled control, where $x = \{a, b, c\}$.

Parameter	Value	Unit
Q_{des}	0	[kW]
v_{DC1}	375	[V]
v_{DC2}	375	[V]
v_{DC}	750	[V]
R_x	50	[mΩ]
L_x	7	[mH]
v_g	100	[V_{RMS}]
f_s	1	[MHz]
f_g	50	[Hz]

in the bottom left of the figure and they are displayed at the top-right side, along with the power. The input values are on the top-left part, with the values of desired active and reactive power in orange blocks. These inputs are processed in the *PI Decoupling* block and v_{dqD} is obtained, which is then transformed back into the *abc* frame. Consequently, the duty cycle for each phase is calculated, and the modulator does its job and translates to binary values for each of the 12 switches serving as inputs for the inverter.

v_{ga}, v_{gb} and v_{gc} were initialised as stated in equations (4.2)–(4.4). A 3-level PWM generator was used and it is assumed that each half of the DC-link (positive and negative side) has a voltage source (i.e., no capacitor is present and hence no capacitor balancing is needed). Three simulations providing a comparative performance of power control were carried out with the following characteristics:

- *Sim 1.* Desired active power $P_{des} = 12$[kW] with $k_p = 15$ and $k_i = 500$.
- *Sim 2.* Desired active power $P_{des} = 12$[kW] with $k_p = 7$ and $k_i = 90$.
- *Sim 3.* Desired active power $P_{des} = 15$[kW] with $k_p = 15$ and $k_i = 500$.

The rest of the parameters used in all the simulations (*Sim 1, Sim 2* and *Sim 3*) are listed in Table 4.1.

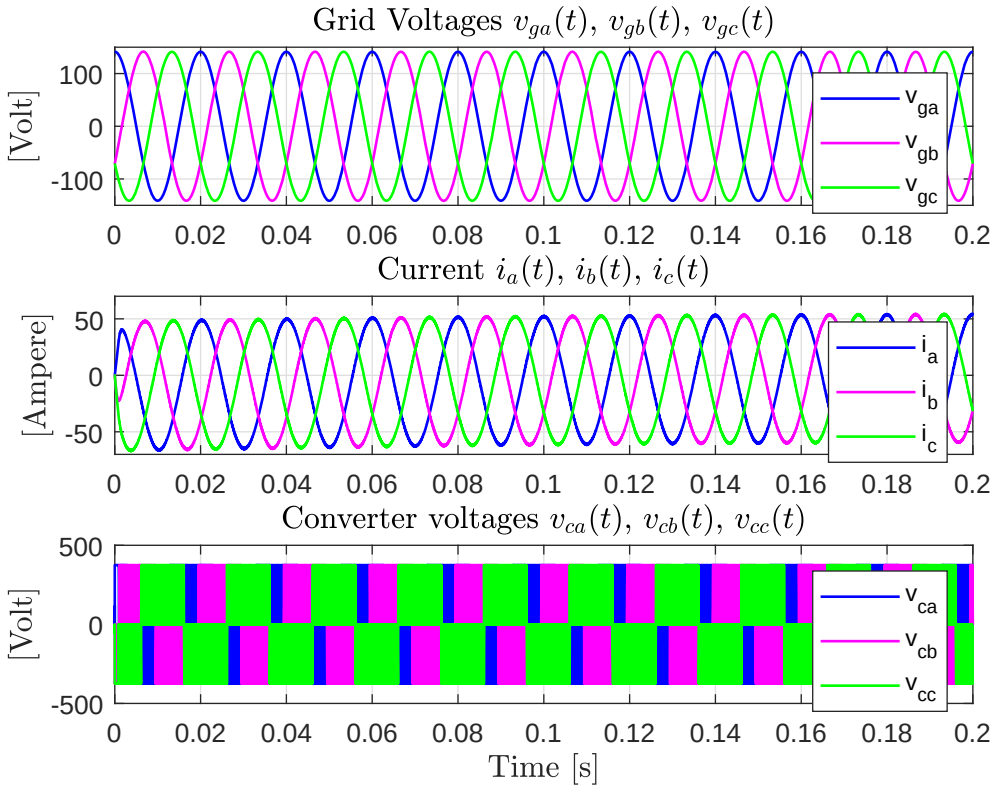


FIGURE 4.12: *Sim 1.* Voltages and currents in *abc* frame for PI decoupled control, $P_{des} = 12$[kW] with $k_p = 15$ and $k_i = 500$.

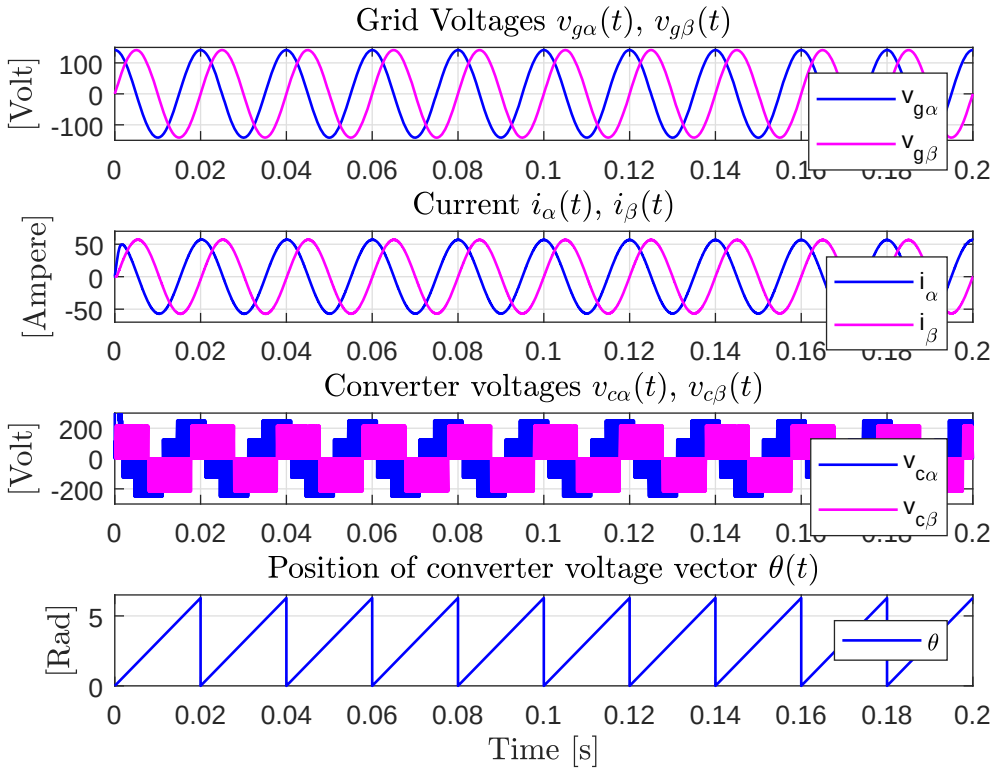


FIGURE 4.13: *Sim 1.* Voltages and currents in $\alpha\beta$ frame for PI decoupled control, $P_{des} = 12$[kW] with $k_p = 15$ and $k_i = 500$.

The time window of the results are kept the same and match with the evolution of the output power. For *Sim 1*, Figure 4.12 presents the grid voltages, currents and converter voltages on a per-phase basis. On the top figure, the grid voltage v_{ga}, can be seen as the blue line starting with a value of v_{g-peak} times $\sin(\pi/2)$ (an assumed grid voltage of 100 $[V_{RMS}]$) at $t = t_0 = 0$[s]. All three voltages v_a,v_b and v_c are shifted 120^o from each other. In the middle figure, it is appreciated that the average value of the currents i_{abc} start with a small offset below zero due to the start-up of the simulation while reaching P_{des} and the three currents show a consistent sinusoidal pattern. The output voltages of the converter v_{ca}, v_{cb} and v_{cc} can be seen in the bottom figure, these can be seen switching between zero and either $+375$[V] or -375[V]. The average value of these three voltages however, represent three sinusoidal voltages 120^o apart from each other. v_{cb} overlaps v_{ca}, but is as well overlapped by v_{cc}. This is what causes v_{cc} to be shown entirely, while the other two voltages are seen partially.

Figure 4.13 shows the corresponding grid voltages, currents and converter voltages in the $\alpha - \beta$ frame. The α component of the grid voltage in blue is v_{g-peak} at t_0, while the β component in magenta is zero at t_0. i_β does not exhibit transient response and i_α shows a slight transient at the very beginning. The converter voltages ($v_{c\alpha\beta}$ are switching between zero and ±375[V], and present a slight lagging with respect to the grid voltages ($v_{g\alpha\beta}$). The bottom graph of Figure 4.13 shows the position of the vector $v_{\alpha\beta}$; starting from 0 [Rad] at t_0, it reaches 2π[Rad] when $v_{g\alpha\beta}$ completes one cycle (i.e., every 20 [ms]).

In Figure 4.14, once again the grid voltages, currents and converter voltages are shown, this time in the dq frame. It can be appreciated here the alignment of v_{gdq} to the d frame (i.e. the vector v_g is attached to v_{gd}) and consequently $v_q = 0$. In the middle graph, i_d reaches a value in a very short period of time and stays there, while i_q is triggered to a negative value and slowly reaches zero. The converter voltages now look close to a sawtooth due to the transformation interacting with

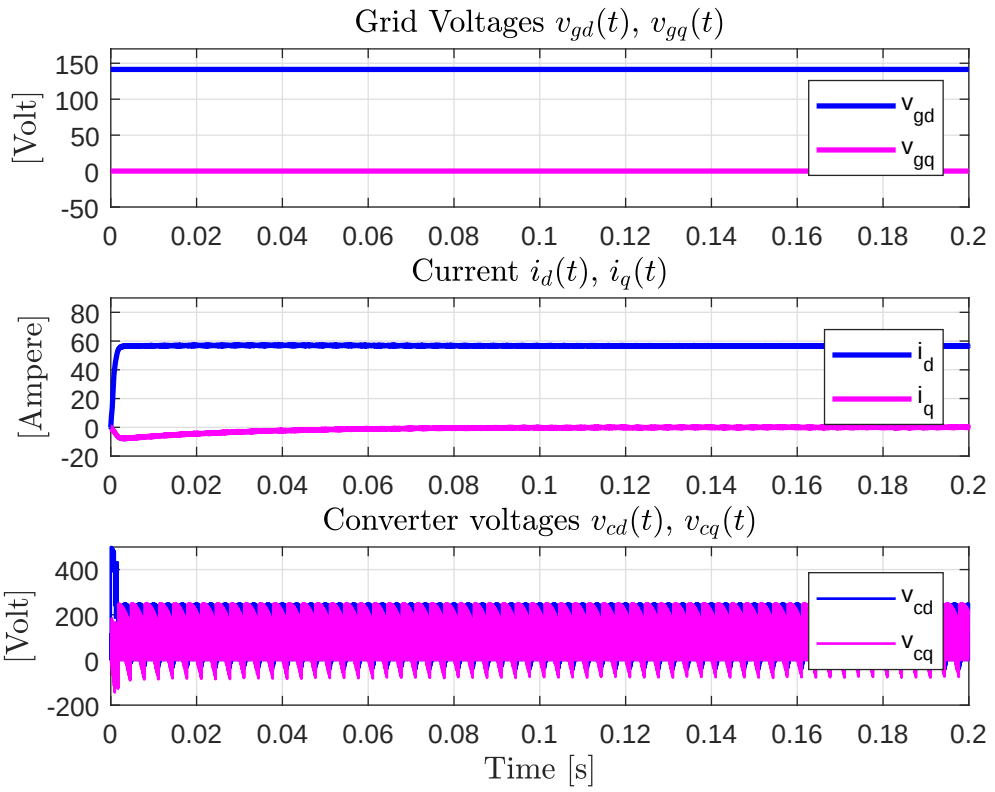


FIGURE 4.14: *Sim 1.* Voltages and currents in dq frame for PI decoupled control, $P_{des} = 12$[kW] with $k_p = 15$ and $k_i = 500$.

the DC-side and PWM ripple. v_{cd} starts with a high peak while v_{cq} starts with a negative peak. v_{cdq} shows very subtle changes in their waveform.

The active power and reactive power are shown in Figure 4.15. The desired active power $P_{des} = 12$[kW] and the desired reactive power $Q_{des} = 0$[kW] are shown in red. In *Sim1* P (blue line) seems to reach P_{des} much before $t = 0.02$[s] while Q (blue line) seems to converge to Q_{des} at $t = 0.1$[s]. However, looking closely in Figure 4.16, P slowly approaches P_{des} and settles at about $t = 0.12$ [s]. Q does not reach its desired value until $t = 0.14$[s]. Therefore, depending on the precision required for P_{des} and Q_{des}, the PI decoupled control could delay the convergence of P and Q.

It can be noted that the initial high or 'deviated' values in $v_{c\alpha\beta}$ and v_{cdq} in Figures 4.13 and 4.14 at $t < 0.005$ [s] correspond to the time it takes P to reach a very close value of P_{des}. Also, i_{abc} in Figure 4.12 shows an interesting behaviour as it corrects

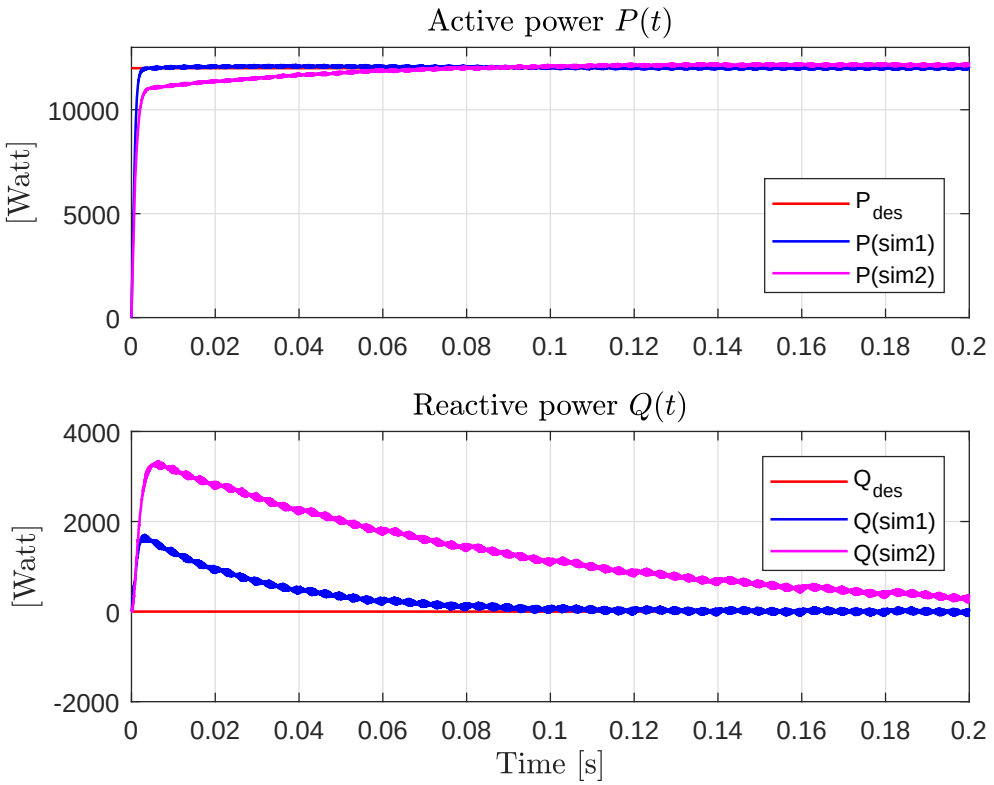


FIGURE 4.15: Power for PI decoupled control. *Sim 1* ($P_{des} = 12$[kW], $k_p = 15$, $k_i = 500$) and *Sim 2* ($P_{des} = 12$[kW], $k_p = 7$, $k_i = 90$).

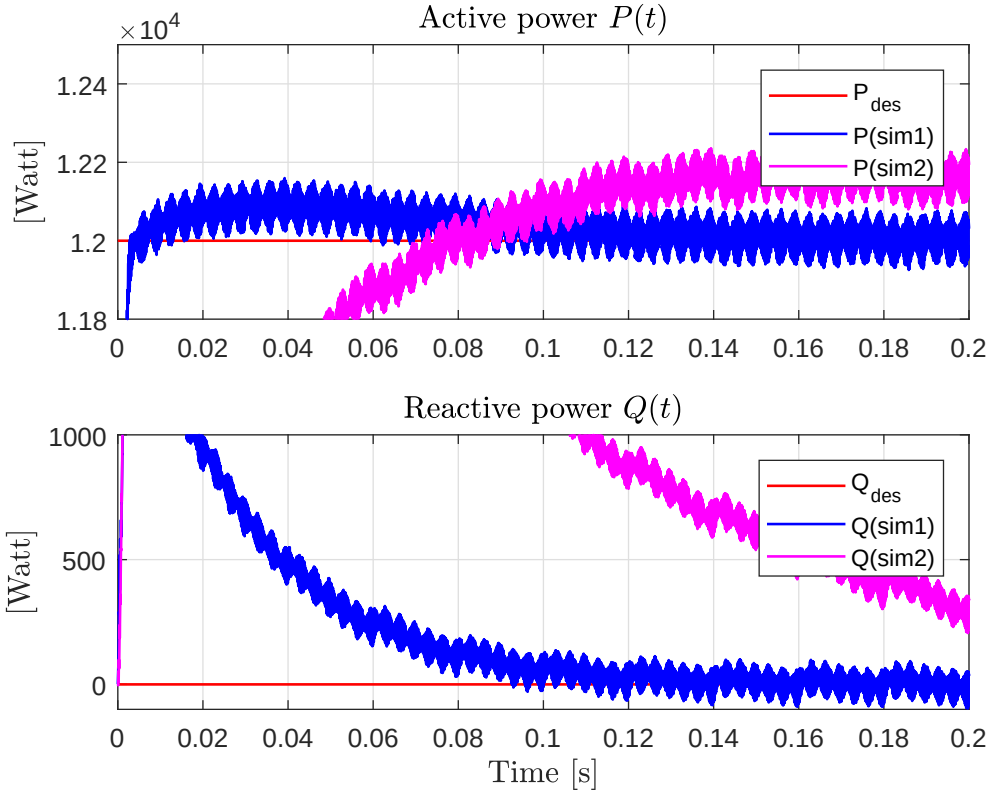


FIGURE 4.16: Detail of power for PI decoupled control. *Sim 1* ($P_{des} = 12$[kW], $k_p = 15$, $k_i = 500$) and *Sim 2* ($P_{des} = 12$[kW], $k_p = 7$, $k_i = 90$).

its mean value towards the centre (zero value) and reaches it when $Q \approx Q_{des}$ in Figure 4.15. The correspondence of the current with the reactive power Q (Figure 4.15) is also noted in i_d (4.14), which approaches zero as Q approaches zero as well.

The gains k_p and k_i in *Sim1* were tuned by testing in order to reach the best trade-off between convergence speed and damping. For this reason, *Sim2* was run with exactly the same set of parameters, but with controller gains were changed to $k_p = 7$ and $k_i = 90$. Results of *Sim2* are shown in Figures 4.15 and 4.16 with magenta lines. In these, P is seen to reach a close value of P_{des}, however, it requires a large amount of time to converge (or achieve a closer value) to P_{des}. 0.2 [s] is not enough time for Q to converge to zero either. For the *Sim2*, the latter figures show a much slower convergence of P and Q, although the gains still keep the damping to a minimum. There are also values of k_i and k_p in which P_{des} and Q_{des} are never reached.

The observation on the gains are as follows. For small k_p and small k_i, P, Q show very slow convergence and some damping, sometimes Q does not converge. For small k_p and large k_i, P and Q usually converge, but damping is very high and there is delay of convergence with precision, also very high overshoot: 6 [kVAR] for Q and 15 [kW] for P. For large k_p and small k_i, P stays very close to convergence but could never fully reach, Q could never reach convergence, also very low overshoot 1.6 [kVAR] for Q (although, theoretically they could converge at some point in time). For large k_p and large k_i, P and Q reach a near-convergence-value very fast, but they have a large delay to reach the precise value, Q initial peak of 1.6 [kVAR] but rapidly within a 200 [VAR] error.

Back to the decoupling control in equations (4.18)-(4.19), the terms $(k_p + k_i/s)e_{id}$ and $(k_p + k_i/s)e_{iq}$ are accountable for the low frequency component, while the terms $-\omega L i_q$ and $+\omega L i_d$ are responsible for the high frequency component. That is, the

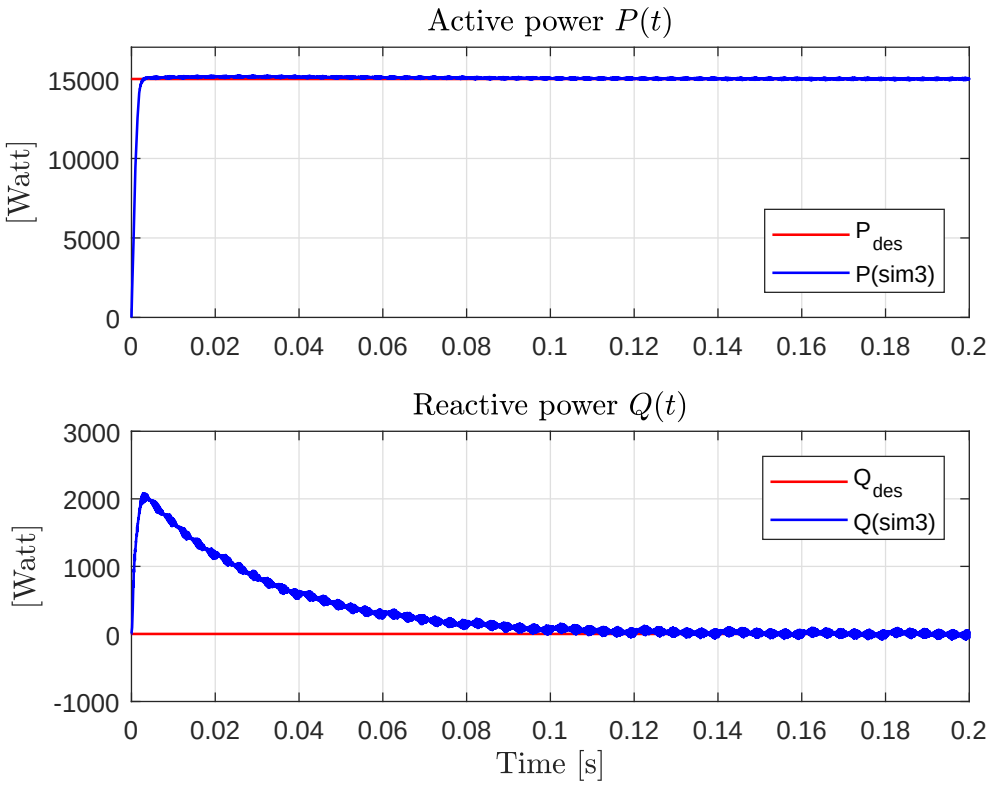


FIGURE 4.17: *Sim 3.* Power for PI decoupled control, $P_{des} = 15$[kW] with $k_p = 15$ and $k_i = 500$.

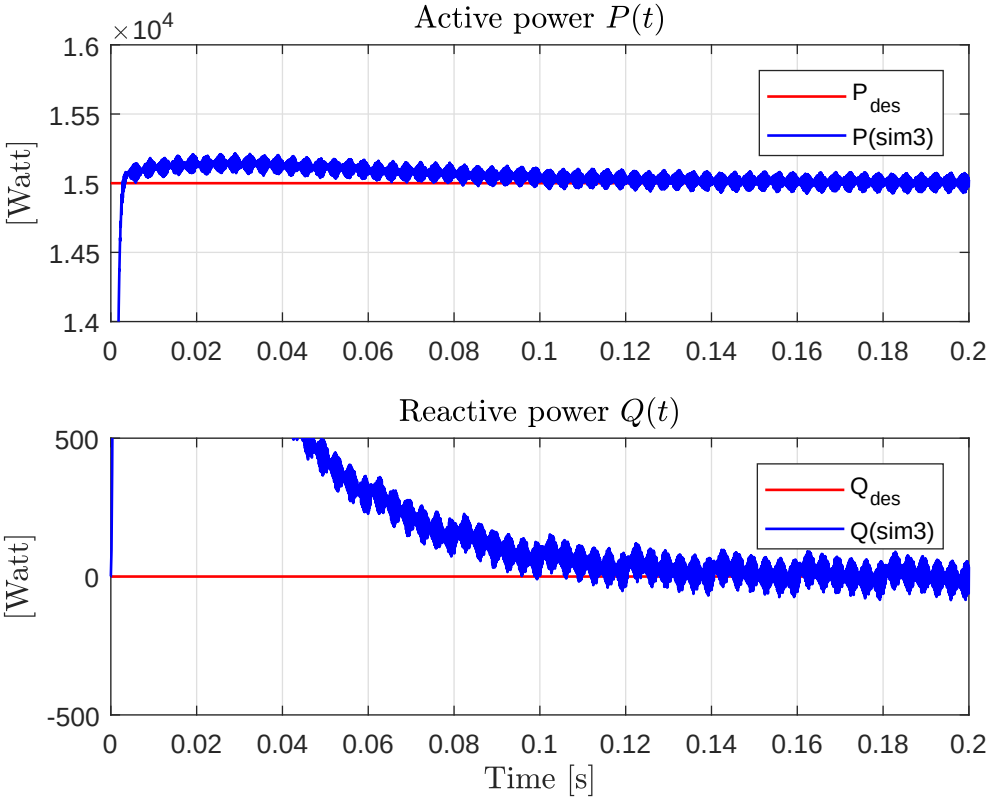


FIGURE 4.18: *Sim 3.* Detail of power for PI decoupled control, $P_{des} = 15$[kW] with $k_p = 15$ and $k_i = 500$.

latter terms allow P and Q to reach (or search) their desired values more precisely once they have reached a near-convergence-value. However, for every set of desired values, the controller gains need to be tuned as they are not robust. Unsuitable value of gains may slow down the convergence or even cause a non-convergence.

Moreover, *Sim3* shows the performance with $k_p = 15$ and $k_i = 500$, this time for $P_{des} = 15$ [kW] and $Q_{des} = 0[kVAR]$. Figure 4.17 shows this simulation performance is very similar to *Sim1* for P and Q as they both seem achieve the desired values at about the same times, except that Q has a larger overshoot in *Sim3*. However, looking closely at the performance of *Sim3*, Figure 4.18 shows that the convergence time is about the same, although Q shows a ripple or a repeated pattern for every cycle of v_g (i.e. a sampling time of f_g). This proves how the reactive power shows convergence difficulties when the converter's output is near its maximum active power.

As the desired active power increases, Q shows a higher peak at the start, both P and Q exhibit slow down of convergence and start showing a high frequency ripple. If the desired reactive power Q_{des} is moved to a higher value, the inverter performs better, but at the expense of reduced efficiency.

One more disadvantage -which is often omitted- is that decoupled control requires knowledge of the filter parameters, in this case L. However, this could become more complex for other filter configurations [57].

4.5 Direct Power Control

Direct Power Control (DPC) is an uncontrolled-switching-frequency technique and it is rooted in a vectorised approach. The parallel between DPC and SVM is that both use the same vector map, but the way the vectors are selected and timed differs. An approach to SVM has been shown in Chapter 3 for a single-phase NPC

inverter, where the vector map consists of two large vectors, four middle vectors and three zero vectors.

TABLE 4.2: Vector number assignation to s_{abc}.

Group	Zero			Large						Medium				
vecn	0	25	26	1	2	3	4	5	6	7	8	9	10	11
s_{abc}	000	-1-1-1	111	1-1-1	11-1	-11-1	-111	-1-11	1-11	10-1	01-1	-110	-101	0-11
Group	Med	Small (C_1 Disch)						Small (C_2 Disch)						
vecn	12	13	15	17	19	21	23	14	16	18	20	22	24	-
s_{abc}	1-10	100	00-1	010	-100	001	0-10	0-1-1	110	-10-1	011	-1-10	101	-

As explained in Chapter 2, for the three-phase type inverter connected to the grid shown in Figure 4.3, only three switching patterns are allowed per leg, $s_x \in \{-1, 0, +1\}$. Since the number of the allowed operating states is defined by S^l, with S being the number of the allowed switching patterns and l the number of legs, there are $3^3 = 27$ total allowed patterns or vectors.

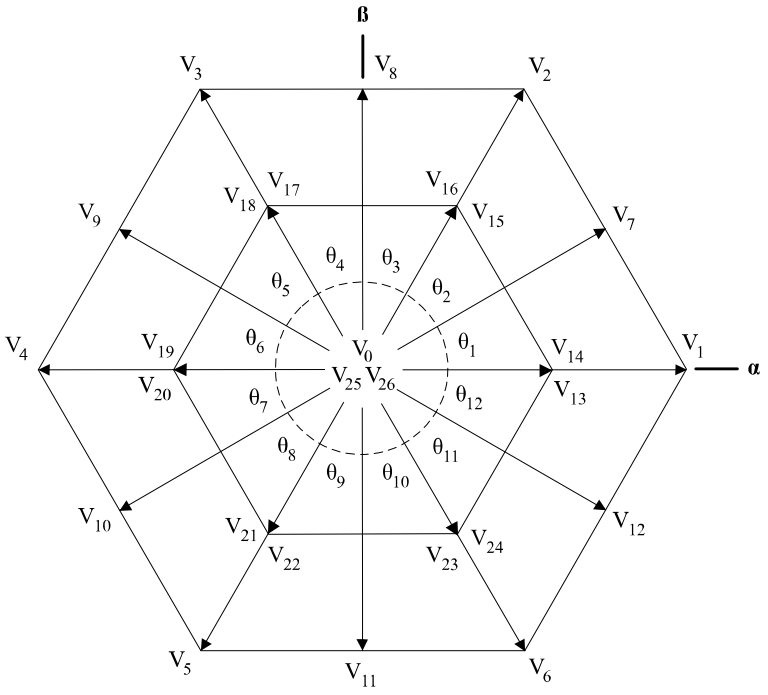


FIGURE 4.19: Vector map for the NPC inverter, where $V_{vector-number} = V_n$.

A vector number *vecn* is a specific switching pattern for each of the three legs, as shown in Table 4.2 and belong to one of the four types of vectors. According to this table, there are three switching patterns that provide zero output voltage, six large vectors which provide $\pm 2/3(v_{DC})$, six medium vectors which provide $\pm v_{DC}/\sqrt{3}$,

and 12 small vectors which provide $\pm v_{DC}/3$. Among the small vectors, six of them discharge the top capacitor C_1 and six of them discharge the bottom capacitor C_2, just as shown in the bottom row of the table. The vector map is shown in Figure 4.19. This map has an hexagonal shape and is formed of six Large Vectors (LV) located in the pentagon corners, six Medium Vectors (MV) located at the middle of the hexagon sides, twelve Small Vectors (SV) half-way towards the hexagon corners and three Zero Vectors (ZV) at the very centre. These vectors are the only possible outputs of the converter, and they switch constantly according to the control algorithm. By placing the grid voltage vector v_g into the $\alpha - \beta$ frame in Figure 4.19, it will present a $v_{g\alpha}$ component and a $v_{g\beta}$ component, varying in such a way that vector $v_{\alpha\beta}$ will be rotating counter-clockwise (CCW) at constant speed. The precise position of the grid voltage vector in the $\alpha\beta$ frame ($v_{g\alpha\beta}$) at a specific time t is θ, and the region in which $v_{g\alpha\beta}$ is, belongs to

$$n(\theta) = \begin{cases} \theta_1 = 1 & \text{for } 0 > \theta \geq \theta_{reg} \\ \theta_2 = 2 & \text{for } \theta_{reg} > \theta \geq 2\theta_{reg} \\ \theta_3 = 3 & \text{for } 2\theta_{reg} > \theta \geq 3\theta_{reg} \\ \theta_4 = 4 & \text{for } 3\theta_{reg} > \theta \geq 4\theta_{reg} \\ \theta_5 = 5 & \text{for } 4\theta_{reg} > \theta \geq 5\theta_{reg} \\ \theta_6 = 6 & \text{for } 5\theta_{reg} > \theta \geq 6\theta_{reg} \\ \theta_7 = 7 & \text{for } 6\theta_{reg} > \theta \geq 7\theta_{reg} \\ \theta_8 = 8 & \text{for } 7\theta_{reg} > \theta \geq 8\theta_{reg} \\ \theta_9 = 9 & \text{for } 8\theta_{reg} > \theta \geq 9\theta_{reg} \\ \theta_{10} = 10 & \text{for } 9\theta_{reg} > \theta \geq 10\theta_{reg} \\ \theta_{11} = 11 & \text{for } 10\theta_{reg} > \theta \geq 11\theta_{reg} \\ \theta_{12} = 12 & \text{for } 11\theta_{reg} > \theta \geq 12\theta_{reg}. \end{cases} \tag{4.24}$$

where $\theta_{reg} = \pi/12$ [rad]. As a result, the vectors in Table 4.2 are switched into the converter output voltage v_c, according to the rotation of vector v_g in the $\alpha - \beta$ frame. The inverter output voltage vectors v_c can change several times in each region.

4.5.1 Design of the Switching Table

A graphical representation of equations (4.9)-(4.10) with $v_{g\alpha\beta}$ aligned to the d-axis, is shown in Figure 4.20. v_g is found anchored to the d axis (i.e. $v_{gq} = 0$) and rotating at ω speed on the $\alpha\beta$ (stationary) frame, where v_c vectors are positioned.

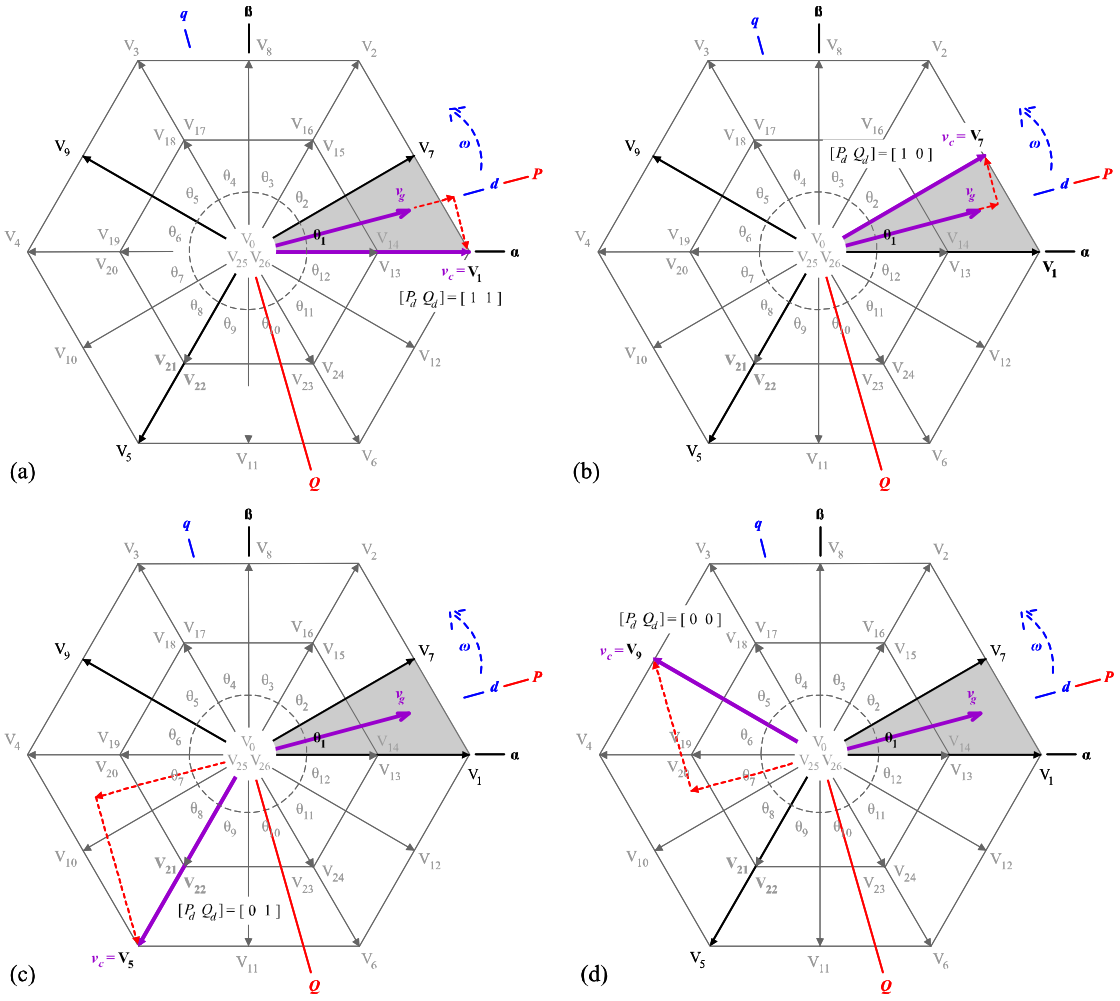


FIGURE 4.20: Vector maps for all possible PQ values.

From the tip of the v_g vector to the tip of the v_c vector, one could assume a current vector i, the dashed red lines in the set of figures represent the dq components of i for every case within θ_1. As in equations (4.9)-(4.10), this means that $P \propto i_d$ and $Q \propto -i_q$, where $+P$ is fixed to the $+d$-axis, and $+Q$ is fixed to the $-q$ axis. Under this arrangement active power P can be placed in the d axis and reactive power Q can be placed in the $-q$ axis, as shown in red.

The selected vectors for all combinations of increase/decrease of P and Q values during θ_1 are shown in Figure 4.20: V_1, V_7, V_5 and V_9. Keep in mind that the vector selected must provide the desired effect on P and Q at the same time whilst v_g is in region 1. The projection of v_c on v_g when $v_c = V_1$ in Figure 4.20(a) is positive upon the d-axis and negative upon the q-axis, causing P to increase and Q to decrease. In Figure 4.20(b), the MV V_7 is selected as v_c and the projections are positive in the d- as well as in the q-axis, increasing P and decreasing Q. Figure 4.20(c) shows a decrease in P and increase in Q, while Figure 4.20(d) shows a decrease in both P and Q. Besides,

$$P_d = \{0, 1\}, \tag{4.25}$$

$$Q_d = \{0, 1\}, \tag{4.26}$$

where the need to increase P or Q is defined as '1' and the need to decrease them as '0'. These are digital values which ease table design and implementation.

Table 4.3: DPC Table

P_d	Q_d	Index	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	θ_{11}	θ_{12}
0	0	0	V_9	V_4	V_{10}	V_5	V_{11}	V_6	V_{12}	V_1	V_7	V_2	V_8	V_3
0	1	1	V_5	V_{11}	V_6	V_{12}	V_1	V_7	V_2	V_8	V_3	V_9	V_4	V_{10}
1	0	2	V_7	V_2	V_8	V_3	V_9	V_4	V_{10}	V_5	V_{11}	V_6	V_{12}	V_1
1	1	3	V_1	V_7	V_2	V_8	V_3	V_9	V_4	V_{10}	V_5	V_{11}	V_6	V_{12}

Under this logic, a column of vectors for $n(\theta) = 1$ have been chosen and shown in Table 4.3. This table shows the principle of DPC, where depending on the

position of the grid voltage vector θ, and the required increase ('1') or decrease ('0') of the active power P_d and reactive power Q_d, a specific vector is selected. Index is simply a variable created to ease table design and and implementation.

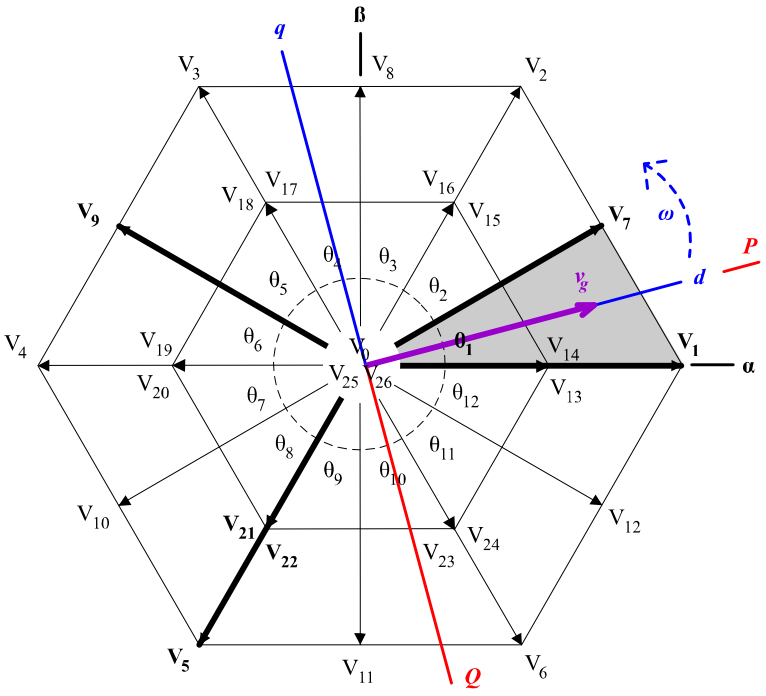


FIGURE 4.21: Vector map for power analysis.

The selection of vectors for $\theta = 1$ is summarized in Figure 4.21. The selection of vectors for the regions 2 to 12 is easy to understand in the same figure when the dq frame (which also implies the attached v_g, P and Q) is seen rotating anti-clockwise (or in ω direction) while the dq frame with the vector map remains static. Alternatively, the $\alpha - \beta$ frame and its vectors V_n can be seen rotating clockwise (CW) while keeping the dq frame static. From the dq frame perspective (static in this case), the direction of the four vectors for each case of P and Q will remain the same, but the actual vectors $vecn$ (in the $\alpha - \beta$ frame) will be constantly changing.

However, small vectors require particular attention for two main reasons. First, they help to keep the average voltage in the NP close or approximate to zero in order to prevent larger voltage ripples on the AC side. Second, they are key to regulating the capacitors voltage: half of the SVs discharge capacitor C_1, while

the other half discharge capacitor C_2, as shown in Table 4.2. Keeping capacitor voltages balanced througout the inverter operation will also help prevent uneven distribution of voltages and therefore stress on particular devices. Therefore, in order to achieve capacitors balance and large ripples, small vectors should also be included and a table of vectors properly designed should be in place.

To address the problem of the capacitor voltage balancing, a direct control for charge or discharge of the capacitors is desirable and this is provided by the small vectors. Consequently, a new control variable, which is a hysteresis type, is required in the vector table. This variable is given by

$$C_{err} = v_{C1} + v_{C2}, \tag{4.27}$$

so that C_1 discharges when $C_{err} \geq 0$ and C_2 discharges when $C_{err} < 0$. C_{err} is directly related to i_{NP} as shown in Table 4.4.

TABLE 4.4: Dependence of i_{NP} on ΔC.

i_{NP}	Charge	Discharge
$(+)$	C_1	C_2
$(-)$	C_2	C_1

Notice that v_{C2} inherits a negative value due to measurement (with NP as a point of reference), and an adding action allows to find the voltage difference between both capacitors. It becomes necessary to define

$$C_d = \{0, 1\} \tag{4.28}$$

as a digital value, where '0' represents the need to use or discharge C_1, and '1' the need to use or discharge C_2. Here, a new DPC table which keeps the capacitors balanced is proposed and shown in Table 4.5.

TABLE 4.5: Switching Table for DPC with capacitor balance

P_d	Q_d	C_d	Inx	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	θ_{11}	θ_{12}
0	0	1	0	V_9	V_{19}	V_{10}	V_{21}	V_{11}	V_{23}	V_{12}	V_{13}	V_7	V_{15}	V_8	V_{17}
0	0	2	1	V_9	V_{20}	V_{10}	V_{22}	V_{11}	V_{24}	V_{12}	V_{14}	V_7	V_{16}	V_8	V_{18}
0	1	1	2	V_{21}	V_{11}	V_{23}	V_{12}	V_{13}	V_7	V_{15}	V_8	V_{17}	V_9	V_{19}	V_{10}
0	1	2	3	V_{22}	V_{11}	V_{24}	V_{12}	V_{14}	V_7	V_{16}	V_8	V_{18}	V_9	V_{20}	V_{10}
1	0	-	4	V_7	V_2	V_8	V_3	V_9	V_4	V_{10}	V_5	V_{11}	V_6	V_{12}	V_1
1	1	-	5	V_1	V_7	V_2	V_8	V_3	V_9	V_4	V_{10}	V_5	V_{11}	V_6	V_{12}

Figure 4.22 illustrates the vector selection for the first two regions on Table 4.5. It shows a set of 5 possible vectors (in bold black) per region, its corresponding PQ values and the discharging capacitor for SVs.

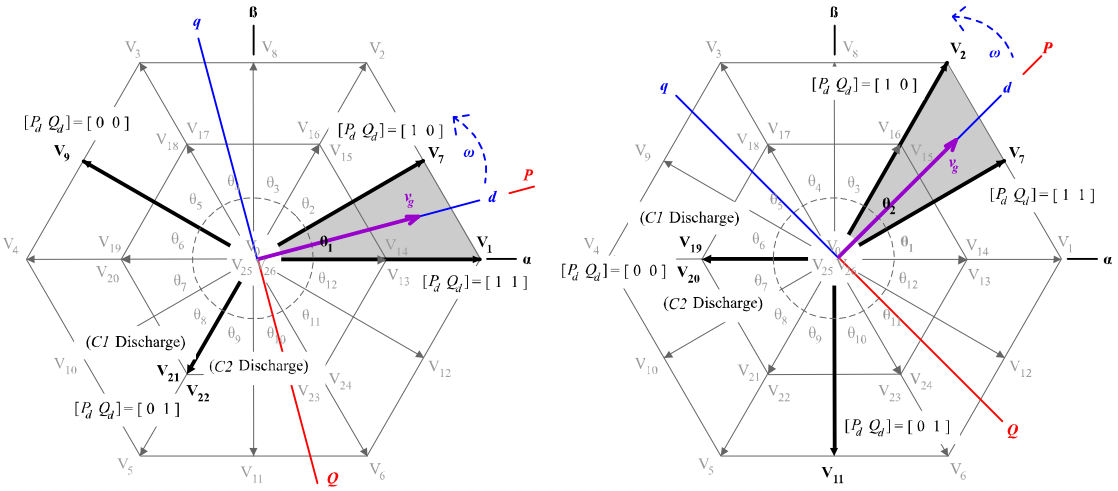


FIGURE 4.22: Selection of vectors for regions θ_1 on the left and θ_2 on the right (from Table 4.5).

As the dq frame rotates ACW, it has been chosen to alternate the SVs as follows. For $[P_d\ Q_d] = [0\ 0]$, small vectors are introduced in every other $n(\theta)$ from region 2. For $[P_d\ Q_d] = [0\ 1]$, SVs are introduced in every other $n(\theta)$ from region 1. For either $[P_d\ Q_d] = [1\ 1]$ or $[P_d\ Q_d] = [1\ 0]$, the same vectors of Table 4.3 are kept. In other words, P is prioritized for best performance and rapid reaction by only using LVs and the capacitor voltages are balanced when P does not require an increase.

Back to Figure 4.22 and taking region 1 as an example, vector V_5 for $[P_d\ Q_d] = [0\ 1]$ in Table 4.3 is replaced by vectors V_{21} or V_{22} in the new Table 4.5 for discharge

of C_1 or C_2, respectively. For $[P_d\ Q_d] = [0\ 0]$, vector V_9 is kept regardless of the capacitors voltage balance. When in region 2, for $[P_d\ Q_d] = [0\ 0]$, V_4 is replaced by V_{19} or V_{20} for discharge of C_1 or C_2 respectively. For $[P_d\ Q_d] = [0\ 1]$, vector V_{11} is kept regardless of capacitors voltage balance.

This means that all Large Vectors (LVs) of $[P_d\ Q_d] = [0\ 0]$ and $[P_d\ Q_d] = [0\ 1]$ are replaced by SVs. This may provoke a slower decrease of P and/or Q, however, due to the rotating nature of the dq frame (or otherwise of $n(\theta)$), SVs will always be alternating with MVs. Thus, decrease of P and Q will not be very slow and increase of P keeps priority.

4.5.2 The Algorithm

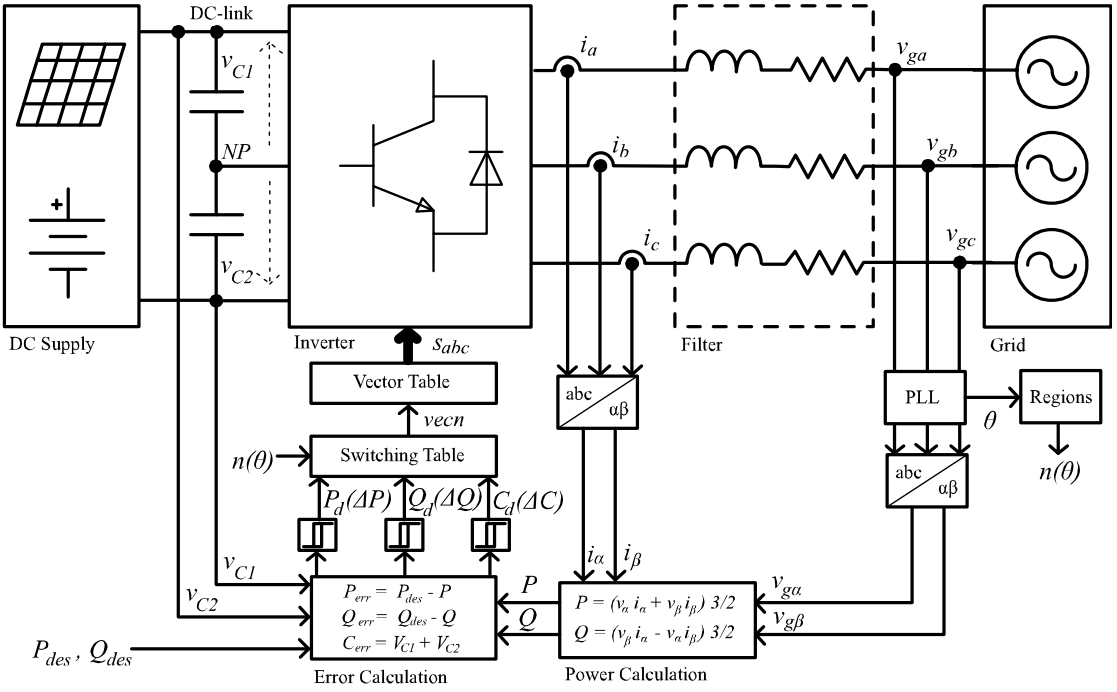


FIGURE 4.23: Illustrative diagram of Direct Power Control in NPC Inverter.

Figure 4.23 shows the complete diagram of the DPC algorithm structure, which in order to input the correct switching pattern s_{abc} into the inverter, the following steps need to take place:

- Acquisition and Transformation. The voltage phase is detected, magnitude of grid voltages and currents are acquired and then transformed to $\alpha\beta$ frame.

- Power Control. The desired and actual power as well as capacitor voltages are read, then the errors of power and capacitors' voltages are output.

- Hysteresis. Three hysteresis blocks compare the errors with preset hysteresis bands and trigger a binary value.

- Switching Tables. The combination of the three binary values points at a specific vector, which is input to the inverter.

Firstly, the voltage phase θ is detected and three-phase voltages and currents, v_{gabc} and i_{abc}, are acquired. The transformation *abc*-$\alpha\beta$ in equation (4.5) is performed to acquire v_{gdq} and i_{dq}. The block *Regions* calculates the region v_g is in $n(\theta)$, according to Figure 4.19. Secondly, the power control is developed in two parts:

- *Power Calculation.* Active power P and reactive power Q are calculated by

$$P = v_\alpha i_\alpha + v_\beta i_\beta (3/2), \tag{4.29}$$

$$Q = v_\beta i_\alpha - v_\alpha i_\beta (3/2). \tag{4.30}$$

- *Error Calculation.* User defines P_{des} and Q_{des}, the block reads P, Q, v_{C1} and v_{C2}. Then, it calculates the errors given by

$$P_{err} = P_{des} - P, \tag{4.31}$$

$$Q_{err} = Q_{des} - Q, \tag{4.32}$$

$$C_{err} = v_{C1} + v_{C2}. \tag{4.33}$$

Thirdly, once the errors have been calculated, the hysteresis stage takes place. The user has to set the trigger values ΔP, ΔQ, ΔC such that

$$P_d = \begin{cases} 0 & \text{for } P_{err} < -\Delta P, \\ 1 & \text{for } P_{err} > +\Delta P, \end{cases} \tag{4.34}$$

$$Q_d = \begin{cases} 0 & \text{for } Q_{err} < -\Delta Q, \\ 1 & \text{for } Q_{err} > +\Delta Q, \end{cases} \tag{4.35}$$

$$C_d = \begin{cases} 0 & \text{for } C_{err} < -\Delta C/2, \\ 1 & \text{for } C_{err} > +\Delta C/2, \end{cases} \tag{4.36}$$

Then, once the hysteresis blocks receive the three errors P_{err}, Q_{err}, C_{err}, they output the corresponding *digital desired values* of active power $P_d \in \{0, 1\}$, reactive power $Q_d \in \{0, 1\}$, and capacitors voltage $C_d \in \{0, 1\}$. The latter, named 1 and 2 to ease the reading.

Finally, the Switching Table 4.5 receives the binary values P_d, Q_d,C_d as well as the region $n(\theta)$, just as in a look-up table, and the resulting voltage vector is sent to the Vector Table 4.2, which decodes the leg values of s_{abc} and then into its 12 values. This is,

$$s_{abc} = \begin{bmatrix} s_a & s_b & s_c \end{bmatrix} = \begin{bmatrix} S_{a1} & S_{a2} & S_{a3} & S_{a4} \\ S_{b1} & S_{b2} & S_{b3} & S_{b4} \\ S_{c1} & S_{c2} & S_{c3} & S_{c4} \end{bmatrix}^\top . \tag{4.37}$$

Note that the transformation to dq is not required in DPC and it is only used for analysis.

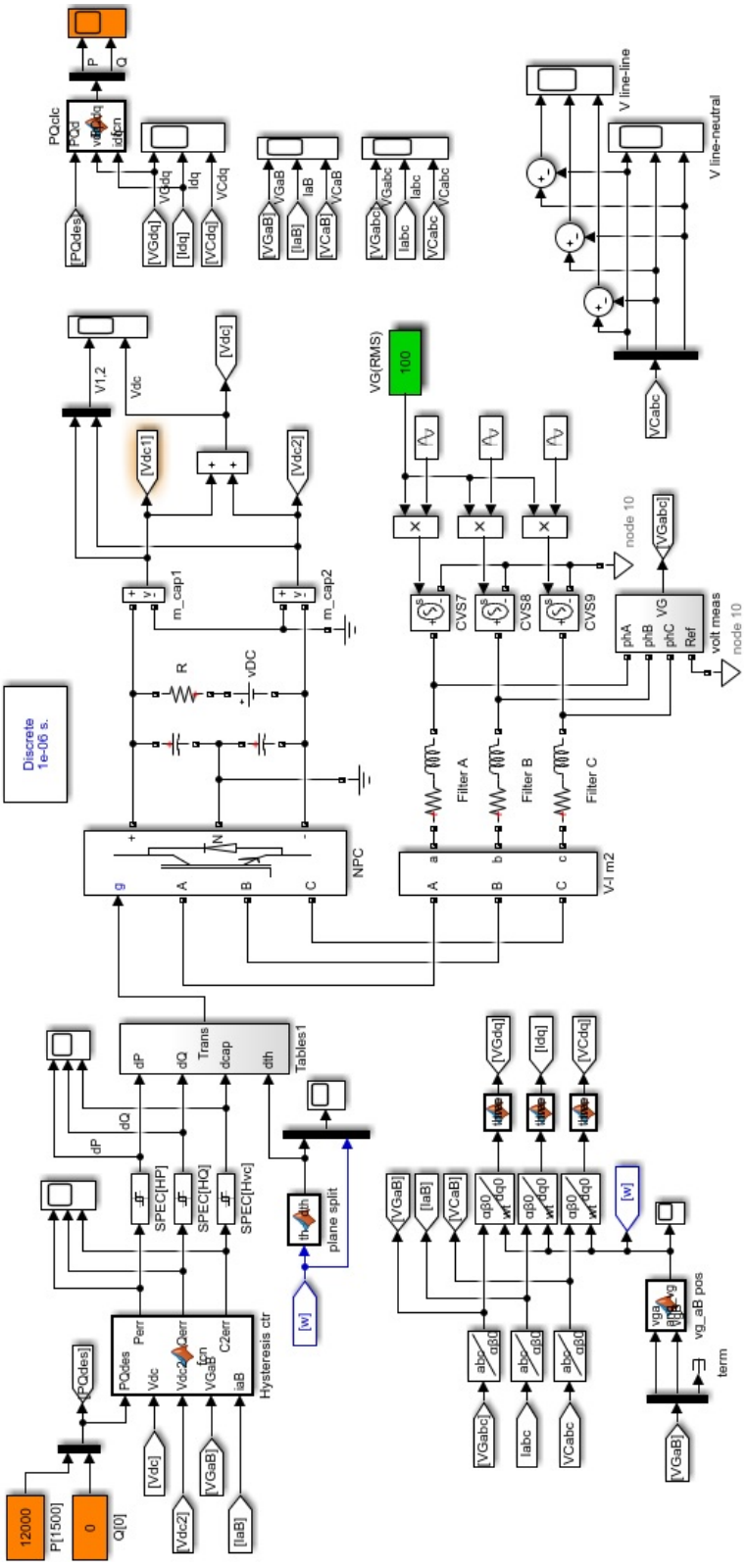


FIGURE 4.24: Simulink schematic of DPC

4.5.3 Simulations

Direct Power Control was implemented in Matlab/Simulink as seen in Figure 4.24. The arrangement of the schematic is similar to the one described in Figure 4.11, except for two parts: the DC-side of the converter is connected to two capacitors (which are fed by a DC source), and there is a hysteresis control (*Hysteresis ctr* together with *SPEC[HP], SPEC[HQ], SPEC[Hvc]*) in charge of switching capacitor discharge, active and reactive power via Table 4.5 (labeled as *Tables1*), which in turn outputs the desired switching pattern into the 12 transistors of the inverter input. Additionally, scopes are placed on the bottom right for display of line-to-line voltages.

For the simulation, Fixed-step solver with size (fundamental sample time) of 1 [μs] was used. v_{ga}, v_{gb} and v_{gc} were initialised as stated in equations (4.2)–(4.4). The new *Switching Table* for DPC in Table 4.5 is implemented, which includes capacitor balancing. Three simulations were carried out with the following characteristics:

TABLE 4.6: Set of values for DPC, where $x = \{a, b, c\}$.

Parameter	Value	Unit
Q_{des}	0	[kW]
$C1$	80	[mF]
$C2$	80	[mF]
v_{DC}	750	[V]
R_x	50	[mΩ]
L_x	7	[mH]
v_g	100	[V_{RMS}]
f_s	1	[MHz]
f_g	50	[Hz]

- *Sim 4.* Desired active power $P_{des} = 12$[kW] with tight hysteresis $\Delta P = 1$[W], $\Delta Q = 1$[VAR], and $\Delta C = 0.5$[V].
- *Sim 5.* Desired active power $P_{des} = 12$[kW] with loose hysteresis $\Delta P = 20$[W], $\Delta Q = 20$[VAR], and $\Delta C = 20$[V].

- *Sim 6.* Desired active power $P_{des} = 15$[kW] with tight hysteresis $\Delta P = 1$[W], $\Delta Q = 1$[VAR], and $\Delta C = 2$[V].

The rest of the parameters are shown in Table 4.6 and are the same for all the three simulations (*Sim 4, Sim 5* and *Sim 6*).

For *Sim 4*, Figure 4.25 presents the set of grid voltages, currents and converter voltages at the top, middle and bottom graph, respectively. v_{ga} at $t = 0$[s] starts at maximum magnitude, while the rest of the leg voltages follow equations (4.2)-(4.4). The peak current values i_{abc} stay close to 50 [V] and present some distortion at the start of the simulation, quickly catching up with the voltage phases. The converter voltage v_{cc} is seen starting at the peak negative value and overlapping converter voltages from the other two legs. The converter voltages v_{cabc}, however, can be seen constantly switching along the expected voltage levels: $-v_{DC}/2, 0$ and $+v_{DC}/2$. This latter set of voltages is shown in a per-phase basis in Figure 4.26, where each converter voltage is compared to the grid voltage and current (which are doubled in amplitude for illustration purposes). Here, the converter voltages can be seen slightly leading the grid voltages and currents in each of the phases. It can also be observed that the currents get in shape (align with grid phases) in about 1.5 [ms]. It is worth noticing that the converter and grid voltages in Figures 4.25-4.26 are measured from the grid-side neutral point (see 'node 10' in Figure 4.24), which is different from the ground used in the DC-side.

Just as the converter voltages show a sinusoidal average value in the previous figure, this sinusoid average value is more visible in Figure 4.27, where line-to-line voltages of the converter are shown (e.g. voltage difference from phase a to phase b is given by $v_{c(ab)}$). The first graph shows the voltage $v_{c(ab)}$, which starts with an increase from a very negative value. The middle graph shows $v_{c(bc)}$ starting at some near-zero value towards a positive peak value and the bottom graph presents $v_{c(ca)}$ starting with a decrease from a peak positive value. This shows the average

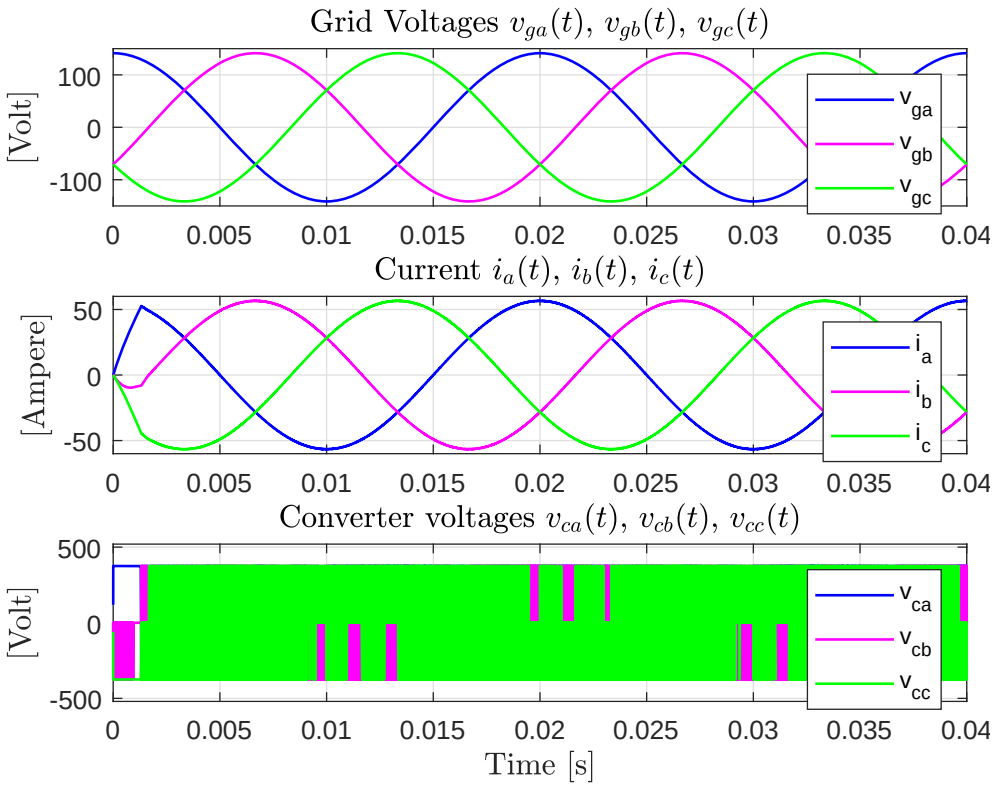


FIGURE 4.25: *Sim 4.* Voltages and currents in *abc* frame (variable basis) for DPC ($P_{des} = 12$[kW], $\Delta P = 1$, $\Delta Q = 1$, $\Delta C = 0.5$).

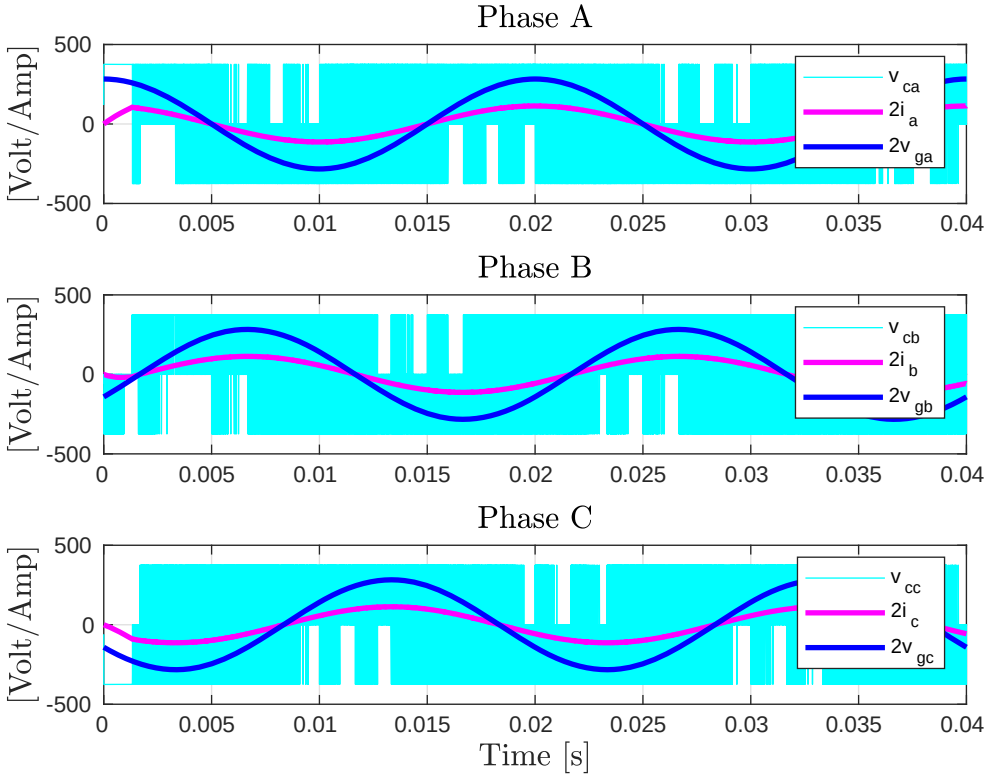


FIGURE 4.26: *Sim 4.* Line to line voltages in *abc* frame (phase basis) for DPC ($P_{des} = 12$[kW], $\Delta P = 1$, $\Delta Q = 1$, $\Delta C = 0.5$). v_g and i are doubled for illustration purposes only.

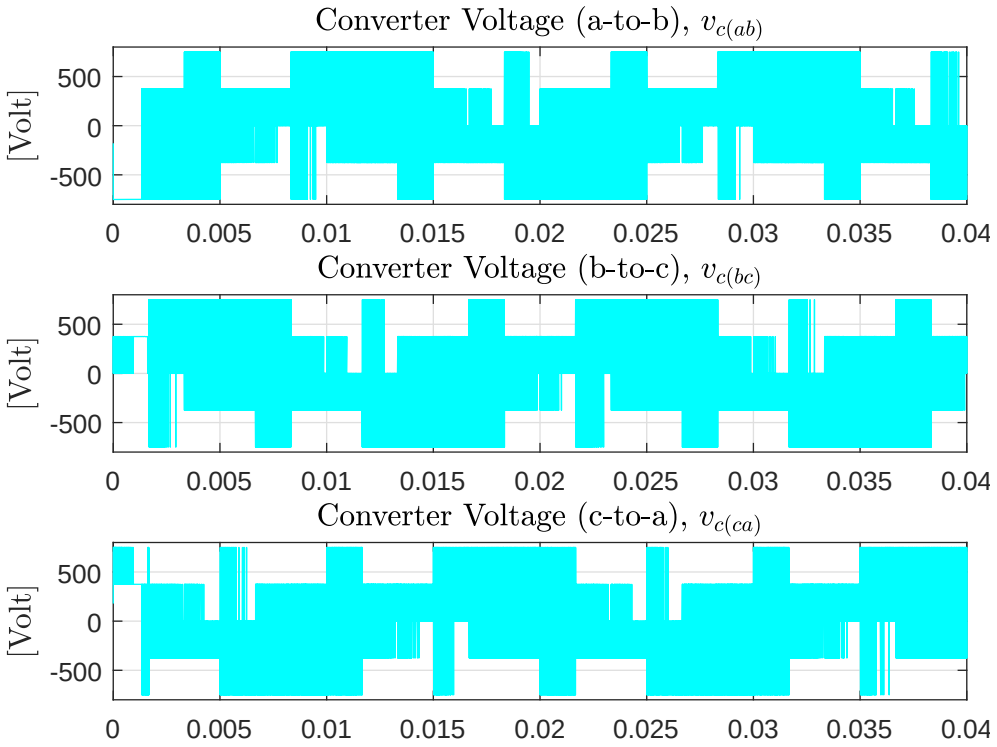


FIGURE 4.27: *Sim 4.* Line to line voltages in *abc* frame for DPC ($P_{des} = 12$[kW], $\Delta P = 1$, $\Delta Q = 1$, $\Delta C = 0.5$).

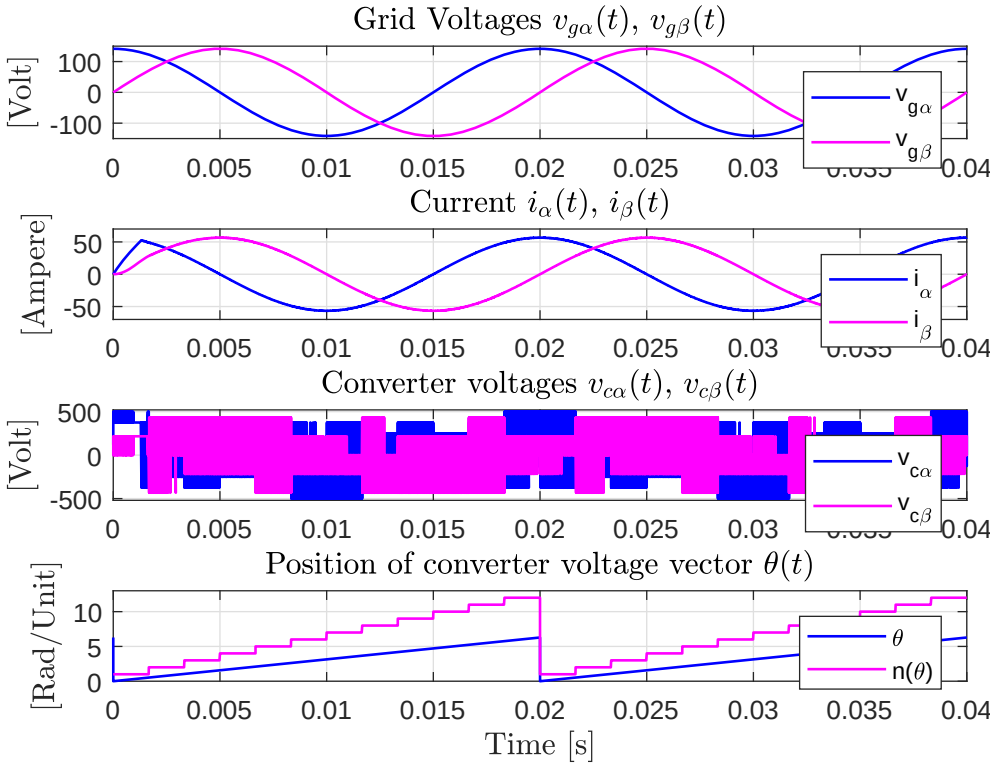


FIGURE 4.28: *Sim 4.* Voltages and currents in $\alpha\beta$ frame for DPC ($P_{des} = 12$[kW], $\Delta P = 1$, $\Delta Q = 1$, $\Delta C = 0.5$).

sinusoids in each phase are synchronised at 120 degrees from each other. The line-to-line voltages are observed to switch between values 0, ± 375, and ± 750 [V], as expected.

The corresponding voltages and currents in the $\alpha\beta$-frame are shown in Figure 4.28. Observing this figure, the top graph shows the grid voltage components $v_{g\alpha}$ and $v_{g\beta}$, which start at 100 [V_{RMS}] and 0 [V], respectively. In the two middle graphs, the currents $i_{\alpha\beta}$ and the converter voltages $v_{c\alpha\beta}$ show a transient at the start of the simulation. The bottom graph in the same figure presents the precise position of the grid voltage vector in the $\alpha\beta$ axis, defined as θ in blue, and the region number $n(\theta)$ in magenta as described in (4.24). This graph is proof that $v_{g\alpha\beta}$ is rotating CCW from region 1 to region 12.

Although dq transformation is not required for the implemented DPC algorithm, simulated measurements of voltages and currents are presented in Figure 4.29, as they provide extra information of the performance. At the top graph of this figure, it can be observed that vector $v_g = v_{gd}$. In the middle graph, i_d is seen linearly raising from 0 [V] until reaching a desired current value. As soon as i_d reaches it to keep it, i_q shows a slight disturbance and then both keep their values. The bottom graph shows v_{cdq} with a range of peak values of 0, 275, and 500 [V], and occasionally 125 [V].

Furthermore, simulated measurements of the capacitor balancing are presented in Figure 4.30. At the top graph, v_{DC} as well as each capacitor voltage are shown. v_{c1} and v_{c2} can be seen with a mirrored or a symmetric horizontal pattern. The middle graph shows the error of the voltages in the two capacitors (i.e. the difference of voltages), which is seen fluctuating in the positive side. The bottom graph presents the capacitor selected to be discharged. When the capacitor balance value reaches half of ΔC (in this case, out of a range of 0.25 [V]) as in equation (4.36), C_d changes from value 2 to 1, indicating that if the difference increases towards the

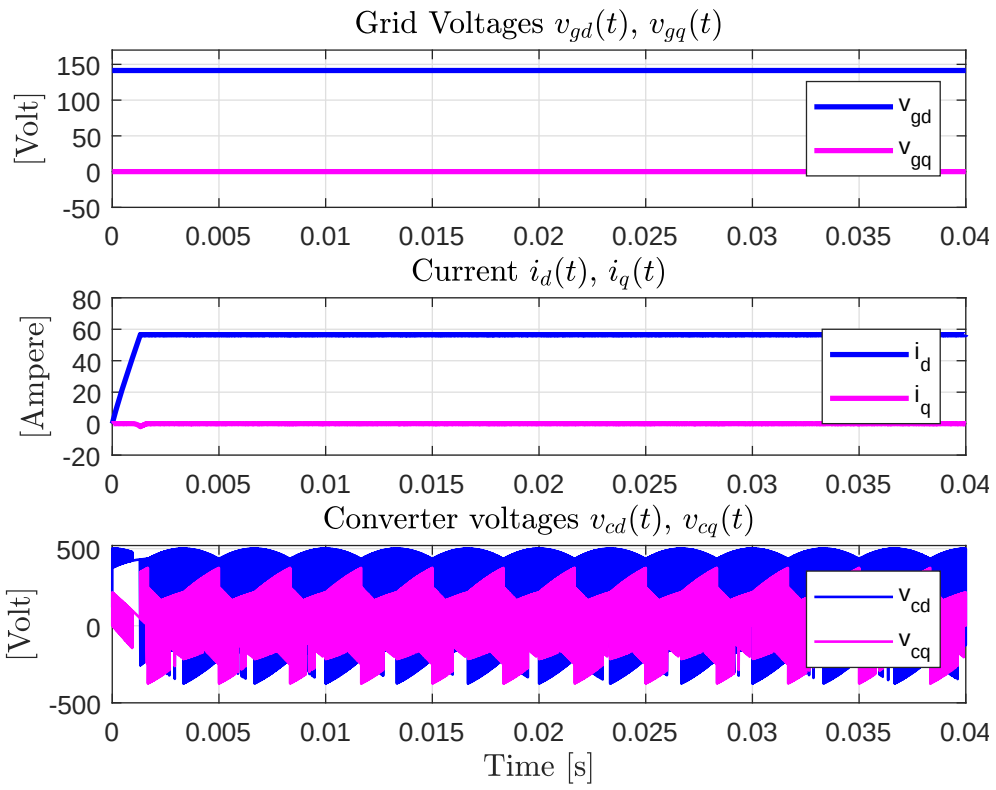


FIGURE 4.29: *Sim 4.* Voltages and currents in dq frame for DPC (P_{des} = 12[kW], $\Delta P = 1$, $\Delta Q = 1$, $\Delta C = 0.5$).

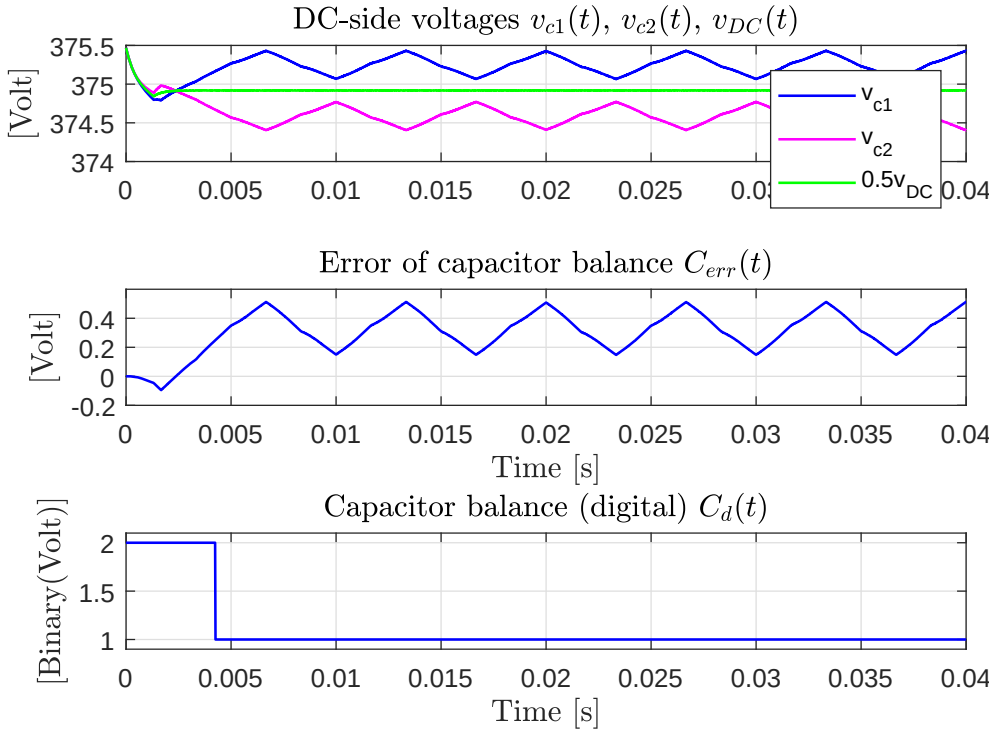


FIGURE 4.30: *Sim 4.* DC-side voltages: capacitor voltages, errors and digital value for DPC (P_{des} = 12[kW], $\Delta P = 1$, $\Delta Q = 1$, $\Delta C = 0.5$).

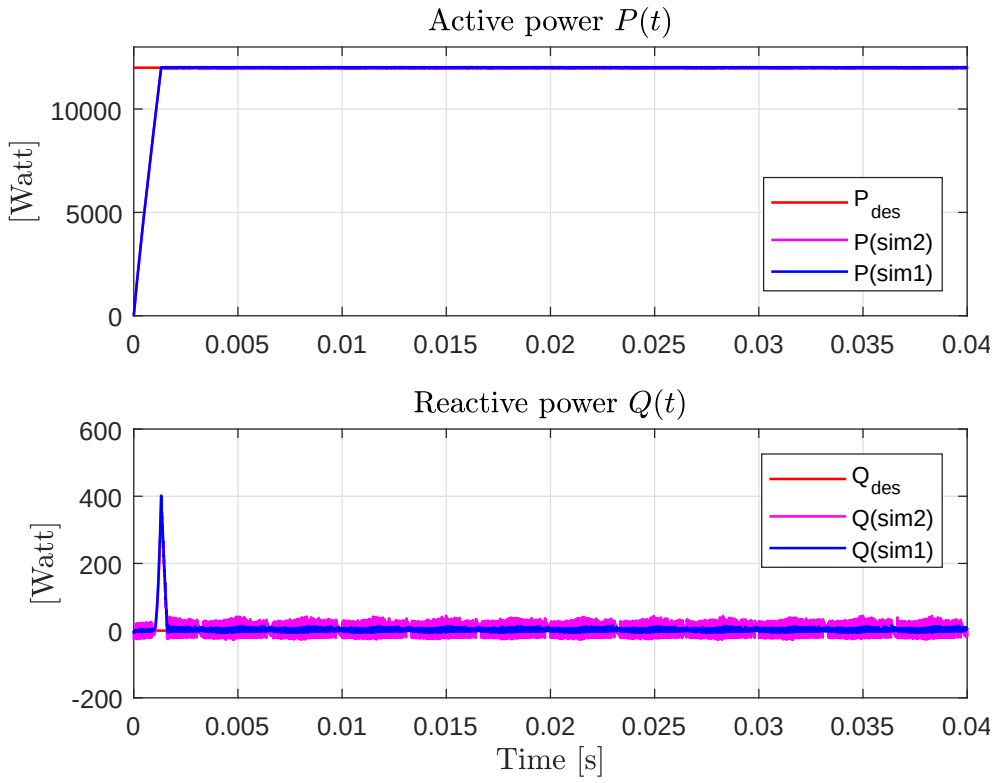


FIGURE 4.31: Power for DPC in *Sim 4* (P_{des} = 12[kW], $\Delta P = 1$, $\Delta Q = 1$, $\Delta C = 0.5$) and *Sim 5* (P_{des} = 12[kW], $\Delta P = 20$, $\Delta Q = 20$, and $\Delta C = 20$).

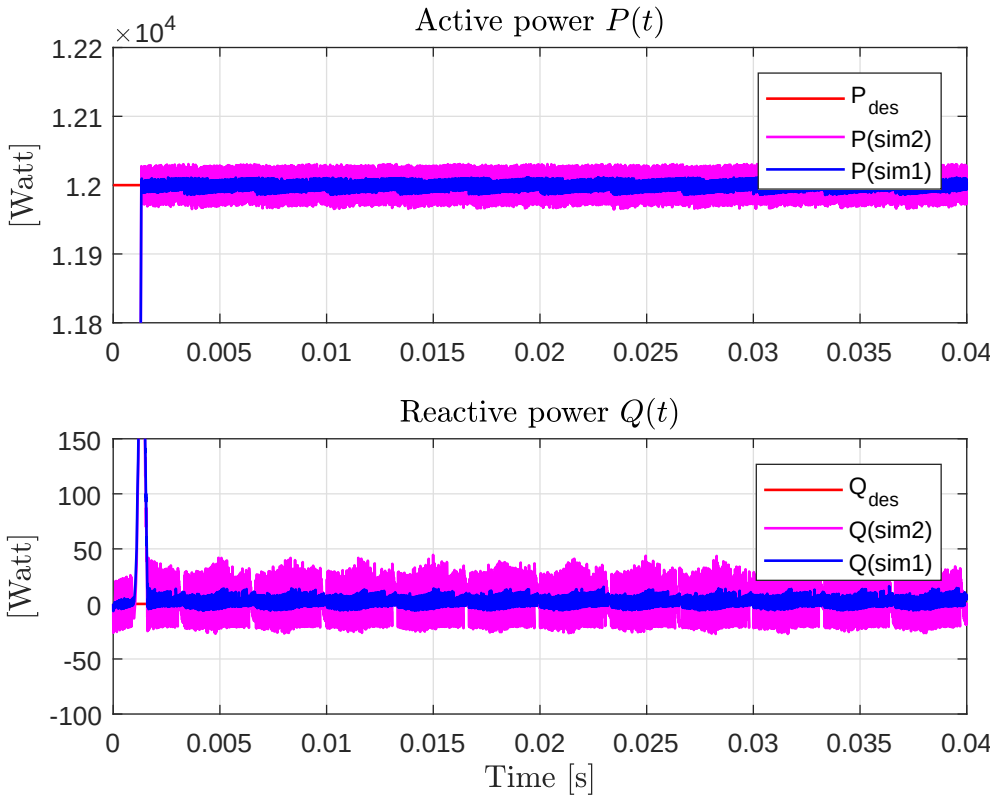


FIGURE 4.32: Detail of power for DPC in *Sim 4* (P_{des} = 12[kW], $\Delta P = 1$, $\Delta Q = 1$, $\Delta C = 0.5$) and *Sim 5* (P_{des} = 12[kW], $\Delta P = 20$, $\Delta Q = 20$, and $\Delta C = 20$).

positive value, capacitor C_1 should be preferred for discharge. The frequency at which the capacitors charge and discharge is proportional to the grid frequency, and more specifically v_{cdq}.

In Figure 4.31, the active power P and reactive power Q are shown. P starts increasing linearly and rapidly from 0 [V] and at $t = 1$[ms] Q starts rising too. Q does not stop rising until P reaches P_{des} at $t = 1.315$ [ms] and Q peaks at 400 [V], starting its abrupt decrease. Q reaches Q_{des} at $t = 1.6$ [ms]. From then on, P and Q stay at their respective desired values. This peak in the current is what explains the initial distortion of v_c, and more visibly of i in Figures 4.25, 4.28 and 4.29. A close view of the power performance is provided in Figure 4.32, where a very small ripple is detected (blue line) for *Sim 4*.

The errors of active and reactive power are pictured in Figure 4.33. P_{err} starts with an error proportional to P_{des}. When P_{err} reaches zero, Q_{err} deviates from its zero value and it returns to zero rapidly. Then, P_{err} and Q_{err} continue with zero or near-zero values until the end of the simulation. Both errors can be seen to be proportional to the performance of P and Q values.

In order to have a closer look at the operation of the digital values of power and capacitor balance, Figure 4.34 presents P_d, Q_d and C_d compared to their error values P_{err}, Q_{err} and C_{err} in a time window defined as $0 < t \leq 5$ [ms]. Digital values in the graphs were multiplied for illustration convenience. The top graph shows a fixed value of $P_d = 1$ up until $P_{err} \leq 1$, then it keeps switching according to the tolerance band indicated previously. The middle graph shows Q_d switching to keep $-1 \leq Q_{err} < +1$ and when Q_{err} deviates to a very negative value, $Q_d = 0$, making the error of the reactive power to reduce. Then Q_d keeps switching according to the tolerance, dealing with Q_{err} ripple. Finally, the bottom graph shows the digital and error variables of C, where the digital value C_d only

changes when the error exceeds 0.25 [V]. Should this value decrease -0.25 [V], C_d will change back to 2.

Sim 5 was simulated with the same set of parameters of *Sim 4*, except that the hysteresis stage was loosen. Due to the wider values of $\Delta P = 20$, $\Delta Q = 20$, and $\Delta C = 20$, it is noticeable in Figures 4.31 and 4.32 that the ripple of P and Q increased to a range of 50 [W]. However, the response of P and Q still has the same convergence speed and similar performance.

For *Sim 6*, the tight hysteresis values are chosen and the required active power is risen to 15 [kW]. The performance of this simulation is shown in Figures 4.35 and 4.36, where P reaches P_{des} at $t = 1.67$ [ms]. Q starts growing at $t = 1$[ms] and it presents a peak of 1630 [W] at $t = 1.67$[ms]. It decreases and begins a swing towards its second peak of 1675 [W] at $t = 3.34$[ms]. At $t = 5.1$[ms] Q reaches Q_{des} and from then on it presents peaks of 450 [W] every 3.3 [ms] approximately, staying at Q_{des} most of the time. This shows the algorithm struggles to keep Q (and consequently i_q) stable.

If P_{des} increases, the controller starts to struggle to keep P and Q at their desired values. First, it causes more and higher peaks in Q (e.g. bottom graph of Figure 4.36), then Q does not stop increasing, becoming unstable. A partial solution to this is increasing Q_{des} and losing efficiency. However, if P_{des} keeps increasing and $Q_{des} = 0$, P starts to derail from P_{des} for some periods of time, and eventually becomes unstable.

The first noticeable fact is that DPC makes the power reach its desired value 60 times faster than using the PI Decoupled Control. An advantage of DPC over the PI decoupled control is that it requires neither $\alpha\beta \rightarrow dq$ transformation, nor an inverse transformation. This saves calculation resources in simulation, but more importantly for implementation.

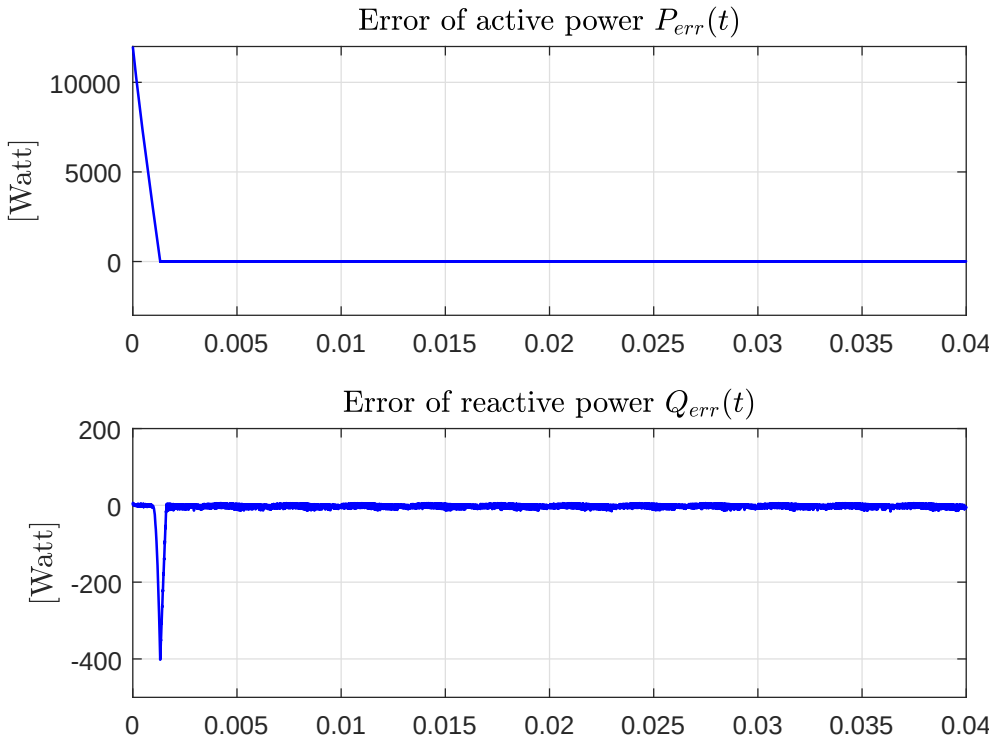


FIGURE 4.33: *Sim 4.* Errors for DPC ($P_{des} = 12$[kW], $\Delta P = 1$, $\Delta Q = 1$, $\Delta C = 0.5$).

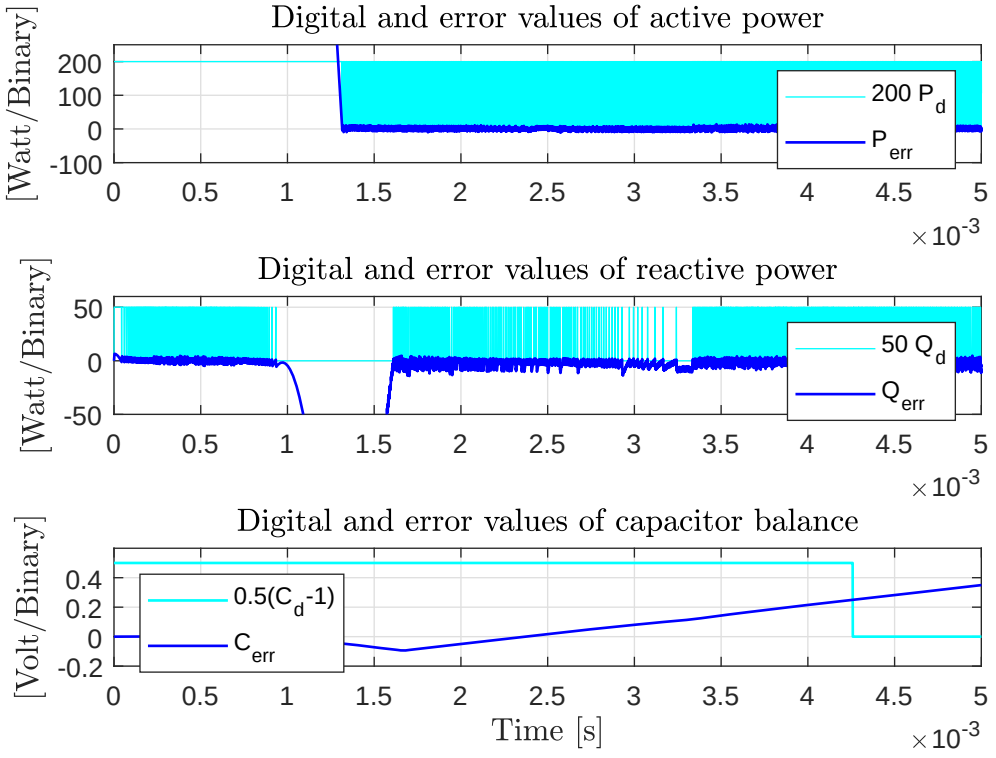


FIGURE 4.34: *Sim 4.* Error and digital (zoom) PQC signals for DPC ($P_{des} = 12$[kW], $\Delta P = 1$, $\Delta Q = 1$, $\Delta C = 0.5$).

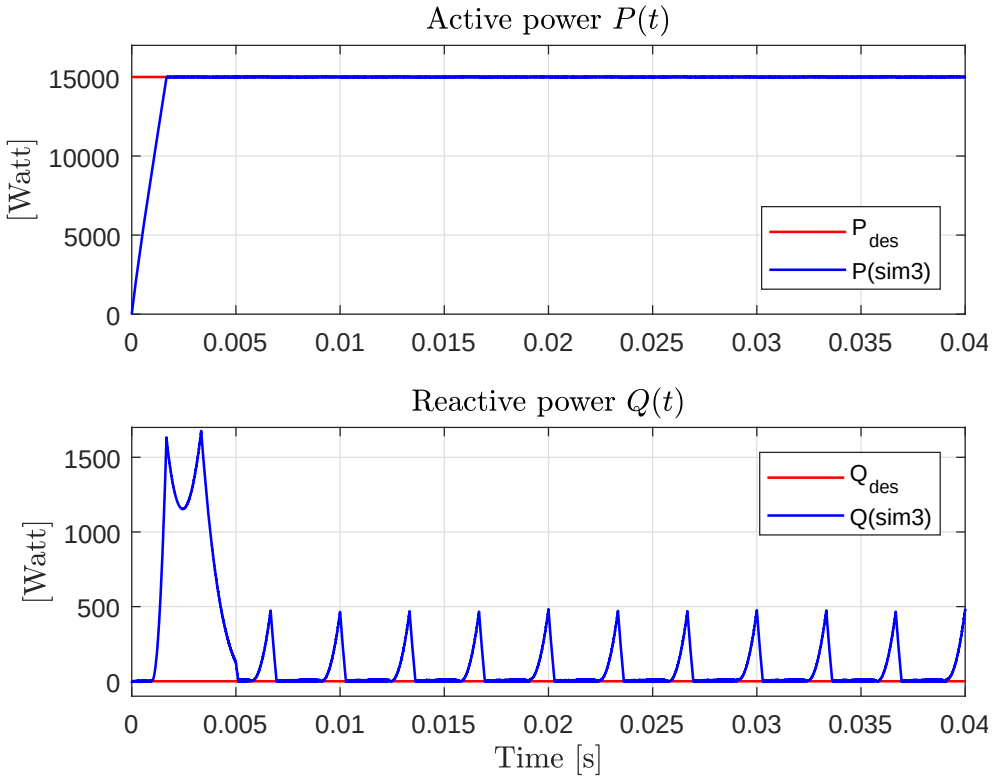


FIGURE 4.35: *Sim 6.* Power for DPC ($P_{des} = 15$[kW], $\Delta P = 1$, $\Delta Q = 1$, $\Delta C = 2$).

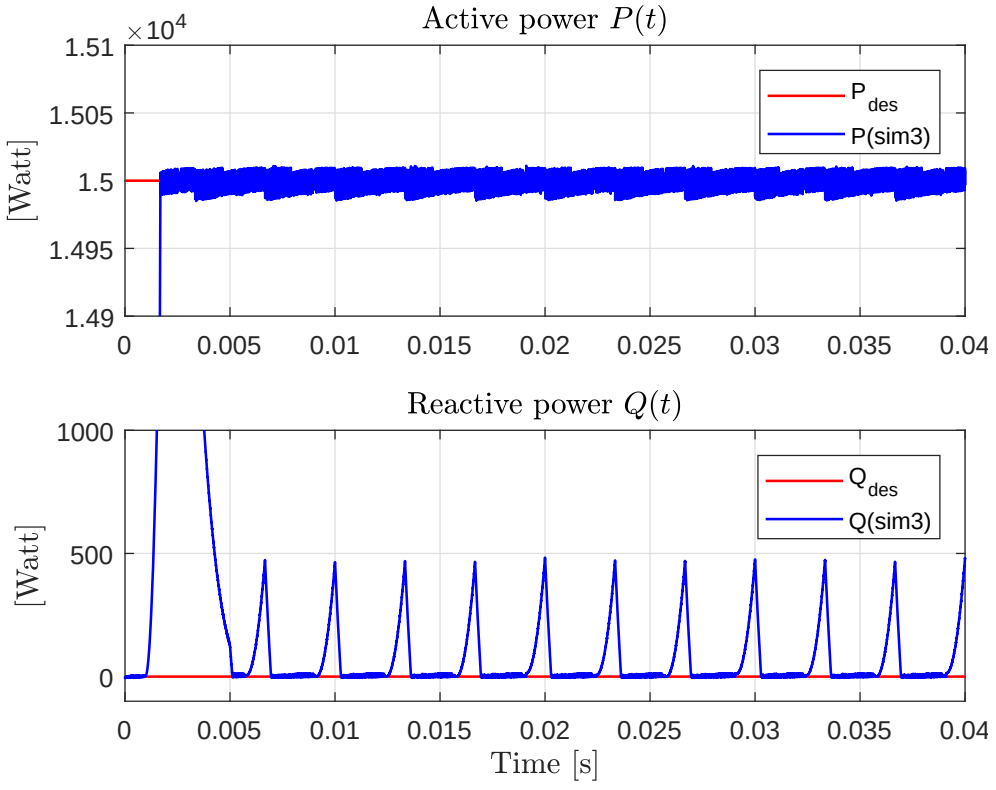


FIGURE 4.36: *Sim 6.* Detail of power for DPC ($P_{des} = 15$[kW], $\Delta P = 1$, $\Delta Q = 1$, $\Delta C = 2$).

Both schemes prioritize P over Q, which is important. The PI decoupled control was simulated with two voltage sources in the DC-link instead of capacitors. It could handle more power without much struggle, but if the PI decoupled control is not carefully tuned, the response of P will be much slower than compared when using DPC.

Lastly, the switching losses can be approximated by measuring the number of pulses per cycle:

$$N_{pulses} = T_g f_s, \tag{4.38}$$

where the grid period $T_g = 0.02$[s] and f_s is the switching frequency. Naturally, the hysteresis gap will directly influence the number of pulses output. The tighter it is, the less ripple and the more switching losses. The wider it is, the more ripple and potentially less losses. It is also important to notice that the DPC scheme is not optimised for smooth switching of s_{abc}. Thus, it is possible to switch from $s_x = [1\ 1\ 0\ 0]$ to $s_x = [0\ 0\ 1\ 1]$, which implies four changes at once. This could be optimised by defining switching patterns.

4.6 Summary

In this chapter two different strategies, DPC and PI Decoupled Control, are compared. Each scheme was explained in detail step by step with particular focus in power control. Briefly, the PI uses a decoupling control to unplug (at some extent) the dependence between active and reactive power, it requires dq transformation and an inverse transformation after processing. It also needs aid of a type of PWM. The DPC requires a design of a switching table ruled by hysteresis values of power, however the only transformation required is to $\alpha\beta$. The new switching table with capacitor balancing was successfully tested in simulations and the convergence to the desired power proved to be much faster than that of the PI decoupled control.

PI decoupled control has longer convergence times, as the decoupling algorithm requires time to deal with the dependency of P and Q, it handles more power and it switched in a controlled manner. DPC has much shorter convergence times, although seems to struggle with higher desired power and it is switched in an uncontrolled manner.

Chapter 5

Experimental Prototype and Results

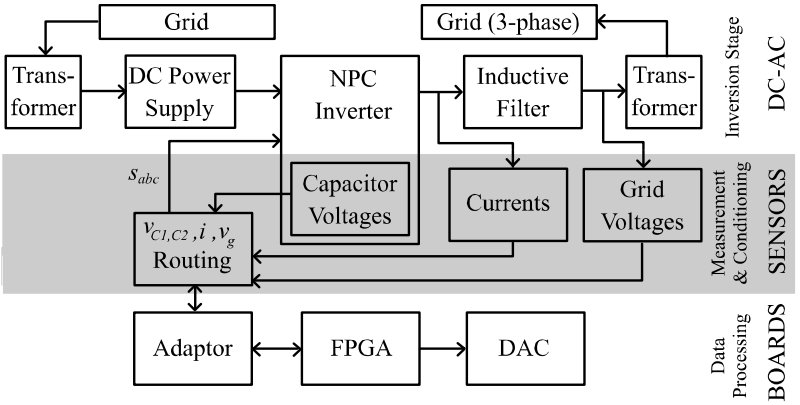


FIGURE 5.1: Block diagram of the experimental setup.

Experimental results are important not only to validate simulation results, but also to test the proposed system under actual conditions and equipment. In this Chapter, the prototype (or experimental bench) built to test the new DPC scheme is described. The prototype is operated using an FPGA board by *Altera*. An FPGA board was used due to its availability in the laboratory resources and its performance is comparable to that of DSPs (Digital Signal Processing Devices). In

fact, other inverters have been successfully connected to the grid using an FPGA board, such as the work in [58].

This Chapter is introduced with the entire experimental bench and list of hardware used. The hardware is described in the three stages shown in Figure 5.1: DC-AC inversion stage, measurement and conditioning, and data processing. Each piece of the equipment used is introduced and detailed information of the construction and operation of the prototype is presented.

Next, software embedded in the FPGA board is described in a block basis. After the required calibration and tests, results are presented and discussed. Observations on data processing limitations and performance are made through the results.

5.1 Hardware

The experimental bench is shown in Figure 5.2. It contains a dashed black rectangle for each of the items to which a number in white background is attached. Each numbered item correspond to the numbers in Table 5.1, where hardware is briefly described. The symbol # will be used througout this Chapter for hardware identification. In the next subsections, the equipment, configuration and connections are detailed.

5.1.1 DC-AC Inversion Stage

A detailed wiring diagram is shown in Figure 5.3. A grid-isolated DC Power Supply feeds DC voltage to the inverter and an NPC inverter transforms the DC voltage to AC voltages according to a switching pattern. Each of the three AC voltages

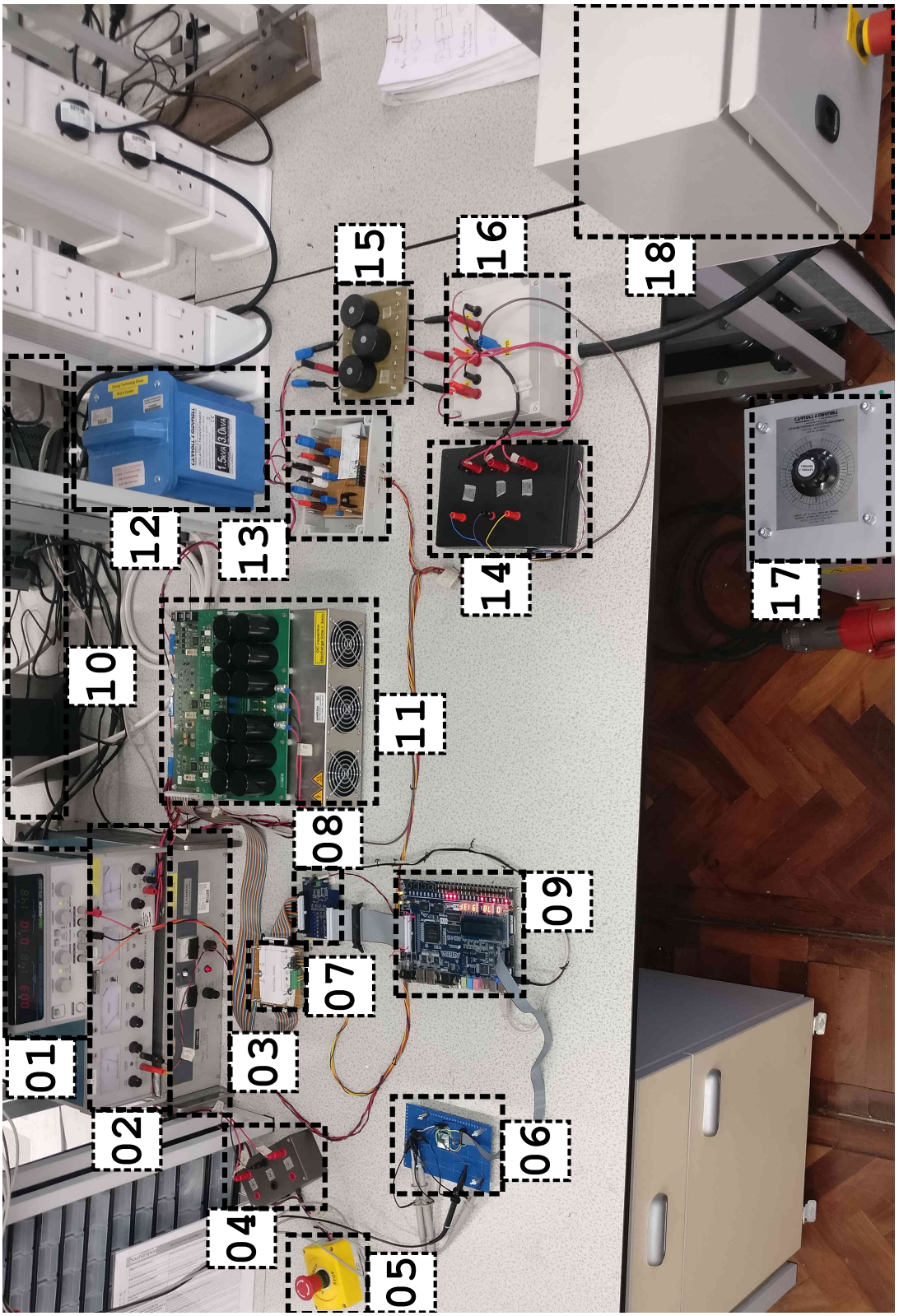


FIGURE 5.2: Experimental Bench

Table 5.1: Corresponding labelled items from Figure 5.2.

#	Item	Model description
01	Sensors power supply	*ISO-TECH IPS2303*
02	Boards power supply	*POWERLINE LAB 522*
03	Inverter power supply	*Lambda LK-352-FM*
04	DC connection case	10[A] fused
05	DC emergency button	*ABB 1RSC-EN*
06	DAC board	*AD7398 serial-input 12-bit resolution*
07	Routing board	Crafted with 3 selectors
08	Adaptor board	*Semikron 3-Level Adapter*
09	FPGA board	*Altera Cyclone IV EP4CE115F29C7*
10	PC for coding	Windows 7 Enterprise & Quartus Prime 16.1 Lite
11	NPC Inverter	*Semikron EVA Inverter*
12	Isolating transformer	*Carrol & Meynell 1.5kVA-CM1500/230*
13	Current sensors	Crafted with *LEM 6-NP* and ICs
14	Voltage sensors	Crafted with *LEM LV 25-P*
15	Inductors	*Schaffner RD5122-10-6M0*
16	AC connection case	With banana sockets
17	Variable transformer	*Carrol & Meynell MVT4.2k/5A-3E*
18	Emergency stop box	*Schneider Electric* with isolating transformer

then pass through an inductive filter to smooth them out and then enter a three-phase transformer to finally make the grid connection. The following description is valid to Figures 5.2 and 5.3.

An isolating transformer *Carrol & Meynell 1.5kVA-CM1500/230* (#12) is connected to the single-phase grid, to which a DC Power Supply is connected. The DC Power Supply (#03) is a *Lambda LK352-FM* capable of handling 60 [V] and up to 15 [A]. In this experiment, the DC power supply is fed by a single-phase AC voltage and outputs a unipolar voltage (0 to +60 [V]). A DC connection case (#04) is used as a connection and measurement point, and its 10 [A] fused output is connected to the DC-side of the NPC inverter. The reference cable (0 [V]) and the $+DC$ voltage of the power supply are connected to the $-DC$ and $+DC$ points of the NPC inverter, respectively. The neutral point of the NPC inverter is floating.

The *3L SKiiP 28 MLI 07E3V1 Evaluation Inverter* or *EVA Inverter* by *Semikron*

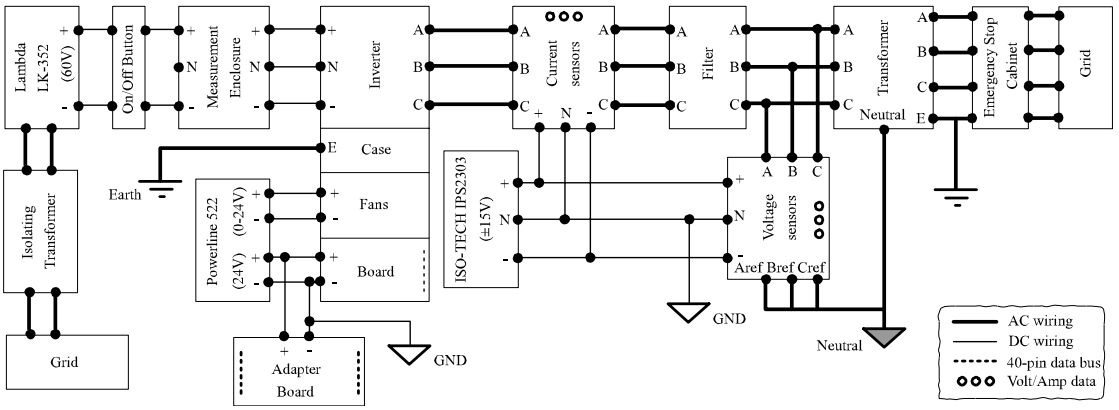


FIGURE 5.3: Power wiring and measurement.

is shown in Figure 5.2 (#11). It has been built using *MiniSKiiP* modules and is capable of handling up to 750 [V] DC input voltage, and 100 [A] RMS for f_{IGBT} = 3 [kHz] or 50 [A] RMS for f_{IGBT} = 20 [kHz]. The inverter lies in a heat sink with forced air cooling. The three fans on the heat sink can be regulated from 5 to 24 $[V]$ to operate from minimum to maximum speed.

The *EVA Inverter* is also equipped with an inverter interface board which requires 24 [V] to operate. Additionally to the IGBT gate drivers, the inverter interface board contains sensors and self-protection circuitry. The sensors in the inverter board measure voltages of the top and bottom capacitors, as well as current and temperature in a leg (or module) basis. The self protection circuitry ensures there is no harmful switching patterns, over–voltage, –current and –temperature. This circuitry also ensures there is a minimum deadtime of 3[μs] for the transistors. The signals for gate triggering, sensors and other status are input/output via a 40-pin connector in the board. These signals are read and written using circuit boards and they will be described in the next subsections.

Relevant parameters of this inverter are presented in Table 5.2. Although this inverter is capable of much higher power, it is used in these experiments at low voltage and current (specified in the results). The voltage output points of the inverter, A, B, C (or U, V, W in the manufacturer's datasheet) are wired to the

TABLE 5.2: Characteristics of the *EVA Inverter* by *Semikron*.

Parameter	Value	Unit
$V_{DC(max)}$	750	[V]
$V_{LL(max)}$	480	[V]
$I_{LL(max)}$	100	$[A_{RMS}]$
C_1	4.08	[mH]
C_2	4.08	[mH]
$f_{IGBT(MAX)}$	20	[kHz]

inductors. Three *Schaffner RD5122-10-6M0* inductors (#15) are used, one for each of the three phases. These inductors can operate at up to 600 [V] AC, up to 64 [A] at 50 or 60 [Hz].

On the other end of the inductors, the three wires of the AC voltages connect to an AC connection case (#16), which in turn is connected to each of the three corresponding windings of a variable transformer (or autotransformer). The three-phase variable transformer *Carrol & Meynell MVT4.2k/5A-3E* (#17) is rated 4.2[kVA] at 467 [V] and 5 [A] maximum per line. Naturally, the transformer is connected to the grid (three-phase) with 415 [V] at 50 [Hz].

The setup has two emergency buttons for extra safety. The DC emergency botton (#05) is located on the DC side, between the *Lambda* Power Supply and the DC connection case, so that the DC Power Supply can be disconnected. The AC emergency button is located on the Emergency stop box (#18) at the AC side, between the transformer and the grid lines, to disconnect from the grid when in an emergency.

5.1.2 Data Processing and Acquisition

The data paths are illustrated in Figure 5.4 and the data processing is formed by three main boards: the FPGA board (#09), the adaptor board (#08) and the DAC board (#06). Starting from the right, voltage and current sensors send

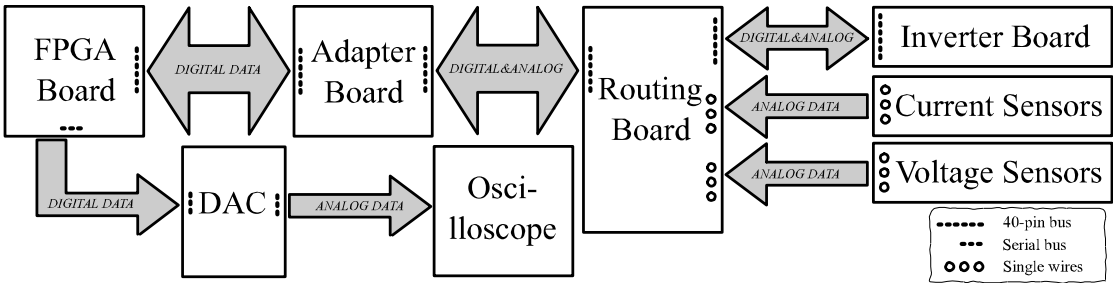


FIGURE 5.4: Data management.

analog readings through the routing board to the adaptor board, while the inverter board and the adaptor board keep a mixed analog and digital commucation channel through the same routing board (including capacitor voltages and fault indicators). The adaptor board then translates the analog signals to digital signals (while bridging digital ones) and sends them to the FPGA board, but it also sends digital and analog signals to the inverter board for transistor switching and fault clearance.

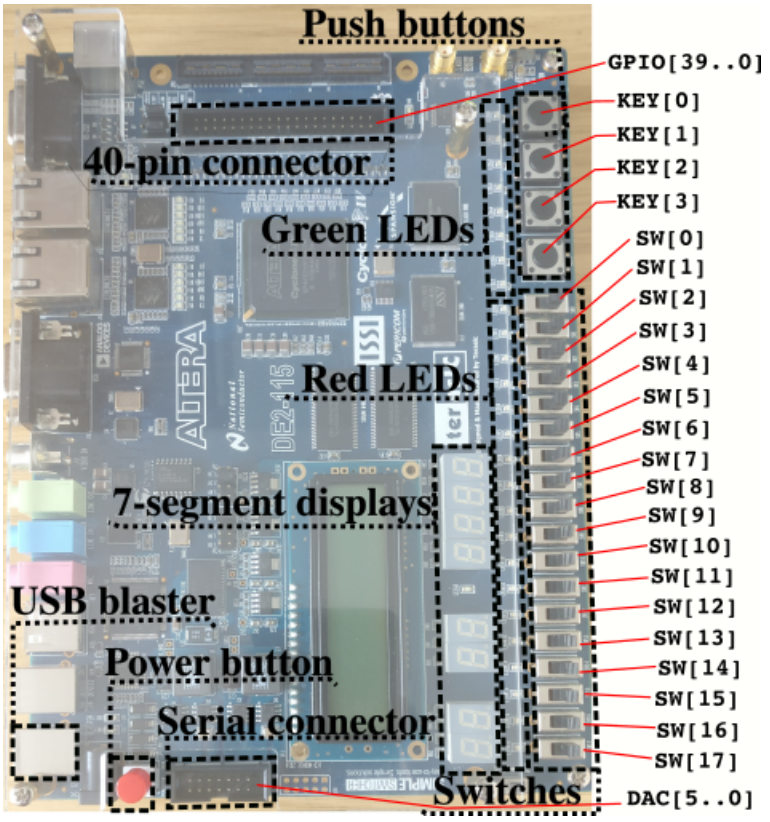


FIGURE 5.5: Altera FPGA

The FPGA board used is the *Altera Cyclone IV EP4CE115F29C7* shown in Figure 5.5, which has a processing speed of 50 [MHz], a 40-pin expansion header, 8 7-segments displays, 18 slide switches, 4 push buttons, and 1 RS-232 connector, among other features. It comes with its own power supply (12 [V]) and a USB Blaster port.

The software *Quartus II* was used to code and compile the DPC algorithm in VHDL (Hardware Description Language) with *.vhd* files and Verilog with *.v* files. The compilation, performed in a PC (#10), results in a file with extension *.sof*, which is then uploaded from the PC to the FPGA board via its USB Blaster port. The operation of the FPGA board is highly configurable and customisable as the supported languages offer significant freedom to the programmer. The software coded is described in Section 5.2.

Once the FPGA has been loaded with the code, it should be capable of reading digital inputs and writing digital outputs via the 40-pin expansion header (also called GPIO connector, from *General Purpose Input-Output*) in real time. Given the digital signal nature of the board and its capabilities, the 'high' state has been configured to 3.3 [V]. Briefly, from the FPGA perspective, outputs are s_{abc} (12 signals) and inputs are v_{gabc}, i_{abc}, v_{C1}, v_{C2} (8 signals), excluding the pins dedicated to fault indicators and protection.

The GPIO connector then goes to the *Semikron 3-Level Adapter* or adaptor board using a customised 40-pin ribbon cable with testing points. This board is specifically designed as an interface between the *EVA Inverter* and a the FPGA board. Its parts are shown in Figure 5.6 and correspond to the following bulletpoints. Further diagrams of each are found in Appendix B and a layout of 40-pin connectors is provided at the end of the following subsection.

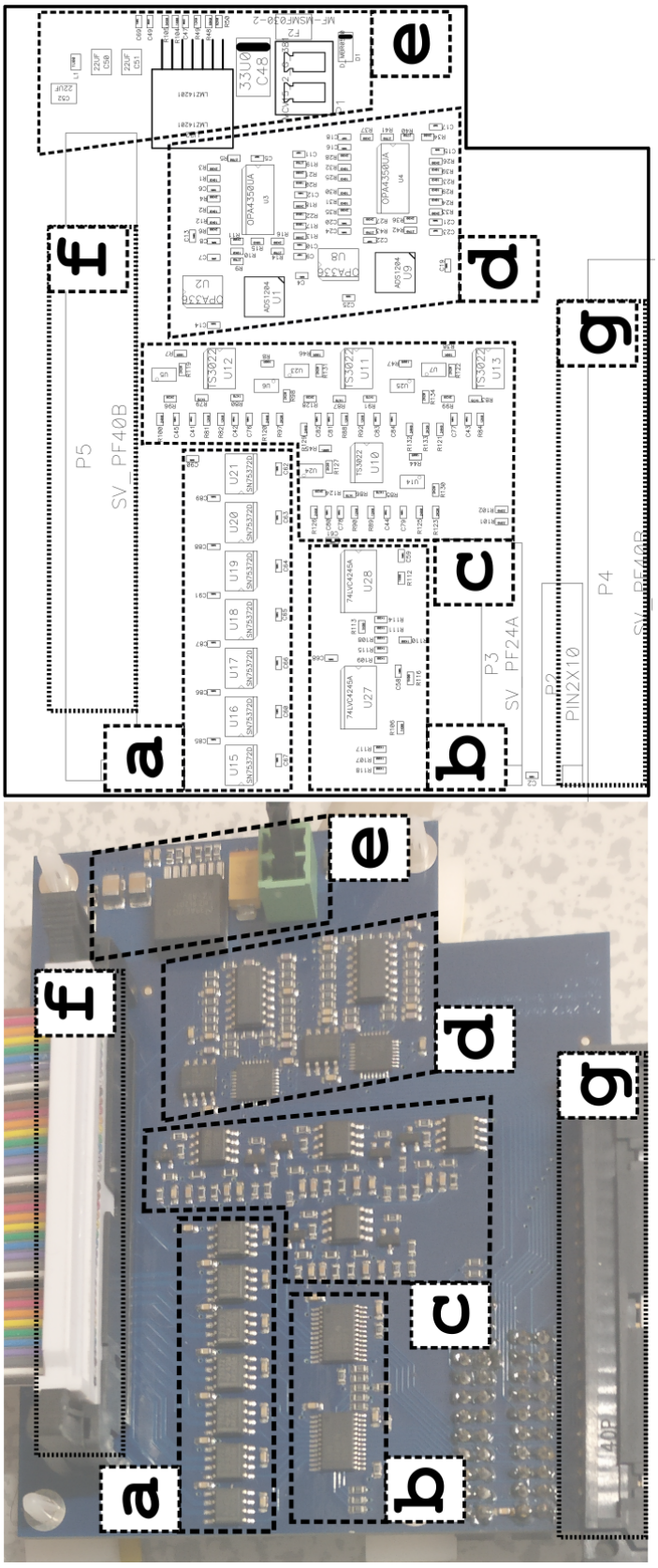


FIGURE 5.6: Parts of the adaptor board.

a) Output driver. Seven dual mosfet drivers SN75372 serve as the IGBT drivers for the inverter. They are configured to output a 12 [V] signal when the 5[V] TTL signal on the low side is high. The drivers have a propagation delay of about 20 [ns]. The time delays from low to high are typically 20 [ns] and 35 [ns] maximum, whereas from high to low are typically 10 [ns] and 20 [ns] maximum.

b) FPGA interface. As the FPGA output signals to the inverter's transistors are 3.3 [V], this interface is responsible for increasing the voltage to 5 [V] with two 74LVC4245A in order to reach the low side of the output driver above.

c) Signal conditioning. An arragement of OPAMPs (TS3022) and MOSFETs (2N7002) configured to decrease voltage signals from the inverter board offers signal conditioning from 12 [V] to 3.3 [V] for the FPGA board. These signals are reserved for status and fault management coming from the inverter board.

d) Analog to Digital Converter (ADC). The analog input voltages from voltage (v_{C1}, v_{C2}), current (i_b, i_b, i_c) and temperature measurements (T_a, T_b, T_c) pass through OPAMPs for conditioning from a $\pm$ 10 [V] range to a +5 [V] range. These signals travel into the $\Sigma\Delta$ modulators (ADS1204) with reference voltage $v_{ref} = 2.5$ [V], so that inputs are $\pm$2.5 [V] with linearity and noise performance kept in the $\pm$ 2 [V] range. The modulators are fixed to use an external oscillator ($CLKIN$) from the FPGA board of up to 32 [MHz] and can be configured to a certain extent with a digital filter, which is provided in Section 5.2.

e) Power supply. The board receives 24 [V] and makes a 5 [V] source available with an LMZ14201. It manages three-voltages internally: 12, 5 and 3.3 [V].

f) GPIO connector to inverter board via routing board.

g) GPIO connector to FPGA board.

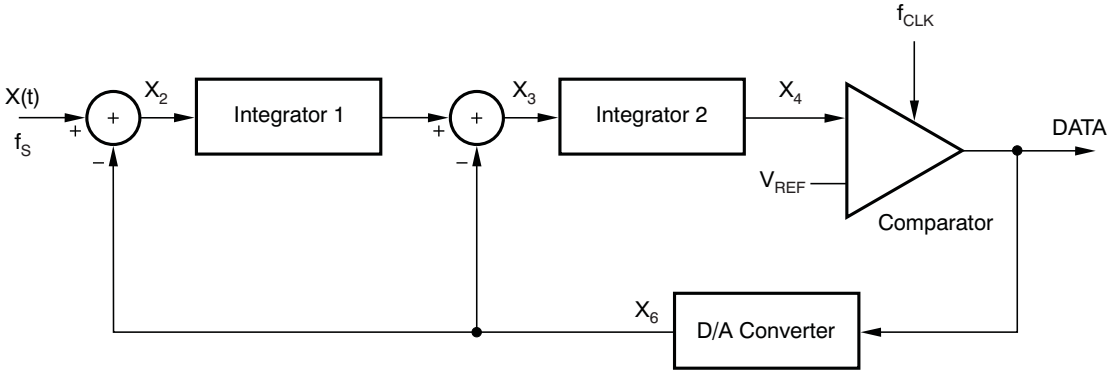


FIGURE 5.7: Block diagram of the 2nd-order modulator ADS1204 [59].

A sensitive part of the adaptor board is the ADC (d) due to its software or digital filter dependency. It becomes important to know further configuration of this part before reaching its software part. Figure 5.7 conceptualises the topology of the second-order, switched-capacitor, $\Delta\Sigma$ modulator. Input voltages are differentiated two times with the D/A Converter voltage, where both voltages are analog. This results in analog voltages X_2 and X_3 directed to their respective integrators, which progress in a negative or positive direction. When X_4 equals the reference voltage in the comparator, this switches from negative to positive, or positive to negative, depending on its original state. The 1-bit D/A Converter then changes its analog output X_6 on the next clock pulse, causing the integrators to progress in the oposite direction. Consequently, the feedback forces the integrator output to track the average of the input.

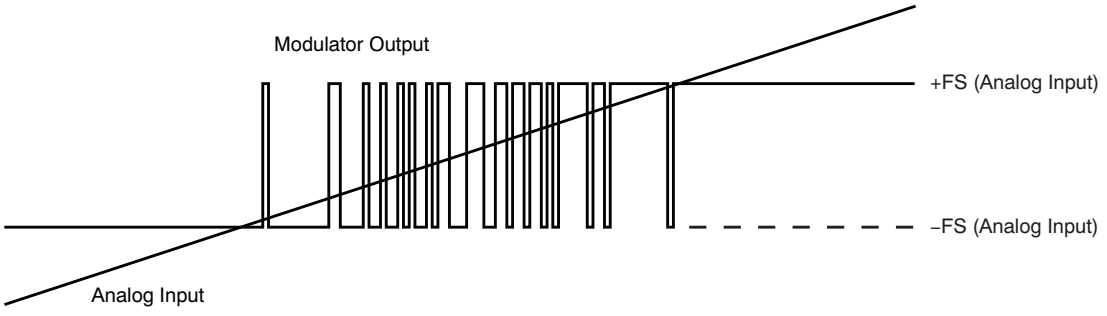


FIGURE 5.8: Analog input vs modulator output of the ADS1204 [59].

The input voltage versus output modulator signal is shown in Figure 5.8. A 0 [V]

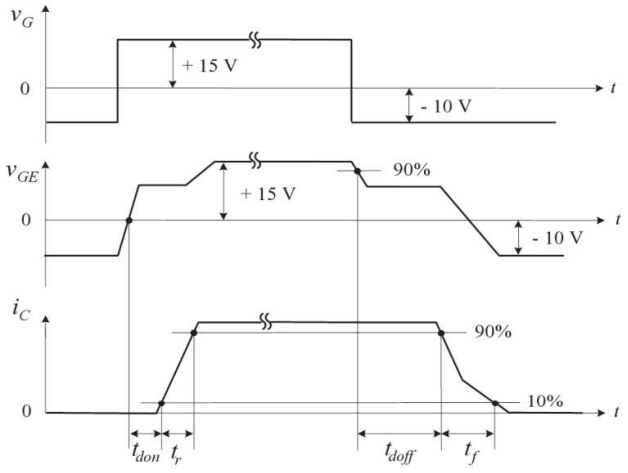


FIGURE 5.9: IGBT switching transition [60].

input will produce a stream of ones and zeros at 50 %, while a differential input of +2 [V] or -2 [V] will produce a stream of ones and zeros that are high 80 or 20 % of the time, respectively. For the modulator, noise increases with frequency and the greater the order of the modulator, the closer that quantization approaches the Nyquist frequency.

A DC power supply at 24 [V] *POWERLINE LAB 522* (#02) is connected to the adaptor board and the inverter board as shown in Figure 5.3. When the signals reach the inverter they go through the programable devices (CPLD) of the inverter board, which has the following characteristics. All deadtimes are set to 3 [μs] (can be programmed longer, but not shorter). It supervises the switch-on and switch-off sequences of the IGBT, the dead times, and the PWM sequences. The SKiiP 28MLI07E3V1 reports the characteristics in Table 5.3 in a transistor pair basis for values at $V_{CE} = 300[V]$, $I_C = 150[A]$, $V_{GE} = \pm 15[V]$, $R_{Gon} = 3[\Omega]$, $R_{Goff} = 1.6[\Omega]$. Semikron does not provide more detailed information on the CPLD and delays. Figure 5.9 illustrate the definition of switching times in IGBTs for reference.

As it is not possible to directly read the digital data in real time from the FPGA alone, a Digital-Analog Converter (DAC) is used. A DAC7398 was used for data display, and a reference diagram of it can be found in Appendix B. The DAC

TABLE 5.3: Timings of transistors through CPLD of the inverter board.

Parameter	T1/T4	T2/T3	Unit
Turn-on delay t_{don}	108	106	[ns]
Rising time t_r	73	64	[ns]
Turn-off delay t_{doff}	268	268	[ns]
Fall time t_f	76	77	[nF]

contains four 12-bit voltage outputs and mantains an output voltage that is the same polarity as the input reference voltage. It can work with nearly any clock frequency with maximums of 11 [MHz] if supplied with 3 to 5 [V], or 16.6 [MHz] if supplied with ±10 [V]. This frequency, as well as the power and four data channels are supplied from the 14-pin serial connector by the FPGA board. This means that signals such as v_{abc}, $v_{\alpha\beta}$, i_{abc}, $i_{\alpha\beta}$, θ, P and Q can be read with the appropriate VHDL code with the help of an oscilloscope.

5.1.3 Measurements, Conditioning and Enhancements

This stage is for acquisition, measurement and conditioning of voltage and current signals. For the test of the DPC scheme in this experiment, low current and voltage values are chosen (less than a 500 [W] output), and internal sensors of the inverter board may not be designed for proper resolution of these low values.

The internal pair of voltage sensors for capacitors in the inverter board provide a linear gradient signal (voltage) with the range of 0 to 10 [V], starting from 0 [V] and the scale 10 [V] is equivalent to 500 [V]. They have a tolerance of ±2 % with regard to the peak voltage measurable 500 [V]. This means that for every unit of measurement (1 [V]) sensed, there is a real voltage change of 50 ±10 [V] (i.e. a ratio of 50:1 with ±10 [V] error margin).

The three internal current sensors in the inverter board (one per phase) provide a linear gradient signal (voltage) with the range of 0 to 10 [V], starting from 0 [V]

and the scale ±10 [V] is equivalent to ±187.5 [A] with a tolerance of ±3 % with regard to the peak current measurable ±187.5 [A]. This means that for every unit of measurement (1 [V]) sensed, there is a real voltage change of $18.75[A] \pm 5.625$ [V] (tolerance)(i.e a ratio of nearly 1:19 with a considerable margin of error).

Although capacitor voltage sensors (internal) in the inverter board are low-resolution, these are enough to provide voltage equilibrium between the two set of capacitors, particularly for such large capacitor size and the low power requirements set later in the experiments. In this case, these sensors are powered with the same 24 [V] of the inverter board. However, internal current sensors are found to be inadequate to handle these low current values (under 6 [A]) as they will provide very poor resolution. Furthermore, the DPC algorithm coded in the FPGA board requires grid voltage measurements. Therefore, as more accurate voltage and current measurements need to be incorporated in the system, the following enhancements have been crafted and installed:

- External voltage sensors
- External current sensors
- Routing board

A set of voltage sensors (#14) has been crafted using three *LEM LV 25-P* transducers. These have been easily conditioned to measure ±120 [V] and output a nearly-linear proportional value of ±10 [V].

A set of current sensors (#13) has also been crafted and conditioned as shown in Figure 5.10. Three current transducers *LEM LTSR 6-NP* (in blue) properly configured for measuring within a range of ±6.4 [A] were installed. The sensors need to be supplied with +5 [V] and the measured current (positive or negative) is proportional to a voltage level in the range of +0.5 to +4.5 [V], such that a measurement of 0 [A] outputs a signal of approximately 2.5 [V].

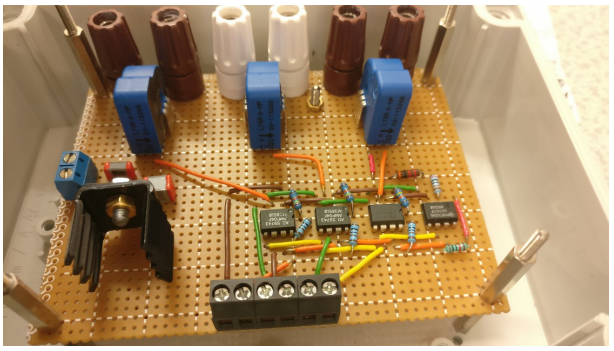


FIGURE 5.10: Current sensor.

As mentioned in subsection 5.1.2, the adaptor board has a range of ±10 [V] for incoming analog signals, and acquiring signals in the range of 0.5 - 4.5 [V] poses a resolution problem. In other words, the lack of range reduces the accuracy of current reading, which is also affected by noise. Consequently, the 'raw' signals from the current sensors need to be conditioned.

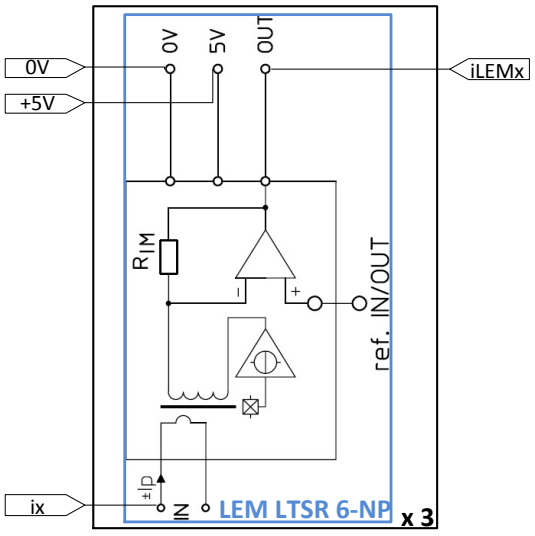


FIGURE 5.11: Diagram of current sensor *LEM LTSR 6-NP* (datasheet)

A datasheet diagram of the current sensors used is shown in Figure 5.11. It has been labelled with *0 V* and *+5 V* on the left, which is its power supply. *ix* is on the left bottom representing the real current flowing through the sensor. Output *iLEMx* is on the right with the aforementioned 0.5 -4.5 [V] range. For the conditioning, each current signal has to be shifted and amplified.

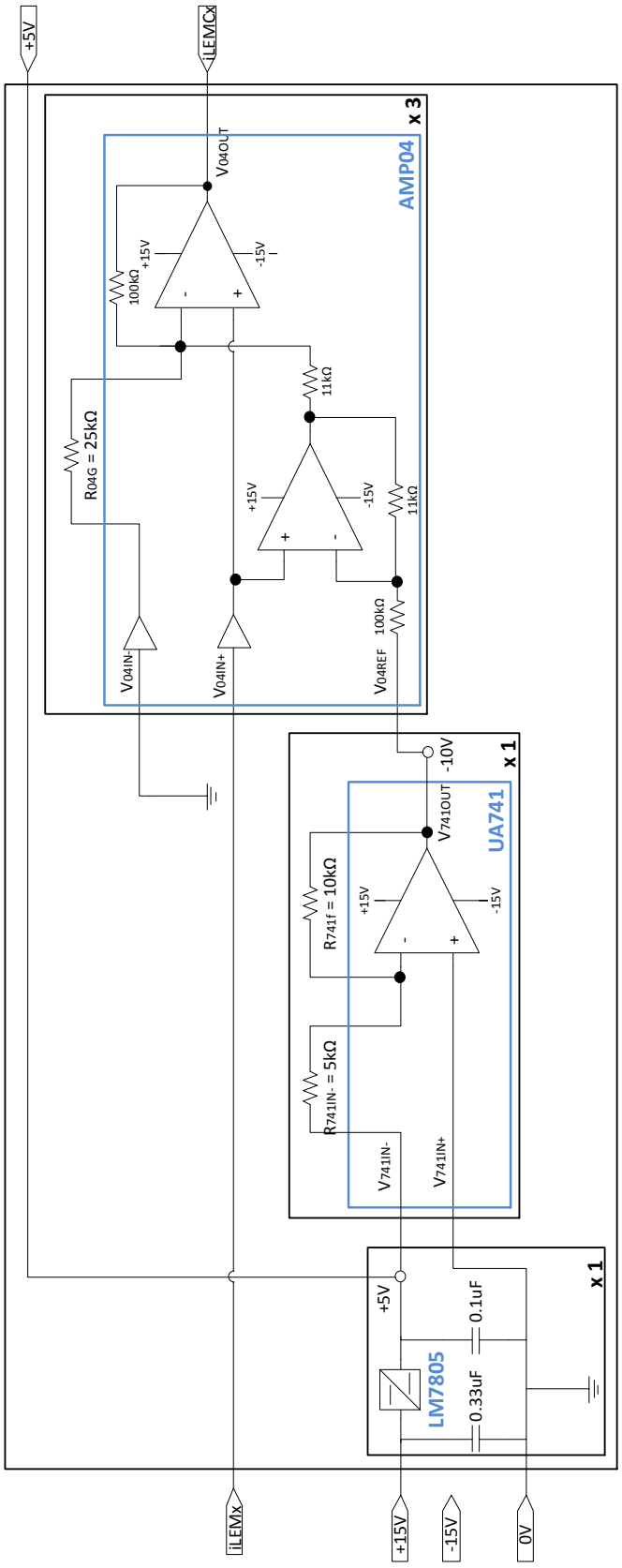


FIGURE 5.12: Signal conditioning for current transducers.

The signal conditioning work for the current sensor set is shown in Figure 5.12. It comprises three steps from left to right, with inputs on the far left +15, 0 and -15 [V] as power inputs and *iLEMx* the input signal coming from the sensor. First, a positive voltage regulator *LM7805* was used to reduce a +15 [V] input to +5 [V], which allows the use of one power supply for the two external sensor sets. This +5[V] becomes an output (placed in the far right) to power the current sensors in Figure 5.11. Second, The operational amplifier *UA741* is configured as an inverting amplifier with the +5 [V] from the voltage regulator as an input and a gain of 2. The output is a reference voltage of -10 [V]. This is

$$Gain_{741} = -R_{741f}/R_{741IN-}, \tag{5.1}$$

$$V_{741OUT} = (V_{741IN+} - V_{741IN-})Gain_{741}, \tag{5.2}$$

with $R_{741f} = 10\ [k\Omega]$, $R_{741IN-} = 5[k\Omega]$, $V_{741IN+} = 0$, and $V_{741IN-} = 5$.

Third, three instrumentation amplifiers *AMP04* are configured as differential amplifiers with a gain of 2, having the current signal (i_{LEMx}) from the sensors and the 0 [V] (or neutral line) as inputs. Then, the -10 [V] is fed as reference voltage for shifting. This is

$$Gain_{04} = 100[k\Omega]/R_{04G}, \tag{5.3}$$

$$V_{04OUT} = (V_{04IN+} - V_{04IN-})Gain_{04} + V_{04REF}, \tag{5.4}$$

with $R_{04G} = 25[k\Omega]$, $V_{04IN+} = i_{LEMx}$, $V_{04IN-} = 0$, and $V_{04REF} = -10[V]$. Thus, $V_{04OUT} = iLEMx$, where 0 [V] $< iLEMx <+$ 10 [V] and $iLEMx = [iLEMa \ \ iLEMB \ \ iLEMc]$.

The Power Supply for the two sets of sensors is an *ISO-TECH IPS2303* (#01) with three voltage terminals: $+15, 0, -15$ [V]. Now that the six signals corresponding to v_{gabc} and i_{abc} have been produced, they need to be injected into the system.

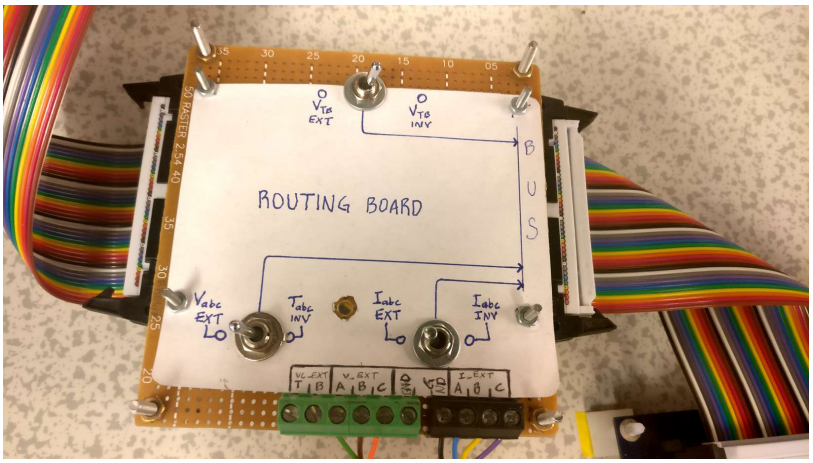


FIGURE 5.13: Routing board

A Routing Board (#07) has been crafted as pictured in Figure 5.13 to customise signal routing (see Figure 5.4 for reference). As temperature signals (T_{abc}) are not of utmost importance due to the low power level handled, these inputs have been used to read the grid voltage instead, while the external current signals have replaced the internal current signals. In other words, the 40-pin ribbon cable connecting the Adaptor Board and the Inverter Board has been *hacked* replacing: T_{abc} (temperature) with v_{gabc} and i_{abc} (internal) with i_{abc} (external). Internal voltage measurements for capacitors are kept. However, original temperature and current signals can be recovered. Should there be external voltage sensors to measure with more precision, they can also be easily integrated with the board by switching to $V_{TB}(INT)$ or $V_{TB}(EXT)$.

A detailed and precise diagram solely for the purpose of signal and pin identification through the 40-pin ribbon cable has been drawn in Figure 5.14. This is a top (or aerial) view of the layout, where connectors which are not flat, specify *TOP/UP* and *BOTTOM/DOWN* in grey colour. The central connector is actually a testing point within the 40-pin bus cable and the blue dashed line represents the actual 40-pin bus cable. The Routing Board at the top left contains signals in bold font *IA**, *IB**, *IC**, *VA**, *VB**, *VC**, *VTP**, *VTP** with an asterisk to note

those are the signals selected for these experiments, but to remind they can be manually re-routed with the switches on the board to acquire a different signal.

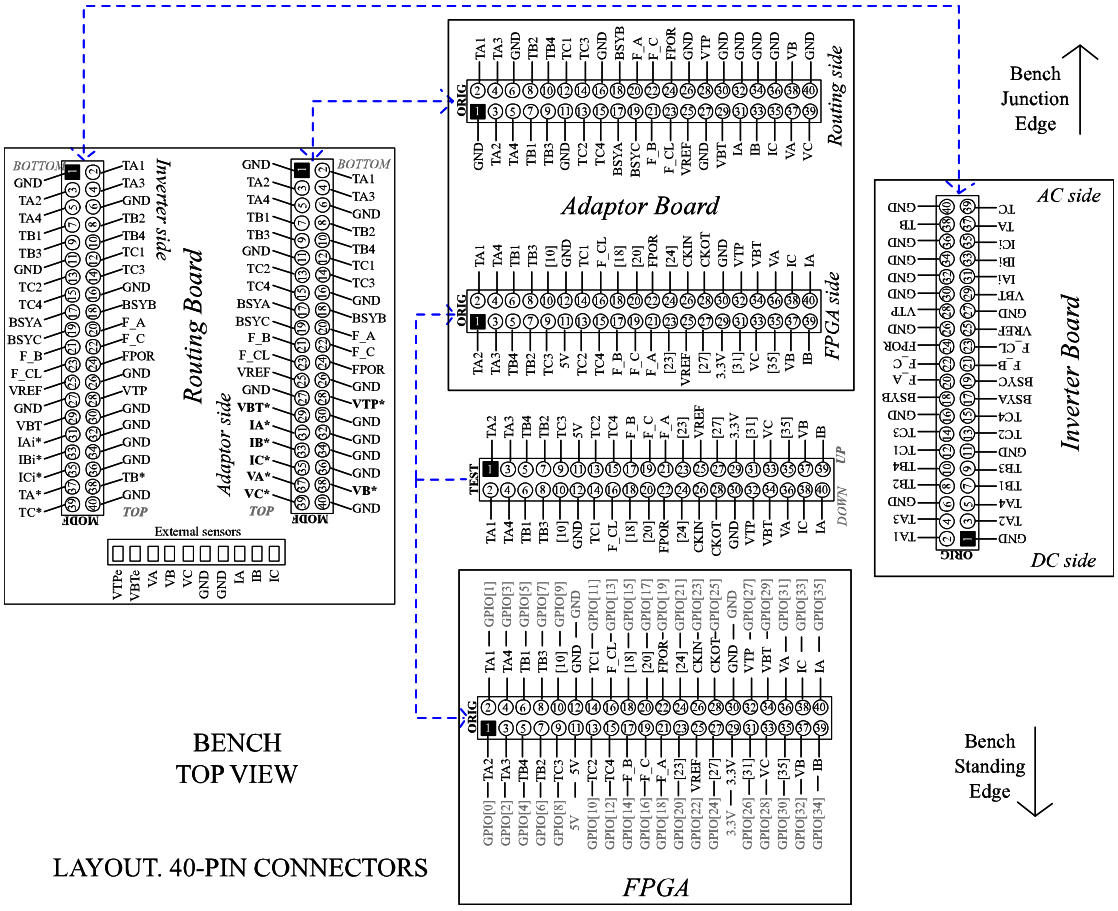


FIGURE 5.14: Layout of the 40-pin connectors

For more accuracy, calibration tests were carried out in section 5.3. However, the proper software has to be in place in order to acquire, process and read measurements. This will naturally be followed by further signal processing in order to design the right architecture for this prototype.

5.2 Software

The FPGA board was coded in VHDL (*.vhd* files) and verilog (*.v* files) languages using *Quartus Prime 16.1 Lite* software, referred as *Quartus* here. Figure 5.4

illustrates how the FPGA board interacts with the whole system. First, a code is developed in VHDL using *Quartus* and after compilation, a file *.sof* is created. Next, this file is uploaded to the board from a computer (PC) offline. Once complete, there is no communication between the PC and the FPGA board. Then, provided all hardware is set, the FPGA algorithm is turned on (with the *reset* switch `SW[0]` shown in Figure 5.5) and the operation of the algorithm starts in real time.

The big picture is composed of the following subsections in a *Quartus* project file (*.qsf*). Each subsection of the project is described and include graphics with the corresponding embedded code. The code as well as nomenclature in the graphics of *Quartus* software is found in `verbatim` font. As the programming language was learnt *on the way*, this code is not optimised in any way. The functions, nevertheless, work as intended.

5.2.1 Clocks and Internal Signals

Internal signals are signal definitions (e.g. $highbit = 1$) and clock signals associated with an oscilator in an specified address inside the board. The rest of the subsections will inevitably have other signal definitions or names, but will be described according to their purpose. Figure 5.15 show a label `HI[31..0]` representing 32 bits made entirely of logic `1` and a label `GND[31..0]` representing 32 bits made entirely of logic `0`. This is convenient for pre-testing code or when data lenght missmatches are presented during coding.

The board contains three internal clocks with a maximum speed of 50 [MHz] and they only operate when the FPGA board is on. These clocks have reserved names, which are assigned in the internal signals part and used in the clocks configuration part. In the latter, the clocks are locked to a desired speed using a *wizard* feature in *Quartus*. These locked clocks are called `PLL` in this programming

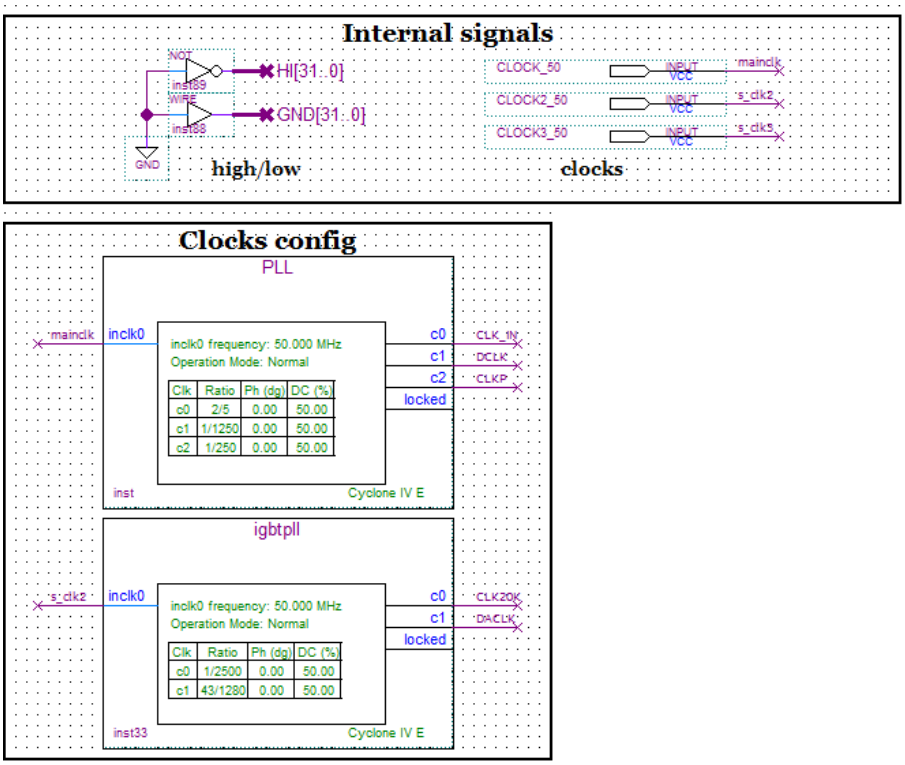


FIGURE 5.15: *Quartus*: Clocks and internal signals

environment, but they will be called simply 'clocks' in this thesis to avoid confusion with the aforementioned PLL of the grid. The *wizard* is then configured to lock a desired clock speed by assigning a division factor (or ratio) from the core 50 [MHz] frequency. Therefore, there are five defined clocks:

```
CLK_IN = 2/5 ratio = 20 [MHz]
DCLK = 1/1250 ratio = 40 [kHz]
CLKP = 1/250 ratio = 200 [kHz]
CLK20K = 1/2500 ratio = 20 [kHz]
DACLK = 43/1280 ratio = 1,679,687 [Hz]
CLK_OUT1 = 10 [MHz]
```

All clocks in the list are generated internally in the FPGA board, except for the last one which is an external clock defined in the following subsection. All the following block descriptions have at least one of these clocks specified in its input.

5.2.2 I/O Signals and LEDs

Looking back at Figures 5.5 and 5.14, here is where the inputs and outputs interact with the software. The manual inputs of the FPGA board are buttons and switches, shown over *IN: buttons* and *IN: switches* in Figure 5.16, respectively. Push buttons are indexed with the reserved word `KEY`, while switches are with the reserved word `SW`. These reserved words indicate the exact address of the signals inside the board, which will be read. All input signals are appropiately labelled for further usage in the software. The signal `reset` works as an activator of the algorithms as well as an standard reset switch. Other relevant signals are `VI[9..0]` corresponding to `SW[1]` through `SW[10]`, and `pause` corresponding to `KEY[1]`. `SW[11]` has been reserved for fault clearing only, explained below. The rest of the switches and buttons were used for pre-testing, calibration and troubleshooting.

Moving to the `GPIO` port in Figure 5.16, the output signals towards the transistors of the inverter are over *OUT:GPIO*. Specific `GPIO` indices are named `TA1` for the top transistor of leg A and down to the bottom transistor `TA4` of the same leg, following the same transistor order for legs B and C. The signals are negated as it was found the adaptor board had the 'low' and 'high' bits reversed for these particular signals.

Over *I/O:GPIO*, `CLK_IN` (specified in the previous subsection) is assigned as an output, which is required by the ADC in the adaptor board. The adaptor board returns `CLK_OUT1` (also specified in the previous subsection) which inputs the FPGA board and is used later in the algorithm. The ADC in the adaptor board is responsible for translating analog measurements of currents, grid voltages and capacitor voltages into 'raw' digital signals. Further over *I/O:GPIO*, digital input signals of capacitor voltages (`VTOP,VBOT`), grid voltages (`V_A,V_B,V_C`) and currents (`I_A,I_B,I_C`) are assigned.

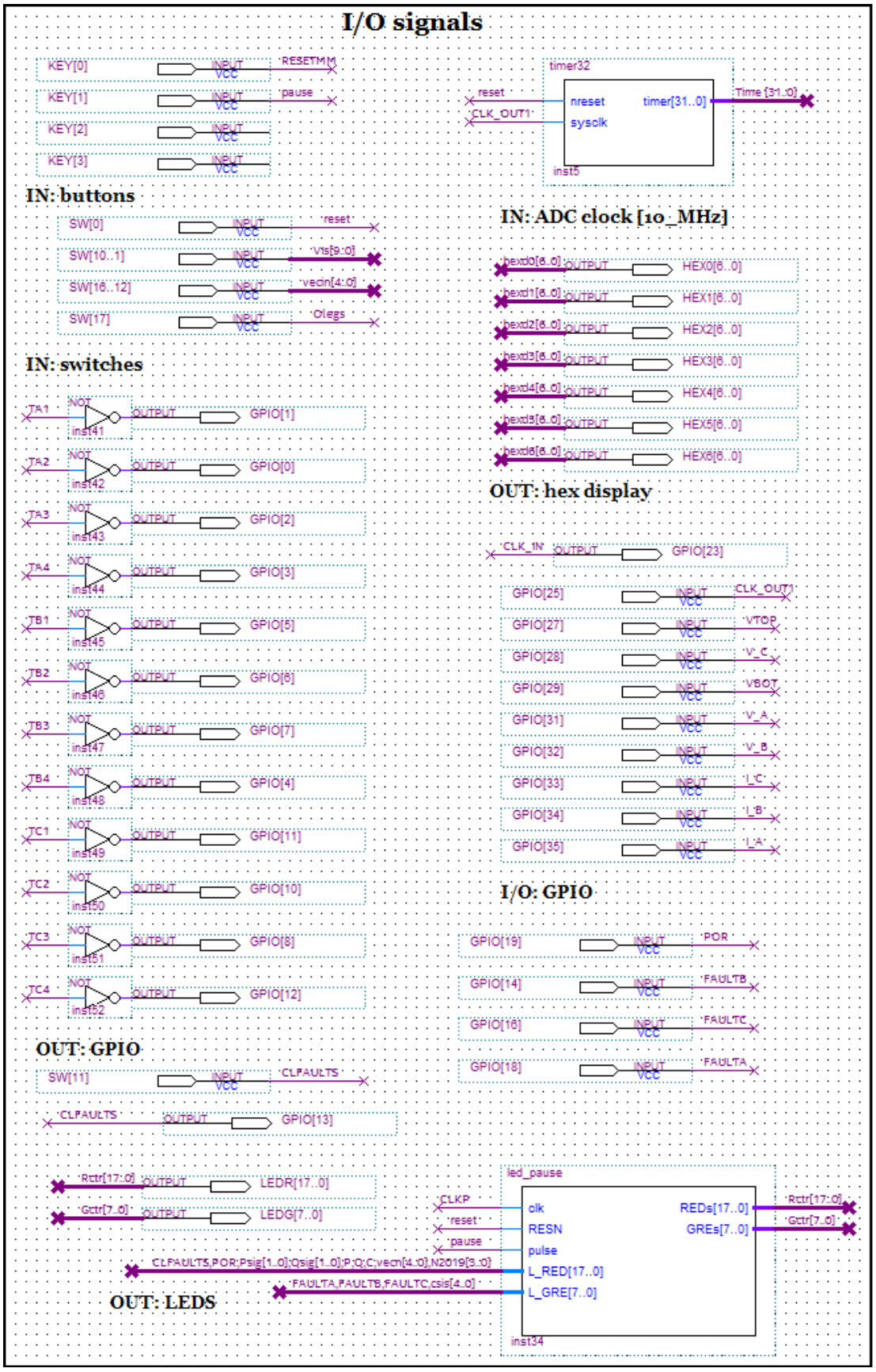


FIGURE 5.16: *Quartus*: I/O signals and LEDs

When the inverter detects an error or a value exceeds the allowed operation limits, it locks itself completely out of operation and the inverter board shows a red *UVLO LED*. The fault might be caused by over–current, –voltage, –temperature or even prohibited switching patterns for the transistors. The inverter board warns of a fault via dedicated `GPIO` indices, which are inputs in the FPGA board, assigned under *I/O:GPIO*. These faults are labelled `FAULTA, FAULTB, FAULTC, POR`, and turn to logic `1` when a fault occur while the inverter locks out of operation.

When this happens, a reset signal has to be sent to the inverter board to clear the faults and turn the inverter back to operational mode. This is the purpose of the manual input `SW[11]` shown under *OUT:GPIO* and named `CLFAULTS`. When switched on and off, a logic 1 and 0 outputs the FPGA board to clear the faults and taking it back to protected operation.

`CLFAULTS` must be off in the FPGA board for the inverter to operate in protected mode. If the faults have been cleared and `CLFAULTS` is left on (switch `SW[11]` left on) in the FPGA board, the inverter operates in unprotected mode (without the protections described above), which poses risks and potential damage. Note all `GPIO` signals correspond to the layout of the 40-pin bus shown earlier in Figure 5.14.

The red and green LED outputs in the FPGA board are indexed over *OUT: LEDS*. Each red LED is located next to a switch in the FPGA board (see Figure 5.5) and are indexed the same. The eight green LEDs are indexed in the same direction and are placed next to the push buttons. As the different variables switch so fast for the human eye to see, a `led_pause` block has been designed with the code below. Signals of interest during code building can then be routed to its red and green LED inputs (`L_RED[17..0]` and `L_GRE[7..0]`, respectively) so that when pushbutton `pause` is pressed, the LED outputs do pause for display (while the algorithm keeps running).

```
-- led_pause.vhd --
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity led_pause is
port (
-- Input ports
   clk      : in std_logic;
   RESN     : in std_logic;
pulse : in std_logic;
L_RED : in std_logic_vector(17 downto 0);
L_GRE : in std_logic_vector(7 downto 0);
-- Output ports
REDs : out std_logic_vector(17 downto 0);
GREs : out std_logic_vector(7 downto 0)
   );
end entity led_pause;
architecture visual2 of led_pause is
signal zeror    : std_logic_vector(17 downto 0);
signal latchr   : std_logic_vector(17 downto 0);
signal zerog    : std_logic_vector(7 downto 0);
signal latchg   : std_logic_vector(7 downto 0);
begin
zeror <= "000000000000000000";
zerog <= "00000000";
process(clk,RESN,pulse)
begin
if RESN = '0' then
       REDs <= zeror;
 GREs <= zerog;
   elsif rising_edge(clk) and RESN = '1'
 and pulse = '1' then
REDs <= L_RED; latchr <= L_RED;
GREs <= L_GRE; latchg <= L_GRE;

elsif rising_edge(clk) and RESN = '1'
and pulse = '0' then
REDs <= latchr;
GREs <= latchg;
end if;
end process;
end visual2;
--
```

Finally, above *IN: ADC clock* in Figure 5.16, a `timer32` block was built. The code inside this block is shown below and its only function is to count the number of pulses from the `CLK_OUT1` clock in 32 bits, and then repeat. This allows to slow-down sample time according to the bit of `Timer[31..0]` chosen.

```
-- timer32.vhd --
-- 32 bit up counter, that wraps around.
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity timer32 is
port (
   nreset   : in std_logic;
   sysclk   : in std_logic;
   timer    : out std_logic_vector(31 downto 0)
   );
end entity timer32;
architecture first of timer32 is
   signal count_lsb  : unsigned (7 downto 0) := "10000000";
   signal count_msb  : unsigned (7 downto 0) := "10000000";
   signal count_nsb  : unsigned (7 downto 0) := "10000000";
   signal count_osb  : unsigned (7 downto 0) := "10000000";
   signal cnt_rst    : std_logic := '0';
   signal carry_lsb  : std_logic;
   signal carry_msb  : std_logic;
   signal carry_nsb  : std_logic;
begin
--------------------------------------------------------------------------
-- generate a counter reset when n_rst goes high.
--------------------------------------------------------------------------
cnt_rst_gen: process (sysclk)
   variable n_rst_d  : std_logic;
begin
   if rising_edge(sysclk) then
      if (nreset = '0' and n_rst_d = '0') then
         cnt_rst <= '1';
      else
         cnt_rst <= '0';
      end if;
```

```
      n_rst_d  := nreset;
      end if;
end process cnt_rst_gen;
-------------------------------------------------------------------------------
-- generate an lsb counter.
-------------------------------------------------------------------------------
cnt_lsb_gen: process (sysclk)
begin
   if rising_edge(sysclk) then
      if (cnt_rst = '1') then
         count_lsb <= (others=>'0');
      else
         count_lsb <= count_lsb + 1;
      end if;

      if (count_lsb = "11111111") then
         carry_lsb <= '1';
      else
         carry_lsb <= '0';
      end if;
   end if;
end process cnt_lsb_gen;
-------------------------------------------------------------------------------
-- generate an lmsb counter.
-------------------------------------------------------------------------------
cnt_msb_gen: process (sysclk)
begin
   if rising_edge(sysclk) then
      if (cnt_rst = '1') then
         count_msb <= (others=>'0');
      elsif (carry_lsb = '1') then
         count_msb <= count_msb + 1;
      end if;

      if (count_msb = "11111111") then
         carry_msb <= '1';
      else
         carry_msb <= '0';
      end if;
   end if;
end process cnt_msb_gen;
-------------------------------------------------------------------------------
-- generate an nsb counter.
-------------------------------------------------------------------------------
cnt_nsb_gen: process (sysclk)
begin
   if rising_edge(sysclk) then
      if (cnt_rst = '1') then
         count_nsb <= (others=>'0');
      elsif (carry_lsb = '1') and (carry_msb = '1') then
         count_nsb <= count_nsb + 1;
      end if;

      if (count_nsb = "11111111") then
         carry_nsb <= '1';
      else
         carry_nsb <= '0';
      end if;
   end if;
end process cnt_nsb_gen;
-------------------------------------------------------------------------------
-- generate an osb counter.
-------------------------------------------------------------------------------
cnt_osb_gen: process (sysclk)
begin
   if rising_edge(sysclk) then
      if (cnt_rst = '1') then
         count_osb <= (others=>'0');
      elsif (carry_lsb = '1') and (carry_msb = '1') and (carry_nsb = '1') then
         count_osb <= count_osb + 1;
      end if;
```

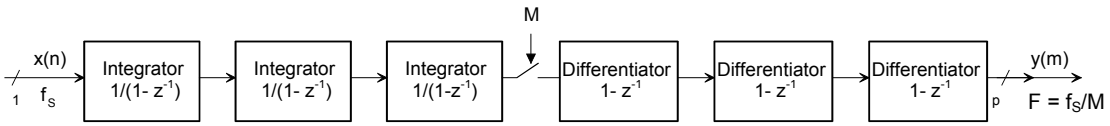


FIGURE 5.17: $Sinc^3$ digital filter topology [61].

```
   end if;
end process cnt_osb_gen;
-------------------------------------------------------------------------------
-- generate an "active high output".
-------------------------------------------------------------------------------
op_gen: process (sysclk)
begin
   if rising_edge(sysclk) then
      timer(7  downto  0) <= std_logic_vector(count_lsb(7 downto 0));
      timer(15 downto  8) <= std_logic_vector(count_msb(7 downto 0));
      timer(23 downto 16) <= std_logic_vector(count_nsb(7 downto 0));
      timer(31 downto 24) <= std_logic_vector(count_osb(7 downto 0));
   end if;
end process op_gen;
end first;
```

5.2.3 Signal Conditioning

Recalling the ADC part of the adaptor board in Figure 5.6, a digital filter is required for configuring the $\Delta\Sigma$ modulator ADS1204. This should be used with a digital filter (software) for two reasons. First, it filters out high frequency noise. Second, it converts the 1-bit data stream at a high sampling rate into a higher-bit data word at a lower rate (called decimation).

A $Sinc^3$ filter was used according to the recommendations in [61] and due to the importance of the filter digitisation, a brief of those recommendations is described next. The filter topology is shown in Figure 5.17, where 3 accumulators are cascaded operating at high sample rate (sampling frequency f_s) followed by 3 stages of cascaded differentiators operating at the lower sample rate, f_s/M (where M is the decimation ratio of the sampling rate compressor). This filter does not require the use of digital multipliers and could be expanded as a $Sinc^k$ filter as

$$H(z) = \left(\frac{1}{M} \quad \frac{1 - z^{-M}}{1 - z^{-1}} \right)^k \tag{5.5}$$

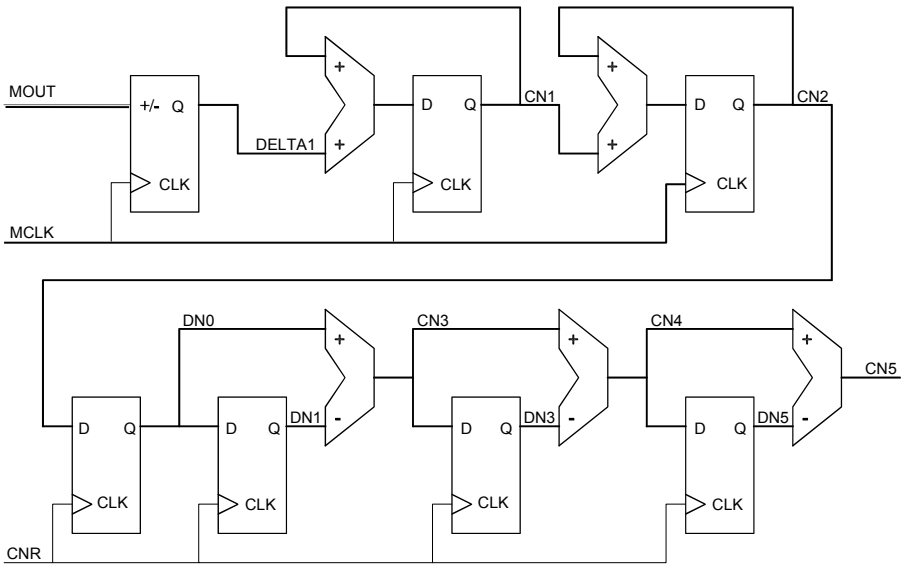


FIGURE 5.18: $Sinc^3$ filter implementation [61].

The $Sinc^3$ filter outputs M word samples after each input, which represents $3(M-1)+1$ input samples. It can be implemented using a linear convolution

$$y(k) = \sum_{n=0}^{3M-1} h(n)x(k-n), \tag{5.6}$$

where $x(i)$ is the input data stream (ones and zeroes), $h(n)$ are the filter coefficients, $y(k)$ is the decimated output data words. $h(n)$ values are calculated based on decimation ratio as

$$h(n) = \frac{n(n+1)}{2} \qquad 0 \leq n \leq M-1 \tag{5.7}$$

$$h(n) = \frac{M(M+1)}{2} + (n+M)(2M-1-n) \qquad M \leq n \leq 2M-1 \tag{5.8}$$

$$h(n) = \frac{(3M-n-1)(3M-n)}{2} \qquad 2M \leq n \leq 3M-1 \tag{5.9}$$

As presented in Figure 5.17, the output of the third integrator is decimated down by M and fed to the first differentiator. Figure 5.18 presents this implementation where `MOUT` is the ones/zeroes stream coming from the modulator and representing the signal of interest $x(i)$, `MCLK` is the modulator frequency $f_s = 10$ [MHz], `CNR`

is the data rate f_s/M i.e. `CNR=MCLK/M`, and the output data is `CN5`. Translating this diagram and matching nomenclature into a *.vhd* file results in the code shown below

```
-- Digital_FLT.vhd --
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
entity Digital_FLT is
port(
     RESN, MOUT, MCLK, CNR : in std_logic;
     CN5                   : out std_logic_vector(24 downto 0)); end Digital_FLT;
architecture RTL of Digital_FLT is
signal DN0, DN1, DN3, DN5 : std_logic_vector(24 downto 0);
signal CN1, CN2, CN3, CN4 : std_logic_vector(24 downto 0);
signal DELTA1             : std_logic_vector(24 downto 0);
begin
   process(MCLK, RESn)
   begin
      if RESn = '0' then

         DELTA1 <= (others => '0');
      elsif MCLK'event and MCLK = '1' then
         if MOUT = '0' then
         DELTA1 <= DELTA1 + 1;
         end if;
      end if;
   end process;
   process(RESN, MCLK)
   begin
      if RESN = '0' then
         CN1 <= (others => '0');
         CN2 <= (others => '0');
      elsif MCLK'event and MCLK = '1' then
         CN1 <= CN1 + DELTA1;
         CN2 <= CN2 + CN1;
      end if;
   end process;
   process(RESN, CNR)
   begin
      if RESN = '0' then
         DN0 <= (others => '0');
         DN1 <= (others => '0');
         DN3 <= (others => '0');
         DN5 <= (others => '0');
      elsif CNR'event and CNR = '1' then
         DN0 <= CN2;

         DN1 <= DN0;
         DN3 <= CN3;
         DN5 <= CN4;
      end if;
   end process;

      CN3 <= DN0 - DN1;
      CN4 <= CN3 - DN3;
      CN5 <= CN4 - DN5;
end RTL;
```

TABLE 5.4: Output word size from different integrators in $Sinc^3$ filter for 1-bit input word

Decimation M	Data rate [kHz]	$Gain_{DC}$	$Gain_{DC}$(bits)	$Buswidth$(bits)
4	2,500.0	64	6	7
8	1,250.0	512	9	10
16	625.0	4,096	12	13
32	312.5	32,768	15	16
64	156.2	262,144	18	19
128	78.1	2,097,152	21	22
256	39.1	16,777,216	24	25

The gain for a $Sinc^k$ filter is given by

$$Gain_{DC} = M^k, \tag{5.10}$$

and the bus width by

$$Buswidth = 1 + k \log_2 M. \tag{5.11}$$

Table 5.4 show the different gain values for a $Sinc^3$ filter, including the data rate. The M value chosen for this prototype is 256 with the third order filter. For a 256 decimation, according to equation (5.10), the input will be multiplied by $256^3 = 16,777,216$, which results in 18 bits. In each filter order, the output word size is increased by $k \log_2 M$. If the input is 1 bit, the output from the first order filter will be an 8-bit word. The second order filter will increase the word size to 16-bit and a third one will to a 24-bit word. Equation (5.11) states that the internal bus of the $Sinc$ filter needs to have a bus width that is one bit wider than the filter's dc gain. More details of the filter such as bandwidth and limitations are found in [59] and [61].

The eight input signals (currents and voltages) are conditioned with the modulators by using a digital filter `Digital_FLT` as shown in Figure 5.19. The embedded code of the `Digital_FLT` block has been shown above and besides the signal of interest `MOUT` as an input, it also has two input frequencies `MCLK=CLK_OUT1` as a conversion clock and `CNR=DCLK` as its data rate. This code is a recommendation

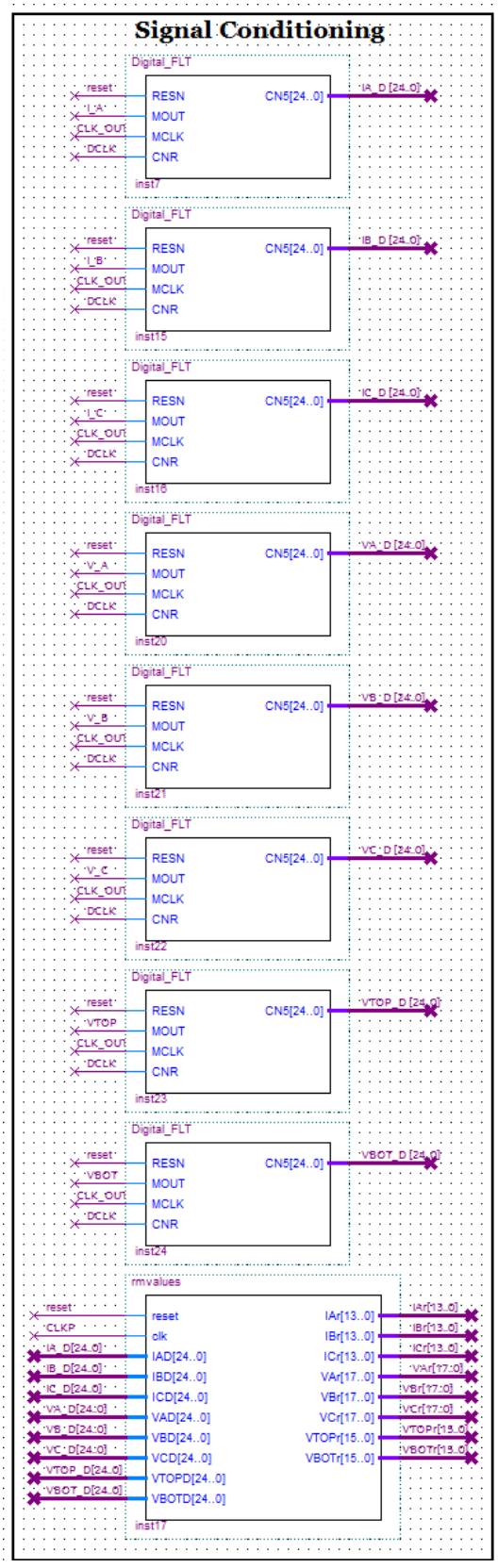


FIGURE 5.19: *Quartus*: Signal conditioning

for the ADC (ADS1202) in SBAA094 Application Report [61] and was adapted for the ADC ADS1204 used in this prototype.

Once the signals have passed through the filtering stage, they need to be calibrated. In a nutshell, this means the zero value of the signals needs to be found in order to manage positive and negative real values. This is done with a slope equation $y = mx+b$, which is described in the next subsection. The code below is embedded in `rmvalues` block from Figure 5.19 and output the representative mili-values of voltages and currents, i.e. [mV] and [mA], respectively. Observe the different data lenghts for the currents `IXr`, grid voltages `VXr` and capacitor voltages `VXOXr` at 14-, 18- and 16-bit, respectively.

```
-- rmvalues.vhd--
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.numeric_std.all;
entity rmvalues is
port
(
-- Input ports
reset :  in std_logic;
clk   :  in  std_logic;
IAD : in std_logic_vector(24 downto 0); -- I max value 16777216 // 33554432
IBD : in std_logic_vector(24 downto 0); -- I max value 16777216
ICD : in std_logic_vector(24 downto 0); -- I max value 16777216
VAD : in std_logic_vector(24 downto 0); -- V max value 16777216
VBD : in std_logic_vector(24 downto 0); -- V max value 16777216
VCD : in std_logic_vector(24 downto 0); -- V max value 16777216

VTOPD : in std_logic_vector(24 downto 0); -- V max value 1048576
VBOTD : in std_logic_vector(24 downto 0); -- V max value 1048576

-- Output ports
IAr : out integer range -8191 to 8191 :=0; -- 2^13
IBr : out integer range -8191 to 8191 :=0;
ICr : out integer range -8191 to 8191 :=0;
VAr : out integer range -131071 to 131071 :=0; --2^17
VBr : out integer range -131071 to 131071 :=0;
VCr : out integer range -131071 to 131071 :=0;
VTOPr : out integer range 0 to 65535 := 0; -- -65535 to 65535 :=0; --2^16
VBOTr : out integer range 0 to 65535 := 0 -- -65535 to 65535 :=0
);
end rmvalues;
architecture rconv of rmvalues is
constant zero:integer := 0;
constant kia : integer:= 819; -- I data from calibration tables
constant kib : integer:= 828;
constant kic : integer:= 842;
constant Nia : integer:= -8408172;
constant Nib : integer:= -8473619;
constant Nic : integer:= -8412461;
signal IADin : integer range 0 to 16777216 := 0;
signal IBDin : integer range 0 to 16777216 := 0;
signal ICDin : integer range 0 to 16777216 := 0;
constant kva : integer:= 44; -- V data from calibration tables
```

```
constant kvb : integer:= 45;
constant kvc : integer:= 44;
constant Nva : integer:= -8363171;
constant Nvb : integer:= -8360455;
constant Nvc : integer:= -8437069;
signal VADin : integer range 0 to 16777216 := 0;
signal VBDin : integer range 0 to 16777216 := 0;
signal VCDin : integer range 0 to 16777216 := 0;
constant kvtop : integer:= 11; --13; -- VCAP data from calibration tables
constant kvbot : integer:= 10; -- 13; -- -13;
constant Nvtop : integer:= -7828555; -- -8437069; -- ERROR?
constant Nvbot : integer:= -7766868; -- -8377459; -- ERROR?
signal VTOPDin : integer range 0 to 65535:= 0; -- 1048576 := 0;
signal VBOTDin : integer range 0 to 65535:= 0; -- 1048576 := 0;
begin
IADin <= to_integer(unsigned(IAD));
IBDin <= to_integer(unsigned(IBD));
ICDin <= to_integer(unsigned(ICD));
VADin <= to_integer(unsigned(VAD));
VBDin <= to_integer(unsigned(VBD));
VCDin <= to_integer(unsigned(VCD));
VTOPDin <= to_integer(unsigned(VTOPD));
VBOTDin <= to_integer(unsigned(VBOTD));
process(reset,clk)
begin
if (reset = '0') then

IAr   <= zero; -- currents
IBr   <= zero;
ICr   <= zero;
VAr   <= zero; -- voltages
VBr   <= zero;
VCr   <= zero;
VTOPr <= zero; -- cap voltages
VBOTr <= zero;
elsif(rising_edge(clk)) then
IAr   <= (IADin + Nia)/kia; --currents
IBr   <= (IBDin + Nib)/kib;
ICr   <= (ICDin + Nic)/kic;
VAr   <= (VADin + Nva)/kva; --voltages
VBr   <= (VBDin + Nvb)/kvb;
VCr   <= (VCDin + Nvc)/kvc;
VTOPr <= (VTOPDin + Nvtop)/kvtop;  -- cap voltages
VBOTr <= (VBOTDin + Nvbot)/kvbot;
end if;
end process;
end rconv;
```

5.2.4 HEX: Signal Display

The main purpose for coding a system to display hexadecimal numbers in the 7-segment displays was to calibrate input measurements (voltages and currents) in a way that is consistent to the real measurements. Figure 5.20 shows a `disp_switcher` block with its embedded code below, serving as a signal selector. At the early stages, some switches `SW` were assigned as selectors, in order to display a specific signal (voltage or current) at a time in the set of 7-segment displays. Looking

closer to the code, a set of 25-bit signals are received and labelled reasonably, and according to which `SW` is selected, a specific signal is output via `Dsig`.

```
--disp_switcher.vhd--
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity disp_switcher is
port (
-- Input ports
   clk      : in std_logic;
   RESN     : in std_logic;
pulse : in std_logic;
sVI    : in std_logic_vector(9 downto 0);
VA_D : in std_logic_vector(24 downto 0);
VB_D : in std_logic_vector(24 downto 0);
VC_D : in std_logic_vector(24 downto 0);
IA_D     : in std_logic_vector(24 downto 0);
IB_D : in std_logic_vector(24 downto 0);
IC_D : in std_logic_vector(24 downto 0);
VTOP_D : in std_logic_vector(24 downto 0);
VBOT_D : in std_logic_vector(24 downto 0);
VAB_D : in std_logic_vector(24 downto 0);
IAB_D : in std_logic_vector(24 downto 0);
-- Output ports
   Dsig     : out std_logic_vector(24 downto 0)
   );
end entity disp_switcher;
architecture visual1 of disp_switcher is
   signal zero    : std_logic_vector(24 downto 0);
   signal latch  : std_logic_vector(24 downto 0);
begin
zero <= "0000000000000000000000000";
process(clk,RESN,pulse,zero,latch,VA_D,VB_D,VC_D,IA_D,IB_D,IC_D)
begin
if RESN = '0' then
          Dsig <= zero;
   elsif rising_edge(clk) and RESN = '1' and pulse = '1' then
case sVI is
when "0000000000" => Dsig <= zero;
when "0000000001" => Dsig <= VA_D; latch <= VA_D;
when "0000000010" => Dsig <= VB_D; latch <= VB_D;
when "0000000100" => Dsig <= VC_D; latch <= VC_D;
when "0000001000" => Dsig <= IA_D; latch <= IA_D;
when "0000010000" => Dsig <= IB_D; latch <= IB_D;
when "0000100000" => Dsig <= IC_D; latch <= IC_D;
when "0001000000" => Dsig <= VTOP_D; latch <= VTOP_D;
when "0010000000" => Dsig <= VBOT_D; latch <= VBOT_D;
when "0100000000" => Dsig <= VAB_D; latch <= VAB_D;
when "1000000000" => Dsig <= IAB_D; latch <= IAB_D;
when others => Dsig <= zero; latch <= zero;
end case;
elsif rising_edge(clk) and RESN = '1' and pulse = '0' then
Dsig <= latch;
end if;
end process;
end visual1;
```

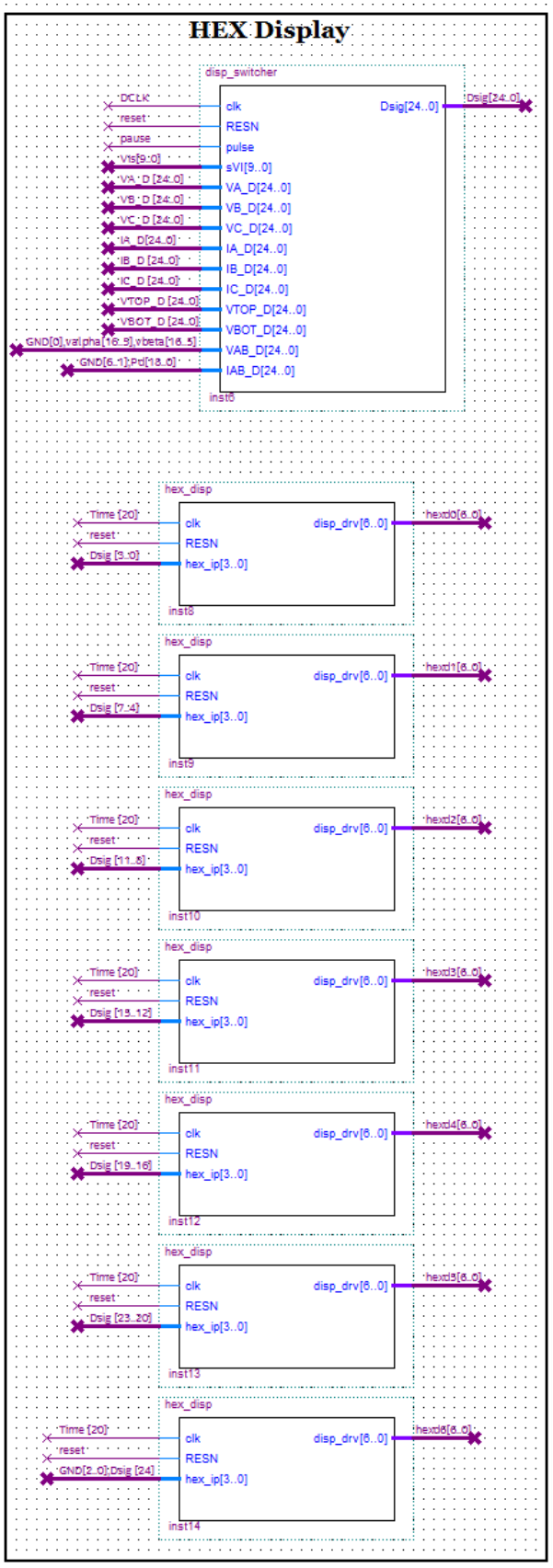


FIGURE 5.20: *Quartus*: Hexadecimal display

```
--hex_disp.vhd --

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity hex_disp is
port (
   clk      : in std_logic;
   RESN     : in std_logic;
   hex_ip   : in std_logic_vector(3 downto 0);
   disp_drv : out std_logic_vector(6 downto 0)
   );
end entity hex_disp;
architecture first of hex_disp is
   signal off_state  : std_logic;
   signal on_state   : std_logic;
begin
off_state   <= '1';
on_state    <= '0';
process(clk,RESN,off_state,on_state)
begin
if RESN = '0' then
  disp_drv(0) <= off_state;
disp_drv(1) <= off_state;
disp_drv(2) <= off_state;
                disp_drv(3) <= off_state;
                disp_drv(4) <= off_state;

                disp_drv(5) <= off_state;
                disp_drv(6) <= on_state;
   elsif rising_edge(clk) and RESN = '1' then
      case hex_ip is
         when "0000" => --0
                disp_drv(0) <= on_state;
                disp_drv(1) <= on_state;
                disp_drv(2) <= on_state;
                disp_drv(3) <= on_state;
                disp_drv(4) <= on_state;
                disp_drv(5) <= on_state;
                disp_drv(6) <= off_state;
         when "0001" => --1
                disp_drv(0) <= off_state;
                disp_drv(1) <= on_state;

                disp_drv(2) <= on_state;
                disp_drv(3) <= off_state;
                disp_drv(4) <= off_state;
                disp_drv(5) <= off_state;
                disp_drv(6) <= off_state;
         when "0010" => --2
                disp_drv(0) <= on_state;
                disp_drv(1) <= on_state;
                disp_drv(2) <= off_state;
                disp_drv(3) <= on_state;
                disp_drv(4) <= on_state;
                disp_drv(5) <= off_state;
                disp_drv(6) <= on_state;
         when "0011" => --3
                disp_drv(0) <= on_state;
                disp_drv(1) <= on_state;
                disp_drv(2) <= on_state;
                disp_drv(3) <= on_state;
                disp_drv(4) <= off_state;
                disp_drv(5) <= off_state;
                disp_drv(6) <= on_state;
         when "0100" => --4
                disp_drv(0) <= off_state;
                disp_drv(1) <= on_state;
                disp_drv(2) <= on_state;

                disp_drv(3) <= off_state;
                disp_drv(4) <= off_state;
                disp_drv(5) <= on_state;
                disp_drv(6) <= on_state;
         when "0101" => --5
                disp_drv(0) <= on_state;
                disp_drv(1) <= off_state;
                disp_drv(2) <= on_state;
                disp_drv(3) <= on_state;
                disp_drv(4) <= off_state;
                disp_drv(5) <= on_state;
                disp_drv(6) <= on_state;
         when "0110" => --6
                disp_drv(0) <= on_state;
                disp_drv(1) <= off_state;
                disp_drv(2) <= on_state;
                disp_drv(3) <= on_state;
                disp_drv(4) <= on_state;

disp_drv(5) <= on_state;
                disp_drv(6) <= on_state;
         when "0111" => --7
                disp_drv(0) <= on_state;
                disp_drv(1) <= on_state;
                disp_drv(2) <= on_state;
                disp_drv(3) <= off_state;
                disp_drv(4) <= off_state;
                disp_drv(5) <= off_state;
                disp_drv(6) <= off_state;
         when "1000" => --8
                disp_drv(0) <= on_state;
                disp_drv(1) <= on_state;
                disp_drv(2) <= on_state;
                disp_drv(3) <= on_state;
                disp_drv(4) <= on_state;
                disp_drv(5) <= on_state;
                disp_drv(6) <= on_state;
         when "1001" => --9
                disp_drv(0) <= on_state;
                disp_drv(1) <= on_state;
                disp_drv(2) <= on_state;
                disp_drv(3) <= on_state;
                disp_drv(4) <= off_state;
                disp_drv(5) <= on_state;

                disp_drv(6) <= on_state;
         when "1010" => --A
                disp_drv(0) <= on_state;
                disp_drv(1) <= on_state;
                disp_drv(2) <= on_state;
                disp_drv(3) <= off_state;
                disp_drv(4) <= on_state;
                disp_drv(5) <= on_state;
                disp_drv(6) <= on_state;
         when "1011" => --b
                disp_drv(0) <= off_state;
                disp_drv(1) <= off_state;
                disp_drv(2) <= on_state;
                disp_drv(3) <= on_state;
                disp_drv(4) <= on_state;
                disp_drv(5) <= on_state;
                disp_drv(6) <= on_state;
         when "1100" => --C
                disp_drv(0) <= on_state;
                disp_drv(1) <= off_state;
                disp_drv(2) <= off_state;
                disp_drv(3) <= on_state;
                disp_drv(4) <= on_state;
                disp_drv(5) <= on_state;
                disp_drv(6) <= off_state;
         --cont...
```

```
--...
        when "1101" => --D
               disp_drv(0) <= off_state;
               disp_drv(1) <= on_state;
               disp_drv(2) <= on_state;
               disp_drv(3) <= on_state;
               disp_drv(4) <= on_state;
               disp_drv(5) <= off_state;
               disp_drv(6) <= on_state;
        when "1110" => --E
               disp_drv(0) <= on_state;
               disp_drv(1) <= off_state;
               disp_drv(2) <= off_state;
               disp_drv(3) <= on_state;
               disp_drv(4) <= on_state;
               disp_drv(5) <= on_state;
               disp_drv(6) <= on_state;
                                                  when "1111" => --F
                                                         disp_drv(0) <= on_state;
                                                         disp_drv(1) <= off_state;
                                                         disp_drv(2) <= off_state;
                                                         disp_drv(3) <= off_state;
                                                         disp_drv(4) <= on_state;
                                                         disp_drv(5) <= on_state;
                                                         disp_drv(6) <= on_state;
                                                  when others =>
                                            disp_drv(0) <= off_state;
                                          disp_drv(1) <= on_state;
                                          disp_drv(2) <= on_state;
                                                         disp_drv(3) <= off_state;
                                                         disp_drv(4) <= on_state;
                                                         disp_drv(5) <= on_state;
                                                         disp_drv(6) <= on_state;
                                               end case;
                                             end if;
                                          end process;
                                          end first;
```

Recalling the `timer32` block explained previously, its output `Time[20]` was chosen as the sampling rate input for the `hex_disp` blocks in order for these numbers to be displayed at rates that the human eye can see. Each `hex_disp` block, also shown in Figure 5.20, operate a 7-segment display and usually have a 4-bit input from the selected signal `Dsig[24..0]`. `hex_disp` blocks are arranged so that the most significant bit (MSB) is output in the far left-bottom display and the least significant bit output in the far right-top display (see Figure 5.5 for reference). The code above is embedded in every `hex_disp` block and it is basically a 7-led on/off arrangement which translates each of the 16 possible combinations (from the 4-bit input) into hexadecimal numbers (0 to F).

5.2.5 Transformations and Calculations

Once the voltages and currents are obtained, they need to be transformed into the $\alpha\beta$-frame. As the code is unable to handle number fractions (or points) directly, a modified transformation of the Clarke transformation in equation (4.5) while keeping the [mili-] units was produced:

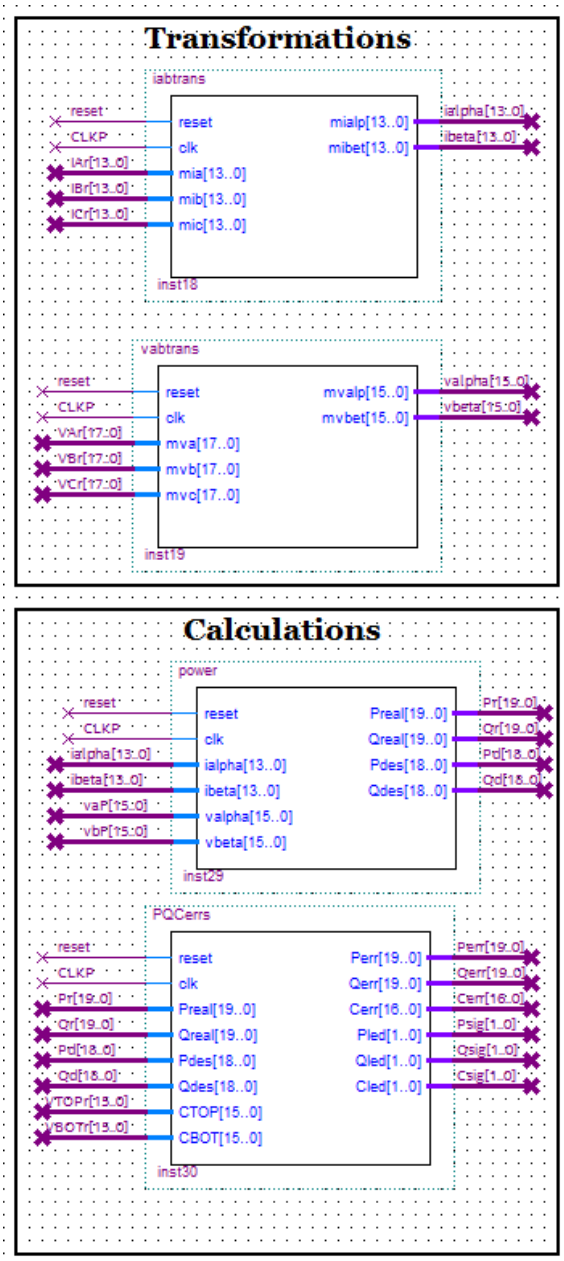


FIGURE 5.21: *Quartus*: Transformations and calculations

$$\begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} = 3000 \begin{bmatrix} 2000 & -1000 & -1000 \\ 0 & 1732 & -1732 \end{bmatrix} \begin{bmatrix} u_a \\ u_b \end{bmatrix}. \tag{5.12}$$

Equation (5.12) was used for the current and grid voltage measurements, which is described in the codes `iabtrans.vhd` and `vabtrans.vhd`, respectively. Their block

form is introduced in Figure 5.21 under *Transformations*. The three 14-bit readings (representing i_{abc}) are transformed to obtain two 14-bit outputs (representing i_α and i_β). The same applies to the 16-bit readings/outputs for the grid voltages.

```
-- iabtrans.vhd --
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.numeric_std.all;
entity iabtrans is
port
(
-- Input ports
reset    :  in std_logic;
clk      :  in  std_logic;
mia : in integer range -8191 to 8191 :=0;
mib : in integer range -8191 to 8191 :=0;
mic : in integer range -8191 to 8191 :=0;
-- Output ports
mialp : out integer range -8191 to 8191 :=0;
mibet : out integer range -8191 to 8191 :=0
);
end iabtrans;
architecture abeta of iabtrans is
constant zero  :integer := 0;
constant con1 :integer := 2000;
constant con2 :integer := 1000;
constant con3 :integer := 1732;
constant den :integer := 3000;
begin
process(reset,clk)
begin
if (reset = '0') then
mialp   <= zero;
mibet   <= zero;
elsif(rising_edge(clk)) then
mialp  <= (con1*mia - con2*mib - con2*mic)/den;
mibet  <= (con3*mib - con3*mic)/den;
end if;
end process;
end abeta;
```

```
-- vabtrans.vhd --
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.numeric_std.all;
entity vabtrans is
port
(
-- Input ports
reset :  in std_logic;
clk : in std_logic;
mva : in integer range -131071 to 131071 :=0;
mvb : in integer range -131071 to 131071 :=0;
mvc : in integer range -131071 to 131071 :=0;
-- Output ports
--mvalp : out integer range -131071 to 131071 :=0;
--mvbet : out integer range -131071 to 131071 :=0
mvalp : out integer range -31071 to 31071 :=0;
mvbet : out integer range -31071 to 31071 :=0
);
end vabtrans;
```

```
architecture abeta2 of vabtrans is
constant zero  :integer := 0;
constant con1 :integer := 2000;
constant con2 :integer := 1000;
constant con3 :integer := 1732;
constant den :integer := 3000;
begin
process(clk,reset)
begin
if (reset = '0') then
mvalp   <= zero; -- currents
mvbet   <= zero;
elsif(rising_edge(clk)) then
-- ORIG transformation
mvalp  <= (con1*mva - con2*mvb - con2*mvc)/den;
mvbet  <= (con3*mvb - con3*mvc)/den;
end if;
end process;
end abeta2;
--
```

Under *Calculations*, the block `power` calculates the current active and reactive power, making sure it outputs 20-bits in [mW] units. This power calculation requires current current measurements `ialpha` and `ibeta` as well as grid follower voltages `vaF` and `vbF` (also in the $\alpha\beta$ plane). These grid follower voltages are produced by the PLL and are used to achieve a less noisy signal due to the transistor switching. This same `power` block corresponds to the `power.vhd` code below and it also sets the desired active and reactive power in [mW] units, using 19-bits.

```
-- power.vhd--
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.numeric_std.all;
entity power is
port
(
-- Input ports
reset :  in std_logic;
clk : in std_logic;
ialpha : in integer range -8191 to 8191 :=0;
ibeta : in integer range -8191 to 8191 :=0;
--ORIG
valpha : in integer range -31071 to 31071 :=0;
vbeta : in integer range -31071 to 31071 :=0;
-- Output ports
Preal : out integer range -524287 to 524287 :=0;
Qreal : out integer range -524287 to 524287 :=0;
Pdes : out integer range 0 to 524287 :=0;
Qdes : out integer range 0 to 524287 :=0
);
end power;
```

```
architecture powpow of power is
constant Pdesired: integer:=150000; --P DES [mW]
constant Qdesired: integer:=0; --Q DES [mW]
constant zero  :integer:= 0;
constant kdmil :integer:=10000;
constant k15 :integer:=15;
begin
process(reset,clk)
begin
if (reset = '0') then
Preal <= zero;
Qreal <= zero;
Pdes <= Pdesired;
Qdes <= Qdesired;
elsif(rising_edge(clk)) then
Preal <=(valpha*ialpha+vbeta*ibeta)*k15/kdmil;
Qreal <=(vbeta*ialpha-valpha*ibeta)*k15/kdmil;
Pdes <= Pdesired;
Qdes <= Qdesired;
end if;
end process;
end powpow;
```

The last block in Figure 5.21 is `PQCerrs` with its corresponding `.vhd` code shown below. This piece of code calculates the power errors, or the difference between the

desired and real power values. The capacitor voltage error is simply the difference between the capacitor voltages as described in the theory. Now the errors of active power `Perr`, reactive power `Qerr`, and capacitor voltages `Cerr` are obtained in 20-, 20- and 17-bits respectively. These outputs are kept in [mili-] units. The `Psig`, `Qsig` and `Csig` values are used as temporary indicators and served during the initial tests.

```
-- PQCerrs.vhd --
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.numeric_std.all;
entity PQCerrs is
port
(
-- Input ports
reset :  in std_logic;
clk : in std_logic;
Preal :  in integer range -524287 to 524287 :=0;
Qreal :  in integer range -524287 to 524287 :=0;
Pdes :  in integer range 0 to 524287 :=0;
Qdes :  in integer range 0 to 524287 :=0;
CTOP     : in integer range 0 to 65535 := 0;
CBOT     : in integer range 0 to 65535 := 0;
-- Output ports
Perr : out integer range -500000 to 500000 :=0;
Qerr : out integer range -500000 to 500000 :=0;
Cerr : out integer range -65535 to 65535 :=0;
Pled : out std_logic_vector(1 downto 0);
Qled : out std_logic_vector(1 downto 0);
Cled : out std_logic_vector(1 downto 0)
);
end PQCerrs;
architecture errors of PQCerrs is
constant zero  :integer := 0;
constant dosmil:integer := 2000;
constant Pup : integer :=  170000;
constant Pdo : integer :=  130000;
constant Qup : integer :=  50000;
constant Qdo : integer := -50000;
constant Cet : integer :=   5000;
constant Ceb : integer :=  -5000;
constant cero : integer :=  0;
begin
process(reset,clk)
begin
if (reset = '0') then
Perr <= zero;
Qerr <= zero;
Cerr <= zero;
elsif(rising_edge(clk)) then
Perr <= Pdes - Preal;
Qerr <= Qdes - Qreal;
Cerr <= CTop - CBot;
if Preal > Pup then
Pled <= "10";
elsif Preal < Pdo then
Pled <= "01";
else
Pled <= "00";
end if;
if Qreal > Qup then
```

```
Qled <= "10";
elsif Qreal < Qdo then
Qled <= "01";
else
Qled <= "00";
end if;
if CTOP < Cet then
Cled <= "10";
elsif CBOT > Ceb then
Cled <= "01";
else
Cled <= "00";
end if;
end if;
end process;
end errors;
```

5.2.6 PLL

The PLL structure was coded using ZVC (Zero Voltage Crossing) and quadrant synchronisation as shown in Figure 5.22. The quadrant detection and desired action is on the left, while the variable speed and LUT (Look-Up Table) is on the right. The description starts on the right hand side with the LUT followed by the variable speed technique, and the quadrant detection is explained last.

Recalling the $\alpha\beta$ frame in Figure 4.5, a given voltage vector can be located in the plane, and it is known that the grid voltage vector rotates in this plane. When translating to a digital language, the target becomes to produce a sine and cosine value for every vector position $0 \leq \theta < 2\pi$. How many vector positions are there is a resolution or accuracy problem. In this PLL, it was chosen to have 63 possible positions (including zero) to homologate with one complete rotation in radians: $2\pi = 6.283185307179586....$

```
-- sinLUT2019.vhd --
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity sinLUT2019 is -- 1 orders more on output
port(
-- Input ports
clk   : in std_logic;
reset : in std_logic;
newTH : in std_logic_vector(6 downto 0);
-- Output ports
sinval: out integer range -100 to 100:=0; -- linked to vBeta
```

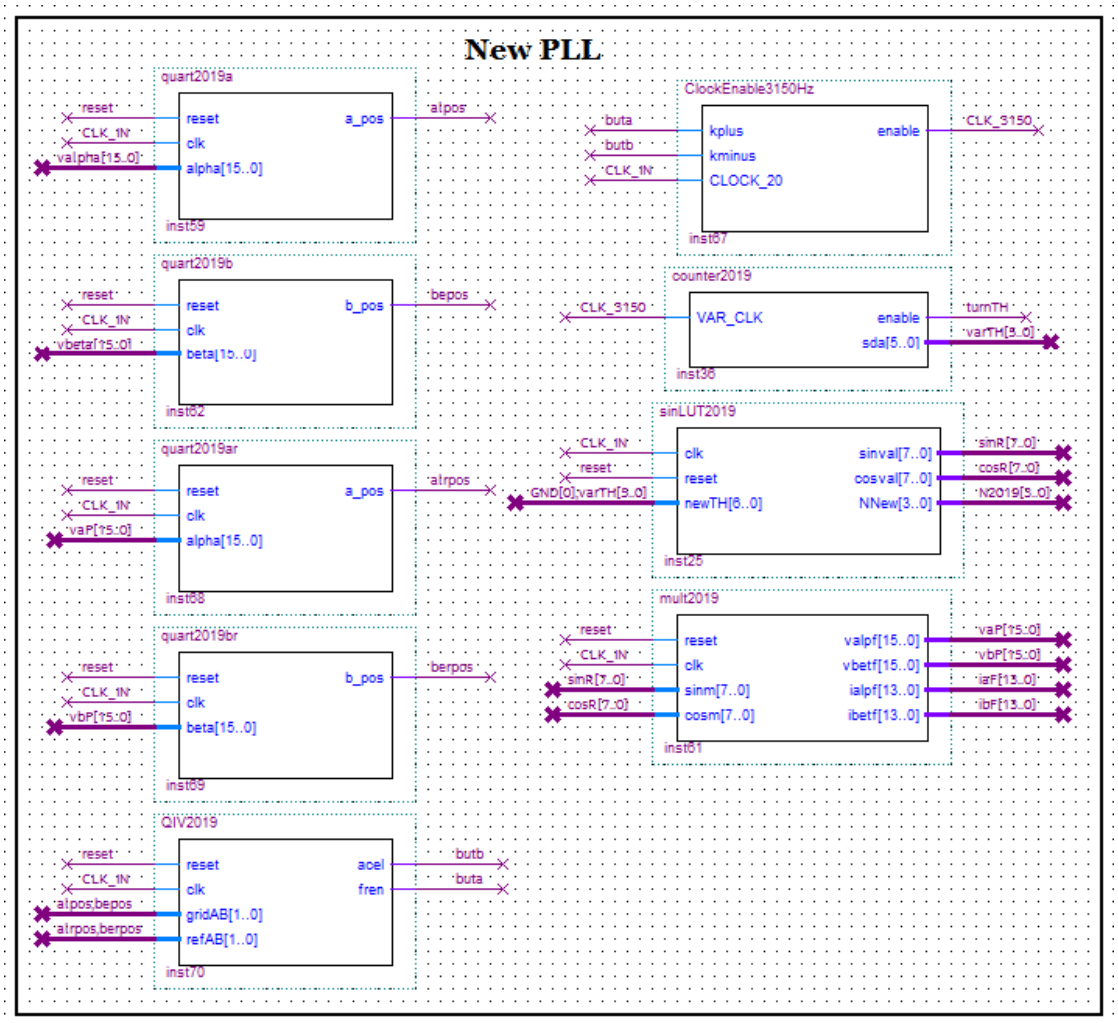


FIGURE 5.22: *Quartus*: Phase Locked Loop

```
cosval: out integer range -100 to 100:=0; -- linked to vAlpha
NNew : out std_logic_vector(3 downto 0)
);
end sinLUT2019;
architecture deer of sinLUT2019 is
    constant Nuno   : std_logic_vector:="0001";
constant Ndos    : std_logic_vector:="0010";
constant Ntres   : std_logic_vector:="0011";
constant Ncuatro  : std_logic_vector:="0100";
constant Ncinco   : std_logic_vector:="0101";
constant Nseis   : std_logic_vector:="0110";
constant Nsiete   : std_logic_vector:="0111";
constant Nocho   : std_logic_vector:="1000";
constant Nnueve   : std_logic_vector:="1001";
constant Ndiez   : std_logic_vector:="1010";
constant Nonce   : std_logic_vector:="1011";
constant Ndoce   : std_logic_vector:="1100";
begin
process(clk,reset)
begin
if reset = '0' then
sinval <= 0;
cosval <= 0;
NNew <= Nuno;
elsif (rising_edge(clk)) then
```

```
case newTH is  -- To syncronize it forward, add +1 or +N to the dependency block
when  "0111111" => sinval <= 2 ; cosval <= 100 ; NNew <= Ncuatro; --***
when "0111110"  => sinval <= -8 ; cosval <= 100 ; NNew <= Ncuatro;
when "0111101"  => sinval <= -18 ; cosval <= 98 ; NNew <= Ncuatro;
when "0111100"  => sinval <= -28 ; cosval <= 96 ; NNew <= Ncuatro;
when "0111011"  => sinval <= -37 ; cosval <= 93 ; NNew <= Ncuatro;
when "0111010"  => sinval <= -46 ; cosval <= 89 ; NNew <= Ncinco;-- NNew <= Ncuatro;
when "0111001"  => sinval <= -55 ; cosval <= 83 ; NNew <= Ncinco; -- .15 ahead
when "0111000"  => sinval <= -63 ; cosval <= 78 ; NNew <= Ncinco;
when "0110111"  => sinval <= -71 ; cosval <= 71 ; NNew <= Ncinco;
when "0110110"  => sinval <= -77 ; cosval <= 63 ; NNew <= Ncinco;
when "0110101"  => sinval <= -83 ; cosval <= 55 ; NNew <= Ncinco;
when "0110100"  => sinval <= -88 ; cosval <= 47 ; NNew <= Nseis;--NNew <= Ncinco;
when "0110011"  => sinval <= -93 ; cosval <= 38 ; NNew <= Nseis;
when "0110010"  => sinval <= -96 ; cosval <= 28 ; NNew <= Nseis;
when "0110001"  => sinval <= -98 ; cosval <= 19 ; NNew <= Nseis;
when "0110000"  => sinval <= -100 ; cosval <= 9 ; NNew <= Nseis;
when "0101111"  => sinval <= -100 ; cosval <= -1 ; NNew <= Nsiete;--NNew <= Nseis;
when "0101110"  => sinval <= -99 ; cosval <= -11 ; NNew <= Nsiete;
when "0101101"  => sinval <= -98 ; cosval <= -21 ; NNew <= Nsiete;
when "0101100"  => sinval <= -95 ; cosval <= -31 ; NNew <= Nsiete;
when "0101011"  => sinval <= -92 ; cosval <= -40 ; NNew <= Nsiete;
when "0101010"  => sinval <= -87 ; cosval <= -49 ; NNew <= Nocho; --NNew <= Nsiete;
when "0101001"  => sinval <= -82 ; cosval <= -57 ; NNew <= Nocho; -- OK
when "0101000"  => sinval <= -76 ; cosval <= -65 ; NNew <= Nocho;
when "0100111"  => sinval <= -69 ; cosval <= -73 ; NNew <= Nocho;
when "0100110"  => sinval <= -61 ; cosval <= -79 ; NNew <= Nocho;
when "0100101"  => sinval <= -53 ; cosval <= -85 ; NNew <= Nnueve;--NNew <= Nocho;
when "0100100"  => sinval <= -44 ; cosval <= -90 ; NNew <= Nnueve; -- .15 ahead
when "0100011"  => sinval <= -35 ; cosval <= -94 ; NNew <= Nnueve;
when "0100010"  => sinval <= -26 ; cosval <= -97 ; NNew <= Nnueve;
when "0100001"  => sinval <= -16 ; cosval <= -99 ; NNew <= Nnueve;
when "0100000"  => sinval <= -6 ; cosval <= -100 ; NNew <= Nnueve;
when "0011111"  => sinval <= 4 ; cosval <= -100 ; NNew <= Ndiez; --NNew <= Nnueve;
when "0011110"  => sinval <= 14 ; cosval <= -99 ; NNew <= Ndiez;
when "0011101"  => sinval <= 24 ; cosval <= -97 ; NNew <= Ndiez;
when "0011100"  => sinval <= 33 ; cosval <= -94 ; NNew <= Ndiez;
when "0011011"  => sinval <= 43 ; cosval <= -90 ; NNew <= Ndiez;
when "0011010"  => sinval <= 52 ; cosval <= -86 ; NNew <= Nonce; --NNew <= Ndiez;
when "0011001"  => sinval <= 60 ; cosval <= -80 ; NNew <= Nonce;
when "0011000"  => sinval <= 68 ; cosval <= -74 ; NNew <= Nonce;
when "0010111"  => sinval <= 75 ; cosval <= -67 ; NNew <= Nonce;
when "0010110"  => sinval <= 81 ; cosval <= -59 ; NNew <= Nonce;
when "0010101"  => sinval <= 86 ; cosval <= -50 ; NNew <= Ndoce;--NNew <= Nonce;
when "0010100"  => sinval <= 91 ; cosval <= -42 ; NNew <= Ndoce; -- OK
when "0010011"  => sinval <= 95 ; cosval <= -32 ; NNew <= Ndoce;
when "0010010"  => sinval <= 97 ; cosval <= -23 ; NNew <= Ndoce;
when "0010001"  => sinval <= 99 ; cosval <= -13 ; NNew <= Ndoce;
when "0010000"  => sinval <= 100 ; cosval <= -3 ; NNew <= Nuno; --NNew <= Ndoce;
when "0001111"  => sinval <= 100 ; cosval <= 7 ; NNew <= Nuno; --extra
when "0001110"  => sinval <= 99 ; cosval <= 17 ; NNew <= Nuno;
when "0001101"  => sinval <= 96 ; cosval <= 27 ; NNew <= Nuno;
when "0001100"  => sinval <= 93 ; cosval <= 36 ; NNew <= Nuno;
when "0001011"  => sinval <= 89 ; cosval <= 45 ; NNew <= Ndos;--NNew <= Nuno;
when "0001010"  => sinval <= 84 ; cosval <= 54 ; NNew <= Ndos;
when "0001001"  => sinval <= 78 ; cosval <= 62 ; NNew <= Ndos; --.50 lag (ADJUST ACCORD. To DIRECTION)
when "0001000"  => sinval <= 72 ; cosval <= 70 ; NNew <= Ndos;
when "0000111"  => sinval <= 64 ; cosval <= 76 ; NNew <= Ndos;
when "0000110"  => sinval <= 56 ; cosval <= 83 ; NNew <= Ndos;
when "0000101"  => sinval <= 48 ; cosval <= 88 ;  NNew <= Ntres;--NNew <= Ndos;
when "0000100"  => sinval <= 39 ; cosval <= 92 ;  NNew <= Ntres;
when "0000011"  => sinval <= 30 ; cosval <= 96 ;  NNew <= Ntres;
when "0000010"  => sinval <= 20 ; cosval <= 98 ;  NNew <= Ntres;
when "0000001"  => sinval <= 10 ; cosval <= 100 ;  NNew <= Ntres;
when "0000000"  => sinval <= 0 ; cosval <= 100 ;   NNew <= Ncuatro;-- NNew <= Ntres;
when others   => sinval <= 0 ; cosval <= 100 ;  NNew <= Nuno; --%00
end case;
end if;
end process;
end deer;
```

The LUT code is shown in `sinLUT2019.vhd` above and it contains the best approximation of a sine and a cosine value multiplied by a factor of 100 in three decimal digits for every value of 10θ. As the voltage vector moves in the cartesian $\alpha\beta$ plane, its position is $0 \leq 10\theta < 63$. In other words, the LUT outputs rounded values of $100 \sin\theta$ and $100 \cos\theta$ for every input value of 10θ from 0 to 62. Although the values $\{0, 62\} \in \theta$ are intertwined in the code for a smooth transition, it is practical to state there are 63 positions. The real resolution of the designed PLL is therefore 0.1 [rad].

As the voltage vector completes a turn in 20 [ms] (it is a 50 [Hz] voltage wave for this prototype), the LUT block should ideally have passed through all its output values rather linearly in that time window. As 63 values need to be somewhat evenly distributed in a 20 [ms] window, it is desired that the LUT output values change every $20[ms]/63 = 317.46[\mu s]$, which is roughly 3.15 [kHz]. It is then expected that θ change value every period of $T_{pll} = 317.46[\mu s]$, or with a frequency $F_{pll} = 3.15$ [kHz]. Another task performed by `sinLUT` is to assign a region number $n(\theta)$ to each of the 63 values of θ, which means between 5 or 6 values of θ assigned to each of the 12 region numbers $n(\theta)$. The result is a 4-bit output `N2019`.

```
-- ClockEnable3150Hz.v --
module ClockEnable3150Hz(
input kplus, //aditional
input kminus, //aditional
input CLOCK_20,
output reg enable
);
reg [25:0] count;
always@(posedge CLOCK_20)
begin
if (count >= 6349+ 6349*kplus - 3175*kminus) // kplus (2*time) - kminus (time/2)
begin
enable <= 1;
count <= 0;
end
else
begin
enable <=0;
count <= count + 1'b1;
end
end
endmodule
```

By all means, it is impossible to obtain a perfect accuracy for two reasons: decimal representation of π never ends, and the grid frequency varies around 50 [Hz]. Nevertheless, the nature of a PLL (and the target of this PLL design) is to constantly correct itself. The way this PLL design corrects itself is by having a variable speed pulse: `ClockEnable3150Hz` block in Figure 5.22.

The variable speed algorithm was coded in verilog `.v` (after a long and persistent problem of speed control due to a lack of expertise in coding for FPGA). Yet, it is a relatively simple algorithm as shown in `ClockEnable3150Hz.v` above. The amount of Samples [Sa] required to step down a 20 [MHz] clock to a 3.15 [kHz] clock is $20[MHz]/3.15[kHz] \approx 6,349[Sa]$. There are three input variables: `kplus` to increase period lenght T_{pll} or 'brake', `kminus` to reduce period lenght T_{pll} or 'accelerate', and `CLOCK_20 = CLK_IN`. This means a 20 [MHz] clock enters the algorithm and an `enable` output is set to 1 when (as shown in the code) an internal counter reaches

```
count >= 6349 + 6349*kplus - 3175*kminus
```

where `kplus` and `kminus` are 1-bit each. The speed adjustment is achieved by varying the counter limit from a normal 6,349 to double the speed when `kplus` is one, or to half the speed when `kminus` is one.

```
-- counter2019.v --
module counter2019(
input VAR_CLK,
output reg enable,
output reg [5:0] sda
);
reg [25:0] count;
always@(posedge VAR_CLK)
begin
if (count >= 62)
begin
enable <= 1;
count <= 0;
sda <= count;
end
else
begin
enable <=0;
```

```
count <= count + 1'b1;
sda <= count;
end
end
endmodule
```

```
-- mult2019.chd --
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.numeric_std.all;
entity mult2019 is
port
(
-- Input ports
reset :  in std_logic;
clk : in std_logic;
sinm :  in integer range -100 to 100 := 0;
cosm :  in integer range -100 to 100 := 0;

--Output ports
valpf : out integer range -31071 to 31071 :=0;
vbetf : out integer range -31071 to 31071 :=0;
ialpf :  out integer range -8191 to 8191 :=0;
ibetf :  out integer range -8191 to 8191 :=0
);
end mult2019;
```

```
architecture neverforget of mult2019 is
constant vpeak : integer:= 28284;
constant ipeak : integer:= 2500;
begin
process(clk,reset)
begin
if (reset = '0') then
valpf   <= 0; -- currents
vbetf   <= 0;
ialpf <= 0;
ibetf <=0;
elsif(rising_edge(clk)) then
valpf <= vpeak*sinm/100; -- remove sin(100) = [mV]
vbetf <= vpeak*cosm/100; -- remove cos(100) = [mV]

ialpf <= ipeak*sinm/100;
ibetf <= ipeak*cosm/100;
end if;
end process;
end neverforget;
```

The block `counter2019` was also coded in verilog `.v` as shown in `counter2019.v`. This is also a counter and its only purpose is to output a count from 0 to 62 repeatedly every time a pulse from `ClockEnable3150Hz` is received. This number is then sent to `sinLUT2019` described earlier.

The above allows to adjust the speed of the pulse stream `enable` being counted in `counter2019`, which is in turn proportional to the speed at which the vector position θ changes, causing the sine and cosine from `sinLUT2019` values to change accordingly. Finally, these values enter into the block `mult2019` which code is shown above, only to be multiplied by the voltage peak value $V_{pk} = 28,268$ [mV] and divided by 100 (to remove the same factor from the sine and cosine stage). This way accuracy is not lost and the block outputs the resulting grid follower voltages `vaF` and `vbF`. Another couple of signals output from this block, which were only for tests.

```
-- quart2019a.vhd --
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.numeric_std.all;
entity quart2019a is
port
(
-- Input ports
reset :  in std_logic;
clk : in std_logic;
alpha : in integer range -31071 to 31071 :=0;
-- Output ports
a_pos : out std_logic
);
end quart2019a;
architecture apos1 of quart2019a is
constant zero  :integer := 0;
begin
process(reset,clk)
begin
if (reset = '0') then
a_pos   <= '0';
--beta   <= zero;
elsif(rising_edge(clk)) then
if alpha >= zero then
a_pos <= '1';
else
a_pos <= '0';
end if;
end if;
end process;
end apos1;
--
--end
```

```
-- quart2019b --
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.numeric_std.all;
entity quart2019b is
port
(
-- Input ports
reset :  in std_logic;
clk : in std_logic;
beta : in integer range -31071 to 31071 :=0;
-- Output ports
b_pos : out std_logic
);
end quart2019b;
architecture bpos1 of quart2019b is
constant zero  :integer := 0;
begin
process(reset,clk)
begin
if (reset = '0') then
b_pos   <= '0';
--beta   <= zero;
elsif(rising_edge(clk)) then
if beta >= zero then
b_pos <= '1';
else
b_pos <= '0';
end if;
end if;
end process;
end bpos1;
--
--end
```

```
-- quart2019ar.vhd --
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.numeric_std.all;
entity quart2019ar is
port
(
-- Input ports
reset :  in std_logic;
clk : in std_logic;
alpha : in integer range -31071 to 31071 :=0; --2^16
-- Output ports
a_pos : out std_logic
);
end quart2019ar;
architecture apos11 of quart2019ar is
constant zero  :integer := 0;
begin
process(reset,clk)
begin
if (reset = '0') then
a_pos   <= '0';
--beta   <= zero;
elsif(rising_edge(clk)) then
if alpha >= zero then
a_pos <= '1';
else
a_pos <= '0';
end if;
end if;
end process;
end apos11;
--
--end
```

```
-- quart2019br.vhd --
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.numeric_std.all;
entity quart2019br is
port
(
-- Input ports
reset :  in std_logic;
clk : in std_logic;
beta : in integer range -31071 to 31071 :=0; --2^17
-- Output ports
b_pos : out std_logic
);
end quart2019br;
architecture apos22 of quart2019br is
constant zero  :integer := 0;
begin
process(reset,clk)
begin
if (reset = '0') then
b_pos   <= '0';
--beta   <= zero;
elsif(rising_edge(clk)) then
if beta >= zero then
b_pos <= '1';
else
b_pos <= '0';
end if;
end if;
end process;
end apos22;
--
--end
```

On the left hand side of Figure5.22, `valpha` and `vbeta` are the real grid voltages, and the blocks `quart2019a.vhd` and `quart2019b.vhd` are ZVC detectors for each, respectively. When `valpha` is greater than or equal to zero, `alpos` output is one; otherwise it is zero. The same applies to `bepos` output.

Codes `quart2019ar.vhd` and `quart2019br.vhd` embedded in their respective blocks perform exactly the same ZVC function for `vaF` and `vbF`. The latter pair is the 'locked' voltage or grid follower voltage (in real time), which is produced by the PLL and feedback from `mult2019.vhd`.

Next, the `QIV2019.vhd` block reads 2-bits from the ZVCs of the real grid voltages as `gridAB`, and reads 2-bits from the ZVCs of the grid follower voltages as `refAB`. Here is where the synchronisation technique can be coded as there is enough data to know in which Qt (Quadrant) each voltage vector is located. This is following the standard quadrant numeration in a cartesian plane (α placed in the horizontal axis and β in the vertical axis), where QtI is at the top-right and $QtII$, $QtIII$, and $QtIV$ follow in an ACW direction.

`QIV2019.vhd` is focused on $QtIV$ with an speed logic as follows. If grid follower voltages `refAB` are located in $QtIV$, do the following: If real grid voltages `gridAB` are in

- $QtIV$: keep speed.
- $QtIII$: break.
- $QtII$: break.
- QtI: accelerate.

In the code `QIV2019.vhd`, `acel` stands for 'accelerate' and `fren` stands for 'break'. These two signals are then input respectively into `kminus` and `kplus` of the next

block to perform what they are intended to. The whole process described in this subsection then iterates.

Evidently, better algorithms can be designed in order to keep track of the peak voltage, a faster update of θ in real time, or deal with various grid issues such as noise and imbalance. This algorithm is only fit to a set voltage value and vulnerable to other grid issues, and although the PLL is an important part of this work, there is no intention to make a contribution in this field.

```
-- QIV2019.vhd --
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.numeric_std.all;
entity QIV2019 is
port
(
-- Input ports
reset :  in std_logic;
clk : in std_logic;

gridAB :
in std_logic_vector (1 downto 0);
refAB :
in std_logic_vector (1 downto 0);
-- Output ports
acel : out std_logic;
fren :  out std_logic
);
end QIV2019;
```

```
architecture unodos3 of QIV2019 is
constant zero  :integer := 0;
begin
process(reset,clk)
begin
if (reset = '0') then
acel <= '0'; fren <= '0'; -- Stay
elsif(rising_edge(clk)) then
if gridAB = "10" then -- QIV
case refAB is
when "10" => acel <= '0'; fren <= '0'; -- QIV, stay
when "00" => acel <= '0'; fren <= '1'; -- QIII, refAB LEADING
when "01" => acel <= '0'; fren <= '1'; -- QII, refAB LEADING
when "11" => acel <= '1'; fren <= '0'; -- QI, refAB LAGGING
when others => acel <= '0'; fren <= '0'; -- Stay
end case;
else
acel <= '0'; fren <= '0'; -- Stay
end if;
end if;
end process;
end unodos3;
```

5.2.7 Hysteresis

The hysteresis part shown in Figure 5.23 receives the `Perr`, `Qerr` and `Cerr` generated by the `PQCerrs` from the *Calculations* part. The block `hysteresis4.vhd` is shown below, where the hysteresis limits are defined as constants `PHval`, `QHval`, and `CHval`. Each value works for both, upper and lower error limits. Then signals `ph`, `qh`, and `ch` make their way to individual blocks `outsig1` in order to produce a latch effect (coded in an amateur level), to finally output 1-bit signals `P`, `Q`, and `C`, which represent P_d, Q_d, and C_d from the theory. The `outsig1.vhd` is also shown below.

```
-- hysteresis4.vhd--
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.numeric_std.all;
entity hysteresis4 is
port
(
-- Input ports
reset :  in std_logic;
clk : in std_logic;
Perr : in integer range -524287 to 524287 :=0;
Qerr : in integer range -524287 to 524287 :=0;
Cerr : in integer range -65535 to 65535 :=0;
-- Output ports
ph : out std_logic_vector(1 downto 0);
qh : out std_logic_vector(1 downto 0);
ch : out std_logic_vector(1 downto 0)
);
end hysteresis4;
architecture hyst4 of hysteresis4 is
constant PHval  :integer := 5000;
-- +/- 5 [W] Hysteresis
constant QHval  :integer := 5000;
-- +/- 5 [W] Hysteresis
constant CHval  :integer := 5000;
-- +/- 5 [V] Hysteresis
constant twoo  :integer := 2;
constant cero  :integer := 0;
begin
process(reset,clk)
begin
if (reset = '0') then
ph <= "11";
qh <= "10";
ch <= "10";
elsif(rising_edge(clk)) then
if Perr <= cero then
if Perr < -PHval then
ph <= "10";
else
ph <= "00";
end if;
else
if Perr> PHval then
ph <= "11";
else
ph <= "00";
end if;
end if;
if Qerr <= cero then
if Qerr < -QHval then
qh <= "10";
else
qh <= "00";
end if;
else
if Qerr> QHval then
qh <= "11";
else
qh <= "00";
end if;
end if;
if Cerr <= cero then
if Cerr < -CHval/twoo then
ch <= "10";
else
ch <= "00";
end if;
else
if Cerr> CHval/twoo then
ch <= "11";
else
ch <= "00";
end if;
end if;
end if;
end process;
end hyst4;
```

```
-- outsig1.vhd--
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity outsig1 is
port (
-- Input ports
   clk      : in std_logic;
   RESN     : in std_logic;
edge : in std_logic;
redge : in std_logic;
   -- Output ports
Osig     : out std_logic
   );
end entity outsig1;
architecture check1 of outsig1 is
   constant zero  : std_logic:='0';
constant onee : std_logic:='1';
signal latch  : std_logic;
begin
process(clk,RESN,edge,redge)
begin
if RESN = '0' then
Osig <= zero;
   elsif rising_edge(clk) and edge = '1' then
```

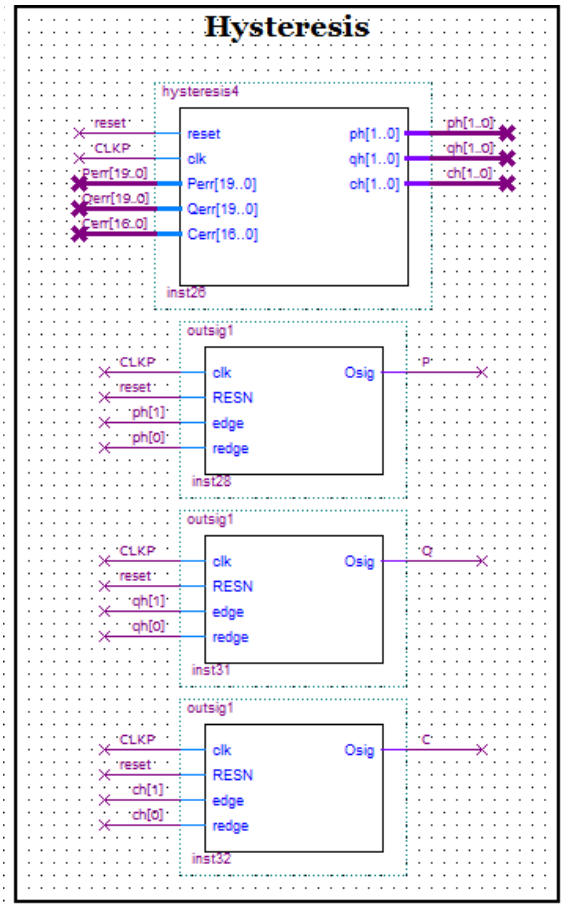


FIGURE 5.23: *Quartus*: Hysteresis

```
case redge is
when '0' => Osig <= zero; latch <= zero;
when others => Osig <= onee; latch <= onee;
end case;

elsif rising_edge(clk) and edge = '0' then
Osig <= latch;
end if;
end process;
end check1;
```

5.2.8 Vector Tables

The VHDL code of the new switching table for DPC requires two inputs. One input is $n(\theta)$ and the other is *Index*, which depends on P_d, Q_d, and C_d, as shown in Table 4.5. The block `vectable2` from Figure 5.24 is divided in two parts as shown in the code below. The first part calculates the `index` value from the inputs

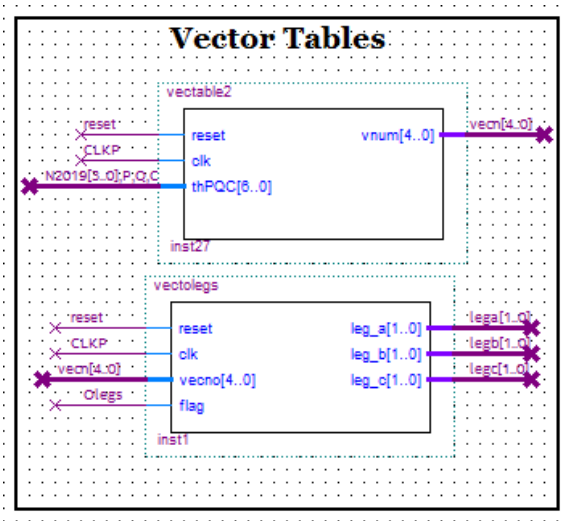


FIGURE 5.24: *Quartus*: Vector tables

and the second part assigns a vector number in binary for each `N2019` and `index` combination, translating Table 4.5. Technically, there is an input formed by 4-bits representing $n(\theta)$ joined by three 1-bit representing P_d, Q_d, and C_d, as well as an output representing $\{1, 2, 3...24\} \in V_n$ in 5-bits. All of these binary signals can now be easily understood by a human eye, as they represent logic signals and low-bit numbers.

```
-- vectable2.vhd--
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity vectable2 is
port
(
-- Input ports
reset : in  STD_LOGIC;
clk : in  std_logic;
thPQC : in  STD_LOGIC_VECTOR(6 DOWNTO 0);
-- Output ports
vnum : out STD_LOGIC_VECTOR(4 DOWNTO 0)
);
end vectable2;
architecture tabs of vectable2 is
signal P :std_logic;
signal Q :std_logic;
signal C :std_logic;
signal theta :std_logic_vector(3 DOWNTO 0);
signal index :std_logic_vector(2 DOWNTO 0);
signal v0,v1,v2,v3,v4,v5,v6,v7,v8 : std_logic_vector(4 downto 0);
signal v9,v10,v11,v12,v13,v14,v15 : std_logic_vector(4 downto 0);
signal v16,v17,v18,v19,v20,v21,v22: std_logic_vector(4 downto 0);
signal v23,v24,v25,v26 : std_logic_vector(4 downto 0);
begin
P <= thPQC(2);
Q <= thPQC(1);
C <= thPQC(0);
```

```
theta <= thPQC(6 DOWNTO 3);
process(clk,reset,P,Q,C,theta)
begin
if (reset = '0') then
vnum <= v0;
elsif(rising_edge(clk)) then
if (P = '0') and (Q = '0') then
if (C = '0') then index <= "000";
else index <= "001";
end if;
else
if (P = '0') and (Q = '1') then
if (C = '0') then index <= "010";
else index <= "011";
end if;
else
if (P = '1') and (Q = '0') then
index <= "100";
else
if (P = '1') and (Q = '1') then
index <= "101";
else index <= "111";
end if;
end if;
end if;
end if;
case index is
when "000" =>
case theta is  --index = 0
when "0001" => vnum <= "01001"; --th1.v9
when "0010" => vnum <= "10011"; --th2.v19
when "0011" => vnum <= "01010"; --th3.v10
when "0100" => vnum <= "10101"; --th4.v21
when "0101" => vnum <= "01011"; --th5.v11
when "0110" => vnum <= "10111"; --th6.v23
when "0111" => vnum <= "01100"; --th7.v12
when "1000" => vnum <= "01101"; --th8.v13
when "1001" => vnum <= "00111"; --th9.v7
when "1010" => vnum <= "01111"; --th10.v15
when "1011" => vnum <= "01000"; --th11.v8
when "1100" => vnum <= "10001"; --th12.v17
when others => vnum <= "00000"; --th?.v0
end case;
when "001" =>
case theta is --index = 1
when "0001" => vnum <= "01001"; -- V9
when "0010" => vnum <= "10100"; -- V20
when "0011" => vnum <= "01010"; -- V10
when "0100" => vnum <= "10110"; -- V22
when "0101" => vnum <= "01011"; -- V11
when "0110" => vnum <= "11000"; -- V24
when "0111" => vnum <= "01100"; -- V12
when "1000" => vnum <= "01110"; -- V14
when "1001" => vnum <= "00111"; -- V7
when "1010" => vnum <= "10000"; -- V16
when "1011" => vnum <= "01000"; -- V8
when "1100" => vnum <= "10010"; -- V18
when others => vnum <= "00000"; --th?.v0
end case;
when "010" =>
case theta is --index = 2
when "0001" => vnum <= "10101"; -- V21
when "0010" => vnum <= "01011"; -- V11
when "0011" => vnum <= "10111"; -- V23
when "0100" => vnum <= "01100"; -- V12
when "0101" => vnum <= "01101"; -- V13
when "0110" => vnum <= "00111"; -- V7
when "0111" => vnum <= "01111"; -- V15
when "1000" => vnum <= "01000"; -- V8
when "1001" => vnum <= "10001"; -- V17
when "1010" => vnum <= "01001"; -- V9
```

```
when "1011" => vnum <= "10011"; -- V19
when "1100" => vnum <= "01010"; -- V10
when others => vnum <= "00000"; --th?.v0
end case;
when "011" =>
case theta is -- index = 3
when "0001" => vnum <= "10110"; -- V22
when "0010" => vnum <= "01011"; -- V11
when "0011" => vnum <= "11000"; -- V24
when "0100" => vnum <= "01100"; -- V12
when "0101" => vnum <= "01110"; -- V14
when "0110" => vnum <= "00111"; -- V7
when "0111" => vnum <= "10000"; -- V16
when "1000" => vnum <= "01000"; -- V8
when "1001" => vnum <= "10010"; -- V18
when "1010" => vnum <= "01001"; -- V9
when "1011" => vnum <= "10100"; -- V20
when "1100" => vnum <= "01010"; -- V10
when others => vnum <= "00000"; --th?.v0
end case;
when "100" =>
case theta is -- index = 4
when "0001" => vnum <= "00111"; -- V7
when "0010" => vnum <= "00010"; -- V2
when "0011" => vnum <= "01000"; -- V8
when "0100" => vnum <= "00011"; -- V3
when "0101" => vnum <= "01001"; -- V9
when "0110" => vnum <= "00100"; -- V4
when "0111" => vnum <= "01010"; -- V10
when "1000" => vnum <= "00101"; -- V5
when "1001" => vnum <= "01011"; -- V11
when "1010" => vnum <= "00110"; -- V6
when "1011" => vnum <= "01100"; -- V12
when "1100" => vnum <= "00001"; -- V1
when others => vnum <= "00000"; --th?.v0
end case;
when "101" =>
case theta is -- index = 5
when "0001" => vnum <= "00001"; -- V1
when "0010" => vnum <= "00111"; -- V7
when "0011" => vnum <= "00010"; -- V2
when "0100" => vnum <= "01000"; -- V8
when "0101" => vnum <= "00011"; -- V3
when "0110" => vnum <= "01001"; -- V9
when "0111" => vnum <= "00100"; -- V4
when "1000" => vnum <= "01010"; -- V10
when "1001" => vnum <= "00101"; -- V5
when "1010" => vnum <= "01011"; -- V11
when "1011" => vnum <= "00110"; -- V6
when "1100" => vnum <= "01100"; -- V12
when others => vnum <= "00000"; --th?.v0
end case;
when others => vnum <= "00000";
   end case;
end if;
end process;
end tabs;
```

The code `vectolegs.vhd` contains a LUT assigning a leg value to each vector. The leg value represents s_x in equation (2.1) and the notation `N=-1 O=0` and `P=1` was used. As three possible values per leg are possible, the output is 2-bit long per leg `lega`, `legb`, and `legc`.

```
-- vectolegs.vhd--
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity vectolegs is
port
(
-- Input ports
reset : in  STD_LOGIC;
clk : in  std_logic;
vecno : in  STD_LOGIC_VECTOR(4 DOWNTO 0);
flag  : in  STD_LOGIC;
-- Output ports
leg_a : out STD_LOGIC_VECTOR(1 DOWNTO 0);
leg_b : out STD_LOGIC_VECTOR(1 DOWNTO 0);
leg_c : out STD_LOGIC_VECTOR(1 DOWNTO 0)
);
end vectolegs;
architecture ARCH of vectolegs is
constant N : std_logic_vector(1 downto 0):="00"; -- Negative (-1)
   constant O : std_logic_vector(1 downto 0):="01"; -- Neutral (0)
   constant P : std_logic_vector(1 downto 0):="10"; -- Positive (+1)
begin
process(clk,vecno,reset)
begin
if(reset = '0' or flag = '1') then
leg_a <= O;
leg_b <= O;
leg_c <= O;
elsif(rising_edge(clk)) then
case vecno is
-- Zero Vectors 000, -1-1-1, 111
when "00000" => leg_a <= O; leg_b <= O; leg_c <= O; -- V0
when "11001" => leg_a <= N; leg_b <= N; leg_c <= N; -- V25
when "11010" => leg_a <= P; leg_b <= P; leg_c <= P; -- v26
-- Large Vectors
when "00001" => leg_a <= P; leg_b <= N; leg_c <= N; -- V1
when "00010" => leg_a <= P; leg_b <= P; leg_c <= N; -- V2
when "00011" => leg_a <= N; leg_b <= P; leg_c <= N; -- V3
when "00100" => leg_a <= N; leg_b <= P; leg_c <= P; -- V4
when "00101" => leg_a <= N; leg_b <= N; leg_c <= P; -- V5
when "00110" => leg_a <= P; leg_b <= N; leg_c <= P; -- V6
-- Medium Vectors
when "00111" => leg_a <= P; leg_b <= O; leg_c <= N; -- V7
when "01000" => leg_a <= O; leg_b <= P; leg_c <= N; -- V8
when "01001" => leg_a <= N; leg_b <= P; leg_c <= O; -- V9
when "01010" => leg_a <= N; leg_b <= O; leg_c <= P; -- V10
when "01011" => leg_a <= O; leg_b <= N; leg_c <= P; -- V11
when "01100" => leg_a <= P; leg_b <= N; leg_c <= O; -- V12
-- Small Vectors C1/C2 Discharge pairs
when "01101" => leg_a <= P; leg_b <= O; leg_c <= O; -- V13  C1d
when "01110" => leg_a <= O; leg_b <= N; leg_c <= N; -- V14  C2d
when "01111" => leg_a <= O; leg_b <= O; leg_c <= N; -- V15
when "10000" => leg_a <= P; leg_b <= P; leg_c <= O; -- V16
when "10001" => leg_a <= O; leg_b <= P; leg_c <= O; -- V17
when "10010" => leg_a <= N; leg_b <= O; leg_c <= N; -- V18
when "10011" => leg_a <= N; leg_b <= O; leg_c <= O; -- V19
when "10100" => leg_a <= O; leg_b <= P; leg_c <= P; -- V20
when "10101" => leg_a <= O; leg_b <= O; leg_c <= P; -- V21
when "10110" => leg_a <= N; leg_b <= N; leg_c <= O; -- V22
when "10111" => leg_a <= O; leg_b <= N; leg_c <= O; -- V23
when "11000" => leg_a <= P; leg_b <= O; leg_c <= P; -- V24
when others => leg_a <= O; leg_b <= O; leg_c <= O; -- V0
end case;
end if;
end process;
end ARCH;
```

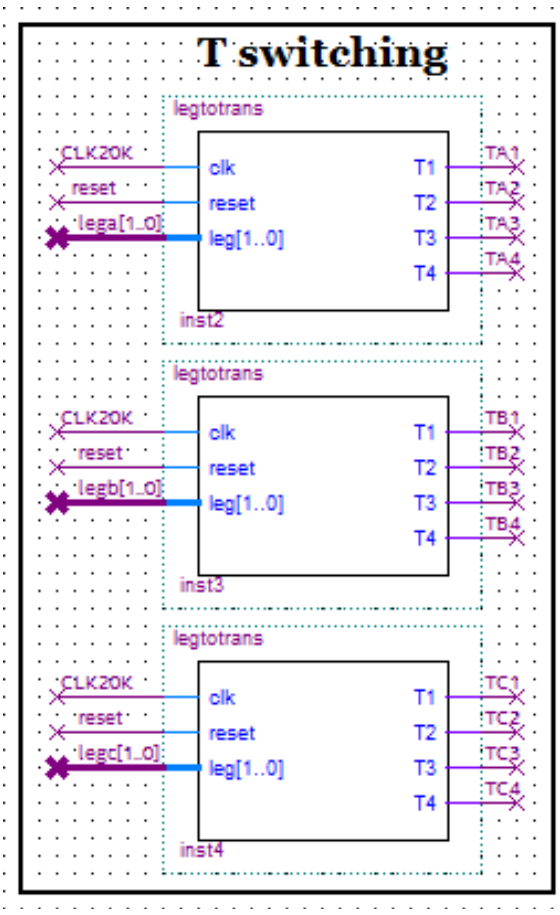


FIGURE 5.25: *Quartus*: Transistor switching

5.2.9 Transistor Switching

Once the leg values are defined in the code, digital translation to the 12 transistors is required. To achieve this, every 2-bit leg value is input to a `legtotrans` block as shown in Figure 5.25. The code `legtotrans.vhd` is self explanatory, considering inputs [`00 01 10`] correspond to leg values $s_x = [-1 \quad 0 \quad +1]$, respectively.

Different from other codes, `legtotrans.vhd` is driven by `CLK20k`, which is a 20 [kHz] clock. This is the maximum frequency the transistors in the inverter can operate according to the manufacturer and it must always be considered for algorithm design in converters. This particular inverter contains a protection in its board which forces a delay to the transistor switching in case the switching period is higher than 1/20[kHz].

```
-- legtotrans.vhd--
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity legtotrans is
port
(
-- Input ports
clk : in  std_logic;
reset : in  STD_LOGIC;
leg : in  STD_LOGIC_VECTOR(1 DOWNTO 0);
-- Output ports
T1 : out STD_LOGIC;
T2 : out STD_LOGIC;
T3 : out STD_LOGIC;
T4 : out STD_LOGIC
);
end legtotrans;
architecture ARCH1 of legtotrans is
-- Concurrent statements (order independent)
signal TOFF : std_logic:= '0'; -- 4T OFF
    signal TON  : std_logic:= '1';
begin
process(clk,leg,reset) -- Sequential statements (order matters)
begin
if(reset = '0') then
T1<= TOFF; T2<=TOFF; T3<=TOFF;  T4<=TOFF;
elsif (falling_edge(clk)) and (reset = '1') then
case leg is
when "00" => T1<= TOFF; T2<=TOFF; T3<=TON;  T4<=TON;
when "01" => T1<= TOFF; T2<=TON;  T3<=TON;  T4<=TOFF;
when "10" => T1<= TON;  T2<=TON;  T3<=TOFF; T4<=TOFF;
when others => T1<= TOFF; T2<=TOFF; T3<=TOFF;  T4<=TOFF;
end case;
end if;
end process;
end ARCH1;
```

5.2.10 Troubleshooting

The troubleshooting part is the ever changing part as it provides temporary calculations or indicators to find bugs or errors in the code. It is also works as a laboratory, where some temporary code is eventually deleted and some other become another piece of the project. The setup shown in Figure 5.26 perform three tasks: current monitoring, capacitor monitoring, and latch implementation.

The code `imax.vhd` monitors whether any the five current values i_{abc} and $i_{\alpha\beta}$ exceeds 6 [A]. If it does, it sends a pulse through `trig` (this is not latched) and it also sends a pulse indicating which current value(s) exceeded the limit through `is_trig` (not latched either). The latter is sent to *OUT:LEDS* in subsection 5.2.1 for button-activated LED latching.

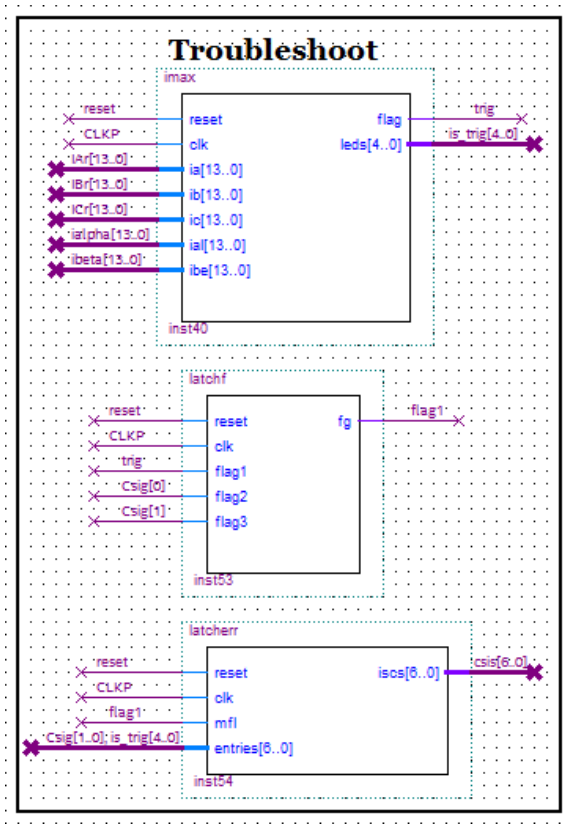


FIGURE 5.26: *Quartus*: Troubleshooting

```
-- imax.vhd --
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.numeric_std.all;
entity imax is
port
(
-- Input ports
reset :  in std_logic;
clk : in std_logic;
ia     : in integer range -8191 to 8191 :=0;
ib     : in integer range -8191 to 8191 :=0;
ic     : in integer range -8191 to 8191 :=0;
ial    : in integer range -8191 to 8191 :=0;
ibe    : in integer range -8191 to 8191 :=0;
-- Output ports
flag     :  out std_logic;
leds :  out std_logic_vector(4 downto 0)
);
end imax;
architecture currs of imax is
constant zero   :integer := 0;
constant six   :integer := 6000;
begin
process(reset,clk)
begin
if (reset = '0') then
flag <= '0';
elsif(rising_edge(clk)) then
if abs(ia) > six then
leds <= "00100";
flag <= '1';
elsif abs(ib) > six then
```

```
leds <= "00010";
flag <= '1';
elsif abs(ic) > six then
leds <= "00001";
flag <= '1';
elsif abs(ial) > six then
    leds <= "10000";
flag <= '1';
elsif abs(ibe) > six then
leds <= "01000";
flag <= '1';
else
leds <= "00000";
flag <= '0';
end if;
end if;
end process;
end currs;
```

The `latchf.vhd` code monitors the capacitor voltage levels using digital inputs described earlier. It also takes `trig` forming a cascading effect for latching a fault indicator. This block only outputs a 1-bit signal to indicate whether there has been a triggered value or not. Finally, the `latcherr.vhd` takes `flag1`, `Csig`, and `is_trig` and whichever value gives a pulse, it latches a '1' to the corresponding signal in the `csis` output. This is then directed to the *OUT:LEDS* part to provide LED indicators.

```
-- latchf.vhd --
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.numeric_std.all;
entity latchf is
port
(
-- Input ports
reset :  in std_logic;
clk : in std_logic;
flag1 : in std_logic;
flag2 :  in std_logic;
flag3 : in std_logic;
-- OUtput ports
fg :  out std_logic
);
end latchf;
architecture imp of latchf is
signal latch   :std_logic := '0';
begin
process(reset,clk)
begin
if (reset = '0') then
latch <= '0';
elsif(rising_edge(clk) and flag1 = '1') then
latch <= '1';  fg <= '1';
elsif(rising_edge(clk) and flag2 = '1') then
latch <= '1';  fg <= '1';
elsif(rising_edge(clk) and flag3 = '1') then
latch <= '1';  fg <= '1';
```

```
elsif rising_edge(clk) then
if latch = '0' then
fg <= '0';
else
fg <='1';
end if;
end if;
end process;
end imp;
```

```
--  latcherr.vhd --
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.numeric_std.all;
entity latcherr is
port
(
-- Input ports
reset :  in std_logic;
clk : in std_logic;
mfl : in std_logic;
entries  :  in std_logic_vector(6 downto 0);
-- OUtput ports
iscs :  out std_logic_vector(6 downto 0)
);
end latcherr;
architecture impp of latcherr is
signal latch   : std_logic_vector(6 downto 0);
signal hola    : std_logic := '0';
begin
process(reset,clk,mfl,entries)
begin
if (reset = '0') then
latch <= "0000000";
elsif (rising_edge(clk) and mfl = '1') then
hola <= '1';
case entries is
when "0000001" => latch <= "0000001";  iscs <= "0000001";
when "0000010" => latch <= "0000010";  iscs <= "0000010";
when "0000100" => latch <= "0000100";  iscs <= "0000100";
when "0001000" => latch <= "0001000";  iscs <= "0001000";
when "0010000" => latch <= "0010000";  iscs <= "0010000";
when "0100000" => latch <= "0100000";  iscs <= "0100000";
when "1000000" => latch <= "1000000";  iscs <= "1000000";
when others => iscs <= "0000000";
end case;
elsif rising_edge(clk) then
if hola = '1' then
iscs <= latch;
else
iscs <= "0000000";
end if;
end if;
end process;
end impp;
```

5.2.11 DAC: Signal Display

This part contains all the software related to signal visualisation in an osciloscope.

As stated previously, an AD7398 with four 12-bit voltage output is used as a DAC

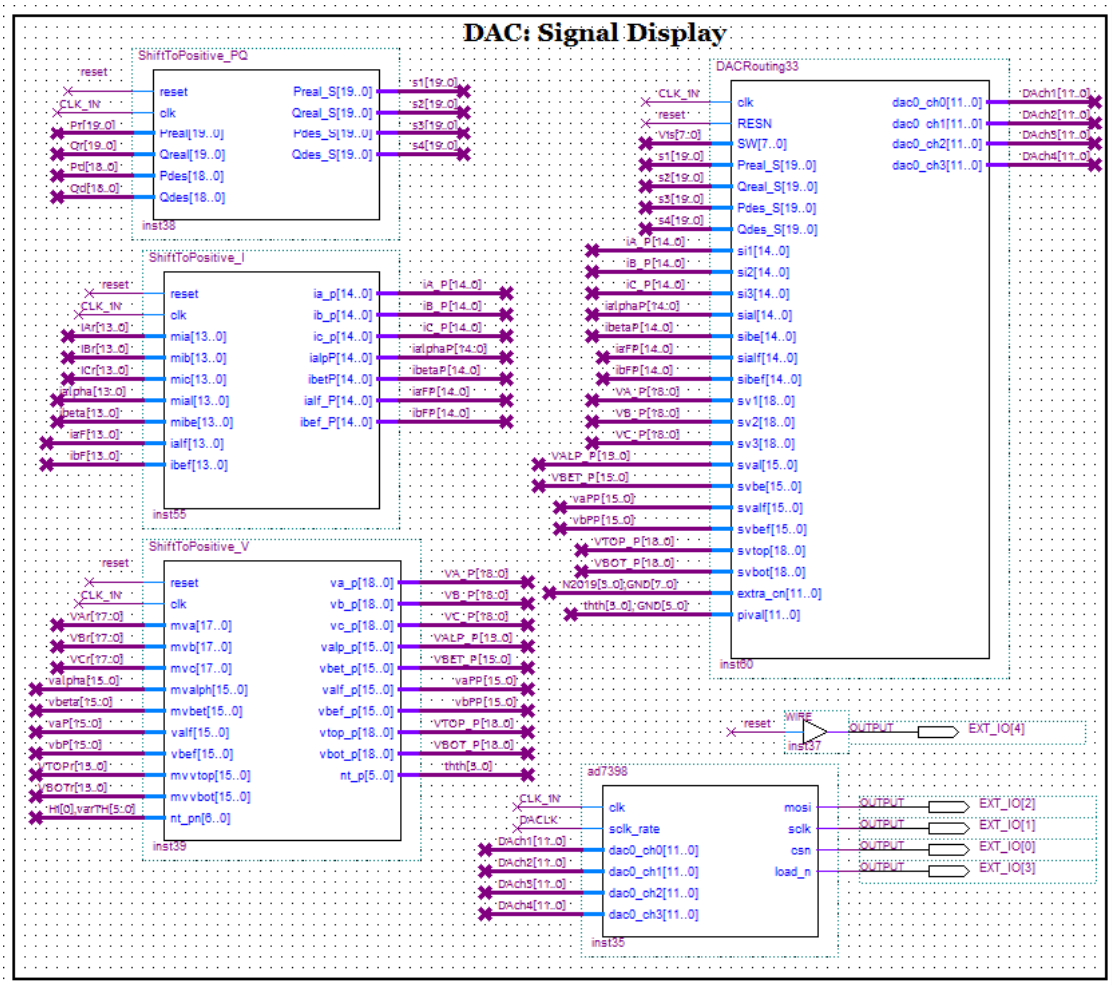


FIGURE 5.27: *Quartus*: DAC Signal display

(Digital to Analog Converter). As this DAC is configured to support only positive numbers, all the signals of interest have to be 'shifted' to positive quantities. Figure 5.27 shows the shifting part on the left side while the routing and conditioning is shown on the right hand side.

On the right hand side, the block `ad7398` is shown and it indeed receives four 12-bit inputs. The maximum decimal number allowed results in $2^{12} = 4,096$. However, there is only one serial input pin for the four channels and it requires data streams of 16-bits. The function of the `ad7398.vhd` code is to pass data to the DAC managing this extra bits for validation and internal register routing, where `mosi` is the data, `sclk` is the clock, `csn` controls when the data is allowed in,

and `load_n` enables four-channel simultaneous updates. Additionally, above the block `ad7398`, there is a separate pin `EXT_IO[4]` which is connected to the power pin VDD of the DAC at 3.3[V] directly from the FPGA board. The grounding is naturally provided by any of the other pins available in the serial port from the FPGA board.

```
-- ad7398.vhd --
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity ad7398 is

port (
   clk         : in std_logic;
   sclk_rate   : in std_logic;
   dac0_ch0    : in std_logic_vector(11 downto 0);
   dac0_ch1    : in std_logic_vector(11 downto 0);
   dac0_ch2    : in std_logic_vector(11 downto 0);
   dac0_ch3    : in std_logic_vector(11 downto 0);
   mosi        : out std_logic;
   sclk        : out std_logic;
   csn         : out std_logic;
   load_n      : out std_logic
);
end entity ad7398;
architecture first of ad7398 is
   signal clk_en        : std_logic;
   signal count         : unsigned(5 downto 0)  := (others=>'0');
   signal channel_num   : unsigned(1 downto 0)  := (others=>'0');
   signal channel_num_d : unsigned(channel_num'range);
   signal ld_data       : std_logic_vector(15 downto 0);
   signal sclk_rate_d   : std_logic;
   signal cs_n_int      : std_logic;
   signal cs_n_int_d    : std_logic;
   signal load_n_int    : std_logic;
   signal shift         : std_logic_vector(15 downto 0);
begin

cken: process (clk)
begin
   if rising_edge(clk) then
      if (sclk_rate = '1' and sclk_rate_d = '0') then
         clk_en   <= '1';
      else
         clk_en   <= '0';
      end if;
      sclk_rate_d <= sclk_rate;
   end if;
end process cken;
cntr: process(clk)
begin
   if rising_edge(clk) then
      if (clk_en = '1') then
         if (count(count'high) = '1') then
            count(count'high)   <= '0';
            count(count'high-1 downto 0)  <= (others=>'1');
            channel_num <= channel_num + 1;
            case (channel_num) is
               when "01"   => ld_data(11 downto 0)   <= dac0_ch1;
                              ld_data(13 downto 12)  <= "01";
               when "10"   => ld_data(11 downto 0)   <= dac0_ch2;
                              ld_data(13 downto 12)  <= "10";
               when "11"   => ld_data(11 downto 0)   <= dac0_ch3;
```

```
                        ld_data(13 downto 12)  <= "11";
               when others => ld_data(11 downto 0)   <= dac0_ch0;
                        ld_data(13 downto 12)  <= "00";
            end case;
            ld_data(13 downto 12)  <= std_logic_vector(channel_num);

         else
            count <= count - 1;
         end if;
         channel_num_d  <= channel_num;
      end if;
   end if;
end process cntr;
ld_data(15 downto 14)  <= "00";
csgen: process(clk)
begin
   if rising_edge(clk) then
      case to_integer(count) is
         when 17  => cs_n_int   <= '0';
         when 1   => cs_n_int   <= '1';
         when others  => null;
      end case;
   end if;
end process csgen;
loadgen: process(clk)
begin
   if rising_edge(clk) then
      case to_integer(count) is
         when 17  => load_n_int  <= '1';
         when 0   => load_n_int  <= '0';
         when others  => null;

      end case;
   end if;
end process loadgen;
shft: process(clk)
begin
   if rising_edge(clk) then
      if (clk_en = '1') then
         if (cs_n_int = '0') then
            shift <= shift(shift'high-1 downto 0) & '0';
         else
            shift <= ld_data;
         end if;
         cs_n_int_d  <= cs_n_int;
         csn      <= cs_n_int;
         load_n   <= load_n_int;
         mosi  <= shift(shift'high);
      end if;
   end if;
end process shft;
sclk  <= not sclk_rate and not cs_n_int_d;
end first;
```

This DAC was first tested at a frequency of 39 [kHz], and it was gradually increased until it was set at approximately 1.6 [MHz] to drive this device. This leaves approximately 400 [MHz] per channel, which is acceptable for this prototype. Besides, the DAC begins to draw significantly more current when exceeding the 1[MHz] mark, according to the datasheet, and due to the power restrictions of the FPGA, it was decided not to put more load on it as a precaution. Therefore, a

dedicated clock `DACLK` was configured to operate at approximately 1.6 [MHz] to drive this device.

The DAC will only take what is in the 12-bit range only. If the signal exceeds the limits from the software (and produces more bits), the signal will start from zero again. However, if the signal consistently exceeds the limits, it will produce a discontinuous signal in the oscilloscope. For example, moving from the very top to the very bottom and viceversa, which could disguise as a value suddenly dropping or topping, but the value would be actually swinging somewhere in the maximum limit or between negative numbers. There might also be a slight error due to the scaling from the 3.3 [V] into the oscilloscope.

```
-- DACRouting33.vhd --
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity DACRouting33 is
port (
-- Input ports
    clk      : in std_logic;
    RESN     : in std_logic;
SW       : in std_logic_vector (7  downto 0);
Preal_S : in std_logic_vector (19 downto 0); --..87 = 18 / 88 = 19
Qreal_S : in std_logic_vector (19 downto 0);
Pdes_S : in std_logic_vector (19 downto 0);
Qdes_S : in std_logic_vector (19 downto 0);
si1 : in std_logic_vector (14 downto 0); --2^17
si2 : in std_logic_vector (14 downto 0);
    si3 : in std_logic_vector (14 downto 0);
sial : in std_logic_vector (14 downto 0);
    sibe : in std_logic_vector (14 downto 0);
sialf : in std_logic_vector (14 downto 0);
    sibef : in std_logic_vector (14 downto 0);
sv1 : in std_logic_vector (18 downto 0); --2^17
sv2 : in std_logic_vector (18 downto 0);
    sv3 : in std_logic_vector (18 downto 0);
sval : in std_logic_vector (15 downto 0);
svbe : in std_logic_vector (15 downto 0);
svalf : in std_logic_vector (15 downto 0); -- NEW
svbef : in std_logic_vector (15 downto 0); -- NEW
svtop : in std_logic_vector (18 downto 0);

svbot : in std_logic_vector (18 downto 0);
extra_cn : in std_logic_vector(11 downto 0);
pival : in std_logic_vector (11 downto 0); -- UNDER TEST
-- Output ports
   dac0_ch0    : out std_logic_vector(11 downto 0);
   dac0_ch1    : out std_logic_vector(11 downto 0);
   dac0_ch2    : out std_logic_vector(11 downto 0);
   dac0_ch3    : out std_logic_vector(11 downto 0)
   );
end entity DACRouting33;
architecture rutas3 of DACRouting33 is
   constant zero    : std_logic_vector(11 downto 0)  := "000000000000" ;
begin
```

```
process(clk,RESN,SW)
begin
   if (RESN = '0') then
         dac0_ch0  <= sv1(17 downto 6);
         dac0_ch1  <= sv2(17 downto 6);
         dac0_ch2  <= sv3(17 downto 6);
         dac0_ch3  <= extra_cn;
   elsif rising_edge(clk)  then
case SW is
when "00000001" =>
         dac0_ch0  <= Preal_S(19 downto 8);
         dac0_ch1  <= Qreal_S(19 downto 8);
         dac0_ch2  <= Pdes_S (19 downto 8);
         dac0_ch3  <= extra_cn(11 downto 0);
when "00000010" =>
         dac0_ch0  <= sv1(17 downto 6);
         dac0_ch1  <= sv2(17 downto 6);

         dac0_ch2  <= sv3(17 downto 6);
         dac0_ch3  <= extra_cn(11 downto 0);
when "00000100" =>
         dac0_ch0  <= si1(14 downto 3);
         dac0_ch1  <= si2(14 downto 3);
         dac0_ch2  <= si3(14 downto 3);
         dac0_ch3  <= extra_cn(11 downto 0);
when "00001000" =>
         dac0_ch0  <= sval(15 downto 4);
         dac0_ch1  <= svbe(15 downto 4);
         dac0_ch2  <= pival(11 downto 0);
 dac0_ch3  <= extra_cn(11 downto 0);
when "00010000" =>
         dac0_ch0  <= sial(14 downto 3);
         dac0_ch1  <= sibe(14 downto 3);
         dac0_ch2  <= zero;
         dac0_ch3  <= extra_cn(11 downto 0);
when "00100000" =>
 dac0_ch0  <= svtop(18 downto 7);
         dac0_ch1  <= svbot(18 downto 7);
         dac0_ch2  <= zero;
         dac0_ch3  <= extra_cn(11 downto 0);
when "01000000" =>
 dac0_ch0  <= sval(15 downto 4);
         dac0_ch1  <= svbe(15 downto 4);--svbe(17 downto 6);

         dac0_ch2  <= svalf(15 downto 4);
 dac0_ch3  <= svbef(15 downto 4);
when "10000000" =>
         dac0_ch0  <= sial(14 downto 3);
         dac0_ch1  <= sibe(14 downto 3);
         dac0_ch2  <= sialf(14 downto 3);
         dac0_ch3  <= sibef(14 downto 3);
when others =>
 dac0_ch0  <= Preal_S(19 downto 8);
         dac0_ch1  <= Qreal_S(19 downto 8);
         dac0_ch2  <= Pdes_S (19 downto 8);
         dac0_ch3  <= Qdes_S (19 downto 8);
end case;
end if;
end process;
end rutas3;
```

The purpose of the block `DACRouting33` is to route any set of four signals to the DAC in 12-bits each. This explains why most signals of interest input the block. As often the signal expected value is known, the 12 most dominant bits are selected to remove the extra bits from the output. The signal is selected with a set

of switches `SW` in the FPGA board, which enter the code `DACRouting33.vhd`. To improve reading of the code, Table 5.5 links the exact switch in the FPGA board (see Figure 5.5) to both, the signal produced by the software and the corresponding theoretical signal.

TABLE 5.5: Assigned inputs in FPGA board for signal display with associated theoretical signal. Signals marked as (*) refer to the equivalent follower value from the PLL (without experimental switching effects).

Input	Experimental signals	Theoretical signals
SW[0]	reset	Run
SW[1]	s1,s2,s3,s4	P, Q, P_{des}, Q_{des}
SW[2]	VA_P,VB_P,VC_P,N2019	$v_{ga}, v_{gb}, v_{gc}, n(\theta)$
SW[3]	iA_P,iB_P,iC_P,N2019	$i_a, i_b, i_c, n(\theta)$
SW[4]	VALP_P,VBET_P, thth, N2019	$v_{g\alpha}, v_{g\beta}, \theta, n(\theta)$
SW[5]	ialphaP,ibetaP, thth, N2019	$i_{g\alpha}, i_{g\beta}, \theta, n(\theta)$
SW[6]	VTOP_P,VBOT_P, 0, N2019	$v_{C1}, v_{C2}, 0, n(\theta)$
SW[7]	VALP_P,VBET_P,vaFP,vvFP	$v_{g\alpha}, v_{g\beta}, v^*_{g\alpha}, v^*_{g\beta}$
SW[8]	ialphaP,ibetaP,iaFP,ibFP	$i_\alpha, i_\beta, i^*_\alpha, i^*_\beta$

On the left hand side of Figure 5.27, the signal shifting takes place. The shifting of a signal to positive values is usually done by taking the maximum possible value of each signal and adding it to the input. For example, the code `ShiftToPositive_I.vhd` takes all the current signals, defines a constant with the maximum 'negative' value (8,191[mV]) and adds this number to all the corresponding current outputs. `ShiftToPositive_V.vhd` and `ShiftToPositive_PQ.vhd` are coded in the same fashion.

```
-- ShiftToPositive_V.vhd --
library IEEE;

use IEEE.STD_LOGIC_1164.all;
use ieee.numeric_std.all;

entity ShiftToPositive_V is
port
(
-- Input ports
reset :  in std_logic;
clk :  in std_logic;
mva :  in integer range -131071 to 131071 :=0;--2^17 std_logic_vector(17 downto 0);
mvb : in integer range -131071 to 131071 :=0;
mvc : in integer range -131071 to 131071 :=0;
mvalph : in integer range -31071 to 31071 :=0;
```

```
mvbet : in integer range -31071 to 31071 :=0;
valf : in integer range -31071 to 31071 :=0;
vbef : in integer range -31071 to 31071 :=0;
mvvtop : in integer range 0 to 65535 :=0;
mvvbot : in integer range 0 to 65535 :=0;
nt_pn : in std_logic_vector(6 downto 0);
-- Output ports
va_p : out integer range 0 to 270000 :=0;
vb_p : out integer range 0 to 270000 :=0;
vc_p : out integer range 0 to 270000 :=0;
valp_p : out integer range 0 to 62150 :=0;
vbet_p : out integer range 0 to 62150 :=0;
valf_p : out integer range 0 to 62150 :=0;
vbef_p : out integer range 0 to 62150 :=0;
vtop_p : out integer range 0 to 270000 :=0;
vbot_p : out integer range 0 to 270000 :=0;

nt_p      :  out integer range 0 to 63 := 0
);
end ShiftToPositive_V;
architecture first of ShiftToPositive_V is
signal nttemp : integer range 0 to 62150 := 0;
   constant zero :integer := 0;
constant con1 :integer := 131071;
constant con2 :integer := 31071;
constant miconst : integer := 5 ;
constant lpf: integer:= 31071;
begin
nttemp <= to_integer(signed(nt_pn));
process(clk,reset)
begin
if (reset = '0') then
va_p   <= zero;
vb_p   <= zero;
vc_p   <= zero;
elsif(rising_edge(clk)) then
va_p   <=  mva+con1;
vb_p   <=  mvb+con1;
vc_p   <=  mvc+con1;
valp_p <=  mvalph+con2;
vbet_p <=  mvbet+con2;
valf_p <=  valf + con2;
vbef_p <=  vbef + con2;
nt_p <= to_integer(unsigned(nt_pn)); -- variable UNDER TEST
vtop_p <=  mvvtop*miconst; -- + const2;
vbot_p <=  mvvbot*miconst; -- + const2;
end if;

end process;
end first;
```

```
-- ShiftToPositive_PQ.vhd --
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.numeric_std.all;
entity ShiftToPositive_PQ is
port
(
-- Input ports
reset :  in std_logic;
clk : in std_logic;
Preal : in integer range -524287 to 524287 :=0;
Qreal : in integer range -524287 to 524287 :=0;
Pdes : in integer range 0 to 524287 :=0;
Qdes : in integer range 0 to 524287 :=0;
-- Output ports
Preal_S : out integer range 0 to 1000000:=0;
```

```
Qreal_S : out integer range 0 to 1000000:=0;
Pdes_S : out integer range 0 to 1000000:=0;
Qdes_S : out integer range 0 to 1000000:=0
);
end ShiftToPositive_PQ;
architecture first of ShiftToPositive_PQ is
   constant zero :integer := 0;
constant con1 :integer := 100000;
begin
process(clk,reset)
begin
if (reset = '0') then
Preal_S   <= zero;
Qreal_S   <= zero;
Pdes_S   <= zero;
Qdes_S   <= zero;
elsif(rising_edge(clk)) then
Preal_S   <= Preal+con1;
Qreal_S   <= Qreal+con1;
Pdes_S   <= Pdes+con1;
Qdes_S   <= Qdes+con1;
end if;
end process;
end first;
```

```
-- ShiftToPositive_I.vhd
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.numeric_std.all;
entity ShiftToPositive_I is
port
(
-- Input ports
reset :   in std_logic;
clk : in std_logic;
mia : in integer range -8191 to 8191 :=0; -- I max value 16777216
mib : in integer range -8191 to 8191 :=0;
mic : in integer range -8191 to 8191 :=0;
mial : in integer range -8191 to 8191 :=0;
mibe : in integer range -8191 to 8191 :=0;
ialf : in integer range -8191 to 8191 :=0;
ibef : in integer range -8191 to 8191 :=0;
-- Output ports
       ia_p : out integer range 0 to 17000 :=0; -- I max value 16777216
ib_p : out integer range 0 to 17000 :=0;
ic_p : out integer range 0 to 17000 :=0;
ialpP : out integer range 0 to 17000 :=0;
ibetP : out integer range 0 to 17000 :=0;
ialf_P : out integer range 0 to 17000 :=0;
ibef_P : out integer range 0 to 17000 :=0
);
end ShiftToPositive_I;
architecture first of ShiftToPositive_I is
   constant zero :integer := 0;
constant con1 :integer := 8191;
begin
process(clk,reset)
begin
if (reset = '0') then
ia_p   <= zero;
ib_p   <= zero;
ic_p   <= zero;
elsif(rising_edge(clk)) then
ia_p   <=  mia+con1;
ib_p   <=  mib+con1;
ic_p   <=  mic+con1;
ialpP  <= mial+con1;
```

```
ibetP  <= mibe+con1;
ialf_P <= ialf+con1;
ibef_P <= ibef+con1;
end if;
end process;
end first;
```

5.3 Calibration and Sensors Datapath

The resolution of the data is determined by the sensors' accuracy, filtering, speed of the FPGA, and ultimately is defined by the algorithm used. However, too much resolution is not very useful if the acquired data is noisy even after filtering. Figure 5.28 presents the technicalities of the required signals, from measurement of v_g and i to acquisition of the representative units of the real measurement.

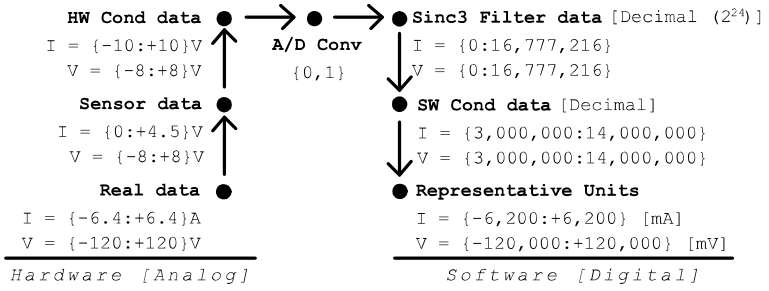


FIGURE 5.28: Data processing and conditioning

On the hardware side, the real currents and voltages need to be measured and the *LEM* sensors previously described are configured to be capable of a measuring range of ±6.4 [A] and ±120 [V] respectively, according to datasheets. The acquisition of grid voltage measurements was set up to read values higher than ± 100 [V], thinking of a future scenario with larger power control, as this would reduce the amount of modifications required in the prototype.

Next on the hardware side, the data acquired directly from the sensors is provided in [V] and have a proportional range of 0 to +4.5 [V] for the currents and −8 to +8 [V] for the voltages. As the current sensors need to be conditioned, the circuitry described with Figure 5.12 is implemented and the proportional ranges

of currents and voltages obtained is ±10 [V] and ±8 [V], respectively. These analog signals then pass through an ADC to be transformed into digital signals with the parameters specified earlier.

On the software side, the digital signals are received in the FPGA algorithm, which will process them according to the code implemented. In this case, the 'raw' digital signals pass through a $Sinc^3$ filter first. This digital filter outputs a signal with a resolution of 24 bits (or 16,777,216 [units]). The decimal values read from the filter usually range from 3,000,000 to 14,000,000 [units]. These digital values are proportional to the voltages and currents coming from the sensors and enter a signal conditioning block for calibration test. Then the real representative units are obtained with range of $\pm 120,000$ and $\pm 6,200$, representing the voltage and current values in [mV] and [mA], respectively. With i_{abc} and v_{abc} already obtained, the transformation and the rest of the processing described in the previous section are carried out.

The calibration of the acquired signals in the FPGA required many iterations of the process in Figure 5.28. The algorithm `disp_switcher` acquired data from the sensors and displayed the hexadecimal values in the 7-segment displays of the FPGA board.

For the calibration of the grid voltage sensors, measurements were taken in the range of -120 to +120 [V] in steps of 10 [V] for up to $\pm 60[V]$, outside this range the measurements were approximated. For the calibration of the current sensors, measurements were taken in the range of -6.1 to +6.1 [V] in increments of 1 [A] using 10 [Ω] resistors, applying the general Ohm Law $I = V/R$. For the calibration of the capacitor voltage sensors from the inverter board, measurements were taken in the range of 0 to 40 [V] in steps of 5 [V]; from $-DC$ to NP for the bottom capacitor and from NP to $+DC$ for the top capacitor.

Two hexadecimal measurements per increment were manually recorded in a leg basis for all current and voltage sensors. Equally for each capacitor voltage sensor. These values were manually input in a spreadsheet, where the two hexadecimal values were converted to decimal values and a mean value was calculated. This mean decimal value is the one used for adjusting the nearly-linear slope of scaling. The calibration was done following the curve equation $y = ax + b$ in such a way that we make $x = 1$, due to integer-values-only restriction.

The general formulas used for the current sensors are the following:

$$y_d = 1000 I_{real}, \tag{5.13}$$

$$w = x - n_{zero}, \tag{5.14}$$

$$k = mean(w/y_d,) \tag{5.15}$$

$$y = \frac{w}{k}, \tag{5.16}$$

where I_{real} is the real measured analog-value, y_d is the desired read digital-value in [m-] units, x is the 'raw' digital input from any sensor signal, n_{zero} is the average digital constant at which $I_{real} = 0$, w is the read shifted/centered digital-value, k is average of all proportions between w and y_d, and y is the actual read digital-value in [m-] units. All in all, this is a slope fitting process. Equations (5.13)-(5.16) are valid in a per-signal basis: 3 (v_g) + 3 (i) + 2 $(v_{C1,2})$ = 8 sensor signals.

The first test consisted in visualising the grid voltages $v_{\alpha\beta}$ and the twelve regions of $n(\theta)$. The results are shown in Figures 5.29 and 5.30. The first shows more than two sine wave cycles, while the second shows one cycle. v_α is shown in yellow and v_β is shown in green. These signals are the result of acquiring v_{abc} from the sensors, through the ADC filter and after the aforementioned $\alpha\beta$- transformation used. The blue steps correspond to $n(\theta)$ from region 1 (bottom) to region 12 (top), which iterate every time a sine cycle is completed.

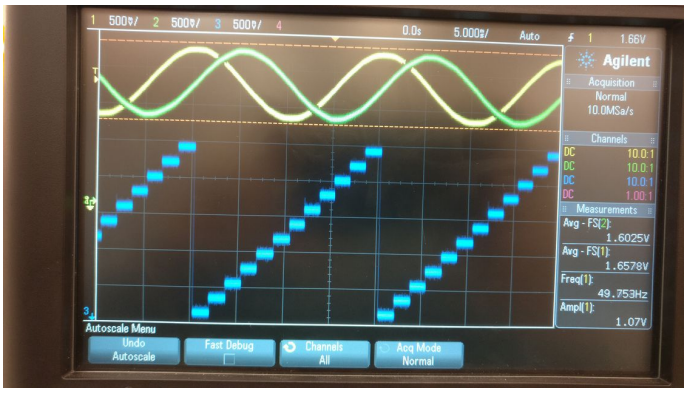


FIGURE 5.29: Picture of v_α in yellow, v_β in green and $n(\theta)$ in blue.

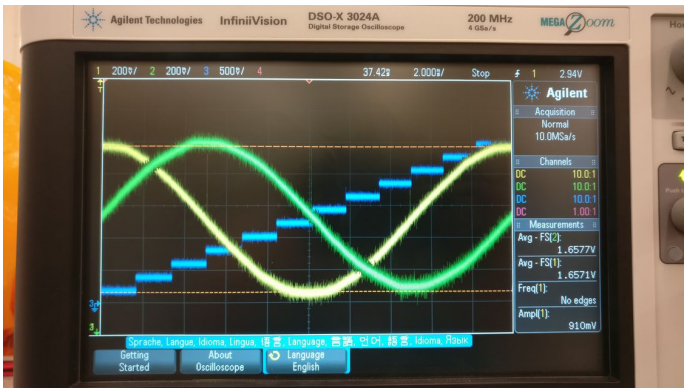


FIGURE 5.30: Zoom picture of v_α in yellow, v_β in green and $n(\theta)$ in blue.

The oscilloscope used througout this work is an *Agilent InfiniiVision DSO-X 3024A* with four analog channels, 200 [MHz] bandwidth and up to 4 [GSa/s]. These previous figures show photographs of this oscilloscope. Eventually, the data obtained from the oscilloscope was no longer recorded in photographs but rather in datapoints exported from the oscilloscope to a USB memory in a *.csv* file. The recorded datapoints from the osciloscope are usually exported with resolution of a data point every 100 $[\mu s]$, depending on the 'zoom' selected. Then, this file was processed in a matlab script $(.m)$ to properly format the data, as will be shown next.

5.4 DPC Results

Simulations in Chapter 4 feature a switching frequency of 1 [MHz], however this is hardly a switching speed of a commercial transistor. The switching frequency in the prototype is restricted by the speed of the FPGA board, but mostly by the transistor switching frequency, which is 20 [kHz].

The development of a prototype brings many problems usually taken for granted in simulations. Some of the problems are:

- The grid voltage has some noise and a transformer is required.
- An inverter operating with a non-isolated DC voltage source and no transformer can inject common-mode current to the grid.
- Transistors have limited switching frequency.
- Measurements of voltages and currents as well as its ADC conversion inject noise.
- Measurements of capacitor voltages have low resolution since the internal voltage sensors are used.
- DC Voltage Supply is unipolar.
- FPGA processes the algorithm at up to 20 [MHz] but outputs transistor signals at 20 [kHz].

It is expected that, due to the limited switching frequency of the real transistors (compared to the ideal ones in simulations) some more ripple appears in the results. Considering the hardware available in the laboratory and a considerable design of a NPC inverter, the parameters shown in Table 5.6 have been deduced. Although, this inverter is capable of much higher power, it is used at the values shown. These are valid for all the results presented in this Chapter.

TABLE 5.6: Set of values for DPC.

Parameter	Value	Unit
Q_{des}	0	[W]
C_1	4.08	[mF]
C_2	4.08	[mF]
v_{DC}	60	[V]
R_x	50	[mΩ]
L_x	6	[mH]
v_g	20	$[V_{RMS}]$
f_s	20	[kHz]
f_g	50	[Hz]

5.4.1 Early Results

The early results are shown next, with selected values of $P = 150,000$ [mW], $\Delta P = 5,000$ [mW], $\Delta Q = 5,000$ [mW], and $\Delta C = 20,000$ [mV]. Most results are presented in a 50 [ms] window and none of them capture the initialisation of the algorithm run, it only captures the data at some point in time t during the run.

Active power P and reactive power Q are shown on top and bottom of Figure 5.31, respectively. It is observed once the active power reaches or exceeds $P_{des} + \Delta P$ it goes down to zero (or lower) shortly after. Then P moves up again until it reaches P_{des} values and it repeats the pattern. The reactive power mostly stays in zero, with peaks up to 120 [W]. Figure 5.32 show the same variables in a 10 [ms] window. Active powers 'climbs' to the desired value and once it is reached, it suddenly drops to zero, and performs the same iteratively. A closer view of the reactive power shows it may not keep a zero value, but it follows it and could be its average value.

The grid voltages v_{abc} and currents i_{abc} are shown in Figures 5.33 and 5.34, respectively. Some noise is observed in v_{abc}, but is consistent with a three-phase grid showing a sine pattern and even phase difference between voltages. The currents i_{abc} present a sawtooth pattern, which could explain the performance of the power. i_{abc} signals are synchronised with the voltage readings in a sine shape manner.

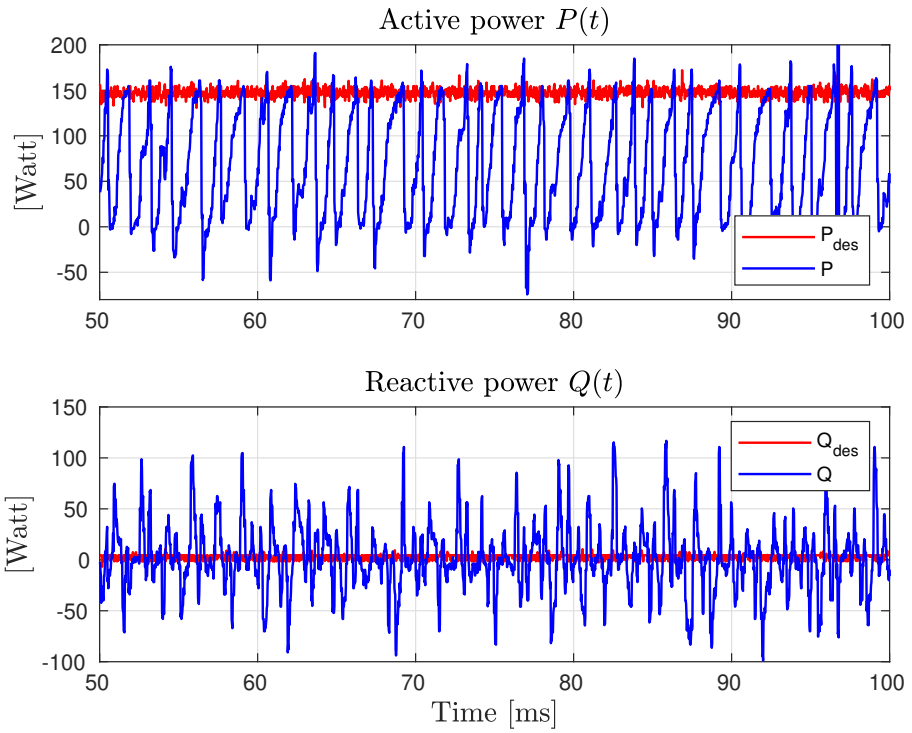


FIGURE 5.31: Experiment. Active and reactive power

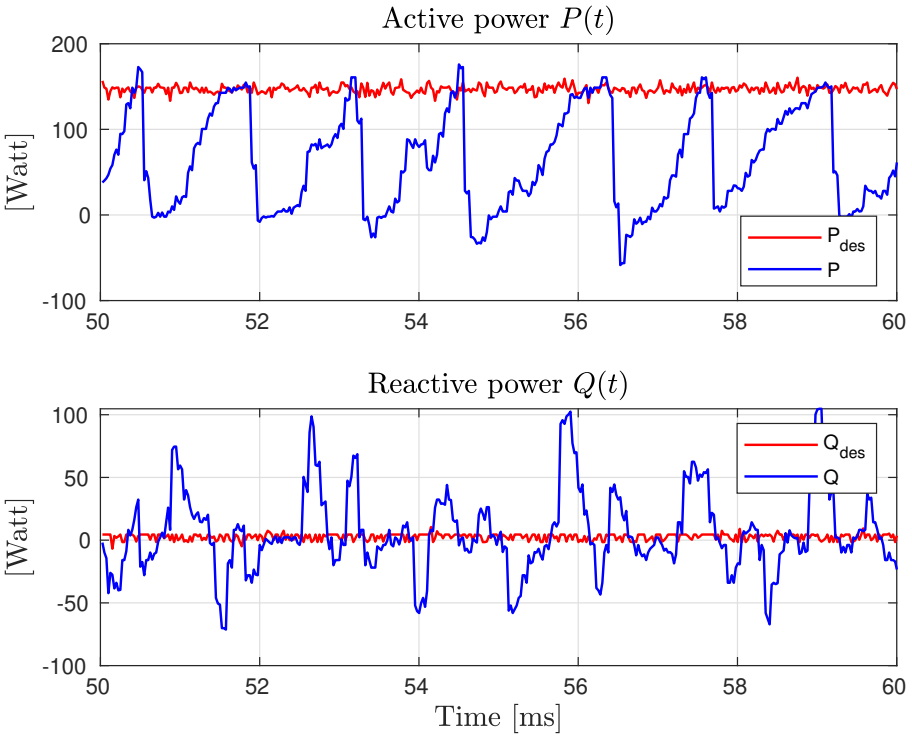


FIGURE 5.32: Experiment. Active and reactive power (zoom)

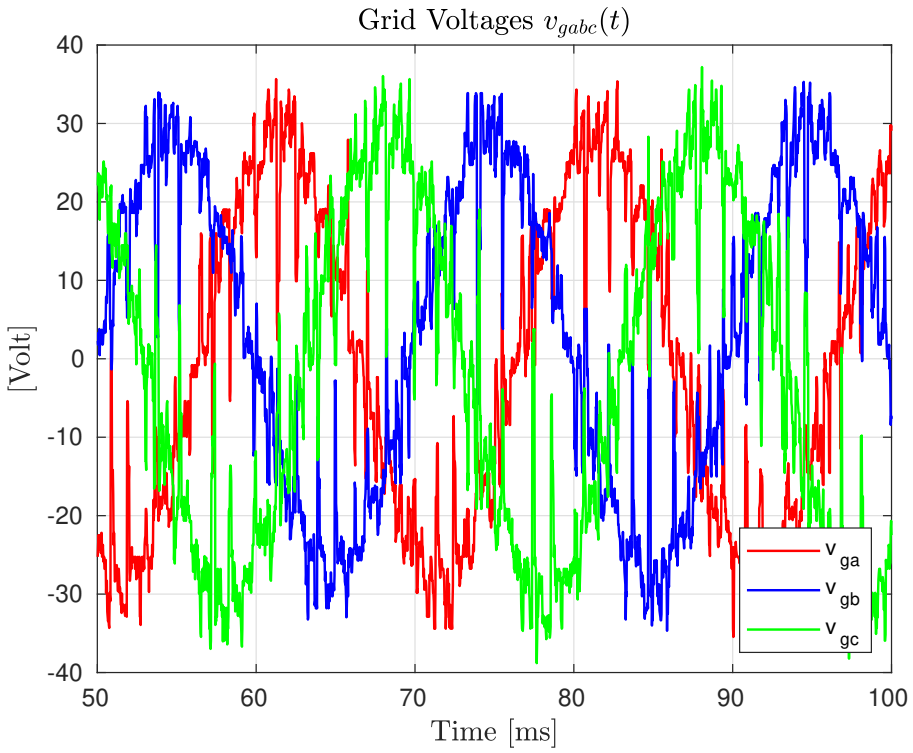


FIGURE 5.33: Experiment. Grid voltages

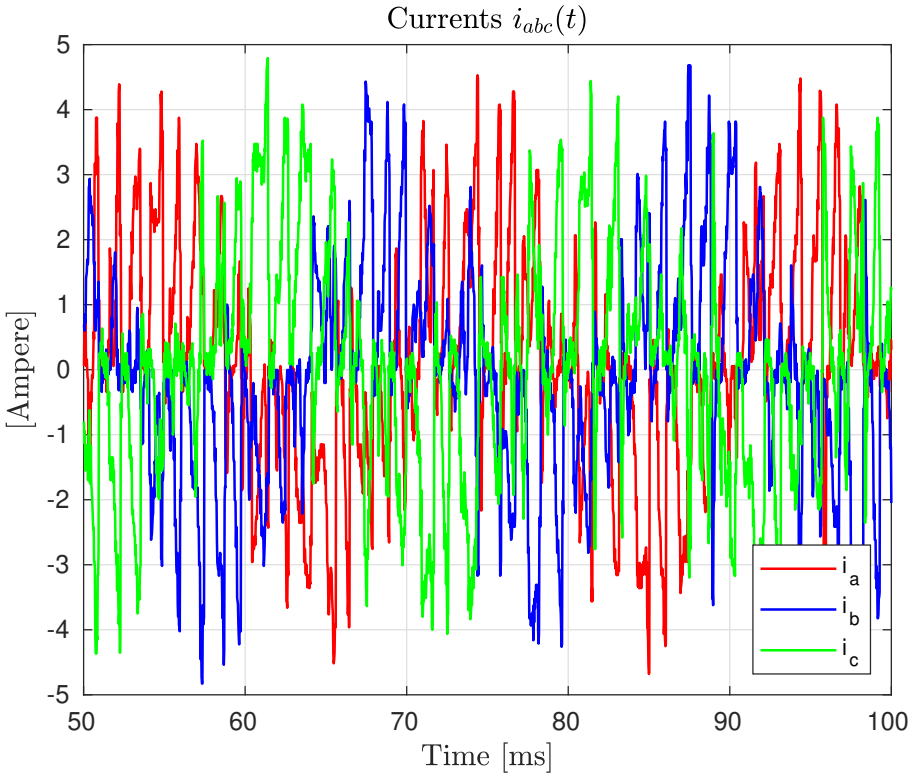


FIGURE 5.34: Experiment. Currents

Figure 5.35 demonstrates voltages after transformation $v_{\alpha\beta}$, which are aligned to a region value $n(\theta)$ ('reversed' from the simulation pattern, but completely valid). $v_{\alpha\beta}$ shows some noise, particularly during the peak values, which is caused by the switching action of the inverter. $n(\theta)$ is shown in black, where the bottom value corresponds to the lowest value '1', increasing linearly and in integers to the top value '12'. It was useful to ensure the regions were in agreement with the phase of voltages $v_{\alpha\beta}$ or otherwise with the position of the grid voltage vector v_g.

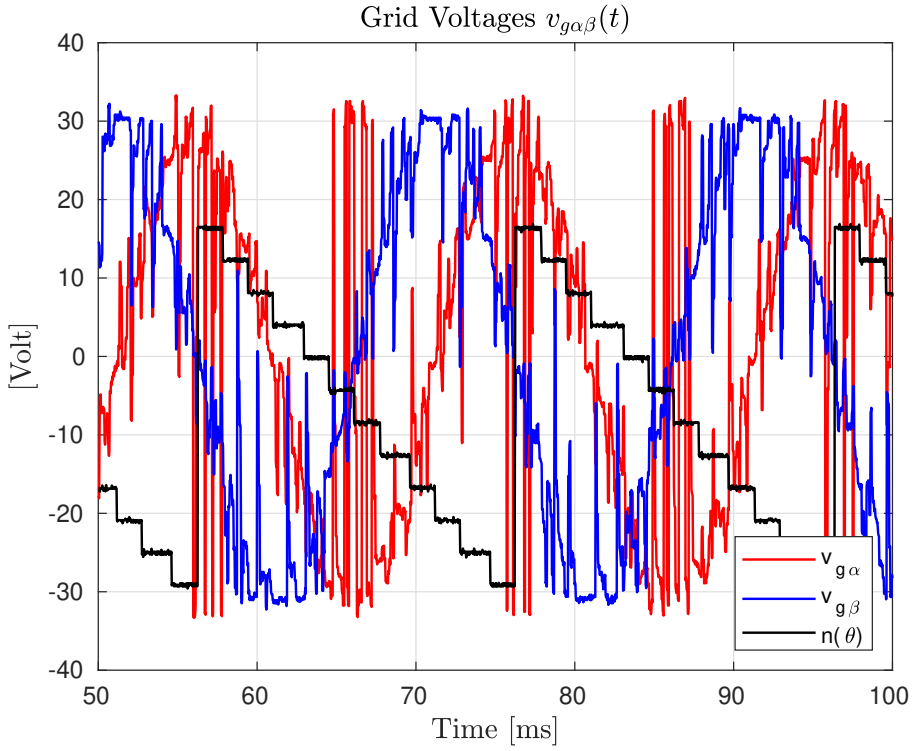


FIGURE 5.35: Experiment. Grid voltages in $\alpha - \beta$ frame.

Figure 5.36 shows the currents in a similar fashion, where i_α is leading i_β. It presents a clearer sawtooth pattern of the currents from left to right, with peak values of about 4 [A]. $n(\theta)$ is again shown with descending values from '12' to '1' for the same purposes. The capacitor voltages show a very steady response in Figure 5.37, which was expected due to the low resolution sensors in the inverter board. The top capacitor v_{C1} is 10 [V] higher than the bottom capacitor v_{C2}. The black line is a zero value from the FPGA board for reference only.

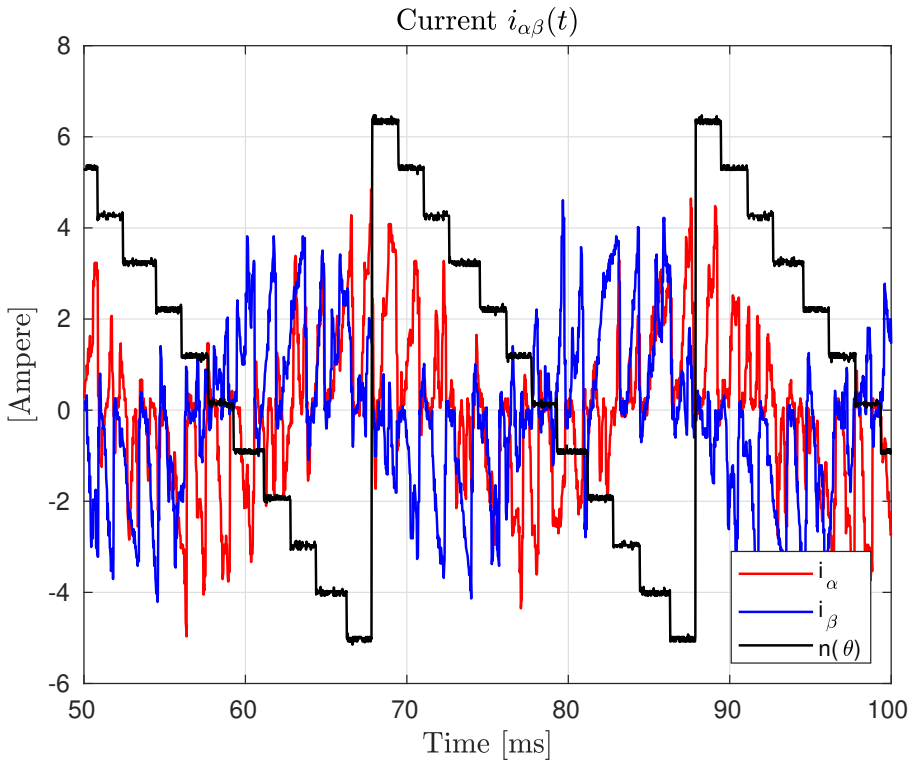


FIGURE 5.36: Experiment. Currents in $\alpha - \beta$ frame.

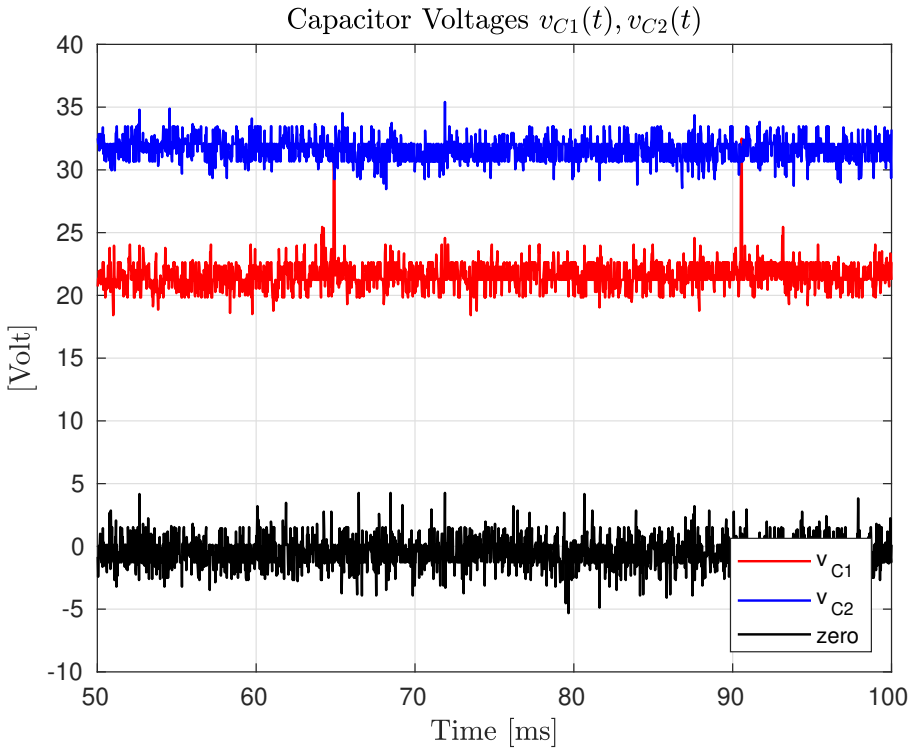


FIGURE 5.37: Experiment. Capacitor voltages.

Figure 5.38 shows $v_{g\alpha}$ vs $v_{g\alpha fol}$ and $v_{g\beta}$ vs $v_{g\beta fol}$. Values $v_{g\alpha fol}$ and $v_{g\beta fol}$ correspond to `vaF` and `vbF` respectively, as described in subsection 5.2.6. This is to show the PLL 'follower' voltages are successfully following the actual grid voltages $v_{g\alpha\beta}$ through a LUT. Consequently, power calculations in the algorithm made from these 'follower' voltages (and actual currents) are performed with reduced ripple. $v_{g\alpha fol}$ and $v_{g\beta fol}$ show a very subtle delay, but this has been compensated with $n(\theta)$ in the LUT by manual adjustment using the oscilloscope and calibrating.

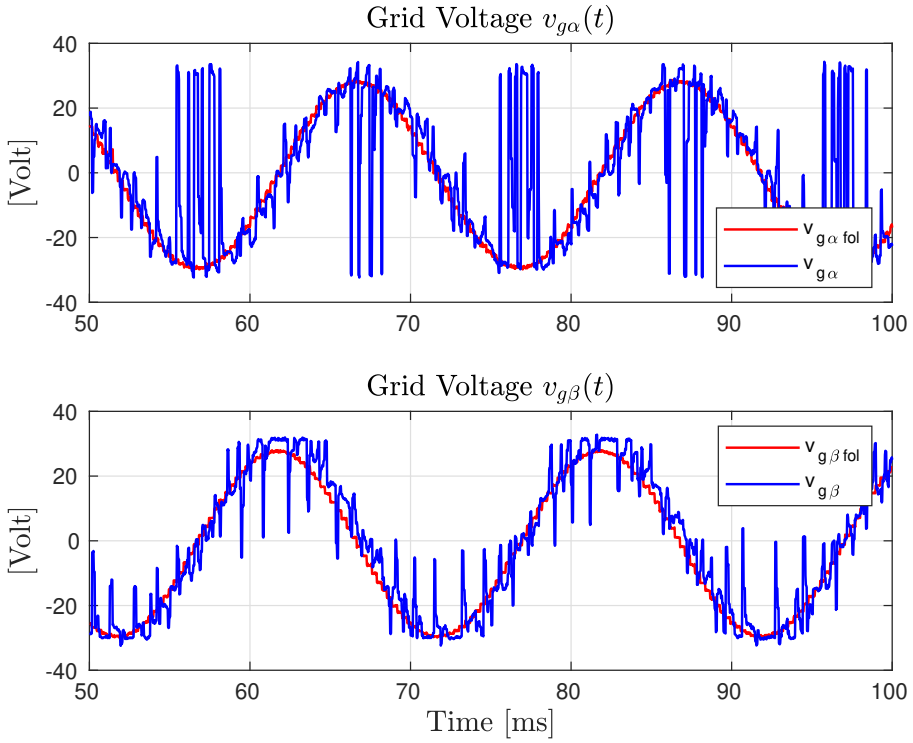


FIGURE 5.38: Experiment. Grid voltages in $\alpha - \beta$ frame. $v_{g\alpha\beta fol}$ is the "follower" and comes from a LUT in the PLL.

Figure 5.39 shows i_α vs $v_{\alpha fol}$ and i_β vs $i_{\beta fol}$. Values $i_{\alpha fol}$ and $i_{\beta fol}$ correspond to `iaF` and `ibF` respectively, as coded in subsection 5.2.6. Although $i_{\alpha fol}$ and $v_{\beta fol}$ are not used for the DPC algorithm, they show an approximation of the expected current sine wave at a given time t. The current $i_{\alpha\beta}$ performs with an expected sine wave pattern, but again, its sawtooth pattern is present.

Overall, tt was expected that transistor switching could cause some ripple, but the ripple presented in P is too large. This is caused by the current high ripple,

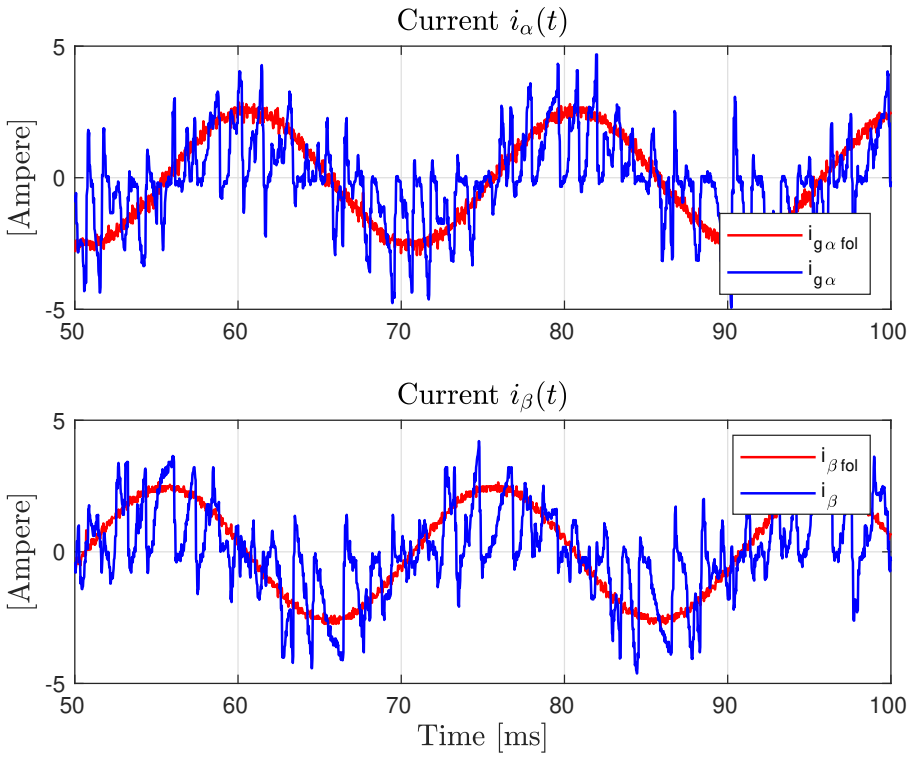


FIGURE 5.39: Experiment. Currents in $\alpha - \beta$ frame. $i_{\alpha\beta fol}$ is the "follower" and comes from a LUT in the PLL.

constantly going to back to zero. The ripple depends on the following: transistor switching frequency, the data input frequency from the sensors through the ADC, the hysteresis values and the vector table.

The transistor switching is kept to the maximum of 20 [kHz] in the prototype. Changing the hysteresis values didn't seem to cause much effect, as the most undesired performance is the sharp decrease of P to zero. Only when $+\Delta P$ was kept significantly large and $-\Delta P$ was kept small, the P ripple could offset, resulting in a more averaged P_{des} value. However, the large jump from P_{des} to the bottom value persisted.

Leaving the transistor frequency (already at its maximum) and the hysteresis values behind, improvements on the PLL algorithm were made. After this, it was decided to leave behind the digital filter (ADC) as well, as it was supposedly working sufficiently fast. Then, attention was put in the vector table along with

the algorithm and hardware that trigger the transistors. There was a constant iteration of improvements in the code to find the fault, and the vector table was analysed several times. Consistent failure to improve the results lead to a new vector table, which will be addressed later.

However, even with the new vector table, the results did not improve significantly. Eventually, a low frequency was spotted in the digital filter (ADC) of the *Quartus* project. It was thought the sampling frequency of the digital filter was 200 [kHz], which had been missread. The values of the digital filter were `CLK_OUT1 = 10 [MHz]` and `DCLK = 40 [kHz]`. What's more, the word lenght of the reading was 25 bits, which requires several periods (clock pulses) in order to obtain a single reading (see subsection 5.2.3 for reference). Furthermore, after the signal is acquired, there is some more time left to process the data in the algorithm before there is a corresponding output. Consequently, further simulation tests were required.

5.4.2 Comparing with Simulations

The *Simulink* simulation shown in Figure 4.24 had been modified to be as accurate as possible to the prototype regarding all the signal processing and constraints described so far in this Chapter. This included a 20 [kHz] block added in the gate of the *NPC* inverter block. Following the development in the previous paragraph, simulation files were re-explored and opposite from the experiments, the acquisition of i_{abc} and v_{abc} was found to be simulated with a much higher speed.

The focus this time was on testing what impact the speed in signal acquisition had on the performance of the algorithm. A variable speed blockset was deviced for the voltage and current measurements, between the $\alpha\beta$ transformation and the signal tags (arrow shaped blocks in Figure 4.24). These sampled signals are then input to the power calculation block.

The simulation was run with a fixed step time of 100 [ns], and it was processed in [mili-] units, just as in the code. The following values were kept: $P = 150,000$ [mW], $\Delta P = 5,000$ [mW], $\Delta Q = 5,000$ [mW], and $\Delta C = 20,000$ [mV].

Firstly, a frequency of $f_{ff} = 40$ [kHz] was tested, and secondly, a frequency of $f_f = 3,125$ [Hz] was. Figure 5.40 shows the desired active power $P_{des} = 150[W]$, and the active and reactive powers shown are functions of nT, where n is the sample number growing to infinity (as long as the experiment keeps running) in integer numbers (or multiples of a period T). $P(nT_{ff})$ is the active power with a sampling period $T_{ff} = 1/f_{ff}$ (green), while $P(nT_f)$ is the active power with a sampling period of $T_f = 1/f_f$ (blue). Similarly for the reactive power Q at the bottom.

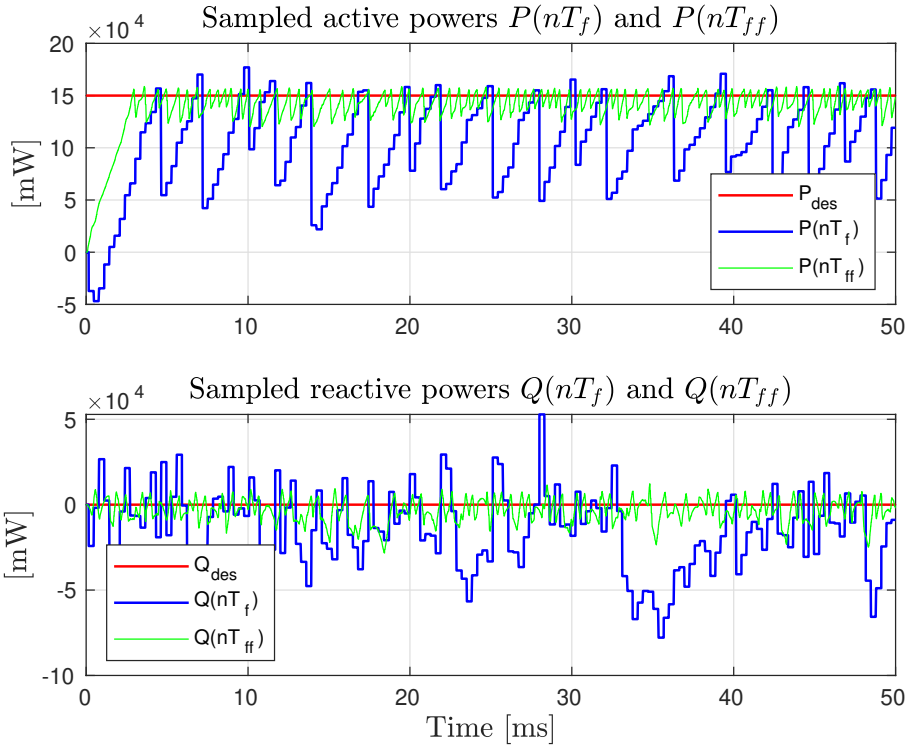


FIGURE 5.40: Simulation. Active and reactive power with $T_f = 1/3,125$[Hz] and $T_{ff} = 1/40$[kHz]

$P(n_{T_{ff}})$ and $Q(n_{T_{ff}})$ show a relatively small ripple, within few thousands of miliwatts. However, when the acquisition of voltage and current signals are reduced from 1/40,000 [s] to 1/3,125 [s], the simulation results in active power $P(nT_f)$ with

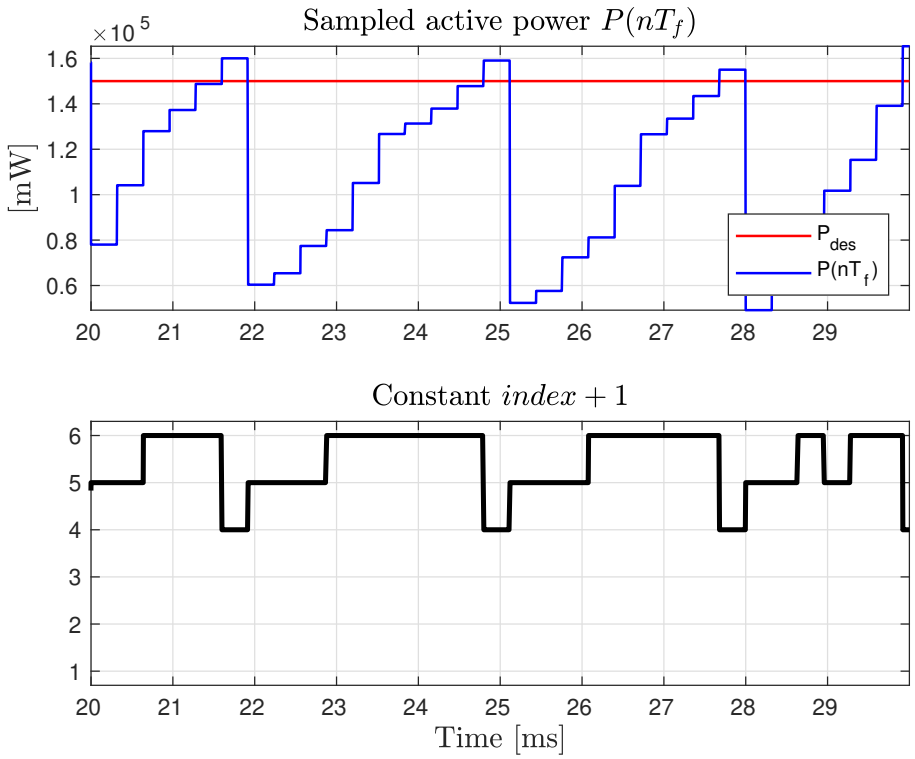


FIGURE 5.41: Simulation. Active power and index reference with $T_f = 1/3,125$[Hz] (zoom)

a significantly high amplitude of a sawtooth pattern, reaching P_{des} gradually to be drastically reduced to a near zero value. Similarly for the reactive power $Q(nT_f)$. $P(nT_{ff})$ and $Q(nT_{ff})$ show a ripple larger than ten thousands of miliwatts.

On top of Figure 5.41 a closer time window of $P(n_{T_f})$ is presented, where the sampling is clearly observed. Below, the corresponding $index+1$ value from Table 4.5 is shown. This value shows a change from the bottom two-rows of the table as P grows, which implies $P_d = 1$ (LVs and MVs), to the above row which implies $P_d = 0$. This corresponds to P reaching $P_{des} + \Delta P$ and chosing a vector allowing a decrease in active power.

Figure 5.42 shows the currents i_{abc} in simulation sample time on top, the same currents sampled at f_f in the middle, and the transformed currents $i_{\alpha\beta}$ also sampled at f_f on the bottom. There is a visible difference between the top graph and the two other sampled graphs. On the top, the decrease to zero of the sawtooth-like

patterns is sharp, but a slight slope is still visible, whereas the rest of the graphs sharply decrease to zero with no visible slope.

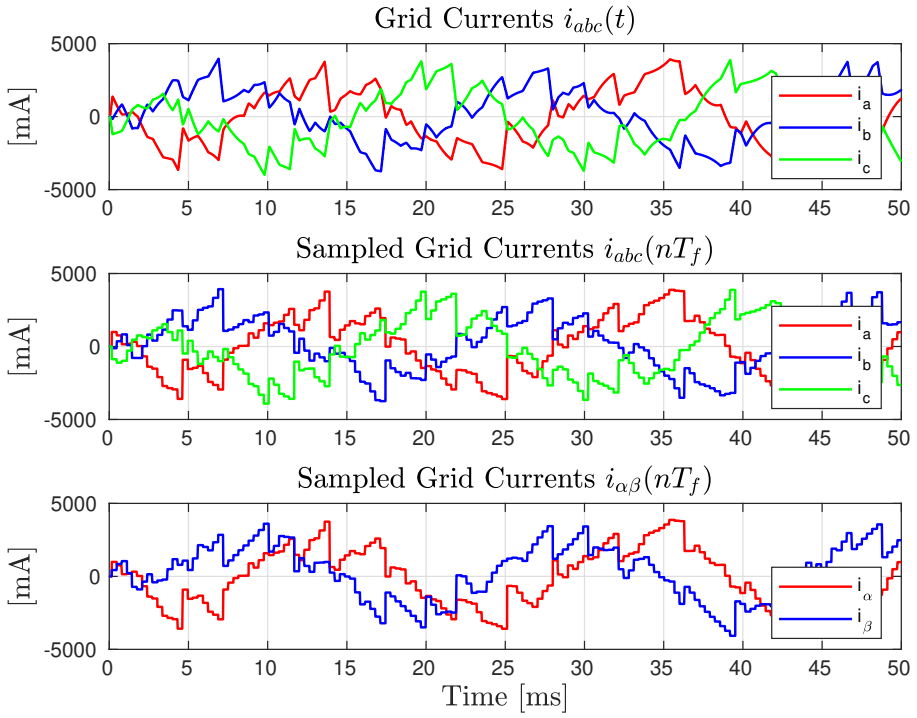


FIGURE 5.42: Simulation. Currents with simulation time t and $T_f = 1/3,125$[Hz]

Figure 5.43 show voltages v_{abc} with injected noise on top at simulation time. In the middle, the corresponding sampled voltages are shown with a clear effect on the sampling. At the bottom, the transformed voltages $v_{\alpha\beta}$ are shown with the same sampling T_f.

This subsection shows how much the DPC technique relies in the small changes of i during operation. These changes directly depend on the transistor switching, which naturally depend on the position of the grid voltage vector. Therefore, if the grid vector position is not updated fast enough or the current reading is not fast enough to detect these changes, the results will be expected to have a similar performance to the above figures. Moreover, provided the grid voltages are accurately tracked by the PLL, the power calculation (and therefore the ripple) will mostly depend on fast and accurate readings of the currents i_{abc}.

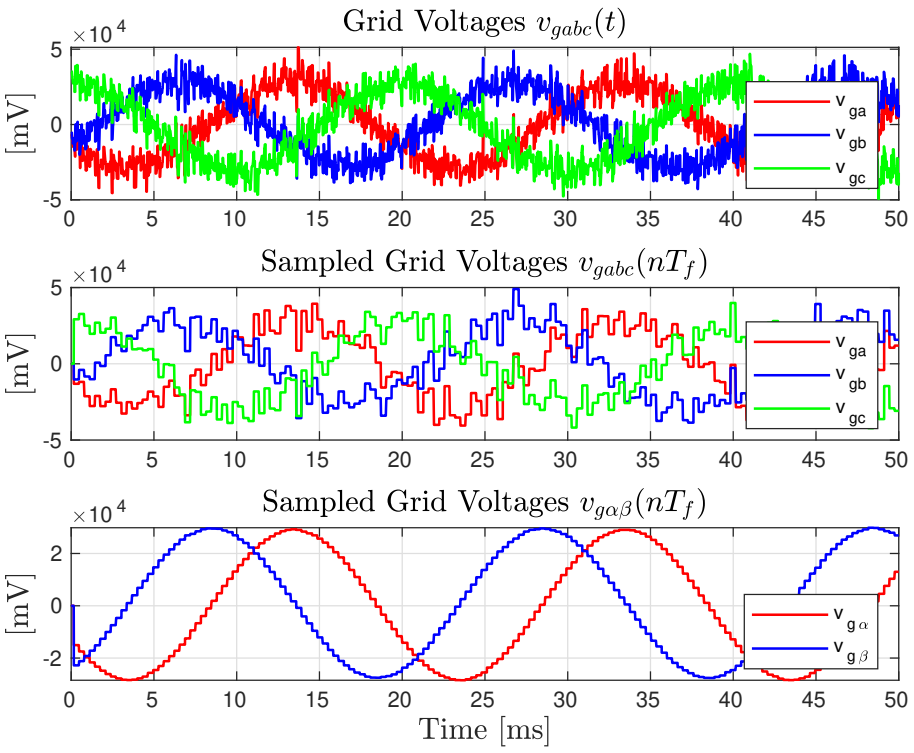


FIGURE 5.43: Simulation. Voltages with simulation time t and $T_f = 1/3,125$[Hz]

In a nutshell, it turned out a very slow signal acquisition does have an impact on the capability of the algorithm to operate properly, as shown in the results. Although this discovery made sense, it was being looked with skepticism (to put it brightly).

5.4.3 Faster Readings and New 'Soft' Vector Table

The next step was to increase the speed of signal acquisition in the algorithm. However, because the solution for this problem is far from simply changing the frequency in the algorithm of the digital filter, the first faster frequency found which provided meaningful values, at `DCLK = 80 [kHz]`, was used. The modulation frequency of the filter `CLK_OUT1 = 10 [MHz]` was kept.

Therefore, after an increase of double the frequency, a calibration stage followed: recording all the sensor readings for i_{abc}, v_{gabc} and v_{C1}, v_{C2}, and obtaining a slope

curve for every voltage and current sensor (repeating steps in subsection 5.1.3). The PLL turned out with some malfunction (worked part of the time) and the following plots were achieved. It is worth noting that even a small change in the algorithm required a long time due to three factors: the low-level language, the compilation time (up to 12 minutes) and the user inexperience. Besides, it was difficult to capture the start of the algorithm run in the prototype as it runs in real time.

The parameters of the following experimental results are again kept the same: $P = 150,000$ [mW], $\Delta P = 5,000$ [mW], $\Delta Q = 5,000$ [mW], and $\Delta C = 20,000$ [mV]. First, results for the 'hard' vector Table 4.5 are presented. Second, a new 'soft' vector table is introduced and the resulting performance follows. All the following figures are experimental.

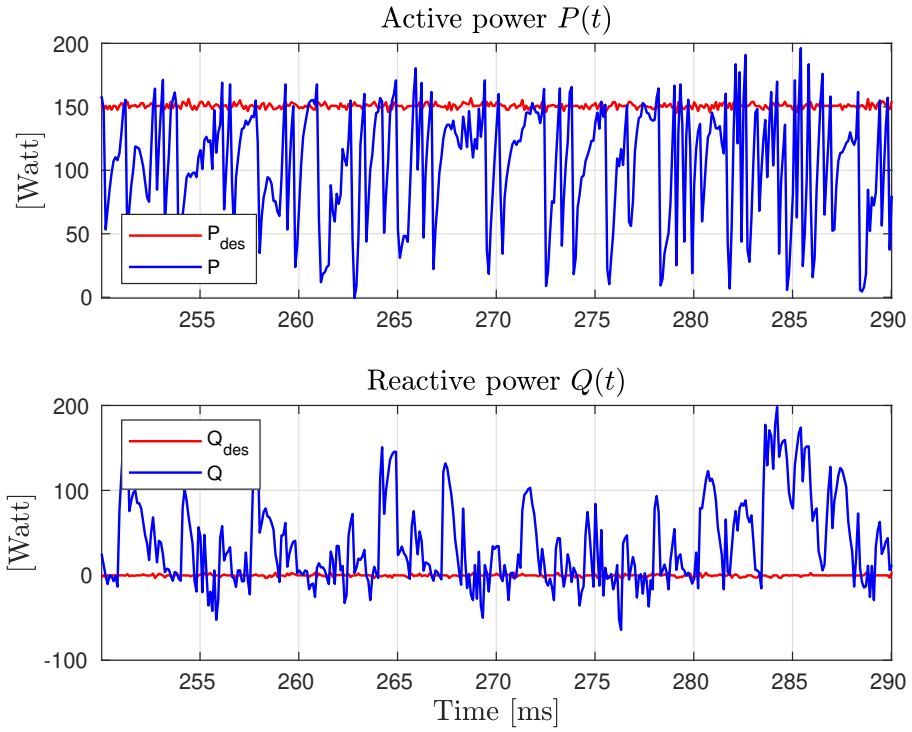


FIGURE 5.44: Experiment. Powers with $P_{des} = 150$[W] and hard vector Table 4.5

Figures 5.44 and 5.45 show the active and reactive power in a wide and narrow time-window, respectively. The pattern does not look as consistent as the one

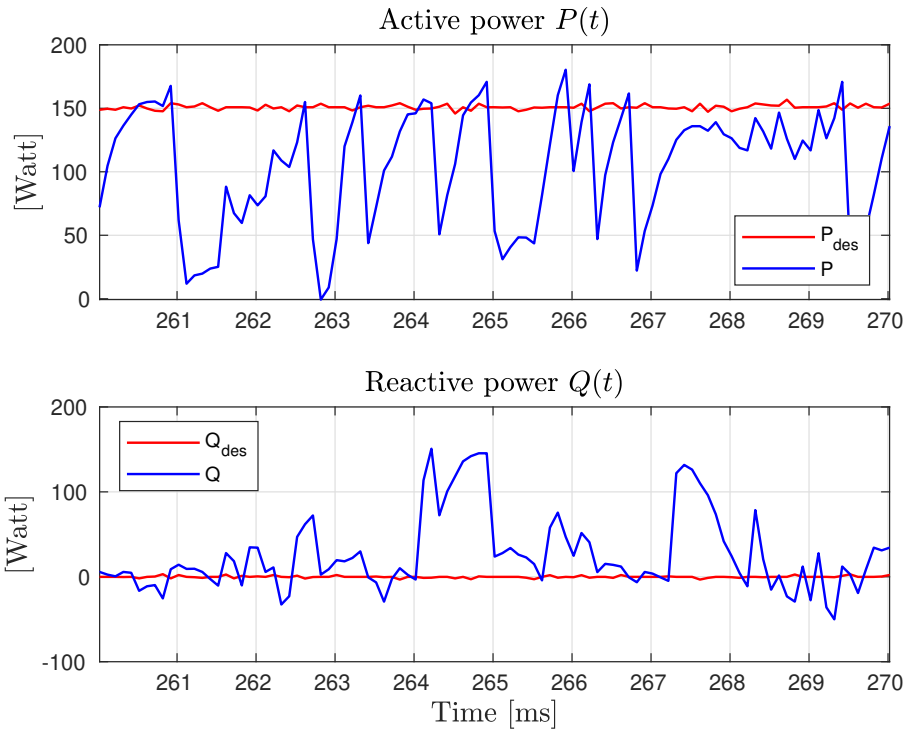


FIGURE 5.45: Experiment. Powers with $P_{des} = 150$[W] and hard vector Table 4.5 (zoom)

in Figure 5.31. In comparison, it does look more irregular, but definitely shows a more 'reactive' (or faster) behaviour. This could mean the speed increase is improving the results. Besides, typical peaks of the new Q are about +100[W] and -10 [W], whereas the Q in Figure 5.31 typical peaks are below +100[W], but can reach as low as -100 [W].

Figure 5.46 shows the transformed currents i_α on top and i_β on the bottom with blue colour. In red are the expected current patterns $i_{\alpha fol}$ and $i_{\beta fol}$ on top and bottom plots, respectively. These pair of values are for reference only and are not used for any calculation in the software. Hence, $i_{\alpha\beta}$ performs in the expected sine pattern (with an irregularity caused by PLL re-synchronising) and it also shows a ripple. This ripple is already smaller than the one presented with a slower digital filter in Figure 5.36. Figure 5.47 presents the grid voltages v_{abc} which are, as expected, affected by the transistor switching. They all show a sine pattern of about 20 [V_{RMS}] and shifted 120^o.

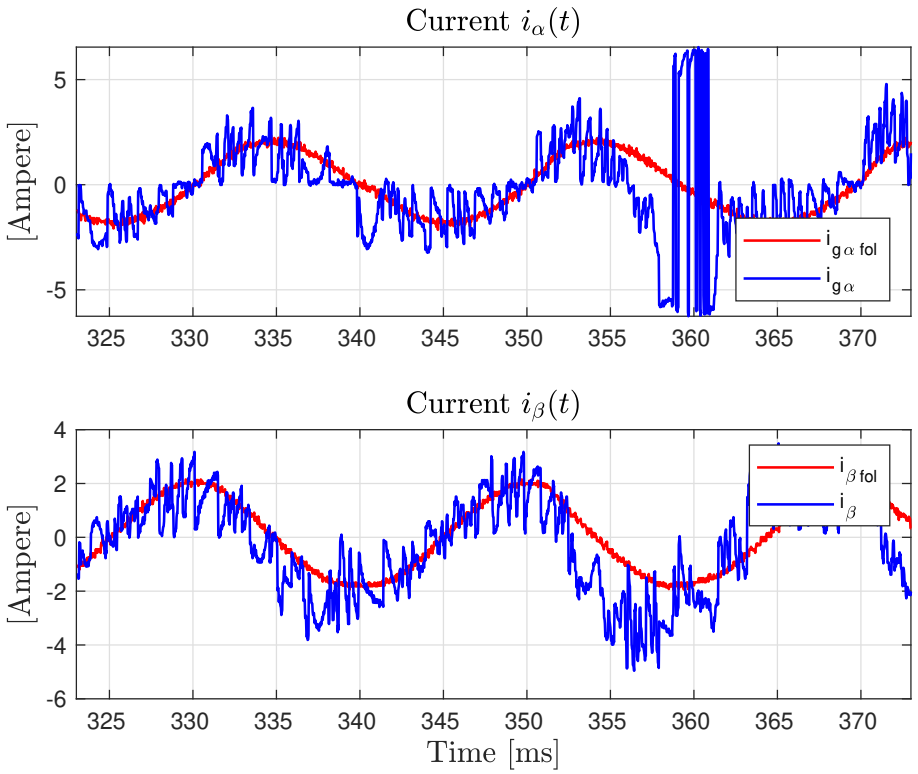


FIGURE 5.46: Experiment. Currents with $P_{des} = 150$[W] and hard vector Table 4.5

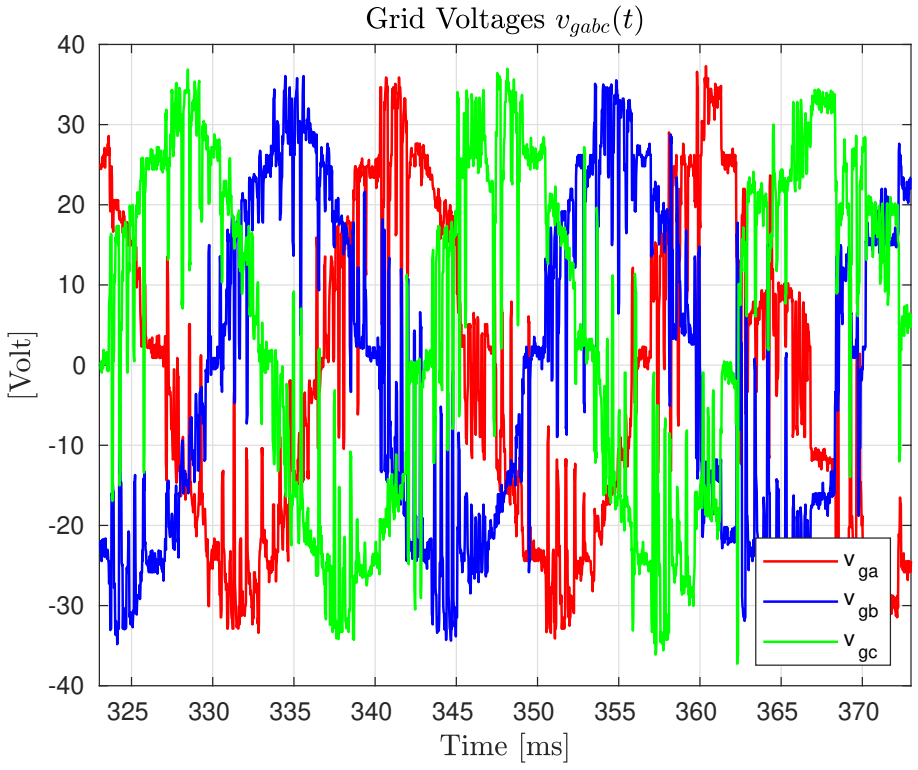


FIGURE 5.47: Experiment. Grid voltages with $P_{des} = 150$[W] and hard vector Table 4.5

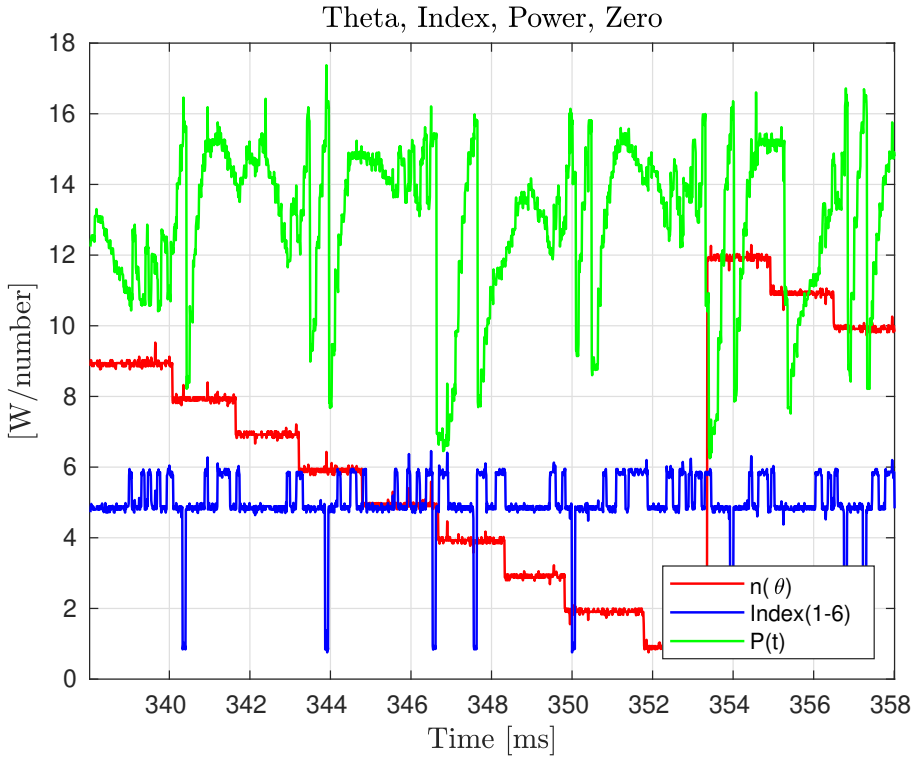


FIGURE 5.48: Experiment. Active power, *index*, and $n(\theta)$ with $P_{des} = 150$[W] and hard vector Table 4.5

In Figure 5.48, the corresponding $index + 1$ value from Table 4.5 is shown. This was achieved by a last minute code for the FPGA board, which jointly with $n(\theta)$, enables deduction of the exact voltage vector active in real time. As soon as P reaches a value above ΔP over P_{des}, $index + 1$ changes (usually to 1) and P inmmediately drops down. $index + 1$ then changes back to values 5 or 6, and the pattern repeats. Although is a different vector, the pattern is consistent with simulation results in Figure 5.41. Besides, the ripple remains significantly high. This strenghtens the hypothesis made earlier that the vector table has been designed too hard or abrupt for when P needs to be reduced.

An additional switching table (or vector table) was designed. Vector Table 4.5 was taken as a base to carefully select alternative voltage vectors, which avoid a drastic reduction of power. Table 5.7 describes the new 'soft' vector table, which was directly tested in the prototype. The following results use this new 'soft' table

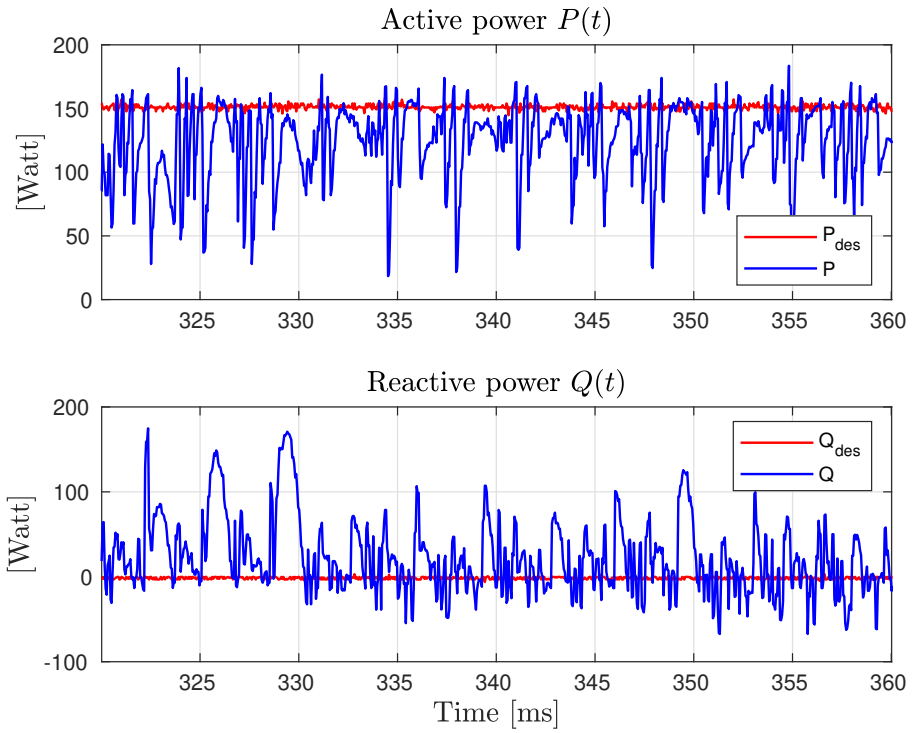


FIGURE 5.49: Experiment. Powers with $P_{des} = 150$[W] and soft vector Table 5.7

and hold the same parameters as usual: $P = 150,000$ [mW], $\Delta P = 5,000$ [mW], $\Delta Q = 5,000$ [mW], and $\Delta C = 20,000$ [mV].

TABLE 5.7: Soft Switching Table for DPC with capacitor balance

P_d	Q_d	C_d	Inx	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	θ_{11}	θ_{12}
0	0	1	0	V_8	V_{17}	V_9	V_{19}	V_{10}	V_{21}	V_{11}	V_{23}	V_{12}	V_{13}	V_7	V_{15}
0	0	2	1	V_8	V_{18}	V_9	V_{20}	V_{10}	V_{22}	V_{11}	V_{24}	V_{12}	V_{14}	V_7	V_{16}
0	1	1	2	V_{23}	V_{12}	V_{13}	V_7	V_{15}	V_8	V_{17}	V_9	V_{19}	V_{10}	V_{21}	V_{11}
0	1	2	3	V_{24}	V_{12}	V_{14}	V_7	V_{16}	V_8	V_{18}	V_9	V_{20}	V_{10}	V_{22}	V_{11}
1	0	-	4	V_7	V_2	V_8	V_3	V_9	V_4	V_{10}	V_5	V_{11}	V_6	V_{12}	V_1
1	1	-	5	V_1	V_7	V_2	V_8	V_3	V_9	V_4	V_{10}	V_5	V_{11}	V_6	V_{12}

Figure 5.49 shows active power P and desired active power P_{des}. Although some sudden and large P drops are present, P follows P_{des} better. Even though the largest P drops nearly reach zero, P immediately ramps up to a close P_{des} value, producing a better average. Reactive power Q also shows some improvement following Q_{des} compared to the previous experiments. Figure 5.50 presents the

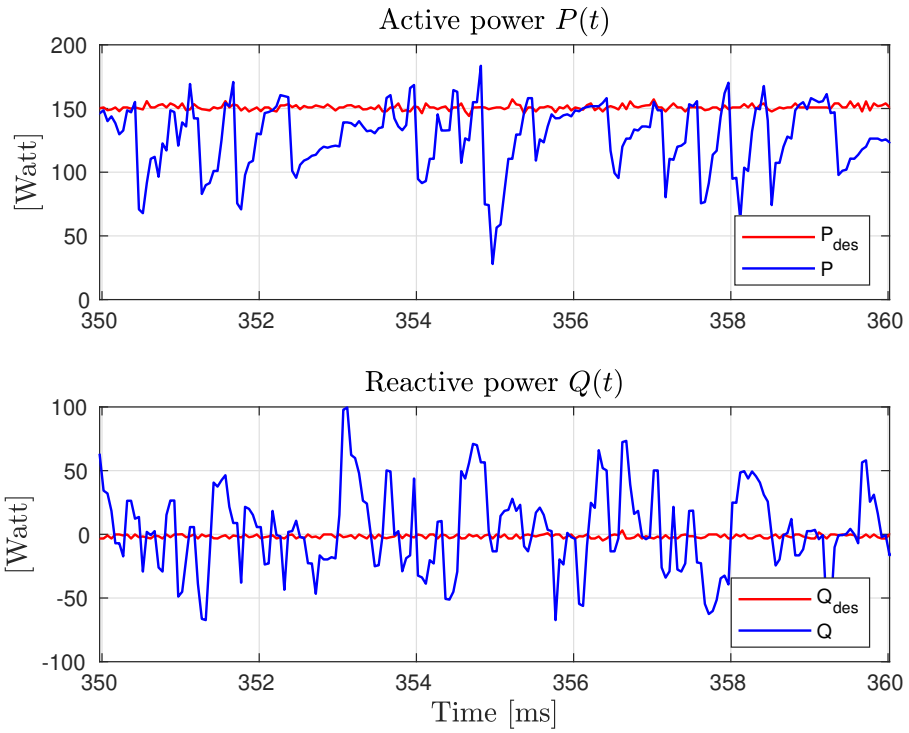


FIGURE 5.50: Experiment. Powers with $P_{des} = 150$[W] and soft vector Table 5.7 (zoom)

same signals in a smaller time-window. Should there be a higher value of $+\Delta P$, an average value of P can reach P_{des} with an average ripple below 100 [W].

Figure 5.51 shows the transformed currents $i_{\alpha\beta}$ with their respective current guides $i_{\alpha\beta fol}$. Apart from the irregular sprint, the currents are smoother with this soft table than the currents presented with the hard table in Figure 5.46. Currents here have smaller ripple and are smooth at times. It also follows the reference better, hence, having a more sinusoidal shape. The grid voltages with the soft table are shown in Figure 5.52, where significant ripple is present, slightly irregular at times. They still follow a sinusoidal pattern with about 30 [V] peak and shifted 120^o between them.

A vector localisation plot with $index + 1$ and $n(\theta)$ is presented in Figure 5.53, along with P. $index + 1$ spends most of the time in values 5 or 6, but when it changes, it usually lands on 3 or 1. It again appears that as soon as any of these

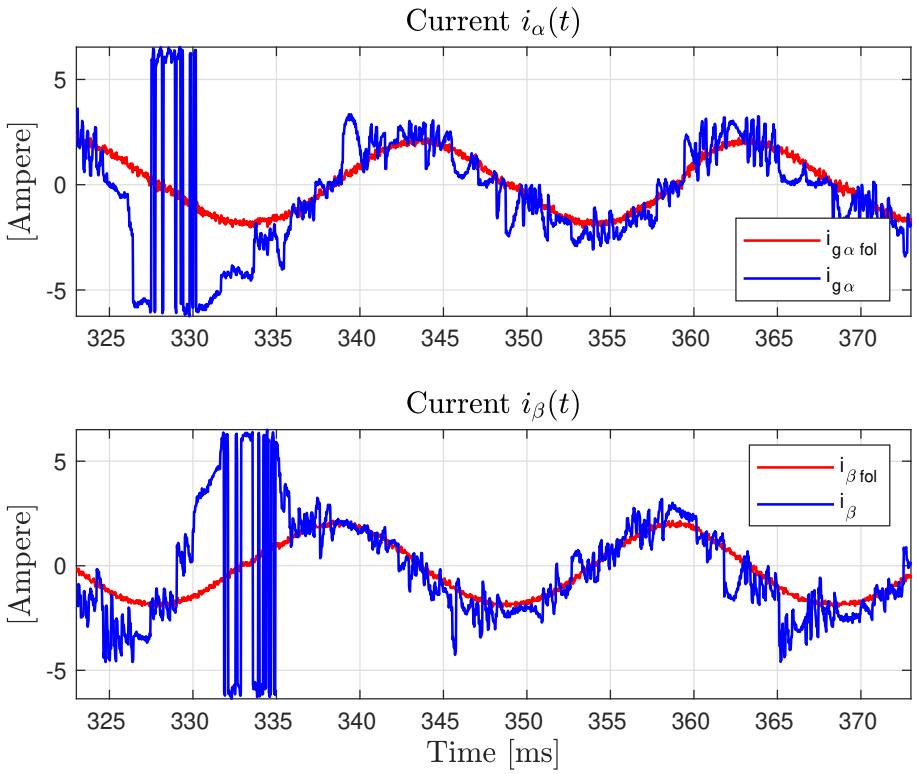


FIGURE 5.51: Experiment. Currents with $P_{des} = 150$[W] and soft vector Table 5.7

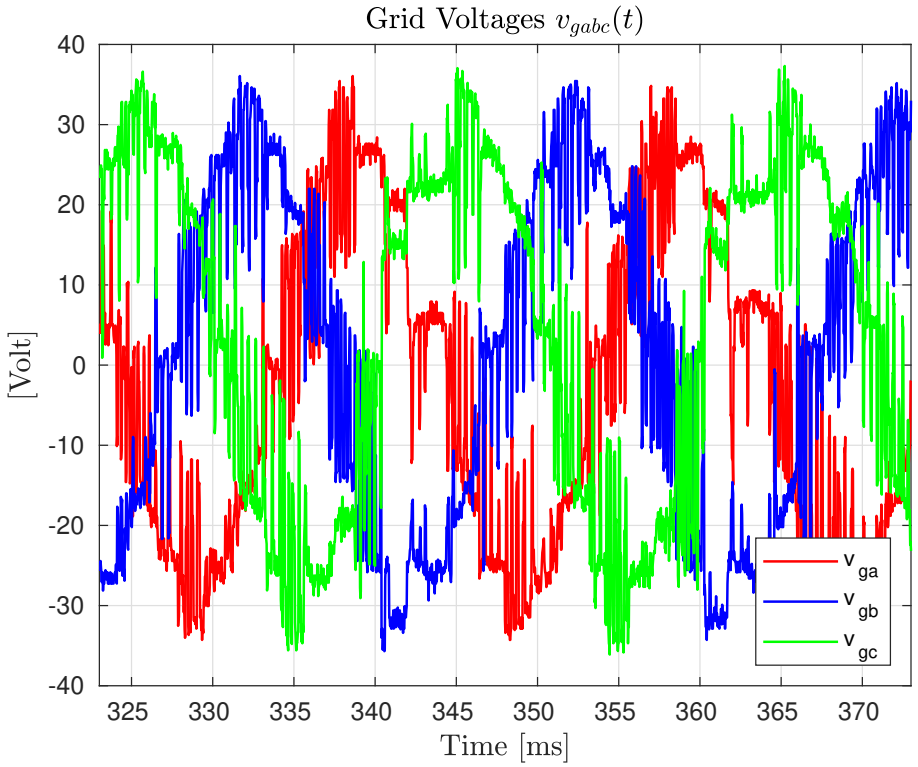


FIGURE 5.52: Experiment. Grid voltages with $P_{des} = 150$[W] and soft vector Table 5.7

FIGURE 5.53: Experiment. Active power, *index*, and $n(\theta)$ with $P_{des} = 150$[W] and soft vector Table 5.7

latter values are selected, P drops. However, this is a more dynamic pattern than the one for the hard table.

Further experiments for $P_{des} = 200,000$ [mW] (while the rest of the parameters are kept the same) were conducted in a similar fashion as in this subsection, and are available in Appendix C.

5.5 Summary

In this chapter, every aspect of the prototype was detailed to the best of the author's knowledge. All the hardware parts were described with their most relevant parameters, from the DC-AC, through the communication and power channels, to the conditioning and calibration, including the own built parts. Labelled photographs, tables, codes, diagrams and schematics, including a few figures from

other sources, were used to support relevant hardware and software properties or operations.

Source code documentation was provided for every piece of the *Quartus* project, including the corresponding theoretical fundations when necessary. Graphics of the connections between software blocks were also reported. Finally, the experimental outcomes were presented and described, mostly in form of plots. Causals of performance issues were discussed and simulated. Experimental outcomes started with initial results on a 'low speed' digital filter and ended with improved results, which included a new 'soft' switching table.

Chapter 6

Conclusions and Future Works

6.1 Conclusions

An NPC type of inverter has been chosen due to its medium and high voltage capability as explained in Chapter 2. After introducing single– and three–phase models of the NPC inverter, the single–phase model is verified with SVPWM in Chapter 3. Here, the model of a single-phase NPC inverter was developed and modelled in state-space form, producing a hybrid system (switched and continuous approach). The model provides very accurate results compared to the standard circuit model in *Simulink/MATLAB*. Also, the SVPWM proved to work successfully with the expected output wave form and a simple control kept the capacitors' voltage under equilibrium in spite of changes in the load and inductance in the grid model.

Therefore, the problem of output voltage regulation for the single–phase NPC inverter and the capacitor balance problem was addressed with the proposed algorithm. The voltage regulation was kept and the effect of changing RL values showed the effectiveness of the control. However, it was seen that it works for a limited amount of current, given the direct dependency of the DC voltage source.

This hybrid model is a unique way to present the NPC inverter, which allows a very simple control and models a detailed switching nature of the system.

A mathematical representation of the inverter aids in the understanding of the current flows and dynamics of the inverter. It is useful for detecting current and voltage patterns, which could lead to improved switching patterns and increased control of the switching losses. However, for grid connected systems, more efforts are put towards the modelling of the grid itself in order to cope with change of impedance and other grid issues.

In Chapter 4 a three-phase NPC inverter was introduced and its interaction with the grid was analysed in depth. Two switching strategies were implemented and its advantages and drawbacks were discussed in terms power ripple, losses, and processing complexity. In general terms, the control technique which presented the most efficient results in terms of switching action, is the PI decoupled control due to its controlled switching approach. Usually, the desired three-phase voltages v_{gabcD} ordered by the algorithm swing in a leg basis from negative to positive, (or from positive to negative) voltage values, always passing through the zero value. For this reason, the switching action on the transistors could be smoother. Besides, the PI Control prioritises convergence of P in the simulations by initially reaching a close value to P_{des} and having Q with a large start peak. Once the decoupling control manages to decrease Q to a close-to-zero value, P converges most precisely to P_{des}. In other words, the closest convergence to both P_{des} and Q_{des} is achieved slowly.

The DPC on the other hand, makes P and Q reach P_{des} and Q_{des} much faster. The speed of convergence relies in a fast processing of the algorithm. The PI Control also relies on this, but also on the design of the decoupling control, which could be different for every scenario. Nevertheless, DPC could result in a more efficient performance only when the switching pattern is controlled and more organised.

With the DPC control, the algorithm will push the boundaries of the switching capacity in the system under test as it is vector based. PI Control on the other hand takes a leg-modulation approach, which at a given frequency, the algorithm will, at some extent, adapt to the restrictions and speed of the system.

PI decoupled control arguably requires more processing resources, as it is composed of four transformation stages, a decoupling control algorithms with integrals, plus a type of PWM. However, this widely depends on the code design, as LUTs could be used; in which case it would need more LUTs than DPC. DPC only needs one transformation, a simple calculation of power and error, plus a switching table. The only obvious LUT for the DPC algorithm is the vector table, which theoretically takes more time to design, but less LUTs and processing time once implemented in the algorithm.

PI Control does not converge as fast (unless the gain values could be found or a more sophisticated control is implemented), but it reaches a close value to the desired powers. This could be good enough for applications were convergence speed is not critical. A potential problem for this, is the difficulty to find the right gains, which usually differ in simulations and experiments.

In a nutshell, PI Control will mostly depend on the control gains for specific values in an NPC inverter, but it will generally have slower convergence. Usually, it will have better efficiency regarding switching, unless DPC is fitted with a more organised switching scheme. DPC requires more design skills as it is more vector focused, but it will not only converge faster to an initial P_{des} and Q_{des}, it will converge to them also faster should a change on these (or other paramters) happen.

Experimental works were carried out using an FPGA board and the results are reported in Chapter 5, were problems with current signals were found. The resolution of the current in the system (or the speed at which the current values are

sampled into the algorithm) is proportional to the magnitude of the power ripple. This holds true for the results presented in this Thesis. Once this resolution is sufficiently high, the power ripple will then depend on the switching speed of the transistors. If the vector table forces an abrupt change in the power direction and the algorithm does not read this change on time, the power could easily exceed ΔP and ΔQ as much as the delay in the reading of the change plus the time the algorithm requires to process it until an output is produced.

In the results presented in this Thesis, because P takes some time to reach P_{des} (hence its ramped shape), the algorithm moderately manages to detect when P reaches $P + \Delta P$ (as it still has time to read power changes). Not so with $P - \Delta P$, due to the much faster change when power is intended to decrease. The power ripple depends on several factors inherent to the DPC algorithm and the hardware/-software implementation. Inherent to the DPC algorithm: maximum transistor switching frequency, hysteresis values, vector table and grid impedance. From the perspective of hardware/software implementation, the power ripple depends on the following: algorithm processing speed, the data input frequency from the sensors through the ADC, and the data output frequency. All these factors are interconnected, however, for this particular prototype the data input frequency from the sensors through the ADC was the underlying cause of the problems.

As the small vectors are far below $P - \Delta P$, that explains why a SV reduce P so drastically, particularly when the small vector is chosen so that the decrease rate of P in the vector map is at its maximum. Because of this, the 'soft' vector table was designed by choosing a vector which is not reducing P at its maximum, but instead, one that reduces P at a lesser extent (same for Q) while still making sense of the desired outcome for both powers. This new 'soft' vector table is effective to soften the large and continuous drops of P, as shown in the experimental results.

Regarding the grid voltages, as long as the PLL accurately detects the phases

and magnitudes, it is possible to use a LUT to produce a smooth voltage vector reference, or θ value. Assuming the output voltage on the AC side of the inverter is known (as well as the grid vector position θ), the algorithm can be driven by the 'follower' voltages produced by the PLL.

The time taken to design, construct, assemble, connect, code and operate all the pieces together provided a better understanding of the capabilities and limitations of the prototype. It also provided skills and experience with: learning VHDL and verilog languages, opamp configuration, analog to digital conversion and filtering, digital to analog conversion, data processing and analog to digital design (memory usage, display systems, etc.), and safety precautions. Moreover, the lesson that simulations of prototypes must be as accurate as the experiments is paramount when integrating different hardware, and particularly for a prototype with an FPGA board, where there is plenty of low-level language processing and signals are not easily displayed to the human eye. Not much so, when the system is simpler or is driven by a more sophisticated equipment such as DSpace, or other dedicated DAQ (data acquisition boards).

All in all, the aim and objectives stated in Section 1.2 resulted in the following contributions:

- An state-space model of a single-phase NPC inverter with SVPWM.
- A new vector table for DPC with capacitor balance for a grid-tied NPC inverter.
- An evaluation between PI control and DPC for a grid-tied NPC inverter by simulations.
- An experimentally tested new 'soft' vector table for low-speed DPC algorithms in a grid-tied NPC inverter.

6.2 Future Work

The power quality delivered in the experiments in spite of numerous code improvements, is not as it was desired. As explained earlier, there was an underlying cause of the power and current ripple: the speed of the signals acquisition, current signals in particular. The future work for the NPC inverters in the context of this Thesis is described in the following subsections.

6.2.1 Signal Acquisition

The underlying cause of the experimental problems was signal acquisition, and this is the most obvious issue for further study. DPC control is more agressive than PI control, and is capable of pushing the limits of the processing speed and in/out capacities of the driving system. This is why it requires very high speed for signal acquisition and algorithm processing. How much capability the available hardware and software can provide remains to be explored.

The critical elements to improve these measurement readings (signal acquisition) are a piece of software and a piece of hardware, the *Digital Filter* and the ADS1204, respectively. Normally, the more readings accumulated in the ADC filter the more accurate the reading can be, but the slower the reading input would be. If the filter input reading is fast, it does not accumulate enough data to be precise.

A further difficulty with the acquisition of current signals is the fact that the design of the ADC with its digital filter is not as easy as designing a filter using standard theory. The filter has to be analysed in the context of the different clock frequencies and it also has to be coded in a low-level or digital language (bit-by-bit), e.g. Coding filters using decimals did not work during the attempts in this prototype.

The voltage signals need to be read with as less switching noise as possible, while still detect the phases accurately for the PLL. The current signals also present ripples, and consequently the VHDL code should be optimised to have reduced noise and still have precise, high resolution, readings of current. The reason for this, is that (as seen in Chapter 5) some vectors can dramatically speed current drops, and these need to be read as soon as possible. How precise these readings are could significantly improve the resolution of the algorithm.

From the algorithm perspective, assuming the processing speed is sufficiently fast, the algorithms are restricted by two main issues: the speed of the signal acquisition, and the speed of the transistors. How fast and accurate this particular hardware can acquire signals, or what specific filter configuration shows the best performance becomes relevant. In an scenario where faster transistors are in place (e.g. SiC MOSFETS or IGBTs), this becomes more relevant because it has to be found whether this particular hardware/software arrangement is enough for driving the algorithm. If it is, this includes investigation on its limits. If it is not, enquiries on its constrains will be risen, whether it is hardware, software or even sensors. This should include answers on what could solve the problem, alternatives such as customised set of opamps for example, or if there are other alternatives for acquisition of these signals.

6.2.2 Grid Impedance and PLL

A notable limitation of the inverter design and implementation in the laboratory is the *RL* filter, which prevents a quality output power and keeps the high frequency component coming from the switching frequency f_s. This could seriously reduce the quality of the power injected into the grid, particularly in high power applications.

The simulation described in subsection 5.4.2 was re-run with different values for the 'grid-side', in which it is assumed the transformer is in. Active and reactive power performance for simulation of two sets of RL values is shown in Figure 6.1. The set $RL1$ comprises $R_g = 3[\Omega]$ and $L_g = 0.1[mH]$, while the set $RL2$ comprises $R_g = 0.01[\Omega]$ and $L_g = 10[mH]$.

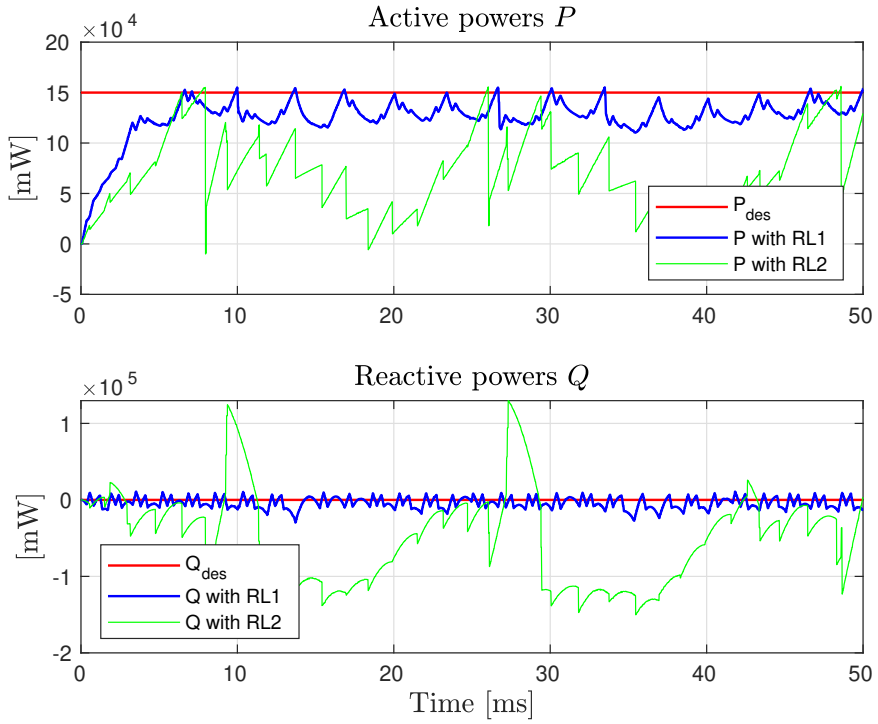


FIGURE 6.1: Power for $RL1$ ($R_g = 3[\Omega]$, $L_g = 0.1[mH]$) and $RL2$ ($R_g = 0.01[\Omega]$, $L_g = 10[mH]$).

Looking at the $RL1$ results in Figure 6.1, the active power P shows a nearly consistent peak pattern, which barely reaches P_{des}. As the resistance increases, P will show a similar pattern but at an even lower power. The reactive power Q performs in a similar pattern, very close to Q_{des}, with a small ripple. Looking at the $RL2$ results on the other hand, P performs very poorly, as it shows a high frequency ripple swinging at the grid frequency, only barely reaching P_{des} once during the peak of the low frequency ripple. The reactive power Q is kept mostly negative, with a sharp increase during P peaks.

In general, the larger the inductance and resistance RL, the more difficult is for P to reach P_{des}. The larger the inductance is, the larger the power ripples will be, and also the more irregular and offset the currents will be. The larger the resistance, the lower the P ripple is and the more triangular the currents become, but P peaks would never reach P_{des}. Increasing of L makes P to oscillate more. If the grid voltage v_g decreases, the current i increases. Ways in which these effects can be managed or mitigated should be investigated as they are related to the transistors and grid frequencies, voltages and passive components.

Another consequence of the high frequency component, is the malfunction of the PLL (as shown in the previous chapter) or the phase detection algorithm. Because the voltage waves are produced by switching different DC voltages with transistors, the PLL could read errors. This is also the reason why the voltage readings should be properly filtered, as noted earlier.

A problem of this particular PLL is the detection of the voltage crest, as the voltage reading could be just below or above the maximum expected peak value, the PLL could get 'confused' as if the crest is moving upwards or downwards already, making it update its θ value constantly in a small period of time.

As the voltage and current waves could change depending on the impedance on the grid side, it could be interesting to model these and find techniques to make the PLL detect these -sometimes small- changes and react accordingly. Even for the simplest change such as a DC voltage or grid voltage change, the PLL could be engineered to operate optimally. Naturally, this could be extended to other grid issues.

Potential solutions could be placing the voltage sensor closer to the grid, estimate the value of the RL components, and using an LCL filter. The implementation of an LCL filter in the laboratory could be interesting to test the performance of the inverter. An adequate configuration with the capacitors could allow to reduce

the high frequency component. The quality of the output ripple could then be measured with and without the filter, and the design of the filter itself could be optimised for the selected parameters.

6.2.3 DPC vs PI Control

An improvement to the DPC scheme could be achieved by defining switching patterns in order to reduce the switching losses. This improvement could lead to more detailed results of the efficiency of each scheme and the importance of prioritising parameters such as power, capacitor balancing or output ripple. For example, a leg pattern $s_x = [1\ 1\ 0\ 0]$ should not change straight to $s_x = [0\ 0\ 1\ 1]$, but must pass through $s_x = [0\ 1\ 1\ 0]$ first. This poses a design problem directly in the DTC Table 4.3 which could achieve higher efficiency by defining these switching patters, but could also lead to reduced speed of convergence of the power to the desired power. How much the design of a pattern affects the convergence speed is left as a future work.

Related to the switching patterns and recalling the 'soft' vector table, the open question is: At what extent the current waveform can be predicted? The answer is inevitably tied to the inverter design and the grid impedance, but it could also hold new approaches to vector table designs on the DPC side and perhaps optimise the already proposed 'soft' vector table for low-speed algorithms. The answer could also be related to improvements for the PI Control, or even other control approaches.

6.2.4 Further improvements

Incorporation of full power control to the single-phase NPC inverter with SVPWM could provide good insights on its advantages and limitations. Its advantages are

expected to be lower output ripple (than an RL filter) due to the LCL filter used. Also, its output power is expected to be limited mainly by the DC voltage source, capacitor values and naturally the voltage-current limits of the transistors used. However, a more detailed study could result in more detailed metrics.

On the modelling and control of both, the single– and three–phase inverter, the study of the bi-directionality in the state-space model and in the prototype could be useful for applications such as G2V and UPS. Exploration of the capabilities and requirements for operation in rectifier mode are completely worth it and it includes the DC voltage ripple with its possible conditioning via a DC-DC converter (buck, boost, or buck-boost). For this, the resolution of the capacitor voltage signals need to be higher, particularly if the parameters in the application are more demanding.

Other improvements on the specific software/hardware implemented, which could increase the performance of the prototype are:

- Digital Filter: A suitable digital filter analysis for faster signal acquisition, particularly for current readings.
- DAC: Incorporating a dedicated ± 10 [V] power supply ensures better resolution and constant output values.
- Sensors: Installation of capacitor voltage sensors fitted for the current voltage values. Re-configuration of voltage sensors to fit the output voltage values.
- Usability: Create library to use LCD screen and display relevant values in decimal (includes library for decimal values conversion). Assign few switches/buttons for signal display and most switches for algorithm change in real time. Basically improve the User-Interface (UI).

- PLL: This has many potential improvements. Design it to 'speed' on the oposite direction (maybe for rectifier mode?). If included with peak-voltage measurement, it could work at different voltage values.

- Software: Include algorithms for calculation and display of more signals of interest: voltage peak values, average power, efficiency, power errors, etc.

- More software: Design any of the control schemes DPC or PI decoupled control with Model Predictive Control (MPC).

Appendix A

Adaptable Model for Si/SiC MOSFET and IGBT

Many existing modelling frameworks for IGBT and MOSFET devices require detailed information of the physics of the real devices, sometimes even including electrochemical equations, to assess strong accuracy in simulation. These models are usually software specific, they are very time-consumming and some of their parameters are nearly inaccessible. In this paper, we propose a flexible modelling framework that allows to reproduce accurately the behavior and assess the "performance" of IGBT and Si/SiC MOSFET devices and in particular the main static and dynamic characteristics. Advantages of this framework are: a) Simple extraction method of the parameters from commercial device datasheet, b) Flexibility to implement on different software platforms because the model is described by an electrical circuit composed of basic linear components and ideal diodes, and c) Versatility to model new transistor materials and technologies, such as SiC device. Tests to verify the modelling framework are presented using an IGBT, Si/SiC MOSFET devices.

Nomenclature

Symbol	Description
F_s	Switching frequency
R_{on}	On-resistance
V_{cc}	DC bus voltage
I_c, I_{cN}	Collector or drain current, Rated I_c
V_{CE}, V_{CEN}	Collector-emitter or drain-source voltage, Rated V_{CE}
V_{GE}	Gate-Emitter or gate-source voltage
V_F, V_{FN}	Diode forward voltage, Rated V_F
Q_{rr}, Q_{rrN}	Reverse recovery charge, Rated Q_{rr}
t_r, t_{rr}, t_f	Rise time, Recovery time, Fall time
t_{rN}, t_{rrN}, t_{fN}	Rated t_r, Rated t_{rr}, Rated t_f
V_{CO}	Threshold V_{CE}
I_{CM}	Maximum collector current
V_{FO}	Diode threshold voltage
C_{gc}	Gate-collector or gate-drain capacitance
C_{ge}	Gate-emitter or gate-source capacitance
C_{ce}	Collector-emitter or drain-source capacitance
C_{ies},C_{iss}	Input capacitance
C_{oes},C_{oss}	Output capacitance
C_{res},C_{rss}	Reverse transfer capacitance
P_t	Forward losses of the transistor
P_d	Forward losses of the diode
P_{on}, P_{rr}, P_{off}	Turn-on, Recovery, Turn-off losses
P_{total}	Total losses per cycle
E_{total}	Total energy per cycle

A.1 Introduction

A power electronic switching device is usually commercialized with a datasheet and eventually by an associated model for simulation purpose. The models provided by manufacturers are generally dedicated to a specific software, such as PSpice or LTSpice. Normally, when a new device is released to the market, it often takes time before the associated model is made available to general users. A newly released device can attract researchers' or system developers' interest to study or to be used for design experiments and a model is thus often required for these purposes. This is frequently the case for wide bandgap devices such as those based on Silicon-Carbide (SiC) and Gallium-Nitride (GaN), including SiC MOSFETs [62, 63, 64] and SiC IGBTs [65, 66]. While many of the devices are currently commercially available, their models are still not easily accessible. In such cases, a model to predict the performance of any device by simply using information available from its datasheet will be extremely useful [67].

For the above reasons, model development of semiconductor devices has been a very active area of research, and in particular regarding SiC and GaN devices, that present improved performance against their Silicon (Si) counterparts [68], with potential applications such as for higher efficiency, high-power-density power converters. In [69], a modelling procedure for a SiC MOSFET with extracted parameters from datasheet is proposed. While the results show a very close match between the simulations and the experiments, some parameters used in this modelling procedure are tightly related to the physical and chemical properties of the semiconductor materials that form the device, making it complex and exclusive to only one or two simulators such as PSpice [70]. In [71], a SiC MOSFET is also modeled in PSpice using an equivalent subcircuit. The proposed model successfully reproduces static and dynamic characteristics of the device, although no explicit description of the model is provided. At the opposite, many models

[72] are too simple and not very accurate. However, a simple modelling method that combines datasheet and measurements could offer excellent results [73], particularly for newly developed devices. While there have been various works on device modelling, those that are easy and quick to implement on a generic software and are suitable for dynamic modelling are pretty rare. Recently, the use of MATLAB/Simulink for the device modelling and characterization has been very popular and increasingly used in dynamic modelling and control power electronics systems. For instance, [74] used MATLAB/Simulink to model the power loss of SiC MOSFET and Si IGBT, and shows good results for "performance" prediction (e.g. in terms of switching losses, turn on and turn off switching time). Other works, such as [75, 76, 77], have focused only on modelling SiC MOSFET and the characterization curves are only presented for low V_{DS}. Therefore, developing a more thorough modelling framework for more general devices which is easy to implement is still an open problem.

Among the existing modelling approaches, the model reported in [78], the so called *Alonso model*, was originally introduced to model an IGBT as an electrical scheme that is easy to implement on various software platforms; the results of the characterization with this model should be consistent as close as possible with the device characteristics. This IGBT model is represented by an equivalent model as a combination of a MOSFET and a Bipolar Junction Transistor (BJT). In this paper, a generalization of the procedure to use the Alonso framework in order to characterize new generation of IGBT and MOSFET is proposed. We choose MATLAB/Simulink environment to implement the model and to perform the characterization. The purpose is to provide a simple library of Si and SiC devices for easy integration to power electronic systems built in any software environment.

The paper starts with a brief description of the original Alonso model including the technique used for the parameters extraction. The generalized Alonso framework is then presented. Later, simulation results for static and dynamic characteristics

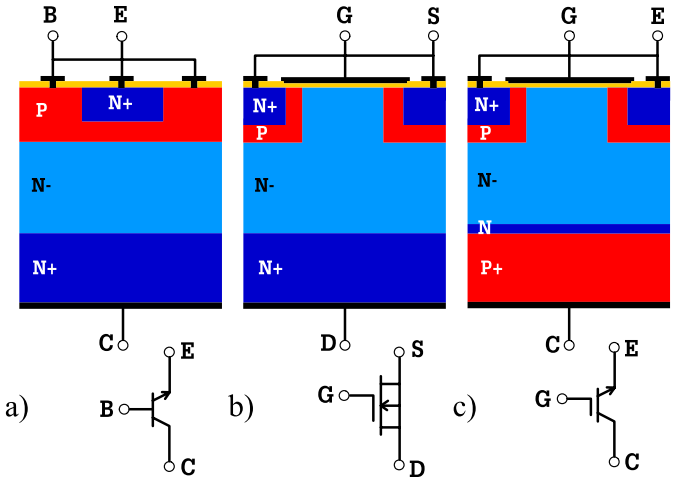


FIGURE A.1: The combination of a a) Bipolar Junction Transistor (BJT) and a b) MOSFET resembling an c) IGBT

of the device are presented, with the focus on Si/SiC MOSFET and IGBT commercial devices. The results are then compared with the characterization curves provided in the datasheets and the real measurements. Finally, losses calculations are presented and the flexibility of the model is summarized in the conclusion.

A.2 Presentation of the Alonso Model

An IGBT is essentially an N-channel power MOSFET on top of a p+ type substrate. Fig. A.1 illustrates the structure of a BJT, a MOSFET and an IGBT; it is well known that the IGBT is practically a bipolar transistor whose base current is provided by a MOSFET.

A power switch can be described and separated into the static (dc) part, which represents the static I-V characteristics, and the dynamical part, which represents the evolution of the working point in the I-V characteristics according to the internal dynamics of the power switch [79]. The principle of the Alonso IGBT model is to "encapsulate" the static part, inside the dynamical part allowing a unification and a coherence of the electrical interactions between the parts.

The Alonso model [80] has been designed to provide an efficient way to combine simulation tools with realistic models. This model has the particularity to be defined by an electrical circuit exclusively based on basic components that can be easily find in any simulator (Fig. A.2); it was initially tested in SUCCESS software.

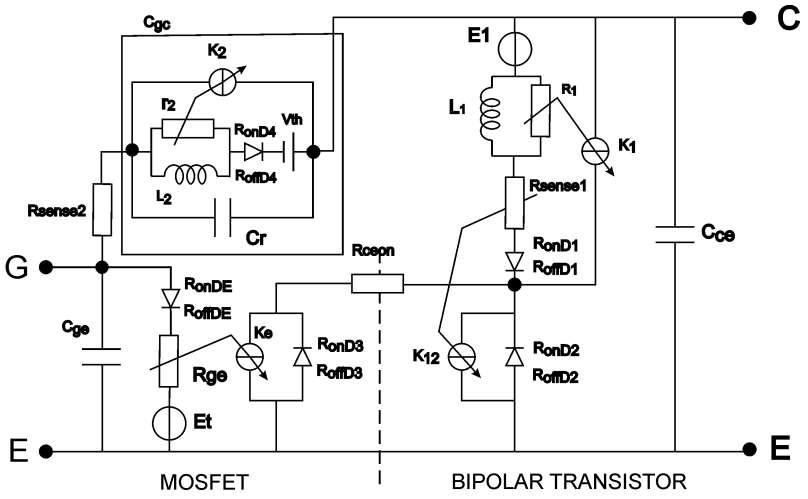


FIGURE A.2: Original Alonso IGBT model.

According to the IGBT concept, we can separate in this model, the MOSFET transistor part and the BJT part. Both are linked and three capacitors C_{gs}, C_{gd} and C_{ds}, whose values depend on the v_{ce} voltage, represent the inter electrodes capacitors. All the parameters are well defined and a complete process allows to calculate them by considering the output characteristic $I_c - V_{ce}$ diagram and the evolution of the capacitors C_{gs}, C_{gd} and C_{ds} given in datasheets. The complete procedure to calibrate to the Alonso model according to any IGBT datasheet is provided in the next section.

A.3 Parametrization of the Alonso Model

A.3.1 Notations

All the components are linear excepted the diodes, which are considered ideal. The voltage points g, c, e represent respectively the grid, the collector and the emitter for an IGBT and the voltage points g, d, s represent respectively the grid, the drain and the source for the MOSFET.

The quantity $(i_c, v_{ce}$, resp. (i_d, v_{ds}) defines a static working point in the I-V plot for the IGBT device, resp. the MOSFET device.

For all diodes, denote R_{onDX} and R_{offDX} respectively the turn-on resistance and the turn-off resistance. X denote the index of the diode inside the circuit. J_e and J_2 are two voltage-controlled current generators such as $J_e = K_e v_{Rge}$ and $J_2 = K_2 v_{Rsense1}$ where v_{Rge} and $v_{Rsense1}$ are respectively the voltages across Rge and R_{sense1}.

A.3.2 Parameter extraction

The parameters extraction procedure indicates that specific current and voltage values from the static I-V curves are extracted. Referring to the parameters usually given in datasheets, it is generally sufficient to extract the following parameters for the static I-V plots: the maximum C-E voltage V_{CES}, the output resistance R_{CE} for the IGBT or R_{DS} for the MOSFET, and the values of currents and voltages extracted at three key points, I_{C1}, I_{C2}, I_{C3}, V_{CE0}, V_{CE1}, V_{CE2}, V_{GEa}, and V_{GEb}, as shown in Fig. A.4 and Fig. A.3 for IGBT, or their equivalence in MOSFET nomenclature.

If some of these are not available from the datasheet, the following procedure should be carried out. While the presented procedure is for an IGBT, the same

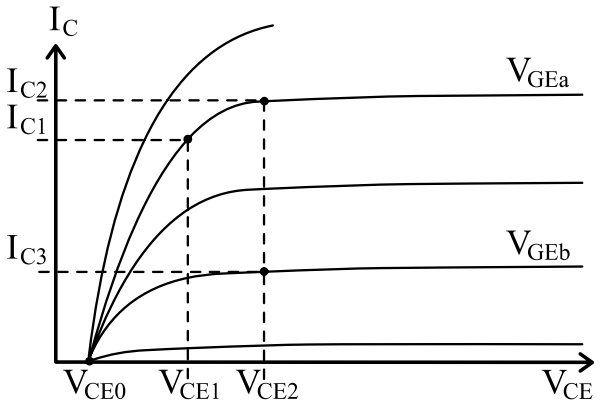


FIGURE A.3: Illustration of the keys points on the generic I-V plot to be extracted.

steps can be applied in case of a MOSFET.

1. Set $V_{CE} = 0$ V and $V_{GE} = V_{GES}$, $I_G = I_{GES}$. Find the value of the gate current, I_G, when terminals C and E are short-circuited and the maximum allowable voltage is applied to the gate.

2. Set $I_C = I_n$ and find the amplitude of the gate decay current, I_{tail}, for a blocking IGBT under its nominal current as shown in Fig. A.4.

3. Extract the operating points from the curves I_C vs V_{CE} as shown in Fig. A.3.

   - Identify $I_C = 0$ A and set V_{CE0}.
   - Set V_{GEa} in one of the higher V_{GE}, avoiding the highest one. Set $I_C = I_{C1}$ and $V_{CE} = V_{CE1}$ at the intersection between the linear zone and the quasi-saturated zone in V_{GEa}.
   - Set $I_C = I_{C2}$ and $V_{CE} = V_{CE2}$ at the intersection between the quasi-saturated zone and the saturated zone in V_{GEa}.
   - Set V_{GEb} in one of the lower V_{GE} and set V_{CE2} and $I_C = I_{C3}$ accordingly.

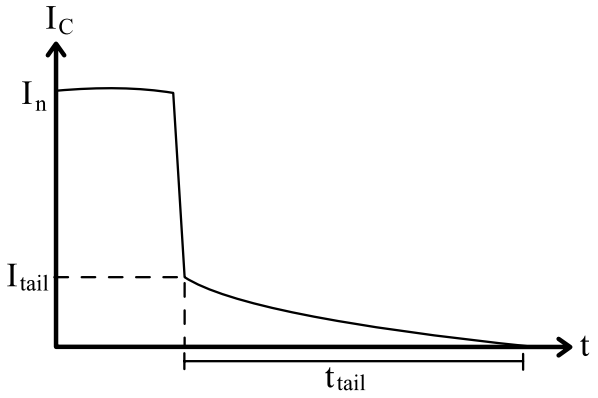


FIGURE A.4: Associated parameter to be extracted for the turn off current extinction.

4. One more operating point must be extracted from one of the three different operating areas below.

   - Set $V_{GE} = V_{GEn}$, $V_{CE} = V_{CE4}$ and $I_C = I_{C4}$, where n is a selected V_{GE} Extract the operating point from the curves I_C vs V_{GE} for $V_{GE} > V_T$.
   - Set $V_{GE} = V_{GEn}$, $V_{CE} = V_{CC}$ and $I_C = I_{CC}$. Extract the operating point when the IGBT suffers short-circuit failure for $V_{GS} > V_T$.
   - Set $V_{GE} = 0$ V, $V_{CE} = V_{CES}$ and $I_C = I_{CES}$. Extract the operating point from technical manuals which represents the measured current I_C while the control connections are short-circuited and the maximum permissible voltage V_{CE} is applied to the component.

This final step implies that the choice of one of these operating points causes an error representation in the other two areas of operation.

A.3.3 Initialization

The parameters requested by the model are calculated from the relationships presented in Tables A.1 or A.2.

TABLE A.1: Calculations for IGBT

Parameter	Parameter
$E_1 = V_{CE0}$	$R_{ceon} = \frac{V_{CE2} - V_{CE0} - (R_{sense1} + R_{onD1})I_{C2}}{(1 - K_{12}R_{sense1})I_{C2}}$
$R_{GE} = \frac{V_{GE}}{I_{GES}}$	$R_{onD3} = \frac{I_{C1}}{I_{C2} - I_{C1}}R_{ceon}$
$R_{onDe} = \frac{R_{ge}}{100}$	$R_{offD2} = \frac{R_{ceon}}{10^{-9}}$
$R_{offDe} = 1000R_{ge}$	$R_{onD2} = \frac{5.1^{-2}I_{C2}R_{ceon}}{5.1^{-1}(I_{C1} - I_{C2}) + I_{C1}I_{C2}R_{ceon}}$
$R_{offD1} = \frac{E_1}{10^{-9}}$	$X = V_{CES} - V_{CE0} - (R_{sense1} + R_{onD1})I_{CES}$
$E_t = \frac{I_{C3}V_{GEa} - I_{C2}V_{GEb}}{I_{C3} - I_{C2}}$	$Y = (1 - K_{12}R_{sense1})R_{offD2}I_{CES}$
$R_{onD1} = \frac{V_{CE1} - V_{CE0}}{I_{C1}}$	$Z = \frac{K_e E_t R_{ge}}{R_{ge} + R_{offDe}}$
$R_{sense1} = \frac{R_{onD1}}{100}$	$R_{offD3} = \frac{YR_{ceon} - X(R_{offD2} + R_{ceon})}{X - Y - ZR_{offD2}}$
$K_{12} = \frac{I_{tail}}{R_{sense1}I_n}$	$K_e = \frac{I_{C2}(1 - K_{12}R_{sense1})}{V_{GEa} - E_t}$
$I_{CES} = I_{C4} + \frac{K_e(V_{GEa} - V_{GEn})}{1 - K_{12}R_{sense1}}$	

As a result, the IGBT Alonso model is completely characterized from commercial datasheet and eventually few extra experimental measurements. Therefore, the model should give "performances" in switching that are close of the real device "performances".

TABLE A.2: Calculations for MOSFET

Parameter	Parameter
$R_{GE} = \frac{V_{GE}}{I_{GES}}$	$R_{onDe} = \frac{R_{ge}}{100}$
$R_{offDe} = 1000 R_{ge}$	$E_t = \frac{I_{C3} V_{GEa} - I_{C2} V_{GEb}}{I_{C3} - I_{C2}}$
$R_{ceon} = \frac{V_{CE2} - V_{CE0} - (R_{sense1} + R_{onD1}) I_{C2}}{(1 - K_{12} R_{sense1}) I_{C2}}$	$R_{onD3} = \frac{I_{C1}}{I_{C2} - I_{C1}} R_{ceon}$
$X = V_{CES} - V_{CE0} - (R_{sense1} + R_{onD1}) I_{CES}$	$Y = (1 - K_{12} R_{sense1}) R_{offD2} I_{CES}$
$Z = \frac{K_e E_t R_{ge}}{R_{ge} + R_{offDe}}$	$R_{offD3} = \frac{Y R_{ceon} - X(R_{offD2} + R_{ceon})}{X - Y - Z R_{offD2}}$
$K_e = \frac{I_{C2}(1 - K_{12} R_{sense1})}{V_{GEa} - E_t}$	

A.4 Generalized Alonso modelling Procedure

We propose a generic procedure as illustrated by Fig. A.5, valid for both IGBTs and MOSFETs, to configure the Alonso model based on the original procedure:

1. Parameter extraction, which is the process of extracting selected parameters that can usually be found in the IGBT datasheet.

2. Initialization code, which is an equation system that uses the extracted parameters as the inputs to obtain another set of parameters, which are used in the Transistor model.

3. Transistor model, which is based on the equivalent circuit of the IGBT device and provides the expected static and dynamic performance as outputs.

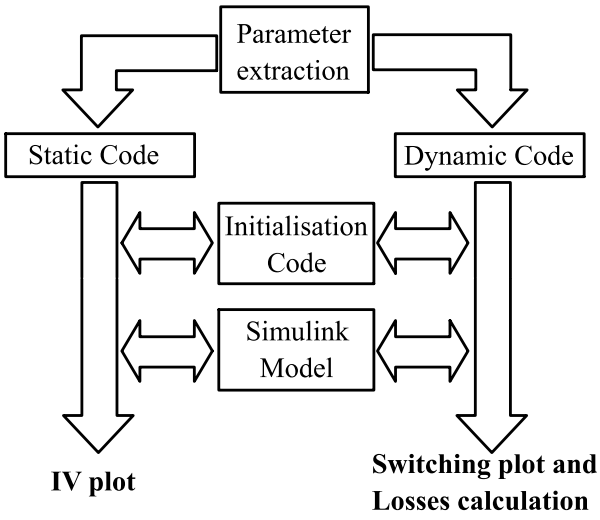


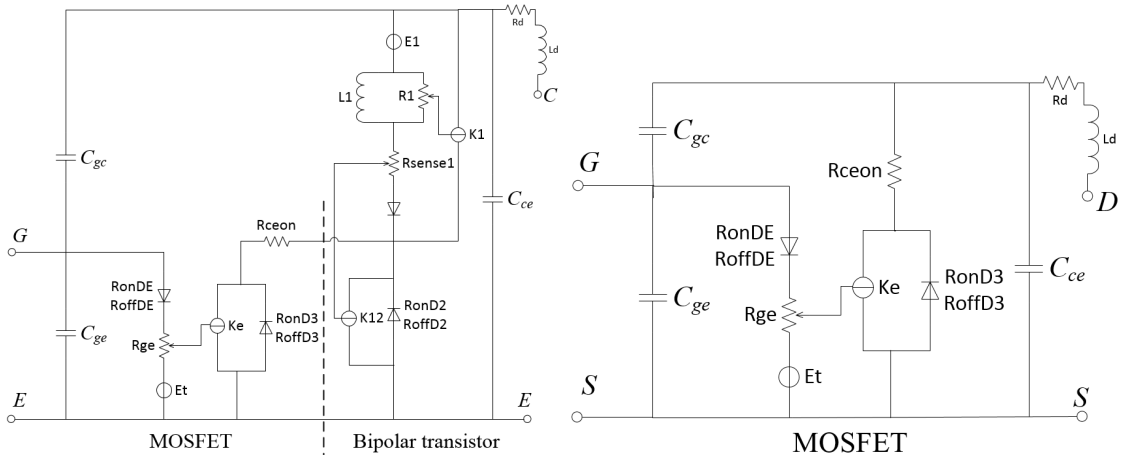
FIGURE A.5: Flow diagram of the generalized Alonso modelling algorithm

4. Static and dynamic code, which is the code where terminal voltages and other conditions are set and displayed. It also runs the Initialization code and the Transistor model.

In this paper, we use the *SimPowerSystems* toolbox of the *Simulink* environment to build the Alonso model and the full procedure is thus run under MATLAB.

A.4.1 Simulink Modelling

To simulate the models in a more realistic environment, we introduce a parasitic resistance R_d and a parasitic inductance L_d in the power side of the switching device [81]. The original IGBT circuit is shown in Fig. A.6a, with the left half of the circuit represents the main MOSFET part of the switching device and the right half represents the bipolarity interface, which is virtually attached to the Drain and the Source terminals of such device. To extend the framework to model both IGBT and MOSFET device, we demonstrate that the whole Alonso model can be used to model an IGBT (Fig. A.6a), while the left part only of the Alonso model can be used to model a MOSFET (Fig. A.6b). Further details on the implementation are provided in the Appendix.



(A) IGBT (B) MOSFET

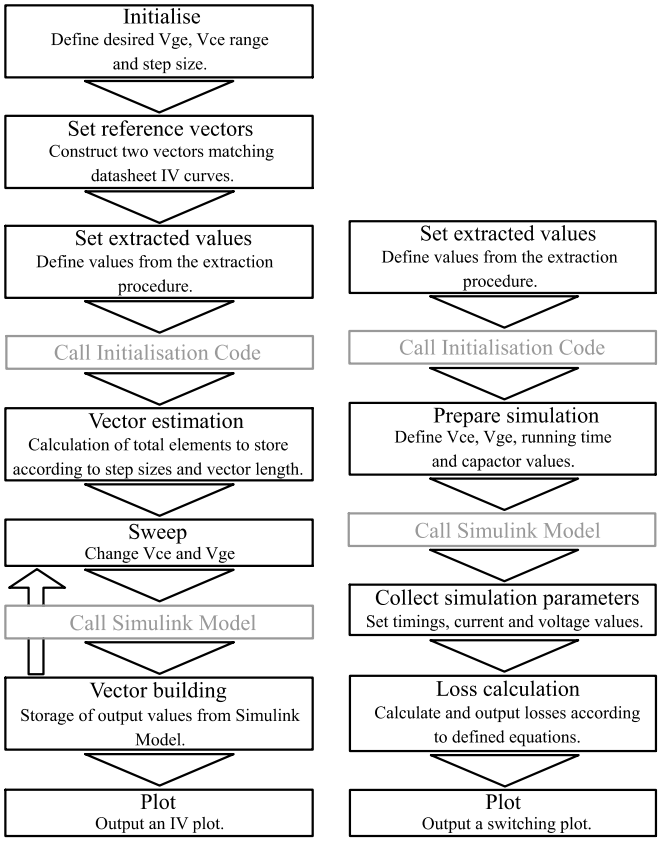
FIGURE A.6: Simulink models

A.4.2 Static Characterization

The procedure for the static characterization uses the *Static Code* that utilizes the *Initialisation Code* and the *Simulink Model* to simulate the static characteristics of a selected transistor. The flow diagram of the static I-V estimation is provided in Fig. A.7a. First, the desired V_{GE} and V_{CE}, including the step size for the test are selected. Second, the reference vectors are outlined so that the software draws the curve according to the datasheet I-V curve. This step is only for comparison purposes and has no impact on the estimation. Third, the *Initialisation Code* is called once and then the *Simulink Model* is called several times to produce output of an I-V plot. This procedure is implemented for either IGBT or MOSFET.

A.4.3 Dynamic Characterization

The dynamic characterization follows the procedure described in the flow diagram shown in Fig. A.7b, implemented in the *Dynamic Code* that allows driving the model with a square signal V_{GS} in the *Simulink* model. The capacitance of each



(A) *Static Code.* (B) *Dynamic Code.*

FIGURE A.7: Flow diagrams of the complete procedure to model IGBT or MOSFET.

capacitor is set from datasheet values using the following definitions:

$$C_{gc} = C_{res}, \tag{A.1}$$

$$C_{ge} = C_{ies} - C_{res}, \tag{A.2}$$

$$C_{ce} = C_{oes} - C_{res}. \tag{A.3}$$

The controlled voltage source V_{GS} triggers the transistor with specific timing, and a RL circuit, with values of R_d, L_d and V_{GS} given in the datasheet, is connected in series at the collector terminal. Secondly, the *Initialisation Code* is executed and other conditions are set to prepare for the simulation. Finally, the *Simulink Model* is then called and the simulation parameters are collected. A loss calculation is provided at the end of the program, and the result is plotted.

A.5 Static Evaluation

In this section, the proposed modelling framework is tested and verified. Parameters of the Alonso model are identified for the following devices: a Si MOSFET IRF640 from Vishay [82] rated at 200V/18A, a SiC MOSFET C2M1000170D from Cree [83] rated at 1700V/5A, a second SiC MOSFET SCT2080KE from ROHM [84] rated at 1200V/40A, and a Si IGBT RGT16NS65D also from ROHM [85] rated 650V/8A. Static evaluation is performed and reported in this section. The modelling procedure proposed in Section A.4, from parameter extraction to the I-V plot, is applied for each device. The extracted models from the commercial devices are then compared to the corresponding datasheets and the experimental measurements in order to verify the accuracy of the proposed modelling framework.

A.5.1 Experiment for the Device Measurement

The I-V characterisation of the IRF640, C2M1000170D, SCT2080KE and RGT16NS65D devices are carried out at a constant room temperature of 22°C, using the Keysight B1505A Power Device Analyser. The 500 A ultra-high current 3-pin inline package socket module was used to connect the devices to the analyzer. Short external terminals pins are directly soldered onto the RGT16NS65D device to make it compatible with the socket module and a particular attention has been taken to minimise additional parasitics. Calibration was also performed on the B1505A to minimise cable parasitics. The DUTs temperature was monitored using a thermocouple throughout the experiment. The V_{DS} of the devices was pulsed to limit self-heating. The experiment parameters are set corresponding to the devices datasheet.

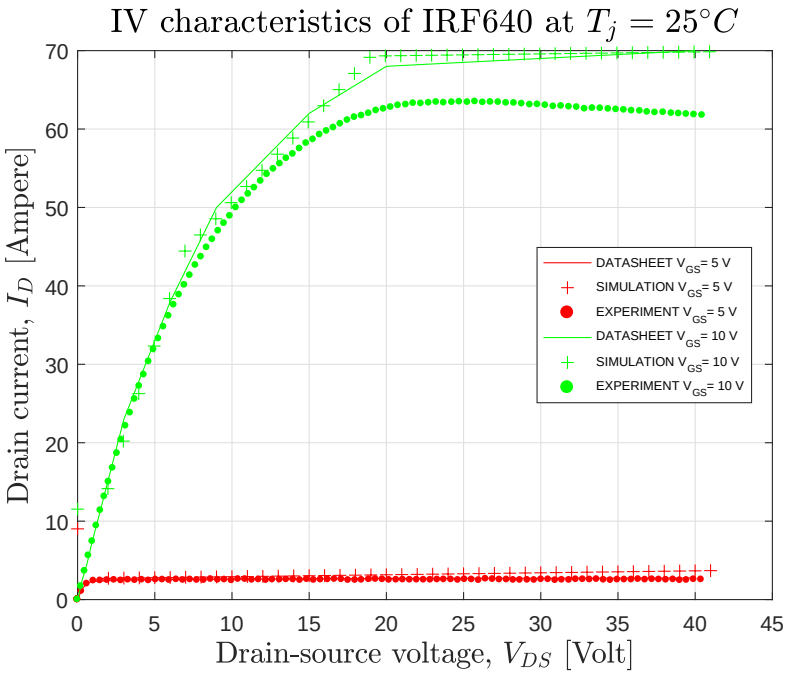


FIGURE A.8: I-V curve for the Si MOSFET IRF640 with Static Code

A.5.1.1 Si MOSFET IRF640

Set V_{GS} = 5 V and 10 V; V_{DS} = sweep 0 to 40 V, step 450 mV, pulse width 20 μs and period of 1 s, with measurement time = 2 μs. The I-V curves are shown in Fig. A.8.

A.5.1.2 SiC MOSFET C2M1000170D

Set V_{GS} = 12 V and 16 V; V_{DS} = sweep 0 to 40 V, step 450 mV, pulse width 150 μs and period of 1 s, with measurement time = 10 μs. The I-V curves are respectively marked and shown in Fig. A.9.

A.5.1.3 SiC MOSFET SCT2080KE

Set V_{GS} = 12 V and 16 V; V_{DS} = sweep 0 to 40 V, step 450 mV, pulse width 150 μs and period of 2 s, with measurement time = 10 μs. The I-V curves are shown in Fig. A.10.

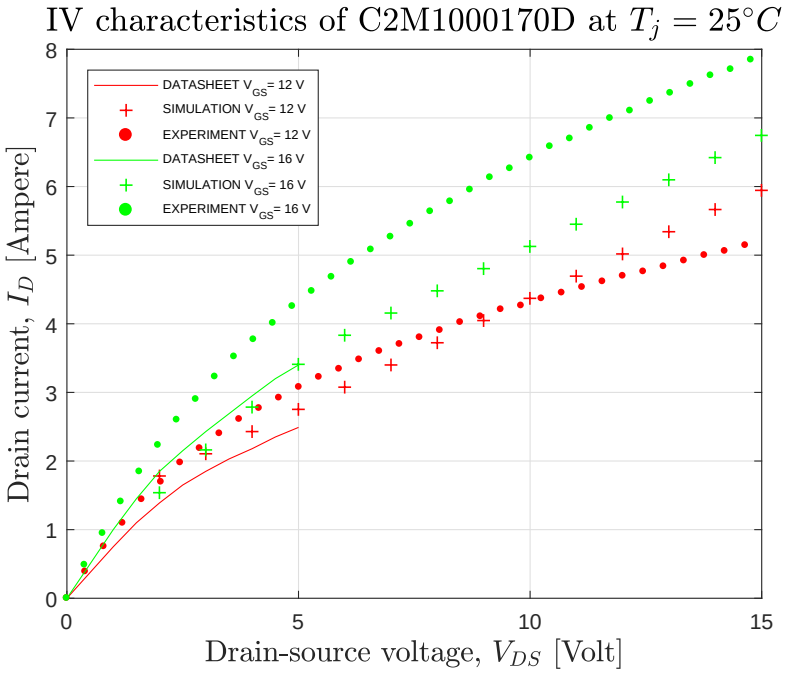


FIGURE A.9: I-V curve for the SiC MOSFET C2M1000170D with Static Code

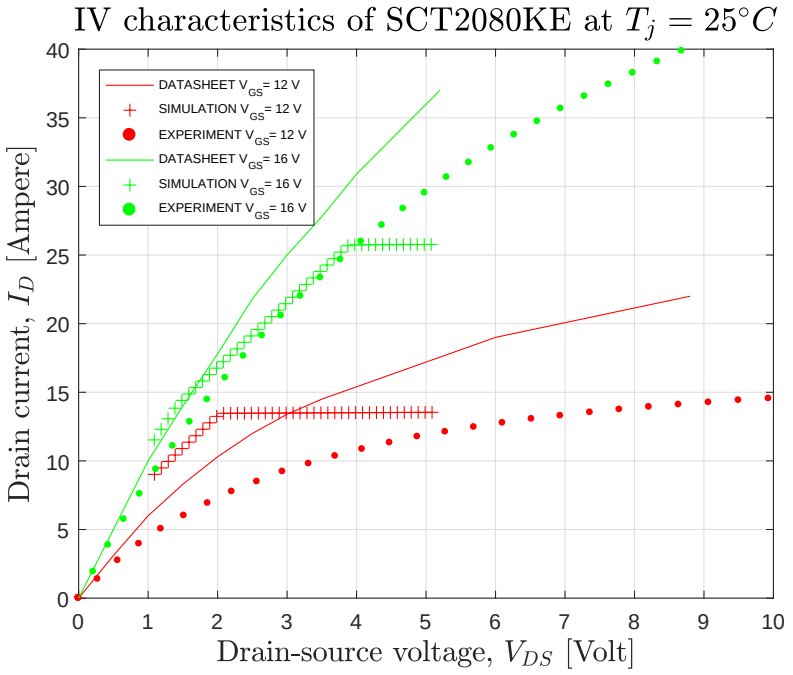


FIGURE A.10: I-V curve for the SiC MOSFET SCT2080KE with Static Code

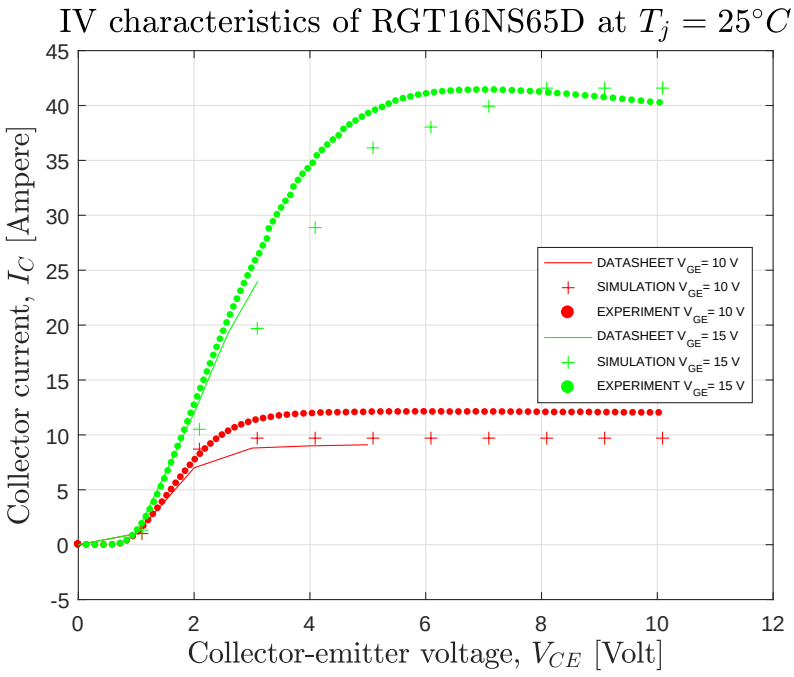


FIGURE A.11: I-V curve for the IGBT RGT16NS65D with Static Code

A.5.1.4 Si IGBT RGT16NS65D

Set V_{GS} = 10 V and 15 V; V_{DS} = sweep 0 to 10 V, step 150 mV, pulse width 100 μs and period of 1 s, with measurement time = 20 μs. The I-V curves are shown in Fig. A.11. As the RGT16NS65D TO-263S package is not compatible with the 500 A ultra-high current 3-pin inline package socket module, three terminals has been soldered onto the Gate, Collector and Emitter extending the original terminals and creating a tab (Collector) terminal.

A.5.2 Static Evaluation Analysis

The I-V plots show a relatively good fit between the model, the datasheet and the experimental measurement for each device. Particularly for the Si MOSFET IRF640, as seen in Fig. A.8, the three plots, each for the same V_{GE}, are very close to each other, which indicates that the proposed generalized Alonso method performs very well in modelling Si MOSFET. We can also make the same observation for

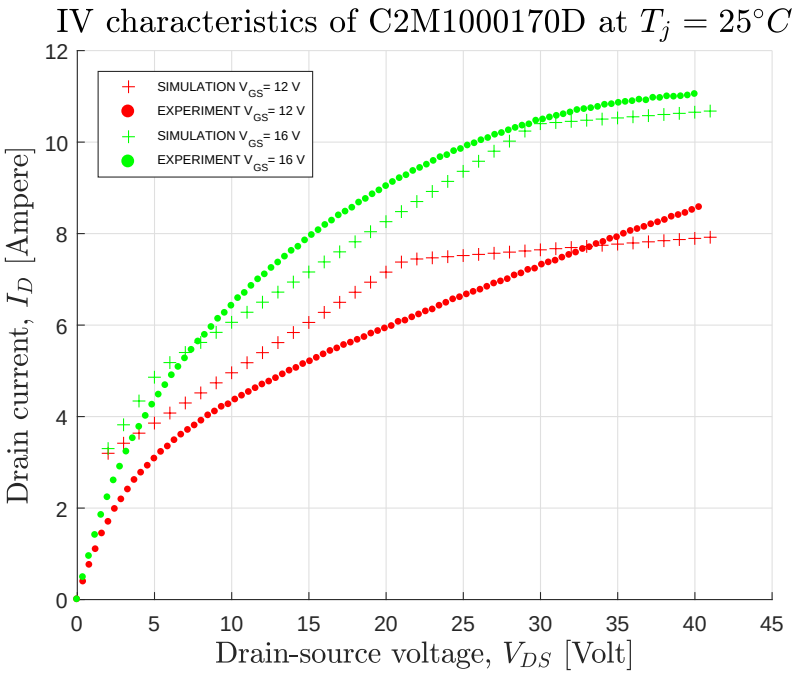


FIGURE A.12: I-V curve for the SiC MOSFET with *Static Code* using experimental-based values

the Si IGBT as shown in Fig A.11. Concerning the SiC MOSFETs (see Figures A.9 and A.10), while the curves still fit fairly well the datasheet curves, it is not as accurate as the Si MOSFET. However, this is not fully due to the modelling algorithm, as explained next.

It is known that for new devices (such as GaN and SiC) the datasheets are still frequently updated due to the continuous development and characterization of these, which have not reached a mature stage [86]; for instance, regarding the case of the SiC MOSFET C2M1000170D, there are several versions of the datasheets [83, 87].

A further comparison shown in Fig. A.12 demonstrated that concerning the SiC MOSFET C2M1000170D a better matching of the I-V curves between the model and the measurement can be achieved if the model is constructed using parameters extracted from the experimental curves rather than from the datasheet (compare

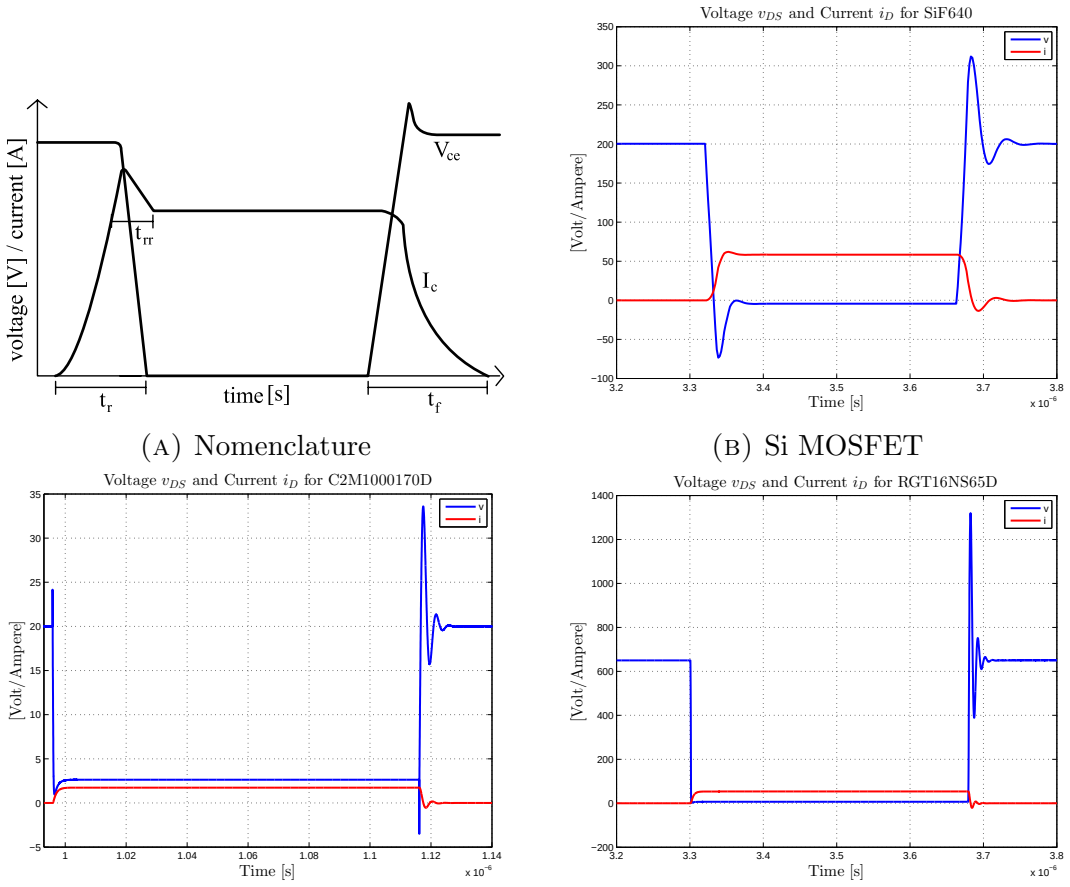
this with Fig. A.9). Hence, it is important to remark that if reliable experimental measurement data are available, it is recommended to take the extraction points from experimental curves rather than from the datasheet. In either case, the advantage of this proposed framework is that if reliable parameters data are available, either from datasheet or from measurement, a quite accurate model of Si/SiC IGBT or MOSFET device can be constructed in quite a simple procedure.

A.6 Dynamic Evaluation Analysis

In this section, the evaluation of the switching performances, particularly in terms of switching losses, is performed following the steps described in Subsection A.4.3. The dynamic evaluation presented in this section is not yet of a great depth as this is still an ongoing research. However, the authors feel that including this section is useful for the sake of completeness and to help readers to see the capability of the proposed modelling framework.

To evaluate the switching losses, we use the simple formulae proposed in [88] that has been originally proposed to calculate the switching losses for inverters, and for which we assume the validity for the simple switching mode used to evaluate the switching losses. The switching parameters are given for each DUT in Table A.3 and have been measured from the standard nomenclature given in Fig. A.13a.

The result of the analysis for the Si MOSFET IRF640 is shown in Fig. A.13b, showing the resonant effect caused by the parasitic inductance. It can be seen in this figure that the settling time of the voltage is 1.115×10^{-6} seconds. Corresponding plot for the other transistors are shown in Fig. A.13c for the SiC MOSFET C2M1000170D and in Fig. A.13d for the IGBT RGT16NS65D, with the respective values of parameters provided in Table A.4.



(A) Nomenclature (B) Si MOSFET

(C) SiC MOSFET (D) IGBT

FIGURE A.13: Switching outputs with Dynamic Code.

Following the general nomenclature, the forward losses of the transistor and the free-wheeling diode are respectively calculated as [88]:

$$P_t = \left(\frac{1}{8} + \frac{M}{3\pi}\right)\frac{V_{CEN} - V_{CO}}{I_{CN}} I_{CM}^2 + \left(\frac{1}{2\pi} + \frac{M}{8}\cos\theta\right) V_{CO} I_{CM}, \tag{A.4}$$

and

$$P_d = \left(\frac{1}{8} - \frac{M}{3\pi}\right)\frac{V_{FN} - V_{FO}}{I_{CN}} I_{CM}^2 + \left(\frac{1}{2\pi} - \frac{M}{8}\cos\theta\right) V_{FO} I_{CM}, \tag{A.5}$$

respectively. Also, the turn-on and recovery losses are given by

$$P_{on} = \frac{1}{8} V_{cc} t_{rN} \frac{I_{CM}^2}{I_{CN}} F_s, \tag{A.6}$$

TABLE A.3: Specifications of the switching parameters.

Symbol	Si MOSFET IRF640	SiC MOSFET C2M1000170D	IGBT RGT16NS65D	Units
F_s	50	50	50	[kHz]
R_{on}	2	2	0.1	[Ω]
V_{cc}	1.7	200	650	[kV]
I_{CN}	20	18	8	[A]
V_{CEN}	$I_{CN}R_{on}$			[V]
V_{FN}	3.3	2	1.4	[V]
Q_{rrN}	24	3.4	24	[nJ]
t_{rN}	10.5	10.5	10.5	[ns]
t_{rrN}	20	20	20	[ns]
t_{fN}	60	60	60	[ns]
$\cos(\theta)$	0.8			-
M	1			-
V_{CO}	0	0	0	[V]
I_{CM}	3.5	18	16	[A]
V_{FO}	0.8	3	6	[V]

TABLE A.4: Parameters for dynamic simulation.

Parameter	Si MOSFET IRF640	SiC MOSFET C2M1000170D	IGBT RGT16NS65D	Units
R_d	3.5	10	12	[Ω]
L_d	10	10	10	[nH]
V_{GS}	20	14	20	[V]

$$P_{rr} = F_s V_{cc}\Big[\Big(0.28+\frac{0.38}{\pi}\frac{I_{CM}}{I_{CN}}+0.015\Big(\frac{I_{CM}}{I_{CN}}\Big)^2\Big)Q_{rrN}+\Big(\frac{0.8}{\pi}+0.05\frac{I_{CM}}{I_{CN}}\Big)I_{CM}t_{rrN}\Big], \tag{A.7}$$

so that

$$P_{onrr} = P_{on} + P_{rr}. \tag{A.8}$$

Last, but not least, the turn-off losses are defined by

$$P_{off} = V_{cc}I_{CM}t_{fN}F_s\Big(\frac{1}{3\pi}+\frac{1}{24}\frac{I_{CM}}{I_{CN}}\Big). \tag{A.9}$$

Hence, the total losses, which is the sum of all the above individual losses, can be expressed as

$$P_{total} = P_t + P_d + P_{onrr} + P_{off}, \tag{A.10}$$

$$E_{total} = \frac{1}{F_s}P_{total}, \tag{A.11}$$

Table A.5: Estimated losses

Symbol	Si MOSFET IRF640	SiC MOSFET C2M1000170D	IGBT RGT16NS65D	Units
P_t	149.75	5.662	5.9	[W]
P_d	2.85	0.1946	2.9	[W]
P_{onrec}	1.35	2.2509	5.5	[W]
P_{off}	1.6	2.0241	5.9	[W]
P_{total}	155.5	10.1316	20	[W]
E_{total}	3.1×10^{-3}	202.6×10^{-6}	404×10^{-6}	[J]

either for power or energy form. Substituting the value of the simulation parameters given in Table A.3, the estimated losses for each device are obtained as listed in Table A.5.

A.7 Conclusions

A modelling framework for MOSFETs and IGBTs, which is the generalization of the Alonso model for IGBT, has been proposed. To validate the framework, it is tested using a set of Si and SiC devices. Static characterization is then performed based on the constructed model for each device, achieving reasonably similar characteristics to that reported in the datasheets as well as real measurements. Moreover, a complementary code provided the dynamic performance for each of the devices, including losses calculation. Simulations of the SiC MOSFET show a faster switching response and less losses than those of the Si MOSFET and IGBT, including the IGBT shown in [78], which is logical since the device under test is a SiC MOSFET and it is expected to be faster. Experimental work provided a comparison reference for the Alonso method with the two types of devices. In a nutshell, the model of any device (Si MOSFET, SiC MOSFET or IGBT) constructed using the proposed extended modelling framework exhibits comparable characteristics as the information provided in the datasheet and the measurement with the real device, both in terms of performance as well as the losses based on the transistor settings such as switching frequency, current and voltage. Particularly

for the SiC device, although precision of the results is acceptable, improvement and optimisation algorithms can be implemented in the code to acquire a more precise fitting.

Appendix B

Datasheets Extract

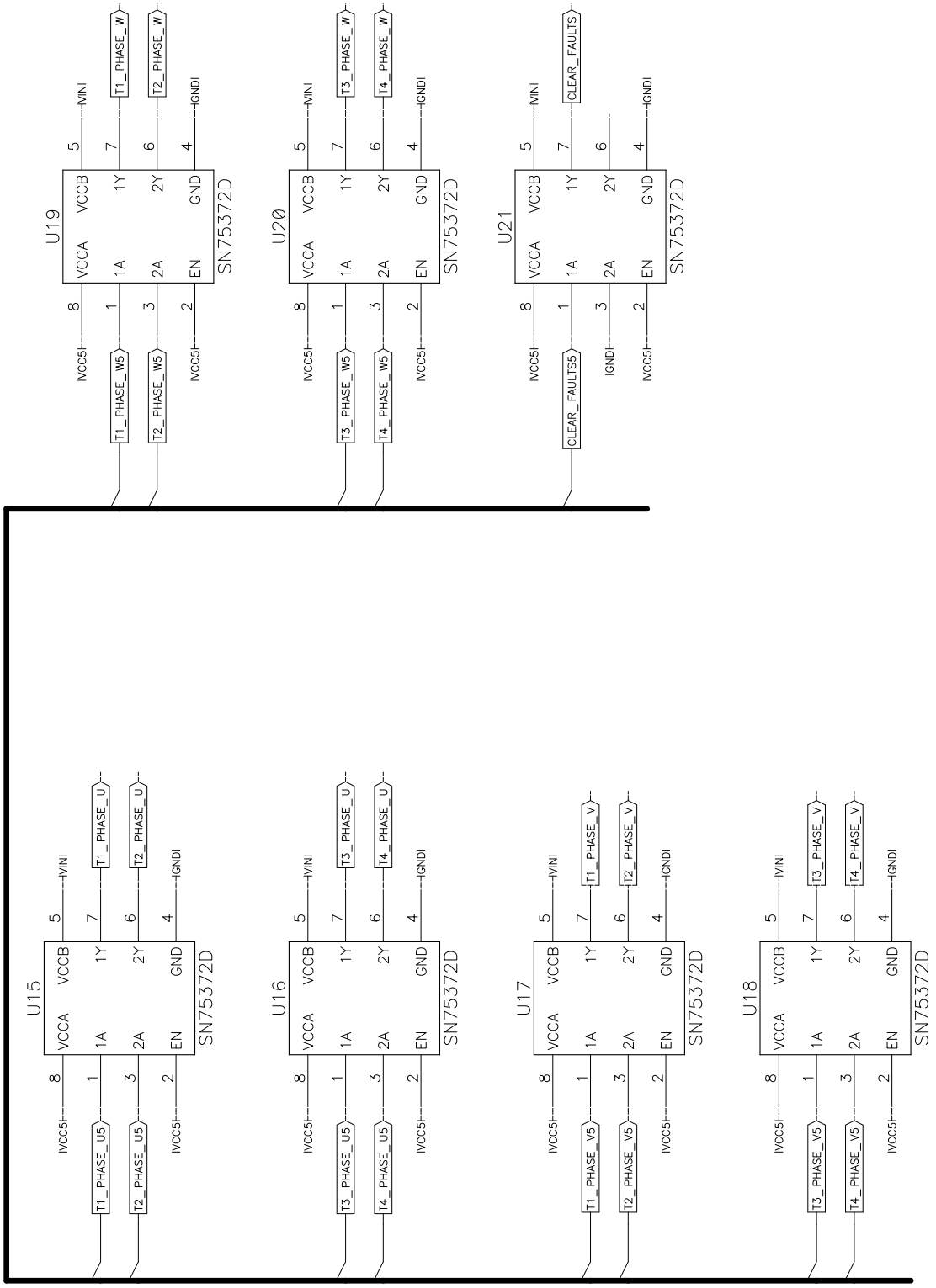


FIGURE B.1: Adaptor board: a) Output driver [89].

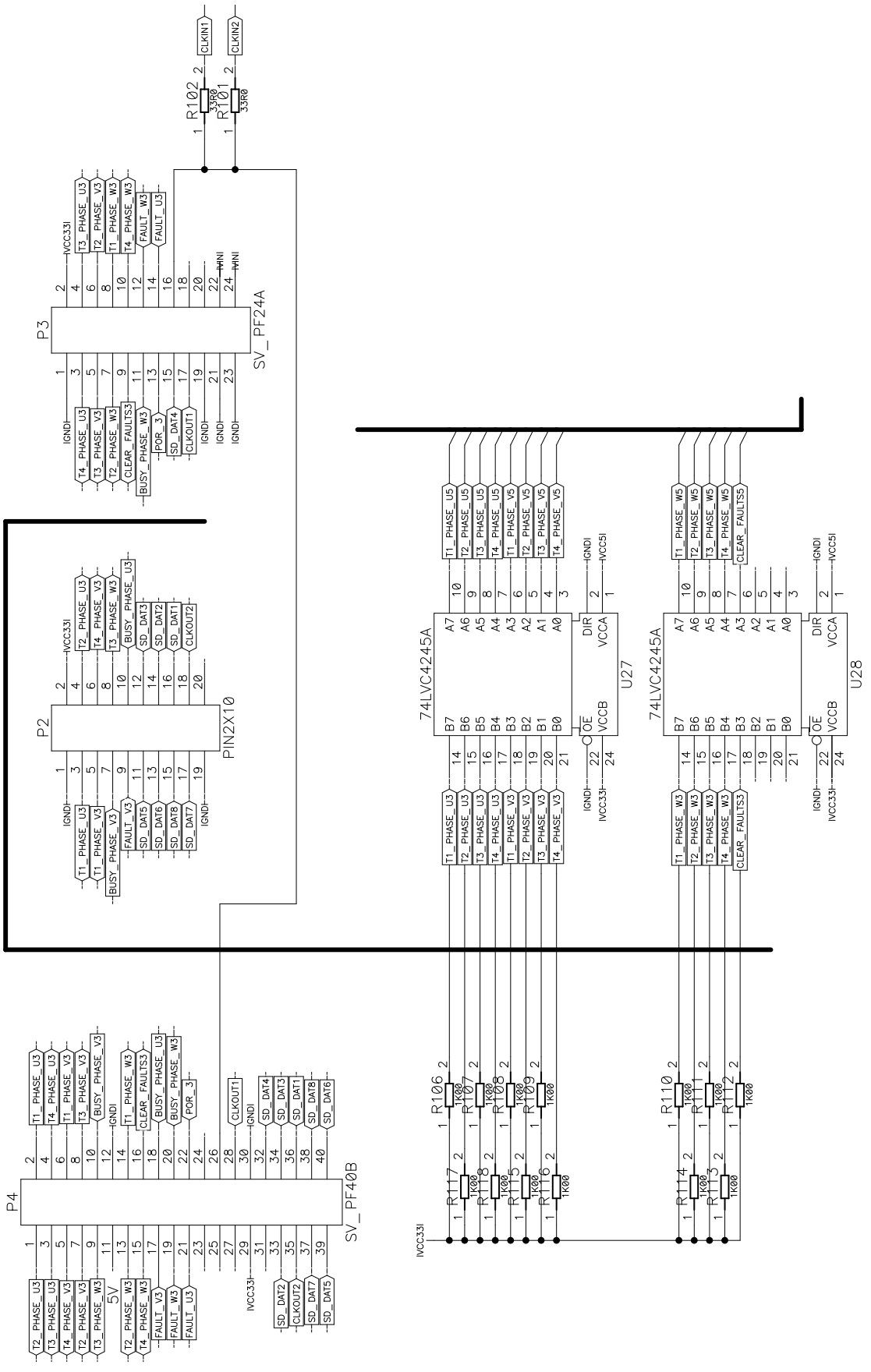


FIGURE B.2: Adaptor board: b) FPGA interface [89].

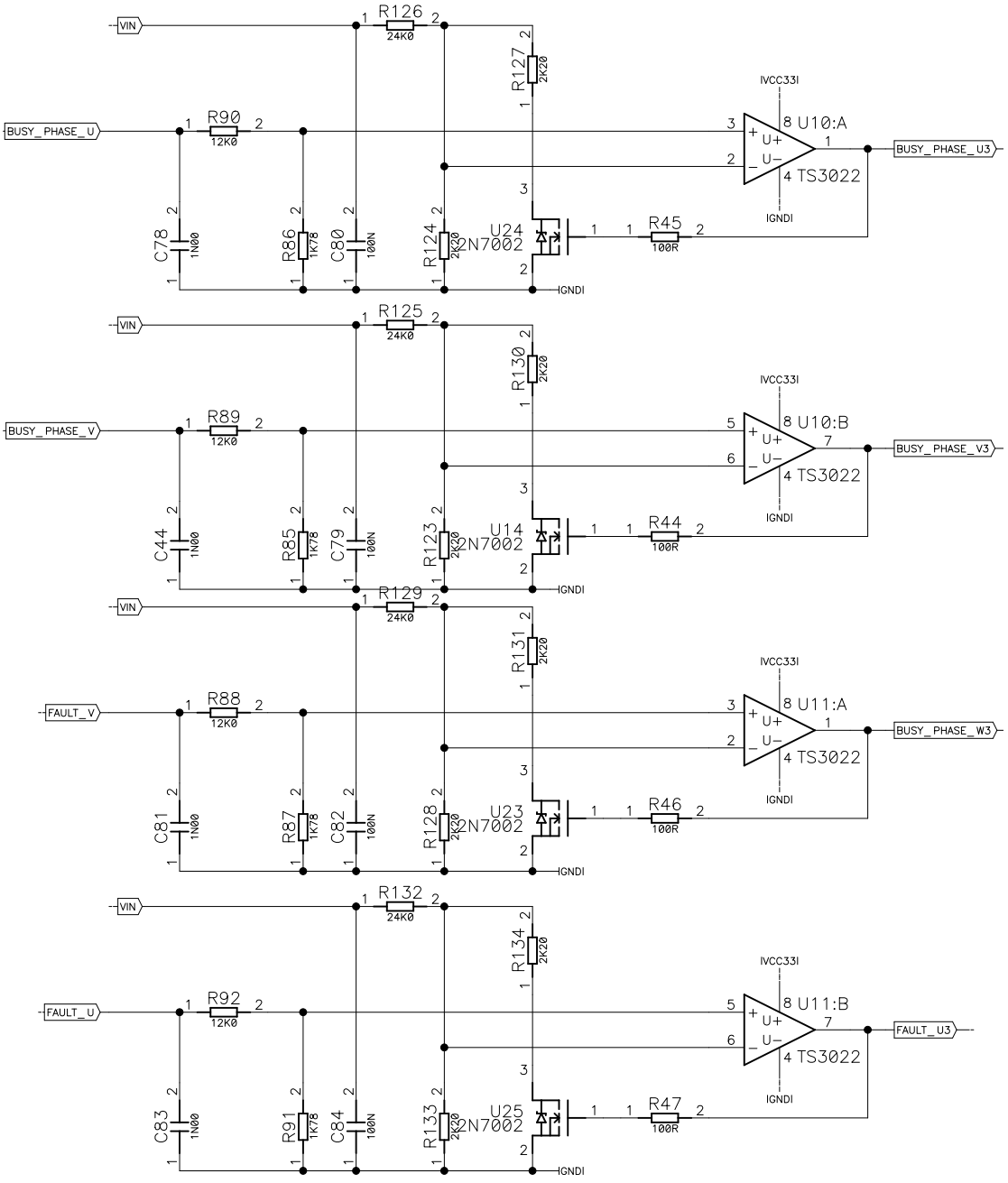


FIGURE B.3: Adaptor board: c) Signal conditioning (part 1) [89].

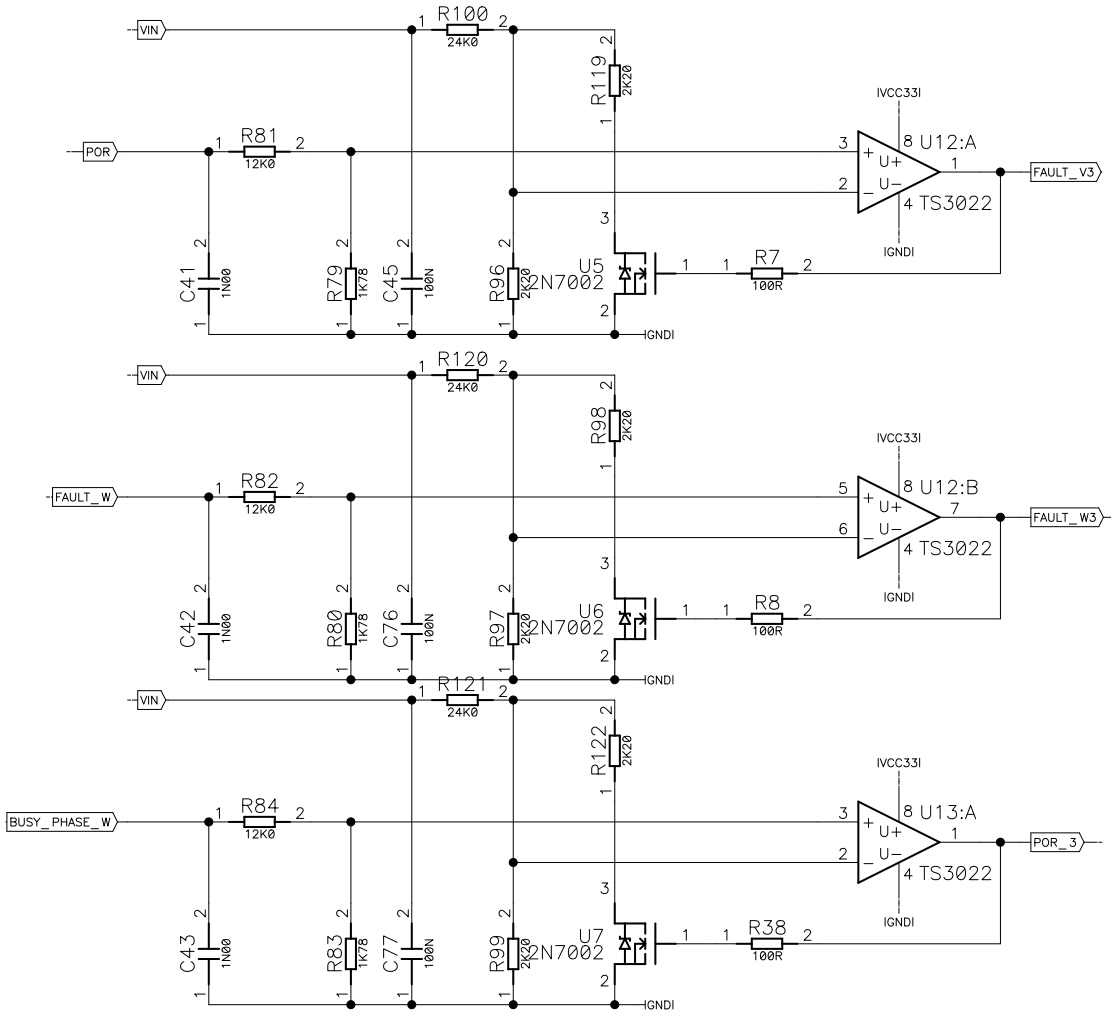


FIGURE B.4: Adaptor board: c) Signal conditioning (part 2) [89].

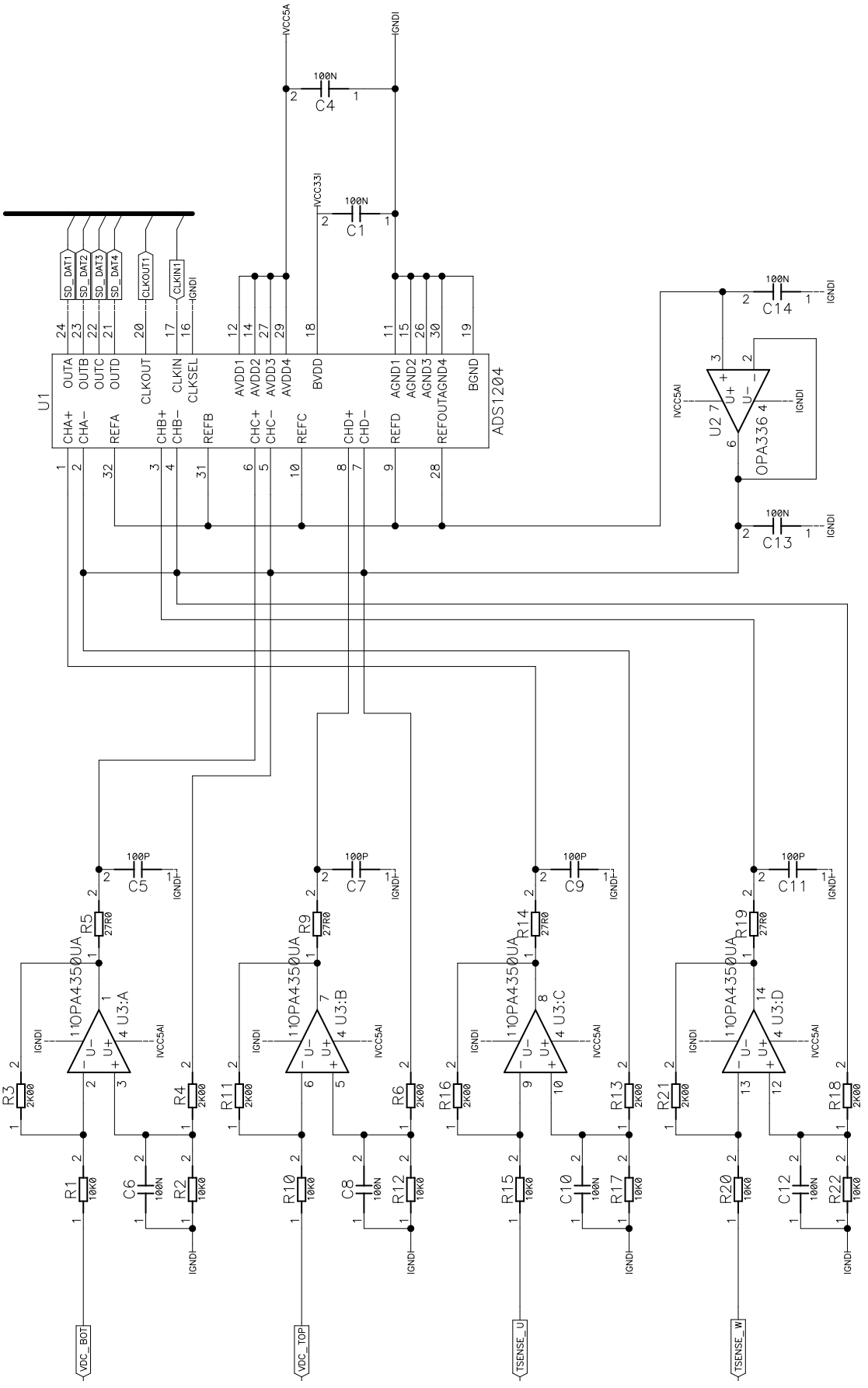


FIGURE B.5: Adaptor board: d) ADC (part 1) [89].

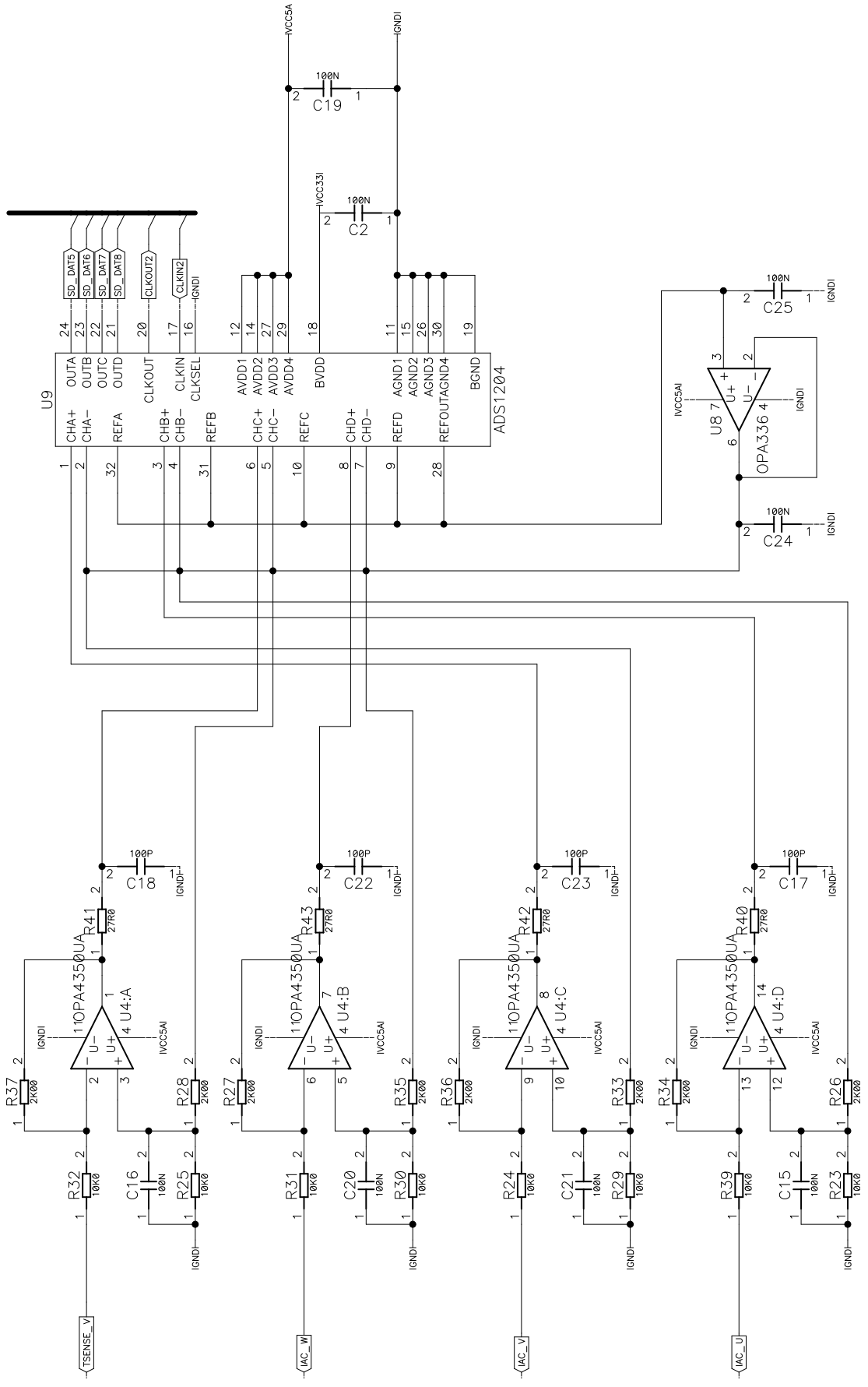


FIGURE B.6: Adaptor board: d) ADC (part 2) [89].

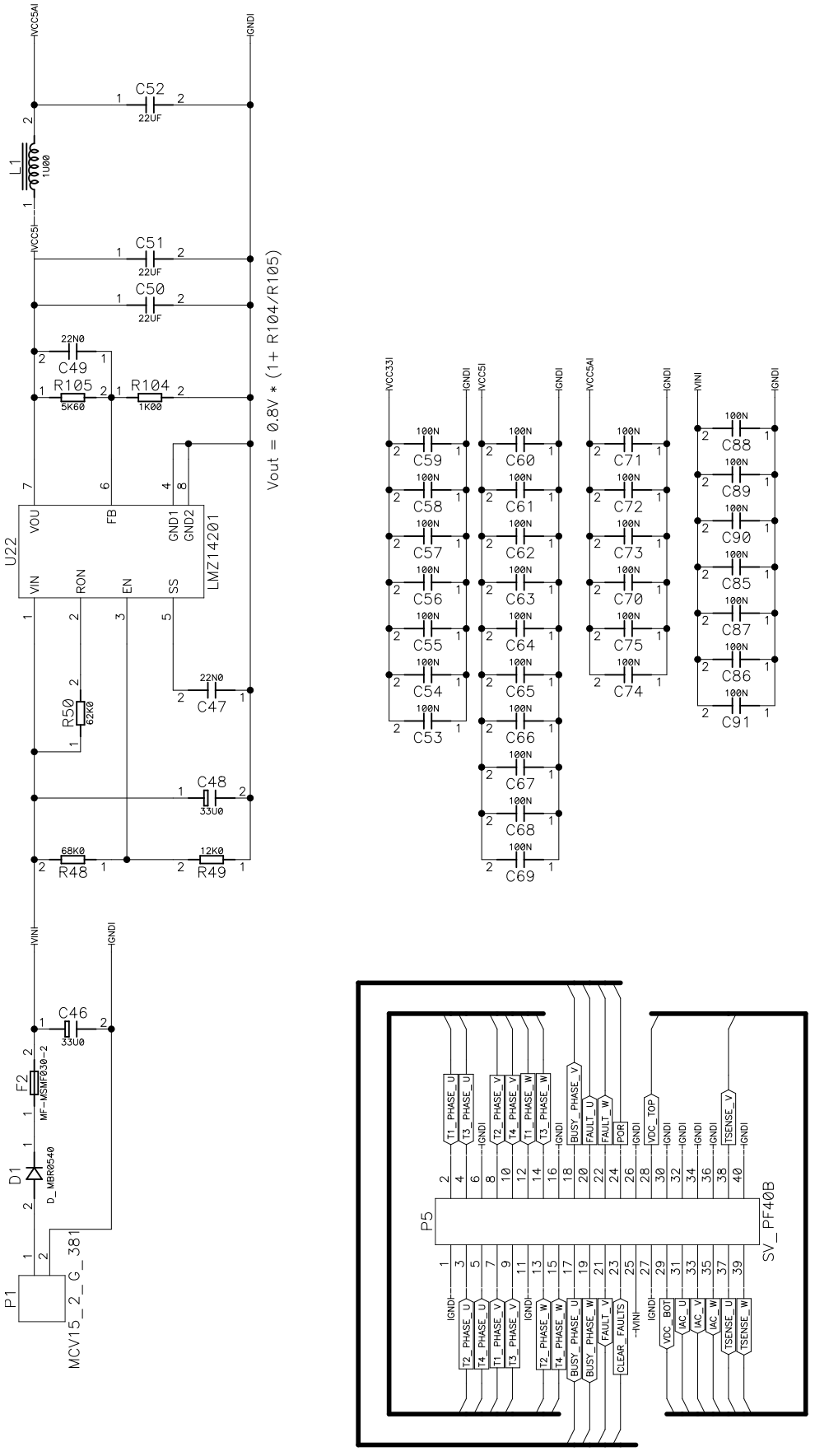


FIGURE B.7: Adaptor board: e) Power supply [89].

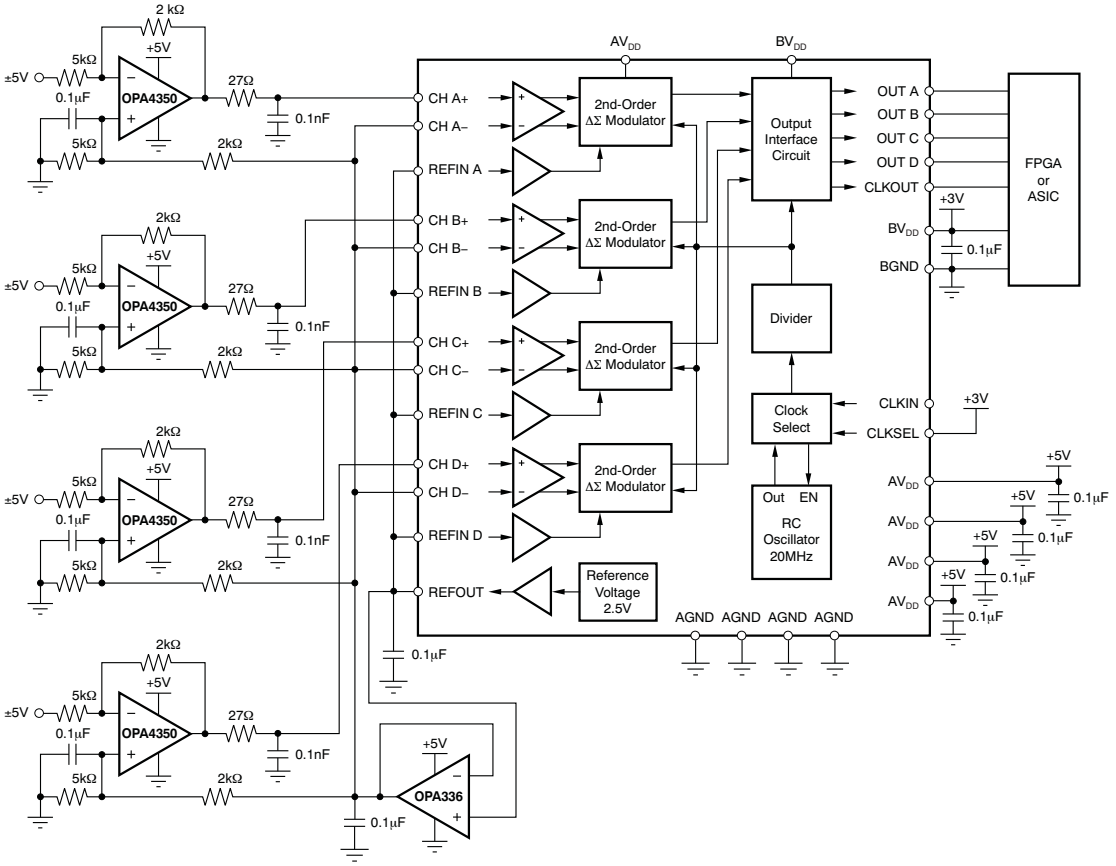


FIGURE B.8: Typical application circuit with the ADS1204 connected to an FPGA (single-ended connection) [59].

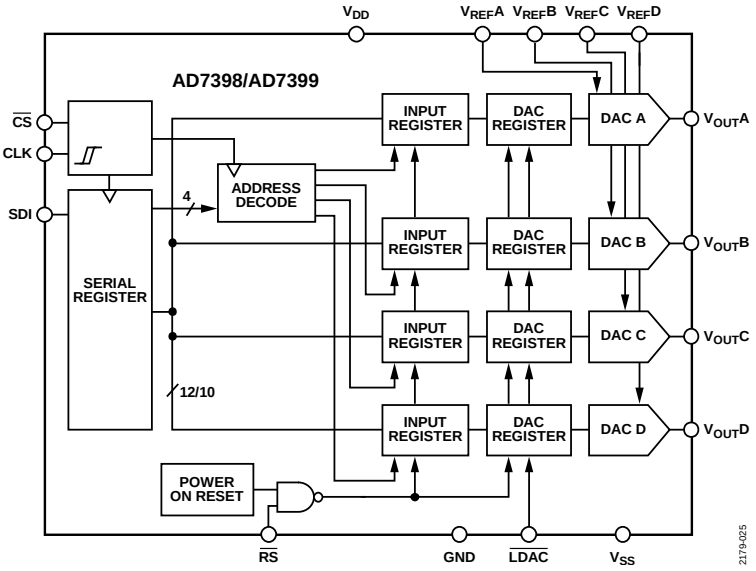


FIGURE B.9: Simplified block diagram of DAC AD7398.

Appendix C

DPC Experiments at Higher Power

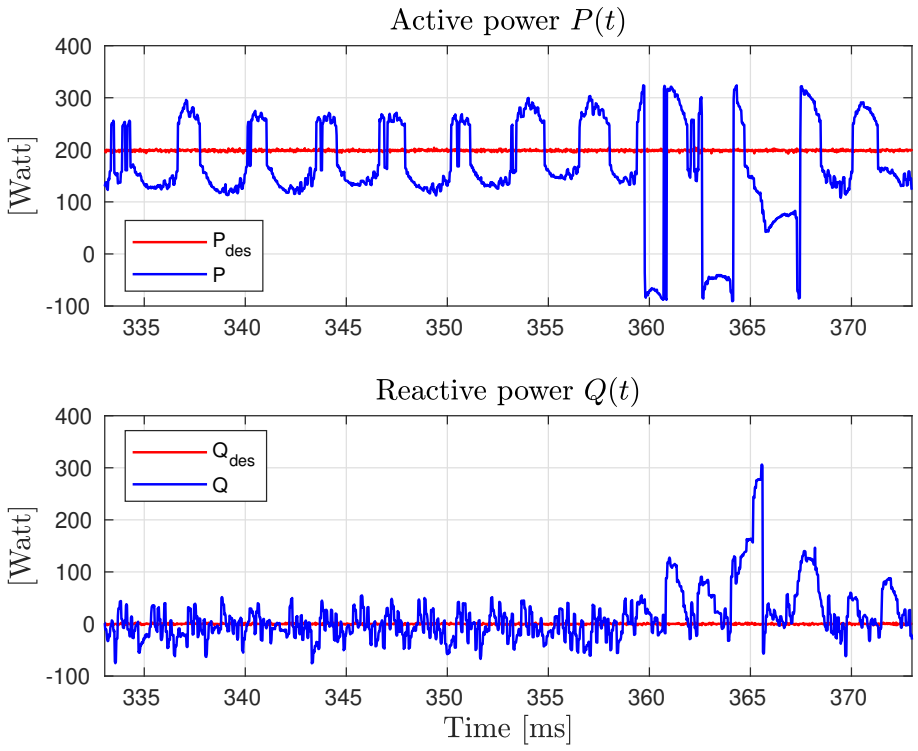


FIGURE C.1: Experiment. Powers at $P_{des} = 200$[W] and hard vector Table 4.5

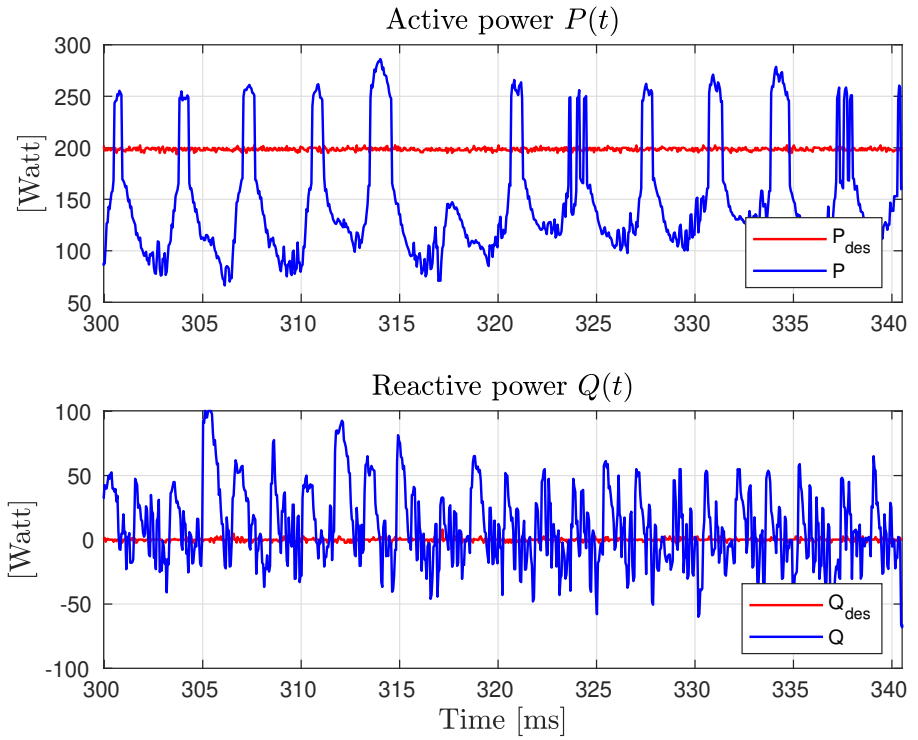


FIGURE C.2: Experiment. Powers at $P_{des} = 200$[W] and soft vector Table 5.7

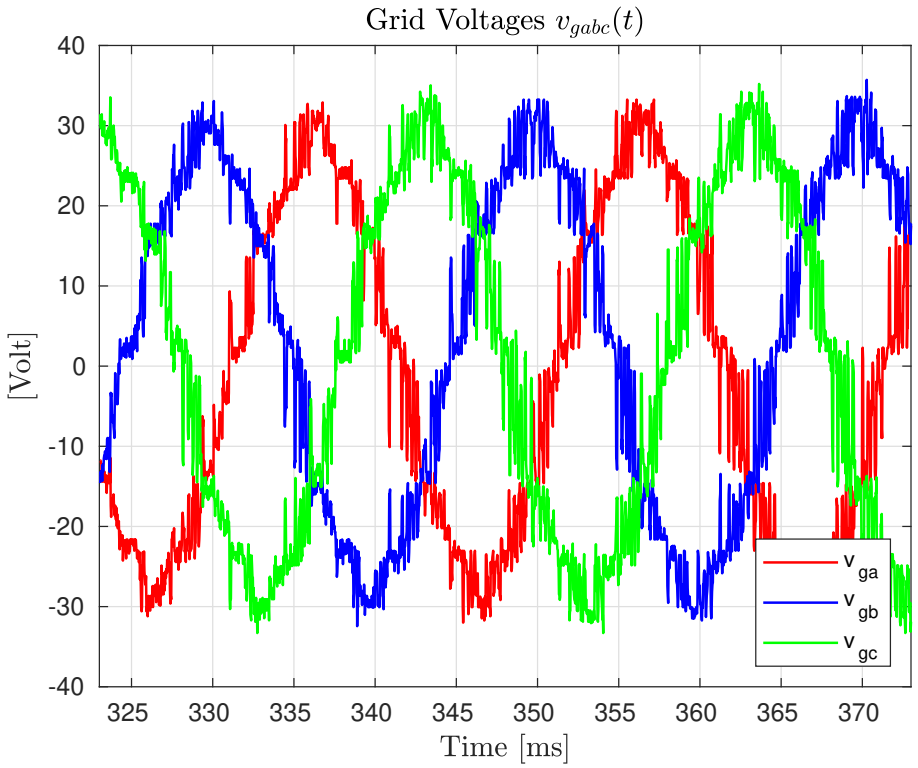


FIGURE C.3: Experiment. Grid voltages at $P_{des} = 200$[W] and hard vector Table 4.5

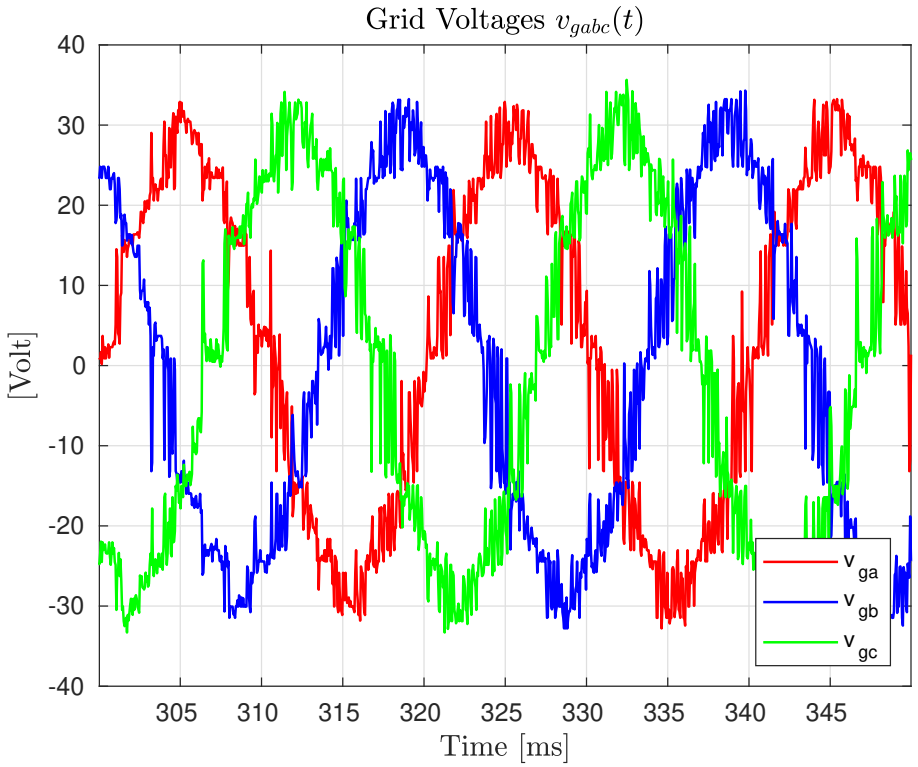


FIGURE C.4: Experiment. Grid voltages at $P_{des} = 200$[W] and soft vector Table 5.7

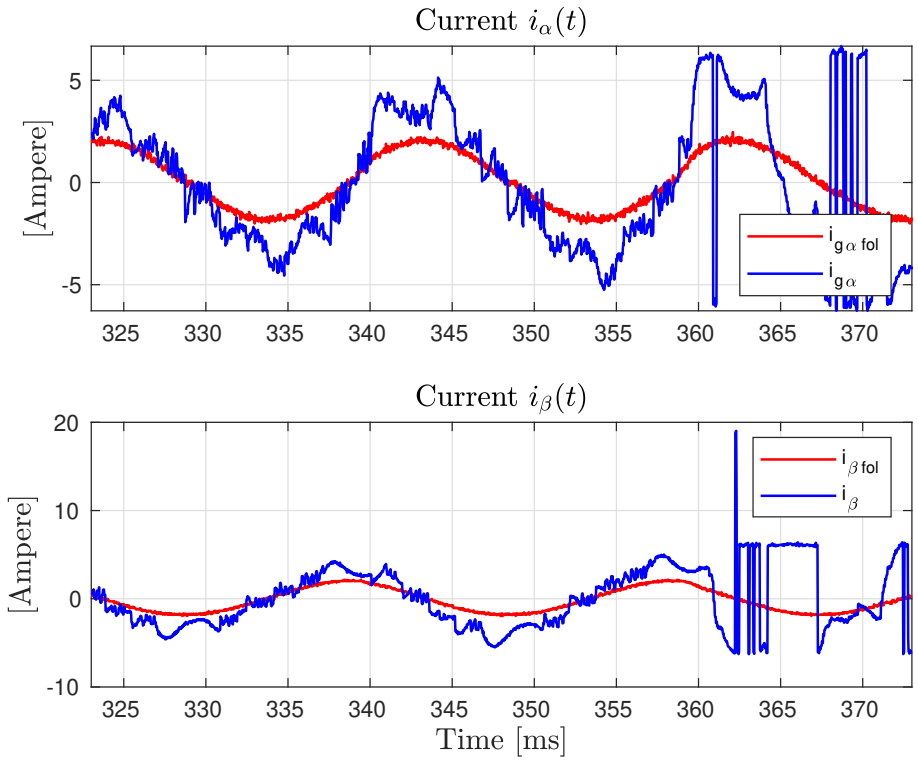


FIGURE C.5: Experiment. Powers at $P_{des} = 200$[W] and hard vector Table 4.5

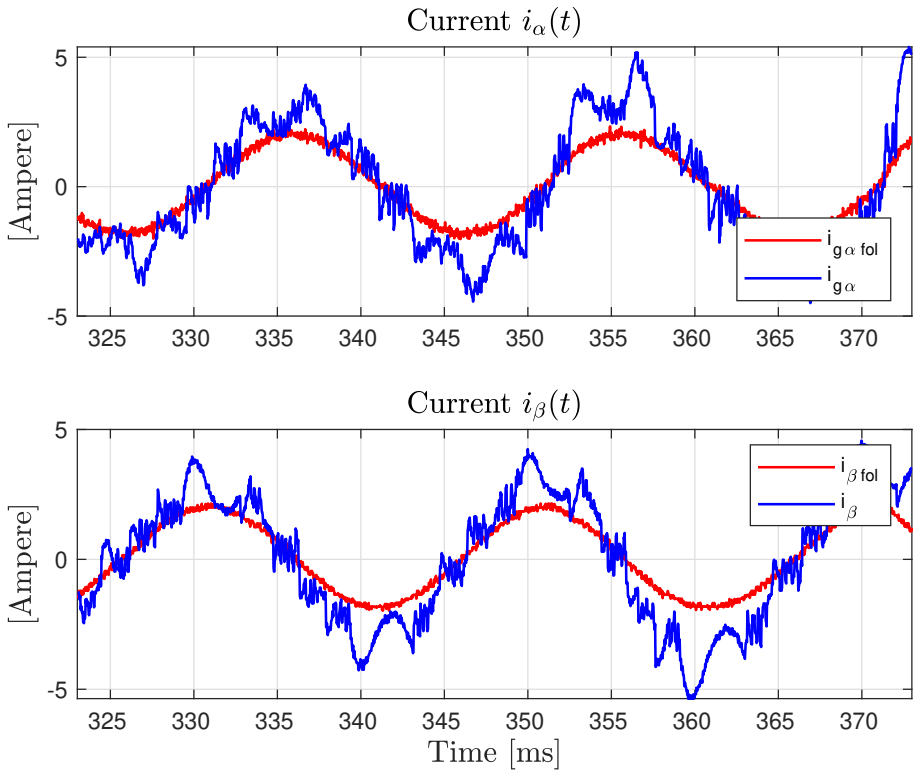


FIGURE C.6: Experiment. Currents at $P_{des} = 200$[W] and soft vector Table 5.7

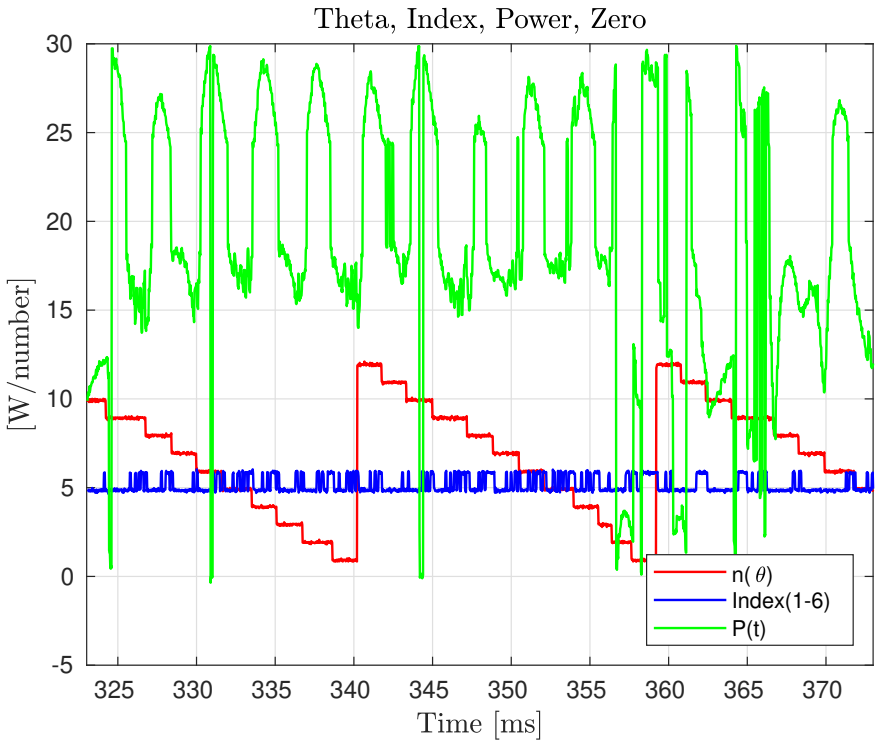


FIGURE C.7: Experiment. Active power, *index*, and $n(\theta)$ with $P_{des} = 200$[W] and hard vector Table 4.5

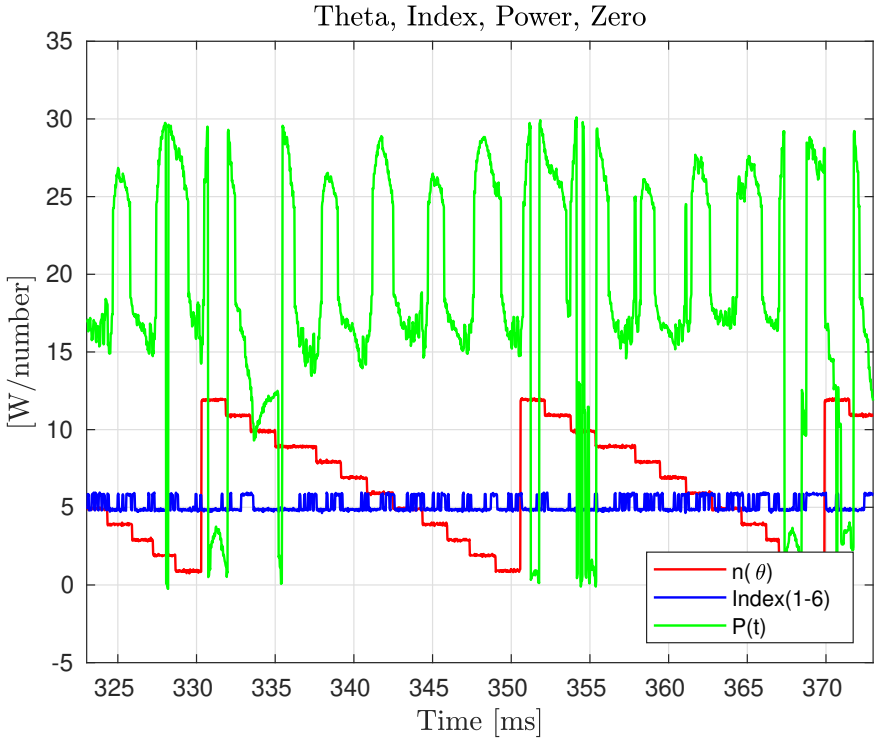


FIGURE C.8: Experiment. Active power, *index*, and $n(\theta)$ with $P_{des} = 200$[W] and soft vector Table 5.7

References

[1] S. A. Sejas, M. Cai, A. Hu, G. A. Meehl, W. Washington, and P. C. Taylor, "Individual Feedback Contributions to the Seasonality of Surface Warming," *J. Clim.*, no. 14, pp. 5653–5669, Jul. 2014. DOI: 10.1175/JCLI-D-13-00658.1.

[2] S. Manabe and R. T. Wetherald, "On the distribution of climate change resulting from an increase in co2 content of the atmosphere," *J. Atmos. Sci.*, vol. 37, no. 1, pp. 99–118, Jan. 1980. DOI: 10.1175/1520-0469.

[3] N. Künzli, R. Kaiser, S. Medina, M. Studnicka, O. Chanel, P. Filliger, M. Herry, F. Horak, V. Puybonnieux-Texier, P. Quénel, J. Schneider, R. Seethaler, J. C. Vergnaud, and H. Sommer, "Public-health impact of outdoor and traffic-related air pollution: a European assessment." *Lancet (London, England)*, vol. 356, no. 9232, pp. 795–801, Sep. 2000. DOI: 10.1016/S0140-6736(00)02653-2.

[4] "Paris agreement," United Nations Framework Convention on Climate Change, Tech. Rep., 2016.

[5] H. Lawson, "Battery storage," *Designing Buildings Wiki*, Jun 2018. [Online]. Available: https://www.designingbuildings.co.uk/wiki/Battery_storage

[6] X. Yan, B. Zhang, X. Xiao, H. Zhao, and L. Yang, "A bidirectional power converter for electric vehicles in V2G systems," in *IEEE International Electric Machines and Drives Conference*, Chicago, US, 2013, pp. 254–259. DOI: 10.1109/IEMDC.2013.6 556 261.

[7] A. K. Verma, B. Singh, and D. T. Shahani, "Grid to vehicle and vehicle to grid energy transfer using single-phase bidirectional ac-dc converter and bidirectional dc-dc converter," in *International Conference on Energy, Automation, and Signal (ICEAS)*, Bhubaneswar, Odisha, 2011, pp. 1–5. DOI: 10.1109/ICEAS.2011.6 147 084.

[8] J. G. Pinto, V. Monteiro, H. Goncalves, B. Exposto, D. Pedrosa, C. Couto, and J. L. Afonso, "Bidirectional battery charger with Grid-to-Vehicle, Vehicle-to-Grid and Vehicle-to-Home technologies," in *The 39th Annual Conference of IEEE Industrial Electronics Society*, Vienna, 2013, pp. 5934–5939. DOI: 10.1109/IECON.2013.6 700 108.

[9] B. Benkendorff and F. W. Fuchs, "Analysis of a high power low voltage npc converter for wind turbines - influence of mechanical design on performance," in *Proceedings of PCIM Europe 2015; International Exhibition and Conference for Power Electronics, Intelligent Motion, Renewable Energy and Energy Management*, May 2015, pp. 1–8.

[10] G. I. Orfanoudakis, M. A. Yuratich, and S. M. Sharkh, "Nearest-vector modulation strategies with minimum amplitude of low-frequency neutral-point voltage oscillations for the neutral-point-clamped converter," *IEEE Transactions on Power Electronics*, vol. 28, no. 10, pp. 4485–4499, Oct 2013.

[11] J. Liu, H. Zeng, W. Tang, Y. Zhou, H. Zhang, and K. Xiong, "The research on security and stability of pwm technology of high-voltage high-power three-level npc inverter," in *2015 5th International Conference on Electric Utility*

Deregulation and Restructuring and Power Technologies (DRPT), Nov 2015, pp. 2276–2280.

[12] M. A. Abusara, J. M. Guerrero, and S. M. Sharkh, "Line-Interactive UPS for Microgrids," *IEEE Trans. Ind. Electron.*, vol. 61, no. 3, pp. 1292–1300, Mar. 2014. DOI: 10.1109/TIE.2013.2262763.

[13] J. Rodriguez, S. Bernet, P. K. Steimer, and I. E. Lizama, "A survey on neutral-point-clamped inverters," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 7, pp. 2219–2230, July 2010.

[14] A. Nabae, I. Takahashi, and H. Akagi, "A new neutral-point-clamped pwm inverter," *IEEE Transactions on Industry Applications*, vol. IA-17, no. 5, pp. 518–523, Sept 1981.

[15] R. Kelly, V. Santibáñez, and A. Loría, *Control of Robot Manipulators in Joint Space*. Germany: Springer-Verlag London, 2005. ISBN: 1852339942.

[16] K. Aström and R. Murray, *Feedback Systems*. New Jersey: Princetown University Press, 2008. ISBN: 978-0-691-13576-2.

[17] N. Kroutikova, C. Hernandez-Aramburo, and T. Green, "State-space model of grid-connected inverters under current control mode," *IET Electr. Power Appl.*, vol. 1, no. 3, pp. 329–338, May. 2007. DOI: 10.1049/iet-epa:20060276.

[18] C. Osawa, Y. Matsumoto, T. Mizukami, and S. Ozaki, "A state-space modeling and a neutral point voltage control for an npc power converter," in *Proceedings of the Power Conversion Conference*, vol. 1, Nagoaka, 1997, pp. 225–230. DOI: 10.1109/PCCON.1997.645 616.

[19] R.-Y. Kim and D.-S. Hyun, "Averaged modeling and control of a single-phase grid-connected two-stage inverter for battery application," in *39th Annual Conference of IEEE Industrial Electronics Society (IECON)*, Vienna, 2013, pp. 489–494. DOI: 10.1109/IECON.2013.6 699 184.

[20] E. Pouresmaeil, D. Montesinos-Miracle, and O. Gomis-Bellmunt, "Control Scheme of Three-Level NPC Inverter for Integration of Renewable Energy Resources Into AC Grid," *IEEE Syst. J.*, vol. 6, no. 2, pp. 242–253, Jun. 2012. DOI: 10.1109/JSYST.2011.2162922.

[21] J. Pou, R. Pindado, D. Boroyevich, and P. Rodriguez, "Evaluation of the low-frequency neutral-point voltage oscillations in the three-level inverter," *IEEE Transactions on Inductrial Electronics*, vol. 52, no. 6, pp. 1582–1588, Dec 2005.

[22] G. F. Guimares, M. M. A. Severo, G. T. Braga, L. R. Muniz, and V. N. Ferreira, "Control of neutral point deviation in 3-level npc converter with selective harmonic elimination, applied in wind generation systems," in *2017 IEEE 8th International Symposium on Power Electronics for Distributed Generation Systems (PEDG)*, April 2017, pp. 1–6.

[23] N. Mohan, T. M. Undeland, and W. P. Robbins, *Power Electronics: Converters, Applications, and Design*, 2nd ed. Michigan: Wiley, 1995. ISBN: 9780471584087.

[24] I. Takahashi and T. Noguchi, "A new quick-response and high-efficiency control strategy of an induction motor," *IEEE Transactions on Industry Applications*, vol. IA-22, no. 5, pp. 820–827, Sept 1986.

[25] E. Joseph, S. M., and D. Pai, "Speed control of bldc motor drive under dtc scheme using oc with modified integrator," in *2015 International Conference on Technological Advancements in Power and Energy (TAP Energy)*, June 2015, pp. 79–84.

[26] T. Noguchi, H. Tomiki, S. Kondo, and I. Takahashi, "Direct power control of pwm converter without power source voltage sensors," in *Industry Applications Conference, 1996. Thirty-First IAS Annual Meeting, IAS '96., Conference Record of the 1996 IEEE*, vol. 2, Oct 1996, pp. 941–946 vol.2.

[27] N. Li, N. Zhi, H. Zhang, Y. Wang, and Z. Wang, "A novel switching table based direct power control strategy of three-level npc rectifier," in *2016 IEEE 8th International Power Electronics and Motion Control Conference (IPEMC-ECCE Asia)*, May 2016, pp. 2378–2381.

[28] Y. Gui, C. Kim, and C. C. Chung, "Grid Voltage Modulated Direct Power Control for Grid Connected Voltage Source Inverters," pp. 2078–2084, 2017.

[29] A. L. Eshkevari and M. Arasteh, "Virtual flux based model-predictive direct power control of three-phase three-level npc pwm rectifier," in *2017 8th Power Electronics, Drive Systems Technologies Conference (PEDSTC)*, Feb 2017, pp. 172–177.

[30] B. Bouzidi, B. E. Badsi, A. Yangui, and A. Masmoudi, "Bus-clamping-dtc strategy of a three-level-iverter fed induction motor drive with reduced switching frequency and torque ripple," in *2013 Eighth International Conference and Exhibition on Ecological Vehicles and Renewable Energies (EVER)*, March 2013, pp. 1–9.

[31] A. Bouafia, J. Gaubert, and F. Krim, "Predictive direct power control of three-phase pulsewidth modulation (pwm) rectifier using space-vector modulation (svm)," *IEEE Transactions on Power Electronics*, vol. 25, no. 1, pp. 228–236, Jan 2010.

[32] B. Weedy, *Electric Power Systems*, 5th ed. John Wiley and Sons Ltd, 2012.

[33] J. Guzman-Guemez, D. Laila, and S. Sharkh, "State-space approach for modelling and control of a single-phase three-level NPC inverter with SVPWM," *2016 IEEE Power and Energy Society General Meeting (PESGM)*, pp. 1–5, July 2016. DOI: 10.1109/PESGM.2016.7741355.

[34] *Technical Data Fronius Primo*, 2017. [Online]. Available: https://www.fronius.com/~/downloads/Solar%20Energy/Datasheets/SE_DS_Fronius_Primo_EN_US.pdf

[35] *Vincotech 70-W624NIA1K8M701-LD00FP70 datasheet*, 2017. [Online]. Available: https://www.vincotech.com/products/by-topology/topology/three-level-npc-i-type.html

[36] P. R. Remus Teodorescu, Marco Liserre, *Grid Converters for Photovoltaic and Wind Power Systems.* John Wiley and Sons, 2011. ISBN: 9780470057513, 2011.

[37] Z. Zhang, Y.-x. Xie, W.-p. Huang, J.-y. Le, and L. Chen, "A new SVPWM method for single-phase three-level NPC inverter and the control method of neutral point voltage balance," in *International Conference on Electrical Machines and Systems*, Tokyo, 2009, pp. 1–4. DOI: 10.1109/ICEMS.2009.5 382 854.

[38] C. Sintamarean, F. Blaabjerg, H. Wang, and Y. Yang, "Real field mission profile oriented design of a SiC-based PV-inverter application," in *IEEE Energy Conversion Congress and Exposition*, Denver, US, 2013, pp. 940–947. DOI: 10.1109/ECCE.2013.6 646 804.

[39] S. Busquets-Monge, J. Bordonau, D. Boroyevich, and S. Somavilla, "The nearest three virtual space vector PWM - a modulation for the comprehensive neutral-point balancing in the three-level NPC inverter," *IEEE Power Electron. Lett.*, vol. 2, no. 1, pp. 11–15, Mar. 2004. DOI: 10.1109/LPEL.2004.828445.

[40] A. Choudhury, P. Pillay, and S. S. Williamson, "A performance comparison study of space-vector and carrier-based PWM techniques for a 3-level neutral

point clamped (NPC) traction inverter drive," in *IEEE International Conference on Power Electronics, Drives and Energy Systems*, Mumbai, 2014, pp. 1–6. DOI: 10.1109/PEDES.2014.7 041 964.

[41] F. Sebaaly and H. Vahedi, "Design and implementation of space vector modulation-based sliding mode control for grid-connected 3l-npc inverter," *IEEE Transactions on Inductrial Electronics*, vol. 124, no. 12, pp. 7854–7863, dec 2016.

[42] F. A. Okou and S. Gauthier, "A novel robust nonlinear control of a three-phase npc inverter based active power filter," in *Proceedings of the 2010 American Control Conference*, 2010, pp. 1737–1742. DOI: 10.1109/ACC.2010.5 531 489.

[43] K. Ma, M. Liserre, and F. Blaabjerg, "Reactive power control methods for improved reliability of wind power inverters under wind speed variations," in *IEEE Energy Conversion Congress and Exposition*, Raleigh, NC, 2012, pp. 3105–3112. DOI: 10.1109/ECCE.2012.6 342 510.

[44] Biying Ren, Xiangdong Sun, Shaoliang An, Xiangui Cao, and Qi Zhang, "Analysis and design of an LCL filter for the three-level grid-connected inverter," in *Proc. 7th Int. Power Electron. Motion Control Conf.*, vol. 3, Harbin, China, 2012, pp. 2023–2027. DOI: 10.1109/IPEMC.2012.6 259 152.

[45] C. Roncero-Clemente, O. Husev, E. Romero-Cadaval, J. Zakis, D. Vinnikov, and M. I. Milanes-Montero, "Simulation study of the grid-connected single-phase impedance-sourced NPC inverter with different control methods," in *IEEE International Conference on Industrial Technology*, Seville, 2015, pp. 2949–2954. DOI: 10.1109/ICIT.2015.7 125 533.

[46] *Simulink: Developing S-Functions*, 8th ed. Mathworks, Natick, MA, 2016.

[47] Y. Kim, H. Cha, B.-M. Song, and K. Y. Lee, "Design and control of a grid-connected three-phase 3-level NPC inverter for Building Integrated Photovoltaic systems," in *2012 IEEE PES Innov. Smart Grid Technol.*, Washington, US, 2012, pp. 1–7. DOI: 10.1109/ISGT.2012.6 175 663.

[48] E. Bueno, S. Cobreces, F. Rodriguez, A. Hernandez, and F. Espinosa, "Design of a Back-to-Back NPC Converter Interface for Wind Turbines With Squirrel-Cage Induction Generator," *IEEE Trans. Energy Convers.*, vol. 23, no. 3, pp. 932–945, Sep. 2008. DOI: 10.1109/TEC.2008.918651.

[49] J.-S. Lee, K.-B. Lee, and F. Blaabjerg, "Open-Switch Fault Detection Method of a Back-to-Back Converter Using NPC Topology for Wind Turbine Systems," *IEEE Trans. Ind. Appl.*, vol. 51, no. 1, pp. 325–335, Jan. 2015. DOI: 10.1109/TIA.2014.2327151.

[50] A. Mnider, D. Atkinson, M. Dahidah, Y. Zbede, and M. Armstrong, "A programmable cascaded lpf based pll scheme for single-phase grid-connected inverters," in *2016 7th International Renewable Energy Congress (IREC)*, Raleigh, NC, 2016, pp. 1–6.

[51] S. K. Chung, "Phase-locked loop for grid-connected three-phase power conversion systems," *IEE Proceedings - Electric Power Applications*, vol. 147, no. 3, pp. 213–219, May 2000.

[52] V. Kaura and V. Blasko, "Operation of a phase locked loop system under distorted utility conditions," *IEEE Transactions on Industry Applications*, vol. 33, no. 1, pp. 58–63, Jan 1997.

[53] T. Ostrem, W. Sulkowski, L. E. Norum, and C. Wang, "Grid connected photovoltaic (pv) inverter with robust phase-locked loop (pll)," in *2006 IEEE/PES Transmission Distribution Conference and Exposition: Latin America*, Aug 2006, pp. 1–7.

[54] V. Blasko and V. Kaura, "A new mathematical model and control of a three-phase ac-dc voltage source converter," *IEEE Transactions on Power Electronics*, vol. 12, no. 1, pp. 116–123, Jan 1997.

[55] F. Karbakhsh, G. B. Gharehpetian, J. Milimonfared, and A. Teymoori, "Three phase photovoltaic grid-tied inverter based on feed-forward decoupling control using fuzzy-pi controller," in *2016 7th Power Electronics and Drive Systems Technologies Conference (PEDSTC)*, Feb 2016, pp. 344–348.

[56] A. Arzani, P. Arunagirinathan, and G. K. Venayagamoorthy, "Development of optimal pi controllers for a grid-tied photovoltaic inverter," in *2015 IEEE Symposium Series on Computational Intelligence*, Dec 2015, pp. 1272–1279.

[57] J. Moreno-Valenzuela and J. Guzman-Guemez, "Experimental Evaluations of Voltage Regulators for a Saturated Boost DC–to–DC Power Converter," *Trans. of the Inst. of Measurement and Control*, vol. 38, no. 3, pp. 327–337, Mar. 2016. DOI: 10.1177/0142331215592459.

[58] R. Ekstrom and M. Leijon, "Fpga control implementation of a grid-connected current-controlled voltage-source inverter," *Journal of Control Science and Engineering*, 2013.

[59] *Four 1-Bit, 10MHz, 2nd-Order Delta-Sigma Modulators, ADS1204 Texas Instruments Datasheet Rev. 2009*, 2003. [Online]. Available: http://www.ti.com/lit/ds/symlink/ads1204.pdf

[60] B. J. Baliga, *Fundamentals of Power Semiconductor Devices*. Germany: Springer-Verlag, 2010. DOI: 10.1007/978-0-387-47314-7.

[61] T. H. Miroslav Oljaca, *Combining the ADS1202 with an FPGA Digital Filter for Current Measurement in Motor Control Applications, SBAA094 Texas Instruments Application Report*, 2003. [Online]. Available: ti.com/lit/an/sbaa094/sbaa094.pdf

[62] M. Cabello, V. Soler, J. Montserrat, J. Rebollo, P. Godignon, and J. Millan, "High channel mobility in 4h-sic n-mosfet using n2o oxidation combined with boron diffusion treatment," in *2017 Spanish Conference on Electron Devices (CDE)*, Feb 2017, pp. 1–4.

[63] H. Shimizu, K. Konishi, A. Shima, Y. Mori, Y. Shimamoto, S. Sato, R. Fujita, M. Sagawa, T. Ishigaki, N. Tega, and K. Kobayashi, "3.3 kv 4h-sic dmosfet with highly reliable gate insulator and body diode," in *Silicon Carbide and Related Materials 2016*, ser. Materials Science Forum, vol. 897. Trans Tech Publications, 6 2017, pp. 493–496.

[64] W. Sung and B. J. Baliga, "On developing one-chip integration of 1.2 kv sic mosfet and jbs diode (jbsfet)," *IEEE Transactions on Industrial Electronics*, vol. PP, no. 99, pp. 1–1, 2017.

[65] K. Vechalapu, S. Bhattacharya, E. V. Brunt, S. H. Ryu, D. Grider, and J. W. Palmour, "Comparative evaluation of 15-kv sic mosfet and 15-kv sic igbt for medium-voltage converter under the same dv/dt conditions," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 5, no. 1, pp. 469–489, March 2017.

[66] V. Naidu and S. Kotamraju, "Improved switching characteristics obtained by using high-k dielectric layers in 4h-sic igbt: Physics-based simulation," in *Silicon Carbide and Related Materials 2016*, ser. Materials Science Forum, vol. 897. Trans Tech Publications, 6 2017, pp. 571–574.

[67] Y. Mukunoki, Y. Nakamura, T. Horiguchi, and S. Kinouchi, "Characterization and modeling of a 1.2-kv 30-a silicon-carbide mosfet," *IEEE Trans. on Electron Devices*, vol. 63, no. 11, pp. 4339–4345, Nov. 2016.

[68] J. Millán, P. Godignon, X. Perpinà, A. Pérez-Tomás, and J. Rebollo, "A survey of wide bandgap power semiconductor devices," *IEEE Transactions on Power Electronics*, vol. 29, no. 5, pp. 2155–2163, May 2014.

[69] M. Mudholkar, S. Ahmed, M. N. Ericson, S. S. Frank, C. L. Britton, H. A. Mantooth, and S. Member, "Datasheet Driven Silicon Carbide Power MOSFET Model," *IEEE Trans. Power Electron.*, vol. 29, no. 5, pp. 2220–2228, May. 2014. DOI: 10.1109/TPEL.2013.2295774.

[70] R. Kraus, P. Turkes, and J. Sigg, "Physics-based models of power semiconductor devices for the circuit simulator SPICE," in *PESC 98 Record 29th Annual IEEE Power Electrononics Specialists Conference*, vol. 2, Fukoka, 1998, pp. 1726–1731. DOI: 10.1109/PESC.1998.703 414.

[71] J. Wang, T. Zhao, J. Li, A. Q. Huang, R. Callanan, F. Husna, and A. Agarwal, "Characterization, Modeling, and Application of 10-kV SiC MOSFET," *IEEE Trans. Electron Devices*, vol. 55, no. 8, pp. 1798–1806, Aug. 2008. DOI: 10.1109/TED.2008.926650.

[72] K. Sheng, B. W. Williams, and S. J. Finney, "A review of IGBT models," *IEEE Transactions on Power Electronics*, vol. 15, no. 6, pp. 1250–1266, Nov 2000.

[73] Z. Duan, T. Fan, X. Wen, and D. Zhang, "Improved sic power mosfet model considering nonlinear junction capacitances," *IEEE Transactions on Power Electronics*, vol. PP, no. 99, pp. 1–1, 2017.

[74] G. L. Kusic and G. F. Reed, "Comparative PSCAD and Matlab/Simulink simulation models of power losses for SiC MOSFET and Si IGBT devices," in *IEEE Power Energy Conference*, Champaign, IL, 2012, pp. 1–5. DOI: 10.1109/PECI.2012.6 184 589.

[75] Y. Cao, L. Yuan, K. Chen, Z. Zhao, T. Lu, and F. He, "Modeling of sic mosfet in matlab/simulink," in *2014 IEEE Conf. Expo. Transportation Electrification Asia-Pacific (ITEC Asia-Pacific)*, Aug 2014, pp. 1–5.

[76] G. Kampitsis, M. Antivachis, S. Kokosis, S. Papathanassiou, and S. Manias, "An accurate matlab/simulink based sic mosfet model for power converter applications," in *2015 IEEE Applied Power Electronics Conf. Expo. (APEC)*, March 2015, pp. 1058–1064.

[77] P. Giammatteo, C. Buccella, and C. Cecati, "Matlab/simulink modeling of sic power mosfets," *Int. Review of Electrical Engineering (IREE)*, vol. 9, no. 4, 2014. [Online]. Available: http://www.praiseworthyprize.org/jsm/index.php?journal=iree&page=article&op=view&path%5B%5D=16085

[78] L. Michel, A. Cheriti, and P. Sicard, "Development of an efficient IGBT simulation model," in *Canadian Conference on Electrical and Computer Engineering*, St. Jons, Canada, 2009, pp. 252–256. DOI: 10.1109/C-CECE.2009.5 090 131.

[79] J. T. Hsu and K. D. T. Ngo, "Behavioral modeling of the igbt using the hammerstein configuration," *IEEE Transactions on Power Electronics*, vol. 11, no. 6, pp. 746–754, Nov 1996.

[80] C. Alonso, T. A. Meynard, H. Foch, C. Batard, and H. Piquet, "A model of GTO compatible with power circuit simulation," in *1993 Fifth European Conf. on Power Electronics and Applications*, Sept 1993, pp. 232–237 vol.2.

[81] Y. Shen, J. Jiang, Y. Xiong, Y. Deng, X. He, and Z. Zeng, "Parasitic inductance effects on the switching loss measurement of power semiconductor devices," in *2006 IEEE International Symposium on Industrial Electronics*, vol. 2, July 2006, pp. 847–852.

[82] *Silicon Power MOSFET, IRF640 Vishay Siliconix Datasheet*, 2015. [Online]. Available: http://www.vishay.com/docs/91036/sihf640.pdf

[83] *Silicon Carbide Power MOSFET, C2M1000170D Cree Datasheet Rev. A*, 2013. [Online]. Available: http://www.wolfspeed.com/media/downloads/173/C2M1000170D.pdf

[84] *Silicon Carbide Power MOSFET, SCT2080KE Rohm Datasheet*, 2015. [Online]. Available: http://rohmfs.rohm.com/en/products/databook/datasheet/discrete/sic/mosfet/sct2080ke-e.pdf

[85] *Field Stop Trench IGBT, RGT16NS65D Rohm Datasheet*, 2015. [Online]. Available: http://www.rohm.com/web/eu/datasheet/RGT16NS65D/rgt16ns65d-e

[86] T. Lee, *Planar Microwave Engineering, A Practical Guide to Theory, Measurement, and Circuits*. Cambridge, UK: Cambridge University Press, 2004.

[87] *Silicon Carbide Power MOSFET, C2M1000170D Cree Datasheet Rev. E*, 2015. [Online]. Available: http://www.szapl.com/manage/images/201514142713775.pdf

[88] F. Casanellas, "Losses in PWM inverters using IGBTs," *IEE Proceedings - Electric Power Applications*, vol. 141, no. 5, pp. 235–239, Sep 1994. DOI: 10.1049/ip-epa:19941349.

[89] *Semikron 3Level Adapter, Devboards Datasheet v1.2*, 2015. [Online]. Available: https://www.devboards.de/en/home/products/product-details/article/semikron-3level-adapter/