5-2016

Correcting Current Imbalances in Three-Phase Four-Wire Distribution Systems

Vinson Joseph Jones
University of Arkansas, Fayetteville

Correcting Current Imbalances in Three-Phase Four-Wire Distribution Systems

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Electrical Engineering

by

Vinson Jones
University of Arkansas
Bachelor of Science in Electrical Engineering, 2014

May 2016
University of Arkansas

This thesis is approved for recommendation to the Graduate Council.

Dr. Juan Carlos Balda
Thesis Director

Dr. Alan Mantooth
Committee Member

Dr. Roy McCann
Committee Member

ABSTRACT

The objective of this thesis is to present the theory, design, construction, and testing of a proposed solution to unbalanced current loading on three-phase four-wire systems. The Unbalanced Current Static Compensator is the name of the prototype; herein referred to as the UCSC. The purpose of this prototype is to redistribute current between the three phases of a distribution system. Through this redistribution, negative- and zero-sequence currents are eliminated and a balanced system is seen upstream from the point of installation.

The UCSC consists of three separate single-phase H-bridge inverters that all share the same dc-link capacitor. Each of these inverters performs independently using a single-phase rotating reference frame controller. Each either draws or injects current onto the distribution system lines to balance the active currents and performs power factor correction for voltage compensation. A 34.5 kV, 6 MVA system was built and simulated in Matlab/Simulink™ to test the validity of this solution. A scaled-down UCSC prototype was then designed and constructed to compensate for a 208 V, 10 kVA system. Results from both the simulations and testing of this UCSC prototype are presented and analyzed.

ACKNOWLEDGEMENTS

Thank you to Dr. Balda for the opportunity to further my knowledge in Electrical Engineering and the support and advice on this project.

TABLE OF CONTENTS

CHAPTER 1

CHAPTER 2

CHAPTER 4

APPENDIX B

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Motivations for this Research Work

In an ideal three-phase four-wire distribution system, each of the phases would be connected to identical customers and the same amount of current would be drawn from each phase. In practical systems, this is not the case as there is much variation in the type and connection timing of customers. This situation leads to an unbalance in the currents that run through each phase. As a result, the currents do no sum to zero at the neutral conductor, which causes several issues in the power transmission system:

- Extra current in the neutral wire of four-wire systems which can result in unsafe conditions, and in some cases, an incorrectly sized conductor [**1**].
- Electric machine overheating due to negative-sequence currents [**2**].
- Magneto-motive force (MMF) generated by negative-sequence currents interferes with the positive-sequence MMF. This along with the machine overheating causes a drop in the efficiency of the system [**3**].
- Zero-sequence fluxes produced in wye-connected ungrounded transformers can cause overheating [**4**].

Several different attempts have been made to mitigate the effects of the negative-sequence and zero-sequence currents. These are described in the next section.

1.2 Existing Solutions

The presented solutions can be classified into two different types of compensators: passive and active compensators. The passive solutions do not react to changes in the system like the active compensators and, as a result, are often not as effective. However, passive systems are usually cheaper and do not require complex control techniques and sensing that active systems do.

1.2.1 Zig-Zag Transformers

To eliminate the zero-sequence current components that would flow through the neutral in three-phase four-wire systems one proposed method is the introduction of a zig-zag transformer in parallel with the load [**5**]. Zig-zag transformers are wound in such a way that they have very low impedance to zero-sequence currents and very large impedance to positive- and negative- sequence currents. The system setup is shown in Fig. 1-1 where Z_s is the impedance of the phase conductor between the source and transformer, Z_{sn} is the impedance of the neutral conductor between the transformer and the source, and Z_{Ln} is the neutral conductor impedance

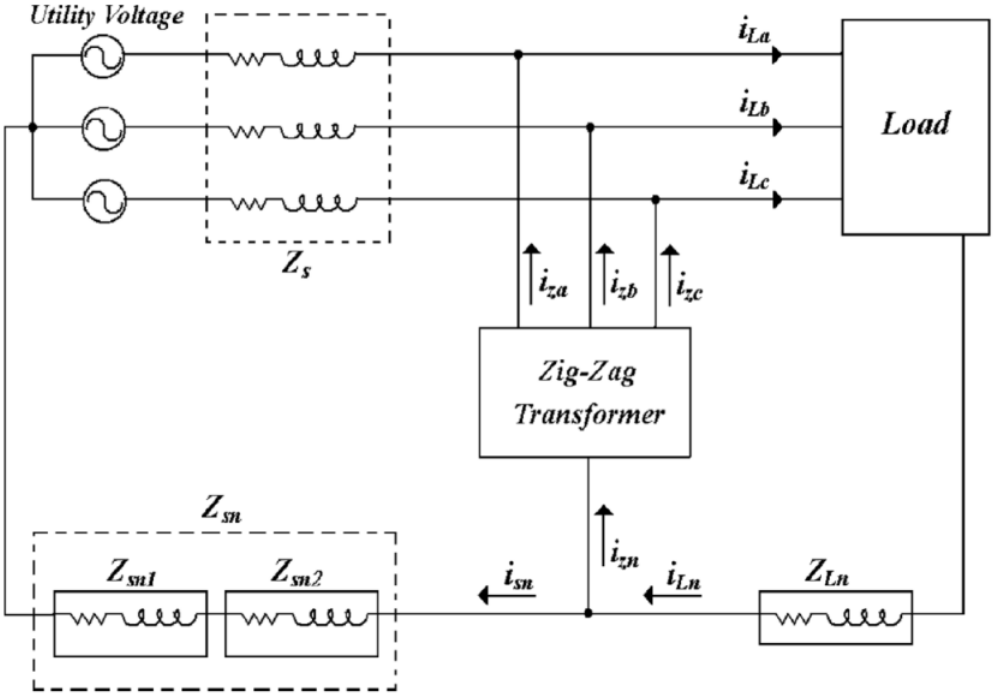


Figure 1-1. Zig-zag transformer compensation system [**1**]

between the load and the transformer. Ideally, this would remove the zero-sequence current injected into the utility system and thus reduce the stress on the system; however, the impedance Z_{sn} plays a large part in the performance of this solution. If Z_{sn} is large then the attenuation is better, though this requires that the transformer is either further away from the utility, thus compensating less load, or that an inductor be placed in series with the neutral, which adds a significant cost to the system and can cause abnormal operation of load-side equipment. Moreover, this solution loses its advantage in unbalanced utility voltage conditions due to the fundamental-component zero-sequence currents which may cause the transformer to overheat. These disadvantages suggest the need for another solution, particularly based on power electronics.

1.2.2 Neutral-Wire Series Compensators

A voltage-source inverter in series with the neutral conductor is applied to reduce the harmonic currents reaching the utility system [**6**]. Fig. 1-2 demonstrates a test case. By inducing a series voltage in the neutral conductor the inverter creates a high impedance path for harmonic currents in the form of a virtual inductor. The performance of the system improves as

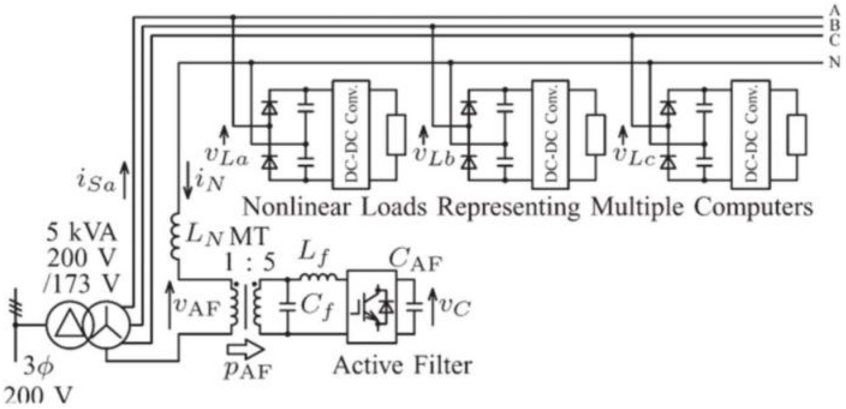


Figure 1-2. Neutral-wire series compensator system [**2**]

the reactance of the inverter increases, but a balance must be achieved between the desired attenuation of neutral current and the level of voltage distortion that is caused as a result.

1.2.3 Neutral Wire Series Compensator/Zig-Zag Transformer Hybrid

This method combines the two previous solutions. In this system there is a zig-zag transformer connected in parallel with the load and in addition there is a voltage-source inverter connected in series with the utility-side neutral conductor [**7**]. Fig. 1-3 demonstrates the described system.

The series inverter helps to force the harmonic zero-sequence currents through the transformer and back to the load by producing a voltage waveform in the neutral that mimics high impedance. The hybrid systems require less knowledge of the system impedance for them to be effective and allow for smaller and cheaper zig-zag transformers. This solution is effective, and reduces current imbalances, but only through mitigating the harmonic currents contribution to this problem. The fundamental zero-sequence currents still flow back to the utility.

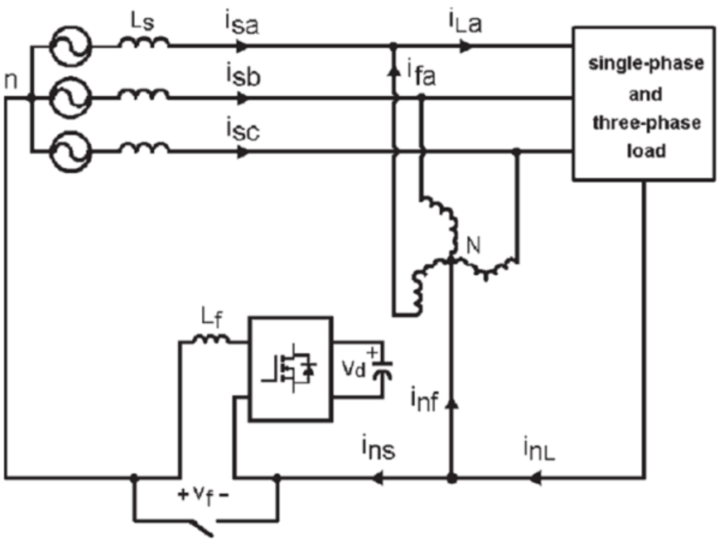


Figure 1-3. Zig-zag and series compensator hybrid system [**3**]

1.2.4 Three-Level Inverter with Neutral Point Clamp (NPC)

This solution uses a NPC three-leg voltage source inverter as shown in Fig. 1-4 where L_s is the inductance of the conductor from the source to the point of common coupling (PCC), R_c is the resistance between the output of the inverter and the PCC, and L_c is the inductance between the output of the inverter and the PCC [**8**]. This setup allows for better compensation of neutral currents due to the added control of the output voltage relative to the neutral. To control this inverter, a switch commutation strategy called space vector modulation is implemented where a map of all possible switching combinations is constructed [**8**]. A different set of output voltage or output vectors is obtained for each of these combinations. Through control in the α-β-0 or other domain, the amount of time spent at each of these vectors is calculated as well as the order. Using this technique an averaged output vector can be produced to allow for precise control over the output voltage.

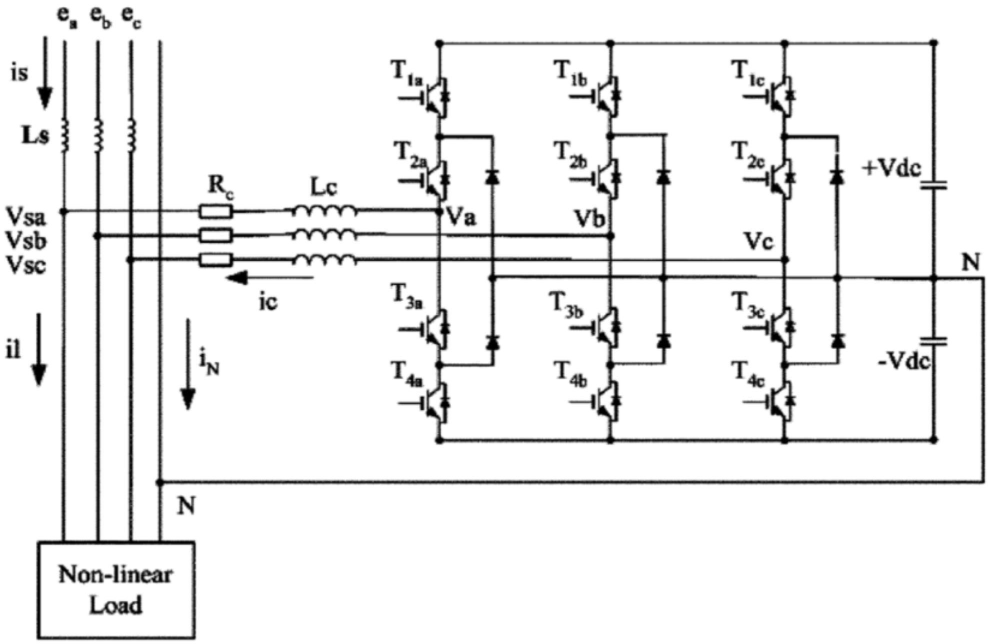


Figure 1-4. NPC three-phase three-level voltage source inverter [**4**]

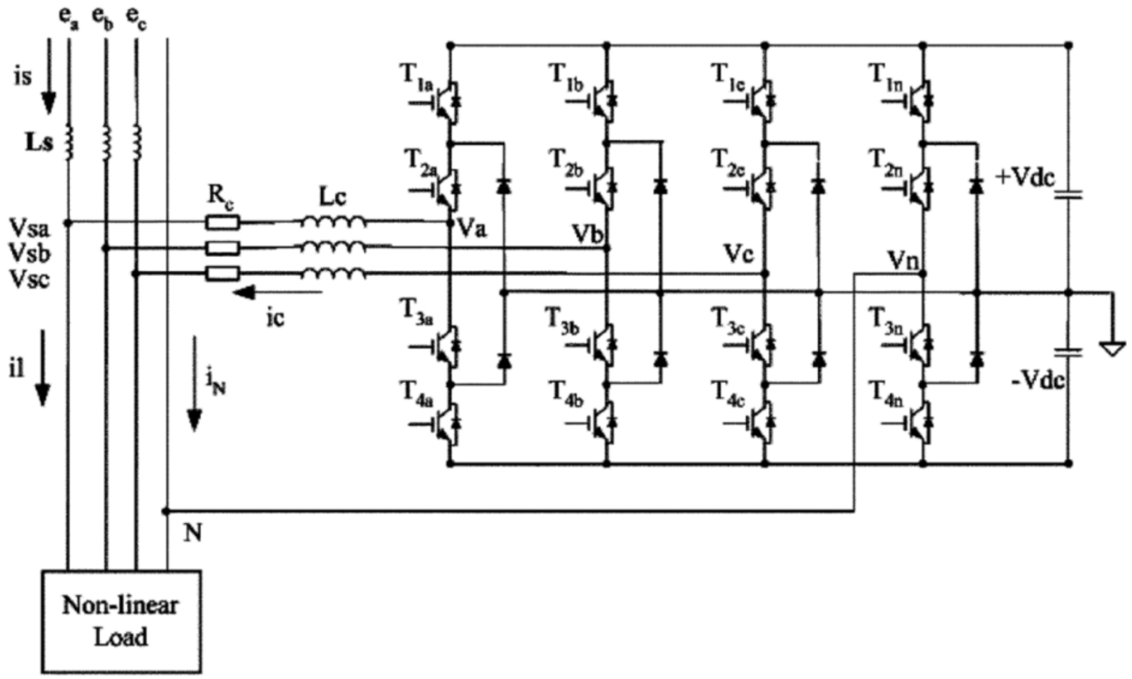


Figure 1-5. Three-level four-leg voltage source inverter [**4**]

This method can be generalized for any number of switching positions and can also be used to control the four-leg converter in Fig. 1-5Figure 1-5. The fourth leg of this inverter gives the user better control over neutral currents as there are now more neutral point switching vectors to choose from; however, it is not only more expensive to add this fourth leg, but the SVM becomes more complex.

The complexity of using SVM, compared to SPWM (sinusoidal pulse width modulation), can be a problem due to the extra computation resources required to implement the modulation technique. Larger computational load can lead to reduction in the frequency at which the controller can be run when implementing these techniques on digital signal processors [**9**]. This disadvantage could be mitigated by delegating the control signal modulation to an FPGA with parallel computing ability, but integration of an FPGA into the design adds more complexity to the final solution.

1.2.5 α-β Instantaneous Power Controller

This solution keeps the controller in the stationary α-β reference frame. Given that the filtering and computational delays are properly dealt with, this type of control allows for the elimination of a phase-lock loop as well as a second transformation into the dq rotating reference frame. The control in [**6**] works by calculating the instantaneous power required by the load and, by normalizing the α and β voltage terms, and a current reference for the inverters is then set which is proportional to the power needed on each phase. This controller achieves full compensation of reactive currents required by the load and current balancing. Also, it can compensate for low-order harmonics present within the distribution system with the use of a low-pass filter on the instantaneous power calculations.

Two different versions of this controller are used. The direct current controller method generates the control signal for an active filter directly by calculating the harmonic and reactive current components that need to be delivered to the load by the filter. The indirect current controller calculates the fundamental-component current that should be the only one supplied by the source. The difference between this fundamental component reference and the actual source current becomes the control signal for the active filter. As a result it provides all current components but the fundamental one. Fig. 1-6Figure 1-6 contains the block diagram for the indirect current controller. The i_L components are the measured load currents, the v_L components are the voltages of each phase at the PCC, and the $\mathrm{i}_{\mathrm{sb}}^{*}$ components are the generated current references for the inverters.

Fig. 1-7Figure 1-7 indicates the topology of the inverter used in the testing of the α-β controller. This topology is convenient because it is not susceptible to the shoot-through current that

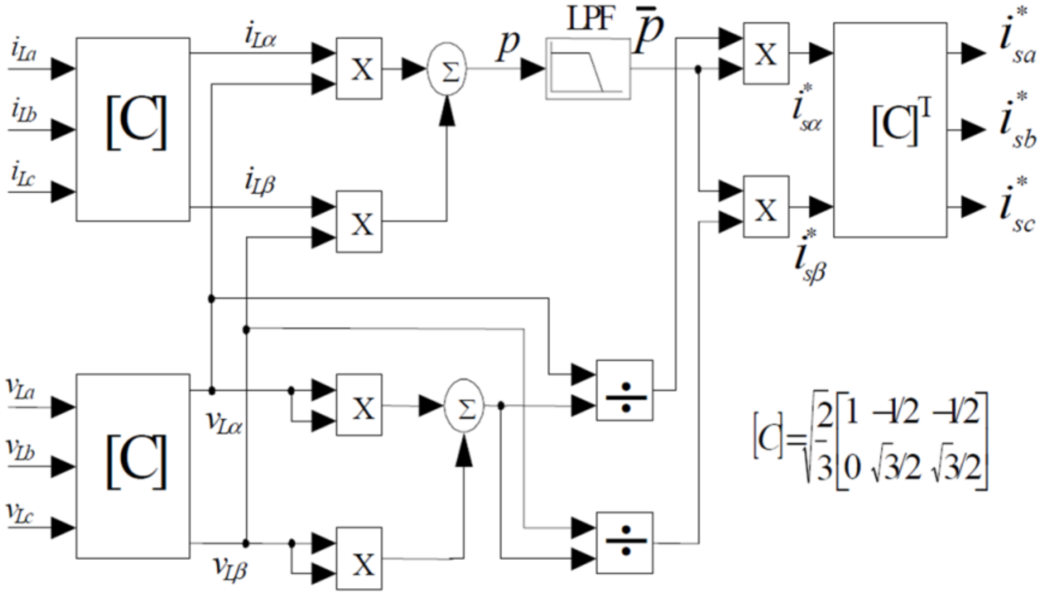


Figure 1-6. Indirect current control block diagram [**5**]

normally is a problem in inverters with cascade devices, and thus reduces the need for the dead time in the switching pulses. This topology has the same number of active switches as a standard three-phase inverter, but has twice the number of diodes and twice the PCC connecting reactors. A NPC configuration is also required for proper functionality, which adds to the cost.

The α-β stationary reference frame controllers are generally easier to implement than d-q synchronous reference frame controllers. This is because they do not require as many reference frame transformations or the phase-lock loops required for the transformations. They may also have better transient behavior as experienced in [**7**] However, their performance can be severely diminished by any magnitude or phase error in the measurements or calculations because these control strategies directly use ac waveforms to construct the current references in the inverter controller [**12**].

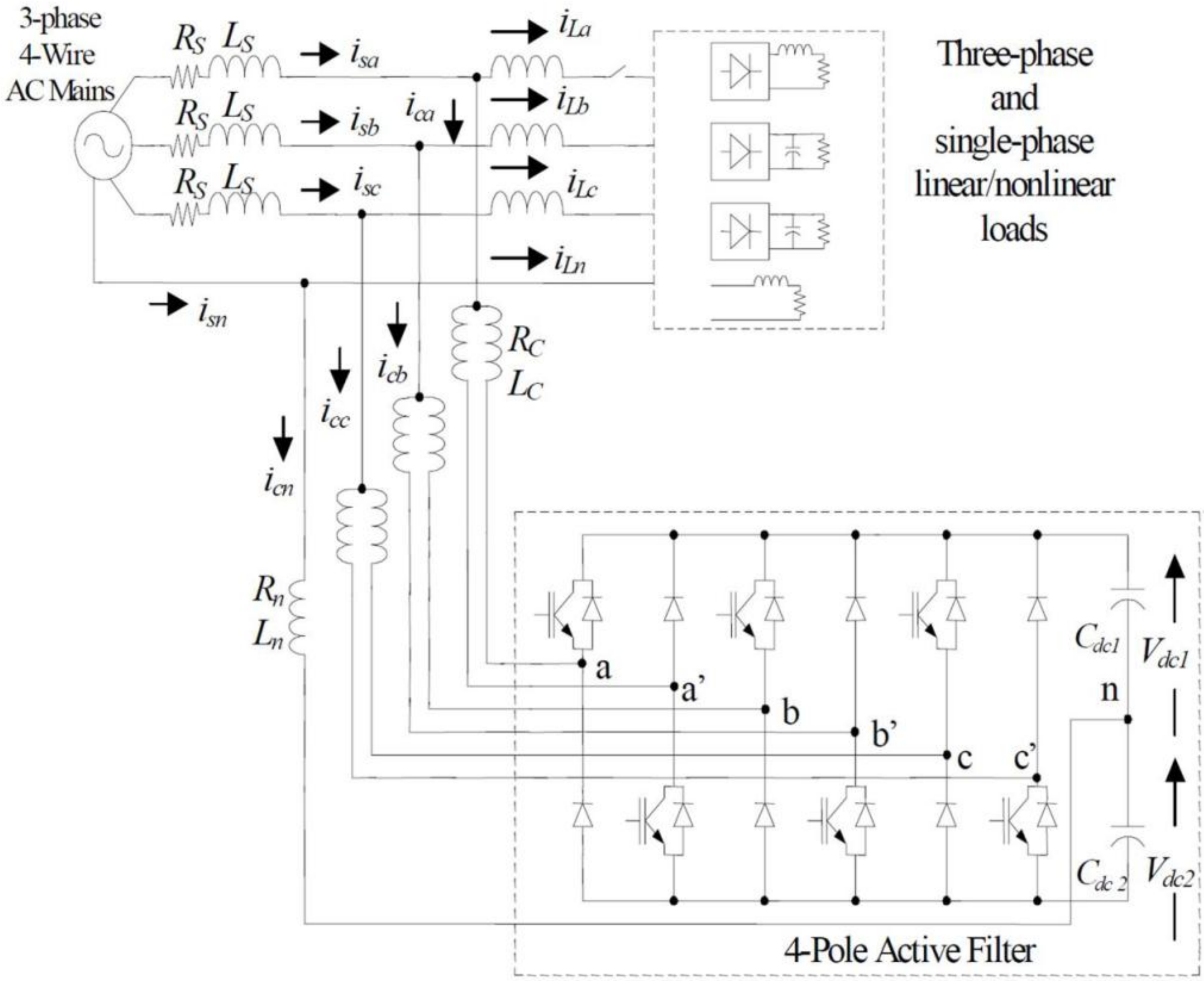


Figure 1-7. Proposed shoot-through current immune topology [**5**]

1.3 Proposed UCSC Solution

A case study is developed to test any possible solutions based on data from a 34.5 kV distribution feeder. At peak power, the three phase loads were 877 kW + 500 kVAR, 707 kW + 300 kVAR, and 753 kW + 300 kVAR, so these load ratings were chosen for the case study [**13**].

The topology selected for the UCSC has three single-phase full-bridge inverters. This is useful for several reasons:

- Each of the inverters can control their respective phases current completely independent of the other phases.
- There is full utilization of the dc bus voltage which is not the case for neutral-point-clamped inverters.
- Control of an H-bridge is relatively simple and does not require more complex modulation schema as required by four-leg inverter topologies.

The largest flaw with the use of this topology is that 12 switching position are required and thus 12 IGBTs and 12 diodes are required.

A diagram showing the connection of the UCSC to the distribution system is displayed in Fig. 1-8Figure 1-8. Each of the inverters shares a single dc-bus capacitor. All three are coupled to the 34.5 kV distribution system using 480V/19.9kV single-phase distribution transformers that boost the voltage to the required value as well as provide the necessary isolation required by industry standards. Not shown in the figure is the line-to-neutral LCL filter at the output of each of the inverters. The UCSC is to be installed just downstream from the distribution substation to compensate for as many downstream loads as possible.

1.4 Objectives of Thesis

The main objectives of this thesis are to improve upon the UCSC solution developed in [**9**] and then, to validate its performance using a constructed prototype. This will be done in three main steps:

1) First, the theoretical background behind the operation of the UCSC will be developed.
2) Second, simulation results will be analyzed for potential improvements.

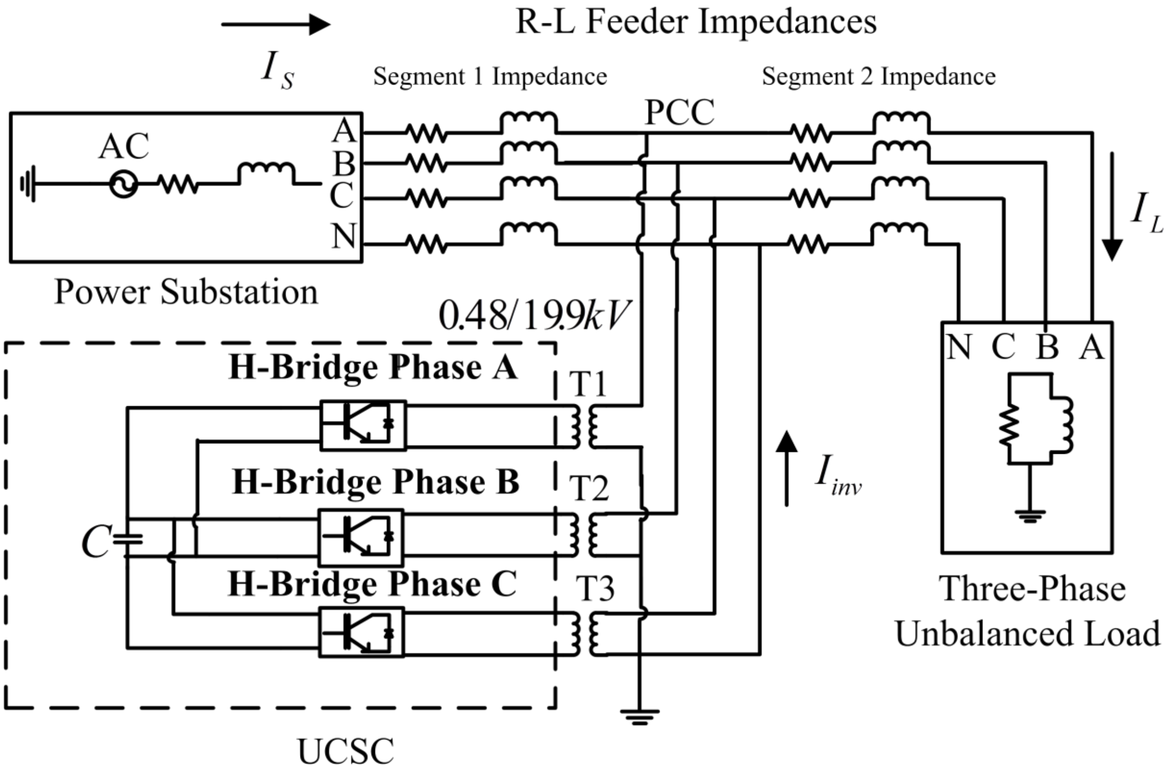


Figure 1-8. Connection of the UCSC to the distribution system [**6**]

3) Third, the scaled down prototype will be built, tested, and the experimental results analyzed.

1.5 Organization of Thesis

The theory behind the operation of the UCSC will be presented in Chapter 2. The simulations of the system will be addressed in Chapter 3. This will include the construction of the simulation models and any changes from previous model in [**13**] and the results from those simulations. The assembly and design of the scaled down prototype is documented in Chapter 4 and the controller implementation is presented in Chapter 5. The measurements and analysis of the results that come from testing of the prototype are found in Chapter 6. Conclusions and recommendations for future work are given in Chapter 7.

1.6 References

[1] J. Kennedy, C. A. Nucci, A. Borghetti, G. Contaxis A. P. S. Meliopoulos, "Power distribution practices in USA and Europe: impact on power quality," in *Proceedings of the 8th International Conference on Harmonics and Quality of Power*, Athens, 1998, pp. 24-29.

[2] R. G. Harley, E.B. Makram, and E.G. Duran, "The effects of unbalanced networks on synchronous and asynchronous machine transient stability," *Electric Power System Research*, vol. 13, no. 2, pp. 119-127, October 1987.

[3] R.H. Salim, R.A. Ramos, and N.G. Bretas, "Analysis of the small signal dynamic performance of synchronous generators under unbalanced operating conditions," in *IEEE Power and Energy Society General Meeting*, 2010, pp. 1-6.

[4] T.A. Short, *Electrical Power Distribution Handbook*. Boca Raton, FL: CRC Press Taylor & Taylor Croup, 2004.

[5] Jou Hurng-Liahng, Wu Jinn-Chang, Wu Kuen-Der, Chiang Wen-Jung, and Chen Yi-Hsun, "Analysis of zig-zag transformer applying tin the three-phase four-wire ditribution power system," *IEEE Transactions on Power Delivery*, vol. 20, no. 2, pp. 1168-1173, April 2005.

[6] S Inoue, T. Shimizu, and K. Wada, "Control Methods and Compensation Characteristics of a Series Active Filter for a Neutral Conductor," *Industrial Electronics, IEEE Transactions on*, vol. 54, no. 1, pp. 433-440, February 2007.

[7] Sewan Choi and Minsoo Jang, "Analysis and Control of a Single-Phase-Inverter-Zigzag-Transformer Hybrid Neutral-Current Suppressor in Three-Phase Four-Wire Systems," *Industrial Electronics, IEEE Tansactions on*, vol. 54, no. 4, pp. 2201-2208, August 2007.

[8] Ning-Yi Dai, Man-Chung Wong, and Ying-Duo Han, "Application of a three-level NPC inverter as a three-phase four-wire power quality compensator by generalized 3DSVM," *Power Electronics, IEEE Transactions on*, vol. 21, no. 2, pp. 440-449, March 2006.

[9] D. Dujic, M. Jones, and Levi E., "Contrinuous Carrier-Based vs. Space Vector PWM for Five-Phase VSI," in *EUROCON, 2007. The International Conference on Computer as a Tool*, Warsaw, 2007, pp. 1772-1779.

[10] Singh G.N and P Rastgoufard, "A new topology of active filter to correct power-factor, compensate harmonics, reactive power and unbalance of three-phase four-wire loads," in *Applied Power Electronics Conference and Exposition*, Miami Beach, 2003, pp. 141-147.

[11] Matias Diaz and Roberto Cardenas, "Analysis of Synchronous and Stationary Reference Frame Control Strategies to Fulfill LVRT Requirements in Wind Energy Conversion Systems," in *Ecological Vehicles and Renewable Energies (EVER), 2014 Ninth*

International Conference on, Monte-Carlo, 2014, pp. 1 - 8.

[12] D. N. Zmood and D. G. Holmes, "Stationary frame current regulation of PWM inverters with zero steady state error," in *Power Electronics Specialists Conference*, Charleston, 1999, pp. 1185-1190.

[13] Manuel A. S. Tejada, "Unbalanced Current Static Compensator," Dept. of Elect. Eng., Univ. of Arkansas, Fayetteville, MS Thesis 2014.

CHAPTER 2

THEORY OF UCSC OPERATION

2.1 Control of the UCSC

The power topology of the UCSC is displayed in Fig. 2-1 and the control block schematic for each inverter is shown in Fig. 2-2Figure 2-2. The UCSC's three single-phase inverters inject or draw current at the point of common coupling (PCC) to compensate for the load currents in such a way that the substation currents are balanced. The various subsystems of the controller will be explained next. The controller first transforms the inverter and load currents along with phase voltages at the PCC into the d-q synchronously-rotating reference frame using a

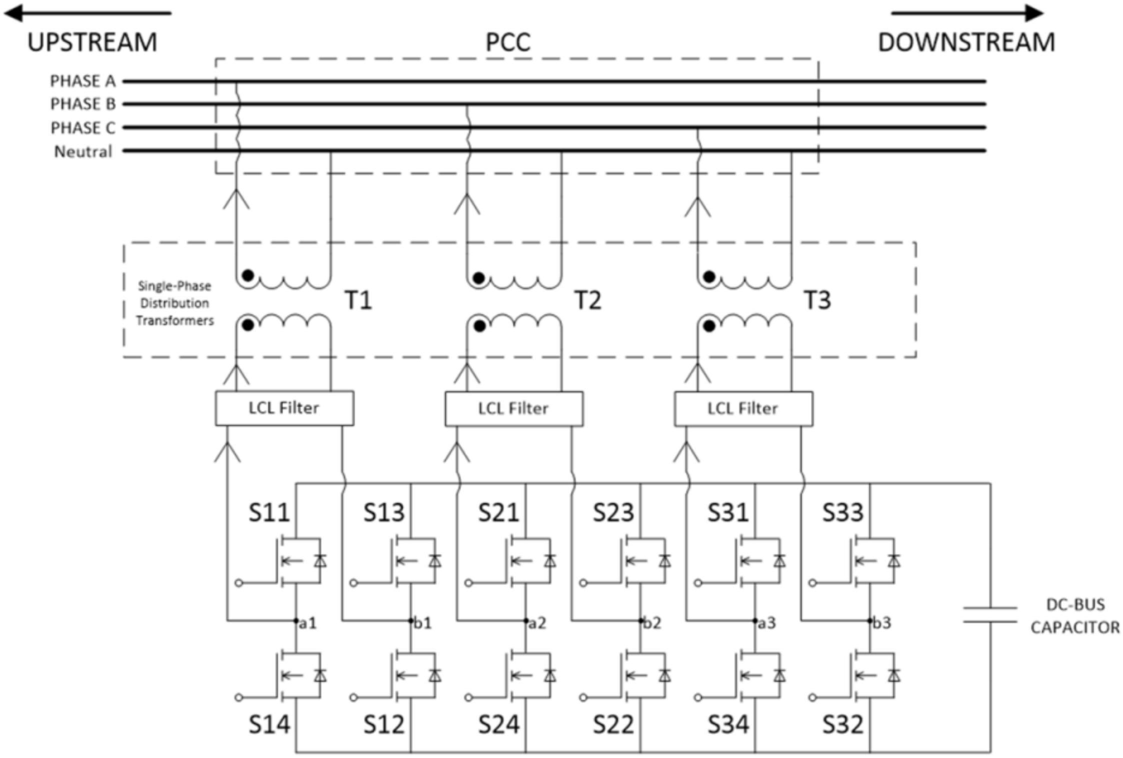


Figure 2-1. The UCSC power topology

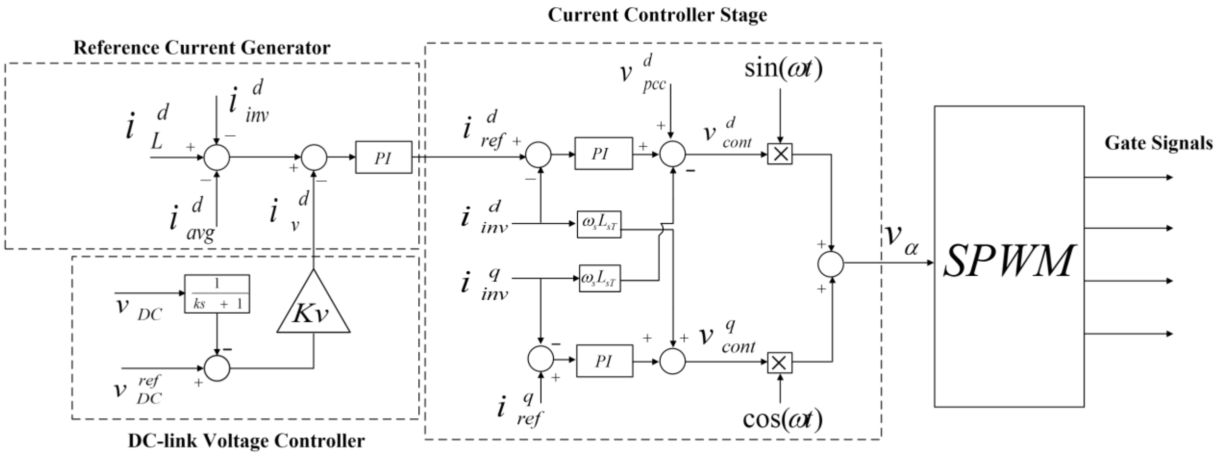


Figure 2-2. UCSC simplified block schematic

single-phase α-β to d-q transformation [**1**].

2.1.1 *Reference Current Generator Stage*

This section of the block schematic is responsible for generating the reference signal for the current controller. Using the d-axis component of the load current from each phase an average load current is calculated i_{avg}^d. The substation currents i_{sa}^d, i_{sb}^d, and i_{sc}^d are then compared to the average load current through current summation at the PCC $\left(i_s^d = -i_{inv}^d - i_L^d\right)$ to develop the respective d-axis current references. The resulting error signal for each phase is summed with i_v^d and routed into a PI controller whose output is the d-axis reference current i_{ref}^d for the current controller.

Because the three single-phase inverters circulate the currents between the phases, there is no net current drawn from the grid into the capacitor except the current used to maintain the dc bus voltage. The PI controller gains $\left(k_{pref}, k_{iref}\right)$ are specified by using the following method from [**2**]:

$$i_{cap} = C_{DC}\frac{dv_{DC}}{dt} = [k_{pref}\left(v_{DC}^{ref} - v_{DC}\right) + k_{iref}\int\left(v_{DC}^{ref} - v_{DC}\right)] \tag{1}$$

In (1) the capacitor equation relating its current and voltage is used to develop a relationship between the error signal $\left(v_{DC}^{ref} - v_{DC}\right)$ and the capacitor voltage. Using the Laplace transform and solving for $\frac{v_{DC}}{v_{DC}^{ref}} = T_{ref}(s)$ in (1) results in:

$$T_{ref}(s) = \frac{v_{DC}}{v_{DC}^{ref}} = \frac{k_{pref}}{C_{DC}}\frac{(s + \frac{k_{iref}}{k_{pref}})}{(s^2 + \frac{k_{pref}}{C_{DC}}s + \frac{k_{iref}}{C_{DC}})} \tag{2}$$

this takes a similar form to the general form of a two-pole transfer function given by:

$$T(s) = \frac{\omega_{cc}^2}{(s^2 + 2\zeta\omega_{cc}s + \omega_{cc}^2)} \tag{3}$$

Using the theory behind this general form, the response of the reference PI controller can be determined through the following relationships:

$$2\zeta\omega_{cc} = \frac{k_{pref}}{(C_{DC})} \quad \longrightarrow \quad k_{pref} = 2\zeta\omega_{cr}(C_{DC}) \tag{4}$$

$$\omega_{cc}^2 = \frac{k_{iref}}{(C_{DC})} \quad \longrightarrow \quad k_{iref} = (C_{DC})\omega_{cr}^2 \tag{5}$$

A damping factor $\zeta = \frac{\sqrt{2}}{2}$ is used and corresponds to a 5% overshoot given a step change in set point.

2.1.2 *Voltage Controller Stage*

The voltage controller regulates the voltage for the dc-link capacitor, common to all three phases of the UCSC, within a certain band; the specified voltage ripple is 1% of the rated voltage The controller senses and filters the bus voltage v_{DC} and compares this instantaneous value to a set reference value v_{DC}^{ref}. The difference in these values is then processed through a gain to regulate the effect of an error in the dc bus voltage versus an error in the currents. The error signal i_v^d, after being modified, is added to the reference current generators error signal.

2.1.3 *Current Controller*

The d-axis current reference i_{ref}^d is compared to the inverter d-axis current i_{inv}^d. The error between these two signals is then routed to a PI controller to establish the response characteristics of the current controller [**1**]. The d-axis component of the grid voltage v_{pcc}^d is added to the signal as a feed-forward element to bias the voltage output of the inverter. Also, a decoupling term is added to take into account the coupling between the q-axis currents and the inductances of the filter and transformer. The same process is done for the q-axis current controller; however, the current reference becomes the q-axis component of the load current (this fully compensates for the reactive power). The feed-forward q-axis grid voltage is assumed to be 0. The decoupling term comes from the interaction between the d-axis current component and the inductances present. To produce a control signal for the PWM generator the signals are converted back to the α-β reference frame, where the α component is the output voltage waveform.

The simplified block diagram for the current controller is shown in Fig. 2-3Figure 2-3. The first block represents the transfer function for the PI controller and the second one the plant model for the output filter. The PI controller gains for the current controller (k_{pcc}, k_{icc}) are specified by using the following method [**1**]. The transfer function L(s) for the open-loop system is given by:

$$L(s) = \frac{(k_{pcc}s + \ K_{icc})}{Ls^2 + Rs} = (\frac{K_{pcc}}{Ls})\frac{(s + \frac{K_{icc}}{K_{pcc}})}{(s + \frac{R}{L})} \tag{6}$$

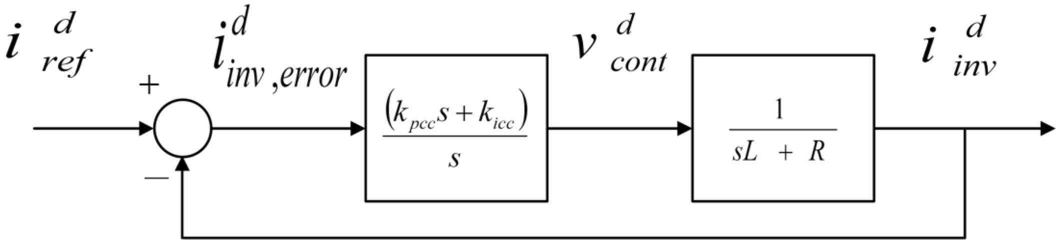


Figure 2-3. Simplified current controller block diagram

By choosing the relationship between K_{icc} and K_{pcc} appropriately $\left(\frac{K_{icc}}{K_{pcc}} = \frac{R}{L}\right)$, the pole and zero cancel out. This is useful because this leads to a much simpler closed-loop transfer function given by:

$$G(s) = \frac{L(s)}{1 + L(s)} = \frac{1}{\left(\frac{L}{K_{pcc}}\right) s + 1} \tag{7}$$

The closed-loop system is drawn in Fig. 2-4Figure 2-4. Using the chosen proportional and integral gain ratio as given above the PI gains can be selected from:

$$\frac{L}{K_{PCC}} = \tau_{cc} \tag{8}$$

2.2 ABC to D-Q Transformation

The control diagram described above requires dc values in order to be stable. This requires the ac voltages and currents that are sensed by the UCSC to be transformed into dc

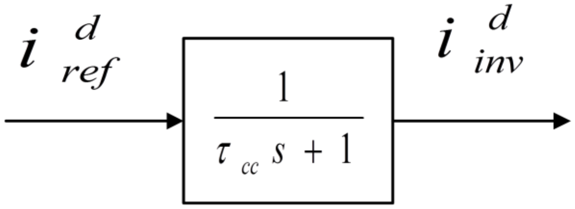


Figure 2-4. Equivalent current controller block diagram

value equivalents in order to be used [**3**]. This is accomplished through the use of the Park's transformation, otherwise known as the d-q-0 transformation, which takes an input vector of the 3 voltage (or current) functions and outputs a vector that consists of dc values in steady state corresponding to the magnitude of the input ac functions.

The d-q-0 transformation is derived from another vector manipulation method known as the Clarke's transformation or the α-β-0 transformation that changes the reference axis that is used to measure the phasor values in a three-phase system [**4**]. The new reference axes are called the α, β, and 0 axis that are rotated such that the zero axis is equidistant from the original a, b, and c axes. The new values for the three phasors are their reflections upon the new reference axes. Thus, all the original phasors in a balanced three-phase system cancel out to zero on the zero axis. This is useful for the control of balanced three-phase systems because it effectively eliminates a variable. A system defined in the abc reference frame can be fully described by the variables α and β. The α-β-0 transformation is given by [**3**]:

$$\begin{bmatrix} \alpha \\ \beta \\ 0 \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{9}$$

With the system in the αβ0 reference frame, the transformation to the dq0 reference frame is given by:

$$\begin{bmatrix} d \\ q \\ 0 \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \cos(\omega t) & \sin(\omega t) & 0 \\ -\sin(\omega t) & \cos(\omega t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ 0 \end{bmatrix} \tag{10}$$

This rotates the α and β reference axes around the 0 axis at the same frequency as the sinusoids that define the phasors. The effect of this rotation is that the periodic signals become DC values. The full d-q-0 transformation and its inverse are shown below:

$$\begin{bmatrix} d \\ q \\ 0 \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \cos\theta & \cos(\theta - \frac{2\pi}{3}) & \cos(\theta + \frac{2\pi}{3}) \\ \sin\theta & \sin(\theta - \frac{2\pi}{3}) & \sin(\theta + \frac{2\pi}{3}) \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad (11)$$

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \cos\theta & -\sin\theta & \frac{\sqrt{2}}{2} \\ \cos(\theta - \frac{2\pi}{3}) & -\sin(\theta - \frac{2\pi}{3}) & \frac{\sqrt{2}}{2} \\ \cos(\theta + \frac{2\pi}{3}) & -\sin(\theta + \frac{2\pi}{3}) & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} d \\ q \\ 0 \end{bmatrix} \quad (12)$$

In the case of unbalanced three-phase systems, the three-phase transformations shown above are not as effective because they do not help to eliminate the number of variables used to define the system. This is because the 0 axis contains a non-zero value. One approach to deal with this problem is the use of single-phase d-q transformations [**3**]. The α-β to d-q transform in this case is a straightforward modification of the above equation and is given by:

$$\begin{bmatrix} d \\ q \end{bmatrix} = \begin{bmatrix} \cos(\omega t) & \sin(\omega t) \\ -\sin(\omega t) & \cos(\omega t) \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \quad (13)$$

The initial transformation process is different from (9). In order to establish the α- and β-axis values for a single-phase system the value used for the α axis is the present value of the function that defines the ac waveform, such as $sin(\omega t)$, and the β axis value is the value of that same function but lagged for a quarter cycle; that is, $sin(\omega t - \frac{\pi}{2})$. This gives the magnitude and phase information for the ac waveform [**3**].

2.3 Second-Order Generalized Integrator (SOGI)

A SOGI is actually the transformation of an integrator in the dq reference frame into its corresponding operation in the stationary αβ reference frame. This is done by transforming

$$y^{dq} = y^{+dq} + y^{-dq} = \left(\frac{1}{s}\right)(x^{+dq} + x^{-dq}) \tag{14}$$

into the stationary reference frame [5]. The results of the transformation are:

$$\frac{y^{\alpha\beta}}{x^{\alpha\beta}} = \frac{1}{(s - j\omega_0)} + \frac{1}{(s + j\omega_0)} = \frac{2s}{(s^2 + \ \omega_0)} \tag{15}$$

where y^{+dq} and y^{-dq} represent the output in the positive and negative sequence rotating reference frames. The frequency response of this system is illustrated in Fig. 2-5Figure 2-5. This shows that the SOGI has infinite gain at 60 Hz and -60 Hz. By using negative feedback the SOGI can be used to track an input signal and as outputs, reproduces the input signal along with a signal of identical amplitude but lagging 90 degrees.

In Matlab/Simulink™, the creation of an orthogonal signal for use in a single-phase d-q transformation is as simple as using a delay block with a step of a quarter cycle. The same operation in the DSP would take up a large amount of space. For a sampling rate of 10 kHz, a quarter-cycle delay would require the saving of 2500 numbers, which takes up a lot of memory especially with a word length of 32 bits, so it is not very efficient.

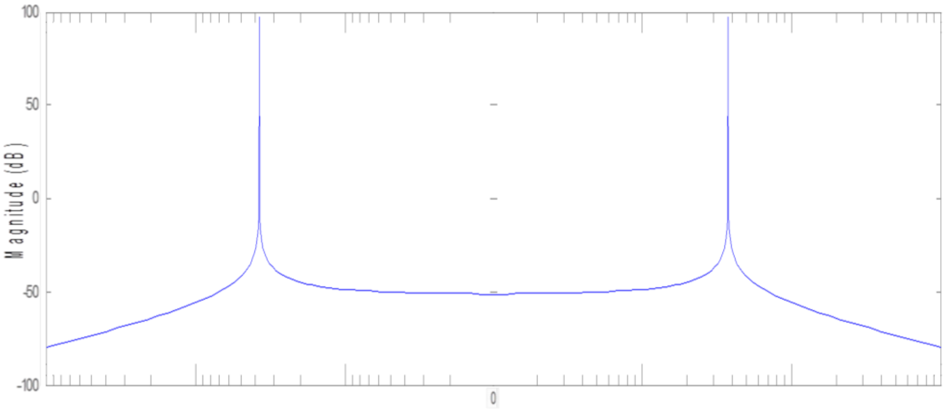


Figure 2-5. Frequency response of the SOGI

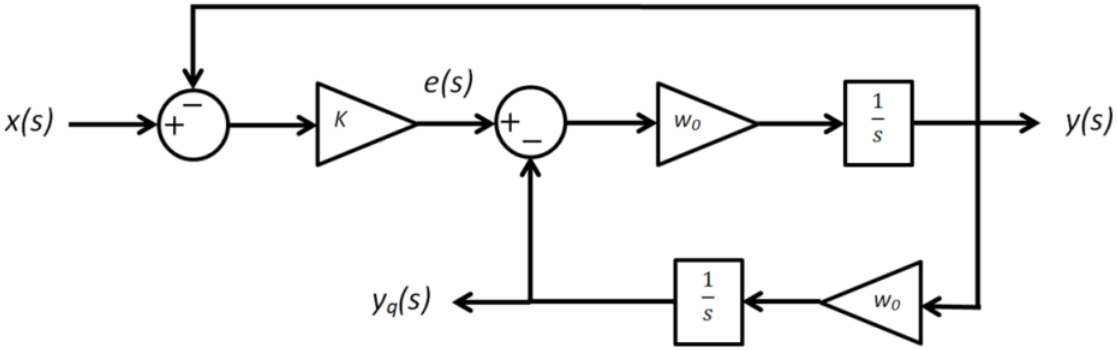


Figure 2-6. Block diagram for the SOGI-QSG

This is where a second-order generalized integrator quadrature signal generator (SOGI-QSG) can be used. This configuration of the SOGI takes in a sinusoidal input signal and then outputs two other sinusoidal signals. One of them is the fundamental frequency sinusoid component of the input and the other output is fundamental frequency sinusoid component of the input but delayed by a quarter cycle. The transfer functions for each of these outputs are:

$$\frac{y(s)}{x(s)} = \frac{K\omega_0 s}{s^2 + K\omega_0 s + \omega_0^2} \tag{16}$$

$$\frac{y_q(s)}{x(s)} = \frac{K\omega_0^2}{s^2 + K\omega_0 s + \omega_0^2} \tag{17}$$

Where ω_0 is the frequency to which the SOGI is tuned. Fig. 2-6Figure 2-6 contains the block diagram of the SOGI-QSG [**5**].

2.4 Design of LCL Filters

An LCL filter has become more commonly used in grid interfacing systems due to its superior current ripple attenuation when compared to the conventional L or LC filters. An L filter is simple, but only has a -20 dB/dec frequency response for the entire frequency spectrum. The LCL filter matches the performance of an L filter for lower frequencies, but can achieve a 60 dB/dec attenuation at higher frequencies, allowing for smaller magnetic components.

The basic structure of the LCL filter is shown in Fig. 2-7Figure 2-7. Because the design of this filter is based on the attenuation of harmonics that arise from active grid-connected devices the value of v_g is considered to be zero as the grid voltage is considered to be a pure 60 Hz voltage source. The output of the inverter is a PWM voltage and the filter is used to eliminate high-frequency current components. This leads to following transfer function [**6**]:

$$T_{LCL}(s) = \frac{i_g}{v_{inv}} = \frac{1}{L_{inv}C_f L_g s^3 + \left(L_{inv} + L_g\right)s} \tag{18}$$

The problem with this response characteristic is that a resonance occurs that might make the LCL filter actually produce worse current ripple in the output of the inverter. A solution for eliminating this resonance is the use of the resistor, R_f in series with the shunt capacitor to lessen the effects of the resonance, but unfortunately results in higher system losses. As a tradeoff for using a passive damping resistor, the 60 dB/dec attenuation after the resonant frequency reduces to 40 dB/dec. The transfer function that takes into account the effect of the damping resistor is given by:

$$T_{LCL}(s) = \frac{i_g}{v_{inv}} = \frac{C_f R_f s + 1}{L_{inv}C_f L_g s^3 + C_f\left(L_{inv} + L_g\right)R_f s^2 + \left(L_{inv} + L_g\right)s} \tag{19}$$

In order to avoid significant change in the power factor of the filter, the parallel capacitor is

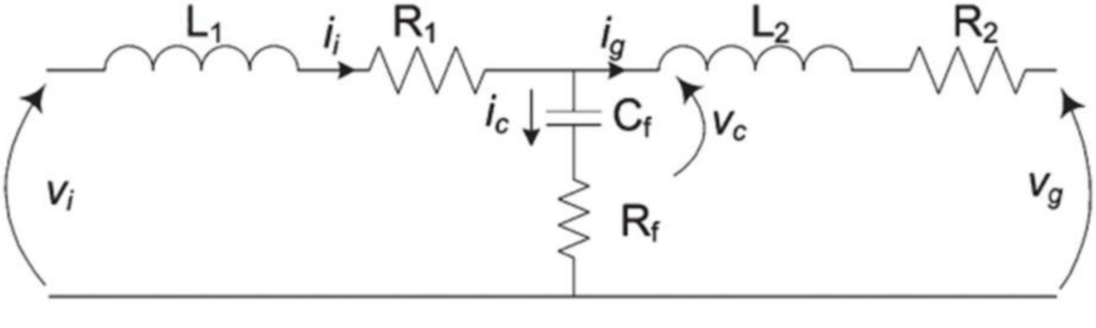


Figure 2-7. Schematic for an LCL filter **[**6**]**

limited to a percentage of the base capacitance of the system [**6**]. This requires knowledge of the power and voltage ratings of the inverter as follows:

$$C_f = (0.02)\frac{1}{\omega X_{base}} = (0.02)C_b = (0.02)\frac{P_{inv}}{\omega V_{lg}^2} \tag{20}$$

If this capacitor is to be connected in a delta configuration then line-to-line voltage V_{LL} would be used instead of line-to-ground voltage V_{LG}. This derivation of the LCL, instead of being designed based on a cutoff frequency to eliminate specified harmonics, is designed to attenuate the overall ripple current amplitude. The inverter-side inductor is determined from an initial ripple attenuation, which is a reduction to 20% as follows [**6**]:

$$\Delta i = \frac{V_{DC}}{8L_{inv}f_{sw}} \leq (0.20)I_{max} \tag{21}$$

The value of the grid side inductor is determined once again by using required ripple current reduction, which is 10%, so resulting attenuation is (20%)(10%) = 2%. Relevant equations are:

$$L_g = rL_{inv} \tag{22}$$

$$\frac{i_o(h_s)}{i_{inv}(h_s)} = \frac{1}{|1 + r(1 - L_{inv}C_b(0.02)(\omega_{sw})^2)|} \tag{23}$$

Due to the effect of the additional inductor the filter actually has a resonant frequency of its own at:

$$\omega_{res} = \sqrt{\frac{L_{inv} + L_g}{L_{inv} \times L_g \times C_f}} \tag{24}$$

This resonant frequency should be chosen at most to be half of the switching frequency of the inverter and at least ten times greater than the fundamental frequency [**6**]. This keeps the filter from amplifying switching noise as well as the lower order harmonics that are potentially produced in a single-phase inverter. In addition to choosing the appropriate resonant frequency a resistor is added in series with the parallel capacitor in order to attenuate the resonance. The value of this resistance is given by:

$$R_{damp} = \frac{1}{3\ x\ \omega_{res} \times C_f} \tag{25}$$

The factor of 3 in the denominator would be eliminated if the resistor is placed in delta configuration.

2.5 Pulse-Width Modulation Algorithm

A sinusoidal pulse width modulation (SPWM) is employed in order to produce the sinusoidal output voltage with an inverter [**7**]. This technique compares a sinusoidal control signal which under normal operating conditions has an amplitude of less than 1. This control signal is compared to a triangular waveform with an amplitude of 1. When the triangular waveform has a lower value than the control signal then the output of the PWM generator is 1 and when it is higher the output of the PWM generator is 0. When this output of the inverter is filtered appropriately the resulting output is a sinusoidal voltage with the same frequency magnitude as the control signal sent to the PWM generator. The frequency of the triangular waveform corresponds to the switching frequency of the inverter. These signals can be applied to the full-bridge inverter in many different ways. An example of this is drawn in Fig. 2-8Figure 2-8.

In bipolar PWM switching one set of signals is sent to the top and bottom switches of the first leg of the inverter and the second leg receives the same signals but inverted. Thus, the output of the inverter can either be –V_{dc} or V_{dc}, where V_{dc} is the voltage of the dc bus. Unipolar PWM switching uses a PWM generator for each leg of the inverter; one generator receives the control signal unmodified and the other receives the same control signal but negated. This type of modulation means that the top two or bottom two switches in the inverter can be on at the same time, which means that the voltage across the load can be 0 V, V_{DC}, or $–V_{DC}$. Unipolar switching doubles the effective switching frequency of the inverter resulting in less harmonic currents.

2.6 Discretization of the UCSC control system functions

Once the UCSC control system has been verified using continuous-domain techniques, the system must be transferred to the discrete domain in order to apply it to a real-world prototype by transforming all control operations into their Z-domain equivalents. The general method for this application would be the unilateral z-transformation [**8**]:

$$X(z) = Z\{x(t)\} = \sum_{n=0}^{\infty} x(nT_s)z^{-n} \tag{26}$$

The continuous function x(t) is broken into discrete parts based on the sampling rate of the system T_s and z^{-n} is an operator that when applied to a variable refers to its value “*n*”

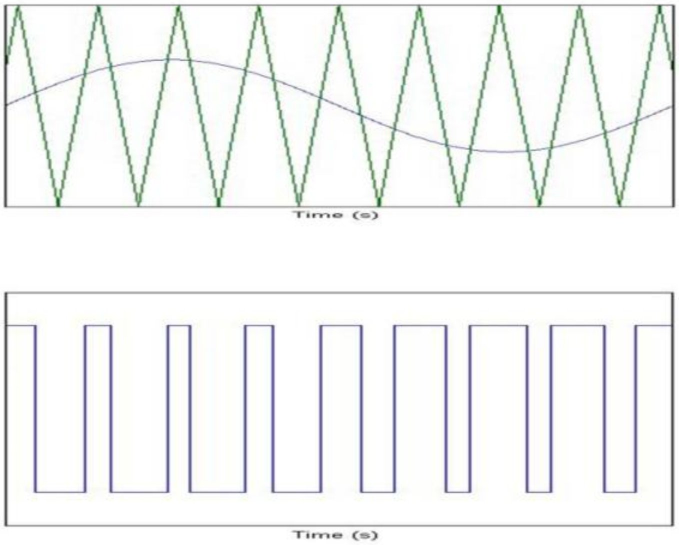


Figure 2-8. SPWM waveforms (top) and the resulting pulses (bottom)

sampling periods earlier. As computing an infinite series can be cumbersome and many control operations are done in the Laplace domain, it is much simpler to make use of s-to-z approximations. Five examples of Laplace-to-discrete transformations are given below [**9**]:

$$\text{Euler Forward: } s = \frac{1-z^{-1}}{T_s z^{-1}}$$

$$\text{Euler Backward: } s = \frac{1-z^{-1}}{T_s}$$

$$\text{Tustin (Trapezoidal): } s = \frac{2}{T_s}\frac{1-z^{-1}}{1+z^{-1}}$$

$$\text{Zero-Order Hold: } s = (1-z^{-1})Z\left\{\frac{H(s)}{s}\right\}$$

$$\text{Matched Pole-Zero: } z = e^{sT_s}$$

2.7 Concluding Remarks

This chapter presented the underlying theory behind the operation and control of the UCSC. This included the reference frames that the controller relies upon, how the controller operates, and the process of developing a controller that works in a real-world implementation. The next chapter will focus on building simulations and resulting results that were used to verify the performance of the UCSC.

2.8 References

[1] Manuel A. S. Tejada, "Unbalanced Current Static Compensator," Department of Electrical Engineering, University of Arkansas, Fayetteville, MS Thesis 2014.

[2] M. S. Huertas, "The use of power electonics for emulating power distribution feeders," Department of Electrical Engineering, University of Arkansas, Fayetteville, Masters Thesis 2008.

[3] U. A. Miranda, L. G. B. Rolim, and M Aredes, "A DQ synchronous reference frame current control for signle-phase converters," in *IEEE 36th Power Electronics Specialists Conference PESC'05*, Recife, 2005, pp. 1377-1381.

[4] P.C. Krause, O. Wasynczuk, and S.D. Sudhoff, *Analysis of Electric Machinery and Drive Systems*, 3rd ed. Somerset, United States of America: Wiley, 2013.

[5] C. A. Busada, S. Gomez Jorge, A. E. Leon, and J. A. Solsona, "Current controller based on reduced order generalized integrators for distributed generation systems," *IEEE Transactions Industrial Electronics*, vol. 59, no. 7, pp. 2898-2909, 2012.

[6] A. Reznik, M.G. Simoes, A. Al-Durra, and S.M. Muyeen, "Filter Design and Performance Analysis for Grid-Interconnected Systems," *Industry Applications, IEEE Transactions on*, vol. 50, no. 2, pp. 1225-1232, March-April 2014.

[7] N. Mohan, T. M. Undeland, and W. P. Robbins, *Power Electronics: Converters Applications and Design*, 3rd ed. New Jersey, United States of America: JW&S Inc, 2003.

[8] H. A. Helm, "The Z transformation," *The Bell System Technical Journal*, vol. 38, no. 1, pp. 177-196, January 1959.

[9] F. J. Rodriguez et al., "Discrete-time implementation of second order generalized integrators for grid converters," in *Proceedings 34th Annual Conference IEEE Industrial Electronics*, 2008, pp. 176-181.

CHAPTER 3

SIMULATION PROCESS AND RESULTS

3.1 Introduction

The system shown in Fig. 3-1 was implemented and tested in Matlab/SIMULINK™ in order to establish the plausibility of the proposed solution. This system is based upon the 6 MVA, 34.5 kV radial feeder case study described in Chapter 1 [**1**]. The inverters are rated 480 $V_{L\text{-}N}$, and have a current rating of 2048 A, giving each a 1 MVA rating.

Fig. 3-2 shows the simulation results of the case study from the model proposed in [**1**]. The UCSC is in 3 different modes of operation. In the first mode from $t = 0.03$ s to $t = 0.15$ s, the UCSC is compensating for the unbalanced loads. From $t = 0.15$ s to $t = 0.60$ s, the UCSC is fully operational. At $t = 0.40$ s, there is a step increase of 30% in one of the phase loads.

3.2 Realized Changes to Previously Suggested Simulation Model

Several changes were made to the original model proposed in [**1**] either to improve its performance or to reduce its computational load. These changes are addressed below.

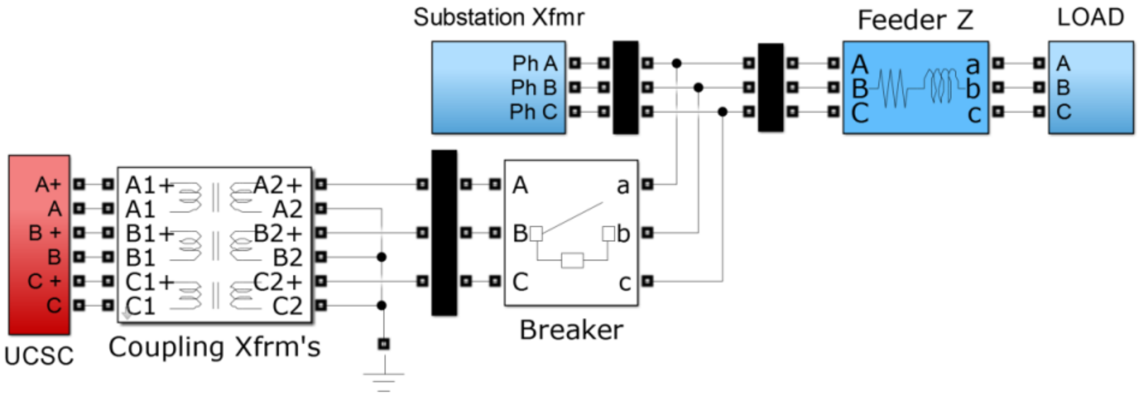


Figure 3-1. Matlab/SIMULINK™ UCSC system

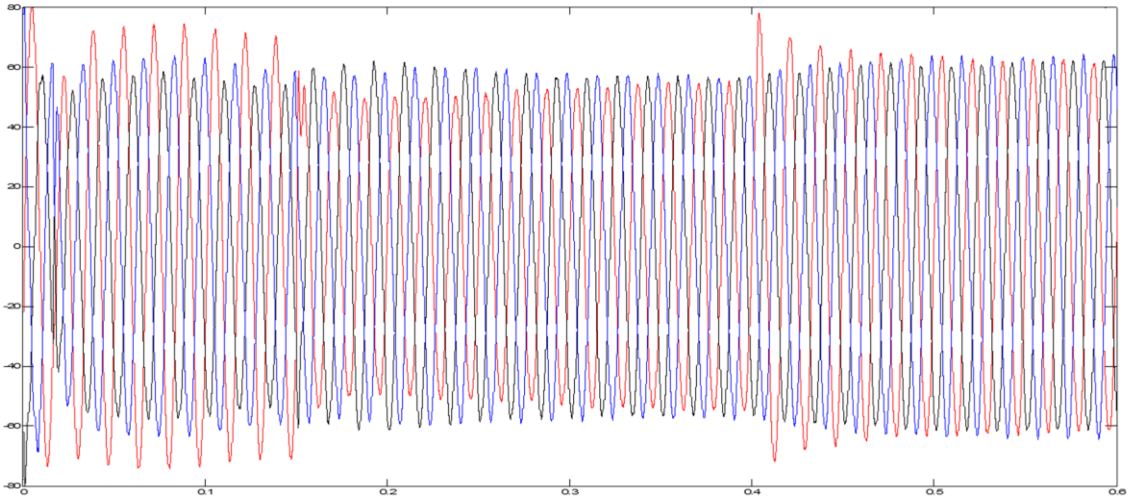


Figure 3-2. Substation currents from the previous model in different UCSC operating modes

3.2.1 PWM Control Signal Normalization

The previous model in [**1**] and the change made to it are shown in Fig. 3-3. If the control signal is normalized as in Fig. 3-3 (a), then the resulting control signal will contain the same 2nd harmonic component that is present in the dc bus voltage. This will result in a 3rd harmonic voltage component being produced in the output of the inverter because of the rotating reference frame. The control signal is now normalized using the instantaneous dc bus voltage to avoid this harmonic as in Fig. 3-3 (b).

3.2.2 Current Reference Generator

In the original model, displayed in Fig. 3-3, a reference current from the dc-bus voltage controller was added to the error between the d-axis inverter current, i_{gad}, and the d-axis current reference, i^*_{gad}. The result of this scheme was a “wind-up” of the PI controllers for the reference current and the voltage controller. An example of this is shown in Fig. 3-5 where the output of PI controllers continues to rise even when the UCSC is in steady-state operation.

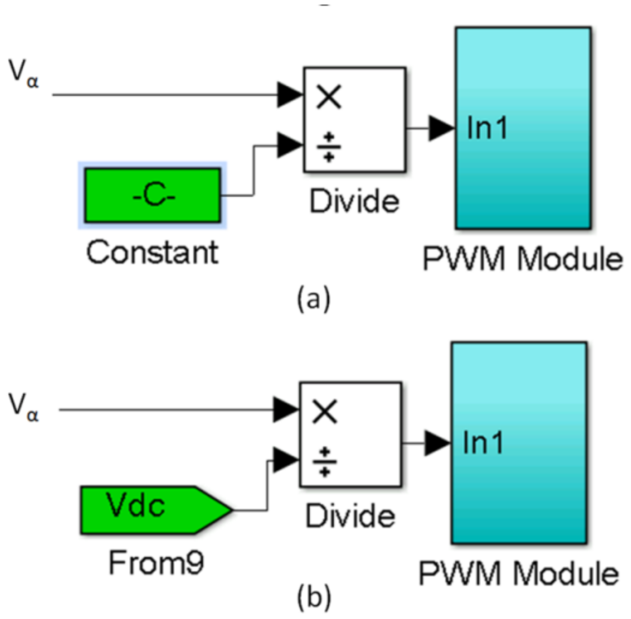


Figure 3-3. PWM control signal normalization

By adding now directly the dc-bus voltage error to the reference current error; this "wind-up" and extra DSP calculations were avoided. The PI controller for the voltage error was eliminated. The resulting output from the current reference PI controller now has the form illustrated in Fig. 3-6. The new system is shown in Fig. 3-7.

In addition to moving the voltage error, a notch filter at 120 Hz was added to the Vdc measurement used to compare to the reference. The elimination of the 2nd harmonic voltage ripple in the control system is desirable because (a) PI controllers are meant to work upon dc values and because (b) 2nd harmonics in the d-q reference frame become 3rd harmonic components in the control waveforms sent to the PWM generator. This results in a 3rd harmonic component in the output voltage of the single-phase inverters used in the UCSC. The voltage measurement before and after the notch filter is shown in Fig. 3-8.

Another change within the reference current generator refers to the sourcing of the

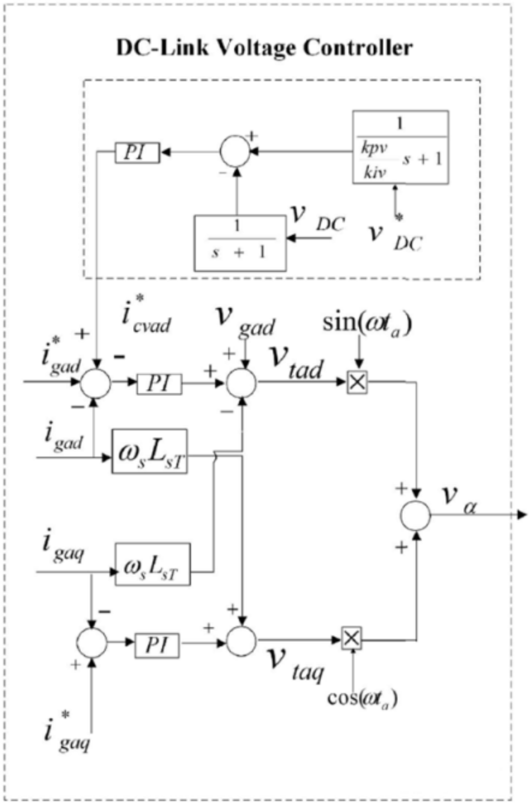


Figure 3-4. Previously suggested controller [1]

current measurements. Instead of measuring all the substation, load and inverter currents, as done previously, the model now just computes the inverter and loads current and uses KCL at the PCC to determine the substation currents. This saves computational time for the DSP to perform the SOGI and d-q transforms (associated with sampling these currents) that make up most of the DSP's calculations.

3.3 Modifying the Time-Domain Simulations to Reflect the Prototype Implementation

To better emulate the behavior of the prototype, the controller was modeled as it is implemented in the DSP and the start-up procedure is modified to be indicative of the laboratory testing. The sections below describe the realized changes.

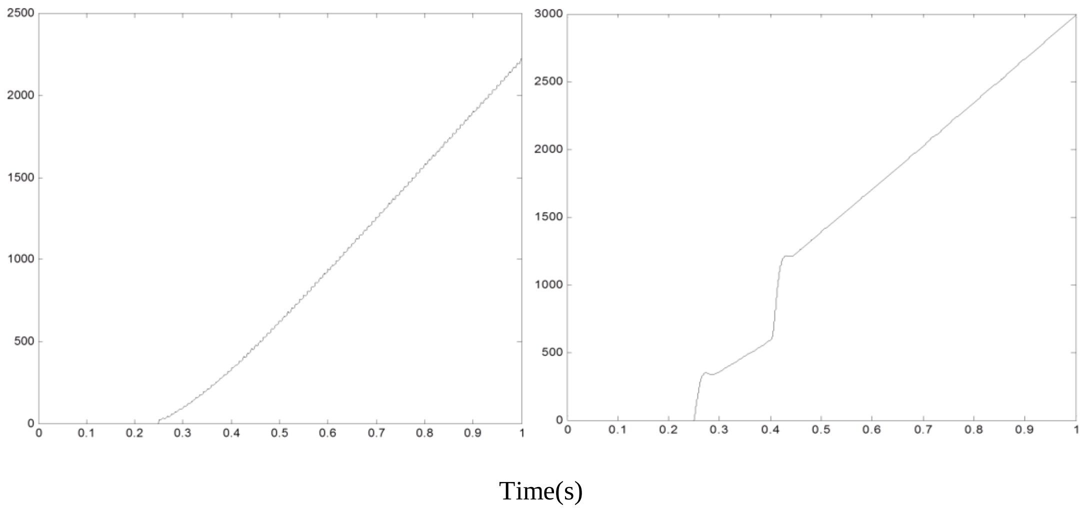


Figure 3-5. Outputs of the current reference and voltage PI controllers in the previous system

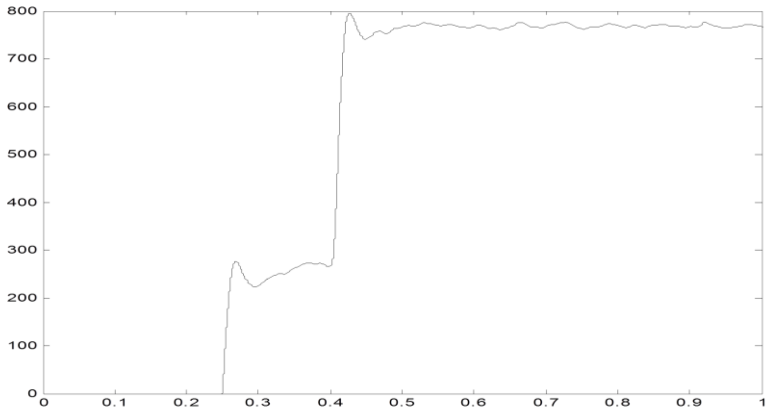


Figure 3-6. Current reference PI controller output for new system

3.3.1 Sensing Circuitry

Fig. 3-9 is the block diagram that emulates the current and voltage measurements as implemented in the actual system. The first gain is the conversion ratio between the actual current or voltage being measured by the sensors and its output, followed by the voltage output of the conditioning circuits that are connected to the sensor’s output. The filter after this first gain is analog filtering as a result of the feedback capacitor in the operational amplifier based conditioning circuit. The second gain is the compensation due to any loss in gain from the effects

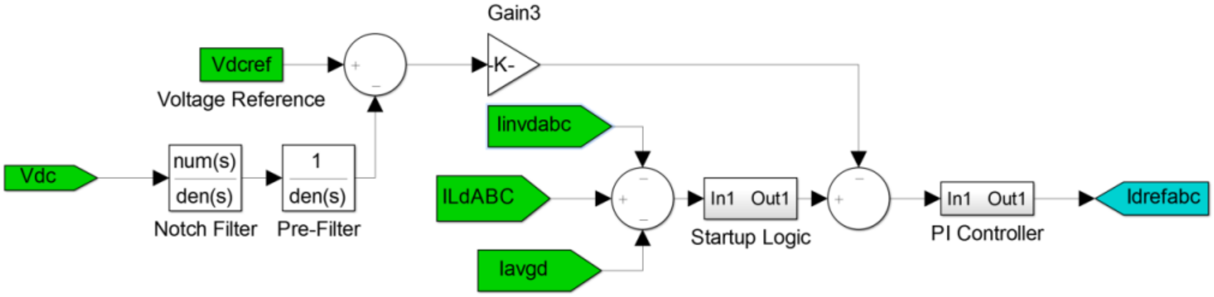


Figure 3-7. Reference generator for the current controller

of the analog filtering.

The uniform random number block generates noise to mimic the noise experienced in the actual ADC measurements. The quantizer block mimics the ADC characteristics of the DSP where the ADC module can only detect 4096 different discrete voltage values between the digital ground and supply voltage (3.0 V). The last block is the digital filter used to eliminate high-frequency noises from the ADC measurements. This digital filter averages the last two samples. This is the discrete domain equivalent to a single-pole continuous transfer function with a cutoff frequency at half of the sampling frequency.

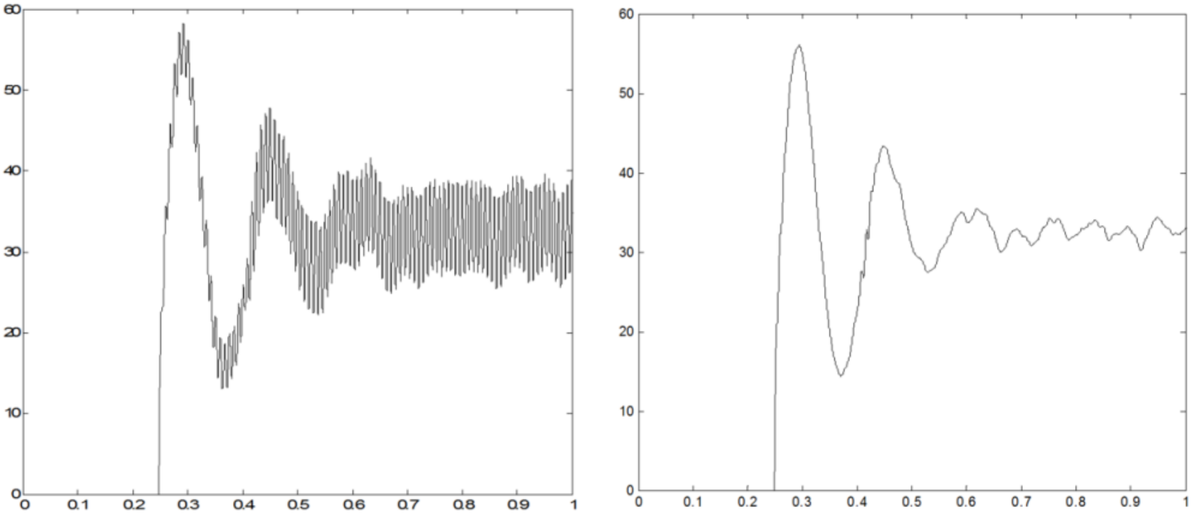


Figure 3-8. Effect of the notch filter before (left) and after (right)

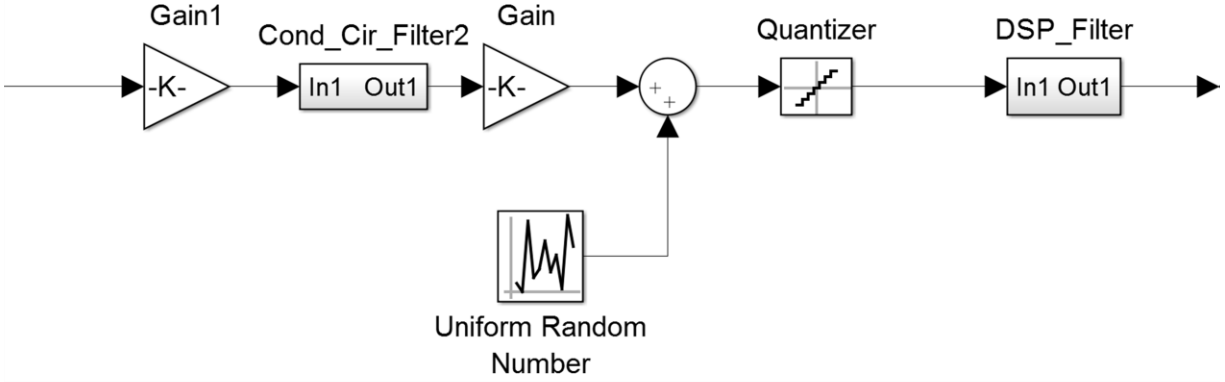


Figure 3-9. Measurement filtering and conditioning used in the simulations

3.3.2 Inverter Devices and Filter Components

When validating the control of the system, semiconductor devices near to ideal are used in the simulations. But to get the appropriate behavior of the inverters, IGBT and diode models that better describe the behavior of the actual IGBT and antiparallel diode co-package found in the prototype are used as shown in Fig. 3-10. The most important parameters are the minimum voltage drop across the IGBT V_{CE}, the on-state resistance of the IGBT r_{on}, the minimum voltage drop across the antiparallel diode V_d, and the on-state resistance of the antiparallel diode r_{don}. Though the models used also allow for package inductances, turn-off and tail current times, and parasitic snubber, these are not as important for a full system simulation as the time step required to witness the phenomenon that these parameters cause is too small to efficiently test the model. These are important when looking at the effects of parasitics, which will be described later.

The prototype IGBTs are the FGH20N60SFDTU from Fairchild Semiconductor whose main parameters are listed in Table 3-1. Fig. 3-11 shows the dialog box for the IGBT and antiparallel diodes used in the simulations. It is also important to use accurate values for the inductances and resistances in the filter because of the decoupling terms required in the current controller for each phase. This includes finding the actual inductances and winding

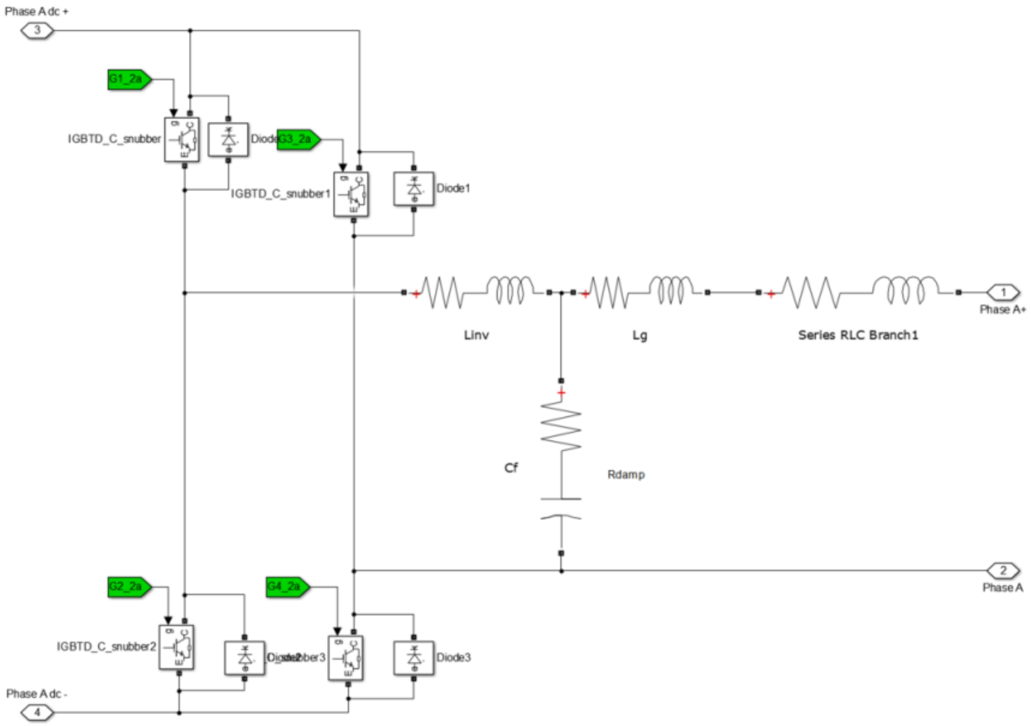


Figure 3-10. Inverter implemented in SIMULINK™

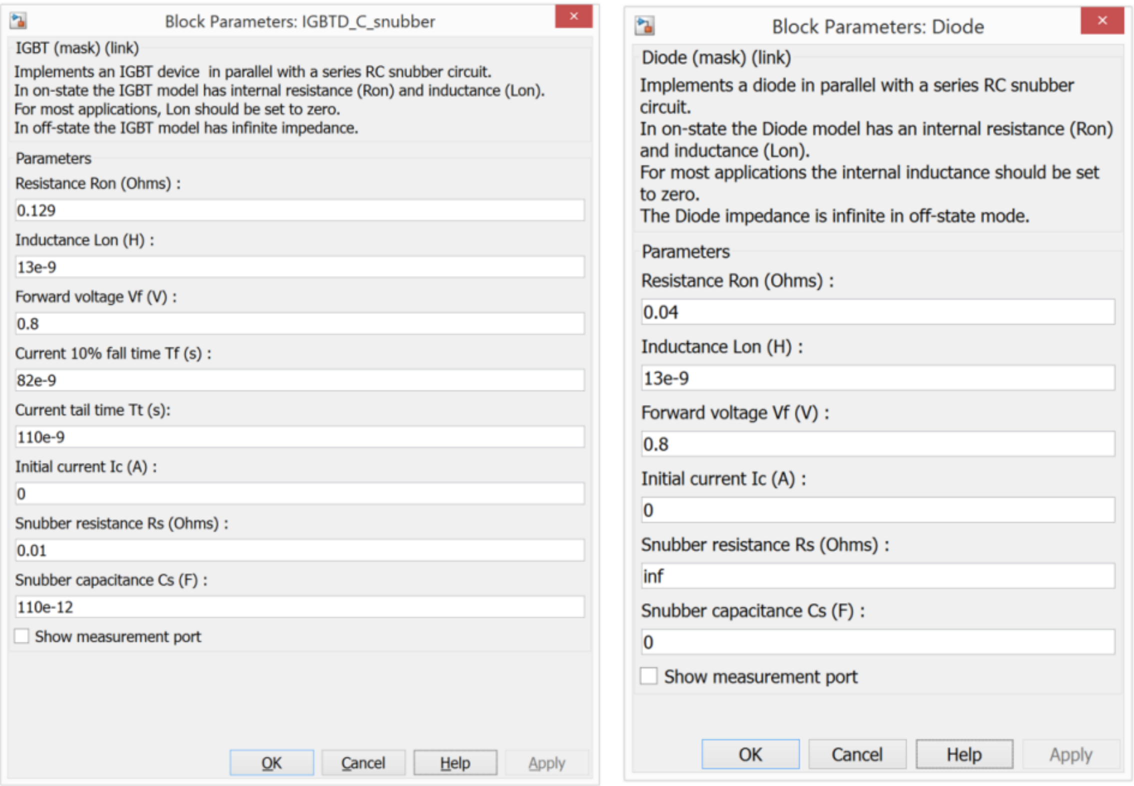
resistances of the filter inductors. There is a slight variation in the inductance between each of the inductors because the inductors were handmade; an impedance meter was used to test the inductance of each of the inductors.

3.3.3 Pre-Filtering of d-q Signals

In [**1**], the output of the d-q transformations were first filtered through single-pole transfer functions such as $\frac{1}{\tau s+1}$. This introduced an additional measurement delay on the signals used in

Table 3-1. Parameters of the prototype IGBTs

Property	**Value**
V_{CE} Breakdown Voltage	600 V
Rated Collector Current @ 25°C	40 A
Maximum Power @ 25°C	165 W
Operating Temperature	-55°C to +150°C



Block Parameters: IGBTD_C_snubber

IGBT (mask) (link)

Implements an IGBT device in parallel with a series RC snubber circuit.
In on-state the IGBT model has internal resistance (Ron) and inductance (Lon).
For most applications, Lon should be set to zero.
In off-state the IGBT model has infinite impedance.

Parameters

Resistance Ron (Ohms) : 0.129

Inductance Lon (H) : 13e-9

Forward voltage Vf (V) : 0.8

Current 10% fall time Tf (s) : 82e-9

Current tail time Tt (s): 110e-9

Initial current Ic (A) : 0

Snubber resistance Rs (Ohms) : 0.01

Snubber capacitance Cs (F) : 110e-12

□ Show measurement port

Block Parameters: Diode

Diode (mask) (link)

Implements a diode in parallel with a series RC snubber circuit.
In on-state the Diode model has an internal resistance (Ron) and inductance (Lon).
For most applications the internal inductance should be set to zero.
The Diode impedance is infinite in off-state mode.

Parameters

Resistance Ron (Ohms) : 0.04

Inductance Lon (H) : 13e-9

Forward voltage Vf (V) : 0.8

Initial current Ic (A) : 0

Snubber resistance Rs (Ohms) : inf

Snubber capacitance Cs (F) : 0

□ Show measurement port

Figure 3-11. Dialog box for the IGBTs and diodes used in the SIMULINK™ simulations

the control algorithm. These filters that can slow down the settling and the response times for any changes in the load currents were removed to avoid these affects.

3.3.4 Coupling Transformers

The transformers in the simulations were characterized using the short-circuit and dc resistance test on the transformers used in the lab testing. The short-circuit test consisted of shorting the low-voltage side of transformer and then increasing the high-side voltage until the rated current. Then, the leakage reactance can be calculated using the current and voltage on the high side. The winding resistance is found by applying a dc voltage to both sides of the transformer and calculating the dc resistance based on the current that flows. The resistance of the windings for a 60 Hz waveform can be approximated by the dc resistance because the skin effect is negligible at low frequencies unless the conductors used have a very large cross-sectional area.

3.3.5 Discrete Phase-Lock Loop Algorithm

The phase-lock loop algorithm implemented in the DSP is summarized in Fig. 3-12, where a normalized grid voltage is the input. Then, a single-phase d-q transformation is performed based on the previously estimated grid angle. If the grid angle estimation is correct, then the d-q transformation will result in a null q component and the d component is the peak magnitude of the grid voltage. Assuming that the estimated grid angle is wrong, the q component has a non-zero value and becomes an error signal. This error signal is processed through a PI controller that acts as a low-pass filter (LPF) to eliminate any measurement noise [**2**]. The transfer function for the LPF is given by:

$$Y_{LPF}(z) = \frac{\left(\frac{2K_P + K_i t_{samp}}{2}\right) - \left(\frac{2K_P + K_i t_{samp}}{2}\right) z^{-1}}{1 - z^{-1}} \qquad (1)$$

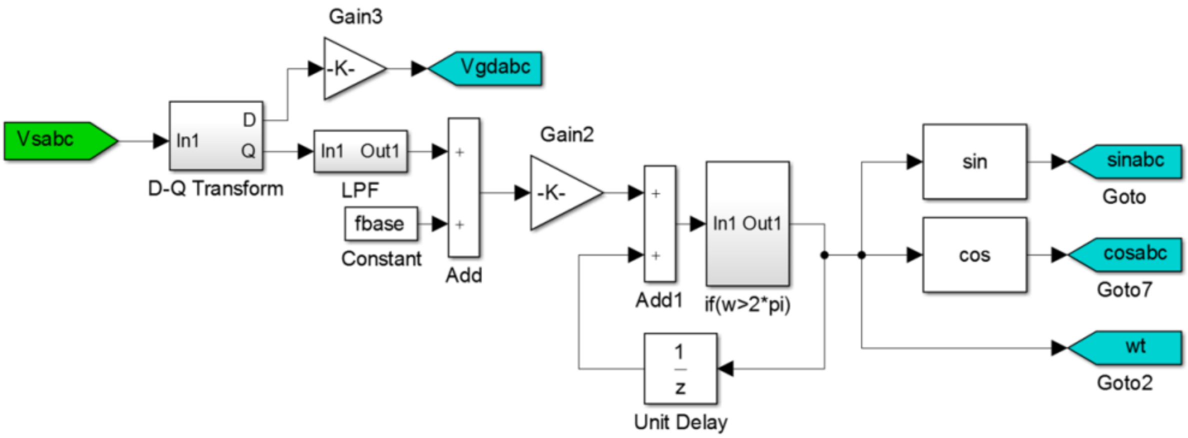


Figure 3-12. Discrete phase-lock loop block diagram

where K_P is the proportional gain, K_i is the integral gain, and t_{samp} is the sampling period.

This filtered error signal is added to the nominal grid value to either increase or decrease the rate of change of the estimated phase angle. This causes the derivative of the phase angle to increase or decrease in order to eliminate the q component. The most recently calculated sine and cosine values are used in the next cycle of the algorithm [**2**]. The sine and cosine functions in the DSP are actually computed using a lookup table (LUT) that includes 512 values to define each quarter of the sine and cosine functions; so, it is assumed that there is a negligible amount of error between the actual function and the values used by the DSP.

3.3.6 Discrete d-q Transformation

The d-q transformation as performed in the DSP is illustrated in Fig. 3-13. A SOGI-QSG is used to produce a quadrature signal for the single-phase transformation [**2**]. The outputs of the SOGI-QSG are used along with the sine and cosine of the grid voltage angle to produce the d- and q-axis outputs. The discrete transfer functions from the Tustin approximation for both outputs of the SOGI-QSG are:

$$\frac{y(z)}{x(z)} = \frac{b_0 + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{2}$$

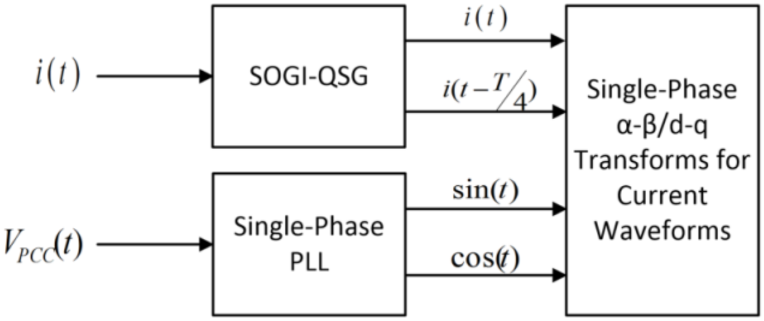


Figure 3-13. D-Q Transformation

$$\frac{y_q(z)}{x(z)} = \frac{qb_0 + qb_1 z^{-1} + qb_2 z^{-2}}{1 - qa_1 z^{-1} - qa_2 z^{-2}} \tag{3}$$

where, b_2, a_1, a_2, qb_0, qb_1, qb_2, qa_1, and qa_2 are constant that are specified in:

$$b_0 = \frac{x}{(x+y+4)}\ , b_2 = \frac{-x}{(x+y+4)} \tag{4}$$

$$a_1 = \frac{2(4-y)}{(x+y+4)}\ , a_2 = \frac{x-y-4}{(x+y+4)} \tag{5}$$

$$qb_0 = \frac{1}{2} qb_1 = qb_2 = \frac{ky}{(x+y+4)}, \tag{6}$$

$$x = 2k\omega_n T_s\ , y = (\omega_n T_s)^2 \tag{7}$$

using the fundamental frequency ω_n, sampling period T_s, and the forward gain k.

3.4 Simulation Results of the New Model

The substations currents from the new model in the continuous domain are shown in Fig. 3-14. This includes most of the changes mentioned above; however, it excludes all discretization modifications. As in Fig. 3-2 the waveforms encompass three different operating modes of the UCSC. From $t = 0.00$ s to $t = 0.15$ s, the UCSC is not operational. From $t = 0.15$ s to $t = 0.25$ s, the UCSC is compensating for the reactive power of the load. From $t = 0.25$ s to $t = 0.60$ s ,the UCSC is fully operational. At $t = 0.40$ s there is a step increase of 30% in one of the phase loads. The substation currents in steady state are brought to a *UBF* of 0.497%, where *UBF* is calculated using:

$$UBF = \frac{|negative\ sequence|}{|positive\ sequence|} x100\% \tag{8}$$

Using the rated currents and voltages for which the UCSC prototype was designed, discrete-domain simulations were run to predict the system performance. Demonstrating

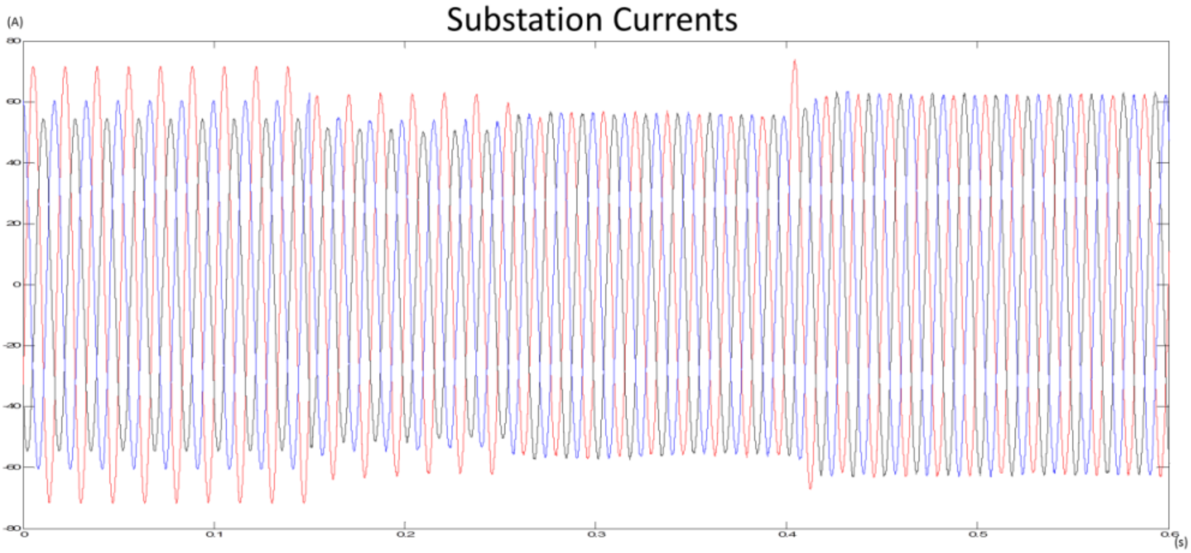


Figure 3-14. Substation currents from new continuous model in different UCSC operating modes

the predicted prototype performance in the same operating modes as the case-study simulations, the substation currents are shown in Fig. 3-15. From $t = 0.20$ s to $t = 0.30$ s, the UCSC is not operational. From $t = 0.30$ s to $t = 0.45$ s, the UCSC is compensating for the reactive power of the load. And from $t = 0.45$ s to $t = 0.80$ s, the UCSC is fully operational. Some neutral current

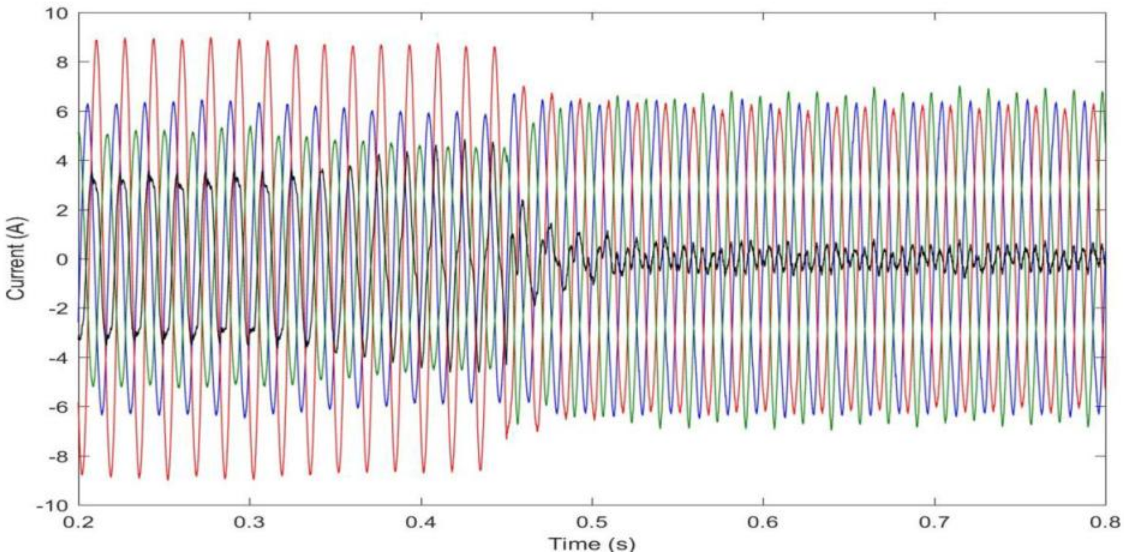


Figure 3-15. Substation currents from the discrete-domain prototype simulation

exists due to the non-idealities that were introduced into the model. The most significant component of this neutral current is the third harmonic produced from dead time effect in the output of the UCSC inverters. In steady state, these simulations yielded an average *UBF* of 0.37%, down from an *UBF* of 18.2% when the UCSC is not operational.

3.4.1 Effect of Incorrect Decoupling Terms

Within the current controller as described in Chapter 2 there are decoupling terms that allow for separate control of the d-axis and q-axis currents. Fig. 3-16(a) and Fig. 3-16(b) show the substation currents from the laboratory prototype simulations with and without the decoupling terms, respectively, during a large load change on one phase. Though they both perform as expected in steady-state, the simulation with the decoupling terms shows better transient response. The absence of the decoupling terms is an extreme case, but a diminished transient performance can also be caused by incorrect assumption about filter inductance values.

3.5 Conclusions

Improvements made upon the UCSC controller improved its dynamic response and made it more stable. Through several other modifications, such as the discretization of the controller

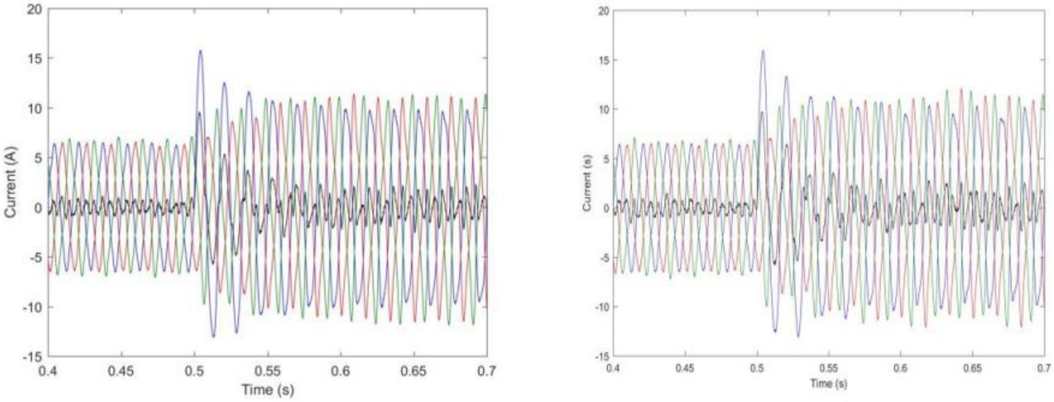


Figure 3-16. Transient reponse of UCSC with (a) and without (b) decoupling terms

and the inclusion of dead time, a model that was more indicative of a real implementation of the UCSC was created. This model showed the compensator was still effective at reducing the negative- and zero-sequence fundamental current components as seen by the substation.

With the viability of the proposed solution confirmed by simulations the next step is the construction of a prototype that can be tested in a laboratory setting. Determining the effect of different component choices on the performance of the UCSC becomes easier with a simulation that emulates the non-ideal behavior of an actual prototype. The simulation can help to validate component choices before they are installed on the prototype.

3.6 References

[1] Manuel A. S. Tejada, "Unbalanced Current Static Compensator," Department of Electrical Engineering, University of Arkansas, Fayetteville, Masters Thesis 2014.

[2] Texas Instruments, C28x Solar Library Module User's Guide, 2014.

CHAPTER 4

DESIGN OF THE SCALED-DOWN UCSC PROTOTYPE

4.1 Introduction

The objective of this chapter is to explain the process for designing and constructing the UCSC scaled-down prototype based on the 34.5 kV, 6 MVA case study. Thus, this chapter begins with the procedure used for scaling down the case study introduced in Chapter 1, and includes component selections as well as the printed circuit board layout. A comprehensive design process for AC inductors is also included because the required inductors are not commercially available.

4.2 Design of the Prototype Power Stage

4.2.1 Scaling-Down the Case Study

A per-unit scaling is used to reduce the power rating of the case study in Chapter 1 to a level that can be produced in a laboratory environment. A base power $S_{base} = 6$ MVA along with a base voltage $V_{base} = 34.5$ kV are used to convert the case study parameters to their per unit equivalents. A base power $S_{base,lab} = 10$ kVA and a base voltage $V_{base,lab} = 208$ V were chosen based on the power delivery and voltage capabilities in the lab. These base values are then used to convert the per unit values from the case study into the real laboratory system values. The power rating for each of the UCSC prototype inverters is chosen based on the active and reactive power demand of the calculated loads. A summary of the results from these calculations is presented in Table 4-1. From these power ratings, each inverter was designed for 1 kVA at a voltage of 120 $V_{L\text{-}N}$.

Table 4-1. Load power ratings for the case study and scaled-down prototype

Parameter	Cast Study	Laboratory
P_{Load1} (kW)	877	1.461
Q_{Load1} (kVAR)	500	0.833
P_{Load2} (kW)	707	1.178
Q_{Load2} (kVAR)	300	0.500
P_{Load3} (kW)	753	1.255
Q_{Load4} (kVAR)	300	0.666

4.2.2 Calculations for the dc Link Capacitor Sizing

Though the UCSC is effectively a three-phase inverter, due to the unbalance in the loads and the varying phase shifts between the three phases, the $(2f_1)$ voltage ripple in the dc bus is still an issue. For a single-phase inverter, the dc capacitor current has a dc component as well as a $(2f_1)$ component [**1**]. In the case of the UCSC, the dc current component is small, flowing into the capacitor to maintain its voltage because there is no power generation on the dc bus.

An assumption made for the UCSC system is that each phase compensate for approximately the same reactive power. Thus, the imbalance between the inverter currents will come from the active current component differences between each phase. Using data from the case study in [**2**], it is assumed that the active current between phase differs at most by 0.15 pu which results in 4.0A for the scaled-down laboratory system. The worst case scenario for ripple current on the dc bus is when one inverter is producing a current of 0.15 pu more than the other two. The equivalent current ripple of the dc bus is that of a single-phase inverter with an output current of 0.15 pu given by:

$$i_{ripple} = \frac{V_0 I_0}{V_d} \cos(2\omega_1 t - \varphi) \tag{1}$$

where V_0 is the peak of the sinusoidal output voltage of the inverter, I_0 is the peak output current of the inverter, and V_d is the dc-bus voltage applied to the inverter. The capacitance C_{dc} is calculated using:

$$\left|i_{ripple}\right| = C_{dc}\Delta V 2\omega_1 \tag{2}$$

where ΔV and is chosen as the desired voltage ripple. The UCSC dc-bus capacitor parameters are given in Table 4-2.

4.2.3 Design of the Gate Driving Circuit

Each IGBT in the UCSC is driven with a bipolar power supply of +17 V and -8.7 V. So +17 V is applied to the gate of the IGBT during turn-on and -8.7 V at turn-off. This, along with a gate driving IC with relatively large current capability, is required to charge and discharge the gate capacitance quickly; translating into faster turn-on and turn-off times. For each IGBT used in the prototype, the CUI Inc VQA-S15-D17-SIP power supply and the Silicon Labs SI8261BCC-C-IS isolated gate driver was used. An isolated gate driver was chosen to reduce the part count and reduce the propagation delay.

The gate driver IC can source up 1.8 A and sink up to 4.0 A from the same output pin. To utilize the larger current sinking ability from the same output pin two separate gate resistors (20 Ω and 6.8 Ω) were used in series with diodes. During turn-on the gate of the IGBT will be driven through a 20 Ω resistor while the current through the 6.8 Ω resistor is blocked by a reverse-

Table 4-2. DC Link Capacitor Parameters

Parameter	Value
Maximum Peak Ripple Current	4 A
Desired Peak Voltage Ripple	1% of rated bus
Capacitance	2200 μF
dc-bus voltage rating	250 V

biased diode. At turn-off the gate driver sinks current through the 6.8 Ω resistor while the current through the 20 Ω resistor is blocked by a reverse-biased diode.

Capacitors rated 25 V and 100 uF were used at the input of the power supply as well as at both outputs. This was done to provide an additional source of charge to compensate for the large currents drawn by the gate driver IC. Additional ceramic capacitors with values of 1.0 μF, 0.1 μF and 0.01 μF were added as close to the gate driver IC as possible to provide compensation for high frequency voltage oscillations experienced during switching transitions. Bleeding resistors that draw 1% of the rated current were also added from supply to ground at the output of the power supply to help with no-load voltage regulation of the power supply.

Voltage protection at the gate is important because exceeding the voltage rating of the gate oxide used in the IGBT (±20 V relative to the emitter) can cause catastrophic device failure. Between gate and emitter of the IGBT two zener diodes are placed in series with a common cathode configuration. One zener has a breakdown voltage of 18 V for over-voltage protection during the on-time of the IGBT and the other has a breakdown voltage of 9 V to protection against undervoltages during the off-time. The diodes are in parallel with a 50 kΩ bleeding resistor placed to help drain charge from the gate capacitance during turn-off as well as protect the IGBT from false triggering due to current oscillations at the gate.

4.2.4 The Effect of Parasitics on System Performance

Connecting components on a PCB or with cables introduces parasitic between the two points of contact. This can be in the form of a capacitance, such as the overlapping of traces between layers or crosstalk between two parallel traces on the same layer. It can also be inductance caused by the orientation of the trace itself or the self-inductance present in all conductors [**1**].

Another source of these parasitic inductances and capacitances comes from the devices and the packages in which they are held. The IGBTs used in the UCSC prototype have parasitic inductances from the leads that solder to the board and from the wire bonds used to connect these leads to the metallization on the substrate. In addition there are parasitic capacitances between the leads due to the characteristics of the semiconductor. When these are taken into account in the simulations they become the source of current and voltage oscillations or "ringing" present in the circuit. The next two subsections highlight the importance of reducing these parasitics through careful system packaging and printed circuit board design.

Power Stage Parasitics

A major concern is the voltage across the semiconductor devices used in the legs of the inverters in the UCSC. The IGBTs have a maximum rated voltage of 600 V that they can handle without breaking down. While the dc-bus voltage used in the prototype is only 250 V, the parasitic inductances can cause very large voltage spikes if countermeasures are not taken. These spikes come from energy stored in the parasitic inductances and the rate of change of the current [**1**]. A large voltage is developed at the collector terminal due to the abrupt change in current at the IGBT turn-off. If the inductance can be minimized then these effects might be negligible, but in many cases there is an unavoidable amount of parasitic inductance from the traces due to size constraints and technologies used. Appendix A explains some of the passive solutions used to mitigate this ringing.

Gate Driver Circuit Parasitics

Because of the short turn-on and turn-off times associated with IGBTs, gate driver circuits have unavoidably large changes in current that can lead to oscillations in the voltage applied to the gate of the IGBT [**1**]. There are two major sources of voltage spikes at the gate.

One is the inductance of the small section of conductor shared by the power stage and the gate driver. During the reverse recovery time of the antiparallel diode there is a significant change in current. This $\frac{di}{dt}$ across the emitter inductance can cause an oscillation in the voltage applied to the gate [**3**]. The other source are the oscillations that occur due to the interaction between the trace inductance and the Miller capacitance of the IGBT, C_{gc}. During the turn-on transient some of the current from the gate-driver IC goes into the C_{gc} as it discharges through the conduction channel of the IGBT. If the gate driver injects too much current too quickly the IGBT cannot respond quickly enough and large voltage swings will result.

Relative to the emitter terminal, most IGBTs can only handle ±20V before the gate oxide begins to breakdown. Devices are normally driven close to these maximum voltages in order to decrease their turn-on and turn-off times and because of this the voltage oscillations at the gate need to be minimized; otherwise, devices failure may occur.

4.2.5 Design of the Printed Circuit Board for the Power Stage

A four-layer printed circuit board (PCB) was designed in order to more easily integrate the gate drivers and IGBTs on the same board. Fig. 4-1 to Fig. 4-4 display the positive photoplots (in color) used for each of the layer of the PCB. These were created using Cadence® Allegro® PCB Designer software.

The bottom layer, shown in Fig. 4-1, contains the positive rail of the dc bus and the midpoints of each leg (outlined in green). The traces were made very wide in order to reduce the parasitic inductances between the dc-bus capacitor and each of three inverters. The negative rail of the dc bus is contained in the bottom-middle layer displayed in Fig. 4-2. This layer is almost all copper to shield the top and top-middle layers from EMI generated by the current in the bottom layer [**4**]. The plane used is split so that the return current to the capacitor does not

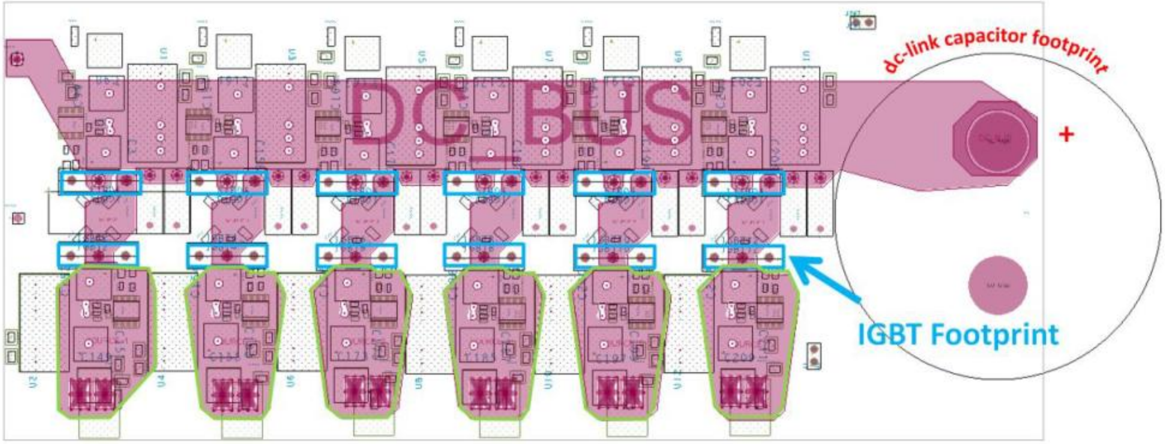


Figure 4-1. Bottom layer photoplot for the power stage PCB used in the UCSC prototype

flow in the outside portions of the plane. By doing this, these portions of the plane only act as shields and do not transfer any switching related EMI to the upper layers.

The traces drawn for the gate driving circuitry can be found in Fig. 4-3 and Fig. 4-4. The ICs mentioned in section 4.2.6, which are represented by the footprints within the indicated are in Fig. 4-3 and Fig. 4-4, are placed near the IGBT to reduce the length of the current loop used to charge the gate capacitor. This along with the use of wide traces reduces the parasitic inductances. Overlapping the power and ground traces for the gate charging loop also

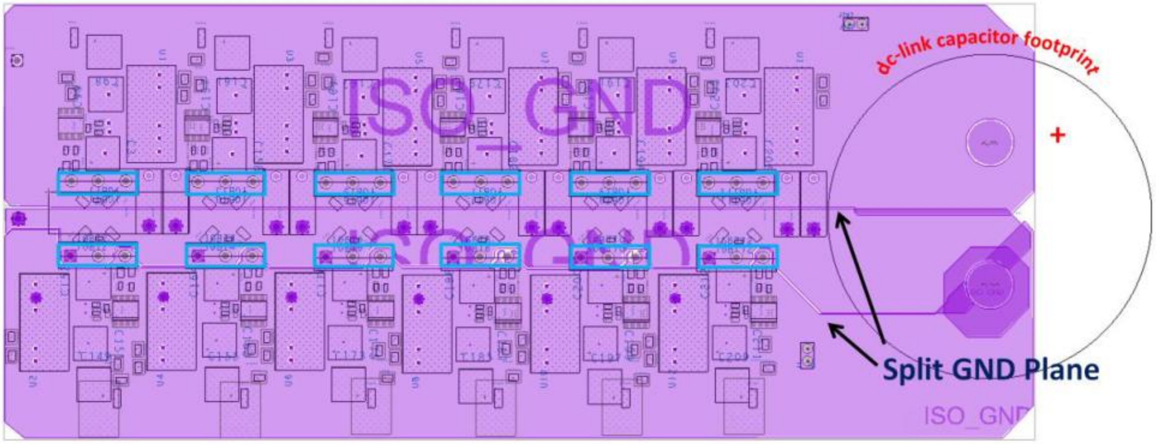


Figure 4-2. Bottom-middle layer photoplot for the power stage PCB used in the UCSC prototype

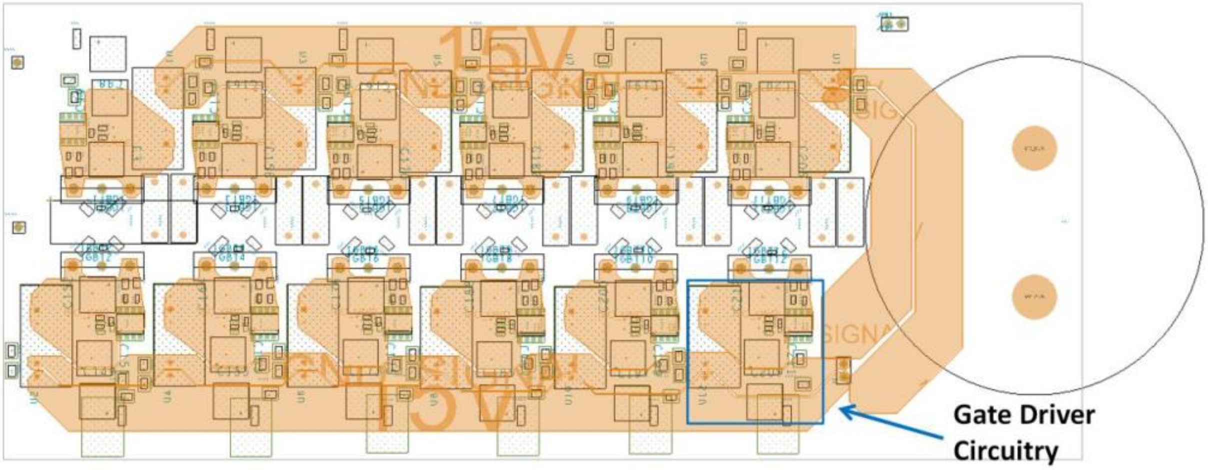


Figure 4-3. Top-middle layer photoplot for the power stage PCB used in the UCSC prototype

help to reduce the inductance by partial cancellation of the self-inductance of each of the traces using the mutual inductance between them [**5**]. This overlapping also creates a deliberate parasitic capacitance between the power rails which helps to filter the unwanted higher frequency currents that are generated in the gate driving path.

4.3 Design of the Filter Inductors

The design process given below was used to design and build the filter inductors that were used in the prototype. The process that comes from [**6**] can be applied to any ac inductor.

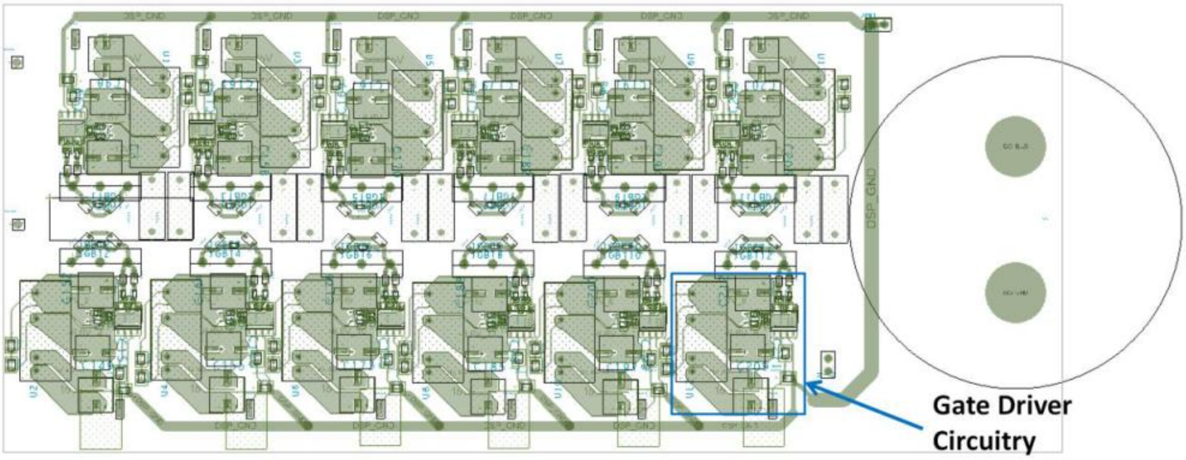


Figure 4-4. Top layer photoplot for the power stage PCB used in the UCSC prototype

After each step, the results for each inductor are in **bold**. The quantities in Table 4-3 were determined before the process can begin with the inductance values determined using the process outlined in section 2.4.

The sinusoidal voltage across the inductor is determined based on its reactance and the rated current through it. The current density is chosen within a normal range of 150-250 A/cm^2 which changes based on the core material. Flux density is determined from the maximum operating flux density of the material that is chosen for the core. Window utilization refers to the percentage of the area that is enclosed by the inductor core that is filled with the wire and its insulation. Waveform factor is a scaling term used in several of the equations which reflects the shape of the applied ac waveform; for a square waveform $K_f = 4.0$, and for a pure sinusoidal $K_f = 4.44$. The temperature rise goal refers to the inductor temperature with respect to room temperature which is considered to be 40 °C,

<u>Step 1 – Power Rating</u>:

Determine the inductor power rating using its voltage and current:

$$S = V_L I_L \quad VA \tag{3}$$

resulting in:

Table 4-3. 2.2 mH and 0.5 mH Inductor Design Parameters

Parameter	2.2 mH	0.5 mH	Unit
Voltage across inductor, V_L	8.29	1.884954	V
Line Current, I_L	10	10	A
Line Frequency, f	60	60	Hz
Current Density, J	200	200	A/cm^2
Efficiency Goal, μ	90	90	%
Flux Density, B_{ac}	1.5	1.5	T
Window Utilization, K_u	0.4	0.4	
Waveform Factor, K_f	4.44	4.44	
Temperature Rise Goal, T_r	20	20	˚C
Desired Inductance, L	0.0022	1.884954	H

2.2 mH: S = 83 VA

0.5 mH: S = 19 VA

Step 2 – Area Product:

The area product is determined based on:

$$A_P = \frac{S}{K_f K_u f B_{ac} J \; x \; 10^4} \quad cm^4 \tag{4}$$

yielding:

2.2 mH: $\boldsymbol{A_P}$ = 17.30 cm^4

0.5 mH: $\boldsymbol{A_P}$ = 3.93 cm^4

This term determines the minimum size core that is required.

Step 3 – Core Selection:

There are many different core shapes as well as materials from which to choose. Each one has specific advantages over the others. For application in this inverter, size was the largest concern, therefore, a core with a large operating flux density is the most helpful. The large capacity for flux means that less core volume is required for the same excitation level. An amorphous core was chosen in this case, because of its maximum operating flux density of 1.6 T. The operating flux density shown in Table 4-3 was chosen as 1.5 T to have a safety margin. Currently, this type of core is mostly limited to C shapes, so that was the chosen shape. The selected cores based on A_P were:

2.2 mH: AMCC0040

0.5 mH: AMCC06R3

Table 4-4 shows data from Hitachi Metals for POWERLITE® amorphous cores [**7**]. MLT is the abbreviation for mean length of turn, which is the average length of wire needed to wrap around the leg of the core which can be estimated using values $A - F$ and Fig. 4-5. A_c is the

Table 4-4. Amorphous Core Data [5]

	A (cm)	B (cm)	C (cm)	D (cm)	E (cm)	F (cm)
AMCC06R3	1.0	1.1	3.3	2.0	3.1	5.3
AMCC0040	1.3	1.5	5.6	3.5	4.1	8.2
	Mass (g)	MLT **(cm)**	A_c **(cm^2)**	W_a **(cm^2)**	A_p **(cm^4)**	A_t **(cm^2)**
AMCC06R3	154	6.0	1.64	3.63	5.95	59.20
AMCC0040	530	9.6	3.73	8.40	31.33	136.54

core sectional area of the core (A x D), W_A is the area contained within the core computed using B x C, and A_P refers to the same value calculated in Step 2, however this value for the core is the product of A_C and W_A. A_t is the core surface area.

Step 4 – Wire Turns:

The number of wire turns is calculated by:

$$N = \frac{V_L}{K_f B_{ac} f A_c} x\ 10^4 \quad turns \tag{5}$$

yielding:

2.2 mH: N = 56 turns

0.5 mH: N = 29 turns

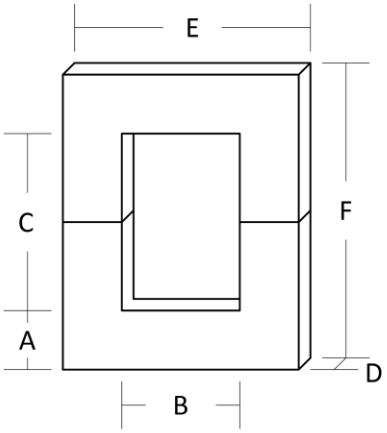


Figure 4-5. Dimensions of a C core pair [5]

Step 5 – Air Gap Calculation:

This is a deliberate gap between the two halves of the C cores that increases the reluctance of the core's flux path. A larger excitation current or magnetic field strength is required to achieve the same flux density because of the increase in reluctance. Though this reduces the effectiveness of the inductor as a whole, this gap prevents the core from saturating as quickly by extending the linear operating region of the core material. Fig. 4-6 demonstrates the effect of the air gap using the B-H curve that is used widely in magnetics [**6**]. The solid line is the core characteristic without an air gap and the dotted line is with an air gap given by:

$$l_{ag} = \frac{0.4\pi N^2 A_C}{L(10^8)} \quad cm \tag{6}$$

yielding:

2.2 mH: $\boldsymbol{l_{ag}}$ = 0.066 cm

0.5 mH: $\boldsymbol{l_{ag}}$ = 0.034 cm

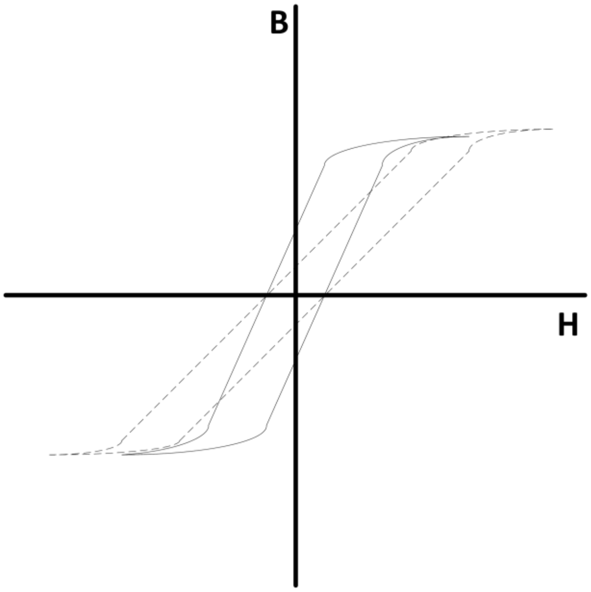


Figure 4-6. B-H Curve without airgap (solid line) and with airgap (dotted line)

Step 6 – Fringing Flux Factor:

The fringing flux factor is a unitless correction term to account for the fringing flux that arises from the air gap. The larger the air gap, the larger the fringing flux, which can cause localized heating around the air gap, also cause the material to saturate prematurely, and is calculated as follows:

$$K_{f\varphi} = 1 + \frac{l_{ag}\ln\left({}^{2l_C}\!/_{l_{ag}} \right)}{\sqrt{A_C}} \tag{7}$$

resulting in:

2.2 mH: $\boldsymbol{K_{f\varphi}}$ = 1.16

0.5 mH: $\boldsymbol{K_{f\varphi}}$ = 1.14

Step 7 – New Number of Turns:

A new number of turns is determined taking into account the fringing flux to avoid any problems that it might cause as follows:

$$N_{new} = \frac{l_{ag}L(10^8)}{0.4\pi A_C K_{f\varphi}} \quad turns \tag{8}$$

yielding:

2.2 mH: $\boldsymbol{N_{new}}$ = 52 turns

0.5 mH: $\boldsymbol{N_{new}}$ = 27 turns

Step 8 – Core Verification:

After determining a new number of turns it is necessary to check whether the core is still suitable for operating at a flux density appropriate for the material. This is done using:

$$B_{ac} = \frac{V_L(10^4)}{K_f N_{new} A_C f} \quad T \tag{9}$$

which yields:

2.2 mH: B_{ac} = 1.61 T

0.5 mH: B_{ac} = 1.60 T

These values are still appropriate for the materials chosen, though they are at the material maximum flux density; however the designed inductors have a larger current rating than that of the UCSC so they should never experience the above flux densities.

Step 9 – Bare Wire Area:

The area of the required wire based on the rated current of the inductor and the desired current density for the wire. The calculation is done per:

$$A_{bw} = \frac{I_L}{J} \quad cm^2 \tag{10}$$

resulting in:

2.2 mH: A_{bw} = 0.05 cm^2

0.5 mH: A_{bw} = 0.05 cm^2

The current density J may vary based on the operating temperature of the inductor and the available space.

Step 10 – Wire Size:

The next step is to select a wire size based upon the previously calculated bare wire area. The conductor chosen should have a bare wire are as large as A_{bw}. A table of AWG conductor areas is easily available from many sources [**8**]. Taking in account the above yields:

2.2 mH: 10 AWG

0.5 mH: 10 AWG

Step 11 – Winding Resistance:

The winding resistance is determined as follows:

$$R_W = N_{new}\gamma(MLT) = N_{new}\frac{\rho}{A_{bw}}(MLT) \quad \Omega \tag{11}$$

where γ is the wire resistance per centimeter. If the resistivity is not known, it can be calculated from the specific resistance, $\rho,$ of the wire material and the bare wire area. For the selected wire size and number of turns, (11) results in:

2.2 mH: 0.0079 Ω

0.5 mH: 0.0211 Ω

Step 12 – Winding Losses:

The winding losses are now determined using:

$$P_W = I_L^2 R_W \tag{12}$$

which yields:

2.2 mH: 0.79 W

0.5 mH: 2.11 W

Step 13 – Hysteresis Losses:

Some of the applied flux actually is cancelled out due to the residual magnetic field that is held by ferromagnetic materials from past applied fluxes. This interaction causes losses in the core that are referred to as hysteresis losses [**6**]. Eddy currents also produce losses; these are currents within the core that are induced in order to oppose the change in flux. These currents flowing within the material cause losses. Some cores are built using insulated laminations to minimize the eddy currents by limiting the length of the current loop that can develop perpendicular to the travel of flux [**6**].

These losses that are determined empirically by the core manufacturer depend on the properties of the material and the applied frequency. Usually they are estimated using graphs like the one in Fig. 4-7. The core losses incurred are determined as follows:

$$\Gamma_{core} = K_c f^{\alpha} B_{ac}{}^{\beta} \quad \frac{W}{kg} \tag{13}$$

where $K_c, f^{\alpha},$ and $B_{ac}{}^{\beta}$ are calculated per:

$$\alpha = \frac{\ln\left({P_{core,C}}/{P_{core,B}} \right)}{\ln\left({f_C}/{f_B} \right)} \tag{14}$$

$$\beta = \frac{\ln\left({P_{core,A}}/{P_{core,B}} \right)}{\ln\left({f_A}/{f_B} \right)} \tag{15}$$

$$K_c = \frac{P_{core,A}}{{f_A}^{\alpha} {B_{ac,A}}^{\beta}} \tag{16}$$

The subscripts A, B, and C represent the value of the variables they operate on at arbitrary points A, B, and C in Fig. 4-7 above. However, points A and B must have the same frequency, but differ from the frequency of point C and points B and C must share the same flux density, but differ from the flux density of point A.

In Fig. 4-7, the manufacturer provides the following values for $K_c, \alpha,$ and β:

$$\boldsymbol{K_c = 6.5}$$

$$\boldsymbol{\alpha = 1.51}$$

$$\boldsymbol{\beta = 1.74}$$

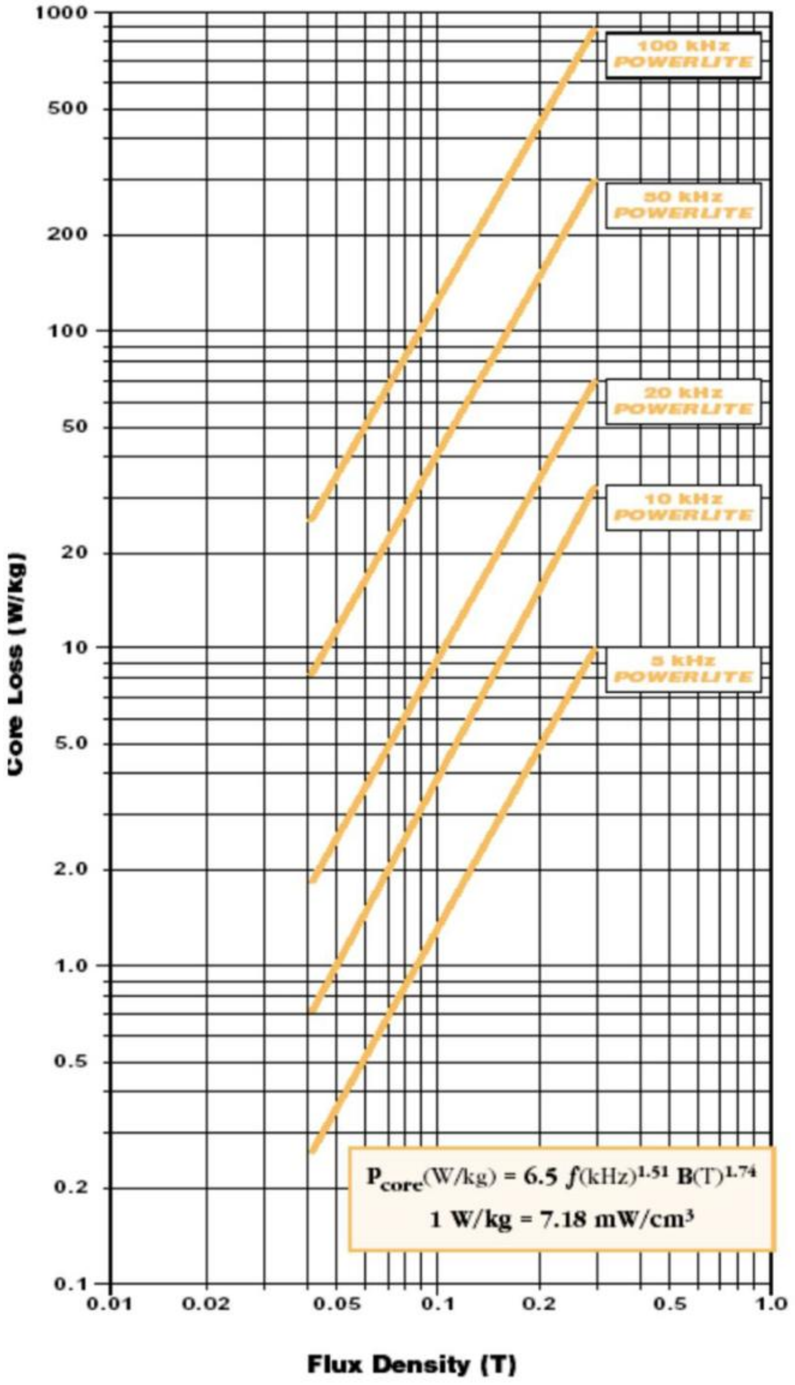


Figure 4-7. Core loss curves for the Hitachi POWERLITE amorphous cores

Step 14 – Total Core Losses:

Core losses are calculated as a function of mass, so the total core losses are given by:

$$P_{core} = \Gamma_{core} m_{core} \tag{17}$$

where m_{core} is the mass of the core. For the selected cores, the total core losses are:

2.2 mH: 0.113 W

0.5 mH: 0.032 W

<u>Step 15 – Loss Power Density</u>:

The temperature analysis begins once the core has been selected. Losses in the core and in the windings have only a limited surface area from which that heat can escape. The power loss per surface area is calculated as follows:

$$\Upsilon_{core} = SA_t(P_{core} + P_W) \ \frac{W}{cm^2} \tag{18}$$

resulting in:

2.2 mH: 0.0163 W

0.5 mH: 0.0139 W

<u>Step 16 – Core Temperature Rise</u>:

The temperature rise of the core is dependent upon the surface area power density and the properties of the material. This equation is used for iron based ferrites [**6**]:

$$T_{rise} = 450(\Upsilon_{core}) \ K \tag{19}$$

The design is satisfactory if this temperature rise is acceptable; otherwise a new core with a larger surface area per unit volume or a larger wire should be chosen. For the selected cores:

2.2 mH: 15 K rise

0.5 mH: 13.2 K rise

which is an acceptable temperature rise. The inductors will not require external cooling and will not overheat adjacent components.

Step 17 – Window Utilization:

The window utilization is computed in order to determine if the goals from Table 4-3 were met. The window utilization is given by:

$$K_u = \frac{NA_{bw}}{(0.9)(0.75)W_a} \tag{20}$$

The constants used in the denominator of (20) are used to modify the bare wire area to compensate for the inefficiencies in the wire winding process. The maximum space usage of a plane by identical circles is around 90%. A significant amount of the resulting diameter of a wire comes from the insulation. In this case it was assumed that the actual copper only accounts for 75% of the diameter of each winding. For the selected cores, this yields:

2.2 mH: 0.48

0.5 mH: 0.58

The window utilization of the inductors wound for this project are larger than the goal because an oversized wire was chosen.

Step 18 – Continuous Wire Length:

Lastly, the continuous length of wire needed to wind the inductor is specified. Using the MLT from the inductor might not be an accurate representation of the average length of the turns; in particular, this is the case when there will be more than one layer of turns used. The wire continuous length is calculated as follows:

$$l = N(MLT) \tag{21}$$

yielding:

2.2 mH: 695 cm

0.5 mH: 260 cm

4.4 Calculations for the IGBT Heat Sink

Due to the modulation of the duty cycle in unipolar switching, an estimation method for the losses incurred in the IGBT and antiparallel diode must take into account a non-constant duty cycle. The equations below from [**9**] use parameters given in the datasheet as well as an average operating point of the switch to estimate the losses. The assumption made in these equations is that the operating point used by the manufacturer to specify values in the datasheet is similar to the actual operating point of the IGBT so linear extrapolation of values is valid; that is:

$$P_{sw}^{IGBT} = \frac{1}{\pi} f_{sw}\left(E_{on} + E_{off}\right) \frac{V_{DC}}{V_{ref}} \frac{\hat{\imath}_L}{i_{ref}} \tag{22}$$

$$P_{sw}^{diode} = \frac{1}{\pi} f_{sw}(E_{rec}) \frac{V_{DC}}{V_{ref}} \frac{\hat{\imath}_L}{i_{ref}} \tag{23}$$

$$P_{conduction}^{IGBT} = \frac{V_{CE,0}\hat{\imath}_L}{2\pi}\left(1 + \frac{M\pi}{4}(pf)\right) + \frac{r_{CE}\hat{\imath}_L^2}{2\pi}\left(\frac{\pi}{4} - \frac{2M}{3}(pf)\right) \tag{24}$$

$$P_{conduction}^{diode} = \frac{V_{F,0}\hat{\imath}_L}{2\pi}\left(1 + \frac{M\pi}{4}(pf)\right) + \frac{r_f\hat{\imath}_L^2}{2\pi}\left(\frac{\pi}{4} - \frac{2M}{3}(pf)\right) \tag{25}$$

where the definitions of the variables used and their values are given in Table 4-5. The equivalent thermal circuit can be solved using these losses, the desired maximum operating temperature and thermal-electric analogies [**1**]. Fig. 4-8 shows this circuit, where R_{J-C} is the junction-to-case thermal resistance R_{C-S} is the case-to-sink thermal resistance and R_{S-A} is the sink-to-ambient resistance. Given the power dissipation of the IGBT and the antiparallel diode, a thermal pad/grease (R_{C-S}) and heat sink (R_{S-A}) must be chosen so that the power dissipation, analogous to current, does not cause a larger temperature rise, analogous to a voltage rise, than is allowed. This circuit must be solved for the IGBT and the diode separately as they do not have the same R_{J-C}.

Table 4-5. Variables for switching and conduction losses in controlled semiconductor devices

Parameter	Explanation	Prototype Value
f_{sw}	Switching frequency of inverter	10 kHz
E_{on}	Turn-on energy of IGBT from datasheet	0.63 mJ
E_{off}	Turn-off energy of IGBT from datasheet	1.39 mJ
V_{DC}	Actual dc bus voltage	250 V
V_{ref}	Test dc bus voltage from datasheet	480 V
$\hat{\imath}_L$	Peak line current	11.5 A
i_{ref}	Test line current from datasheet	17 A
M	Nominal modulation index	0.7
pf	Power factor of inverter	1.0
$V_{CE,0}$	On-state voltage of IGBT	1.8 V
r_{CE}	On-state resistance of IGBT	0.129 Ω
$V_{F,0}$	On-state voltage of diode	1.8 V
r_f	On-state resistance of diode	0.04 Ω

The circuits solved for the prototype are shown in Fig. 4-9. A thermal pad with a thermal resistance of 1.5 °C/W is the chosen isolating material between the IGBT package and the heat sink, so the required R_{S-A} is determined and a heat sink must be chosen accordingly. An ambient temperature of 40 °C is chosen to represent a worst case scenario. To limit the maximum junction temperature in either device to 85 °C, the maximum R_{S-A} is 2.55 °C/W. A

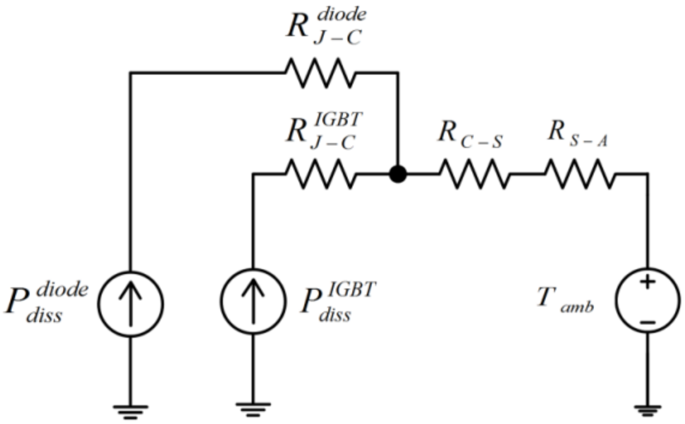


Figure 4-8. Equivalent thermal circuit for IGBT and antiparallel diode

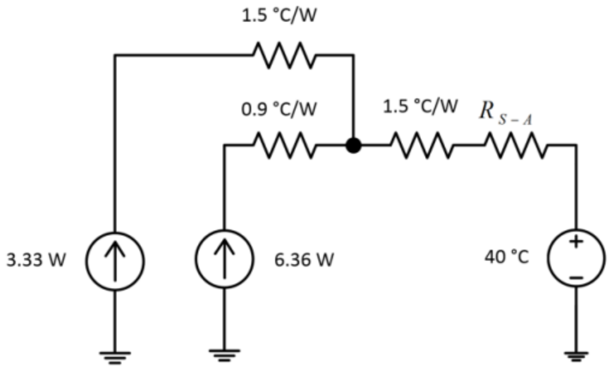


Figure 4-9. Thermal circuits for the IGBT and antiparallel diode

heat sink with a thermal resistance of 2.55 °C/W or less must be chosen. The heat sink that was chosen has a R_{S-A} = 2.2 °C/W, but only with a forced convection of at least 300 ft/min parallel to the blades. A fan was chosen with an air flow of 17.5 CFM and an output area of approximately 0.0341 ft^2. Using

$$Air\ Speed = \frac{\left(\frac{ft^3}{min}\right)}{(ft^2)} \tag{26}$$

results in an air speed of 513 ft/min, which is adequate.

4.5 Sensors and Measurement Conditioning

Isolated hall-effect sensors were used for measurement of the currents and voltages needed for the control of the UCSC. The LEM HY-10P current sensor was used to measure all currents and the LEM LV-20P to measure voltages. Isolated sensors are useful because they decouple the measurement from any noise that exists in the ground of the power stage and reduce the requirements of the PCB ground design. The sensors are also located away from the IGBTs and LCL filter to reduce the EMI that they experience. The outputs of these sensors become the input to the operational amplifier (op-amp) based conditioning circuits that modify the outputs of the sensor to the appropriate voltage levels for use by the DSP. The op-amp used for the prototype is the Tl081a from Texas Instruments and is fed by the same +15 V/-15 V supply rails that supply the sensors which allow this 30 V range at each stage of the conditioning circuit. The first stage of the conditioning circuit scales the voltage output from the sensor while the second stage shifts the dc offset of the signal to 1.5 V, which corresponds to the half of the maximum voltage (3.0 V) allowed by the analog-to-digital converter used by the digital controller. The conditioning circuits for the current measurements also contained a phase-lagging third stage that if utilized could replace the digital SOGI implemented in the digital signal processor and save computation resources.

4.6 Concluding Remarks

The methods for designing the scaled-down UCSC prototype were disclosed addressed in this chapter. This included the per-unit scaling process used to reduce the 6 MVA case study to a 10 kVA laboratory system and the selection of components in addition to dead time selection. A synopsis of the effects of parasitics on the power stage provided the motivation for the design of the PCB presented in subsection 4.2.8. Almost all the components used in the scaled-down prototype were over-designed by 15 % to increase the robustness and flexibility of the prototype. Reliability is important because device failures can be costly and repairs may require long period of down time.

An appropriate implementation of the control in a digital signal processor is also important for reliability. Using an appropriate start-up and shutdown procedures as well as overcurrent protection can increase the lifetime of the devices that are used. Achieving the necessary performance from a digital controller is the topic of the next chapter.

4.7 Ref**erences**

[1] N. Mohan, T. M. Undeland, and W. P. Robbins, *Power Electronics: Converters Applications and Design*, 3rd ed. New Jersey, United States of America: JW&S Inc, 2003.

[2] Manuel A. S. Tejada, "Unbalanced Current Static Compensator," Dept. of Elect. Eng., Univ. of Arkansas, Fayetteville, MS Thesis 2014.

[3] Semikron, "High-Frequency Oscillations due to Driver Coupling," *Power Electronics Europe*, no. 6, pp. 2-4, 2013.

[4] Texas Instruments Incorporated, PCB Design Guiudelines for Reduced EMI, 1999.

[5] Richard K. Ulrich and William D. Brown, *Advanced Electronic Packaging*, 2nd ed.: Wiley-IEEE Press, 2006.

[6] Colonel Wm. T. McLyman, *Transformer and Inductor Design Handbook*, 4th ed. United States of America: CRC Press, 2004.

[7] Hitachi Metals America Ltd. (2011) Hitachi Metals Web Site. [Online]. http://www.elnamagnetics.com/wp-content/uploads/catalogs/metglas/powerlite.pdf

[8] PowerStream Technology. (2016, March) Wire Gauge and Current Limits Including Skin Depth and Strength. [Online]. http://www.powerstream.com/Wire_Size.htm

[9] W.-T. Franke, M. Mohr, and F.W. Fuchs, "Comparison of a Z-source inverter and a voltage-source inverter linked with a DC/DC-boost-converter for wind turbines concerning their efficiency and installed semiconductor power," in *Power Electronics Specialists Conference, 2008*, 2008, pp. 1814-1820.

CHAPTER 5

IMPLEMENTATION OF THE CONTROL ALGORITHM IN A MICROCONTROLLER

5.1 Introduction

The objective of this chapter is to present the methods used to implement the controller presented in Chapter 2 in the Texas Instruments TMS320F28335 Digital Signal Processor (DSP); a 32-bit floating-point microcontroller. The computational load of the presented controller is large due to the single-phase d-q transformations, so the initial implementation used three DSPs to iterate through the discrete controller at the desired rate of 10 kHz. Each DSP controlled an inverter and the necessary d-axis load current and voltage bus data was exchanged between these three using the SCI serial communication ability on board each DSP. This separation of computational load made debugging very inefficient and any auxiliary functions, such as protection and operating mode changes, difficult to implement. A different solution using only one DSP was desired; the newly devised system is the subject of this chapter.

5.2 Developing a Single-DSP System

A single-DSP solution was developed after further experience, significant reading, and consultation from individuals more experienced on DSPs. The problem with the original implementation is the location of the program used to implement the controller inside the DSP. The program must be assigned to a non-volatile memory location to run the DSP in stand-alone mode because volatile memory loses its data when the power is removed. The drawback of this non-volatile memory, called FLASH memory, is that it requires many more CPU cycles to access it for each instruction than it would for the same code stored in RAM; a volatile memory. The key to running the code all in one DSP, but also in stand-alone mode, is to store the program in FLASH, but copy the program to RAM at run-time. In this way the DSP does not require a computer connection or suffers from the speed penalties that come from running the program out of FLASH memory. With the above modifications it is now possible to run the entire program used to control the UCSC at a rate of 20 kHz.

5.3 Controller Implementation in the DSP

The DSP implements all of the control functions presented in Chapter 2 as illustrated in Fig. 5-1. The auxiliary processes performed outside the DSP controller code are shown inside the red section in Fig. 5-1. These actions, that occur as fast as the DSP can perform them, include the polling of the current controller flag bit ‘ccbit’ for proper timing of the current controller, checking for overcurrent, and polling a PWM disable input. When an overcurrent or the PWM disable input is detected, the PWM module on the DSP drives all gate outputs to 0 shutting down the system.

The CPU timer interrupt that can transpire at any step within the red section normally occurs every 0.00005 s (period of a 20 kHz signal). This interrupt sets ‘ccbit’ so that the current controller will run at the same frequency as the interrupt. The timing of the current controller is crucial because the accuracies of the z-domain functions used to implement the controller operations are dependent upon the frequency at which they are run.

The blue section in Fig. 5-1 has the current controller operations. The controller starts by sampling all of the required current and voltage signals and then, calculates the phase-lock loop and the d-q transformations for the currents. The next action depends on the mode selected by the user. The inverter has three modes: (1) only dc-bus voltage regulation, (2) compensation for load reactive power and dc-bus voltage regulation, and (3) full current balancing operation. The errors processed by the PI controllers will change depending on the operating mode.

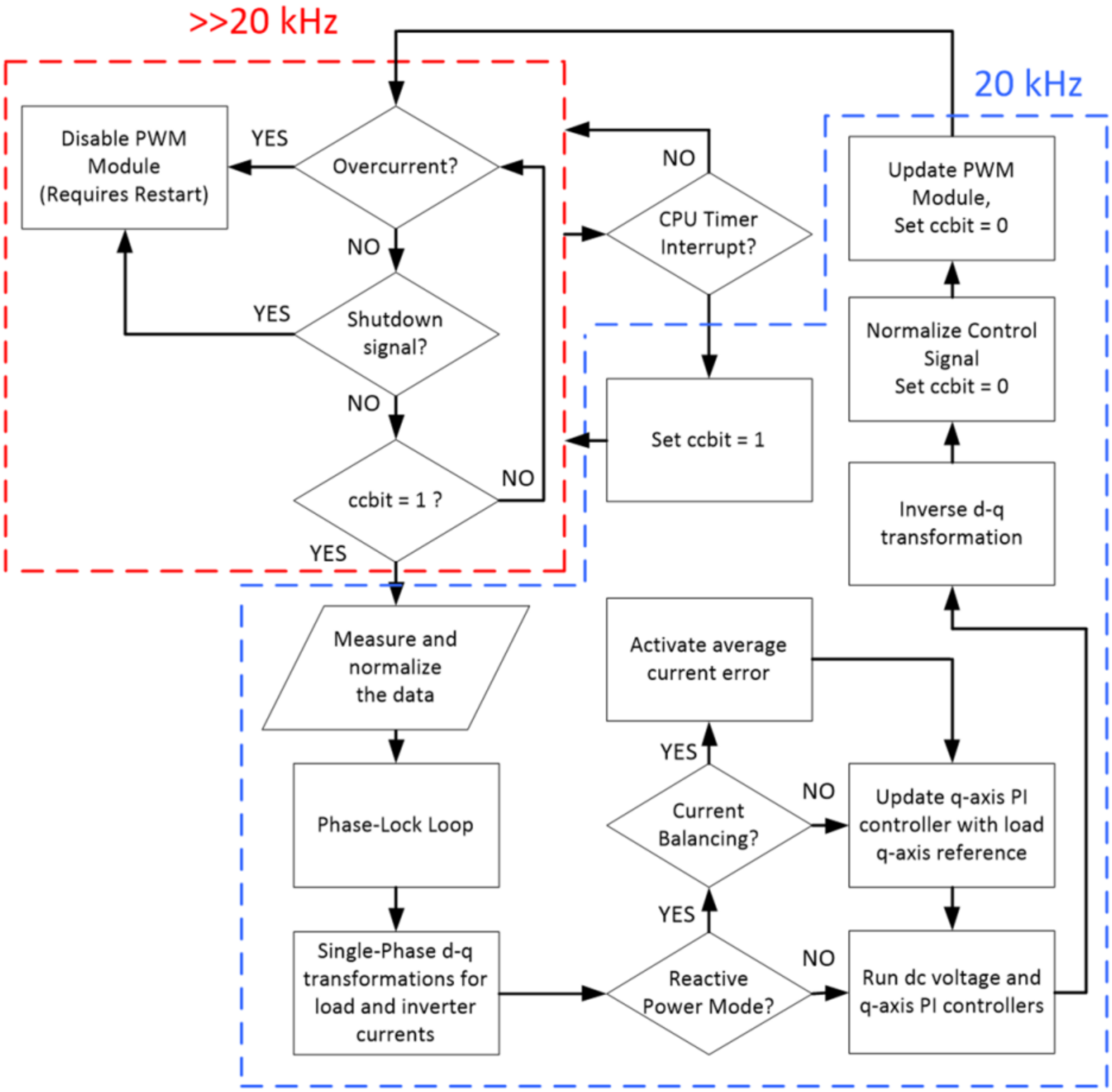

>>20 kHz
20 kHz
Disable PWM Module (Requires Restart)
YES
Overcurrent?
NO
YES
Shutdown signal?
NO
ccbit = 1 ?
NO
YES
NO
CPU Timer Interrupt?
Set ccbit = 1
Update PWM Module, Set ccbit = 0
Normalize Control Signal Set ccbit = 0
Inverse d-q transformation
Measure and normalize the data
Phase-Lock Loop
Single-Phase d-q transformations for load and inverter currents
Activate average current error
YES
Current Balancing?
NO
YES
Reactive Power Mode?
NO
Update q-axis PI controller with load q-axis reference
Run dc voltage and q-axis PI controllers

Figure 5-1. Flowchart for the DSP processes

Then, the controller must perform the inverse d-q transformations of the output voltage and develop a control signal for to the PWM module which produces the gate signals for the IGBT in the three single-phase inverters.

5.4 Start-Up and Shutdown Procedure

The controller in section 5.3 defines the behavior of the DSP during steady-state operation, but the DSP and user behavior must change during start-up to avoid large inrush currents that might damage the semiconductor devices used in the power stage. To soft start the UCSC scaled-down prototype, the follow procedure is used:

1) Adjust the autotransformer that acts as the system voltage source to lower the grid voltage down to 0 V.
2) Close the breaker that isolates the UCSC from the grid to connect it to the grid.
3) Increase the system voltage is until the UCSC experiences a dc-bus voltage of 10 V.
4) Turn-on the DSP. The DSP forces the PWM module to shut off the gate signals while it runs the phase-lock loop for 0.5 s. After this time the DSP enables the PWM gate signals and the rest of the controller for each inverter when the corresponding grid voltage is at the zero crossover point.
5) Then, the dc-bus reference voltage in the voltage controller ramps from 10 V to 250 V at a rate of 20 V/s, forcing the UCSC to raise the voltage of the dc-bus to 250 V. Increase the grid voltage linearly by adjusting the autotransformer to follow the now rising dc-bus voltage while keeping the peak value of the grid voltage below the dc-bus voltage.
6) With the dc-bus at 250 V and the grid voltage at 110 $V_{L\text{-}N}$, pull the first input pin low to enable reactive current compensation mode inside the controller. The DSP will ramp the reactive current reference from 0 to 100% of the full value linearly in 2.0 s to avoid a large step change in the reference.
7) After the UCSC has compensated for all of the reactive current, pull the second input pin low to force the UCSC into full compensation mode. Again, the DSP ramps the current reference value from 0 to 100 % linearly over 2.0 s.

The following procedure is used to shut down the system:

1) When data acquisition for the run test is complete, pull the shutdown input pin low to shut off the PWM module.
2) Adjust of the autotransformer to 0 V and open the breaker that connects the UCSC to the grid. The charge in the dc-bus capacitor will dissipate through a bleeding resistor.

For installation on a distribution system, this method would not be acceptable as the system voltage cannot be controlled in the same manner. In this case, a strategy that allows for direct connection to an energized system would need to be employed.

5.5 Initializing the DSP Modules

Each of the hardware modules in the DSP requires register initializations that dictate how a particular module will act when activated. The proceeding subsections describe the initialization settings for the hardware modules used in the DSP.

5.5.1 Analog-to-Digital Converter

The analog-to-digital converter (ADC) converts the continuous-time signals from the current and voltage sensors into series of discrete samples. For the scaled-down prototype the ADC is initialized to sample at the fastest available sampling rate of 12.5 million samples per second (MSPS) and was set to sample ten measurements continuously as follows:

1. Phase A inverter current
2. Phase A load current
3. Phase A grid voltage
4. dc-bus voltage
5. Phase B inverter current
6. Phase B load current
7. Phase B grid voltage
8. Phase C inverter current
9. Phase C load current
10. Phase C grid voltage

As a result the present values of each current and voltage sensors are always available in the ADC conversion result register. Because the ADC module actually consists of two separate eight-input multiplexors, ADCA and ADCB, they were initialized into cascade mode where they can act as one sixteen-input multiplexor.

5.5.2 Pulse-Width Modulator

The pulse-width modulation hardware module, known as the ePWM module in the DSP documentation, works on the same principle as described in section 2.6. The time-based counter or timer that is used to emulate the carrier waveform in sinusoidal PWM is set to the up_down count mode. This creates a symmetrical triangular waveform as opposed to up or down count modes, which produce asymmetrical sawtooth carrier waveforms that in turn would produce asymmetrical gate pulses. The period of the timer is set to ½ the period of the desired carrier signal measured in clock cycles or 15000_{10}.

Of the six ePWM modules used for the scaled-down prototype, ePWM1 is used as the master module and dictates the timing of the remaining ePWM modules so that all six modules count up and down in unison. When the timer is equal to zero, the value of the calculated control signal for that switching cycle is loaded into the module and the pulses are generated directly by the hardware without any further software intervention.

The “dead-band” sub-module within each ePWM module is initialized to automatically insert a dead time between any gate signal transitions. Active-high complementary mode is used so that the signals sent to the top and bottom IGBT of each half bridge in the UCSC are always opposites. A dead band of 150_{10} clock cycles is also specified which translates to a dead time of 1.0 μs.

The “trip-zone” sub-module also resides within the PWM module and allows the user to specify the behavior of the ePWM module during software or hardware events. The trip-zone sub-module for each of the ePWM modules is configured to force all gate signals low when an input pin is pulled low or action is triggered by software.

5.5.3 CPU Timer

A CPU Timer is a timer module with the DSP that waits for a specified amount of time and then generates an interrupt. This was used in the UCSC to give the appropriate timing for the discrete controller. The period value for CPU TIMER 0 was chosen as 7500_{10} to generate an interrupt at a rate of 20 kHz based on a clock frequency of 150 MHz.

5.6 Calculating the Dead Time for the Inverter-Leg Semiconductor Devices

A non-ideal transistor does not transition from on to off and vice versa instantaneously; there is a specific time [**1**]. In gate controlled transistors like the MOSFET and IGBT, there is a certain amount of charge stored in the gate that must be injected or removed in order to turn the device on and off, respectively. The time to fully charge or discharge this gate-emitter capacitance C_{GE} is a period of time when the transistor is still on or off even though the gate signal coming from the controller is indicating the opposite should be true. For IGBTs after the initial MOSFET action there is an additional amount of charge that must be depleted from within the device itself adding to the turn-off time. This charge is in the form of minority carriers that build up in the device [**2**]. An IGBT will still conduct a current while these carriers exist. This extra current component is known as the “tail current” due to the shape it forms in a switching diagram like the one shown in Fig. 5-2.

The IGBT itself has a turn-off delay, $t_{d,IGBT}$, as well as a current fall time, $t_{f,IGBT}$. Another concern is the delay of the gate driver circuitry. The isolator and gate driver ICs used

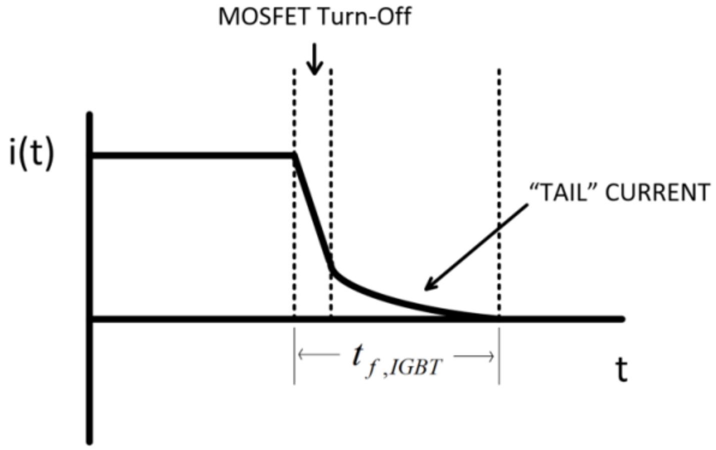


Figure 5-2. Turn-off current waveform for an IGBT [**2**]

have a range of delay times $t_{d,iso}$ and $t_{d,gd}$, as well as fall times, $t_{f,iso}$ and $t_{f,gd}$. These added together are the minimum dead time to be used. This is summarized in:

$$t_{dead,min} = t_{d,iso} + t_{f,iso} + t_{d,gd} + t_{f,gd} + t_{d,IGBT} + t_{f,IGBT} \tag{1}$$

These values can be taken from the manufacturer's datasheet for each IC and are displayed in Table 5-1. For driving the IGBTs in the scaled-down prototype a single chip (Silicon Labs SI8261BCC-C-IS) was used for isolation and driving so the isolator delays are left blank. To account for any variation in the performance among the ICs due to age, operating conditions, temperature, and circuit damage, a safety margin of 15%-50% is added to the minimum dead time. For the UCSC the dead time value is established as 1 µs.

Table 5-1. Gate Driver Circuit IC Datasheet Parameters

Parameter	**Value (ns)**
Maximum gate driver delay time	200
Maximum isolator delay time	*
Maximum isolator fall time	*
Maximum gate driver fall time	80
Maximum IGBT delay time	350
Maximum IGBT fall time	230

* isolator delay included in the gate driver delay

5.7 Modifications for EMI Reduction

The low voltages used in the DSP make it more susceptible to any interference from the switching action of power electronic converters and the $\frac{di}{dt}$ from the gate signals that it triggers [**3**]. These effects can cause false triggering of inputs and corrupted ADC measurements at higher current and voltage levels. Internal pull-up resistors inside the DSP help to mitigate false triggering in low EMI environments, but they are not sufficient at higher interference levels. Placing a relatively small pull-up resistor (1-10 kΩ) external to the DSP input pin will "strengthen" that input pin against a noisy environment because a larger current is required to drive the pin low.

As suggested in [**3**], an RC filter with a high cut-off frequency was placed near the input of each ADC pin. The capacitor near the input acts a charge bank for the sample-and-hold circuit used in the ADC. This counteracts the voltage drop from any series inductance between the signal conditioning circuits and the ADC input while also helping to eliminate high-frequency noise.

At the ePWM output pins, an IC is used to boost the current and voltage output from the 4 mA and 3.3 V that can be produced by the DSP to 20 mA and 5.0 V. This reduces the stress on the ePWM outputs and also drives the isolating IC in the gate driver circuitry with more current, helping to increase the ratio of signal to common-mode noise that it experiences. In addition to this twisted pair ribbon cables were used to help compensate for the relatively long distance that the gate and sensor signals must travel. This helps to cancel out the inductive coupling that causes noise in single wires [**4**]

5.8 Concluding Remarks

An overview of the controller as implemented in a DSP has been presented including separate start-up and shutdown procedures because this was not addressed in the simulations. The highlights for initializing each module used in the DSP were summarized and a complete copy of the C program is given in Appendix B. With the controller defined in this chapter and the prototype designed in Chapter 4, the prototype system was tested and the results are presented in Chapter 6.

5.9 References

[1] Infineon Technologies AG, "How to calculate and minimize the dead time requirement for IGBTs properly," Appl. Note AN2007-04 2007.

[2] N. Mohan, T. M. Undeland, and W. P. Robbins, *Power Electronics: Converters Applications and Design*, 3rd ed. New Jersey, United States of America: JW&S Inc, 2003.

[3] Texas Instruments, "Hardware Design Guidelines for TMS320F28xx and TMS320F28xxx DSCs," Application Note SPRAAS1C 2015.

[4] C. R. Paul and J. W. McKnight, "Prediction of Crosstalk Involving Twisted Pairs of Wires - Part I: A Transmission-Line Model for Twisted-Wire Pairs," *IEEE Transactions on Electromagnetic Compatability*, vol. EMC-21, no. 2, pp. 92-105, May 1979.

CHAPTER 6

TESTING OF THE UCSC PROTOTYPE

6.1 Experimental Setup

The experimental setup used to test the UCSC prototype is depicted in Fig. 6-1. The system consists of:

- A variable ac source with a maximum value of 208 $V_{L\text{-}L}$ that acts as the substation.
- The load is represented by high-amperage three-phase line reactors with a cumulative value of 3.0 mH in series with an unbalanced resistive load with equivalent resistance values of 20, 30, and 40 Ω.

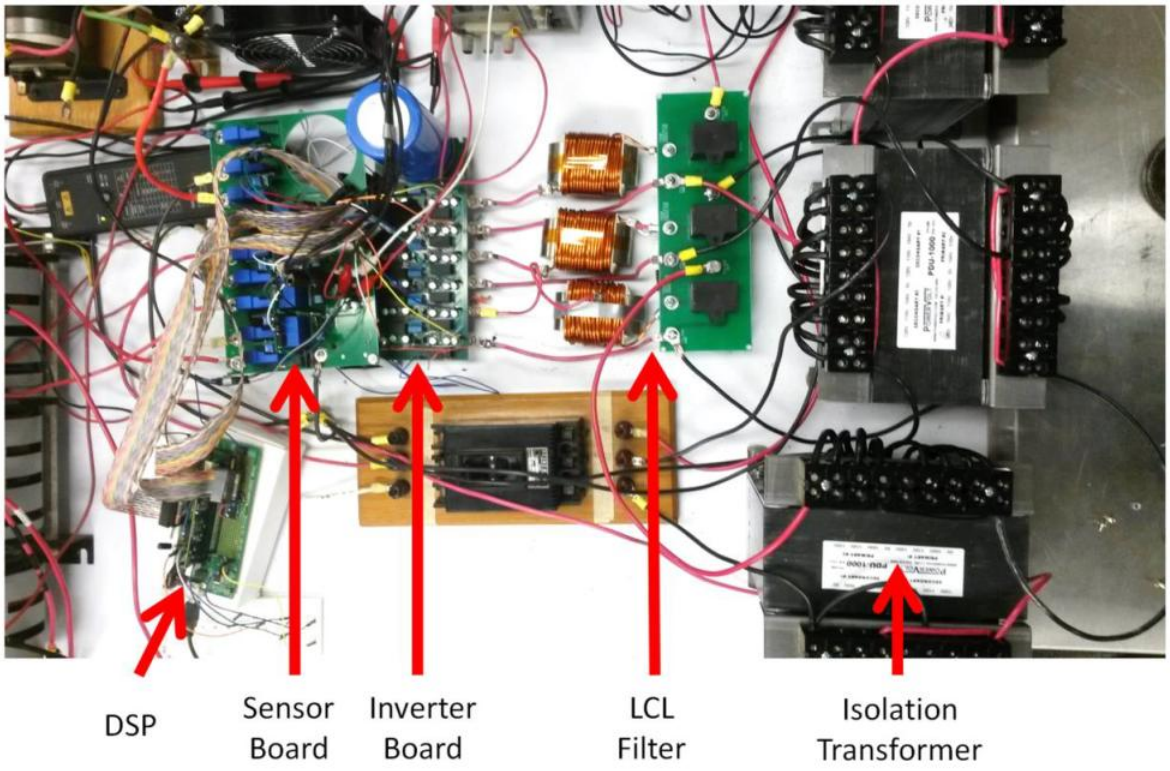


Figure 6-1. Experimental setup for testing of the UCSC prototype

- Connected in parallel with the load immediately after the ac source is the UCSC which is connected to the PCC through three 1:1 isolation transformers.
- For protection of the system, the isolation transformers are connected via a 15 A fuse and a 15 A breaker.

6.2 Testing of the UCSC Prototype

Displayed in Fig. 6-2 on the left side are the unbalanced currents that the UCSC has to compensate which correspond to a UBF of 8.8%. The resulting substation currents during full operation of the UCSC prototype are shown in Fig. 6-2 on the right side. The phase currents in each of the figures are displayed in red, blue, and green, with the neutral current in blank. In Fig. 6-3 the phase shift between the phase voltage (blue) and current (black) before and after compensation is shown. There is an improvement in power factor from 0.92 without compensate to 0.999 with compensation.

The UCSC is effective at reducing the fundamental neutral current at the substation and brings the system UBF to approximately 0.7%; however, some neutral current remains. These

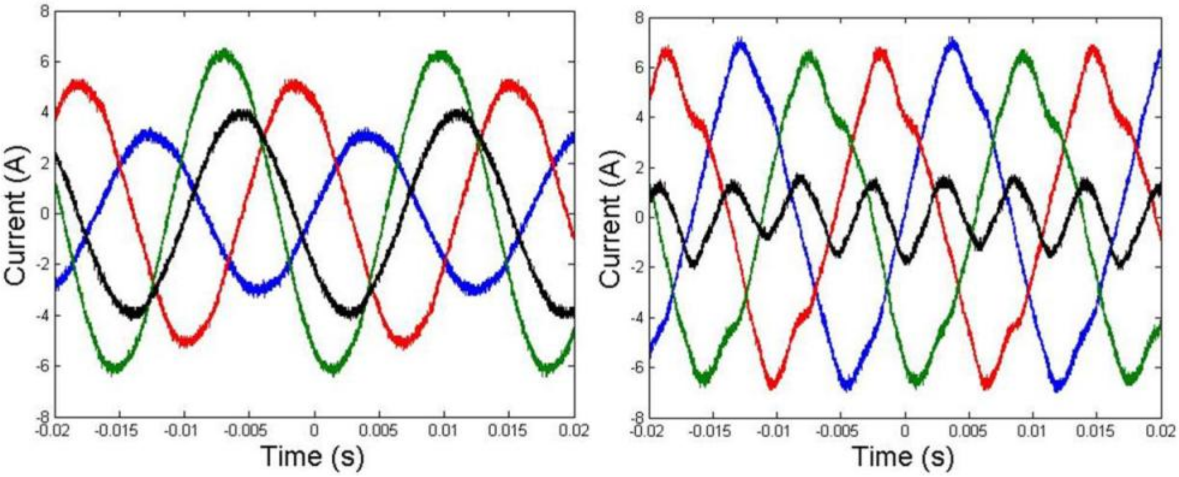


Figure 6-2. Unbalanced load currents (left) and substation current (right) during testing

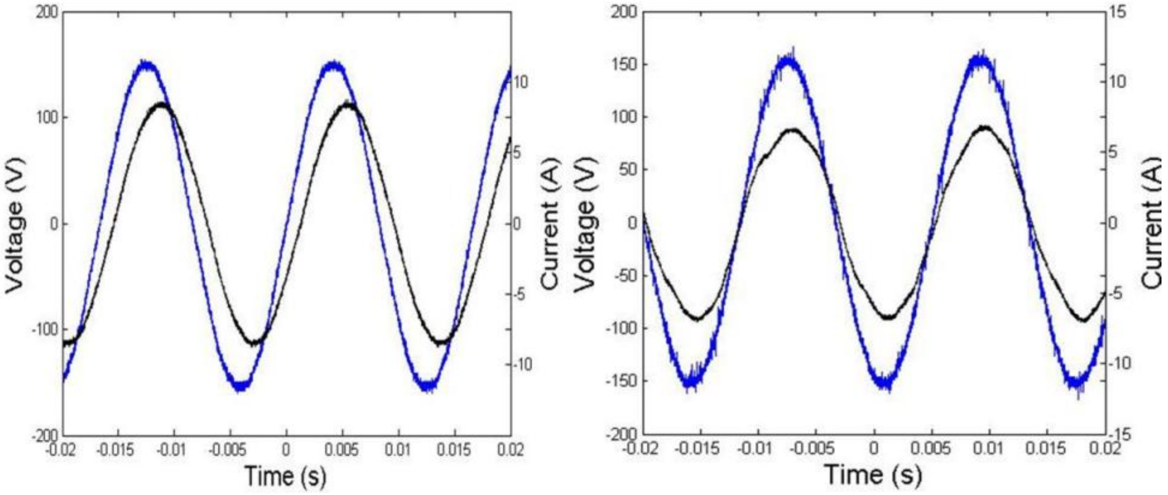


Figure 6-3. Phase Voltage and Current before (left) and during (right) UCSC full operation

currents are produced by the inverters due to dead time distortion that occurs in a standard unipolar switching scheme [**1**]. This distortion is easily seen in Fig. 6-4, which shows the phase inverter currents during full operation and the FFT of the substation neutral current. The FFT shows that the peak fundamental current component in the neutral has been reduced to 0.15 A from 4.0 A, but a 3rd harmonic peak current of 0.8 A is present. An FFT of one of the

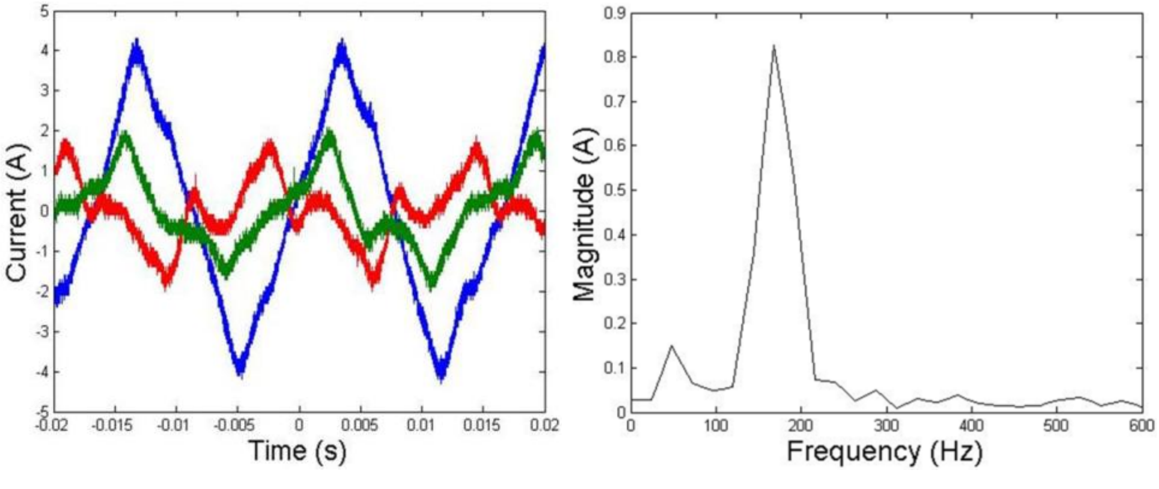


Figure 6-4. UCSC inverter currents (left) and FFT of the substation neutral current (right) during full UCSC operation

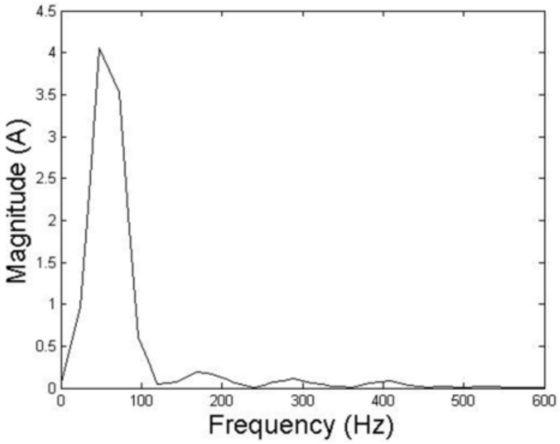


Figure 6-5. FFT of the substation current

substation phase currents is presented in Fig. 6-5 has a 6.0 % 3rd harmonic current component; the fundamental component is 4.0 A peak.

In an attempt to improve the performance of the UCSC several different tests were run to decrease the effect of the zero-crossover distortion and thus the harmonic current output of the UCSC inverters. The following sections describe the differences between these tests and their effects.

6.2.1 Effect of Increasing the dc-bus Voltage

The dc bus was raised from 250 V to 300 V to test the performance of the inverter. The resulting inverter currents are shown in Fig. 6-6 together with an FFT of the substation neutral current. There is no significant change in the harmonic performance of the inverters.

6.2.2 Effect of a Decrease in the Dead Time

The dead time used during the transitions between the high and low switches was decreased from 1.0 μs to 0.833 μs. An improvement is expected because the dead time is the cause of distortion at the output of the inverters [**1**]. Fig. 6-7 shows the inverter current

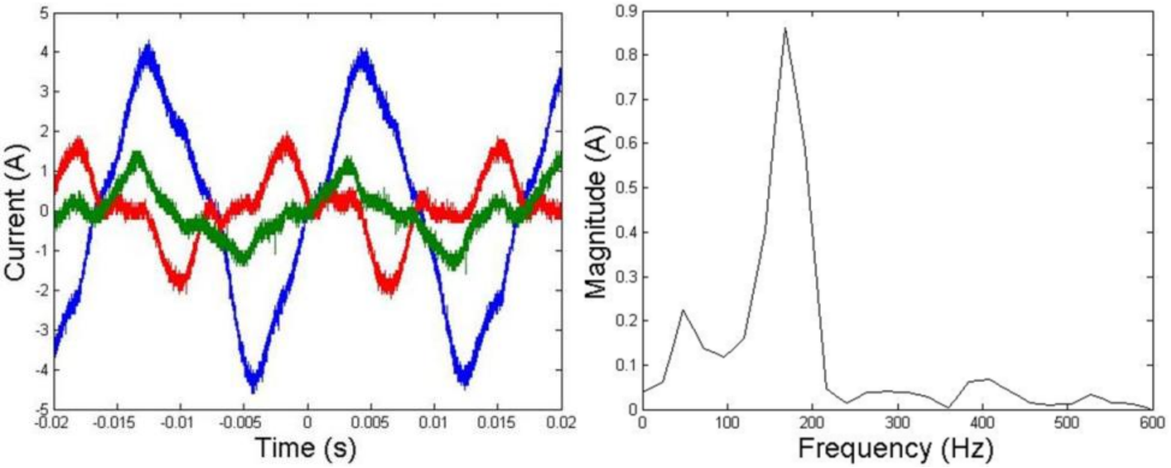


Figure 6-6. Inverter Currents (left) and FFT of substation neutral current (right) with 300 V dc bus

waveforms in addition to an FFT of the substation neutral current for this test. There is no significant change in the harmonic content.

This could be due to the fact that the reduction in dead time was not large enough to affect much change, or that other effects which are discussed in section 6.3 are responsible.

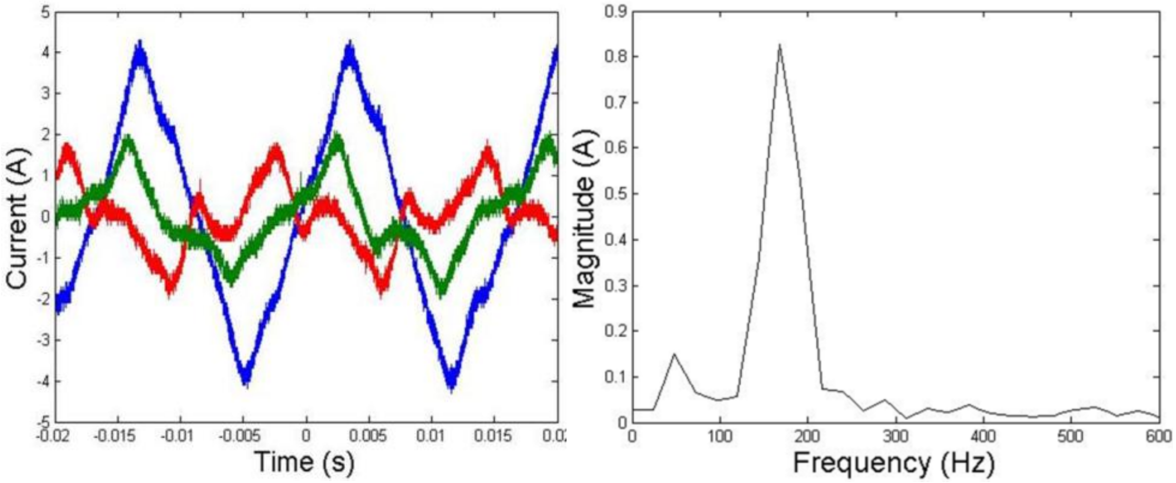


Figure 6-7. Inverter Current (left) and FFT of substation neutral current (right) with 0.833 μs dead time

6.2.3 Effect of Increased Inverter Loading

A large induction motor with no shaft load is used to increase the reactive power compensation requirement for the UCSC. The induction motor draws current at a power factor of 0.12 lagging which results in 2.0 A of inductive current. The load currents for which the UCSC has to compensate are shown in Fig. 6-8, which still has a UBF of 8.8% because the additional load is balanced. An FFT of the neutral current and the inverter currents from this test are shown in Fig. 6-9. The magnitude of the 3rd harmonic current is approximately 0.65 A_{peak} down from the 0.8 A_{peak} in the initial testing. This reduction is due to the change in phase shift between the inverter phase currents, which causes some of the 3rd harmonics to cancel out. A significant 3rd harmonic still flows back to the substation as shown in Fig. 6-10.

6.3 Causes of Output Voltage Distortion in a Single-Phase Inverter

As mentioned previously in section 5.6, a dead time must be included in the transition between the top and bottom switches conduction to avoid shorting the dc bus when using a half-bridge configuration. When neither IGBT is conducting in the presence of an inductive load, the

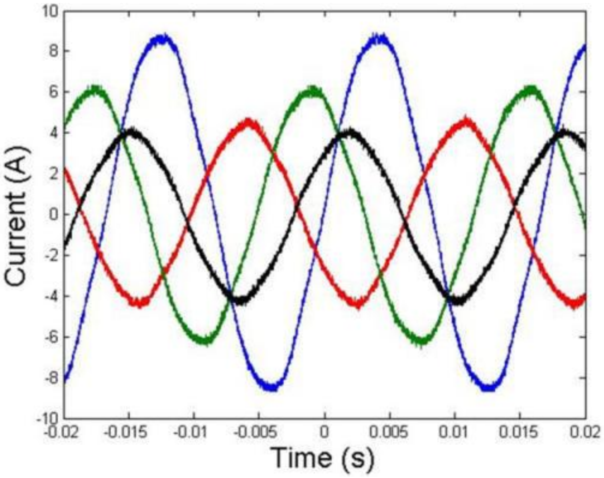


Figure 6-8. Unbalanced load currents during increased load testing

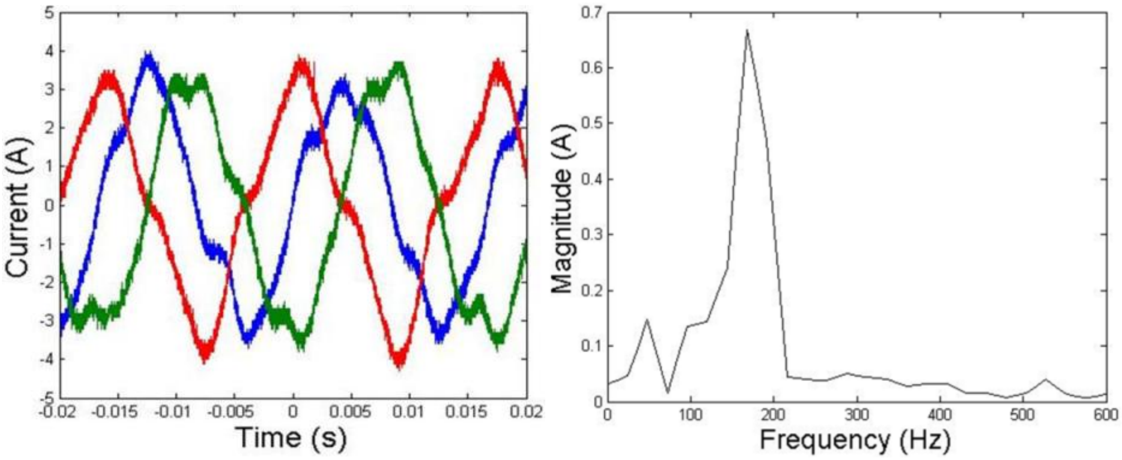


Figure 6-9. Inverter currents (left) and FFT of the substation neutral current (right) with increased inverter loading

voltage distortion that results is dependent upon the direction of the load current [**2**]. When current is flowing towards the load, the freewheeling diode paired with the bottom switch conducts during the off-time of the top switch, clamping the output voltage to the negative dc-bus voltage. This results in a lower output voltage than is dictated by the control waveform sent to the PWM generator. The output voltage is higher when the current is flowing towards the half-bridge.

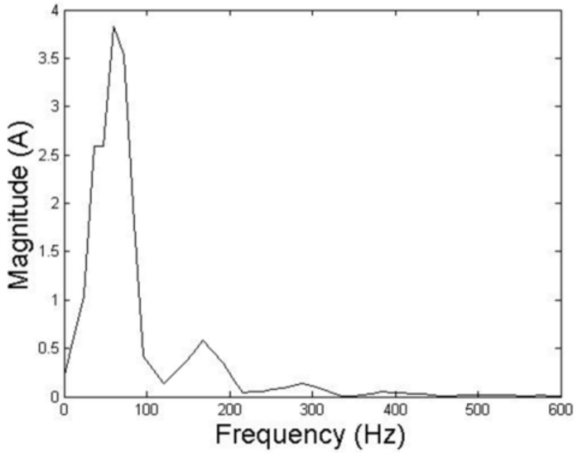


Figure 6-10. FFT of substation current with increased inverter loading

A solution for this current based distortion is suggested in [**3**] where a square waveform is added to the sinusoidal output voltage. The square waveform is in phase with the output current and has a magnitude ΔV given by:

$$\Delta V = \frac{t_d}{T_c} V_{dc} \tag{1}$$

where t_d is the dead time, T_c is the period of the carrier waveform, and V_{dc} is the dc-bus voltage.

Around the zero-crossing point of the voltage waveform, the PWM pulses are very small and close to the width of the dead time band. This may result in the total elimination of a pulse, or a pulse that is so short that the IGBTs cannot respond fast enough and thus lost. This paired with the non-zero minimum voltage drop across the IGBT during conduction (on-state) results in a distorted voltage at the output. Fig. 6-11 is an oscilloscope capture of the output voltage (yellow) and the output current (blue) from one the inverters used in the UCSC. The load was purely resistive in this case so the current and voltage distortion (encircled in red) looks identical, but the current distortion will not appear at the zero-crossover point in the case of a leading or lagging load power factor.

A potential solution for the voltage distortion based on [**1**] is a hybrid unipolar and bipolar switching scheme. The inverter switches using unipolar modulation, but the inverter

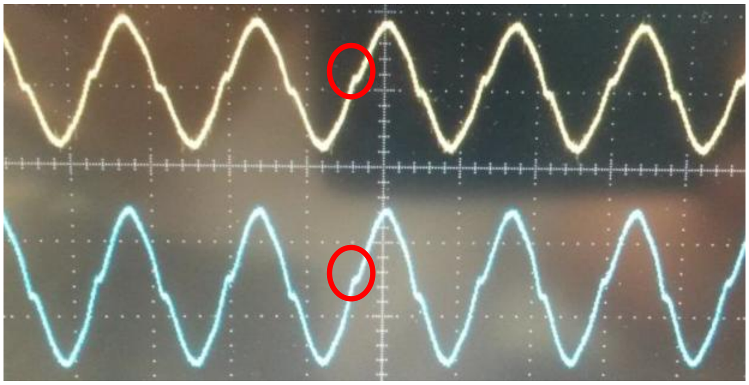


Figure 6-11. Unipolar inverter output distortion due to dead time

changes to bipolar modulation near the zero-voltage crossover points. A simulation is run to compare this modulation scheme to unipolar switching and the resulting voltage waveforms at the output of the LCL filter are shown in Fig. 6-12. In the figure, bipolar modulation is used in a 30 degree band centered on either of the zero crossover points.

The effect on the magnitude of the harmonic current components produced is shown in Fig. 6-13. The magnitude of the odd-integer current harmonics that are produced by the inverter are reduced significantly in the hybrid scheme when compared to the pure unipolar switching but, there is no significant change in the total THD. This is likely due to the higher harmonic content that comes from bipolar switching. If this hybrid modulation scheme were used a more effective output filter would need to be considered.

Fig. 6-14 shows several FFT of the neutral current from simulations a UCSC inverter to demonstrate the benefits of compensating for these effect. Fig. 6-14(a) contains the FFT of the system with dead time and device voltage drops, Fig. 6-14(b) if the FFT when there is no dead time, Fig. 6-14(c) is the FFT when there is no device voltage drop and Fig. 6-14(d) shows the FFT with neither dead time nor device voltage drop. The 3rd harmonic component drop from 0.6 A to 0.18 A with dead time, 0.35 A with no device voltage drop and 0.01 A when neither are

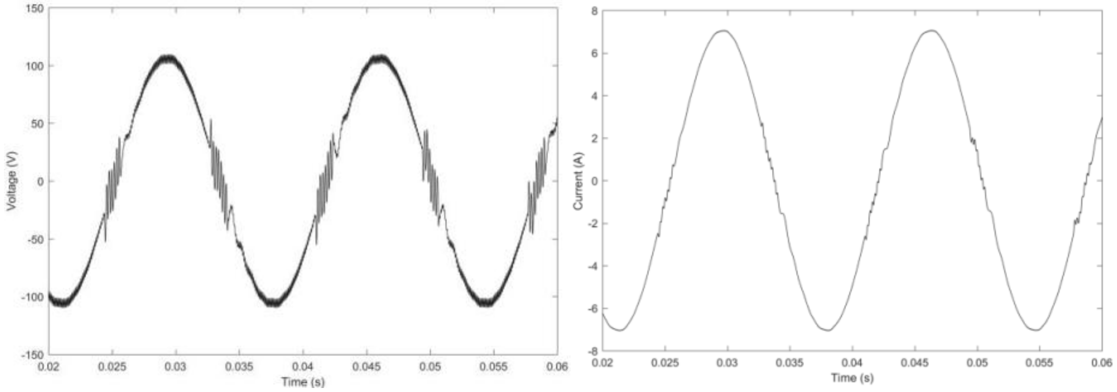


Figure 6-12. Voltage (left) and current (right) waveforms produced from the hybrid modulation scheme

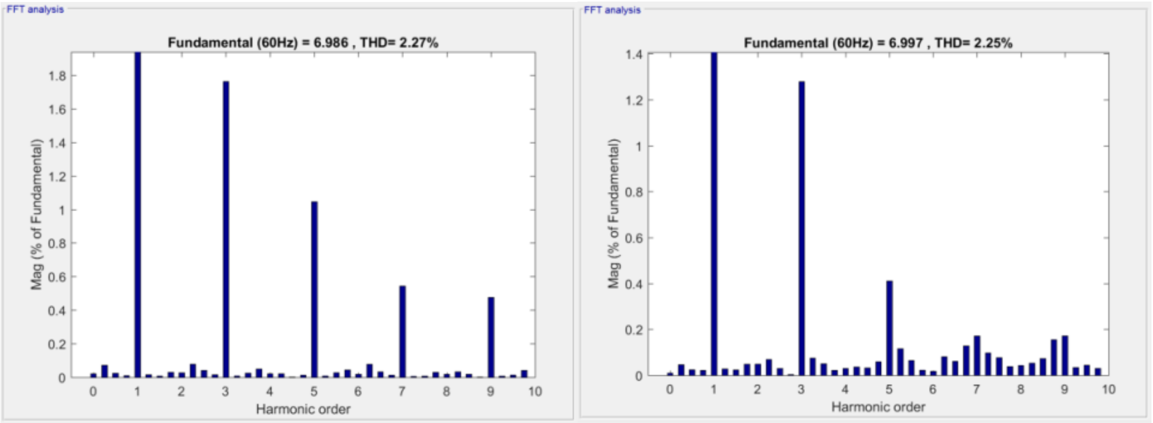
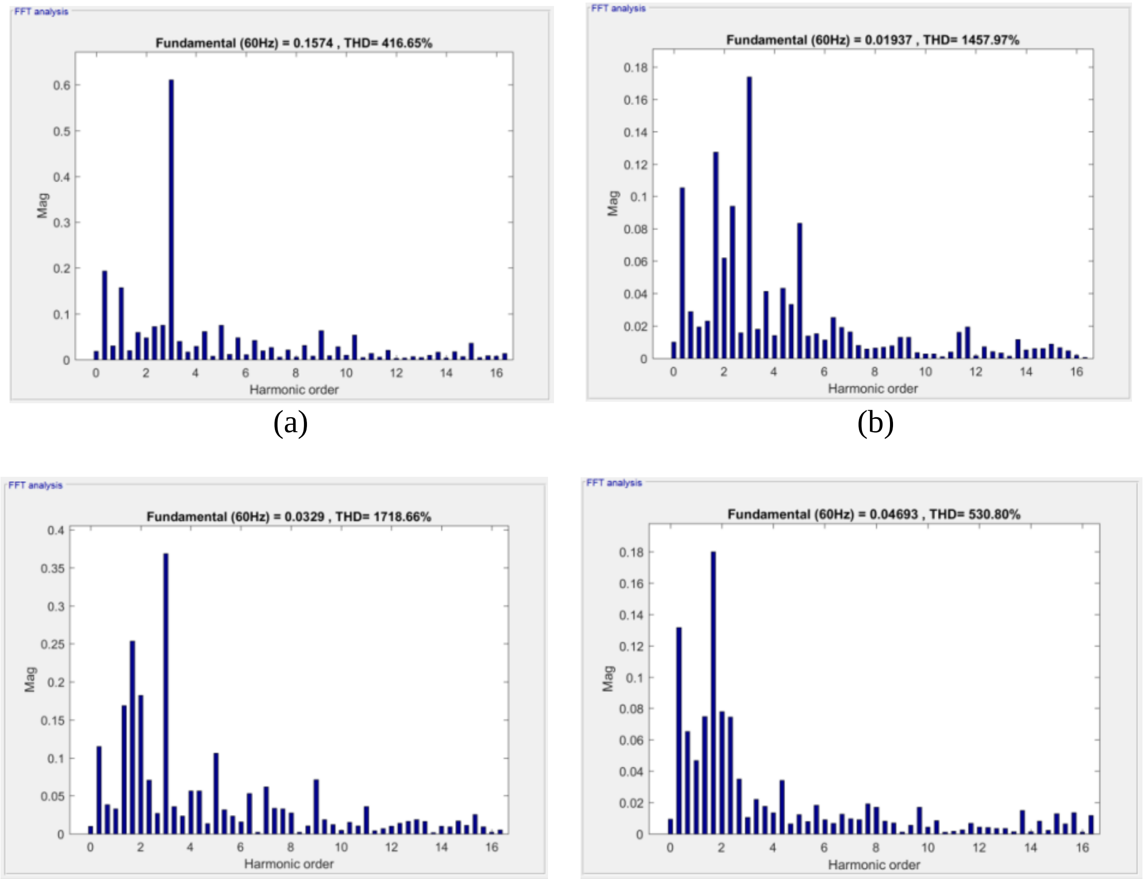


Figure 6-13. FFT of inverter output current with unipolar (left) and hybrid (right) modulation scheme



(a)

(b)

(c)

(d)

Figure 6-14. Simulation FFTs of substation neutral current under different operating conditions

present.

6.4 Effect of Harmonic Voltage Components

The voltage in the power outlet in the testing lab contains a 3rd harmonic component. This voltage distortion is caused by harmonic currents produced by non-linear equipment in the building (e.g., computers). This can cause issues with grid-connected inverters if there is no compensation method for these harmonic voltages. This is evident in Fig. 6-15 below which shows the UCSC working at the same operating point with and without a 5% 3rd harmonic grid voltage component. The compensator is working as intended in both instances but the waveforms on the right show a significant 3rd harmonic current that flows from the source and into the UCSC. This is because there is no mechanism in this version of the UCSC controller that compensates for harmonics present in the grid voltage.

An isolation transformer connected in a delta-wye configuration was used at the source to eliminate this problem associated with this version of the UCSC. Because the 3rd harmonic voltage component in three-phase balanced systems has the same phase angle in each of the

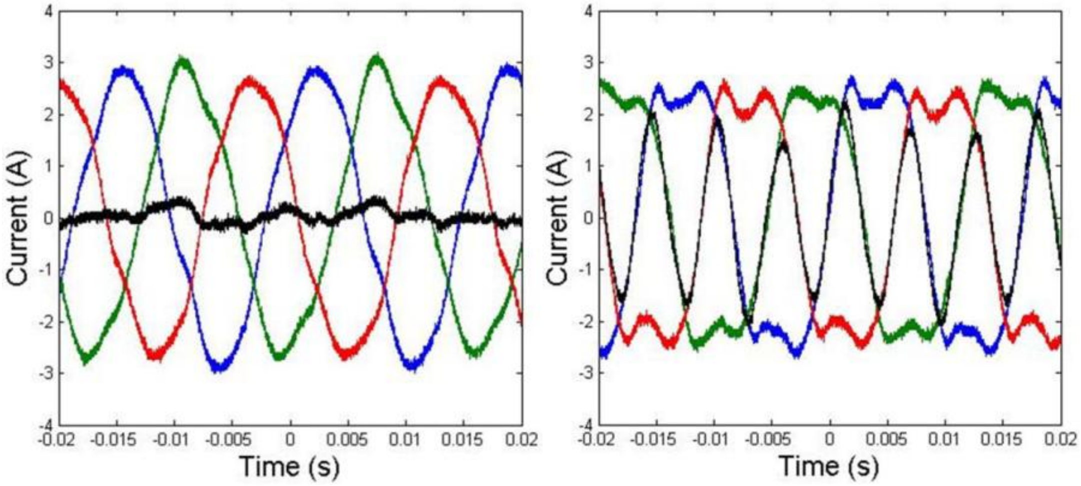


Figure 6-15. Substation currents during UCSC operation without (left) and with (right) grid voltage distortion

three-phases, the resulting line-to-line voltages that the delta-connected side of the isolation transformer experiences is not affected by the distortion due to triplen harmonics [**4**]. This means that the wye-grounded side of the isolation transformer at the source no longer has the 3rd harmonic voltage component, so it is eliminated from the system used to test the UCSC. Obviously, this represents a more ideal system, and thus less indicative of a real-world system, but it enables the testing of this UCSC version which was not designed to compensate for harmonic voltage distortion.

6.5 Effect of Improper Measurements

Using improper probes or testing points can lead to misleading current and voltage measurements. Ideally, the probes used to measure voltage would have unlimited impedance and have no effect on oscilloscope readings. However, actual probes have an impedance network that interacts with parasitics that exist in the circuit being measured. The length of the probe contributes to its parasitic inductance and this along with its effective capacitance can make it unsuitable for high-frequency signals.

The turn-on gate-to-emitter voltage measurement of an IGBT as taken by an unnecessarily long probe is shown in Fig. 6-16 on the left side. There is a significant amount of voltage oscillation in the waveform, which suggests a problem with the gate drive circuitry. The same measurement was taken on the right side, but using a probe that has much shorter leads. The oscillations are no longer present, confirming that the gate driver circuit is preforming as expected.

6.6 Conclusions

The scaled-down UCSC prototype was used to compensate for system currents that had a UBF of 8.8% and a peak fundamental neutral current of 4.0 A. During steady-state operation the

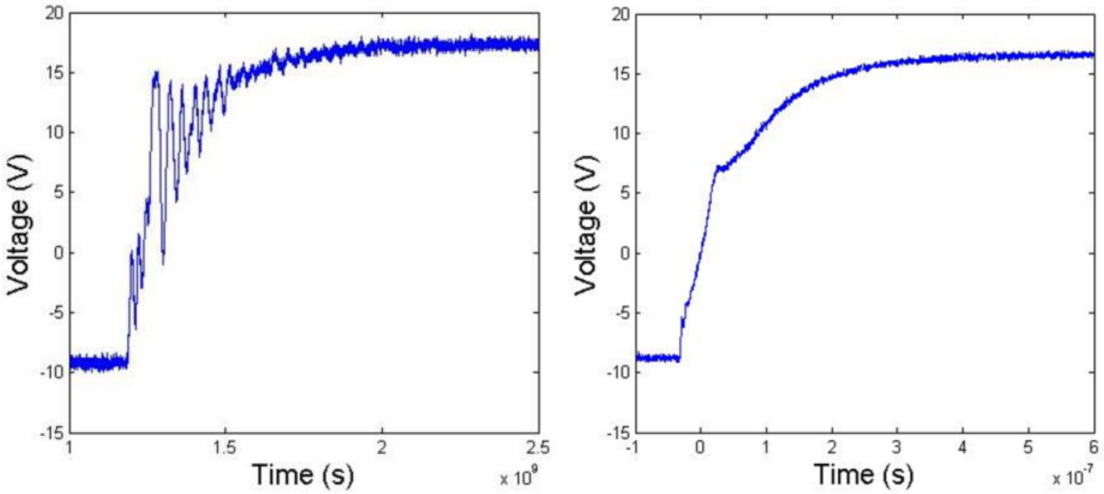


Figure 6-16. IGBT Turn-on Vce without (left) and with (right) appropriate voltage probes

UCSC was able to reduce the UBF of the system to 0.7% and the peak fundamental neutral current to 0.15 A while achieving near unity power factor at the PCC. This proved that the UCSC was effective at compensating for the fundamental negative- and zero-sequence current components produced by unbalanced loads. A 3rd harmonic current of 0.57 A was produced in the neutral conductor due to distortion in the voltage output of the inverters. Three more tests were conducted in an attempt to reduce this output voltage distortion. An increase in the dc-bus voltage from 250 V to 300 V had no effect on the THD of the inverter output current and neither did a small reduction of the dead time from 1.0 μs to 0.833 μs or an increase in the loading of the inverter. It is important to note that results from tests with a distorted grid voltage showed that without the ability to compensate for the 3rd harmonic voltage, 3rd harmonic currents can be expected in the output of the UCSC.

6.7 References

[1] N. Mohan, T. M. Undeland, and W. P. Robbins, *Power Electronics: Converters Applications and Design*, 3rd ed. New Jersey, United States of America: JW&S Inc, 2003.

[2] Seung-Gi Jeong and Min-Ho Park, "The Analysis and Compensation of Dead-Time Effects in PWM Inverters," *IEEE Transaction on Inducstrial Electronics*, vol. 38, no. 2, pp. 108-114, April 1991.

[3] T. F. Wu, C. L. Kuo, K. H. Sun, and H. C. Hsieh, "Combined Unipolar and Bipolar PWM for Current Distortion Improvement During Power Compensation," *IEEE Transactions on Power Electronics*, vol. 29, no. 4, pp. 1702-1709, October 2013.

[4] Roger C. Dugan, Mark F. McGranaghan, Surya Santoso, and H. Wayne Beaty, *Electrical Power Systems Quality*, 3rd ed.: McGraw-Hill, 2012.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

As elaborated in Chapter 1, several adverse effects can arise from unbalanced loading on a distribution feeder. These include: potential overcurrent on the neutral conductor [**1**], electric machine overheating [**2**], negative-sequence MMF interference with positive-sequence MMF [**3**], and overheating of wye-ungrounded transformers [**4**]. An overview of existing solutions for these issues was presented followed by a description of the solution developed in this paper, the Unbalanced Current Static Compensator which is installed just downstream of the distribution power transformer. It consists of three single-phase inverters coupled to grid using readily available and low-cost distribution transformers.

The principles behind the operation of each part of the UCSC as well as a thorough description of the controller used to run the system was shown in Chapter 2. This was followed by the application of those control principles to develop Matlab/Simulink™-based simulations in Chapter 3 that were indicative of the real-world implementation. With the viability of the proposed solution confirmed through the simulation results, the construction of a scaled-down prototype was addressed in Chapter 4. The constructed prototype was built to fully compensate for a 10 kVA, 208 $V_{L\text{-}L}$ system and consisted of three single-phase inverters each rated 1 kVA, 120 $V_{L\text{-}N}$ coupled to the system through 1:1 isolation transformers.

The discrete controller implemented in the DSP was presented in Chapter 5. An explanation of the steps taken to reduce the susceptibility of the DSP to EMI was given in addition to a procedure for soft-starting the compensator and an orderly shutdown procedure to avoid large currents. Using these techniques, testing data from the laboratory prototype were captured and analyzed in Chapter 6. The UCSC was found to be effective at reducing the peak fundamental current component in the neutral from 4.0 A to 0.15 A and the UBF of the system from 8.8 % to approximately 0.7 % with power factor of 0.999 at the PCC. Unfortunately, there was a third harmonic current of 0.57 A that flowed through the neutral conductor because of distortion caused by the dead time used between gate pulses. A dc bus voltage increase from 250 V to 300 V, a dead time reduction from 1.0 μs to 0.833 μs, and an increase in the loading of the inverter were all ineffective for reducing the distortion caused by the dead time.

The UCSC developed for the case study required three 1 MVA inverters to compensate a 6 MVA system in the case study. Rating each inverter in the compensator at nearly 17 % of the rated power for the system is not economical. The installation of power factor correction capacitors close to the UCSC could be a less expensive method for full power factor correction at the substation.

7.2 Recommendations for Future Work

The ensuing subsections contain recommendations for future research based on the existing solutions, the economics of the proposed solution, and the performance of the laboratory prototype.

7.2.1 Implement a method for compensating the distortion due to dead time

To deploy the UCSC in the field the THD that it produces is dictated by standards for grid connected equipment [**5**]. The performance of this UCSC prototype would not meet these standards for grid connection and requires a dead time compensation method such as those presented in [**6**] and [**7**].

7.2.2 Include of a mechanism for compensating for grid voltage distortion

The same argument as above applies to the inability of the controller to compensate for grid voltage distortion. Without this capability the current drawn from the grid by the inverter will contain the same harmonic content that exists in the grid voltage.

7.2.3 Develop applications for the UCSC in mesh networks

The proposed solution is only suited for power flow in one direction along a radial feeder. With the higher penetration of distributed generation, bi-directional power flow may become much more common. An area of research could be finding the appropriate position for the UCSC in this type of system and giving the device the ability to compensate in either direction.

7.2.4 Apply the UCSC control to different three-phase inverter topologies

The proposed solution requires 12 IGBTs and 12 diodes compared to a three-phase four-leg inverter which only requires 8 IGBTs and 8 diodes. Obviously, the four-leg topology is cheaper, but it more complex to control. A performance comparison of these two topologies as well as others such as in Fig. 7-1 can be done.

7.2.5 Augment the UCSC controller to compensate for harmonic currents

Elimination of harmonic currents from the source will improve system efficiency and has some of the same effect as eliminating the fundamental negative- and zero-sequence current components. Because the power electronics provide full control over output voltages, the addition of this ability to the UCSCs controller does not require a redesign of the system.

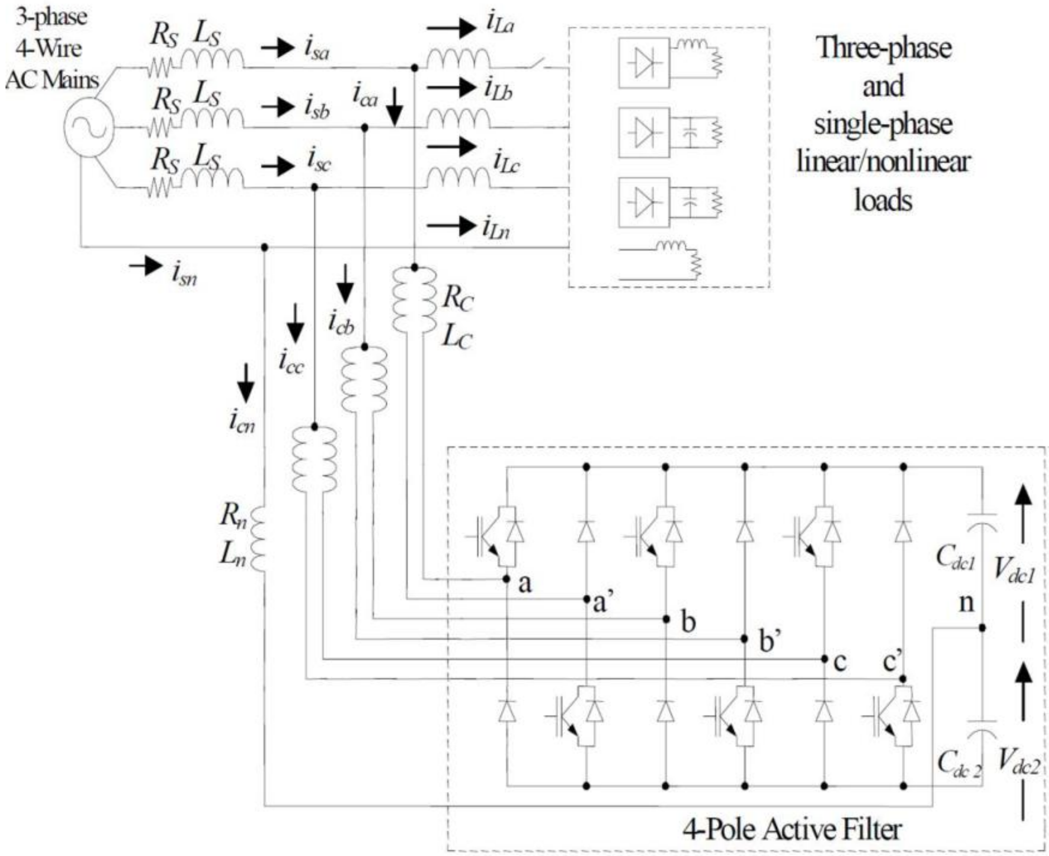


Figure 7-1. Shoot-through current immune topology [9]

7.2.6 Add the ability to partially compensate for reactive power

The majority of the current rating for the UCSC comes from the requirement for full reactive power compensation. The ability to control the power factor at the PCC at a non-unity value would be useful, because it could reduce the power rating required for the UCSC solution and allow the UCSC to maintain its balancing capabilities even when the current limits for the inverters have been reached. Implement a stationary reference frame controller for comparison

Stationary reference frame controllers can be less computationally taxing than synchronous frame controllers because neither a phase-lock loop nor a d-q transformation is required and they may have better a transient response as a result [**8**]. A comparison between an instantaneous power controller in the stationary reference frame such as in [**9**], and the proposed controller could provide insight into advantages and disadvantages of either method.

7.3 References

[1] J. Kennedy, C. A. Nucci, A. Borghetti, G. Contaxis A. P. S. Meliopoulos, "Power distribution practices in USA and Europe: impact on power quality," in *Proceedings of the 8th International Conference on Harmonics and Quality of Power*, Athens, 1998, pp. 24-29.

[2] R. G. Harley, E.B. Makram, and E.G. Duran, "The effects of unbalanced networks on synchronous and asynchronous machine transient stability," *Electric Power System Research*, vol. 13, no. 2, pp. 119-127, October 1987.

[3] R.H. Salim, R.A. Ramos, and N.G. Bretas, "Analysis of the small signal dynamic performance of synchronous generators under unbalanced operating conditions," in *IEEE Power and Energy Society General Meeting*, 2010, pp. 1-6.

[4] T.A. Short, *Electrical Power Distribution Handbook*. Boca Raton, FL: CRC Press Taylor & Taylor Croup, 2004.

[5] Roger C. Dugan, Mark F. McGranaghan, Surya Santoso, and H. Wayne Beaty, *Electrical Power Systems Quality*, 3rd ed.: McGraw-Hill, 2012.

[6] Seung-Gi Jeong and Min-Ho Park, "The Analysis and Compensation of Dead-Time Effects in PWM Inverters," *IEEE Transaction on Inducstrial Electronics*, vol. 38, no. 2, pp. 108-114, April 1991.

[7] T. F. Wu, C. L. Kuo, K. H. Sun, and H. C. Hsieh, "Combined Unipolar and Bipolar PWM for Current Distortion Improvement During Power Compensation," *IEEE Transactions on Power Electronics*, vol. 29, no. 4, pp. 1702-1709, October 2013.

[8] Matias Diaz and Roberto Cardenas, "Analysis of Synchronous and Stationary Reference Frame Control Strategies to Fulfill LVRT Requirements in Wind Energy Conversion Systems," in *Ecological Vehicles and Renewable Energies (EVER), 2014 Ninth International Conference on*, Monte-Carlo, 2014, pp. 1 - 8.

[9] Singh G.N and P Rastgoufard, "A new topology of active filter to correct power-factor, compensate harmonics, reactive power and unbalance of three-phase four-wire loads," in *Applied Power Electronics Conference and Exposition*, Miami Beach, 2003, pp. 141-147.

APPENDIX A

SNUBBERS FOR SEMICONDUCTOR DEVICES

A.1 RCD Snubber

One solution to the problem of the oscillations is the use of a snubber circuit in parallel with the IBGTs. Snubber circuits absorb the stored energy from the inductances and then disperse that energy in a non-destructive way. Several different snubber topologies exist. One of the basics is the RCD snubber used to absorb energy from the turn-off of a transistor. Fig. A-1 shows this snubber structure.

During the IGBT turn-off, the diode conducts the current that the parasitic inductances are trying to maintain [1]. This results in the snubber capacitor being charged up to the applied voltage level. When the IGBT turns on again the voltage across the capacitor drops dramatically and it’s is drained through the snubber resistor. As a result, the resistor must dissipate this

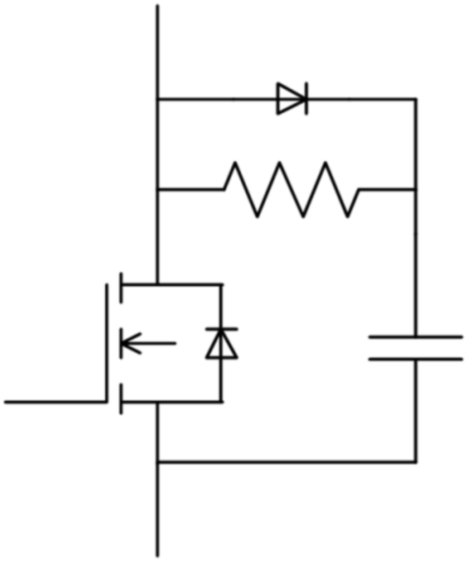


Figure A-1. RCD Snubber Circuit

energy, but this is easier to manage than extra turn-on losses that would result in the IGBT otherwise.

A.2 Overvoltage Snubber

Another form of protection is an overvoltage snubber that acts much like the turn-off snubber, but the capacitor is connected across the dc-bus instead of the IGBT, and remains charged to the bus value [1]. The diode conducts and charges the capacitor when the voltage of the bus becomes larger than that of the capacitor like when there are oscillations. The excess energy in the capacitor is drained through a resistor back to the dc bus. The circuit for this is displayed in Fig. A-2.

A.3 Turn-On Snubber

In some cases, such as in half-bridge circuits, there are adverse effects that come from the use of turn-off snubbers like the one mentioned above. This is due to the charging currents that the snubber capacitors draw each switch cycle [1]. The bottom IGBT takes the load current when turning on, but there is an extra current component flowing through the bottom switch because the top switch’s snubber capacitor is charging. Because this extra current occurs during the turn-on of the bottom switch, when both the voltage and current are non-zero, it can cause a significant increase in switching losses for the bottom switch. To mitigate the extra losses that might arise from this arrangement a turn-on snubber can be employed. One type of turn-on snubber uses an inductor in series with the switch. This inductor, due to the changing current during the turn-on of a device, takes some of the voltage that would normally be experienced completely by the switch. At turn-off, the energy stored in the inductor is bled through a series diode and resistor branch that is in parallel with the inductor. This avoids voltage oscillations

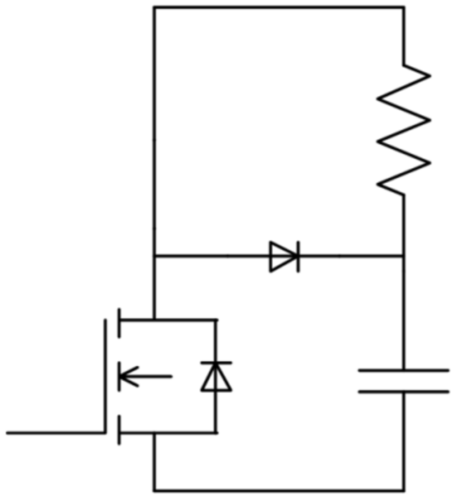


Figure A-2. Overvoltage snubber

across the switch. A drawback to using this solution is the complexity it adds to the circuit.

A.4 Undeland Snubber

Employing all three of these snubbers on a half bridge circuit can be expensive and requires significant space on the PCB, which can be a problem from a power density perspective. The Undeland snubber combines all three of these snubbers into a hybrid snubber with a reduced footprint and amount of components [1]. Fig. A-3 shows an implementation of this snubber.

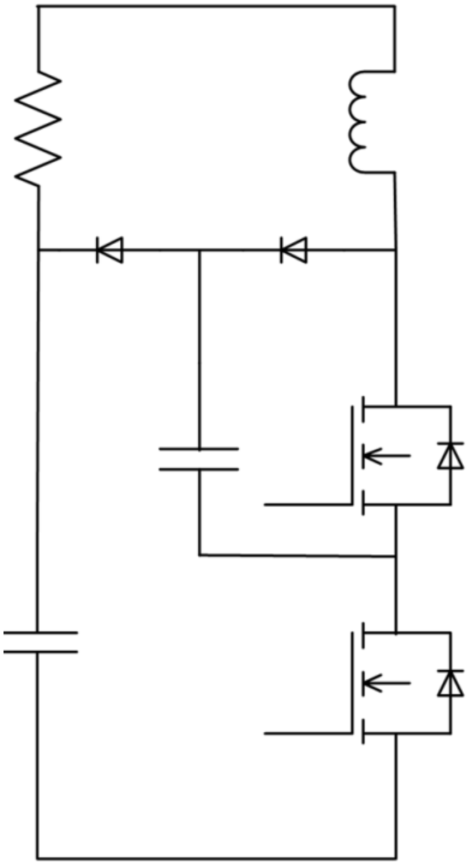


Figure A-3. Undeland Snubber Circuit

A.5 References

[1] N. Mohan, T. M. Undeland, and W. P. Robbins, *Power Electronics: Converters Applications and Design*, 3rd ed. New Jersey, United States of America: JW&S Inc, 2003.

APPENDIX B

DIGITAL SIGNAL PROCESSOR CODE

Written in C below is the main.c file that is uploaded to the DSP. This contains the register initialization presented in chapter 5 and the code used to implement the controller.

B.1 The Main.c File

```
#include "DSP28x_Project.h"     // Device Headerfile and Examples Include File
#include "Solar_F.h"
#include "UCSC_Functions.h"
#include "Phase.h"
#include "SPLL_1ph_SOGI_MOD.h"

#define PI 3.141592654
#define TWO_PI 6.283185308
#define Grid_freq 60.0 // control signal frequency of 60 Hz
#define VoltagePeakMax 220.0 // Maximum instantaneous voltage measured by the voltage
sensors
#define dcbusref 250.0 // Reference level for the DC bus
#define omegaLcoup 12.02 // 2*pi*60*Lcoup*CurrentMax ( 0.85 * 10 * sqrt(2) ),
accounts for scaling
#define FilterR 0.048
#define FilterL 0.00226
#define CurrentMax 14.14 // the maximum instantaneous current measured by the current
sensors
#define PWM1_TIMER_TBPRD 15000 // 15000 = 5 kHz, 7500 = 10 kHz, 18750 = 4 kHz,
Configure the period for ePWM Time-Based Counter
#define rc_freq 10.0
#define cc_freq 2000.0
#define deadtimecycles 150

#define FLASH_SELECT 1

///////////////////////////////////////////////////////////////////////////////////////
// If you change this you must change the discrete filter constants./////////
#define MEASUREFREQ 20000.0 //frequency at which values are
sampled///////////////////////////////
///////////////////////////////////////////////////////////////////////////////////////

#pragma CODE_SECTION(CurrentControl, "UserFuncs");
#pragma CODE_SECTION(ReferenceGenerator, "UserFuncs");
#pragma CODE_SECTION(OverCurrentProtection, "UserFuncs");
#pragma CODE_SECTION(cpu_timer0_isr, "UserFuncs");

// Prototype statements for functions found within this file. //

void AdcInitialize(void); // ADC initializing function
interrupt void cpu_timer0_isr(void); //
```

```
void InitEPwm1Timer(void); // ePWM modules
void InitEPwm2Timer(void);
void InitEPwm3Timer(void);
void InitEPwm4Timer(void);
void InitEPwm5Timer(void);
void InitEPwm6Timer(void);

void CurrentControl(void); // current controller function
void ReferenceGenerator(void); // PI error generator function
void OverCurrentProtection(void);

//Variables to update the PWM registers
float control_a = 0.0, control_b = 0.0, control_c = 0.0;

int ccbit = 0;

// Class objects for sampling ADC
ADC VaADC, IainvADC, IaLADC, VbADC, IbinvADC, IbLADC, VcADC, IcinvADC, IcLADC,
VbusADC;

// Class objects for PI controllers
PI_controller Curr_Cont_Da, Curr_Cont_Qa, Curr_Cont_Refa, Curr_Cont_Db, Curr_Cont_Qb,
Curr_Cont_Refb, Curr_Cont_Dc, Curr_Cont_Qc, Curr_Cont_Refc;

// Class objects for OSG and d-q transformations
SPLL_1ph_SOGI_F PARKS_Ia_inv, PARKS_Ia_L, VaPLL, PARKS_Ib_inv, PARKS_Ib_L, VbPLL,
PARKS_Ic_inv, PARKS_Ic_L, VcPLL;

// Class objects for notch filters
NotchFilter VbusNotch;

int i = 0, j = 0, ocdelay = 0;
int ClearTrip = 0, RunControl = 0, EnablePWM = 1;
float gain_a = 0.0, alpha_a = 0.0, gain_b = 0.0, alpha_b = 0.0, gain_c = 0.0, alpha_c
= 0.0;
float DaxisMag_a = 0.0, QaxisMag_a = 0.0, DaxisMag_b = 0.0, QaxisMag_b = 0.0,
DaxisMag_c = 0.0, QaxisMag_c = 0.0;
float dead_comp_a = 0.0, dead_comp_b = 0.0, dead_comp_c = 0.0;

//static float radian_a = 0, radian_b = 0, radian_c = 0, Ts =
TWO_PI*Grid_freq/MEASUREFREQ;
float PhALegA = 0.0, PhALegB = 0.0, PhBLegA = 0.0, PhBLegB = 0.0, PhCLegA = 0.0,
PhCLegB = 0.0;
float ia = 0.0,ib = 0.0,ic = 0.0;


void main(void)
{

#if FLASH_SELECT
        // This function is found in the MemCopy.c file and is required for stand-
alone mode
        MemCopy(&RamfuncsLoadStart, &RamfuncsLoadEnd, &RamfuncsRunStart);
        MemCopy(&fastRTS_LoadStart, &fastRTS_LoadEnd, &fastRTS_RunStart);
        MemCopy(&UserFuncsLoadStart, &UserFuncsLoadEnd, &UserFuncsRunStart);
        InitFlash();
#endif

        // PLL, WatchDog, enable Peripheral Clocks
        // This example function is found in the DSP2833x_SysCtrl.c file.
        InitSysCtrl();

        // Initialize PIE control registers to their default state.
        // The default state is all PIE interrupts disabled and flags
        // are cleared.
        // This function is found in the DSP2833x_PieCtrl.c file.
        InitPieCtrl();

        // Initialize the PIE vector table with pointers to the shell Interrupt
        // Service Routines (ISR).
        // This will populate the entire table, even if the interrupt
        // is not used in this example.  This is useful for debug purposes.
        // The shell ISR routines are found in DSP2833x_DefaultIsr.c.
        // This function is found in DSP2833x_PieVect.c.
        InitPieVectTable();

        // Step 3. Clear all interrupts and initialize PIE vector table:

        DINT; // Disable CPU interrupts

        // Disable CPU interrupts and clear all CPU interrupt flags:
        IER = 0x0000;
        IFR = 0x0000;

        //------ Populate the Interrupt vector addresses -------//

        EALLOW;
        PieVectTable.TINT0 = &cpu_timer0_isr;
        EDIS;

        SPLL_1ph_SOGI_F_init(Grid_freq,((float)(1.0/(MEASUREFREQ))),&VaPLL);
        SPLL_1ph_SOGI_F_coeff_update(((float)(1.0/(MEASUREFREQ))),(float)(2*PI*Grid_fr
eq),&VaPLL);

        SPLL_1ph_SOGI_F_init(Grid_freq,((float)(1.0/(MEASUREFREQ))),&PARKS_Ia_inv);
        SPLL_1ph_SOGI_F_coeff_update(((float)(1.0/(MEASUREFREQ))),(float)(2*PI*Grid_fr
eq),&PARKS_Ia_inv);

        SPLL_1ph_SOGI_F_init(Grid_freq,((float)(1.0/(MEASUREFREQ))),&PARKS_Ia_L);
        SPLL_1ph_SOGI_F_coeff_update(((float)(1.0/(MEASUREFREQ))),(float)(2*PI*Grid_fr
eq),&PARKS_Ia_L);

        SPLL_1ph_SOGI_F_init(Grid_freq,((float)(1.0/(MEASUREFREQ))),&VbPLL);
        SPLL_1ph_SOGI_F_coeff_update(((float)(1.0/(MEASUREFREQ))),(float)(2*PI*Grid_fr
eq),&VbPLL);

        SPLL_1ph_SOGI_F_init(Grid_freq,((float)(1.0/(MEASUREFREQ))),&PARKS_Ib_inv);
        SPLL_1ph_SOGI_F_coeff_update(((float)(1.0/(MEASUREFREQ))),(float)(2*PI*Grid_fr
eq),&PARKS_Ib_inv);
```

```
	SPLL_1ph_SOGI_F_init(Grid_freq,((float)(1.0/(MEASUREFREQ))),&PARKS_Ib_L);
	SPLL_1ph_SOGI_F_coeff_update(((float)(1.0/(MEASUREFREQ))),(float)(2*PI*Grid_fr
eq),&PARKS_Ib_L);

	SPLL_1ph_SOGI_F_init(Grid_freq,((float)(1.0/(MEASUREFREQ))),&VcPLL);
	SPLL_1ph_SOGI_F_coeff_update(((float)(1.0/(MEASUREFREQ))),(float)(2*PI*Grid_fr
eq),&VcPLL);

	SPLL_1ph_SOGI_F_init(Grid_freq,((float)(1.0/(MEASUREFREQ))),&PARKS_Ic_inv);
	SPLL_1ph_SOGI_F_coeff_update(((float)(1.0/(MEASUREFREQ))),(float)(2*PI*Grid_fr
eq),&PARKS_Ic_inv);

	SPLL_1ph_SOGI_F_init(Grid_freq,((float)(1.0/(MEASUREFREQ))),&PARKS_Ic_L);
	SPLL_1ph_SOGI_F_coeff_update(((float)(1.0/(MEASUREFREQ))),(float)(2*PI*Grid_fr
eq),&PARKS_Ic_L);

	//----------- PI Controller Initialization -------------//

	Curr_Cont_Da.Ts = (float)(1.0/(MEASUREFREQ)); //Period of Current controller
	Curr_Cont_Da.Ki = FilterR * cc_freq * CurrentMax; //Define the PI constants
for Current Controller
	Curr_Cont_Da.Kp = FilterL * cc_freq * CurrentMax;
	PI_controller_init(&Curr_Cont_Da);
	Curr_Cont_Qa.Ts = (float)(1.0/(MEASUREFREQ));
	Curr_Cont_Qa.Ki = FilterR * cc_freq * CurrentMax;
	Curr_Cont_Qa.Kp = FilterL * cc_freq * CurrentMax;
	PI_controller_init(&Curr_Cont_Qa);
	Curr_Cont_Refa.Ts = (float)(1.0/(MEASUREFREQ));
	Curr_Cont_Refa.Ki = ( TWO_PI * rc_freq ) * ( TWO_PI * rc_freq ) * 0.0022; // /
3.0; //0.08 * 1.5;
	Curr_Cont_Refa.Kp = sqrt(2) * TWO_PI * rc_freq * 0.0022; //0.05 * 1.5;
	PI_controller_init(&Curr_Cont_Refa);

	Curr_Cont_Db.Ts = (float)(1.0/(MEASUREFREQ)); //Period of Current controller
	Curr_Cont_Db.Ki = FilterR * cc_freq * CurrentMax; //Define the PI constants
for Current Controller
	Curr_Cont_Db.Kp = FilterL * cc_freq * CurrentMax;
	PI_controller_init(&Curr_Cont_Db);
	Curr_Cont_Qb.Ts = (float)(1.0/(MEASUREFREQ));
	Curr_Cont_Qb.Ki = FilterR * cc_freq * CurrentMax;
	Curr_Cont_Qb.Kp = FilterL * cc_freq * CurrentMax;
	PI_controller_init(&Curr_Cont_Qb);
	Curr_Cont_Refb.Ts = (float)(1.0/(MEASUREFREQ));
	Curr_Cont_Refb.Ki = ( TWO_PI * rc_freq ) * ( TWO_PI * rc_freq ) * 0.0022; // /
3.0; //0.08 * 1.5;
	Curr_Cont_Refb.Kp = sqrt(2) * TWO_PI * rc_freq * 0.0022; //0.05 * 1.5;
	PI_controller_init(&Curr_Cont_Refb);

	Curr_Cont_Dc.Ts = (float)(1.0/(MEASUREFREQ)); //Period of Current controller
	Curr_Cont_Dc.Ki = FilterR * cc_freq * CurrentMax; //Define the PI constants
for Current Controller
	Curr_Cont_Dc.Kp = FilterL * cc_freq * CurrentMax;
	PI_controller_init(&Curr_Cont_Dc);
	Curr_Cont_Qc.Ts = (float)(1.0/(MEASUREFREQ));
	Curr_Cont_Qc.Ki = FilterR * cc_freq * CurrentMax;
	Curr_Cont_Qc.Kp = FilterL * cc_freq * CurrentMax;
	PI_controller_init(&Curr_Cont_Qc);
	Curr_Cont_Refc.Ts = (float)(1.0/(MEASUREFREQ));
	Curr_Cont_Refc.Ki = ( TWO_PI * rc_freq ) * ( TWO_PI * rc_freq ) * 0.0022; // /
3.0; //0.08 * 1.5;
	Curr_Cont_Refc.Kp = sqrt(2) * TWO_PI * rc_freq * 0.0022; //0.05 * 1.5;
	PI_controller_init(&Curr_Cont_Refc);

	//------------- Initialize Notch Filter --------------//

	Notch_Init(((float)(1.0/(MEASUREFREQ))),(float)(2*PI*Grid_freq),(float)0.00001
,(float)0.1,&VbusNotch);

	//--------- ADC Read Function Initialization -----------//

	VaADC.midpoint = 2062;
	IainvADC.midpoint = 2044;
	IaLADC.midpoint = 2042;
	VbADC.midpoint = 2061;
	IbinvADC.midpoint = 2042;
	IbLADC.midpoint = 2055;
	VcADC.midpoint = 2063;
	IcinvADC.midpoint = 2044;
	IcLADC.midpoint = 2035;
	VbusADC.midpoint = 0;

	//----------- Initialize Device Peripherals ------------//

	InitEPwm1Timer();    // Initialize ePWMs
	InitEPwm2Timer();
	InitEPwm3Timer();
	InitEPwm4Timer();
	InitEPwm5Timer();
	InitEPwm6Timer();

	EPwm1Regs.TZCLR.bit.OST = 1;
	EPwm2Regs.TZCLR.bit.OST = 1;
	EPwm3Regs.TZCLR.bit.OST = 1;
	EPwm4Regs.TZCLR.bit.OST = 1;
	EPwm5Regs.TZCLR.bit.OST = 1;
	EPwm6Regs.TZCLR.bit.OST = 1;

	EALLOW;
	EPwm1Regs.TZFRC.bit.OST = 1;
	EPwm2Regs.TZFRC.bit.OST = 1;
	EPwm3Regs.TZFRC.bit.OST = 1;
	EPwm4Regs.TZFRC.bit.OST = 1;
	EPwm5Regs.TZFRC.bit.OST = 1;
	EPwm6Regs.TZFRC.bit.OST = 1;
	EDIS;

	InitAdc(); // Init the ADC as far as turning it on, enabling clock,
calibration
	AdcInitialize(); // Init the ADC for program specific settings
```

```
    InitCpuTimers();
    ConfigCpuTimer(&CpuTimer0, 150, (1/MEASUREFREQ)*1000000); // Configure the period
for the CPU TIMER0
    CpuTimer0Regs.TCR.all = 0x4001; // write to the entire register to make sure
everything is set correctly after the InitCpuTimers() and ConfigCpuTimer() functions

      //--------------- Define GPIO pins used ----------------//

    EALLOW;
    // Inputs
    GpioCtrlRegs.GPAMUX2.bit.GPIO31 = 0; // '0' means general purpose use. otherwise
it is used by a peripheral
    GpioCtrlRegs.GPADIR.bit.GPIO31 = 0; // 0 = input, 1 = output
    GpioCtrlRegs.GPAMUX2.bit.GPIO29 = 0;
    GpioCtrlRegs.GPADIR.bit.GPIO29 = 0;
    GpioCtrlRegs.GPBMUX1.bit.GPIO34 = 0;
    GpioCtrlRegs.GPBDIR.bit.GPIO34 = 0;
    GpioCtrlRegs.GPBMUX1.bit.GPIO32 = 0;
    GpioCtrlRegs.GPBDIR.bit.GPIO32 = 0;

    GpioCtrlRegs.GPAMUX2.bit.GPIO30 = 0;
    GpioCtrlRegs.GPADIR.bit.GPIO30 = 0;
    GpioCtrlRegs.GPAPUD.bit.GPIO30 = 0;

    // Outputs

    EDIS;

      AdcRegs.ADCTRL2.bit.SOC_SEQ1 = 1; // Start of conversion bit for sequence 1

      //-------------- Turn on/Enable Interrupts -------------//

      PieCtrlRegs.PIEIER1.bit.INTx7 = 1; // Enable the CPUTIMER0 INTn in the PIE

      IER |= M_INT1; // Enable CPU INT1 which is connected to TINT0

      EINT;   // Enable Global interrupt INTM
      ERTM;   // Enable Global realtime interrupt DBGM

      EALLOW;
      SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 1;         // Start all the timers sync'd
      EDIS;

      for(;;)
      {

             OverCurrentProtection();

             CurrentControl();

      }
}

void AdcInitialize(void)
{
	// Specific ADC setup for this example:
	AdcRegs.ADCTRL1.bit.ACQ_PS = 3;  // Determine the number of clock cycles the
Sample and Hold circuit holds the charge.
	AdcRegs.ADCTRL3.bit.ADCCLKPS = 3; // Determine the clock speed of the ADC.
	// ADCCLK = HSPCLK / (ADCCLKPS*2) / (CPS+1)
	// ADC module clock = 75.0MHz / (3*2) / (1+1) = 6.25MHz
	AdcRegs.ADCTRL1.bit.CPS = 1;                        // Clock prescalar. 0 =
(clock/1) , 1 = (clock/2)
	AdcRegs.ADCREFSEL.bit.REF_SEL = 0x0;     // Selects the internal voltage
reference, which should be 3 V

	AdcRegs.ADCTRL3.bit.SMODE_SEL = 0;  // Sets the mode of operation. 1 =
simultaneous, 0 = sequential
	AdcRegs.ADCMAXCONV.all = 0x0009;        // Sets the maximum number of auto
conversions. There will be ADCMAXCONV + 1 conversions
	AdcRegs.ADCTRL1.bit.SEQ_CASC = 1;       // 1  Cascaded mode. So both
sequencers are acting as one.

	AdcRegs.ADCCHSELSEQ1.bit.CONV00 = 0x0; // Sets up a conversion from ADCINA0;
which samples Iainv
	AdcRegs.ADCCHSELSEQ1.bit.CONV01 = 0x1; // ADCINA1; IaL
	AdcRegs.ADCCHSELSEQ1.bit.CONV02 = 0x2; // ADCINA2; Va
	AdcRegs.ADCCHSELSEQ1.bit.CONV03 = 0x3; // ADCINA3; Vbus
	AdcRegs.ADCCHSELSEQ2.bit.CONV04 = 0x4;  // ADCINA4; Ibinv
	AdcRegs.ADCCHSELSEQ2.bit.CONV05 = 0x5;  // ADCINA5; IbL
	AdcRegs.ADCCHSELSEQ2.bit.CONV06 = 0x6;  // ADCINA6; Vb
	AdcRegs.ADCCHSELSEQ2.bit.CONV07 = 0x7;  // ADCINA7; Icinv
	AdcRegs.ADCCHSELSEQ3.bit.CONV08 = 0x8;  // ADCINB0; IcL
	AdcRegs.ADCCHSELSEQ3.bit.CONV09 = 0x9;  // ADCINB1; Vc

	AdcRegs.ADCTRL1.bit.CONT_RUN = 1;       // Setup whether or not it is in
continuous run mode. '0' means not.

	/// Setup the offset register. Refer to page 31 of the ADC documentation.
	AdcRegs.ADCOFFTRIM.bit.OFFSET_TRIM = 0x000; // The format is 2's complement
which allows negative offset correction. Only 9 bits.
}

void InitEPwm1Timer()
{

	EALLOW;
	SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0;      // Stop all the TB clocks
	EDIS;

	/// Initialize the Ports (GPIO Pins)
	InitEPwm1Gpio(); // enables pull-ups on ePWM1's output pins and select ePWM1's
output pins
	/// Determine the Clock Prescalar
	EPwm1Regs.TBCTL.bit.HSPCLKDIV = 0; // Corresponds to a scalar of 1. This sets
the clock to be 150 MHz. (T = 6.666667 ns)
	/// Determine the Timer Clock Prescalar
	EPwm1Regs.TBCTL.bit.CLKDIV = 0; // Corresponds to a scalar of of 1. This keeps
the clock at 150 Mhz.
	/// Period of Timer
```

```
       EPwm1Regs.TBPRD = PWM1_TIMER_TBPRD;
       /// Phase Shift-
       EPwm1Regs.TBPHS.half.TBPHS = 0; // Upon phase syncing. The counter resets to
this value.
       // This is also the starting phase shift of the module. If it is ignoring
phase synching then this just the starting point.
       /// Counting Mode of Timer
       EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;    // Count up to TBPRD and the
immediately count back down to 0. Allows for symmetrical PWM signal.
       /// Allow each timer to be sync'ed
       EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE; // This bit determines whether or not
the ePWM module acknowledges the phase synch. Right now it's the 'Master Module'.
       /// Determine the use of Shadow Period Load Register
       EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW; // Loads the TBPRD register when the
timer counts to 0. This is good for changing periods, which is not the case here.
       /// Setup Sync
       EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_CTR_ZERO; // This determines how the module
deals with phase synching. This has all down-stream modules synced with PWM1.
       /// Determine use of Shadow Counter-Compare Registers (A and B)
       EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW; // This uses the shadow register.
This is good when counter-compare values are constantly changing like in PWM.
       EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW; // see line above. This is for the
counter value B.
       /// Determine when Counter-Compare Module is loaded
       EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO_PRD; // This loads the values for
register A at counter = 0.
       EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO_PRD; // This loads the values for
register B at counter = 0.
       /// Determines PWM action for counter = counter-compare
       EPwm1Regs.AQCTLA.bit.CAU = AQ_SET; // set on up count
       EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR; // clears on down count
       // only A is set because B is complement unless specified otherwise in POLSEL
register. Need CAU and CAD because of UPDOWN count.
       /// Enable the Dead-Band Module
       EPwm1Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE; // enables output dead-band for
EPWM1A and EPWM1B. Necessary for avoiding shoot-through current.
       /// Determine the relationship between A and B
       EPwm1Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC; //Active High Complementary Mode. B
is opposite of A.
       /// Determine the Dead-Band length
       EPwm1Regs.DBRED = deadtimecycles; // Number of clock cycles for rise. (50 *
6.666667ns)
       EPwm1Regs.DBFED = deadtimecycles; // NUmber of clock cycles for fall.

       EALLOW;
       // Trip zone 1 used for ePWM1
       EPwm1Regs.TZSEL.bit.OSHT1 = 1;
       EPwm1Regs.TZSEL.bit.CBC1 = 1;
       // What do we want the TZ1 to do?
       EPwm1Regs.TZCTL.bit.TZA = TZ_FORCE_LO;
       EPwm1Regs.TZCTL.bit.TZB = TZ_FORCE_LO;
       // Enable TZ interrupt
       EPwm1Regs.TZEINT.bit.OST = 1;
       EPwm1Regs.TZEINT.bit.CBC = 1;
       EDIS;
```

```
}

void InitEPwm2Timer()
{

        EALLOW;
        SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0;      // Stop all the TB clocks
        EDIS;

        /// Initialize the Ports (GPIO Pins)
        InitEPwm2Gpio(); // enables pull-ups on ePWM1's output pins and select ePWM1's
output pins
        /// Determine the Clock Prescalar
        EPwm2Regs.TBCTL.bit.HSPCLKDIV = 0; // Corresponds to a scalar of 1. This sets
the clock to be 150 MHz. (T = 6.666667 ns)
        /// Determine the Timer Clock Prescalar
        EPwm2Regs.TBCTL.bit.CLKDIV = 0; // Corresponds to a scalar of 1. This keeps
the clock at 150 Mhz.
        /// Period of Timer
        EPwm2Regs.TBPRD = PWM1_TIMER_TBPRD;
        /// Phase Shift-
        EPwm2Regs.TBPHS.half.TBPHS = 0; // Upon phase syncing. The counter resets to
this value.
        // This is also the starting phase shift of the module. If it is ignoring
phase synching then this just the starting point.
        /// Counting Mode of Timer
        EPwm2Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;    // Count up to TBPRD and the
immediately count back down to 0. Allows for symmetrical PWM signal.
        /// Allow each timer to be sync'ed
        EPwm2Regs.TBCTL.bit.PHSEN = TB_ENABLE; // This bit determines whether or not
the ePWM module acknowledges the phase synch. Right now it's a 'Slave Module'.
        /// Determine the use of Shadow Period Load Register
        EPwm2Regs.TBCTL.bit.PRDLD = TB_SHADOW; // Loads the TBPRD register when the
timer counts to 0. This is good for changing periods, which is not the case here.
        /// Setup Sync
        EPwm2Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN; // This determines how the module
deals with phase synching. This takes in a sync signal from upstream.
        /// Determine use of Shadow Counter-Compare Registers (A and B)
        EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW; // This uses the shadow register.
This is good when counter-compare values are constantly changing like in PWM.
        EPwm2Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW; // see line above. This is for the
counter value B.
        /// Determine when Counter-Compare Module is loaded
        EPwm2Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // This loads the values for
register A at counter = 0.
        EPwm2Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // This loads the values for
register B at counter = 0.
        /// Determines PWM action for counter = counter-compare
        EPwm2Regs.AQCTLA.bit.CAU = AQ_SET; // toggles output for up-count
        EPwm2Regs.AQCTLA.bit.CAD = AQ_CLEAR; // does nothing for down-count
        // only A is set because B is complement unless specified otherwise in POLSEL
register. Need CAU and CAD because of UPDOWN count.
        /// Enable the Dead-Band Module
        EPwm2Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE; // enables output dead-band for
EPWM1A and EPWM1B. Necessary for avoiding shoot-through current.
```

```
	/// Determine the relationship between A and B
	EPwm2Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC; //Active High Complementary Mode. B 
is opposite of A.
	/// Determine the Dead-Band length
	EPwm2Regs.DBRED = deadtimecycles; // Number of clock cycles for rise. (50 * 
6.666667ns)
	EPwm2Regs.DBFED = deadtimecycles; // NUmber of clock cycles for fall.

	EPwm2Regs.ETSEL.bit.INTSEL = ET_CTR_ZERO;     // Select INT on Zero event
	EPwm2Regs.ETSEL.bit.INTEN = 0;  // Enable INT
	EPwm2Regs.ETPS.bit.INTPRD = ET_1ST;           // Generate INT on 1st event, 
can also do 2nd and 3rd

	EALLOW;
	// Trip Zone 1 used for ePWM2
	EPwm2Regs.TZSEL.bit.OSHT1 = 1;
	EPwm2Regs.TZSEL.bit.CBC1 = 1;
	// What do we want the TZ1 to do?
	EPwm2Regs.TZCTL.bit.TZA = TZ_FORCE_LO;
	EPwm2Regs.TZCTL.bit.TZB = TZ_FORCE_LO;
	// Enable TZ interrupt
	EPwm2Regs.TZEINT.bit.OST = 1;
	EPwm2Regs.TZEINT.bit.CBC = 1;
	EDIS;

}

void InitEPwm3Timer()
{

	EALLOW;
	SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0;      // Stop all the TB clocks
	EDIS;

	/// Initialize the Ports (GPIO Pins)
	InitEPwm3Gpio(); // enables pull-ups on ePWM1's output pins and select ePWM1's 
output pins
	/// Determine the Clock Prescalar
	EPwm3Regs.TBCTL.bit.HSPCLKDIV = 0; // Corresponds to a scalar of 1. This sets 
the clock to be 150 MHz. (T = 6.666667 ns)
	/// Determine the Timer Clock Prescalar
	EPwm3Regs.TBCTL.bit.CLKDIV = 0; // Corresponds to a scalar of of 1. This keeps 
the clock at 150 Mhz.
	/// Period of Timer
	EPwm3Regs.TBPRD = PWM1_TIMER_TBPRD;
	/// Phase Shift-
	EPwm3Regs.TBPHS.half.TBPHS = 0; // Upon phase syncing. The counter resets to 
this value.
	// This is also the starting phase shift of the module. If it is ignoring 
phase synching then this just the starting point.
	/// Counting Mode of Timer
	EPwm3Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;    // Count up to TBPRD and the 
immediately count back down to 0. Allows for symmetrical PWM signal.
	/// Allow each timer to be sync'ed
	EPwm3Regs.TBCTL.bit.PHSEN = TB_ENABLE; // This bit determines whether or not
```

```
the ePWM module acknowledges the phase synch. Right now it's a 'Slave Module'.
    /// Determine the use of Shadow Period Load Register
    EPwm3Regs.TBCTL.bit.PRDLD = TB_SHADOW; // Loads the TBPRD register when the
timer counts to 0. This is good for changing periods, which is not the case here.
    /// Setup Sync
    EPwm3Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN; // This determines how the module
deals with phase synching. This takes in a sync signal from upstream.
    /// Determine use of Shadow Counter-Compare Registers (A and B)
    EPwm3Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW; // This uses the shadow register.
This is good when counter-compare values are constantly changing like in PWM.
    EPwm3Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW; // see line above. This is for the
counter value B.
    /// Determine when Counter-Compare Module is loaded
    EPwm3Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // This loads the values for
register A at counter = 0.
    EPwm3Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // This loads the values for
register B at counter = 0.
    /// Determines PWM action for counter = counter-compare
    EPwm3Regs.AQCTLA.bit.CAU = AQ_SET; // toggles output for up-count
    EPwm3Regs.AQCTLA.bit.CAD = AQ_CLEAR; // does nothing for down-count
    // only A is set because B is complement unless specified otherwise in POLSEL
register. Need CAU and CAD because of UPDOWN count.
    /// Enable the Dead-Band Module
    EPwm3Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE; // enables output dead-band for
EPWM1A and EPWM1B. Necessary for avoiding shoot-through current.
    /// Determine the relationship between A and B
    EPwm3Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC; //Active High Complementary Mode. B
is opposite of A.
    /// Determine the Dead-Band length
    EPwm3Regs.DBRED = deadtimecycles; // Number of clock cycles for rise. (50 *
6.666667ns)
    EPwm3Regs.DBFED = deadtimecycles; // NUmber of clock cycles for fall.

    EPwm3Regs.ETSEL.bit.INTSEL = ET_CTR_ZERO;     // Select INT on Zero event
    EPwm3Regs.ETSEL.bit.INTEN = 0;  // Enable INT
    EPwm3Regs.ETPS.bit.INTPRD = ET_1ST;           // Generate INT on 1st event,
can also do 2nd and 3rd

    EALLOW;
    // Trip Zone 1 used for ePwm3
    EPwm3Regs.TZSEL.bit.OSHT1 = 1;
    EPwm3Regs.TZSEL.bit.CBC1 = 1;
    // What do we want the TZ1 to do?
    EPwm3Regs.TZCTL.bit.TZA = TZ_FORCE_LO;
    EPwm3Regs.TZCTL.bit.TZB = TZ_FORCE_LO;
    // Enable TZ interrupt
    EPwm3Regs.TZEINT.bit.OST = 1;
    EPwm3Regs.TZEINT.bit.CBC = 1;
    EDIS;

}

void InitEPwm4Timer()
{
    EALLOW;
    SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0;      // Stop all the TB clocks
    EDIS;

    /// Initialize the Ports (GPIO Pins)
    InitEPwm4Gpio(); // enables pull-ups on ePWM1's output pins and select ePWM1's
output pins
    /// Determine the Clock Prescalar
    EPwm4Regs.TBCTL.bit.HSPCLKDIV = 0; // Corresponds to a scalar of 1. This sets
the clock to be 150 MHz. (T = 6.666667 ns)
    /// Determine the Timer Clock Prescalar
    EPwm4Regs.TBCTL.bit.CLKDIV = 0; // Corresponds to a scalar of of 1. This keeps
the clock at 150 Mhz.
    /// Period of Timer
    EPwm4Regs.TBPRD = PWM1_TIMER_TBPRD;
    /// Phase Shift-
    EPwm4Regs.TBPHS.half.TBPHS = 0; // Upon phase syncing. The counter resets to
this value.
    // This is also the starting phase shift of the module. If it is ignoring
phase synching then this just the starting point.
    /// Counting Mode of Timer
    EPwm4Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;    // Count up to TBPRD and the
immediately count back down to 0. Allows for symmetrical PWM signal.
    /// Allow each timer to be sync'ed
    EPwm4Regs.TBCTL.bit.PHSEN = TB_ENABLE; // This bit determines whether or not
the ePWM module acknowledges the phase synch. Right now it's a 'Slave Module'.
    /// Determine the use of Shadow Period Load Register
    EPwm4Regs.TBCTL.bit.PRDLD = TB_SHADOW; // Loads the TBPRD register when the
timer counts to 0. This is good for changing periods, which is not the case here.
    /// Setup Sync
    EPwm4Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN; // This determines how the module
deals with phase synching. This takes in a sync signal from upstream.
    /// Determine use of Shadow Counter-Compare Registers (A and B)
    EPwm4Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW; // This uses the shadow register.
This is good when counter-compare values are constantly changing like in PWM.
    EPwm4Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW; // see line above. This is for the
counter value B.
    /// Determine when Counter-Compare Module is loaded
    EPwm4Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // This loads the values for
register A at counter = 0.
    EPwm4Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // This loads the values for
register B at counter = 0.
    /// Determines PWM action for counter = counter-compare
    EPwm4Regs.AQCTLA.bit.CAU = AQ_SET; // toggles output for up-count
    EPwm4Regs.AQCTLA.bit.CAD = AQ_CLEAR; // does nothing for down-count
    // only A is set because B is complement unless specified otherwise in POLSEL
register. Need CAU and CAD because of UPDOWN count.
    /// Enable the Dead-Band Module
    EPwm4Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE; // enables output dead-band for
EPWM1A and EPWM1B. Necessary for avoiding shoot-through current.
    /// Determine the relationship between A and B
    EPwm4Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC; //Active High Complementary Mode. B
is opposite of A.
    /// Determine the Dead-Band length
    EPwm4Regs.DBRED = deadtimecycles; // Number of clock cycles for rise. (50 *
6.666667ns)
	EPwm4Regs.DBFED = deadtimecycles; // NUmber of clock cycles for fall.

	EPwm4Regs.ETSEL.bit.INTSEL = ET_CTR_ZERO;     // Select INT on Zero event
	EPwm4Regs.ETSEL.bit.INTEN = 0;  // Enable INT
	EPwm4Regs.ETPS.bit.INTPRD = ET_1ST;           // Generate INT on 1st event,
can also do 2nd and 3rd

	EALLOW;
	// Trip Zone 1 used for ePwm4
	EPwm4Regs.TZSEL.bit.OSHT1 = 1;
	EPwm4Regs.TZSEL.bit.CBC1 = 1;
	// What do we want the TZ1 to do?
	EPwm4Regs.TZCTL.bit.TZA = TZ_FORCE_LO;
	EPwm4Regs.TZCTL.bit.TZB = TZ_FORCE_LO;
	// Enable TZ interrupt
	EPwm4Regs.TZEINT.bit.OST = 1;
	EPwm4Regs.TZEINT.bit.CBC = 1;
	EDIS;

}

void InitEPwm5Timer()
{

	EALLOW;
	SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0;      // Stop all the TB clocks
	EDIS;

	/// Initialize the Ports (GPIO Pins)
	InitEPwm5Gpio(); // enables pull-ups on ePWM1's output pins and select ePWM1's
output pins
	/// Determine the Clock Prescalar
	EPwm5Regs.TBCTL.bit.HSPCLKDIV = 0; // Corresponds to a scalar of 1. This sets
the clock to be 150 MHz. (T = 6.666667 ns)
	/// Determine the Timer Clock Prescalar
	EPwm5Regs.TBCTL.bit.CLKDIV = 0; // Corresponds to a scalar of of 1. This keeps
the clock at 150 Mhz.
	/// Period of Timer
	EPwm5Regs.TBPRD = PWM1_TIMER_TBPRD;
	/// Phase Shift-
	EPwm5Regs.TBPHS.half.TBPHS = 0; // Upon phase syncing. The counter resets to
this value.
	// This is also the starting phase shift of the module. If it is ignoring
phase synching then this just the starting point.
	/// Counting Mode of Timer
	EPwm5Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;    // Count up to TBPRD and the
immediately count back down to 0. Allows for symmetrical PWM signal.
	/// Allow each timer to be sync'ed
	EPwm5Regs.TBCTL.bit.PHSEN = TB_ENABLE; // This bit determines whether or not
the ePWM module acknowledges the phase synch. Right now it's a 'Slave Module'.
	/// Determine the use of Shadow Period Load Register
	EPwm5Regs.TBCTL.bit.PRDLD = TB_SHADOW; // Loads the TBPRD register when the
timer counts to 0. This is good for changing periods, which is not the case here.
	/// Setup Sync
```

```
        EPwm5Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN; // This determines how the module
deals with phase synching. This takes in a sync signal from upstream.
        /// Determine use of Shadow Counter-Compare Registers (A and B)
        EPwm5Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW; // This uses the shadow register.
This is good when counter-compare values are constantly changing like in PWM.
        EPwm5Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW; // see line above. This is for the
counter value B.
        /// Determine when Counter-Compare Module is loaded
        EPwm5Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // This loads the values for
register A at counter = 0.
        EPwm5Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // This loads the values for
register B at counter = 0.
        /// Determines PWM action for counter = counter-compare
        EPwm5Regs.AQCTLA.bit.CAU = AQ_SET; // toggles output for up-count
        EPwm5Regs.AQCTLA.bit.CAD = AQ_CLEAR; // does nothing for down-count
        // only A is set because B is complement unless specified otherwise in POLSEL
register. Need CAU and CAD because of UPDOWN count.
        /// Enable the Dead-Band Module
        EPwm5Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE; // enables output dead-band for
EPWM1A and EPWM1B. Necessary for avoiding shoot-through current.
        /// Determine the relationship between A and B
        EPwm5Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC; //Active High Complementary Mode. B
is opposite of A.
        /// Determine the Dead-Band length
        EPwm5Regs.DBRED = deadtimecycles; // Number of clock cycles for rise. (50 *
6.666667ns)
        EPwm5Regs.DBFED = deadtimecycles; // NUmber of clock cycles for fall.

        EPwm5Regs.ETSEL.bit.INTSEL = ET_CTR_ZERO;     // Select INT on Zero event
        EPwm5Regs.ETSEL.bit.INTEN = 0;  // Enable INT
        EPwm5Regs.ETPS.bit.INTPRD = ET_1ST;           // Generate INT on 1st event,
can also do 2nd and 3rd

        EALLOW;
        // Trip Zone 1 used for ePwm5
        EPwm5Regs.TZSEL.bit.OSHT1 = 1;
        EPwm5Regs.TZSEL.bit.CBC1 = 1;
        // What do we want the TZ1 to do?
        EPwm5Regs.TZCTL.bit.TZA = TZ_FORCE_LO;
        EPwm5Regs.TZCTL.bit.TZB = TZ_FORCE_LO;
        // Enable TZ interrupt
        EPwm5Regs.TZEINT.bit.OST = 1;
        EPwm5Regs.TZEINT.bit.CBC = 1;
        EDIS;

}

void InitEPwm6Timer()
{

        EALLOW;
        SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0;      // Stop all the TB clocks
        EDIS;

        /// Initialize the Ports (GPIO Pins)
        InitEPwm6Gpio(); // enables pull-ups on ePWM1's output pins and select ePWM1's
output pins
        /// Determine the Clock Prescalar
        EPwm6Regs.TBCTL.bit.HSPCLKDIV = 0; // Corresponds to a scalar of 1. This sets
the clock to be 150 MHz. (T = 6.666667 ns)
        /// Determine the Timer Clock Prescalar
        EPwm6Regs.TBCTL.bit.CLKDIV = 0; // Corresponds to a scalar of of 1. This keeps
the clock at 150 Mhz.
        /// Period of Timer
        EPwm6Regs.TBPRD = PWM1_TIMER_TBPRD;
        /// Phase Shift-
        EPwm6Regs.TBPHS.half.TBPHS = 0; // Upon phase syncing. The counter resets to
this value.
        // This is also the starting phase shift of the module. If it is ignoring
phase synching then this just the starting point.
        /// Counting Mode of Timer
        EPwm6Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;    // Count up to TBPRD and the
immediately count back down to 0. Allows for symmetrical PWM signal.
        /// Allow each timer to be sync'ed
        EPwm6Regs.TBCTL.bit.PHSEN = TB_ENABLE; // This bit determines whether or not
the ePWM module acknowledges the phase synch. Right now it's a 'Slave Module'.
        /// Determine the use of Shadow Period Load Register
        EPwm6Regs.TBCTL.bit.PRDLD = TB_SHADOW; // Loads the TBPRD register when the
timer counts to 0. This is good for changing periods, which is not the case here.
        /// Setup Sync
        EPwm6Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN; // This determines how the module
deals with phase synching. This takes in a sync signal from upstream.
        /// Determine use of Shadow Counter-Compare Registers (A and B)
        EPwm6Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW; // This uses the shadow register.
This is good when counter-compare values are constantly changing like in PWM.
        EPwm6Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW; // see line above. This is for the
counter value B.
        /// Determine when Counter-Compare Module is loaded
        EPwm6Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // This loads the values for
register A at counter = 0.
        EPwm6Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // This loads the values for
register B at counter = 0.
        /// Determines PWM action for counter = counter-compare
        EPwm6Regs.AQCTLA.bit.CAU = AQ_SET; // toggles output for up-count
        EPwm6Regs.AQCTLA.bit.CAD = AQ_CLEAR; // does nothing for down-count
        // only A is set because B is complement unless specified otherwise in POLSEL
register. Need CAU and CAD because of UPDOWN count.
        /// Enable the Dead-Band Module
        EPwm6Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE; // enables output dead-band for
EPWM1A and EPWM1B. Necessary for avoiding shoot-through current.
        /// Determine the relationship between A and B
        EPwm6Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC; //Active High Complementary Mode. B
is opposite of A.
        /// Determine the Dead-Band length
        EPwm6Regs.DBRED = deadtimecycles; // Number of clock cycles for rise. (50 *
6.666667ns)
        EPwm6Regs.DBFED = deadtimecycles; // NUmber of clock cycles for fall.

        EPwm6Regs.ETSEL.bit.INTSEL = ET_CTR_ZERO;     // Select INT on Zero event
        EPwm6Regs.ETSEL.bit.INTEN = 0;  // Enable INT
```

```
       EPwm6Regs.ETPS.bit.INTPRD = ET_1ST;          // Generate INT on 1st event, 
can also do 2nd and 3rd

       EALLOW;
       // Trip Zone 1 used for ePwm6
       EPwm6Regs.TZSEL.bit.OSHT1 = 1;
       EPwm6Regs.TZSEL.bit.CBC1 = 1;
       // What do we want the TZ1 to do?
       EPwm6Regs.TZCTL.bit.TZA = TZ_FORCE_LO;
       EPwm6Regs.TZCTL.bit.TZB = TZ_FORCE_LO;
       // Enable TZ interrupt
       EPwm6Regs.TZEINT.bit.OST = 1;
       EPwm6Regs.TZEINT.bit.CBC = 1;
       EDIS;

}


interrupt void cpu_timer0_isr(void)
{

       ccbit = 1; // Turn on this bit to allow the current controller to be run

//     CurrentControl();

       CpuTimer0Regs.TCR.bit.TIF = 1;
       CpuTimer0Regs.TCR.bit.TSS = 0;
       CpuTimer0Regs.TCR.bit.TRB = 1;

       PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;// Acknowledge this interrupt to 
receive more interrupts from group 1

}

void ReferenceGenerator()
{

       float average, vbus_err;
       static float Vref = 10.0;
       static float k = 0.0, m = 0.0;

       average = (PARKS_Ia_L.u_D[0] + PARKS_Ib_L.u_D[0] + PARKS_Ic_L.u_D[0]) / 3.0;
       vbus_err = (VbusNotch.y[0] - Vref) * 0.0707;

       if(!GpioDataRegs.GPADAT.bit.GPIO31) // go into reactive-power-compensation/dc-
bus mode
       {
              if(!GpioDataRegs.GPBDAT.bit.GPIO34) // go into current 
balancing/reactive-power-compensation/dc-bus mode
              {
                     if (m < 1.0)
                     {
                            m += 0.00001;
                     }
```

```
            Curr_Cont_Refa.err = vbus_err - (PARKS_Ia_L.u_D[0] -
PARKS_Ia_inv.u_D[0] - average) * m;
            Curr_Cont_Refb.err = vbus_err - (PARKS_Ib_L.u_D[0] -
PARKS_Ib_inv.u_D[0] - average) * m;
            Curr_Cont_Refc.err = vbus_err - (PARKS_Ic_L.u_D[0] -
PARKS_Ic_inv.u_D[0] - average) * m;
        }
        else // stay in reactive-power-compensation/dc-bus mode
        {
            Curr_Cont_Refa.err = vbus_err;
            Curr_Cont_Refb.err = vbus_err;
            Curr_Cont_Refc.err = vbus_err;
        }

        if (k < 1.0)
        {
            k += 0.00001;
        }

        Curr_Cont_Qa.err = PARKS_Ia_L.u_Q[0] * k - PARKS_Ia_inv.u_Q[0]; //
update the q-axis current PI controller
        Curr_Cont_Qb.err = PARKS_Ib_L.u_Q[0] * k - PARKS_Ib_inv.u_Q[0]; //
update the q-axis current PI controller
        Curr_Cont_Qc.err = PARKS_Ic_L.u_Q[0] * k - PARKS_Ic_inv.u_Q[0]; //
update the q-axis current PI controller
    }
    else // This is the starting mode, just maintains the dc-bus voltage
    {

        Curr_Cont_Refa.err = vbus_err;
        Curr_Cont_Refb.err = vbus_err;
        Curr_Cont_Refc.err = vbus_err;
        Curr_Cont_Qa.err = 0.0 - PARKS_Ia_inv.u_Q[0];
        Curr_Cont_Qb.err = 0.0 - PARKS_Ib_inv.u_Q[0];
        Curr_Cont_Qc.err = 0.0 - PARKS_Ic_inv.u_Q[0];
    }

    PI_controller_FUNC(&Curr_Cont_Refa); // run the reference generator PI
controller
    PI_controller_FUNC(&Curr_Cont_Refb); // run the reference generator PI
controller
    PI_controller_FUNC(&Curr_Cont_Refc); // run the reference generator PI
controller

    if(Vref < dcbusref) // Soft start the first inverter
    {
        Vref += 0.0005;
//      Curr_Cont_Da.Ki = 0.1;
        if(Curr_Cont_Refa.U < -0.25){Curr_Cont_Refa.U = -0.25;
Curr_Cont_Refa.U_1 = -0.25;}
        if(Curr_Cont_Refb.U < -0.25){Curr_Cont_Refb.U = -0.25;
Curr_Cont_Refb.U_1 = -0.25;}
        if(Curr_Cont_Refc.U < -0.25){Curr_Cont_Refc.U = -0.25;
Curr_Cont_Refc.U_1 = -0.25;}
    }
```

```
}

void CurrentControl()
{

	if (ccbit == 1) // This bit switched in PWM interrupt to time the current
controller
	{

		float norm_denom;
//		static float DaxisMag_a_prev = 0.0, QaxisMag_a_prev = 0.0,
DaxisMag_b_prev = 0.0, QaxisMag_b_prev = 0.0, DaxisMag_c_prev = 0.0, QaxisMag_c_prev
= 0.0;
//		static float Iainv_prev = 0.0, Ibinv_prev = 0.0, Icinv_prev = 0.0,
Vbus_prev = 9.0;

		if(j < 10000)
		{
			j++;
			ClearTrip = 0;
		}
		else{ClearTrip = 1;}

		//---------- ADC Sample Collection and Conditioning ------------//

		ADC_Read(&IainvADC, AdcMirror.ADCRESULT0);
		ADC_Read(&IbinvADC, AdcMirror.ADCRESULT4);
		ADC_Read(&IcinvADC, AdcMirror.ADCRESULT7);

		ADC_Read(&IaLADC, AdcMirror.ADCRESULT1);
		ADC_Read(&IbLADC, AdcMirror.ADCRESULT5);
		ADC_Read(&IcLADC, AdcMirror.ADCRESULT8);

		ADC_Read(&VaADC, AdcMirror.ADCRESULT2);
		ADC_Read(&VbADC, AdcMirror.ADCRESULT6);
		ADC_Read(&VcADC, AdcMirror.ADCRESULT9);

		ADC_Read(&VbusADC, AdcMirror.ADCRESULT3);
		VbusADC.out *= 225; // 175 = 350Vmax/2
		Notch_Filter_Func(&VbusNotch, VbusADC.out);

		//----------------- PLL and Magnitude -------------------//

		VaPLL.u[0] = VaADC.out;
		SPLL_1ph_SOGI_F_FUNC(&VaPLL);

		VbPLL.u[0] = VbADC.out;
		SPLL_1ph_SOGI_F_FUNC(&VbPLL);

		VcPLL.u[0] = VcADC.out;
		SPLL_1ph_SOGI_F_FUNC(&VcPLL);

		//-----------   Park's Transformations   ----------------//
```

```
        PARKS_Ia_inv.u[0] = IainvADC.out;
        SPLL_1ph_SOGI_F_FUNC_SOGIPARK(&PARKS_Ia_inv, VaPLL.sin, VaPLL.cos); //
Outputs: -PARKS_Ia_inv.u_D[0], PARKS_Ia_inv.u_Q[0], for some reason D-axis is neg;

        PARKS_Ia_L.u[0] = IaLADC.out;
        SPLL_1ph_SOGI_F_FUNC_SOGIPARK(&PARKS_Ia_L, VaPLL.sin, VaPLL.cos);

        PARKS_Ib_inv.u[0] = IbinvADC.out;
        SPLL_1ph_SOGI_F_FUNC_SOGIPARK(&PARKS_Ib_inv, VbPLL.sin, VbPLL.cos);

        PARKS_Ib_L.u[0] = IbLADC.out;
        SPLL_1ph_SOGI_F_FUNC_SOGIPARK(&PARKS_Ib_L, VbPLL.sin, VbPLL.cos);

        PARKS_Ic_inv.u[0] = IcinvADC.out;
        SPLL_1ph_SOGI_F_FUNC_SOGIPARK(&PARKS_Ic_inv, VcPLL.sin, VcPLL.cos); //
Outputs: -PARKS_Ia_inv.u_D[0], PARKS_Ia_inv.u_Q[0], for some reason D-axis is neg

        PARKS_Ic_L.u[0] = IcLADC.out;
        SPLL_1ph_SOGI_F_FUNC_SOGIPARK(&PARKS_Ic_L, VcPLL.sin, VcPLL.cos);

        //////////////////////////////////////////////////////////
        //---------------- Current Controller -------------------//
        //////////////////////////////////////////////////////////


        if(RunControl == 1)
        {

            ReferenceGenerator();

            Curr_Cont_Da.err = Curr_Cont_Refa.U + PARKS_Ia_inv.u_D[0]; //
update the d-axis current PI controller
            PI_controller_FUNC(&Curr_Cont_Da); // run the d- and q- axis
current PI controllers
            PI_controller_FUNC(&Curr_Cont_Qa);

            Curr_Cont_Db.err = Curr_Cont_Refb.U + PARKS_Ib_inv.u_D[0]; //
update the d-axis current PI controller
            PI_controller_FUNC(&Curr_Cont_Db); // run the d- and q- axis
current PI controllers
            PI_controller_FUNC(&Curr_Cont_Qb);

            Curr_Cont_Dc.err = Curr_Cont_Refc.U + PARKS_Ic_inv.u_D[0]; //
update the d-axis current PI controller
            PI_controller_FUNC(&Curr_Cont_Dc); // run the d- and q- axis
current PI controllers
            PI_controller_FUNC(&Curr_Cont_Qc);

            // Determine the d- and q- axis component of the voltage output

            DaxisMag_a = Curr_Cont_Da.U - ( omegaLcoup * PARKS_Ia_inv.u_Q[0]
) + VaPLL.u_D[0] * VoltagePeakMax;
            QaxisMag_a = Curr_Cont_Qa.U - ( omegaLcoup * PARKS_Ia_inv.u_D[0]
);
```

```
                    DaxisMag_b = Curr_Cont_Db.U - ( omegaLcoup * PARKS_Ib_inv.u_Q[0]
) + VbPLL.u_D[0] * VoltagePeakMax;
                    QaxisMag_b = Curr_Cont_Qb.U - ( omegaLcoup * PARKS_Ib_inv.u_D[0]
);

                    DaxisMag_c = Curr_Cont_Dc.U - ( omegaLcoup * PARKS_Ic_inv.u_Q[0]
) + VcPLL.u_D[0] * VoltagePeakMax;
                    QaxisMag_c = Curr_Cont_Qc.U - ( omegaLcoup * PARKS_Ic_inv.u_D[0]
);

                    //----- Find control waveform function and its mag ------//

                    //|Asin(x) + Bcos(x)| = sqrt( A^2 + B^2 )
                    gain_a = sqrt(DaxisMag_a * DaxisMag_a + QaxisMag_a * QaxisMag_a);
                    alpha_a = DaxisMag_a * VaPLL.sin + QaxisMag_a * VaPLL.cos;

                    gain_b = sqrt(DaxisMag_b * DaxisMag_b + QaxisMag_b * QaxisMag_b);
                    alpha_b = DaxisMag_b * VbPLL.sin + QaxisMag_b * VbPLL.cos;

                    gain_c = sqrt(DaxisMag_c * DaxisMag_c + QaxisMag_c * QaxisMag_c);
                    alpha_c = DaxisMag_c * VcPLL.sin + QaxisMag_c * VcPLL.cos;

                    //---------- Normalize the control waveform -------------//

                    norm_denom = 2.0*VbusADC.out*VbusADC.out;

                    control_a = gain_a / (norm_denom) * alpha_a + 0.5;
                    control_b = gain_b / (norm_denom) * alpha_b + 0.5;
                    control_c = gain_c / (norm_denom) * alpha_c + 0.5;


                    ia = IainvADC.out;
                    ib = IbinvADC.out;
                    ic = IcinvADC.out;

                    if(ia > 0.0){PhALegA = control_a + 0.0; PhALegB = control_a -
0.0;}
                    if(ia < 0.0){PhALegA = control_a - 0.0; PhALegB = control_a +
0.0;}
                    if(ib > 0.0){PhBLegA = control_b + 0.0; PhBLegB = control_b -
0.0;}
                    if(ib < 0.0){PhBLegA = control_b - 0.0; PhBLegB = control_b +
0.0;}
                    if(ic > 0.0){PhCLegA = control_c + 0.0; PhCLegB = control_c -
0.0;}
                    if(ic < 0.0){PhCLegA = control_c - 0.0; PhCLegB = control_c +
0.0;}


                    //-----------------Update the PWM Module -------------------//

                    EPwm1Regs.CMPA.half.CMPA = PWM1_TIMER_TBPRD * ( 1.0 - PhALegA );
// Counter Compare value updated for Phase A, Leg 1
                    EPwm2Regs.CMPA.half.CMPA = PWM1_TIMER_TBPRD * ( PhALegB ); //
Counter Compare value updated for Phase A, Leg 2
```

```
                    EPwm3Regs.CMPA.half.CMPA = PWM1_TIMER_TBPRD * ( 1.0 - PhBLegA );
// Counter Compare value updated for Phase B, Leg 1
                    EPwm4Regs.CMPA.half.CMPA = PWM1_TIMER_TBPRD * ( PhBLegB ); //
Counter Compare value updated for Phase B, Leg 2

                    EPwm5Regs.CMPA.half.CMPA = PWM1_TIMER_TBPRD * ( 1.0 - PhCLegA );
// Counter Compare value updated for Phase C, Leg 1
                    EPwm6Regs.CMPA.half.CMPA = PWM1_TIMER_TBPRD * ( PhCLegB ); //
Counter Compare value updated for Phase C, Leg 2

             } // end bracket for controller ON/OFF flag

             ccbit = 0; // turn off this bit till the next CPUTIMER0 interrupt, so
that the controller doesn't continuously run.
       } // ccbit if statement end bracket

}

void OverCurrentProtection()
{

       if(ocdelay == 2000)
       {

             if (AdcMirror.ADCRESULT0 > 3789 || AdcMirror.ADCRESULT0 < 307) //---
overcurrent trip protection, turns off inverter if d-axis current is too big
             {
                    EALLOW;
                    EPwm1Regs.TZFRC.bit.OST = 1;
                    EPwm2Regs.TZFRC.bit.OST = 1;
                    EDIS;
                    EnablePWM = 0;

             }
             else
             {
                    if(ClearTrip == 1 && VaPLL.ZCD == 1 && EnablePWM == 1)
                    {
                          RunControl = 1;
                          EALLOW;
                          EPwm1Regs.TZCLR.bit.OST = 1;
                          EPwm2Regs.TZCLR.bit.OST = 1;
                          EDIS;
                    }
             }

             if (AdcMirror.ADCRESULT4 > 3789 || AdcMirror.ADCRESULT4 < 307) //---
overcurrent trip protection, turns off inverter if d-axis current is too big
             {
                    EALLOW;
                    EPwm3Regs.TZFRC.bit.OST = 1;
                    EPwm4Regs.TZFRC.bit.OST = 1;
                    EDIS;
                    EnablePWM = 0;
```

```
        }
        else
        {
            if(ClearTrip == 1 && VbPLL.ZCD == 1 && EnablePWM == 1)
            {
                RunControl = 1;
                EALLOW;
                EPwm3Regs.TZCLR.bit.OST = 1;
                EPwm4Regs.TZCLR.bit.OST = 1;
                EDIS;
            }
        }

        if (AdcMirror.ADCRESULT7 > 3789 || AdcMirror.ADCRESULT7 < 307) //---
overcurrent trip protection, turns off inverter if d-axis current is too big
        {
            EALLOW;
            EPwm5Regs.TZFRC.bit.OST = 1;
            EPwm6Regs.TZFRC.bit.OST = 1;
            EDIS;
            EnablePWM = 0;
        }
        else
        {
            if(ClearTrip == 1 && VcPLL.ZCD == 1 && EnablePWM == 1)
            {
                RunControl = 1;
                EALLOW;
                EPwm5Regs.TZCLR.bit.OST = 1;
                EPwm6Regs.TZCLR.bit.OST = 1;
                EDIS;
            }
        }

    } // oc end bracket
    else{ocdelay++;}

    if(!GpioDataRegs.GPADAT.bit.GPIO30)
    {
        EALLOW;
        EPwm1Regs.TZFRC.bit.OST = 1;
        EPwm2Regs.TZFRC.bit.OST = 1;
        EPwm3Regs.TZFRC.bit.OST = 1;
        EPwm4Regs.TZFRC.bit.OST = 1;
        EPwm5Regs.TZFRC.bit.OST = 1;
        EPwm6Regs.TZFRC.bit.OST = 1;
        EDIS;
        RunControl = 0;
        ClearTrip = 0;
        EnablePWM = 0;
    }

}

//============================================================================
// No more.
//============================================================================
```