UNIVERSITY OF CALIFORNIA SANTA CRUZ

SENIOR DESIGN FINAL REPORT

Hardware Validation of a Robust Hybrid Power Inverter

Author:
Ryan RODRIGUEZ,
Benjamin CHAINEY

Supervisor:
Dr. Pat MANTEY,
Dr. Ricardo SANFELICE,
Paul NAUD,
Jun CHAI

A technical report submitted in fulfilment of the requirements for the degree of Bachelor of Science in Electrical Engineering

to

CITRIS, and the Hybrid Systems Lab
Department of Electrical Engineering

May 2015

Declaration of Authorship

We, Ryan Rodriguez and Ben Chainey, declare that this paper titled, 'Hardware Validation of a Robust Hybrid Power Inverter' and the work presented in it are our own. I confirm that:

- This work was done wholly or mainly while in candidature for a bachelors degree at this University.
- Where we have consulted the published work of others, this is always clearly attributed.
- Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this report is entirely our own work.
- We have acknowledged all main sources of help.
- Where the report is based on work done by ourselves jointly with others, we have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Signed: _____

Date: _____

"Don't think, Feel. It is like a finger pointing out to the moon - don't concentrate on the finger, or you will miss all that heavenly glory."

Bruce Lee

UNIVERSITY OF CALIFORNIA SANTA CRUZ

Abstract

Dr. Pat Mantey, Dr. Ricardo Sanfelice
Department of Electrical Engineering

Bachelor of Science in Electrical Engineering

Hardware Validation of a Robust Hybrid Power Inverter

by Ryan RODRIGUEZ,
Benjamin CHAINEY

The need for clean and renewable energy sources - coupled with the rapid growth in commercial and residential solar microgrid installations - has driven the development of a new hybrid algorithm for power conversion.

In a departure from traditional pulse-width modulation, or PWM, techniques, the new hybrid controller leverages a reference solution derived from the physical model of our circuit to determine a tracking band, or neighborhood, around the desired output sinusoid. Numerical results have shown that this technique results in a spectral content that is 'cleaner' than its PWM counterpart.[1]

In this design we utilize the canonical power inverter topology consisting of an H-Bridge followed by an RLC low-pass filter. We aim to confirm the computational results of this research with an implementation on a full microinverter system.

Acknowledgements

The Hybrid Inverter Team would like to acknowledge the generous support of CITRIS, the Center for Information Technology Research in the Interest of Society, and the Hybrid Systems Lab for their financial and intellectual support. This work would not have been possible without their efforts. We would also like to personally thank Dr. Pat Mantey for his trust in our abilities, and Dr. Ricardo Sanfelice for extending this project to the senior design course here at UC Santa Cruz.

This acknowledgement would be far from complete if we didn't speak to the tireless efforts of the rest of the SDP cadre, and in particular Paul Naud and Jun Chai for imparting their wisdom when we needed it most.

Ben would like to thank ...

Ryan Rodriguez' personal acknowledgement:
I would like to thank both the departments of electrical and computer engineering at UC Santa Cruz for offering me one of the most formidable challenges of my life, and the faculty of these departments who have pushed me to reach farther than I thought I could.

To my peers in the Jack Baskin School of Engineering I extend my best wishes, and my appreciation for your friendship and support - we did it! A special acknowledgement to Ben Chainey for taking this ride with me on the crowning achievement of our undergraduate career. It has been a pleasure.

I would like to extend my deepest thanks to my family for offering me their unconditional support and encouragement. In particular I'd like to thank my mom and my brother for helping me to become the man that I am. I owe you both everything.

Finally, I'd like to thank my best friend and companion, Lisa Carmack. Thank you for letting me into your world.

Contents

List of Figures

List of Tables

Abbreviations

DSP	**D**igital **S**ignal **P**rocessor
LAH	**L**ist **A**bbreviations **H**ere
ISR	**I**nterrupt **S**ervice **R**outine
PV	**P**hoto **V**oltaic
THD	**T**otal **H**armonic **D**istortion
RMS	**R**oot **M**ean **S**quared
μ **C**	**M**icro **C**ontroller

Symbols

P	Power	W (Js^{-1})
V	Voltage	V (JC^{-1})
I	Current	A (Cs^{-1})
ω	angular frequency	$rads^{-1}$

Strictly for my niggas…

Chapter 1

The Problems with Solar

1.1 Motivation

The increasing demand for energy, combined with the surging interest in green technologies worldwide, has accelerated the need for a highly reliable, cost-efficient, and self-sustained electric power grid. Scientists and engineers everywhere agree without a shadow of a doubt that human power generation is the primary driver of the climate change phenomena wreaking havoc on our planet. Future energy distribution systems should be capable of interconnecting diverse power sources including fossil and nuclear-fueled generators, as well as renewable sources such as hydropower generators, wind turbines, and photovoltaic arrays.

Renewable energy sources depend heavily on highly variable environmental conditions, and hence, require robust power conversion techniques that can handle unstable and highly varying input power[2]. The focus of this research-oriented senior design project has been to explore the viability and technical challenges involved with implementing new hybrid control techniques for solar power conversion. This research is of great interest to us, and the public in general, as it may prove to be a valuable addition to the power system designers toolkit for situations necessitating exceptionally clean output with robust stability characteristics in the face of highly variable load and source conditions.

We seek to understand the reported benefits of hybrid control techniques in the application of solar power converters. In particular, the conversion between the non-linear DC output of solar panels to steady AC power used in the power grid will be studied. In order to test the techniques which have been described analytically in [2], the Hybrid Inverter Team, in partial fulfillment of the requirements for the Bachelors of Science in

Electrical Engineering, will develop a small-scale power inverter capable of switching between pulse-width modulation and hybrid inverter algorithms. This development platform will allow for the straightforward comparison between the two. This report is intended to be an exhaustive account of our methodology and research in the pursuit of reaching these goals.

1.2 Background

In order to meet the challenge of building next-generation power conversion technologies capable of handling highly variable sources while simultaneously mitigating the problem of noise injection into the power grid, we must first understand the current landscape of power converters, analyze their strengths and weaknesses, and then assess how we might improve upon their implementations by leveraging today's inexpensive real-time microcontrollers, which have been tuned for power today's demanding power conversion applications. To this end, we will undertake a brief overview of the pulse-width modulation technique, the applicability of hybrid control algorithms to the problem of power conversion, and finally, we will briefly review the additional constraints that power conversion in the context of solar energy places on our design.

1.2.1 The Problems with Solar

With the cost of photovoltaics rapidly decreasing, we have seen a rush toward the adoption of solar micro-grids which seek to exploit the most abundant source of power known to mankind - the sun. However, our heliocentric conundrum - namely the fact that the earth orbits the sun - dictates that most places on earth receive time-dependent quantities of solar irradiance over the course of any given day. This high variability in the context of power conversion implies major challenges to our modern power grid which guarantees nearly continuous up-time. Additionally, today's power conversion technologies are not robust to highly variable input sources like solar power. Some other challenges we face when converting DC solar power to AC power which is usable in our power grid is the problem of shading or partial irradiance of solar arrays, and the non-linear nature of the photovoltaics themselves. In particular, this non-linear behavior necessitates the implementation of maximum power point tracking or MPPT algorithms which comprehend this phenomena and harvest the most power from solar sources which typically have a finite lifespan. This is critical for obtaining the maximum benefit from the non-trivial investment required to operate solar arrays today.

1.2.2 Square and Modified Square Inverters

Square wave inverters are the simples method for generating pseudo-sinusoidal outputs using an H-Bridge. Emulating the positive and negative half cycles of the sine with a simple bipolar scheme where the output is given be $+V_{dc}$ and $-V_{dc}$ respectively. After a filtering stage, the output can be shaped into something resembling a sine wave by only allowing the low frequency components of the square wave to pass. Note that the difference between a pure square wave and a modified square wave is the allowance of the zero state determined by the quantity α shown in Figure 1.1. Note that the Fourier series of the output is given by

$$v_0(t) = \sum_{n,odd} V_n \sin n\omega_0 t \tag{1.1}$$

where

$$V_n = \frac{4V_d c}{n\pi} cosn\alpha \tag{1.2}$$

This would add considerably to the complexity of any inverter system seeking to control both parameters simultaneously. The harmonic content can be controlled by varying the quantity α, and the amplitude can also be controlled by α. With some analysis, it could be shown that the harmonic content or the amlpitude of the output can be controlled, but not both simultaneously unless we have a vairbale V_{dc} input.

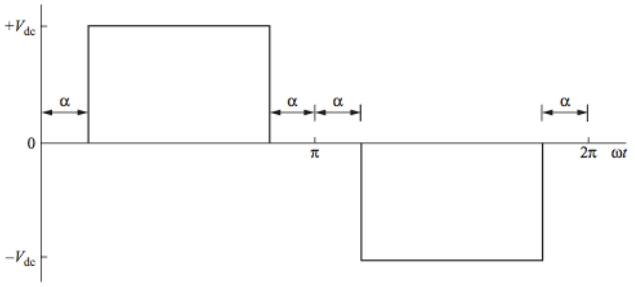

FIGURE 1.1: Modified Square Wave

The 'deal-breaker' with the square wave inverter is the fact the the spectral content of the output is particularly rich at all odd harmonics of the fundamental. This means that we are left with a bulk of the harmonic content at low frequencies which are particularly hard to filter with analog components.

1.2.3 Linear Control and the PWM approach

Compared to square and modified square wave inverters, much of the filtering burden is alleviated due to the fact that spectral content in the vicinity of the fundamental is

located near the frequency modulation index of the PWM signal. Pulse-width modulation (PWM), or bang-bang techniques, are a popular means for taking a fixed input voltage and varying the effective power seen at the output. This is done by rapidly switching the input off and on such that the the output is some fraction of the input including zero and one hundred percent. This fraction corresponds to the duty cycle. If we pair the PWM technique with the ability to drive current bidirectionally, say, through an H-Bridge circuit, we can use this technique to trace out a pseudo-sinusoidal output.

In contrast to the square wave inverters discussed above, with PWM it is possible to vary amplitude and frequency independently. The modulation index, which is given by $m_f = \frac{f_{tri}}{f_{sin}}$ where f_{tri} is the frequency of the triangular carrier frequency, and f_{sin} is the sinusoidal reference signal.

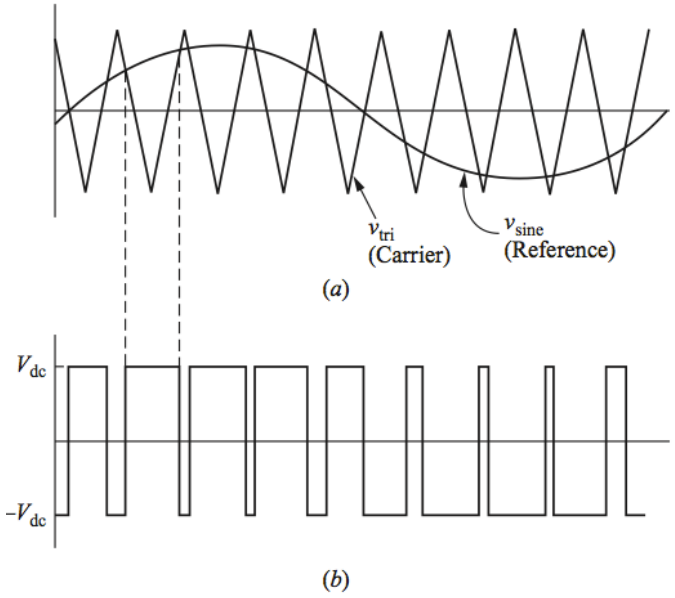

FIGURE 1.2: Bipolar Switching

The switch in this case can take the form of a relay, but more commonly in power applications, takes the form of a field-effect transistor (MOSFET) or insulated gate bipolar transistor (IGBT). The effective voltage or current is varied by varying the duration of on-time of the switch. One can think of this as int terms of a common household faucet with only two states, either fully on or off. If we were able to modulate the amount of time that the water flowed from the faucet, clearly we would be able to choose the flow-rate of the water from the two extrema - on or off - to an arbitrary level of precision depending on how fast we were able to turn the faucet on or off. The scenario put forth in the 'kitchen sink' analogy is analogous to the situation we face in a modern power inverter. In this case, we are faced with the challenge of taking a typically fixed

input voltage and switching it in such a way that we achieve a close approximation to a sinusoidal output voltage. Given the clear explanation of how PWM techniques can vary from fully on to fully off in the description above, it is easy to see why the PWM approach has become the standard for power inverters. Additionally, today's microcontrollers have extremely sophisticated peripheral modules built specifically for very fast and PWM signal generation with resolution down to the nanosecond level becoming quite common.

@todo:cite figures!

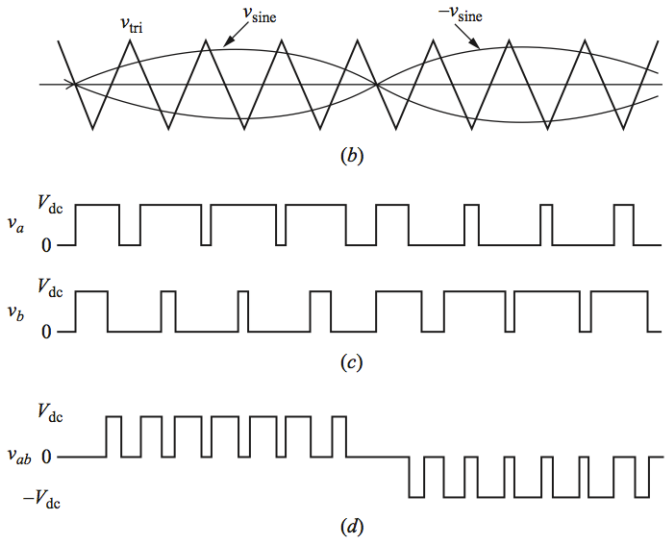

FIGURE 1.3: Unipolar Switching

The Fourier analysis for either bipolar or unipolar PWM output signals is not as straightforward as that of the square wave, so a full derivation is ommited from this discussion. However, the fact that the harmonic content occurs at multiples of the modulation frequency can be clearly observed in Figure 1.4.

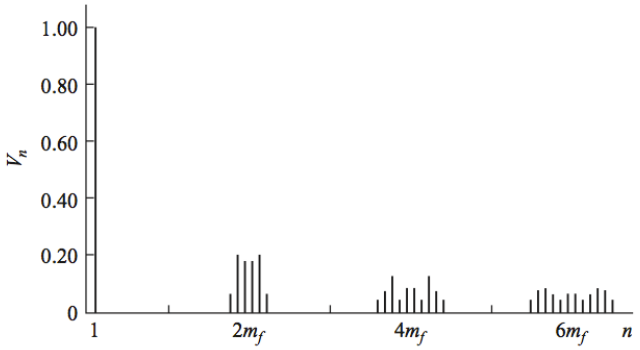

FIGURE 1.4: Fourier Spectrum Frequency Diagram for PWM Inverter Control

Although the PWM method for signal modulation has the clear advantages of widespread adoption and ease of implementation, with the widespread adoption of renewable energy in the form of highly local and decentralized micro-grids, we are increasingly faced with the problem of introducing increasing amounts of high-order noise into the power grid. Further, the typical proportional, integral, derivative, or PID methods for control suffer from a ringing phenomena in the presence of input disturbances which are typical of renewable energy applications. In light of these issues with the PWM technique for power conversion, we seek to explore alternate strategies for switching that might alleviate the vulnerability to input disturbances and the problem of an unintentionally rich spectrum at the output of PWM power converters.

1.2.4 Hybrid Systems and Hybrid Control

The study of hybrid systems has emerged in recent years as a response to the increasing entanglement of physical systems and computer control. Hybrid systems are those that exhibit both continuous and discrete-time dynamics. The system in question, namely the canonical power inverter topology, which is the focus of this research, is an excellent candidate for study under the lens of hybrid control due to the continuous evolution of a linear oscillator - our output filter - and the discrete time dynamics of the switch - in our case, a transistor receiving control signals from a micro.

We seek to understand how hybrid control might alleviate some of the challenges associated with PWM techniques, a few of which have been outlined above. While a detailed explanation of the hybrid control algorithm is saved for later on in this paper, in a nutshell it has been found that with relatively simple physical models of our system, and the availability of discrete time switching, that we can generate outputs that closely resemble the desired sinusoidal outputs with less switching noise, and with a robust response to highly variable input voltages. For these reasons, we focus this research on the physical realization of such a hybrid system. One of the goals of this paper is to describe the algorithm designed by Jun Chai and Dr. Ricardo Sanfelice in more detail, and outline some of the challenges associated with realizing this implementation on a real world system [2].

1.3 Approach

Switching power inverters are an enabling technology found in energy conversion systems. Photovoltaic (PV) panels harvest energy from the sun to generate the alternating

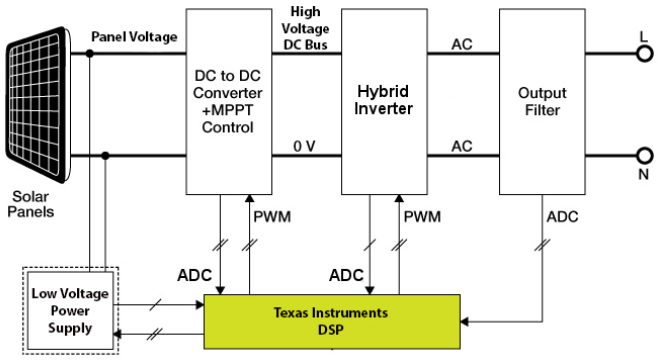

FIGURE 1.5: Inverter System Overview

current (AC) electricity used in the grid. PV arrays output direct current (DC) electricity, thus an inverter is required to generate the versatile AC power used in standard loads as shown in Figure 1.5. Energy sourced from renewable resources is variable in nature and necessitates feedback controlled electronics to maintain stable power production. This project is focused on delivering a microinverter system as a robust solution for solar power applications.

Microinverter deployment strategies for solar systems offer improved reliability and reduced installation costs. Widespread integration of microinverters with individual solar panels to form AC power generation units removes single-point failure risks found in common centralized PV inverter arrangements. Microinverters for single panels can have longer mean times between failures since they are often designed for the 200W power range and operate with lower temperatures due to reduced heat waste. The economies of scale for manufacturing many consumer microinverters reduces hardware costs found in building few larger high power PV inverters.[3] Another benefit of this microinverter approach is that the feedback control optimizes the power generation of all panels in the system to operate most efficiently considering each *unit's* individual conditions. This makes them robust in situations like partial shading.

Traditional inverters use a technique know as pulse width modulation (PWM) to generate AC power. This method is not as well adapted to input energy variation and also contributes some harmonic distortions to the output. A new hybrid systems based feedback control scheme for power switching is being utilized in this inverter as an alternative to PWM. AC output stability is achieved through comparison of real time measured voltages and currents compared to a defined reference sinusoid.[2] The hardware implementation of a microinverter through this project will highlight the advantages of hybrid control to PWM.

Chapter 2

Hardware

2.1 Hardware

2.1.1 Inverter System Overview

The inverter is responsible for converting the DC power output by the photovoltaic panels into the 120Vrms power used by the power grid. The heart of the inverter is the C2000 microcontroller by Texas Instruments which implements the Hybrid control algorithm for power switching. The microinverter has three main power conversion stages: boost converter, inverter, and output filter. The boost converter increases low panel voltage up to 200V while implementing maximum power point tracking for the PV panel. The inverter consists of an H-Bridge that creates the AC wave with 169.73V peak voltage. The output filter removes high frequency switching noise and passes the 60Hz intended for standard loads. Feedback control operation requires voltages and currents to be measured in real time by the microcontroller analog-to-digital converter. Logic and signal conditioning power is sourced through an auxiliary power supply which steps down solar input voltage. This system is represented in the depiction found in Figure **??**.

2.1.2 Logic Power Supply

A logic power supply was required for the inverter board to run the microcontroller, driver circuits, peripheral sensor network. Three different voltage rails were needed at 12V, 5V and 3.3V. The input from the solar panel was utilized to create these different power rails through a small connection circuit. Efficiency of this type of system is considered crucial, so cascaded switching DC/DC buck converters were used to create

the 12V and 5V rails. A fast responding low-dropout linear regulator created the final 3.3V from the 5V rail. The circuit schematic for the logic power supply is shown in Figure A.1.

A variety of features were added to this front input interface for system protection and functionality. There are two options for sourcing power to the inverter board: banana jack connections for solar input and a DC barrel jack plug for testing input. These two circuits are configured so that only the barrel jack will source power if both happened connected and energized at the same time. Two switches are included for toggling the DC jack input and for switching conversion power into the main system. A ten amp blow fuse is included in series with the input to the inverter to prevent damaging short circuit conditions. A green LED on the 3.3V rail turns on to show the system is properly functioning.

2.1.3 Boost Converter

A standard PV microinverter contains two principle energy conversion stages with a DC/DC module and DC/AC module. This section highlights the design process of a DC/DC boost converter for solar microinverter implementations. The boost circuit includes many elements and considerations of the final inverter system including power switching hardware and sensor circuits.

The boost stage designed is essential for photovoltaics since it allows consistent maximum power sourcing from the panel. Solar panels have nonlinear relationships between output voltage and current. The maximum power product of the electrical values will vary depending on light intensity levels, loading, and temperature. A microcontroller unit (MCU) is used with this boost converter for create a feedback control scheme called a maximum power point tracker (MPPT). A versatile DC/DC boost stage is needed for microinverters since it can dynamically vary power switching based on the MPPT to compensate for the wide range of potential solar panel voltages. This converter topology utilizes a discrete MOSFET for switching since microcontroller interfacing is required. Traditional boost converter integrated circuits (ICs) do not satisfy the MPPT control needs or power requirements needed for this type of system.

A generic boost converter circuit shown in Figure 2.1. Operation of a boost converter has two distinct phases depending on the switch mode of the power MOSFET. The gate of the MOSFET is sent different logic values at specific duty cycles through PWM. When the drive circuit outputs logic high in Figure 3, the drain to source path will conduct in the saturation region. This provides a path to ground for the current flowing through the inductor and reverse biases the clamp diode. Over time energy is built up

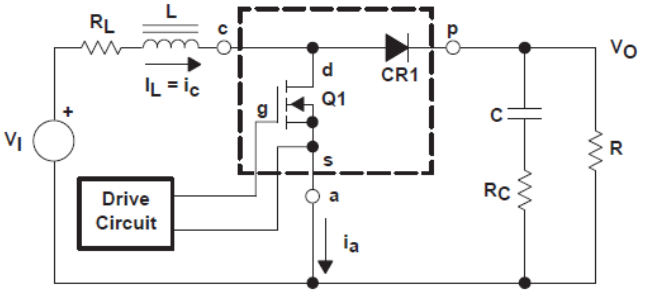

FIGURE 2.1: Traditional Boost Converter Circuit

Parameter	Value
$V_{in} = V_{panel}$	0V to 40V DC
I_{in}	5.47A DC max
$V_{out} = V_{load}$	169.73V Min to 200V max
I_{out}	1.18A DC max
P_{rated}	200W
$f_{switching}$	50 kHz

TABLE 2.1: Electrical Specifications for Boost Converter

in the magnetic field of the inductor coil from this solar panel sourced current. Then logic low is sent to the MOSFET to shut it down. Considering the sudden change in current flow, a large voltage kickback occurs across the inductor and forward biases the diode. Current now can flow to charge the output capacitor and feed the load.

When the MOSFET is conducting, the output capacitor provides power to the load until it receives new charge during the second period. Due to the switching characteristics and inductor physics, a large ripple current flows through the inductor. An important design constraint it to make sure the inductor current always remains greater than zero such that continuous conduction mode (CCM) is maintained. CCM ensures a desired voltage transfer function for gain in this design. Additionally, there are some parasitic elements included in the circuit that contribute to power loss. These undesired effects include the DC resistance of the inductor windings and the equivalent series resistance of the capacitor. The electrical specifications of this boost converter are listed in Table 2.1.

The Sharp solar panels, utilized in this design as an input power source, will have varying voltage and current output capabilities depending on lighting conditions. Maximum output power for the 170W panels occurs at $V_{panel} = 34.8V$ and $I_{panel} = 4.9A$. A input voltage range will be selected to define a sourcing range with usable power output considering the panels highly nonlinear relationship regarding I vs V shown in Figure 2.2. Output voltage is defined based on the minimum value needed for the

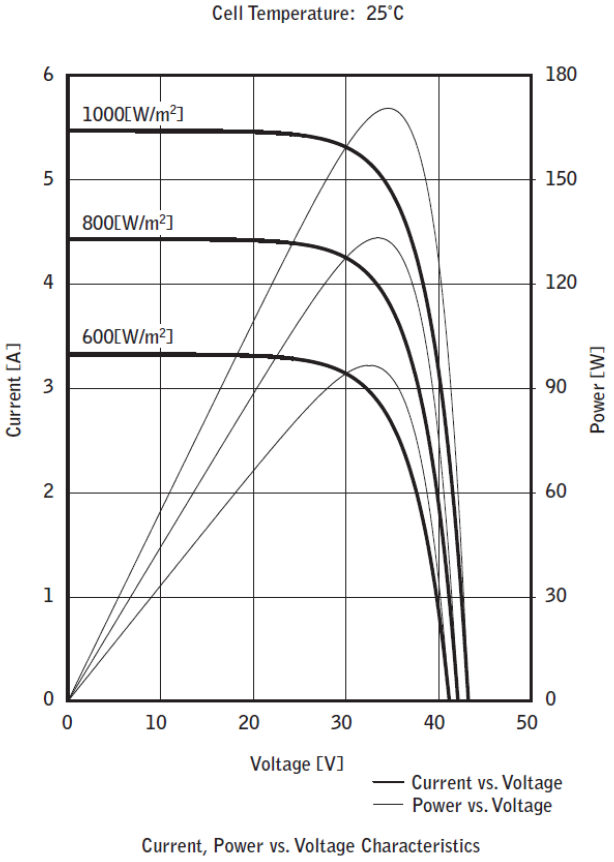

FIGURE 2.2: Solar Panel IV Characteristics

120Vrms/0.707 = 169.73V peak value needed for the DC/AC inverter stage input. Maximum power of 200W is an upper limit buffer set for this design considering the solar panel approach with microinverter topology. Switching frequency is set at 50 kHz based on recommended ranges and to compare to the TI Solar Explorer Development Board also used in this project.[4]

The process of designing the boost circuit hardware involved research of reference application notes and in running PSPICE simulations. PSPICE computer software produced by Cadence\OrCAD was used to simulate the boost converter circuit. The max power voltage of the PV panels was selected for primary simulation testing. Calculations were performed to best select power components for operations of the boost converter at high power values. These design steps considered power dissipation thermal limits, critical inductance, and critical capacitance. The PSPICE circuit encapsulating the entire boost converter design is shown in Figure 2.3.

The operation of the boost converter in PSPICE required a switching signal to control the power MOSFET. Implementation of the converter will utilize C2000 MCU logic signals and Silicon Labs gate driver circuits for MOSFET switching. This simulation uses a variable duty cycle pulse source for continuous switching at the set frequency.

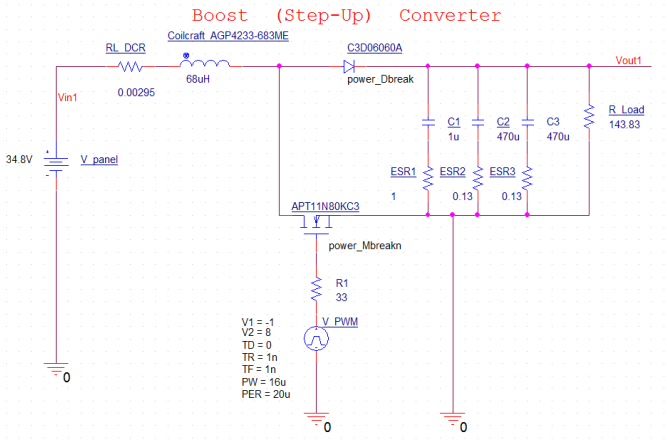

FIGURE 2.3: PSPICE Boost Converter Simulation

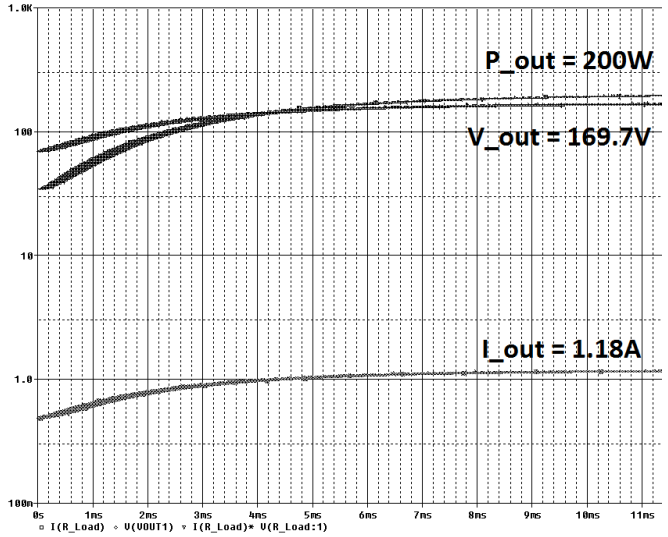

FIGURE 2.4: Boost Converter Approaching Steady State

This configuration is currently being run open-loop, but will include a compensation feedback control circuit to maintain stable output voltage in the final design. The solar panel 34.8V input at max power required a duty cycle set at 80.13% for the PWM switching signal to achieve 169.73Vdc out. A load of 143.83Ω was connected at the output to simulate max power current draw and realize the 200W capability of the system. Real output power will be less in practice due to additional inefficiencies, panel operating conditions, and sourcing configuration capabilities. The output power, current and voltage as the boost converter approaching steady-state conditions from start-up are shown in the simulation plot of Figure 2.4.

This boost converter was designed to operate in continuous condition mode (CCM). Ensuring this condition meant that the inductor current will never fall to zero and the panels are always sourcing current. This was achieved by selecting an inductor value above the calculated critical inductance and making sure it has appropriate current

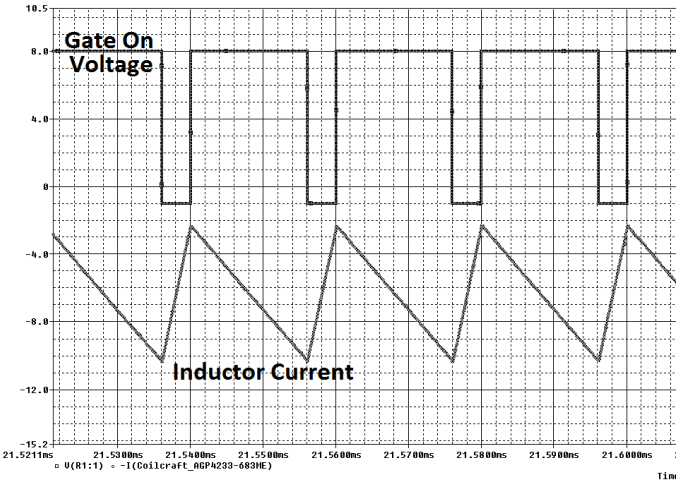

FIGURE 2.5: Switching Signal and Inductor Current

handling capabilities. Figure 2.5 shows the rippled inductor current in CCM and the MOSFET switching signal that creates these dynamic effects in the boost converter.

Another factor that will influence the switching of the MOSFET is the maximum power-point tracking (MPPT) algorithm implemented by the microcontroller. This works to vary the input impedance of the converter as seen by the panel to optimize power output at 34.8V. This method uses feedback control based on the voltage and currents measured by the ADC to perturb and observe the boost condition. Since the solar panel voltage can vary, the simulation circuit was run at the minimum 20V to check that the power output voltage could be maintained. The simulation results confirm this with $V_{out} = 169.73V$, $V_{panel,in} = 20V$, and a new duty cycle set at 89.25%.

Ratings on voltages and currents concerning components were determined through worst case calculations.[5]

$$I_L = I_{diode} = I_{drain} = \frac{2}{\sqrt[2]{3}} I_{in,rms} \tag{2.1}$$

$$V_{cap} = V_{out} = 1.5V_{out,typ} = 2(169.73V) = 254.6V \tag{2.2}$$

$$V_{diode} = V_{DS,FET} = 2(169.73V) = 339.46V \tag{2.3}$$

$$I_{cap} = I_{out,rms} = 1.18A \tag{2.4}$$

The clamp diode was selected to be a SiC schottky type since it offers fast recovery times with low reverse recovery charge Q_{rr} for reduced switching losses. Since the diode resides in the power conduction path, energy dissipation was analyzed. The diode was modeled as a series circuit of a temperature dependent voltage source V_{diode}

and resistance R_{diode}.[6]

$$V_{diode} = \alpha T_{junction} + V_{diode,0} = 0.95V \tag{2.5}$$

$$R_{diode} = \beta T_{junction} + R_{diode,0} = 0.103\Omega \tag{2.6}$$

$$P_{cond} = {I_{diode,rms}}^2 R_{diode} + I_{out,max} V_{diode} = 2.76W \tag{2.7}$$

$$I_{diode,rms} = \frac{I_{out}}{\eta} \sqrt[2]{\frac{16V_{out}}{3\pi V_{in}}} = 3.99A \tag{2.8}$$

$$P_{switching} = Q_{c,diode} V_{out} f_{sw} = 0.14W \tag{2.9}$$

$$P_{dissipation,diode,total} = P_{conduction} + P_{switching} = 2.91W \tag{2.10}$$

This power dissipation falls within the limits of the diode, but for good measure it will be heat sinked with a TO-220 bolt-on type sink and thermal grease. The output capacitance was designed as a parallel arrangement of two types of capacitors for fast and slow transient response. This increased the total capacitance while reducing the equivalent series resistances (ESR) of the devices. This helps reduce of the output voltage ripple of the boost converter. Since DC voltage output is required, maximum ripple of 50mV = δV_{out} is selected as the tolerance limit. To achieve this voltage ripple design, a critical minimum output capacitance was calculated.

$$Duty\ Cycle = D = 1 - \frac{V_{out}}{V_{in}} = 79.81\% \tag{2.11}$$

$$C_{critical} \geq \frac{V_{out} D}{F_{sw} \delta V_{out} R_{load,maxpower}} = 370\mu F \tag{2.12}$$

Additionally, the continuous conduction mode requirement sets a minimum valued critical inductance. This was calculated to ensure current always remained flowing through the inductor.

$$L_{critical} \geq \frac{R_{load,maxpower} D(1-D)^2}{2 f_{sw}} = 50\mu F \tag{2.13}$$

The philosophy of modular design is being practiced with the hardware development. The boost converter circuit is first being prototyped as a stand-alone unit that can be tested. Necessary interface connections with this board will be input power from the solar panel, output power for H-bridge, PWM control, input voltage feedback, output voltage feedback, input current feedback, and switch current feedback. Additionally, logic power rails of 12V, 5V and 3.3V will be supplied. A abstracted representation of the boost converter is detailed in Figure 2.3.

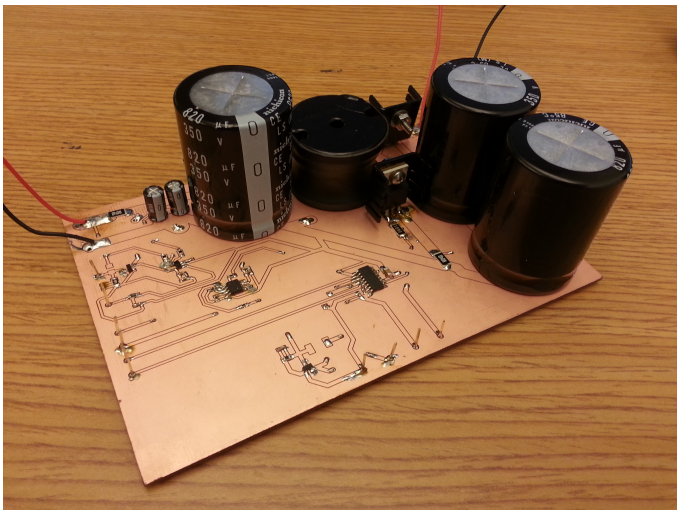

FIGURE 2.6: Assembled Boost Circuit

The board was schematic captured and PCB laid out in Eagle. The circuit schematic shows the conventional boost converter, the associated sensing and signal conditions circuits, and the gate driver circuit. Figure A.8 contains this schematic which had many design concepts inspired by the Texas Instruments Solar Explorer Development environment.[7] The energy conversion portion of the boost board is shown in Figure A.2 with an additional input filter capacitor bank. The sensing of the PV panel voltage and output voltage consisted of basic divider circuits with MCU pin protection diodes as shown in Figure A.3 and Figure A.7. The input DC PV current is measured with a current shunt resistor IC with high common-mode rejection in Figure A.4. The switching transistor current is measured with a differential op-amp configuration as shown in Figure A.6. Lastly, the switched MOSFET has a low-side gate driver IC taking in PWM as depicted in Figure A.5.

The top layer of the PCB is shown in Figure A.9 and the bottom is shown in Figure A.10. The first generation prototype boost board PCB was created using a LPKF Protomat M60 router. This machine allows modified PCB gerber files to be milled into two-layer boards from standard copper clad plates. The resulting unpopulated boost board is shown in Figure A.11. The board was then populated with components into a completed circuit as shown in Figure 2.6 Testing of the boost circuit configured its voltage gain capabilities with outputs reaching in excess of 120V with open-loop testing.

2.1.4 H-Bridge

Arguably, the lynch-pin of any inverter is the H-Bridge circuit. In order to generate a pseudo-sinusoid from a DC source, we must first have the means to switch this DC

voltage on and off, and also to drive current bidirectionally. Thus, careful design of the H-Bridge circuit was a critical step in our development.

For a successful H-Bridge design, we needed to consider a number of factors including the speed of switching, deadband, transistor type, transistor gate drivers, thermal analysis, and careful layout due to the speed of the signals involved and the resultant EMI.

After some analysis of the algorithm, further discussed in Section (CITE ME!), we found that the speed of switching with the hybrid algorithm - unlike the switching speed of PWM algorithms which modulate over a fixed carrier frequency - was highly dependent on the width of the tracking band (@todo:Make sure that the tracking band is introduced in BG!). Consequently, there is also a dependence on the resolution of the ADC, and the frequency at which our interrupt is serviced. Through simulations in Matlab we were able to determine that the speed at which the algorithm caused the inverter to switch were well under 100kHz for a 'small-enough' tracking band.

Because the switching speed is a function of width of the tracking band, sampling rate, and ADC resolution it is by no means a straightforward calculation to determine the frequency of switching. Contrast this to the PWM approach where this determination is trivial. In any case, because the switching speed was tunable by way of modifications to the with of the tracking band, it was decided that it was not necessary to go with more advanced FET technologies like Galium Nitride that are more than capable of switching in the MHz range under full load.

Shoot through is another important consideration with the H-Bridge circuit, as the possibility of a nearly direct path from power to ground can exist as the circuit switches from outputting +Vdc to -Vdc. To prevent this, a mandatory off time just be implemented in hardware of software. We took the software approach, although the gate drivers we used also implement a minimal headband in hardware. In the configuration of the PWM driver in our micro controller code, a 20ns headband was implemented, based off of the 60MHz clock.

For the transistors in the H-Bridge, we decided on field effect transistors, (FETs), and in particular the CoolMOS^{TM} variety built by Infineon. The IPP60 comes in a standard TO-220 package allowing for the use of standard heatsinks, has a Vds max of 650V - about three times what we would need - and a maximum continuous current of 31 Amps. Additionally, these FETs have a very low gate charge (low capacitance, smaller switching losses), low Rds on, and great slew rate. The fact that the drain to source resistance is low helps with our thermal budget, and also means higher efficiency overall. Because of the bipolar switching that the hybrid algorithm called for, we felt that it

was wise to over specify our switches. Finally, the freewheeling diode found on many H-Bridge circuits was not needed here since the diode inherent to the construction of the FET was sufficient to handle any expected back currents in our load.

2.1.4.1 Switching Loss and Conduction Loss

One of the primary factors in the overall efficiency of a modern switching power supply is the switching loss inherent to FET transistor technology, and also the conduction loss associated with Rds on, the resistance of the FET while it is conducting. Switching loss occurs whenever the FET changes state, and can be roughly understood as the net amount of charge it takes to drive the gate of the device. This amount of charge corresponds to a loss of current that could have gone to drive the load in question. This effect can be mitigated in some cases with parallel capacitance at the gate, using 'faster' FETs, and choice of the freewheeling diode[8].

The heat sinks in our design are connected to the AC ground. Both the low-side and high-side transistors are insulated from the heat sink with thermal pads and insulating shoulder washers. According to the Transform application note we used as reference during our design, "For the high-side transistor, capacitance between the TO220 tab and the heat sink will add to switching loss, and so a thick and/or low permittivity insulator should be used."[9]

The gate drivers we chose, the SI823x family from Silicon Labs, have a built-in under voltage protection to prevent 'nuisance trips' and to add a good deal of noise margin. The gate drivers utilize a typical 'bootstrap' circuit to operate as both a high-side and low-side driver.

Because of the relatively high speeds involved with the H-Bridge, careful layout was especially important here. Parasitic capacitance on gate and drain loops are a min cause of overshoot and ringing in the circuit, and thus the total enclosed area between the gate drive and the FETs was kept to the absolute minimum. In order to minimize inductance in the output current path, high current power and ground planes were utilized. Small ferrite beads were used between the gate drive output and the gates of the FETs to reduce ringing cause by coupling of the drain current to the gate drive loop- these were found to be more useful than small resistances which are sometimes used [9]. High voltage SMD ceramic bypass capacitors were placed directly underneath either side of the H-Bridge circuit to minimize the series inductance in the circuit.

2.1.5 RLC Filter

In a typical PWM design, the driver behind the cutoff frequency of the output filter is the carrier frequency of the PWM signal. In most cases, this carrier frequency is invariant, and therefore allows for a relatively straightforward design parameter during the time where components are selected. Of course, THD is also a primary driver of part selection and valuation of the analog components.

In order to ensure that the vector fields on the power plane are such that the hybrid algorithm can ensure forward invariance, and that the solution of the system converges to the tracking band in finite time, it is necessary that our design adhere to a set of constraints on the RLC filter described in [2].

Namely, our filter components must meet the following constraints: first, we must satisfy the condition that $LC\omega^2 > 1$ - this property ensures our vector fields are oriented correctly throughout the desired trajectory on the VI plane. Second, we have that the capacitor value must be determined by the output voltage amplitude and current amplitude by the relation:$\frac{I_l\omega}{V_c}$ where I_l is the target output current, and V_c is the target output voltage. From this final condition we observe that the value of the capacitance can be driven up by increasing the target current, decreasing the target voltage, or increasing the frequency of operation.

Let's examine this mathematical condition through the lens of circuit analysis. By inspection, we note the similarity of this condition to the condition for resonance in a series RLC circuit - which is the subject of study in [2]. This condition is given by $\omega_0 = \frac{1}{\sqrt{LC}}$ Taking the square of both sides in the expression, find that $\omega_0^2 LC = 1$, and we see that the condition on the filter components given states that the resonant frequency of the circuit ought to be greater than unity. If we suppress the variable for capacitance in $LC\omega^2 > 1$ given the condition $C = \frac{I_l\omega}{V_c}$, we obtain:

$$L > \frac{V_c}{I_l\omega^2} \tag{2.14}$$

The expression obtained in 2.14 adds a considerable degree of inductance compared to that in a typical PWM inverter. it was considered initially that the quality factor, Q might be at work in the conditions on the filter, but we find that the quality factor for the series RLC filter is given as $Q = \frac{1}{R}\sqrt{\frac{1}{LC}}$, and the analysis in [2] makes no mention of the damping term.

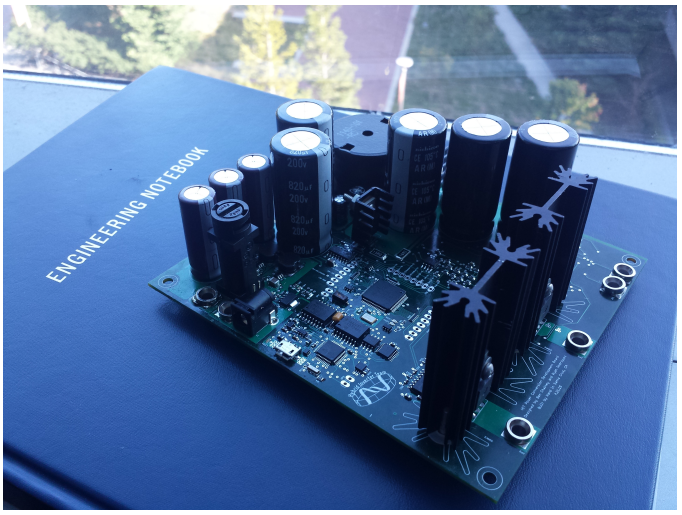

FIGURE 2.7: The Hybrid Inverter Team's assembled PCB

2.1.6 AC Output Sensors

2.1.7 Microcontroller and USB-JTAG Interface

2.1.8 Complete Hardware System

The final PCB layout utilized a four layer design with power and ground planes on the two middle layers. Multiple layers allowed for components to be placed closer together, and achieve a smaller overall form factor. Additionally, we were able to use polygon pours on the central power layers to boost current carrying capacity and aid in the dissipation of heat. Ground plane isolation was used to separate areas with switching power signal and digital logic.

Input and output connection points were placed around the perimeter of the board for easy interfacing with cables. Eagle PCB was utilized for this design and the board house Advanced Circuits was used for board fabrication. Parts were sourced from the Digikey. The top layer of the PCB is shown in Figure A.12 and the bottom layer in Figure A.13. The completed circuit board is displayed in Figure **??**.

The inverter sits in an enclosure mounted to the back of a solar panel to follow the microinverter topology. The output filter and transformer are constructed using turret board, and is also mounted on the panel mounting system. For testing in outdoor sunlight environments, a wooden stand for the solar panel and inverter system as shown in Figure 2.8.

FIGURE 2.8: Solar Panel Stand

Chapter 3

Software

3.1 Software and the Hybrid Algorithm

3.1.1 The Hybrid Algorithm

Without being a rehash of Dr. Sanfelice and Jun Chai's paper on the hybrid control of the power inverter[2], we aim to briefly clarify the mechanisms behind how the hybrid inverter works, and discuss the process we undertook in implementing the algorithm on our custom hardware validation platform.

The derivation of the hybrid control algorithm is roughly as follows: given that the response of a series RLC filter can be thought of as a linear oscillator with a damping term, and given that we have a set of desired output parameters - namely the amplitude of the output voltage, the amplitude of the output current, and the angular frequency ω, then by solving the resultant linear differential equation, we can derive a reference solution for the given system to a perfect sinusoidal driving signal. In reality we will not have a sinusoidal driver, but rather some permutation of a square wave capable of switching between $+V_{dc}$ and $-V_{dc}$, where V_{dc} is the DC bus voltage at the input to the H-Bridge controlled by the state variable q. We consider $q = 1$ to correspond to $+V_{dc}$, $q = -1$ to correspond to $-V_{dc}$, and $q = 0$ to correspond to 0.

We can think of the resultant solution to the sinusoidal driver as the ideal to which our system should strive, and therefore any deviation from this ideal can be considered as an error that needs to be corrected by the controller. These errors are detected by building a tracking band around the reference solution; this is done by choosing a neighborhood around the reference solution. Collectively, this region is known as the tracking band. Now, if we consider the instantaneous state of the system to be a vector made up of the capacitor voltage and the current through the inductor, then we can

measure the location of the vector in relation to the tracking band on the VI plane. Since the inductor and capacitor are at all times 90 degrees out of phase without any load or perturbation, the ideal solution takes the shape of an ellipse on the VI plane. By taking the state vector's position relative to the tracking band, and knowing the trajectory at a given region on the VI plane, it is straightforward to develop a small set of rules governing the switching of the inverter, i.e. the jump between the three states of the H-bridge circuit, namely $q = 1$, $q = -1$, and $q = 0$. With this brief foundation, we are able to steer the state of the system around the VI plane, with the resultant output being a pseudo-sinusoid with the desired voltage and current amplitude and frequency.

3.1.2 Analysis of the Switching Waveform Given by The Jump Map of the Hybrid Algorithm

As stated previously, PWM algorithms offer a relatively simple means for analyzing the switching waveform since we are modulating a carrier signal of known frequency. This is not quite the case for the hybrid algorithm where the switching occurs in response to where and when the state of the system leaves or enters the tracking band, how often we check it, and the with of the regions $M1$ and $M2$ shown in Figure 3.1.

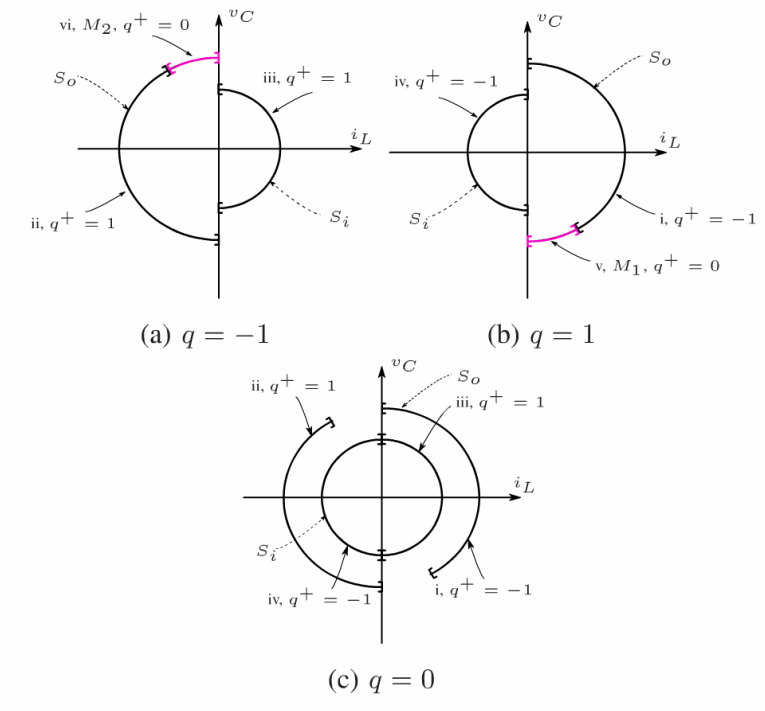

FIGURE 3.1: Jump Map of the Hybrid Algorithm [2]

In effect $M1$ and $M2$

3.1.3 Selecting the μC

With the myriad of choices available to designers of power systems, the task of selecting a microcontroller for a power conversion system is no easy task. The first step in making an educated decision as to the choice of controller was to survey the current landscape and find some examples of the most popular microcontroller choices currently being used in power applications today. The controllers that I found most frequently were the dsPic family by Microchip, and the Piccolo family by Texas Instruments. The reasons that these two families have found dominance in this field became obvious: they were low cost, had widespread support, support for 'real-time' control, and excellent libraries available for common power tasks. With the choice narrowed down to two families, it became a simpler task to compare the benefits of both families. Both had similar power consumption, but the TI family of Piccolo controllers came with the option of faster clock speeds, and correspondingly, faster ADC sampling times. Additionally, the resolution of the PWM and ADC modules was higher than those of comparable Microchip controllers. The final check-box in the TI column was the tight integration of the Piccolo family of controllers into two of the power development boards that we were considering. With code portability from our development kit to a final implementation being of paramount importance, it was decided that we should go with the Piccolo family. In particular, we chose the F28035 for its mix of speed, low-cost, and wide array of digital control capabilities including a unique co-processor feature known as the CLA which enables offloading of many control tasks to a secondary chip that can be programmed in assembly. With the integration of this Control Law Accelerator or CLA, we are able to significantly increase the complexity of our algorithms for a given clock speed while running multiple state machines and servicing interrupts occuring at multiple frequencies and to do it *faster* than comparable microcontrollers might be able to.

3.1.4 Software System Overview

The software system overview is as follows: as with any microcontroller system, it is necessary to first configure the various low level peripherals such as the ADC, PWM, phase synchronization of PWM interrupts, PWM safety trips to avoid over-voltage conditions in our switching circuits, clock and PLL configurations, and etc. The next step in the development was to design a logical, well organized and most importantly, extensible, software system to work with. The firs phase in this development was to develop a state framework for the organization of the various tasks associated with our power conversion system, namely the MPPT algorithm, tasks associated with power-up and

shutdown and instances where the power from the solar source is no longer sufficient to meet our output power guarantees. Quite secondarily, we also utilize these state machines to run our LED visual indicators that the code is looping through the FSMs as expected. The high level overview of the software can be seen in Fig. 3.2.

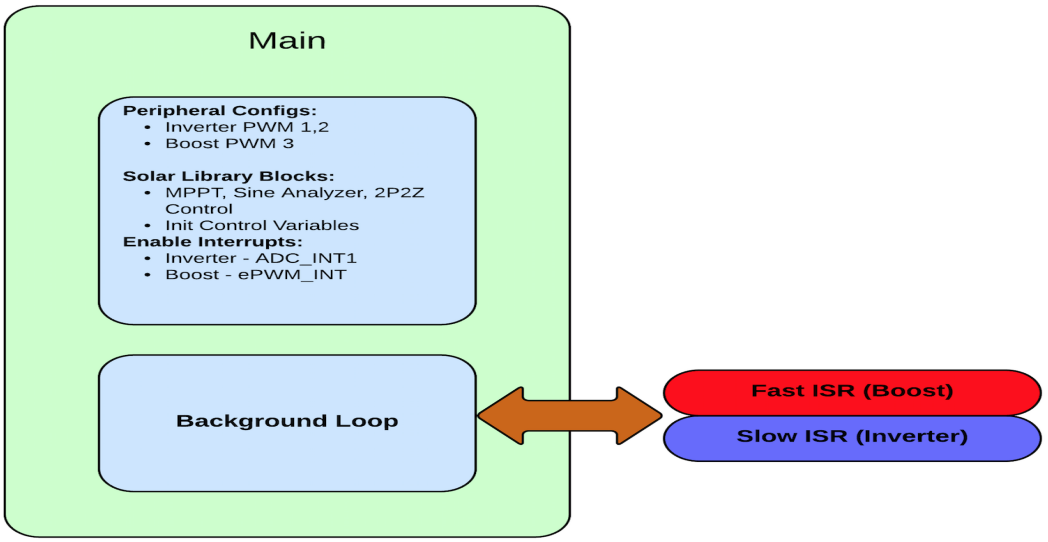

FIGURE 3.2: Software System OVerview

The two primary tasks of the software system are undoubtedly the control of the DC boost circuit, and the control of the inverter hardware. These are pictured in Fig.3.3 and Fig.[**?**] respectively. These tasks are accomplished using two interrupt mechanisms which are tied to the frequency of two separate channels of the controllersPWM modules. Because we need these interrupts to operate at potentially different frequencies corresponding to the desired bandwidth of the control loops, it is necessary to synchronize the phases of these control algorithms using the Piccolo's PWM phase sync capabilities. This functionality ensures that our control loops are not stepping on eachothers toes and trying to get serviced at the same time.

3.1.5 State Framework

Although simple state frameworks are possible using functions as a sort of 'pseudo state,' these implementations tend to become exceedingly complex and ahrd to follow for non-trivial state machines. Further, cobbling together new state machines or adding new states can be a daunting task with the resulting spaghetti code.

For these reasons, we decided to 'roll our own' state framework using objected oriented patterns in C. This involved creating an FSM structure that holds a reference to an FSM

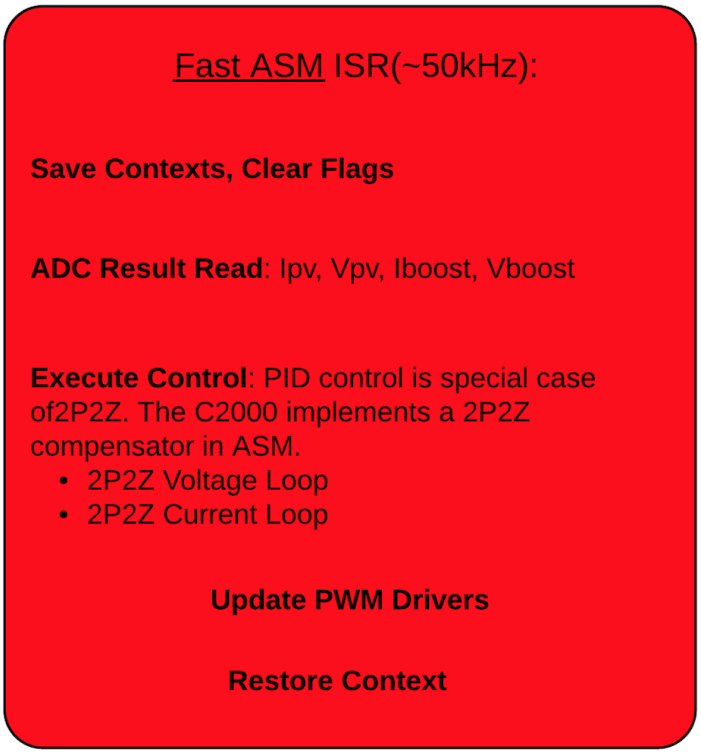

FIGURE 3.3: The Fast ISR

object. This FSM object is actually a type defined variable that is, loosely speaking, of the type 'pointer to function'. Additionally, this strategy necessitated the implementation of several helper functions for various tasks like constructing the FSM object, state initialization, state transitions, and event dispatching.

The custom 'pointer to function' type is declared in C as follows:

```
typedef void (*State)(Fsm *, Event const *);
```

The definition above used the C keyword typedef. What typedef allows us to do is to declare a new data type for our own use. In this case, our type is 'pointer to function.' This is an enabling tool for representing the state of a machine as a function.

The next step trick is to use structures to organize all of our data in a meaningful way - for our purposes, you can think of these structs as classes, save for the fact that their methods are declared outside of the structure itself. Allow me to make a structure for the FSM itself, and call it 'Fsm.' Here's how we do it:

```
struct Fsm
{
```

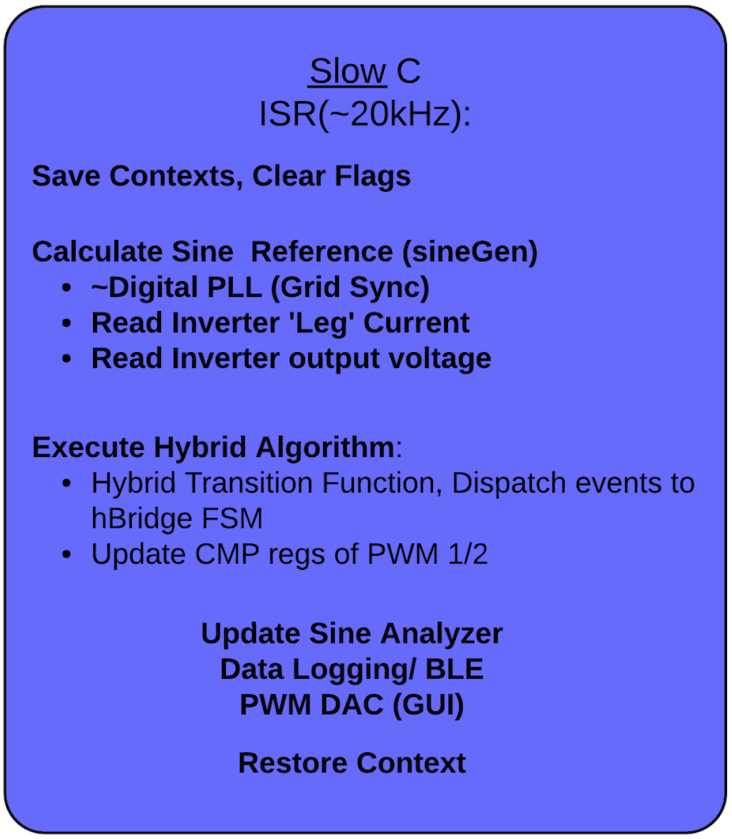

FIGURE 3.4: The Slow ISR

```
        State state__; /* the current state */
};
```

The Fsm struct stores the function pointer as its sole attribute. Note that I'm using the trailing underscores to indicate that the variable is private and should not be tampered with in a nod to the Python style of adding leading underscores to indicate such members of classes. I will probably switch to leading underscores in a later revision, but this is wholly unrelated to the current discussion.

Students of electrical and computer engineering should be familiar with the concept that state machines respond to what are formally known as 'input alphabets.' These alphabets have strict definitions from a mathematical standpoint, but for our purposes, all we need to understand is that the machine should transition according to a certain mapping if it gets an event, and should - typically - do nothing if we do not send it an event. Accordingly, we should also declare some structure to manage event signals. This is done with

```
struct Event
```

```
{
        Signal signal;
};
```

At this point we're ready to implement the functions that will manage this framework and allow for the creation of arbitrary state machines. These functions are inlined for speed - they eliminate a lot of the overhead of normal function calls - and they are well suited to this implementation since we are very concerned with speed and efficiency on our micros.

First up, we need to initialize the current state, i.e. give it something to point to, i.e. we will create an Fsm struct and invoke a function pointer for it to use. We will also set this pointer to the initial state of the FSM. We will do this with:

```
#define _FsmCtor_(self_, init_) ((self_)->state__ = (State)(init_))
```

Next, it is desirable to go about triggering our initial transition. This is to say, our Fsm now points to the function we are using as the initialization state where we can do all the prep work we may need for the smooth operation of the Fsm, but now we want to get to work and make the machine run. We will trigger the initial transition with the following:

```
#define FsmInit(self_, e_) (*(self_)->state__)((self_), (e_))
```

Now we've got a state machine that is in some state - the Fsm 'class' is pointing to some function we are using as a state - and we are ready to respond to events. How do we do it? With this inlined macro:

```
#define FsmDispatch(self_, e_) (*(self_)->state__)((self_), (e_))
```

This will take some member of our particular Event structure that we defined above, and send the event to the function for it to respond to. This response is usually some action or state transition, or both.Finally, to round out our Fsm back-bone functions, we need something to actually implement the state transition, i.e. some function that updates the pointer and sets it to the new state when appropriate. This is accomplished with the following macro:

```
#define _FsmTran_(self_, targ_) ((self_)->state__ = (State)(targ_))
```

Because the implementation of the state framework was, in our collective opinion, a major design hurdle, we feel it is productive to give the user a sense of the simplicity

that this implementation imparts on FSM implementations. This is a particularly useful feature for us since we have several state machines in our systems, and the overall software design goal is extensibility.

```
int main()
{
    int returner = 0;
    hBridge k;
    hBridgeCtor(&k);
    FsmInit((Fsm *)&k, 0);
    for (;;)
    {
        hBridgeEvent ke;
        ke.code = getc(stdin); //replaced by ADC input in uC
        getc(stdin);
        returner = hBridgeTransitionFunction(k, &ke);
        if(returner == -1) return 0;
        FsmDispatch((Fsm *)&k, (Event *)&ke);  //dispatch
    }
    return 0;
}
```

While this brief example departs from our actual implementation on the micro, it gives a quick overview of why the time spent developing this state framework was time well-spent, and hopefully provides some insight as to why the research into this area was worthwhile.

3.1.6 The Digital Power Library

One of the key features that made the Piccolo family of micro's an attractive option was the extensive power library that the the micro offered. In particular, the chip has the capability to run digital compensators of the 2P2Z form described in the section on the boost controller, and 3P3Z compensators as well. In addition, we have the ability to implement PID controllers with on-the-fly coefficient tuning via JTAG.

3.1.7 Hybrid Algorithm Implementation Details

In the initial stages of research and development, the hybrid inverter team first sought to recreate the simulations described in [2]. These simulations were best accomplished using the Hybrid Equations Toolbox in Matlab. Although the particulars of this implementation would be quite different than the final implementation in hardware, it was an excellent exercise in understanding the particulars of the algorithm, and also to give us a reference while implementing the hybrid controller on the C2000 micro controller in C.

3.1.7.1 The Matlab Implementation

The first step in building the algorithm with Matlab was to translate the flow and jump sets into Matlab code. Determining which set the solution of the RLC filter is a member is critical in determining when to switch, and when to allow the differential equation solver to flow.

These sets roughly translate as follows. Note that it can be helpful to refer to Figure 3.1 when deciphering the subsequent expressions. The flow set was translated as follows:

```
%================================
%Hs Controller - Hfw in the loop
%================================
if(p == 1)     % if were between the tracking bands, flow
    if((Vz0 >= cin) && (Vz0 <= cout))
        inC = 1;
    else    % not in the tracking bands, report not-flowing
        inC = 0;
    end
%================================
%Hs Controller - Hg in the loop
%================================
else
    if(Vz0 >= cout)        %if were beyond Co
        inC = 1;
    elseif(Vz0 <= cin);       %if were within Ci
        inC = 1;
    else
        inC = 0;
```

```
    end
end

%======================
%For the Hfw Controller
%======================
if(p == 1)
    if((Vz0 >= cin) && (Vz0 <= cout))
        inC = 1;
    else
        inC = 0;
    end
else
    inC = 0;
end

%======================
%For the Hg Controller
%======================
if(p == 2)
    if((Vz0 <= cin) || (Vz0 >= cout))
        inC = 1;
    else
        inC = 0;
    end
else
    inC = 0;
end
```

The translation of the jump set is shown in the following code section. The jump set signals to the framework that it is time to change states. This requires that the solver change the set of equations that it is operating on.

```
%====================================
%Determine if Solution is in M1 or M2
%====================================
mEpsilon = .5; %dependant on the current and voltage targets
if ((abs(Vz0 - cout) < err) && ((il >= 0) && (il <= mEpsilon)) && (vc
   <= 0))
```

```
    M1 = 1;
else
    M1 = 0;
end
if ((abs(Vz0 - cout) < err) && ((il >= -mEpsilon) && (il <= 0)) &&
    (vc >= 0))
    M2 = 1;
else
    M2 = 0;
end

%======================
%For the Hs Controller
%======================

%p == 1 -> Hfw in the loop
%p == 2 -> Hg in the loop

if(p == 2)
    if((Vz0 >= cin) && (Vz0 <= cout))
        inD = 1;
    end
else
    inD = 0;
end

%======================
%For the Hfw Controller
%======================
if(p == 1)
    if(q ~= 0)
       if( (abs(Vz0-cin) <= err) && (il*q <= 0))
           inD = 1;
       elseif( (abs(Vz0-cout) <= err) && (il*q >= 0))
           inD = 1;
       end
    elseif (q == 0)
        if( (abs(Vz0-cin) <= err) && (q == 0))
            inD = 1;
```

```
        end
    end
end

%======================
%For the Hg Controller
%======================
if(p == 2)
    if((Vz0 >= cin) && (Vz0 <= cout))
        inD = 1;
    else
        inD = 0;
    end
end
```

Now that we've determined whether we're in the flow set C, or the jump set D, we can perform the requisite logic for the controller. Here we give priority to jumps; this is to say, if we're in the sets C and D, give priority to the jump set and perform the necessary state transition instead of continuing to flow continuously. If we're in the jump set, this signals to the controller that we ought to execute a transition according to:

```
%======================
%For the Hs Controller
%======================
if(p == 2)
    if((Vz0 >= cin) && (Vz0 <= cout))
        pplus = 1;
    end
else
    pplus = p;
end

%======================
%For the Hfw Controller
%======================
if(p == 1)
    if(q ~= -1)
        if( ((abs(Vz0-cout) <= err) && (il >= 0) && (~M1)) ||
   (((abs(Vz0-cin) <= err)) && (il <= 0)) )
```

```
        qplus = -1;
        end
    elseif ( ((M1) && (abs(il - mEpsilon) >= err) && (q == 1)) ||
   ((M2) && (abs(il + mEpsilon) >= err) && (q == -1)) )
            qplus = 0;
    elseif(q ~= 1)
        if( ((abs(Vz0 - cout) <= err) && (il <= 0) && (~M2)) ||
   (((abs(Vz0 - cin) <= err)) && (il >= 0)) )
            qplus = 1;
        end
    else
        qplus = q;
    end
end

%======================
%For the Hg Controller
%======================
if(p == 2)
    if((Vz0 >= cin) && (Vz0 < cout))
        pplus = 1;
    else
        pplus = p;
    end
end
```

In the code above, 'qplus' refers to the state that we're going to jump to, and 'pplus' refers to a change of controller. For example, if q is equal to zero, and 'qplus' is found to be one, then the next state of the H-Bridge will be to output $+V_{DC}$. Likewise, if the current controller variable p is equal to two - indicating that the global controller is in the loop - and 'pplus' is found to be one, then the forward controller will take over on the next update.

This is the bulk of the code for the Matlab simulations, though we have omitted the description of the behavior of the system as the solution flows, as this is covered in detail in [2].

3.1.7.2 The C Implementation on the C2000

With the logic for the controller worked out in Matlab, the work of porting the code to embedded C on the Texas Instruments DSP was a matter of fitting this logic within the bounds of the hardware modules onboard. In the sections above, namely Section 3.1.4, we discussed that the general flow of the software on the C2000 DSP is to first initialize and configure the hardware modules onboard, then to service interrupts for the boost converter and the inverter. Of chief concern in this section is the inverter ISR where we call upon a state machine that determines membership in the jump set, then informs the hardware of the appropriate action to take. The state machine is implemented using the custom state framework described above.

Because the proper operation of the controller depends it's interface with hardware, we will cover briefly the initialization of the PWM driver that we use to interface to the state machine running the hybrid algorithm. Because the hardware modules have a good deal of 'native' support for running inverter systems, adjacent PWM modules are designed to operate adjacent half-bridge modules that make up a full H-Bridge in an inverter. This means we can configure parameters like deadband, counting mode, period and phase synchronization quite simply for adjacent PWM modules. Note that in our design we utilize PWM modules one and two.

```
#define PWMDRV_Hybrid(v)
 if ( v == VDC)
 {
  (*ePWM[n]).CMPA.half.CMPA = 0 ;
  (*ePWM[n+1]).CMPA.half.CMPA = TBPRD + 1;
 }
 else if (v == ZERO_VDC){
  (*ePWM[n]).CMPA.half.CMPA = 0 ;
  (*ePWM[n+1]).CMPA.half.CMPA = 0;
 }
 else if (v == NEG_VDC)
 {
  (*ePWM[n]).CMPA.half.CMPA =  TBPRD + 1;
  (*ePWM[n+1]).CMPA.half.CMPA = 0 ;
 }
#endif
```

After having properly configured the PWM modules for operation of an H-bridge, we can use the driver function to update the state of the H-Bridge with deadband for shoot-through protection. This driver function is called after the transition logic is executed,

and the transition logic is executed only after the current state variable is constructed. Let's examine the update of the state variable on each iteration of the inverter ISR.

@todo: update this as we calculate the phase and clarify logic!

```
Vac_in = (long)((long)Vac_FB<<9)-Offset_Volt; // shift to convert to Q21
inv_ref_cur_inst = _IQ24mpy(inv_Iset, (((long) (InvSine)) << 9)) ;

inv_meas_cur_lleg1_inst=(((long) Ileg1_fb) <<12)-_IQ24(0.5);
inv_meas_cur_lleg2_inst=(((long) Ileg2_fb) <<12)-_IQ24(0.5);

inv_meas_cur_diff_inst = (inv_meas_cur_lleg1_inst -
   inv_meas_cur_lleg2_inst)<<1;

inv_meas_vol_inst =((long)((long)Vac_FB<<12)-_IQ24(0.5))<<1; // shift to
   convert to Q24


updateState(&state, inv_ref_cur_inst, inv_meas_vol_inst, phase);  //update
   the
```

Once the state variable has been updated for the current iteration of the ISR, the state variable can be passed to the state machine operating the hybrid algorithm. This is done with the following code:

```
char HBridgeTransitionFunction(HBridge self, HBridgeEvent *e, StateVariable
    state)
{
    void    *funptr = self.super_.state__;

    //Reset set membership status
    inC = false;
    inD = false;

    Vz0 = (state.current/ALPHA)^2 + (state.voltage/BETA)^2;  //The current
    solution to the system


    /**
     * Supervisory Controller
     * Determine if we need to switch controllers, depending on where the
    state variable is
     */
    if(state.controller == GLOBAL){
```

```
    if((Vz0 >= CIN) && (Vz0 <= COUT)){      // are we between the two
tracking bands? -> select forward controller
        state -> controller = FORWARD;
        //inD = true;
    }
}
else if(state.controller == FORWARD){
    if((Vz0 >= COUT) || (Vz0 <= CIN)){      // are we inside both, or
outside both tracking bands? -> select global controller
        state -> controller = GLOBAL;
    }
}

/**
 * D:
 * Determining Jump Set membership
 */

/**
 * Determine if we are in 'fast-swithcing regions' M1 or M2 so that we
may respond accordingly
 */
M1 = ((_IQabs(Vz0-cout) < ERROR) && ((state.current >= 0) &&
(state.current <= EPSILON)) && (state.voltage <= 0)) ? true:false;
M2 = ((_IQabs(Vz0 - COUT) < ERROR) && ((state.current >= -EPSILON) &&
(state.current <= 0)) && (state.voltage >= 0)) ? true:false;

/** Forward Controller Check */
if(state.controller == FORWARD)
{
    if(q != 0){

       if( (abs(Vz0-cin) <= err) && (il*q <= 0)){
           inD = 1;
       }
       else if( (abs(Vz0-cout) <= err) && (il*q >= 0)){
           inD = 1;
       }
   }
    else if (q == 0){
        if( (abs(Vz0-cin) <= err) && (q == 0)){
            inD = 1;
        }
    }
```

```
}

/** Global Controller CHeck */
if(state.controller == GLOBAL){
    if((Vz0 >= cin) && (Vz0 <= cout)){
        inD = 1;
    }
}

/**
 * If we're in the jump set, determine which state transition to make
 * Else, dont waste clock cycles!
 */
if(inD){

    /**
     * For the Hfw Controller
     */
    if(State.controller == FORWARD){
        if(State.bridgeState != NEG_VDC){
            if( ((abs(Vz0-cout) <= err) && (il >= 0) && (~M1)) ||
(((abs(Vz0-cin) <= err)) && (il <= 0)) ){
            qplus = NEG_VDC;
            }
        }
        else if ( ((M1) && (abs(il - mEpsilon) >= err) && (q == 1)) ||
((M2) && (abs(il + mEpsilon) >= err) && (q == -1)) ){
                qplus = ZERO_VDC;
            }
        else if(State.bridgeState != VDC){
            if( ((abs(Vz0 - cout) <= err) && (il <= 0) && (~M2)) ||
(((abs(Vz0 - cin) <= err)) && (il >= 0)) ){
                qplus = VDC;
            }
        }
        else{
            qplus = NO_EVENT;
        }
    }

    /**
     * For the Hg Controller
     */
    else if(State.controller == GLOBAL){
```

```
            if(Vz0 <= CIN){
                qplus = VDC;
            }
            else if(Vz0 >= COUT){
                qplus = ZERO_VDC;
            }
            else{
                qplus = NO_EVENT;
            }
        }
    }

    if(pplus != NO_EVENT){
        e->super_.signal = qplus;
    }

    return 0;
}
```

The result of the C code above is identical to that in the Matlab iteration, save for the fact that it is superfluous to know whether or not we are in the flow set because they system is not solving any differential equations, but rather it is causing real hardware to execute in real-time. Therefore, it is sufficient to determine whether or not we are in the jump set. Note that in the above code segment, there is no interface to hardware, only a change in the signal of the event 'e.' After the signal member of the event structure is set, this event gets dispatched to the current state (function), which performs the state transition. The current state of the machine is echoed in hardware via the hybrid PWM driver function. Thus, we have executed the full hybrid controller in C on our DSP.

Chapter 4

Linear Control of the Boost Converter

4.1 DC Boost Controller

In order to better understand the control mechanisms at play in a typical power inverter, we undertook a course of study to learn about the state-of-the-art in DC boost control. The the DC boost circuit is a highly non-linear mechanism - much of my initial research was into the various mathematical techniques employed to develop linear models of the circuit. Amongst them are the state space averaging technique, circuit linearization via transformation, numerical methods, and small signal analysis.

In the particular case of the DC-DC boost converter circuit shown in Fig. 4.1, the design of a digital or analog controller is complicated by the non-linear nature of the system. From linear control theory, we know that positive gain and phase margins are necessary to ensure the stability of a system in the presence of disturbances. Gain margin can be understood as a safety margin for model uncertainty, while the phase margin provides a safety factor of additional phase lag or lead to ensure system stability. In order to achieve this goal while simultaneously designing for quick response, we set out to study the design of compensators for controlling this class of PWM boost converters.

Subsequent analysis will show that the transfer function for the DC-DC converter displays unstable characteristics in the presence of a RHP pole. A Bode plot of the system reveals negative gain and phase margins. In order to rectify these unstable characteristics, controllers with proportional-integral control are employed. Compensators are specialized filters designed to provide a specific gain and phase shift at a particular frequency. Because the DC boost circuit introduces a phase lag to the system, we must

comprehend this in the design and implementation of our feedback loop. If we performed feedback on a lagging signal without compensation, wild oscillations would likely occur as constructive interference would never result in zero error.

The hybrid inverter team is interested in this type of control in order to reliably step up DC voltage from a small set of solar panels, and make it a viable source to the hybrid inverter whose output will be targeted at 120VRMS. Additionally, by changing the voltage reference, we will be able to implement a MPPT algorithm in order to harvest the most energy possible from the panels.

The rest of this paper is structured as follows: in the next section, we will cover the various methods of linearization for the DC boost plant model, as well as the fundamental choice between voltage and current mode operation. Next, we will cover some fundamental topics in control theory, and justify the design of my proportional-integral lag compensator.

Finally, we will discuss the steps needed to implement the compensator digitally, and cover some of the special considerations of implementation on a micro controller.

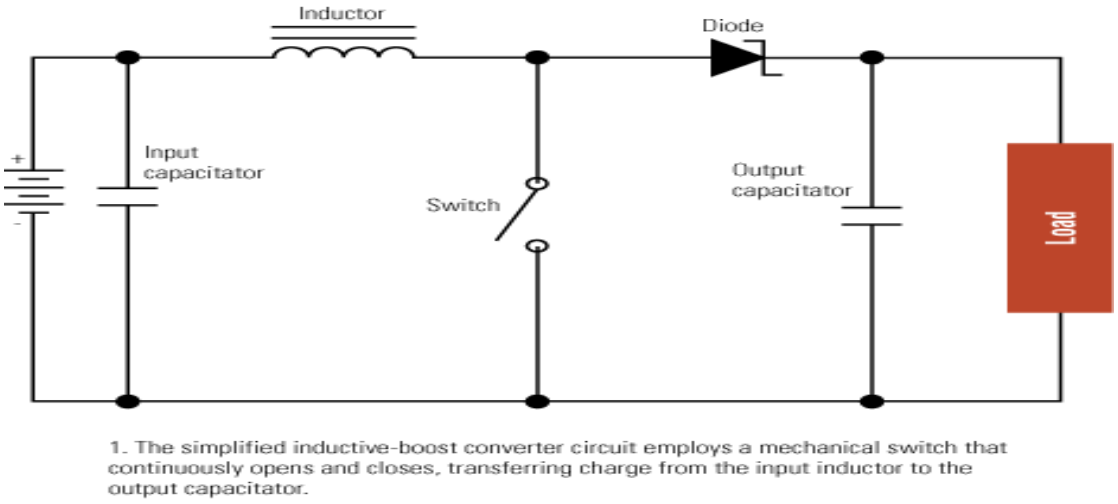

FIGURE 4.1: The Canonical DC-DC Boost Circuit

4.1.0.3 Towards a Linear Model of the DC Boost

The DC boost circuit in Fig. 4.1 has been the subject of countless dissertations dating back to the nineteen seventies. Accordingly, the landscape is a dense one. Fortunately, most researchers agree on a small signal model developed by Dr. Raymond Ridley in

the nineties. Ridley's innovation was his synergistic approach, applying both numerical, sampled-data techniques with his current control model. This model employed a linear model of PWM which allowed for the simplified analysis of all SWMPS topologies. This is the model that we've used to develop my controller, and it is found to be third-order. See Equation 4.17. See Equation. For the sake of being thorough, we will cover some alternative methods of analysis and circle-back to a discussion on Dr. Ridley's small signal and modeling approach.

4.1.0.4 Dynamic Averaging

In [10], a method for modeling the DC boost circuit with an ideal switch, an ideal transformer, and ideal current sources allows for a straightforward analysis using basic circuit analysis. After linearizing the circuit model, perturbation theory is applied to allow for the treatment of the non-linear variables as a sum of they DC and AC components. For example, the function $x(t)$ would be represented as $x(t) = X + \tilde{x}$, where X is the DC components, and $\tilde{x}$ is the AC component of the small signal representation.

The resulting linear equations for $\tilde{v}$ and $\tilde{i}$, the voltage on the output loop and the current through the input loop respectively are given by:

$$\begin{aligned} \tilde{v_{cp}}(t) &= D\tilde{v_{vp}} + V_{vp}\tilde{d} \\ \tilde{i_{vp}}(t) &= D\tilde{i_{cp}} + I_{cp}\tilde{d} \end{aligned} \tag{4.1}$$

where the subscripts vp or cp indicate the voltage and current paths respectively. In Mohan's analysis, the boost circuit is split into two loops, the input loop being the current path, the output loop being modeled as the voltage path.

Finally, the transfer function resulting from this analysis is given by:

$$\frac{\tilde{v}}{\tilde{d}} = (1 - \frac{sL_e}{R})\frac{1 + sRC}{L_eC(s^2 + s(\frac{1}{RC} + \frac{R}{L_eC}) + \frac{1}{L_eC})} \tag{4.2}$$

For a duty cycle D, effective inductance L_e, capacitance C. Note that the effective inductance goes as the inverse square of the prime duty cycle, where the prime duty cycle refers to one minus the duty.

Mohan's analysis was a useful stepping stone for understanding the basis for circuit linearization, in addition to his concise coverage of the modes of operation seen on DC boost converters. Namely, the two DC steady states either off or fully on, in addition to discontinuous and continuous conduction regimes.

4.1.0.5 State Space Averaging

Because the DC boost circuit can be viewed as occupying one of two states - off or on - we can view a linearized model of the circuit as being the average of the two states, based on the duty cycle of the PWM.

For state one where the switch is closes,

$$\begin{aligned} V_s &= V_L \\ V_s &= L\frac{di}{dt} \\ \frac{di}{dt} &= \frac{V_s}{L} \end{aligned} \tag{4.3}$$

And state two where the switch is open:

$$\begin{aligned} V_s &= V_L + V_o \\ V_s &= L\frac{di}{dt} + V_O \\ \frac{di}{dt} &= \frac{V_s}{L} - \frac{V_o}{L} \end{aligned} \tag{4.4}$$

Because the variable duty cycle gives rise to a weighted average, we define the time that the switch is closed as δT_s and the time that the switch is open as $(1-\delta)T_s$

So the averaged equations become:

$$\begin{aligned} \dot{i_L} &= \frac{1}{T_s}[\delta T_s \frac{V_s}{L} + (1-\delta)T_s(\frac{V_s}{L} - \frac{V_o}{L})] \\ \dot{i_L} &= \frac{V_s}{L} - \frac{(1-\delta)v_o}{L} \end{aligned} \tag{4.5}$$

$$\begin{aligned} \dot{v_o} &= \frac{-\delta T_s v_o}{RC} + (1-\delta)T_s(\frac{i_L}{C} - \frac{v_o}{RC}) \\ \dot{v_o} &= \frac{(1-\delta)i_L}{C} - \frac{v_o}{RC} \end{aligned} \tag{4.6}$$

With the averaged equations defined, we are free to apply the laplace transform with perturbation terms as above.

$$\begin{aligned}\delta(I_L + \hat{i_L}) &= \frac{V_s}{L} - \frac{(1 - D - \tilde{d})(V_o + \hat{v_o})}{L} \\ \delta(V_o + \hat{v_o}) &= \frac{(1 - D - \tilde{d})(I_L \hat{i_L})}{\delta t} - \frac{(V_o + \hat{v_o})}{RC}\end{aligned} \tag{4.7}$$

After expansion and elimination:

$$\begin{aligned}\hat{i_L} &= \frac{\hat{\delta} V_o}{L} - \frac{\hat{v_o}}{L} + \frac{D\hat{v_o}}{L} \\ \hat{v_o} &= \frac{(1 - D)\hat{i_L}}{C} - \frac{\delta I_L}{C} + \frac{\hat{v_o}}{RC}\end{aligned} \tag{4.8}$$

By Laplace transform we arrive at:

$$\begin{aligned}\hat{\delta} V_o &= sL\hat{i_L} - (1 - D)\hat{v_o} \\ \hat{\delta} I_L &= \frac{(1 - D)\hat{i_L}}{C} - (sC = \frac{1}{R})\hat{v_o}\end{aligned} \tag{4.9}$$

Which leads naturally to the state space representation given below:

$$\begin{bmatrix} V_0 \\ I_L \end{bmatrix} = \begin{bmatrix} sL & (1 - D) \\ (1 - D) & -(sC + \frac{1}{R}) \end{bmatrix} \begin{bmatrix} \hat{i_L} \\ \hat{v_o} \end{bmatrix} \tag{4.10}$$

Since we want $\frac{\hat{v_o}}{\hat{\delta}}$, we take the inverse of the matrix and get

$$\frac{1}{\hat{\delta}} \begin{bmatrix} V_0 \\ I_L \end{bmatrix} = \begin{bmatrix} sL & (1 - D) \\ (1 - D) & -(sC + \frac{1}{R}) \end{bmatrix}^{-1} \begin{bmatrix} \hat{i_L} \\ \hat{v_o} \end{bmatrix} \tag{4.11}$$

with the inverse matrix given as:

$$A^{-1} = \frac{1}{tf} \begin{bmatrix} sRC + 1 & R(1 - D) \\ R(1 - D) & -(sC + \frac{1}{R})) \end{bmatrix} \tag{4.12}$$

Where tf is given by:

$$s^2 RLC + sL + R(1 - D)^2 \tag{4.13}$$

So we have that

$$\frac{\hat{V_o}}{\hat{\delta}} = \frac{V_o}{(1 - D)} [\frac{-sL + R(1 - D)^2}{s^2 RLC + sL + R(1 - D)^2}] \tag{4.14}$$

Note the deviation in this result from the previous one obtained by the average dynamic model. We did not utilize this method for designing the feedback loop, but it was a useful and enlightening exercise nonetheless.

4.1.0.6 Numerical Methods in Matlab

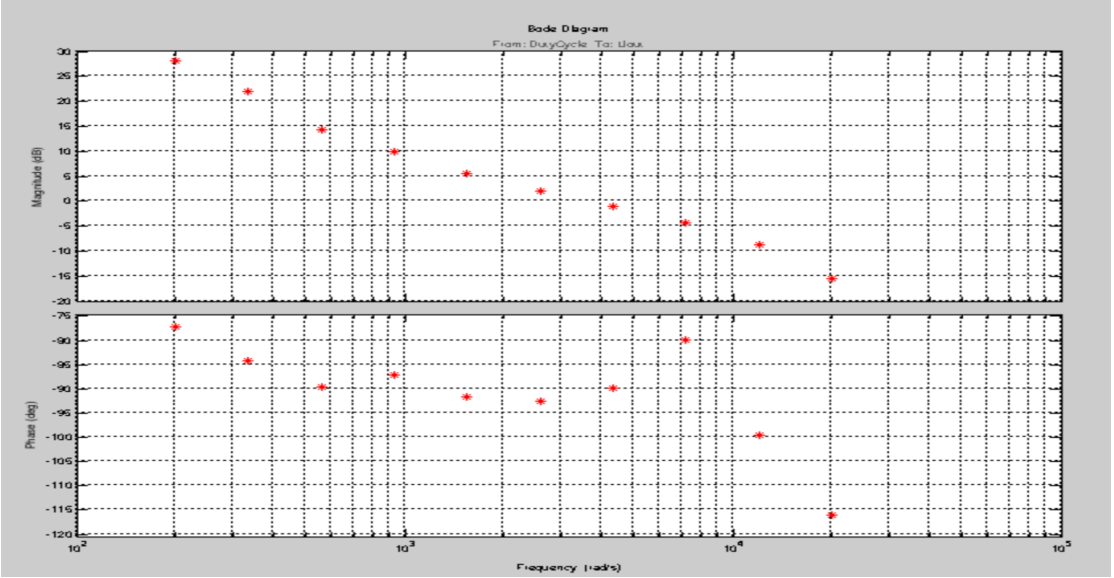

FIGURE 4.2: Sampled Data Before Estimation

One of the first methods that we used was modeling via simulink. By utilizing an off-the-shelf inverter model, we was able to ascertain the transfer function by following the steps outlined below.

We began by opening the model below. The *mdl* variable is reused, so do not omit it.

```
mdl = 'iddemo_boost_converter';
open_system(mdl);
```

The frequency response input and output points are created using the *linio* command and for this example are the outputs of the DutyCycle and Voltage Measurement blocks.

```
ios = [...
linio([mdl,'/DutyCycle'],1,'input');...
linio([mdl,'/Voltage Measurement'],
1,'output')];
```

Use the frest.Sinestream command to define the sinusoids to inject at the input point. We are interested in the frequency range 200 to 20k rad/s, and want to perturb the duty cycle by 0.03. After the following commands are entered, the Bode p[lot of the estimated transfer function is shown as above in Fig. 4.3. This series of commands is given as follows:

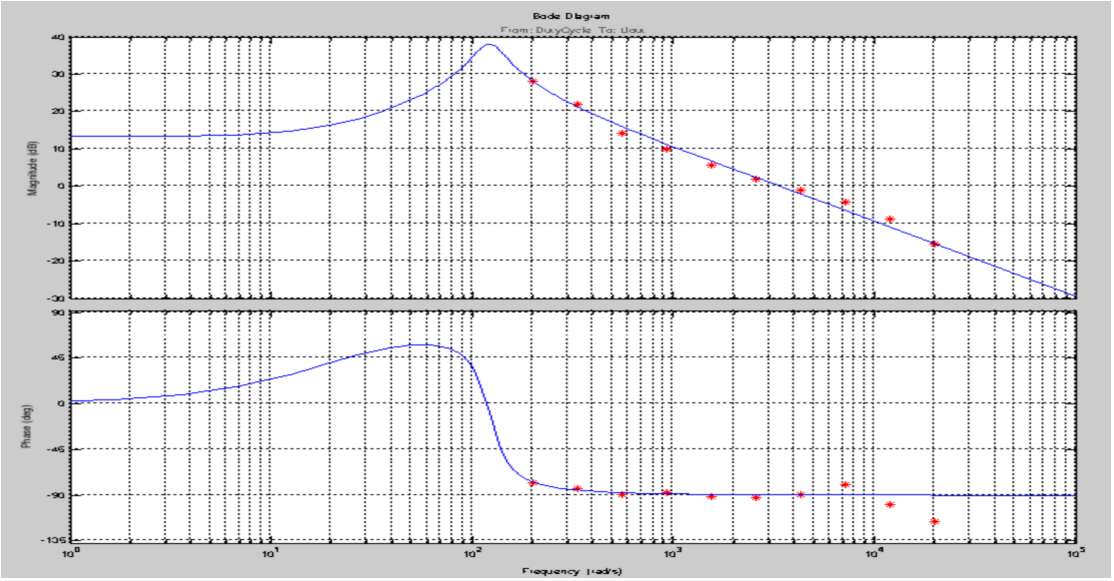

FIGURE 4.3: Sampled Data with Estimation

```
f = logspace(log10(200),
         log10(20000),10);
in = frest.Sinestream('Frequency',
         f,'Amplitude',0.03);

getSimulationTime(in)/0.02

[sysData,simlog] =
         frestimate(mdl,ios,in);
bopt               = bodeoptions;
bopt.Grid          = 'on';
bopt.PhaseMatching = 'on';
figure, bode(sysData,'*r',bopt)
```

4.1.0.7 Small Signal Analysis

After a bit of derivation, we arrive at our final point of analysis, the small signal model proposed by Dr. Ray Ridley in his PhD dissertation circa 1990. Using a combination of the technique describes above, Dr. Ridley was able to come up with a linear model of the PWM block that could be used alongside numerical methods. The resulting Transfer function has proven to be the most accurate and reliable for designers of SWMPS. Before we jump into his methods, We'd like to touch on the two fundamental methods for controlling the DC boost circuit.

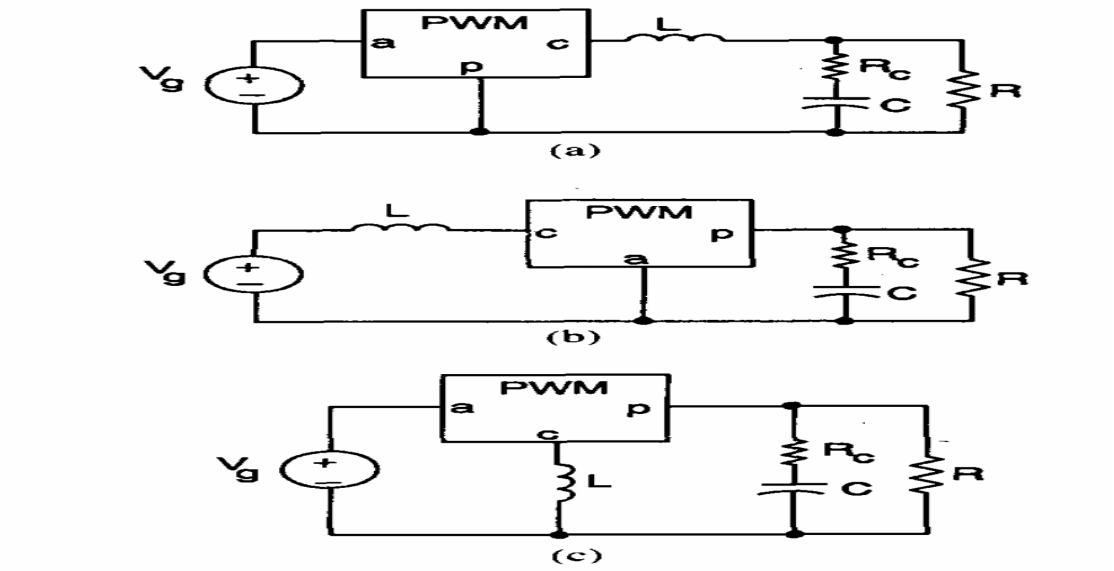

FIGURE 4.4: Linearized Model of the PWM per Ridley[1]

The first method is known as voltage mode control, and as its name implies, relies on voltage feedback from the boost circuits output to regulate itself. This technique is problematic for a few reasons. Changes in the loading condition of the circuit must be sensed as an output change first, then get corrected by the feedback loop. This delay results in slow response. Next, The output filter contributes two poles to the control loop requiring either a dominant pole for low frequency roll-off at the error amplifier, or an added zero in compensation. Lastly, compensation is complicated by the fact that the loop gain varies with the input voltage.

The preferred method of control for model switching supplies is known as current mode control. As you may have guessed, it operates by sensing current through the inductor, or through the FET switch. This method has the advantages that as its inductor current rises w/ a slope determined by $(V_in - V_o)$ the waveform will respondd immediately to line collage changes eliminating the delayed response and gain variation with changes to the input voltage. Additionally, since the error amplifier if now used to command an output current rather than an output voltage, the effect of loading is minimized and only a single pole is contributed to the feedback loop. This allows for simpler compensation and higher bandwidth over a comparable voltage mode controller.

Dr. Ridley's model is designed with current mode control in mind. We will omit a lengthy coverage of the entire 200 page dissertation, but will include the critical results - namely, that the transfer function of a buck converter is found to be:

$$f_p(s) = \frac{k[1+\frac{s}{w_z}][1-\frac{s}{w_{z,RHP}}]}{[1+\frac{s}{w_p}]} \tag{4.15}$$

Where $omega_p = \frac{2}{RC}$,$w_z = \frac{1}{R_cC}$,R_c is the ESR of the cap, and $W_{z,RHP} = \frac{R(1-D)^2}{L}$.

The difference between the average inductor current and the dc value of the sampled inductor current can cause instability for certain operating conditions. This instability is known as sub-harmonic oscillation, which occurs when the inductor ripple current does not return to its initial value by the start of next switching cycle. These oscillations are characteristic of boost circuits using current mode control. Sub-harmonic oscillation is normally characterized by observing alternating wide and narrow pulses at the switch node. This term contributes to the total transfer function and is given by:

$$f_h(s) = \frac{1}{\frac{s^2}{w_n^2} + \frac{s}{w_n Q_p} + 1} \tag{4.16}$$

We summarize their constituent expressions here:

$$\begin{aligned} m_c &= 1 + \frac{s_e}{s_n} \\ s_e &= \frac{Vpp}{Ts} \\ s_n &= \frac{Von}{L} \\ \omega_n &= \frac{\pi}{Ts} \\ Q_p &= \frac{1}{\pi(m_c D' - 1/2)} \end{aligned}$$

Where V_{on} is the inductor voltage with the switch on, and R_i is the gain from the inductor current, implying that R_i is the sense resistor.

Therefore, the total transfer function given by Ridley in [[1]]is found to be:

$$f_p(s)f_h(s) = \frac{k[1+\frac{s}{w_z}][1-\frac{s}{w_{z,RHP}}]}{[1+\frac{s}{w_p}][\frac{s^2}{w_n^2} + \frac{s}{w_n Q_p} + 1]} \tag{4.17}$$

From here, we are ready to analyze the resultant Bode plot to determine what type of compensation will be necessary for this plant model. This plot is shown below in Fig.4.5. The poles and zeros are placed in such a way that the system has enough phase

margin, and to minimize the effect of maximum phase lag due to a Right-Half plane zero.

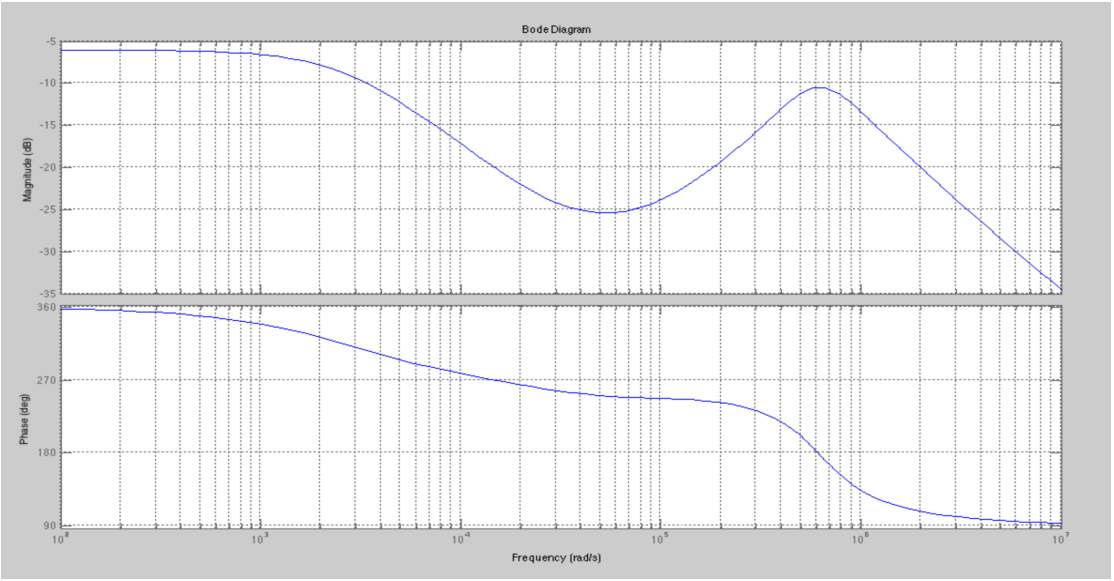

FIGURE 4.5: The total transfer function $f_p(s)f_h(s)$ for the DC boost circuit

From [10], we have that a compensator for this system in current mode control is given by

$$G_c(s) = \frac{k_c(1+\frac{s}{\omega_z})}{s(1+\frac{s}{\omega_p})} \tag{4.18}$$

Choosing our desired phase margin to be $\phi_{boost} = 60°$, we have that our key parameters are:

$$k_{boost} = tan(45° + \frac{\phi_{boost}}{2})$$
$$f_z = \frac{f_c}{k_{boost}}$$
$$f_p = f_c k_{boost}$$
$$k_c = \frac{\omega_z}{|G_{ps}(s)|_{f_c}}$$

By selecting a resonable crossover frequency f_c, we can calculate the require parameters of this expression. The resulting compensator is shown in Fig. 4.6.

Finally, calculating the close loop with compensator, we get the stabilized system shown in the third panel of Fig. 4.7.

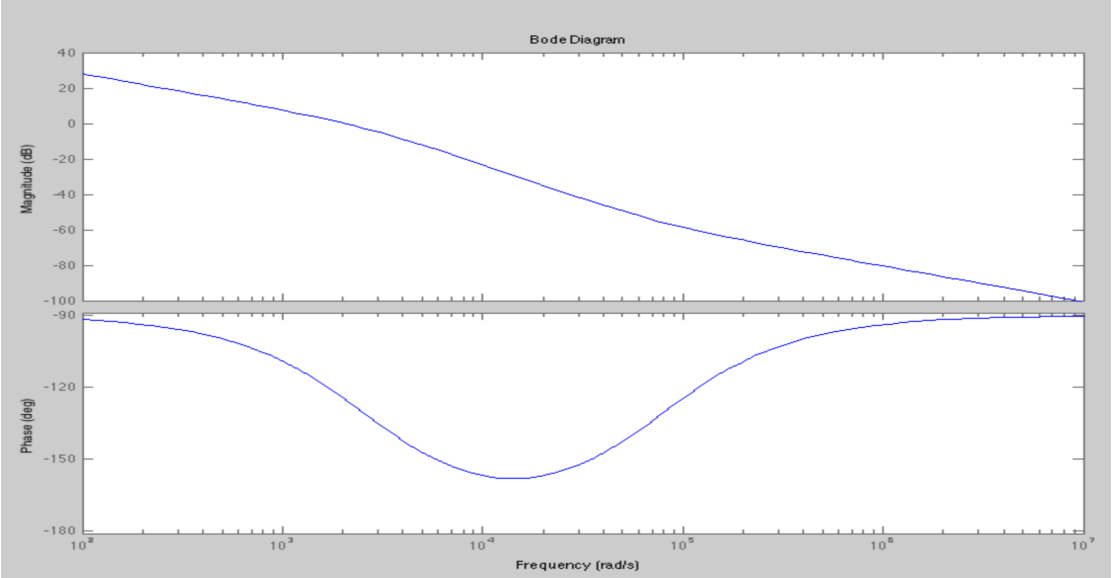

FIGURE 4.6: The lag compensator for the DC boost circuit

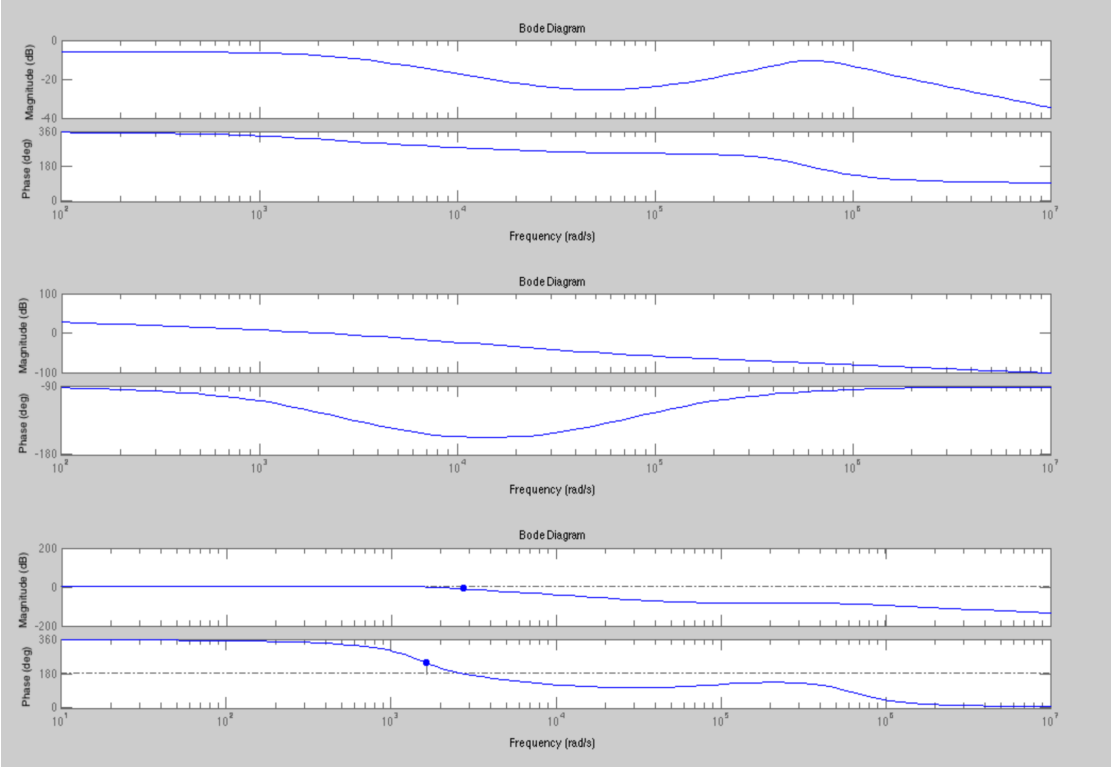

FIGURE 4.7: The Plant, Compensator, and Closed Feedback Loop Bode Plots for the DC boost circuit

For a closer inspection, see Fig. 4.8 which clearly shows the gain and phase margins make for a stable system.

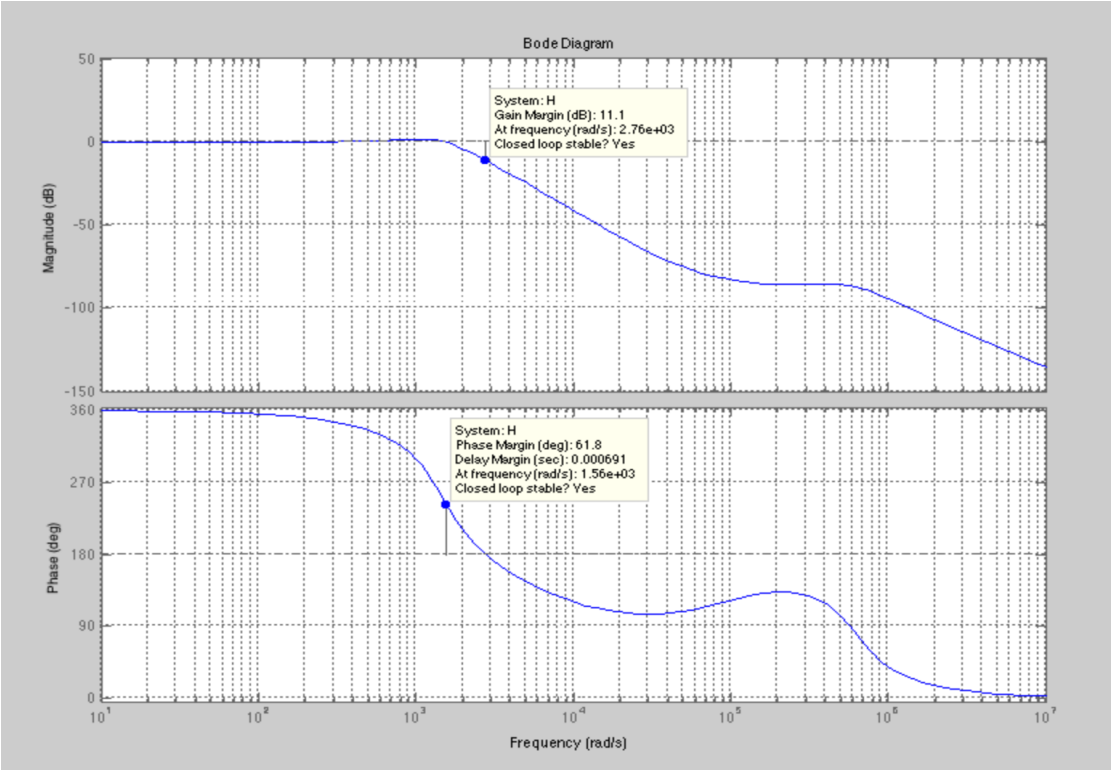

FIGURE 4.8: Bode plot of the closed loop transfer function with gain and phase margin labels

4.1.1 2P2Z

As we can see, the lag compensator was simply a special case of PI control. A 2P2Z can implement the continuous time version by way of a bilinear or Tustin transformation. The rule for the bilinear transform is that $s \leftarrow \frac{2z-1}{Tz+1}$, Performing this transformation on our controller equation results in an expression of the same form, implementable in assembly on a C2000 Piccolo micro controller by Ti. A toll recommended by TI is the Biricha tool that takes the placement of your poles and zeroes and converted them to the necessary coefficients for the micro.

4.1.2 Boost Controller Conclusion

After a fairly exhaustive review of the literature regarding the accuracy of the linearized DC boost model, and the control theory behind their employment in SWMPS, We were able to get favorable results in simulation, namely a stable system. While there are

multiple methods for linearizing the boost circuit, the most widely accepted models today utilize that of Dr. Ridley's due to its highly accurate predictions. This model was used in the design of a lag compensator. The Final phase involved the conversion of my linear model to a discrete model suitable for implementation on a micro controller.

Chapter 5

Analysis and Conclusion

5.1 PWM

5.2 Hybrid

5.3 Conclusion

The hybrid dynamical system approach for feedback control of a power inverter has shown great promise. This project is realizing this new technology as a engineering endeavor to create a viable hardware platform as a solar microinverter. This alternate control topology, compared existing techniques on the market, aims to define a new category of robust renewable inverter solutions.

Appendix A

Appendix Title Here

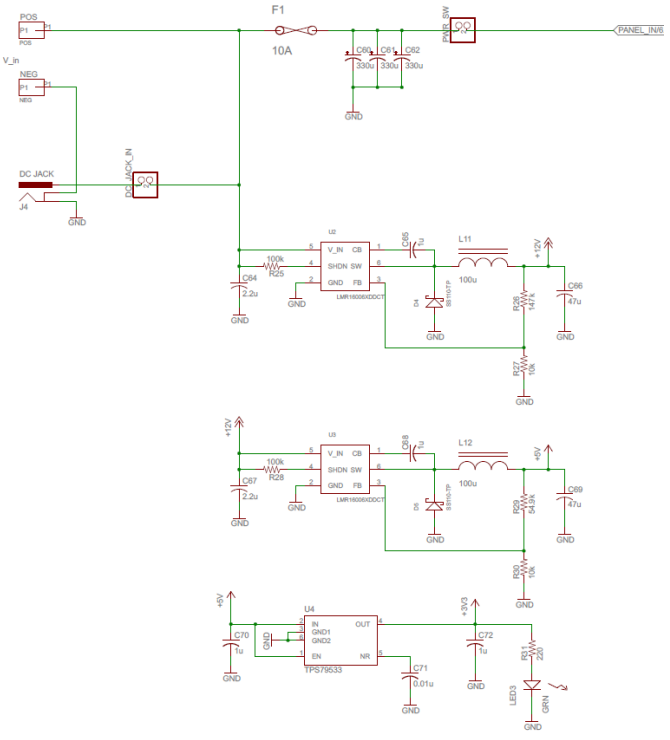

FIGURE A.1: Logic Power Supply Circuit

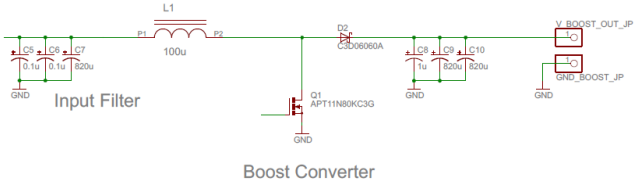

FIGURE A.2: Boost Circuit

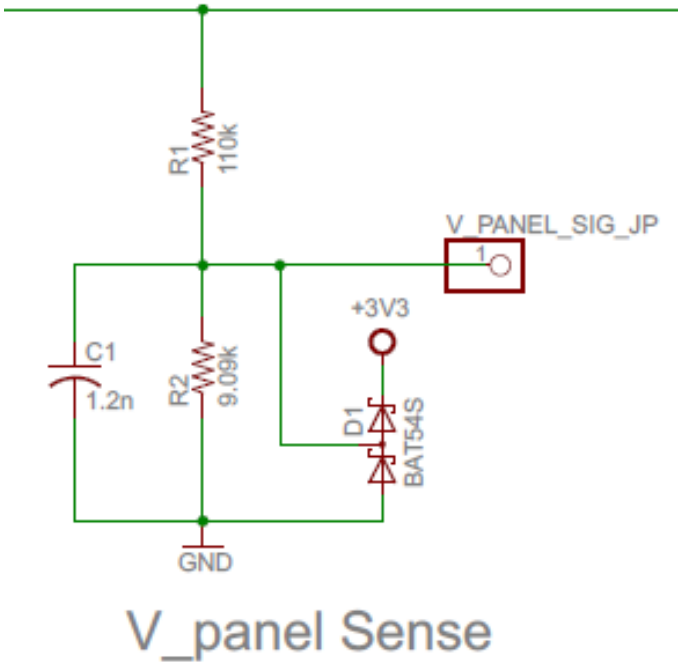

FIGURE A.3: PV Voltage Sense Circuit

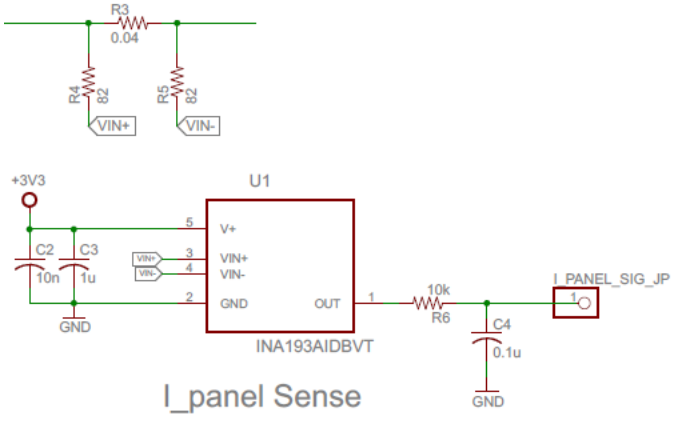

FIGURE A.4: PV Current Sense Circuit

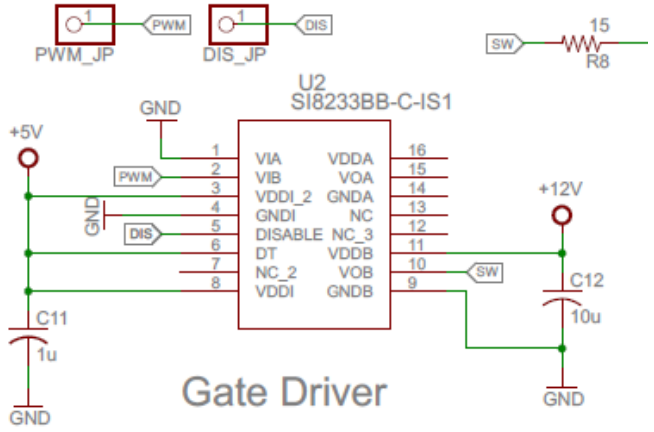

FIGURE A.5: Gate Driver Circuit

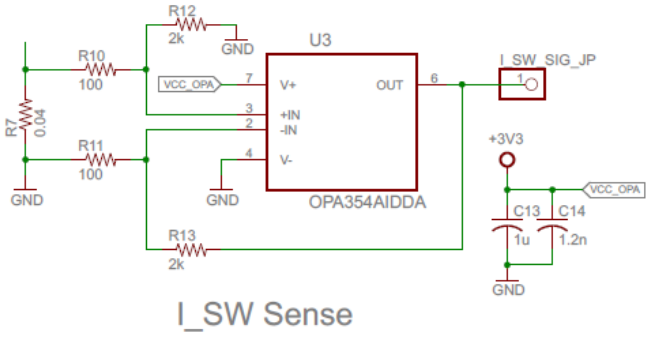

FIGURE A.6: Switch Current Sense Circuit

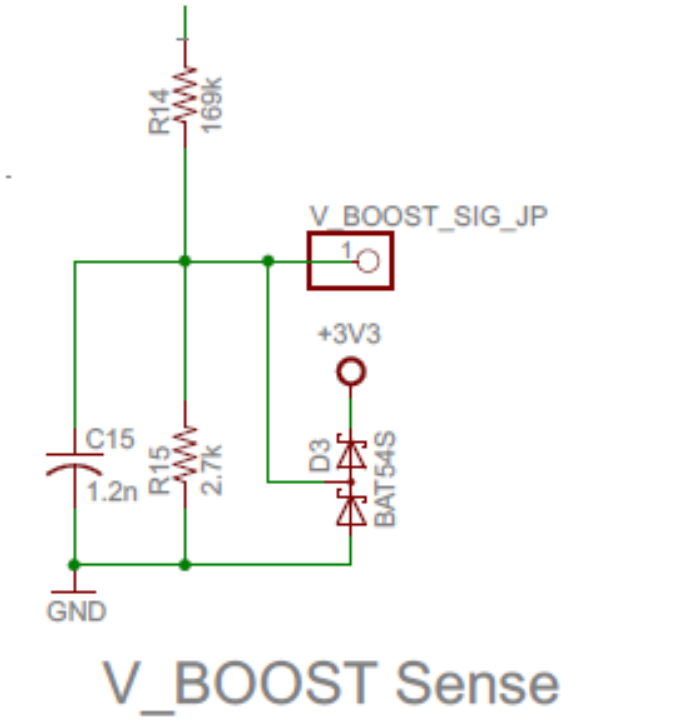

FIGURE A.7: Boosted Voltage Sense Circuit

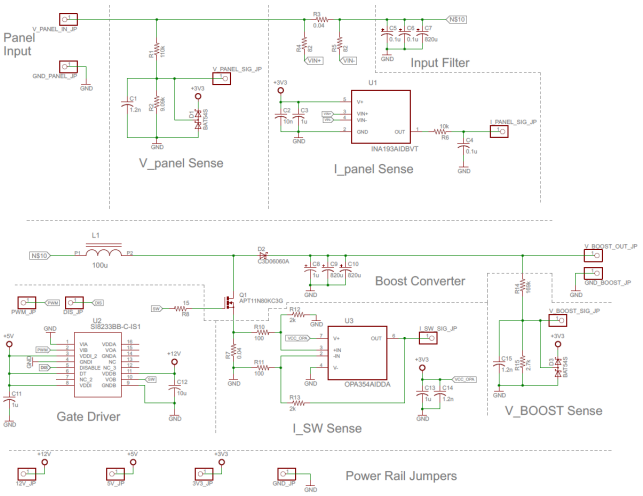

FIGURE A.8: Boost Converter Schematic

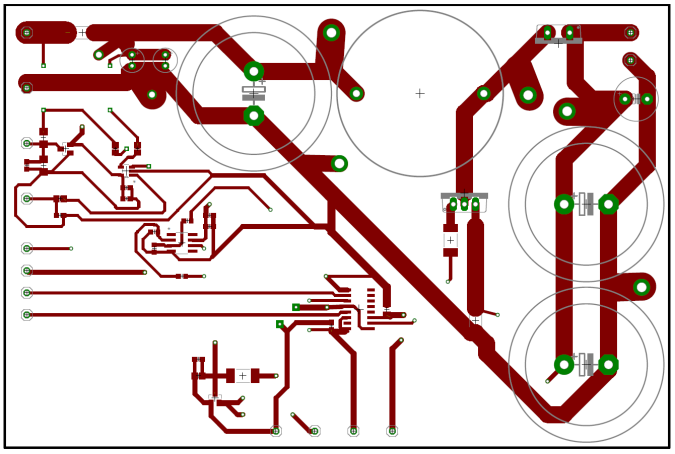

FIGURE A.9: Boost Board PCB Top

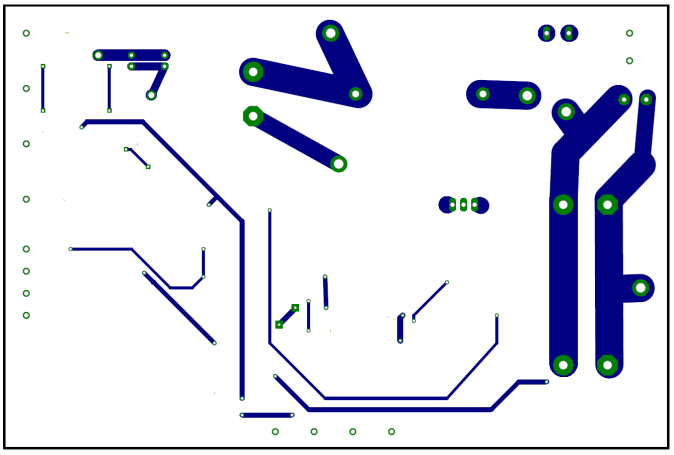

FIGURE A.10: Boost Board PCB Bottom

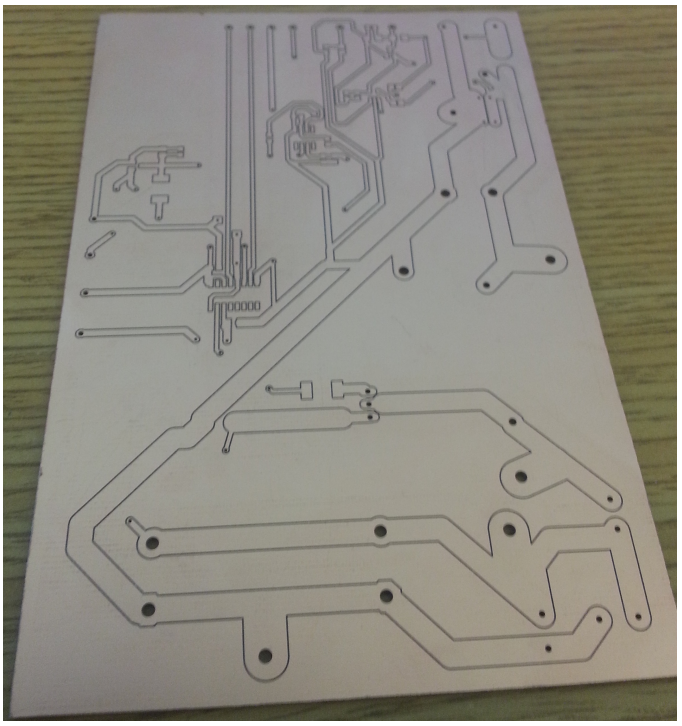

FIGURE A.11: Board Board PCB

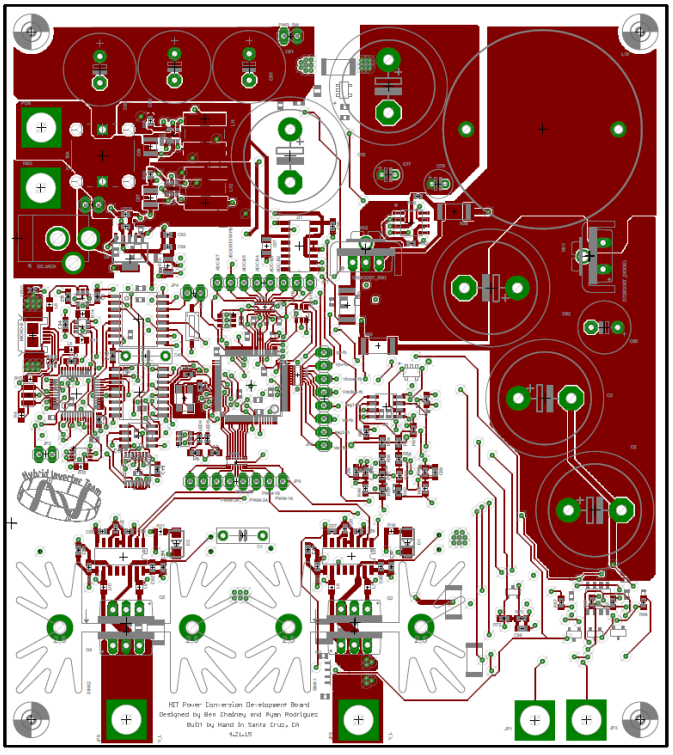

FIGURE A.12: PCB Top Layer

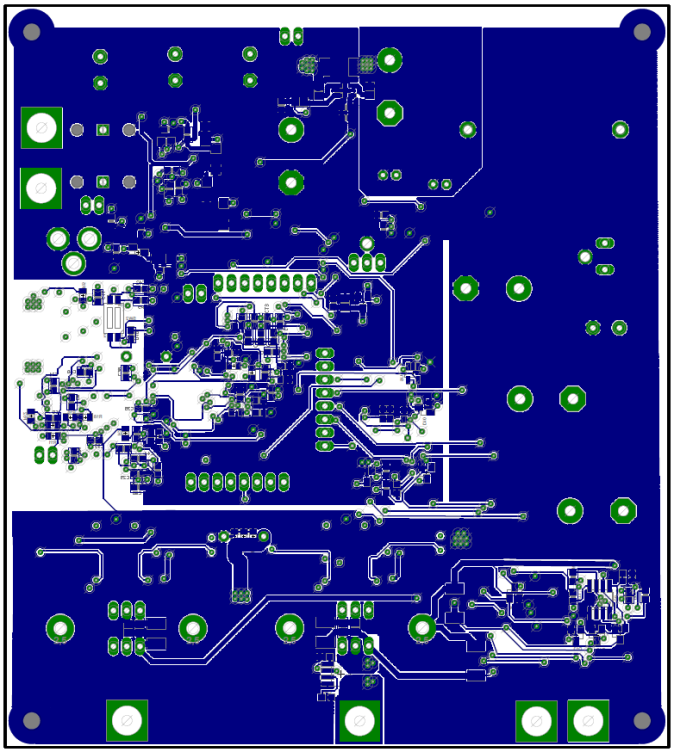

FIGURE A.13: PCB bottom

Bibliography

[1] R. Ridley. *A New Small-Signal Model for Current-Mode Control*. PhD thesis, Virginia Tech, 1990.

[2] J. Chai and R. G. Sanfelice. A robust hybrid control algorithm for a single-phase dc/ac inverter with variable input voltage. In *Proceedings of the 2014 American Control Conference*, pages 1420–1425, 2014.

[3] Microchip Technology Inc. *Microchip® Grid-Connected Solar Microinverter Reference Design Using a dsPIC Digital Signal Controller* . AN13338. 2010-2011.

[4] Sharp Electronics Corporation. *Sharp® Poly-Crystalline Silicon Photovoltaic Module with 170W Maximum Power*. SESG-170U1-607. 2007.

[5] A. Kwasinski. *University of Texas® EE462L DC-DC Boost Converter*. 2014.

[6] Cree, Inc. *CREE® Selection Guide of SiC Schottky Diode in CCM PFC Applications* . CPWR-AN05, REV A. 2012.

[7] Texas Instruments. *Texas Instruments® PV Inverter Design Using Solar Explorer Kit*. SPRABR4A. July 2013.

[8] P. Haaf and J. Harper. Understnading diode reverse recovery and its effect on switching losses, 2007.

[9] Transphorm. *Transphorm ® Designing Hard-switched Bridges with GaN*. AN0004. 2014.

[10] N. Mohan. *Power Electronics: A First Course*. 2012.