CASE WESTERN RESERVE UNIVERSITY
SCHOOL OF GRADUATE STUDIES

We hereby approve the thesis/dissertation of

Morgan Taylor McClure

__________________________

candidate for the Master of Science degree *.

(signed) Kenneth Loparo

(chair of the committee)

Francis Merat

Vira Chankong

(date) 6-25-2012

*We also certify that written approval has been obtained for any proprietary material contained therein.
# Table of Contents

List of Tables .......................................................... iii
List of Figures .......................................................... iv
Acknowledgements ....................................................... v
Abstract ................................................................. 1

1 Introduction and Background .......................................... 2
  1.1 Introduction ....................................................... 2
  1.2 System Architecture ............................................. 4
    1.2.1 Software Controller Architecture .......................... 6
  1.3 Switching Power Converters ..................................... 6
    1.3.1 Buck Converter ............................................. 8
    1.3.2 Boost Converter ........................................... 10
    1.3.3 H-Bridge Driven Transformer ............................... 12
  1.4 Reactive Power Generation ...................................... 16

2 DC-DC Conversion .................................................... 18
  2.1 DC-DC Converter Design ........................................ 18
    2.1.1 Basic Structure ............................................ 18
  2.2 Power Electronics Design Choices ............................... 19
    2.2.1 Scaling Considerations .................................... 19
    2.2.2 Power Semiconductor Selection ........................... 20
    2.2.3 Passive Component Selection ............................... 24
  2.3 Source of Losses ................................................ 29
    2.3.1 Resistive Losses .......................................... 29
    2.3.2 Gate Drive Losses ......................................... 30
    2.3.3 Control Losses ............................................ 30
## 3.6.4 High Level Controller

### 4 Conclusions and Future Work

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Future Work</td>
<td>97</td>
</tr>
<tr>
<td>4.2 Conclusions</td>
<td>98</td>
</tr>
</tbody>
</table>

### A DC-DC Schematics

### B DC-DC Verilog Source Code

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1 Top Level Design</td>
<td>107</td>
</tr>
<tr>
<td>B.2 ADC Sampling</td>
<td>108</td>
</tr>
<tr>
<td>B.3 Current Source Block</td>
<td>110</td>
</tr>
<tr>
<td>B.4 Duty Cycle Calculator</td>
<td>111</td>
</tr>
<tr>
<td>B.5 PWM Generation</td>
<td>113</td>
</tr>
<tr>
<td>B.6 Compliance Monitoring</td>
<td>116</td>
</tr>
<tr>
<td>B.7 Maximum Power Point Tracking</td>
<td>119</td>
</tr>
</tbody>
</table>

### C DC-AC Schematics

### D DC-AC Verilog Source Code

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>D.1 ADC Sampling Block</td>
<td>134</td>
</tr>
<tr>
<td>D.2 Filtering Block</td>
<td>137</td>
</tr>
<tr>
<td>D.3 Parameterized PI Loop</td>
<td>139</td>
</tr>
<tr>
<td>D.4 Differential PWM Generator</td>
<td>140</td>
</tr>
<tr>
<td>D.5 Phase Locked Loop</td>
<td>143</td>
</tr>
<tr>
<td>D.6 Boost Converter Controller</td>
<td>147</td>
</tr>
<tr>
<td>D.7 Boost PWM Generator</td>
<td>148</td>
</tr>
</tbody>
</table>
List of Tables

1.1 A short survey of commercial inverters ........................................ 4

2.1 DC-DC Converter Design Specifications ........................................ 19

2.2 Major Resistive Loss Values .......................................................... 29

3.1 DC-AC Converter Design Specifications ........................................ 66
List of Figures

1.1 Current PV array connection schemes .......................... 3
1.2 The idealized current bus architecture ........................ 5
1.3 A high level view of the major components in the system..... 7
1.4 A traditional buck converter .................................. 8
1.5 An illustration of the two switching states of a buck supply . 10
1.6 A traditional boost converter ................................. 11
1.7 An illustration of the two switching states of a boost supply . 12
1.8 An H-Bridge driven transformer ............................. 13
1.9 The switching states for sign magnitude control (only one polarity is shown) ............................................. 14
1.10 The switching states for fully differential control ............. 14

2.1 The proposed DC-DC converter topology ..................... 18
2.2 A non-isolated P-Channel drive circuit ....................... 22
2.3 A bootstrap circuit for powering an N-Channel mosfet gate . 23
2.4 An idealized mosfet switching waveform in a boost converter 32
2.5 The step response of the current controller in strictly buck mode ... 34
2.6 The step response of the current controller while changing modes from buck to boost .......................... 36
2.7 The step response of the current controller while changing modes from boost to buck ........................................ 37
2.8 Ringing at the midpoint between the two inductors ............ 39
2.9 Ringing at the midpoint of the Half-Bridge .................... 40
2.10 Ringing at the midpoint between the two inductors after the addition of an RC snubber .................................. 41
2.11 8V_{PK-PL} ringing measured across a 1cm trace 2.12 ........ 42
2.12 Probe configuration for measurement ........................ 43
2.13 Input node of the ADC while sampling
2.14 A detailed view of the second prototype layout
2.15 The controller sweeping the current setpoint demonstrating a non-linearity when $V_{In} = V_{Out}$
2.16 The controller sweeping the current setpoint with the input power supply circuitry
2.17 The controller’s measured power curve before fixing shunt configuration
2.18 The actuator effort plot for the complete system demonstrating rough linearity
2.19 A timing diagram for the PWM generation block
2.20 A timing diagram for the ADC block
2.21 A triangle wave input vs raw ADC values
2.22 A triangle wave input vs rectangular filtered ADC values
2.23 Power curve and operation point for MPPT on a resistive source
2.24 Power curve and operation point for MPPT on a solar cell
3.1 DC-AC power electronics diagram
3.2 States of the boost converter in the DC-AC inverter
3.3 States of the output H-Bridge in the DC-AC inverter
3.4 The analog signal measured at the ADC input, correlated with the SPI commands
3.5 The same analog signal with lower input impedance
3.6 ADC input structure equivalent circuit
3.7 The VAC divider with original component values
3.8 Oscillation between voltage mode and current mode controllers
3.9 Saturation based current setpoint limiting. Yellow=Output Voltage, Blue=Measured system input current, Red=Input current-Original Setpoint
3.10 Proportional Gain 8, Integral $1/256$
3.11 Too much proportional, Proportional Gain 32, Integral $1/256$
3.12 Too much integral, Proportional Gain 16, Integral $1/128$
3.13 Final gains, Proportional Gain 16, Integral $1/256$
3.14 Final gains, Proportional Gain 16, Integral $1/256$
3.15 The boost controller subjected to a rapidly changing load

vii
3.16 Sinusoidal output waveform ................................................ 82
3.17 A plot showing $\frac{d}{dt}$ of the H-Bridge versus duty cycle with 45 volts in for various values of $V_{Out}$ demonstrating uncontrollability near 0 because of duty cycle limiting ................................................ 82
3.18 A plot demonstrating the effects of duty cycle limiting in differential driven mode at the output ................................................ 83
3.19 A plot demonstrating the effects of duty cycle limiting in differently driven mode, Plot taken at half-bridge midpoints with time averaging to demonstrate average voltage ................................................ 84
3.20 A plot demonstrating the improved duty cycle limiting in differently driven mode ................................................ 85
3.21 Output voltage without any extra filtering .................................. 86
3.22 Output voltage with a single turn of ferrite .................................. 87
3.23 Output voltage with 4 turns of ferrite on input and output ............. 88
3.24 Feed-forward output current overshoot ...................................... 90
3.25 Fully differential output sine wave .......................................... 90

A.1 The top level schematic diagram .............................................. 101
A.2 Cyclone layout ................................................................. 102
A.3 Power electronics schematic .................................................. 103
A.4 Logic Power Supplies ........................................................ 104
A.5 ADC connections and safety circuitry .................................... 105

B.1 The top level FPGA design ..................................................... 107
B.2 The current source controller block ......................................... 110
B.3 The RAM based MPPT block .................................................. 119

C.1 The top level schematic diagram .............................................. 126
C.2 Cyclone layout ................................................................. 127
C.3 Cyclone JTAG and Power Pins .............................................. 128
C.4 Power electronics schematic .................................................. 129
C.5 Signal Conditioning and ADC ............................................... 130
C.6 Safety Circuitry ................................................................. 131
C.7 Differential RC Filter ........................................................ 132
C.8 Differential Serial Link ......................................................... 132
C.9 Non Isolated Gate Drive .................................................. 132
C.10 Floating Gate Drive ........................................................ 132
C.11 Logic Power Supplies ...................................................... 133
Acknowledgements

I would like to take a moment to thank the numerous people who made this thesis possible:

The generous support of the Nuimui Ji fellowship.

Ken Loparo’s support as my thesis advisor and advisor to my various student projects throughout the past 6 years has greatly influenced my development as an electrical engineer.

Frank Merat’s assistance helped me to more thoroughly understand the magnetics for this project.

Ed Burwell’s numerous hours of assistance were instrumental in keeping my prototypes from blowing up.

And to the numerous EECS students and friends who I bounced ideas off of, thank you.

Finally I would like to thank my family whose unrelenting support has gotten me through this last year.
A Modular Architecture for DC-AC Conversion

Abstract

by

MORGAN TAYLOR MCCLURE

This thesis describes the design and implementation of a modular grid-tie inverter architecture comprised of any number of DC-DC converters and a single DC-AC converter. The DC-DC converters perform maximum power point tracking autonomously on an input photovoltaic source. At the same time, the converters output as current sources allowing the paralleling of multiple devices. The DC-AC converter functions to regulate DC-Bus voltage, charge and discharge the attached battery and will buy and sell power to the grid. With the ability to operate in all four quadrants, the DC-AC converter can help improve the power factor of the grid even while consuming real power.
Chapter 1

Introduction and Background

1.1 Introduction

Over the past few decades, the nation’s electric grid has seen an increasing demand without a matching increase in capacity. With the electric grid stretched thin, the market is looking towards renewables and other forms of distributed generation in order to help provide energy for the increased demand.

While increased local generation is beneficial in terms of curbing back the peak demands seen by utility operators, it has the potential to be even more beneficial if local generation has reactive power generation capability. By supplying the grid with local reactive generation capability, it should be possible to increase reliability and offset the damaging effects of non-linear and phase-shifted loads.

One of the more promising forms of renewable energy sources is solar photovoltaic arrays. During the 2011 year, European solar installations grew by almost 70%; this growth was almost 30% of new generation capacity [1]. In 2011 the American solar industry total installed capacity increased by 977MW to 1855 MW [1], more than doubling the market share. With this explosive growth, there is increased interest in
extracting every bit of efficiency from solar installations.

The current state of the art for connecting solar photovoltaic arrays to the AC electric grid is by the use of two methods [2]. Either panels are connected in series and then to a single high-voltage inverter or they are connected in much smaller chains to so called micro-inverters (see figure 1.1).

![Current PV array connection schemes](image)

Figure 1.1: Current PV array connection schemes

The disadvantage of high voltage inverters comes from the inherent efficiency losses in the series-parallel arrangement. Because the maximum power point is heavily dependent on the illumination of the panel, it is apparent that any variation in illumination across a large array will result in efficiency losses. With variable illumination, large PV arrays can have multiple power peaks making the task of extracting maximum power much more difficult [3]. In addition, cells could be damaged by overheating when connected and not uniformly illuminated. Worst of all, there is efficiency loss when each cell is not operating at the maximum power point for it’s given illumination.

A newcomer to the scene, the micro-inverter helps to solve the uneven illumination
problem by minimizing individual array size. However, because array sizes are smaller, each installation must purchase many more converters per watt of array. This becomes a problem during design as a cost-efficiency optimization is inevitably necessary. This optimization will need to put heavy emphasis on minimizing cost in order to make the overall solution economical.

A comparison of commercially available microinverters and California Energy Commission (CEC) efficiencies is shown in table 1.1.

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Model</th>
<th>Wattage</th>
<th>CEC efficiency (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnPhase Energy</td>
<td>D380</td>
<td>190</td>
<td>95</td>
</tr>
<tr>
<td>EnPhase Energy</td>
<td>M210</td>
<td>190</td>
<td>95.5</td>
</tr>
<tr>
<td>EnPhase Energy</td>
<td>M215</td>
<td>215</td>
<td>96</td>
</tr>
<tr>
<td>Ensupra</td>
<td></td>
<td>500</td>
<td>90</td>
</tr>
<tr>
<td>Solectra</td>
<td>PVI1800</td>
<td>1800</td>
<td>92.5</td>
</tr>
<tr>
<td>Solectra</td>
<td>PVI2500</td>
<td>2500</td>
<td>93</td>
</tr>
</tbody>
</table>

Table 1.1: A short survey of commercial inverters

1.2 System Architecture

To improve on the micro-inverter, we propose a modification to the architecture with potential to improve the efficiency of the overall system. The assertion is made that efficient DC-DC conversion is more cost effective than efficient DC-AC inversion. The difficulties in DC-AC inversion come from several things including the need for much larger magnetics and the need for substantial bus capacitance in order to decouple the DC input current from the 60Hz AC output current. Based on the assumption that efficient DC-DC conversion is cheap, we propose the micro-converter architecture. The design is a modular DC-AC conversion step utilizing a shared DC Bus. A DC-DC converter is attached to a small sub-array and performs maximum power point tracking. The controller for the DC-DC converter makes the converter appear at its output to be a relatively ideal current source with a voltage compliance. Because
the converter appears to be a current source, it will allow its output to be paralleled with several identical converters. At the other end of the DC bus is a single DC-AC inverter. Since there is only a single DC-AC inverter (assumed to be the expensive per-unit-of-efficiency device) we can justify greater cost in order to gain extra efficiency.

At the input of the DC-AC inverter, a relay connects a battery across the DC bus. This battery along with the input controller help to stabilize the bus to a steady DC voltage even while being fed with current sources. The input stage of the inverter is controlled as a current source. By operating an outer control loop on the input current source it is possible to regulate the charge and discharge of the battery. This charging configuration has a benefit over others because there is no charge controller in series with the battery. If a charge controller were to be used in series then there would be some amount of pass-element losses, by controlling the battery in shunt configuration there is no extra loss. The input current-controlled stage of the inverter is a boost converter which boosts the input voltage into an extremely large (18mf) capacitor. Because of the size of the bus capacitor, it is possible for the input current

Figure 1.2: The idealized current bus architecture
source to be mostly de-coupled from the output source except at DC. The output stage of the DC-AC inverter can be modeled as an AC current source, and because of this, the inverter can easily supply reactive power.

### 1.2.1 Software Controller Architecture

The software controllers in this design are all joined in a master/slave configuration to the inverter. Over this bus, the master can send down requests for the slave to configure several operating parameters. First, the master controller can set the bus voltage compliance and the current limit. By allowing the master to send these messages, it is possible to operate the micro-converters in a non-maximum power region in the event of too much capacity for the DC-AC Inverter to regulate. In addition, the DC-AC Inverter can send a shutdown and startup command. For ease of prototyping this design, the implementation used non-isolated full duplex differential serial links. In actual implementations, this could be changed to any number of bus protocols in order to simplify the design and allow for more expandability (see figure 1.3).

### 1.3 Switching Power Converters

In order to understand the following thesis, it is first necessary to examine the three basic switching topologies that are used in the design. The basic principle of operation for most switching power converters is to use switches to manipulate a reactive component to move power. The topologies covered here are buck converters for voltage step-down, boost converters for voltage step-up, and H-Bridge driven transformers for DC-AC conversion.
Figure 1.3: A high level view of the major components in the system
1.3.1 Buck Converter

The operation of a boost converter can most simply be thought of as a pulse width modulated square wave with an LC filter afterwards. The structure is illustrated in figure 1.4.

![Figure 1.4: A traditional buck converter](image)

The operation of the buck converter is relatively simple; the switch is modulated on and off at a high frequency to output an average voltage. When the switch is on, the inductor is establishing a forward current that ramps linearly. Once the switch turns off, the inductor does not want to allow the current to change instantaneously so a build-up of electrons at the input side pushes the node below ground until the diode turns on and holds the node at ground. The inductor current then ramps down linearly until the start of the next cycle.

In all modes of operation, the voltage drops are across reactive components and therefore the device could reach a theoretical conversion efficiency of 100%. In the designs presented in this thesis, every effort is made to select components with the lowest possible parasitic resistance thereby allowing the design to approach the desired 100% efficiency. We recognize that this is impossible in practice as all components in the system have some parasitic resistances and the diode will have a forward voltage...
drop. In order to improve the performance of this design, a common improvement is the addition of a second switch in parallel with the diode in order to reduce the losses while the inductor is free-wheeling. This configuration is most commonly referred to as synchronous switching because the bottom switch and top switch operate exactly out of phase.

With synchronous switching, it is necessary to consider a third state in the switching cycle. In the event that the output current is less than half the ripple current, the output will enter discontinuous conduction mode. Discontinuous conduction implies that the inductor current will fall to zero during the off cycle. In a single switch design, this is not a problem because the inductor current will just decrease to zero and then reverse bias the diode and stop. However, in a synchronous design, it is possible to have the current continue past zero and reverse as a result of the diode being bypassed and therefore no longer preventing reverse flow. If this is allowed to occur, the converter will burn extra power by sloshing energy into and out of the reactive components. It is therefore necessary to know when the inductor current reaches zero in order to shut off the switch and prevent reverse flow. An alternative approach which is taken in this design is to simply operate without the bottom switch when the output current is too low.

1.3.1.1 System Modeling

In order to more fully understand the operation of the device and setup for designing a control algorithm, it is necessary to come up with a more formal model for the system. First, the model will make the assumption that the device is in continuous conduction for which the two states are illustrated in figure 1.5.

If the model is discretized to the operating frequency, then the model can be more simply analyzed. From these two drawings, the following differential equations can
describe the behavior of the circuit

\begin{align}
\Delta i_{L_{on}} &= \frac{V_{in} - V_{out}}{L} \quad (1.1a) \\
\Delta i_{L_{off}} &= \frac{-V_{out}}{L} \quad (1.1b) \\
\Delta i_{L} &= D\frac{V_{in} - V_{out}}{L} \quad (1.1c) \\
\Delta V_{out} &= \frac{i_{L} - \frac{V_{out}}{R_{\text{load}}}}{C_{out}} \quad (1.2)
\end{align}

1.3.2 Boost Converter

A boost converter is a switching topology that is used to raise a lower input voltage to a higher output voltage. The operation of the boost converter is slightly different than that of the buck converter but also bears some similarities. The structure is illustrated in figure 1.6.

Unlike the buck supply where the inductor can simply be thought of as a filtering element, in a boost supply, the inductor functions as an energy storage device. In the first half of the cycle, the inductor is shorted from the input to ground. According to $V = L\frac{di}{dt}$, the current in the inductor builds up based on $\frac{V_{in}}{L}$. When the switch is released, the inductor will not allow the current to change instantaneously and forward
biases the diode. During this time, the current will decrease based on $\frac{V_{out} - V_{in}}{L}$. The ratio of the on-time to off-time (duty cycle) and the input output differential will determine the average output current.

During the different parts of the switching cycle, the inductor can be thought of as alternating between storing energy and then pushing that energy to the output. For the on-cycle the current in the inductor rises linearly and the energy stored in the inductor is given by $E = \frac{1}{2}Li^2$. During the off-cycle, the inductor current falls linearly as the energy is transferred from the inductor to the output.

Just like the buck converter, the boost converter can also enter discontinuous conduction and force the inductor current to zero. This is, again, a problem with synchronous switching that, for these prototypes, is solved by simply not using the top switch when the output current was too low.

1.3.2.1 System Modeling

Similar to the boost converter the differential equations are derived for the two switching states. Just like the boost converter, we apply $V = L \frac{di}{dt}$ to the switching states
of the device to obtain the differential equations (see figure 1.7).

\[
\Delta i_{L_{on}} = \frac{V_{in}}{L} \quad (1.3a)
\]
\[
\Delta i_{L_{off}} = \frac{V_{in} - V_{out}}{L} \quad (1.3b)
\]
\[
\Delta i_L = \frac{V_{out} \times (D - 1) + V_{in}}{L} \quad (1.3c)
\]

1.3.3 H-Bridge Driven Transformer

One highly effective way to generate a pure AC sine wave out of DC is to drive a transformer with an H-Bridge in order generate a ground-referenced pure sine wave. When driving the transformer, it is necessary to filter out the high-frequency currents so that the transformer core will not increase its losses and the output will look more sinusoidal. The structure is illustrated in figure 1.8.

1.3.3.1 Sign-Magnitude System Modeling

When controlling this architecture, there are two ways to switch the device. The first method examined for control is often referred to as sign-magnitude switching because it is an apt description of the mode of operation. Simply interpreted, one half bridge is responsible for the sign of the output voltage and the other half bridge
bridge determines the magnitude. There are two ways of effectively implementing this (although both should theoretically produce the same output).

The first way to implement this design is that the sign leg and the magnitude leg swap for a polarity change. In this scenario, the sign leg is always shorted to ground and the magnitude leg pulse width modulates to control the magnitude of the output voltage.

The second way to implement sign-magnitude is to dedicate one leg for sign and one leg for magnitude. In this scenario, the sign leg alternates being tied to the high rail and ground depending on the desired polarity of the output. This scenario is more complicated because on a sign change the magnitude leg PWM signal needs to be inverted or else the sign of the system gain changes.

The switching states are illustrated in figure 1.9.

\[
\Delta i_{L_{on}} = \frac{V_{in} - V_{out}}{L} \quad \text{(1.4a)}
\]

\[
\Delta i_{L_{off}} = \frac{-V_{out}}{L} \quad \text{(1.4b)}
\]

\[
\Delta i_{L} = D\frac{V_{in} - V_{out}}{l} \quad \text{(1.4c)}
\]
\[ \Delta V_{Out} = \frac{i_L - i_T}{C_{Out}} \]  
\[ \Delta i_T = \frac{V_{Out}}{L} \]

1.3.3.2 Fully Differential System Modeling

For the case where the bridge is driven fully differentially, the polarity of the bridge is constantly flipping. The two switching states are illustrated in figure 1.10.
\[ \Delta i_{L_{On}} = \frac{V_{In} - V_{Out}}{L} \ast D \ast \frac{1}{f_{\text{switch}}} \]  
(1.7a)

\[ \Delta i_{L_{Off}} = \frac{-V_{Out} - V_{In}}{L} \ast D \ast \frac{1}{f_{\text{switch}}} \]  
(1.7b)

\[ \Delta i_L = \frac{2 \ast D \ast \frac{V_{In}}{V_{Out}} - \frac{V_{In}}{V_{Out}} - 1}{L} \]  
(1.7c)

\[ D = \frac{V_{Out}}{V_{In}} + 1 \]  
(1.7d)

\[ I_{\text{Ripple}} = \frac{(V_{In} - V_{Out})}{L} \ast \frac{V_{Out}}{V_{In}} + 1 \]  
(1.8)

\[ \Delta V_{Out} = \frac{i_L - i_T}{C_{Out}} \]  
(1.9)

\[ \Delta i_T = \frac{V_{Out}}{L} \]  
(1.10)

The downside to fully differential driving of the bridge is that the ripple current is twice as large as sign-magnitude. The strong benefit that comes with fully differential driving is that there is no discontinuity at the zero-crossing. With sign magnitude, when the setpoint changes sign, the output configuration flips. Depending on how quickly it approaches the sign change, the state change can cause the system to severely overshoot.

One of the other problems with this technique is more subtle and related to the parasitic elements in the design. In the event that there is parasitic capacitance on the output transformer from primary to secondary (essentially guaranteed), the 50KHz common mode signal will be coupled onto the output. The effects of this can be somewhat mitigated by the use of a high-frequency common mode choke on the
primary of the transformer; however, this adds an extra order of complexity to the system dynamics.

1.4 Reactive Power Generation

The frequency stability of the AC electric grid is dependent on the ability to deliver reactive power. The problem with reactive power in the grid lies in the fact that it is partially wasted. While the reactive power contributes to the current carrying capacity of the transmission lines, it is not actually consumed. Instead, reactive power is essentially borrowed and then returned to the grid by non-resistive loads. Since the model of any transmission line will include resistive losses, the reactive power flows back and forth in the line dissipating power across the resistance of the line.

Reactive power consumption is the result of connecting inductive, capacitive or non-linear loads to the AC grid. The reactive load presents a phase shift from current to voltage. This phase shift is often quantified by a metric known as power factor. Power factor for a linear load is defined as the cosine of the phase angle between the current and voltage of the line. Because power factor is a dimensionless number that is always between 0 and 1, it is usually specified as either leading or lagging to indicate whether the load is capacitive or inductive. The power factor of non-linear loads is much more difficult to parameterize although usually it is classified by the total harmonic distortion of the current waveform.

Enough current phase shift can serve to either advance or retard the frequency of the system which can have massively destabilizing consequences (and potentially lead to cascading failures). In order to stabilize the grid, the utility generally has a sophisticated distributed control system to maintain frequency stability. On the large scale, the utility generates reactive power to handle the needs of the grid. On the local level,
power factor of the line is managed by local passive power factor correction. Located at substations and large industrial facilities, shunt capacitors are switched into and out of the line in order to counteract the power factor effects of local inductive loads.

Because of the extra losses incurred in transmitting reactive power, it is beneficial to generate and manage reactive power locally. By generating the reactive power close to the reactive loads, the reactive power is more effectively utilized. While the current system generates almost all reactive power at the major power plants (augmented by the distributed passive power factor correction), it is possible to envision a distributed system of small reactive power generators helping to augment system frequency stability and ensure more efficient consumption of reactive power. In order to make this feasible as a method of producing grid stability, a sophisticated distributed control system will need to be developed in order for the utility operator to command local reactive power generation. While this is a problem not addressed in this thesis, strictly the ability to generate reactive power is discussed.

The device proposed in this thesis should be capable of generating reactive power on the local scale and thereby contribute to the overall stability of the grid.
Chapter 2

DC-DC Conversion

2.1 DC-DC Converter Design

2.1.1 Basic Structure

The main goal with the DC-DC converter was to build a uni-directional current source. For this to work over a wide range of input voltages, it was necessary to build a switching topology capable of both increasing and decreasing the input voltage. With this in mind the switching architecture in figure 2.1 was chosen.

![Figure 2.1: The proposed DC-DC converter topology](image)

A reader familiar with the most common switching topologies will note that this cir-
cuit is essentially a superposition of a classical buck and boost supply. The operation of this topology is then best analyzed as two different systems one in buck mode, another in boost mode.

2.2 Power Electronics Design Choices

The selection of components is a challenging optimization problem that has to be solved first by settling on specifications.

<table>
<thead>
<tr>
<th>Maximum Specifications:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Current (Continuous)</td>
</tr>
<tr>
<td>Output Current (Continuous)</td>
</tr>
<tr>
<td>Input Voltage</td>
</tr>
<tr>
<td>Output Voltage</td>
</tr>
<tr>
<td>Peak power</td>
</tr>
</tbody>
</table>

Table 2.1: DC-DC Converter Design Specifications

2.2.1 Scaling Considerations

The original project concept was designed to be a prototype for a much higher power system. Therefore, it is necessary throughout the design process to keep this in mind. Although the design in this case is optimized to use mosfets, in the higher power application insulated gate bipolar transistors (IGBTs) will be superior. Specifically in high voltage versions, the linear power increase of IGBTs will win out against the quadratic power increase inherent to the \(i^2R\) losses of mosfets.

The design of this prototype is intended to be fully scalable to the multi-kilowatt scale. As a result of this design requirement, the switching frequency of the converters will be strictly limited to 50KHz. The choice of 50KHz is based on the assumption that a multi-kilowatt version would be using IGBTs and bearing in mind the speed of most IGBTs, it is impractical to switch any faster without incurring heavy switching losses.
Although there are a few situations where IGBTs can be switched faster, they are almost always resonant applications with zero-current or zero-voltage switching. We recognize that this would not be possible in this application.

\subsection*{2.2.2 Power Semiconductor Selection}

When selecting the power switches for the design, the driving decision factor was conduction losses. Conduction losses are the losses incurred by power dissipation in the power switches. For this type of design, there is usually two devices that are appropriate for the design, either bipolar devices (most likely IGBTs) or MOS devices. The conduction losses in these two devices comes from different sources. In a bipolar device, the conduction losses are primarily based on the collector-emitter saturation voltage ($V_{ceSat}$). $V_{ceSat}$ is a process dependent parameter of the bipolar transistor that is mostly proportional to maximum $V_{ce}$. A quick browsing of available parts shows that standard $V_{ceSat}$ values range from as low as 1.15Volts all the way up to 8Volts. In the power class of bipolar transistor needed for this prototype, $V_{ceSat}$ can be expected to be approximately 1.5Volts. Given this value it’s possible to calculate peak conduction losses as:

$$V_{ceSat} \times I_{Max} = P_{WLoss} \quad 1.5 \times 5 = 7.5\text{Watts}$$

Since there are always two power transistors in the current path, this is a conduction loss of 15Watts or 15% of our total peak power. In addition, since losses go up linearly with increasing current, this works out to 15% loss in all operating regions. Obviously 15% loss is not going to be competitive with other devices so mosfets are the desirable alternative. Mosfets are a type of transistor that has seen an explosion in performance improvements over the past few years. It is now relatively cheap and easy to obtain mosfets
with extremely low on-resistance. The mosfets chosen for this project were the NXP PSMN8R2-80YS. This FET has a $V_{ds}$ of 80V and an $R_{dsOn}$ of only 8.5mΩ. With this 8.5mΩ of on-resistance we can calculate our peak conduction losses as:

$$I_{Max}^2 * R_{dsOn} = P_{Loss} = 0.0085^2 * 213 = 213mWatts$$

The options clearly favor the mosfets at only 0.2% peak conduction losses. It is also important to note that conduction losses will go down as the square of the current; at 50% power output, the conduction losses drop to 0.1%.

Given the choice of mosfets, it is necessary to consider the possibility of using a complimentary pair bridge. A complimentary bridge is a half-bridge with a P-Channel on top and an N-Channel on bottom. The downside of P-Channel FETs is that they are, as a general rule, two or three times more resistive than an N-Channel FETs of the same size and process. This resistivity is mostly due to the majority carriers being holes which have much lower mobility than electrons, the majority carriers in N-Channel devices. P-Channel FETs, as a general rule, also have much larger $Q_{Gate}$ which means increased power consumption in the gate drive circuitry.

The use of a P-Channel device on top is mostly a matter of convenience for the circuit designer. With a P-Channel device on top it is not necessary to have an isolated gate drive circuit. Gate drive of a P-Channel FET can be accomplished with a simple switched current source and a pull-up resistor between the drain and gate of the FET (see figure 2.2).

As a result of the high $Q_{Gate}$ in this configuration the designer will have to make a difficult design optimization. With a small pull-up resistor, the current source will need to be large and the circuit’s gate drive losses go up. The problem with a large pull-up resistor is that the FET will turn on much faster than it turns off. This problem is exacerbated by the large $Q_{Gate}$ which makes the RC time constant of the
pull-up circuitry slow. In order to get the fastest switching edge rates, it is necessary to build a symmetric gate driver capable of pushing and pulling large peak currents. A current source totem pole driver could be made to solve this problem at the expense of circuitry complexity.

With an N-Channel device on the topside of the bridge, it is necessary to bring the gate well above the input bus voltage in order to fully turn it on. Conventionally this is solved in one of a few ways. In applications where the bridge is continuously switching, it is possible to build a simple floating supply that generates its voltage from the switching of the bridge. This is often referred to as a bootstrap supply [5].

This is not a feasible solution in this design because the bridge always requires one half bridge to have its topside FET turned on continuously. If the bottom side does not turn on at least briefly, there will be nothing to keep the bootstrap supply charged.

Another excellent solution in applications with continuous switching is a gate-drive transformer. A gate-drive transformer is a specially wound transformer with one primary and two or four secondary windings. By connecting the secondaries to the gate of the FETs in opposite orientation it is possible to drive both gates with a single

Figure 2.2: A non-isolated P-Channel drive circuit
Figure 2.3: A bootstrap circuit for powering an N-Channel mosfet gate

signal. In addition, because of the single wire drive, it is possible to get closer to full 0 or 100% duty cycle. In other schemes, it is necessary to insert dead-time between switching to avoid shoot-through which inherently limits the ability to achieve duty cycles close to either extreme.

The use of a single control signal means the possibility for shoot-through in the bridge is essentially non-existent. The only problem with gate drive transformers is the requirement that you use an AC signal, which is not an option for this application.

This leads us to the solution that is used in this thesis - a pre-packaged isolated switching supply module. However, even with the isolated supply it is still necessary to get the control signal to the topside mosfet. In order to do this an opto-isolator is used. The chosen opto-isolator imposed a pretty substantial duty-cycle limit on the design. The nature of the opto-isolator is that the turn off propagation delay is substantially longer than the turn on delay. As a result of this duty cycle asymmetry, the minimum duty cycle limit is relatively severe.
2.2.3 Passive Component Selection

Given the switching frequency requirements, it is possible to calculate an optimum inductor value based on allowable current ripple.

Inductor ripple in the supply design is a challenging trade-off we had to make. Inductor ripple is to be avoided for two reasons: first, the AC-Current is what causes core heating in the inductor. Anytime the core heats up, the device is losing efficiency to hysteresis losses. In addition, current is the variable to be regulated at the output so it is important that it can be controlled with some reasonable accuracy.

In the choice of inductor, it is important to consider that this supply operates as a boost converter and a buck converter. The ripple current is very different in the two different modes of operation.

2.2.3.1 Inductor Ripple In Buck Mode

For buck mode, it is possible to calculate ripple current for the design by knowing input and output voltage.

The ripple is broken into two parts; the current rise during the on cycle and the current fall during the off cycle. At steady state, the two $\Delta I$ values are equal but opposite. From basic application of $V = L \frac{di}{dt}$ to the circuit and assuming continuous conduction, the result is:
\[ \Delta I_{T_{On}} = \frac{V_{In} - V_{Out}}{L} * T_{On} \quad (2.1a) \]
\[ \Delta I_{T_{Off}} = \frac{-V_{Out}}{L} * T_{Off} \quad (2.1b) \]
\[ \Delta I_{T_{On}} = -\Delta I_{T_{Off}} \quad (2.1c) \]
\[ \frac{1 - D}{D} = \frac{V_{In} - V_{Out}}{V_{Out}} \quad (2.1d) \]
\[ \frac{V_{In}}{V_{Out}} - 1 = \frac{1}{D} - 1 \quad (2.1e) \]
\[ \frac{V_{Out}}{V_{In}} = D \quad (2.1f) \]

From there it follows that assuming continuous conduction:

\[ I_{Ripple} = \left( \frac{V_{In} - V_{Out}}{L} \right) * \frac{V_{Out}}{V_{In}} * \frac{1}{f_{switch}} \quad (2.2) \]

With X being the normalized voltage ratio of output to input voltages, this becomes:

\[ I_{Ripple} \propto (1 - X) * X \quad (2.3) \]

Plotting or taking the derivative, \( I_{Ripple} \) hits a maximum when the output to input voltage ratio is 0.5. Therefore, the worst case ripple in buck is when the input is 35Volts and the output is 17.5Volts. In addition, from 2.1 the duty cycle is exactly 50%. If continuous conduction is assumed, the ripple current will become:

\[ I_{Ripple} = \frac{1.75 \times 10^{-4}}{L} \quad \text{or} \quad L = \frac{1.75 \times 10^{-4}}{I_{Ripple}} \]

This presents a problem when searching for an inductor. At a current rating of
5+Amps, it begins to be hard to find inductors of sufficient inductance in stock at standard suppliers. The original desire had been to shoot for 20% ripple worst case; however, this was not feasible with the readily available inductors. On the first revision of this prototype, a pair of 33uH inductors were used in series to get a total of 66uH of filter inductance. This left a total worst case ripple of 2.65Amps or 53% of peak output. This value was a slightly high for comfort so for the sake of the prototype it was decided to not use an input higher than 20Volts which would keep the ripple to less than 1.5Amps - an acceptable but large value.

2.2.3.2 Inductor Current Ripple in Boost Mode

Just like buck mode, inductor current ripple can be calculated from just the input and output voltage. Similar to buck mode apply \( V = L \frac{di}{dt} \) to the boost mode conditions and see that:

\[
\Delta I_{T_{On}} = \left( \frac{V_{In}}{L} \right) \\
\Delta I_{T_{Off}} = \left( \frac{V_{In} - V_{Out}}{L} \right) \\
\Delta I_{T_{On}} + \Delta I_{T_{Off}} = 0 \\
D = \frac{(V_{Out} - V_{In})}{V_{Out}} = 1 - \frac{V_{In}}{V_{Out}} \\
I_{Ripple} = \Delta I_{T_{On}} \ast D \ast \frac{1}{f_{\text{switch}}} \\
I_{Ripple} = \frac{V_{In} \ast (V_{In} - V_{Out})}{L \ast V_{Out}} \ast \frac{1}{f_{\text{switch}}} \\
I_{Ripple} \propto (1 - X) \ast -X
\]
Therefore current ripple is worst at maximum output voltage (18Volts) and minimum input voltage (8Volts). This means:

\[ I_{Ripple} = 9 \times 10^{-5} \]

\[ L = \frac{9 \times 10^{-5}}{I_{Ripple}} \]  

(2.11)

Given this information, boost mode will not be the limiting factor. The equation for buck mode inductance yields an inductor roughly twice as large.

### 2.2.3.3 Capacitor Ripple

In this system architecture, there will be many devices all operating at the same switching frequency. Thus it is important that they are thoroughly decoupled from each other. For this analysis, we will assume each inductor is modellable as a pair of paralleled ideal current sources: one AC, one DC (corresponding to the AC and DC components of the inductor current). If the capacitors have a low enough high frequency impedance then the AC current will travel exclusively through the capacitor and the bus will not see any of the high frequency ripple coming from the inductor.

Given that the source for this device is a solar cell, expected system dynamics are extremely slow with respect to the switching period. As a result of this, an extremely low output impedance at the switching frequency is perfectly acceptable. In order to keep the output current stable, it was decided to have a large output capacitance. While this does lower the output impedance of the device that is supposed to be a perfect current source, the addition of large bus capacitance helps to prevent high frequency dynamics from causing instabilities between the other devices on the bus. In addition, extra capacitance makes the DC-AC controller’s required bandwidth smaller.

When looking at voltage ripple on the output of a switcher, the primary cause is the effective impedance that the capacitor presents at the given switching frequency. No
capacitor is truly just a capacitor and every capacitor available has some amount of lead inductance and lead resistance. Because of this, the capacitor’s impedance is the sum of the equivalent series resistance, inductance and capacitance (ESR, ESL, and ESC) all at frequency.

One way to consider the capacitor sizing decision is to consider the inductor as a charge source. It is then necessary to ensure that in the worst case inductor ripple scenario, the output capacitor is holding several orders of magnitude more energy than the inductor. We have the equation, that for any capacitor, \( E = \frac{1}{2} C \times V^2 \) and it follows that energy goes as the square of voltage. Therefore doubling the output capacitance will cut the voltage ripple in four. Given that the energy in an inductor is \( E = \frac{1}{2} L \times i^2 \), it follows that at peak output (5Amps), the inductor is holding 825\( \mu \)Joules. Assuming 10volts out, then a capacitance of 16.5\( \mu \)F will also hold 825\( \mu \)Joules at 10Volts.

When choosing the bus capacitance, the decision was made to go with aluminum polymer electrolytic capacitors. Given our low output voltage requirements, aluminum polymer electrolytics provide an unbeatable combination of ESR, ESL and capacity. In the family of parts most appropriate for this job, the package has roughly 22m\( \Omega \) of ESR and 10nH of ESL. The largest capacitor available in this size range was 330\( \mu \)F. At 10Volts, this is 16.5mJoules or roughly 20 times the peak energy in the inductor. In order to cut down the charge based ripple even further two capacitors were used in parallel.

Given that the ratio of energy storage is so far skewed towards the capacitor, a more accurate analysis method is to look at effective impedance at frequency. The arrangement of two capacitors chosen for the prototype have an effective impedance
at 50KHz of:

\[ Z_{50KHz} = \frac{1}{2 \pi \times 50KHz \times 330\mu F} + \frac{2 \pi \times 50KHz \times 10nH + 22m\Omega}{2} = 17.4m\Omega \]

This means that the peak ripple current of 2.65Amps would only yield about 46mV of voltage ripple on the output. In this case 46mV on the 0-20V range is pretty near the noise floor of the analog to digital converter (ADC) so it’s reasonably safe to say that it will not affect other controllers. In addition, the FPGA over-samples the input data and filters it so it is highly unlikely to cause any effect (see section 2.5.3.4).

### 2.3 Source of Losses

Performance in this application can be quantified quite nicely in terms of efficiency. One of the primary design goals of this project is efficiency. Therefore, it is necessary to thoroughly analyze all sources of efficiency loss in the device in order to understand where the design can be improved.

#### 2.3.1 Resistive Losses

In all of the connections and all of the components in this design, there are parasitic resistances. Most of them are small but at peak power they add up to a non-zero part of the efficiency losses. The key components to consider:

<table>
<thead>
<tr>
<th>Resistive Sources</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_{dsOn} )</td>
<td>8.5m\Omega</td>
</tr>
<tr>
<td>( R_{Sense} )</td>
<td>10m\Omega</td>
</tr>
<tr>
<td>( R_L )</td>
<td>34m\Omega</td>
</tr>
</tbody>
</table>

**Table 2.2: Major Resistive Loss Values**

This works out to a minimum of 61m\Omega in the DC path which is a total of 1.525Watts
at 5Amps in or out.

In addition, there is a DC ESR of 11mΩ in series with the input and output capacitors. Assuming a ripple of 1.5Amps, that is another 49.5mW lost.

### 2.3.2 Gate Drive Losses

The gate drive circuitry causes losses for the whole system. Primarily, this loss is actually from two sources. The first loss to consider is the actual loss from charging and discharging a gate. The mosfets chosen have a $Q_{\text{Gate}}$ of 60nC. Assuming the gates are charged and discharged to 15Volts:

$$Q_{\text{Gate}} \times F_{\text{Switch}} \times V_{\text{Gate}} = P_{\text{WGate}} = 45\,\text{mW}$$

In this design, there are always two gates switching resulting in a total of 90mW for charging and discharging gates. The second loss to be considered is the power supply for the gate drive circuitry. The design requires three 5-15V isolated converters to power all the gate drive circuits. Each of these converters has its own set of losses that work out to roughly 70% efficiency. The last thing to consider with respect to this design is the energy to drive the opto-couplers. With the opto-couplers on this design, it is necessary to drive them with 10mA from the 3.3Volt rail. Since the 3.3Volt rail is linearly regulated from 5Volts, the loss can be calculated as if being driven from 5Volts which yields a loss of 50mW per opto-coupler or 100mW total.

### 2.3.3 Control Losses

One of the other non-trivial losses in this design is the control electronics and associated user interface. Although every effort has been made to minimize these losses, they are nonetheless there. The FPGA is a switching MOS device and therefore the losses can be improved by optimizing the algorithm implementation to run at strictly the minimum required operating frequency. This will offer some marginal
performance benefits at the expense of extra clock domains to manage inside the chip adding extra complexity to the software design.

In addition to the required losses, there are also losses incurred because of linear regulators in the control power circuitry. In order to improve efficiency, future designs should be implemented with a strictly switching supply topology.

### 2.3.4 Switching Losses

Since it takes a finite time for the voltage on the gate of the mosfet to rise and fall, the mosfet will always transition through the linear mode of operation. During this short transition time, the mosfet may dissipate hundreds of times more power than in the on state. This is essentially governed by how long it takes for the mosfet to transition out of saturation and into the linear region at which point that the transconductance is low enough to force the inductor’s voltage above the input rail. It is intuitive that the less time the mosfet is conducting with a sizable drain-source voltage the better.

During the switch on period, there are three distinct regions: The mosfet starts to turn on and starts to share the load with the diode which remains forward biased. During this time the current through the mosfet rises linearly but the voltage remains constant. Once the current through the diode reaches zero, there is a period of time $T_{rr}$ where the mosfet is also discharging the junction capacitance of the diode to reverse bias it. The power dissipation reaches a maximum at the middle of the $T_{rr}$. At this point the voltage across the mosfet falls until it is completely on (see figure 2.4)

It is difficult to actually calculate this switching loss exactly because power mosfets have highly non-linear gate charge curves. A simple linear approximation is taken to
estimate the energy lost as:

\[ \frac{I_L}{2} \cdot V_{\text{rail}} \cdot (T_{\text{rise}} + T_{\text{fall}}) \cdot f_{\text{switch}} \]

In this design that works out to a worst case 0.175 Watts or 0.19% of peak output power.

In practice, switching losses can be approximated by measuring steady state thermal rise on the switching devices or by subtracting from other efficiency losses.

### 2.4 First Prototype Design

For the first prototype, the design was based around the Atmel AtXMega256A3B. The decision to use this microcontroller was made mostly on the basis of familiarity with the device. The Xmega boasts a few key features that made it attractive for this project. The on-board ADC is able to transfer data using DMA for reduced CPU overhead. Lastly, the on-board pulse width modulation (PWM) generators are capable of dead-time insertion at the hardware level which simplifies the coding of
H-Bridge control and reduces the number of required independent timers.

2.4.1 Software Architecture

The structure of the completed software was relatively simple. There are two main tasks for the CPU to handle. The low priority main loop continuously refreshes the screen with values of all four measured variables. There is also a set of interrupt triggered routines to calculate inner control loops. The outer controller operates much slower (5Hz) and is continuously trying to converge on maximum power extraction.

2.4.1.1 Constant Current Controller

The constant current controller takes a current setpoint and then modifies the control register of the two PWM generators to affect the duty cycle of the output as well as change modes from buck to boost. The problem of mode-swapping presented a challenge. The solution attempted on this prototype was a pair of controllers one for buck mode, one for boost mode and mode-swapping logic for each. This mode swapping logic proved to be especially troublesome.

The current controller was implemented to trigger at 50KHz directly in phase with the PWM generator to avoid any double transitions. On triggering, the controller calls one of two controllers depending on a global state variable "Buck".

Each controller consists of similar basic proportional integral controller. The choice of simple proportional integral feedback (PI) control is chosen because it is exceptionally easy to tune a poorly characterized system to a usable state. For easy tuning, the "Lazy Engineer’s" method of PI tuning was used. A step function is fed into the controller input and proportional gain is turned up until some oscillation is observed. The proportional gain is then backed off and integral gain is turned up to remove
steady state error at the cost of overshoot. If using a derivative term, it would also be possible to gently increase it to dampen the overshoot from the integral. In this prototype, the analog noise was too great to allow any derivative gain without loss of stability, and therefore it was omitted. This is by no means the most effective method for controller design, however, it was running within a few hours. If this was to be productized a proper Z-Domain controller should be designed with a well defined system model.

The Lazy Engineer’s method was good enough to get the system running satisfactorily for the required system dynamics. Testing with a step function that is half of full scale in buck mode, figure 2.5 demonstrates the controller’s rise time of roughly 1.4mS. It is not a great response but is passable for further testing of the device.

![Figure 2.5: The step response of the current controller in strictly buck mode](image)

In order to handle switching modes, the active controller’s PI loop looks at its integral accumulator after each cycle. If in boost mode and the integral has hit a negative saturation limit (defined to be the value of the accumulator times integral-gain), then the controller changes the global state to buck mode. Likewise, in buck mode, if the integral accumulator hits a positive saturation limit, it changes the global state to
boost mode. After changing global state, the PI controller also resets the integral accumulator to prevent windup issues.

This method of switching modes is not workable in application. When input and output voltage are close to equal, there is a hysteretic snapping action between modes. This is demonstrated in figures 2.6 and 2.7. In these two figures, the yellow and purple traces are $V_{In}$ and $V_{Out}$ respectively. The green trace is input current measured with a Tektronix clamp on current probe. The device is configured with a 27 volt source that has 6Ω of series impedance. The output of the device is connected to a $V_{be}$ multiplier functioning as a voltage shunt. The current controller is fed a step waveform designed to push the controller between buck and boost modes. It is quite easy to see where the controller sits waiting for the integral to wind up and then upon switching modes, there is an obvious current undershoot that shows up in all traces while the buck controller integral winds up.

### 2.4.1.2 Maximum Power Point Tracking Controller

While the current controller runs in sync with the PWM at 50KHz, the Maximum Power Point Tracking (MPPT) controller runs at a rate of 5Hz. The MPPT loop looks at the power being presented at the input terminals and compares it to the previous power value. If the power has increased then the controller will increment or decrement the current setpoint based on whichever it did previously. If the power has decreased then the controller will reverse direction of incrementation. This algorithm was slow but reasonably effective when tested with a resistive load. However, it was very susceptible to noise in the input and never settled well.
Figure 2.6: The step response of the current controller while changing modes from buck to boost
Figure 2.7: The step response of the current controller while changing modes from boost to buck
2.4.2 Analog Noise

Ignoring the problems with transitioning modes, an attempt was made to understand random loss of control in the device. During testing the PI controllers would get drastically de-stabilized randomly several times a second. After overshooting and recovering it would happen again. This problem was traced to noise in the analog system. There were several sources of analog noise in this design.

2.4.2.1 Switching Noise

One of the difficult problems in switcher design is the trade-off of switching time versus radiated electromagnetic interference (EMI). With mosfets it is possible to get insanely fast switching times, but the extremely high $\frac{dv}{dt}$ at that node can capacitively couple onto signal lines. In this design, the switching rates were lengthened from 8nS to 12nS using a $1\Omega$ gate drive resistor. This was in order to help reduce switching EMI though the problem was hardly solved.

Worse still, in this design, the pair of inductors had a substantial amount of parasitic parallel capacitance. In this arrangement, the fast rise times (12nS) used in this design were exciting the resonant node formed between the two inductors. The resonant node was probed and the following two plots were taken - one at the center node of the inductors and the other at the midpoint of the H-Bridge (see figure 2.8 and 2.9).

The system was consistently resonant around 5MHz. This means the parasitic capacitance can be calculated to be 15pF from $f_{res} = \frac{1}{2\pi\sqrt{LC}}$. The nearest readily available value was 22pF. The resistance was then chosen to be 110Ω (chosen to be roughly an order of magnitude less than the filter impedance at resonance). The two components were connected in series and attached in parallel with one inductor. The result was a reasonably damped waveform at the center point (see figure 2.10).
Figure 2.8: Ringing at the midpoint between the two inductors
Figure 2.9: Ringing at the midpoint of the Half-Bridge
Figure 2.10: Ringing at the midpoint between the two inductors after the addition of an RC snubber
Another source of noise was identified to be the reverse recovery charge of the body diodes in the mosfets section 2.3.4. The reverse current surge results in large EMI spikes that radiate everywhere. A schottky diode with a much smaller voltage drop and much less junction capacitance was placed in parallel with each mosfet. By using a lower voltage drop diode, the junction capacitance of the mosfet body diode will store much less charge and therefore the largest contributor to $Q_{rr}$ is the lower voltage drop diode. This extra parallel diode calmed the analog signals down even further.

Further examination revealed that one of the biggest remaining problems was the inductance between the bridge mosfets and the bypass capacitors. In the layout process the bottom mosfet was separated from the local bypass capacitor by a 1cm trace and a via. Using both a passive probe and an active probe roughly $8V_{P_{k}-P_{k}}$ was measured across the ground trace. This noise is clearly visible correlated on analog traces throughout the system.

![Figure 2.11: 8$V_{P_{k}-P_{k}}$ ringing measured across a 1cm trace](image)

2.4.2.2 ADC Noise

The last source of noise in the system was identified to be the analog to digital converter on the AtX Mega. The input stage of the AtX Mega is a capacitive sample and hold circuit. In order to assure the highest signal to noise ratio (SNR), the inputs were driven with an op-amp to buffer the input signals. However, the buffer
Figure 2.12: Probe configuration for measurement

was separated from the microcontroller inputs by roughly 3 inches of trace. This length of trace proved to have too much inductance for the sampling capacitance of the ADC. The trace below shows the input of the ADC upon sampling:

Attaching a 10nF capacitor to ground immediately adjacent to the input pin attenuated this spiking by approximately half, although the noise was still visible while calibrating the ADC.

2.4.3 Sensing Circuitry

The AtXMega contains two internal 12 bit ADCs capable of sampling up to 1Msps. In this design it was decided to measure 4 different state variables for control and safety logic. Input and output bus voltage are measured with a simple resistive divider and low-pass filter to prevent aliasing.

In order to measure input and output current, an IC from Allegro was chosen that
does non-contact hall-effect based current sensing. One of the nice benefits to this chip is that it includes an open collector output that latches low when the current exceeds 110% of full-scale (roughly 12.5Amps).

The hall effect sensor proved to be the part that truly broke the design however. Periodically the sensor saturates and takes a long time to recover. This saturation behavior is the source of the control loop de-stabilization.

A further examination of the problem concluded that the saturation of the sensor was a result of the poor shielding on the inductors or improper layout.

2.4.4 Safety Circuitry

Inherent to the design of a DC-DC converter is the potential that a controller mistake could cause the design to go well outside its operating limits and cause permanent damage. In this design, there were a few major safety concerns for which safety circuitry was built to protect.

The first damaging condition to be avoided is shoot-through in the half-bridges. Shoot- through is the condition where for whatever reason both mosfets are on at the same time. This shorts out the rails which are backed on both sides by a fairly
substantial bus capacitance. If this condition is not quickly caught, the mosfet dies
will fail catastrophically. In addition, it is desirable to prevent an output short circuit
or uncontrolled current output.

The other major concern is an unloaded output. In the unloaded output condition,
the converter could boost the voltage uncontrollably until the output capacitors failed
or the mosfets hit their voltage limit. The output capacitors have a voltage rating of
20V. To prevent this bus from exceeding the capacitor voltage rating, a comparator
circuit is built up to trip when the bus exceeds a potentiometer settable voltage. The
output signal from the comparator is fed into the set side of an S-R latch. The reset
side of the S-R latch is fed to the microcontroller so that it can reset an over-voltage
condition when it is safe to do so.

In order to prevent these conditions, all the gate drive signals are routed through
AND gates with a gate drive enable signal. The gate drive enable signal is an open
collector bus signal with a pull-up and bypass capacitor immediately next to the AND
gate IC. Both input and output current sensing have their fault signals tied to that
line as well as the drain of a mosfet connected to the voltage safety S-R latch.

2.4.5 Design Conclusions

After finishing this design, various conclusions were reached about the design. The
first and most critical thing we found about this design is faulty layout. The local
bypass capacitors for the half bridges were isolated by far too much inductance to
make them effective. In addition, shielded inductors are a clear necessity to prevent
radiated noise. The testing of the allegro hall effect IC’s also made it clear that they
were unsuitable for the task. Lastly, it is important to have a manual drive disable
switch on a testing circuit - numerous mosfets were destroyed by glitches during
microcontroller re-programmings.
It is speculated that this microcontroller could have handled stabilizing and controlling this circuit but the analog noise was too great. With some digital filtering, this might have been feasible but the microcontroller did not have the available clock cycles to handle digital filtering.

2.5 Second Prototype Design

2.5.1 Incremental Hardware Improvements

The second prototype of the DC-DC converter was a massive incremental improvement on the first prototype’s hardware. The most notable difference was the conversion to an FPGA from the microcontroller. The FPGA offers many advantages to the algorithm design process because of its massive flexibility in the world of digital signal processing (DSP). The FPGA chosen for use was the Altera Cyclone III EP3C16. This FPGA has 15 thousand logic elements (4 input LUT plus a latch) and includes 56 dedicated 18x18 bit multiplier units [6].

The choice of an FPGA for a controller dictated the need for an external ADC for which a nice part from Analog Devices was selected. The AD7265 is a dual ADC with a 12 port input MUX (either single ended or differential). It’s capable of sampling two signals simultaneously at 1Msps. With this sample rate it is possible to reject a lot of the switching noise inherent to the system (see section 2.5.3.4).

As a result of the failures experienced with hall-effect style current measurement, it was decided to use a resistive shunt and a high gain differential amplifier to measure input and output current.

After testing the previous iteration with and without schottky diodes in parallel with the bridge mosfets, surface mount ultra low forward voltage schottky diodes were
added to the design.

Because the primary problem with the previous iteration was noise concerns, great effort was put towards minimizing it. This included switching from a pair of semi-shielded bobbin-style 33uH inductors to a single torroidal 100uH inductor. The added inductance helped to reduce the worst case inductor ripple current to a more reasonable value of 1.4Amps. The use of a toroid also meant extremely low EMI.

2.5.1.1 Layout Considerations

Before laying out this design, several resources were consulted regarding designing the tightest layout possible. In this application, the parasitic components introduced by a poor layout will result in large EMI radiating from the supply which causes large problems for analog circuitry. The book [7] was also heavily consulted for advice in laying out a low noise design.

The most important design goal of the re-design was to make the high current paths as short as possible. In the previous design, the inductance of a via and 1Cm of trace resulted in $8V_{P_k-P_k}$ of high frequency noise. All designs will have some parasitic inductance introduced in any trace and this inductance can be extremely problematic in high speed switching applications. Not only does the inductance cause voltage spikes that can be destructive to mosfets, it also acts as an antenna. Therefore it is critical to a low noise design to have the high-frequency high current paths as short as possible. By keeping the lines as short as possible, it minimizes stray inductance and therefore limits damaging voltage spikes and large EMI concerns. The inspiration for the half-bridge layout came from a piece of previous work by Ed Burwell[8]. This design used additional ceramic capacitance extremely close to the mosfets. In addition, the ground plane and power path between the large capacitance is separated by only 0.01 inches. By keeping power and return as close as possible together like this,
the enclosed loop area of the conductor (and therefore the inductance) is decreased. In figure 2.14 one can see that the high-current paths are about as short as possible.

![Image of a detailed view of the second prototype layout](image)

Figure 2.14: A detailed view of the second prototype layout

In order to ensure an ultra low impedance ground, every unused area on a layer is allocated to a ground plane. This means many areas of board actually have three layers of ground. In addition, in order to keep ground bounce to a minimum the source leads of the bottom side mosfets are tied to the main ground plane with an
array of vias.

Careful consideration was made to the routing of the shunt resistor’s sense leads. Since the sense leads go to a differential amplifier with excellent common mode rejection ratio, it is important that any noise coupled in is coupled into each equally as common mode. In order to help ensure this, the two traces are spaced 0.01 inches apart and are always parallel throughout their length. In addition, the ground fill surrounding the traces is augmented with ground vias every cm along its length. This design idea is taken from a reference in [7] stating this technique provides approximately 6dB of noise reduction.

2.5.1.2 Hardware Testing Results

When the hardware was actually tested, there was almost no noise to be measured in the analog system. The effort involved in the layout yielded excellent results in terms of measured overshoot and noise. Measured at the output of the differential amplifier, the common mode noise was difficult to see in the current shunt measurement.

2.5.2 Hardware Errors

Although the layout effectively squashed all the previous noise concerns, there were a few mistakes that needed to be corrected. In order to operate the ADC in differential mode, it needed an analog supply of 5Volts not 3.3Volts. This was easily remedied with some 30 gauge wire and some soldering. Although this fix was workable, the ADC is now supplied with a switcher feeding logic which is less than desirable. On a re-build, a ferrite bead should be used to de-couple the analog supply from some of the digital noise.
2.5.2.1 Comparator Re-Wiring

More of a problem than the supply voltage for the ADC was the wiring of the safety circuit comparator. Due to a mistake in a comparator footprint, the over-current protection circuitry would not work properly. To remedy this, some careful surgery with kapton tape and 30 gauge wire re-directed the inputs. In addition an n-channel inverter was connected to fix the logic polarity.

2.5.2.2 Sense Resistors

The more troublesome bug to track down was an oversight on the designer’s part. The logic circuitry is powered from a wide input range buck converter. In order to allow the inverter to draw power from the bus to run in low-voltage conditions, both the input and output are diode-connected to the input of the buck supply. This connection is made before the input sense resistor and therefore any current the logic draws from the input terminal is unmeasured by the ADC. If this was simply a small DC offset it would not show up as much of an error, but an interesting behavior occurs because of the negative impedance the converter presents. As a result of this unmeasured component, the converter is unable to properly find maximum power. In the figure 2.15, the converter is connected to a voltage source through a series impedance of several ohms. The controller is set to linearly sweep input current from 0 to a maximum value. The blue trace is measured input current at the input terminal using a tek current probe, the pink trace is the shunt resistor measured with a differential probe.

It is clear that the controller is properly regulating the variable correctly in a linear slope (and the analog signal chain is working properly); however this is not the apparent input current. If the controller is not regulating the net input current, it will not be able to correctly identify the maximum power point. The reason for the
Figure 2.15: The controller sweeping the current setpoint demonstrating a non-linearity when $V_{In} = V_{Out}$

discontinuity becomes more apparent with figure 2.16. This is the same plot except the blue trace is now the current through the input side to the logic supply.

The blue trace in that figure is the measured current through the diode from the input to the logic supply. The whole picture begins to make sense with these two measurements. As the input voltage drops because of increased current draw, the logic regulator draws more current to deliver more power. When the input voltage finally equals the output voltage, the output side diode turns on as current is steered through the converter before going to the logic supply. During the time while the input diode is turning off and the output diode is turning on, the current at the actual input is not rising because the same current is now just being steered through the converter but the net power consumption of the device over this range remains constant. Because of this error, the converter sees an input power curve with a discontinuity. Figure 2.17 shows the measured power curve of a voltage source with a 6Ω load from the
point of view of the converter. The x-axis is increasing current with the y-axis being measured input power.

This discontinuity will prevent the device from truly finding maximum power. To fix the issue, the input power diode is now connected on the other side of the input current shunt.

2.5.3 Software Architecture

The software for this device had to be entirely re-written because the programming language for the FPGA has to be Verilog not C. Despite the extra effort, the FPGA proved to be extremely flexible as a controller. In the FPGA logic can be built up and run in parallel, allowing the engineer to avoid the complication of serializing a very multi-tasked operation.
Figure 2.17: The controller’s measured power curve before fixing shunt configuration
The overall design of the controller was similar to the original design but the FPGA design performs better because of the processing bandwidth. The controller consists of an inner controller that modulates the H-bridge to look like a current sink to the input. An outer control loop then modulates the current controller’s setpoint to achieve maximum power point tracking.

2.5.3.1 Current Controller Block

In order to facilitate easy controller design, a Verilog parameterized PI controller was designed. The block is designed to take input and output variable widths along with saturation limits and gain settings as parameters. The block is capable of both negative and positive gains on all gain terms. This creates for an incredibly modular design allowing a thoroughly tested PI controller to be used throughout all the FPGA designs in this project. In addition to being fully parameterized, the block has an input to reset the integrator and an input to freeze the integrator.

The current controller block consists of an instantiation of the PI block with gains chosen via the Lazy-Engineer’s method.

2.5.3.2 PWM Block

One of the biggest performance increases that the FPGA provided was the ability to re-consider the PWM control scheme as a continuous actuator. The first prototype controller was exceptionally bad because of the hysteresis between modes. If the system is re-imagined as a single current controller modulating a single value. The PWM generating block can then determine whether to switch the bridge as a buck or boost converter on a cycle by cycle basis. The transition between buck and boost is so smooth because a duty cycle of 100% buck is the same thing as a duty cycle of 0% boost. This means that the system is continuous (if not differentiable) at
the crossover point. The system is non-linear by nature and before attempting this design, an experiment was carried out to examine how discontinuous the system gain is at the crossover point.

Figure 2.18 shows a sample sweep of $V_{in}$ from 5Volts to 35Volts against duty-cycle from 0-2, the Y-Axis is DeltaI per cycle and demonstrates the available actuation range. The derivative of the traces is effectively the system gain. The output voltage is a constant 13.8Volts. The key thing to note is that at either extreme of voltage in, the sudden change in derivative at a duty cycle of 1 is not too large and over the majority of the operating range the output could be approximated as linear; even more important, the region where the crossover is most likely to occur ($V_{in} \approx V_{out}$) is the most linear part of the plot.

Figure 2.18: The actuator effort plot for the complete system demonstrating rough linearity

With this in mind, a PWM generating Verilog block was written. The block was designed to be clocked at 128MHz so to minimize the discretization effects of PWM steps. The PWM block consists of a state machine with all its transitions analyzed
for safety. The only legal state transitions are A->B, B->C, C->A, C->E, E->F, F->G, G->E, G->A. Each one of these transitions is designed to prevent shoot through conditions. The outputs of the state machine are diagrammed in figure 2.19.

![Diagram of state transitions](image)

Figure 2.19: A timing diagram for the PWM generation block

### 2.5.3.3 ADC Sampling

The ADC chosen for this design was the Analog Devices AD7265. The ADC is actually two ADCs in one package, each capable of 1Msps. To take advantage of this, a Verilog block was written to capture all four variables at 500Ksps. A simplified timing diagram (figure 2.20) is presented to demonstrate the block functionality.

![Timing diagram for ADC block](image)

Figure 2.20: A timing diagram for the ADC block
2.5.3.4 Digital Filtering

One of the substantial benefits of using the high speed ADC and FPGA is that clever filtering techniques can help eliminate the effects of high frequency noise. It is expected that with large ripple current, both current and voltage measurements will be triangular and periodic with the switching frequency. The value trying to be regulated is the actual periodic average of that triangular waveform. In order to filter out the triangular ripple component a simple rectangular window averaging filter was designed. By making the window precisely 10 samples, the filtered signal should be exactly the signal of interest (the DC component). Figures 2.21 and 2.22 show the raw samples of a 50KHz triangle wave and then the filtered version.

Figure 2.21: A triangle wave input vs raw ADC values
2.5.3.5 Perturb and Measure

The first MPPT algorithm to be implemented was identical to the algorithm used in the microcontroller to track maximum power. A state machine essentially is always disturbing the setpoint up or down to perform a gradient ascent algorithm. The implementation used a fixed step size for the increment instead of also including the derivative of measured power. A fixed step size has the advantage of being simpler to implement but it suffers from a guaranteed steady state oscillation. In the worst case, the output will always be stepping between at least three values at steady state.

The pseudo code for this algorithm is summarized below:

\[
\begin{align*}
PW_{new} & \leftarrow I_n \times V_n \\
Delta & \leftarrow Delta \times \text{Sign}(PW_{new} - PW_{old}) \\
PW_{old} & \leftarrow PW_{new} \\
Setpoint & \leftarrow Setpoint + Delta
\end{align*}
\]
The implemented algorithm worked quite well on a variable resistive source but tended to settle slightly to either side of the power curve. This appears to be caused by the very gentle maximum of the resistive source. When the algorithm was tested with an actual solar cell, it reliably settled within a very small margin of maximum power.

A potentially beneficial improvement to the algorithm is to use a variable step size. Instead of incrementing by a fixed delta, the derivative can be calculated each time and used to set the next increment. Example pseudo-code:

\[
PW_{\text{new}} \leftarrow I_{\text{in}} \cdot V_{\text{in}} \\
PW_{\text{old}} \leftarrow PW_{\text{new}} \\
\text{Delta} = (PW_{\text{new}} - PW_{\text{old}}) \cdot \text{Gain} \cdot \text{Delta} \\
\text{Setpoint} \leftarrow \text{Setpoint} + \text{Delta}
\]

This method offers the benefit of potentially faster peak-finding but no guarantee of stability. If too small of a gain factor is chosen, it will never change the value due to integer truncations; however, if the gain is too large, the system will oscillate on either side of maximum power.

2.5.3.6 RAM Based MPPT

During the development of the Perturb and Measure algorithm, a Verilog block was built up to constantly map the power curve of the source load. A state machine was written to manipulate a RAM block and store data of current vs power. The RAM is a dual port RAM and the other port is always sequentially streaming the RAM data for other blocks to manipulate. Initially, this was written exclusively for the purpose of debugging but proved to be a useful tool for building an algorithm.

One of the problems that inhibits quick response time in the power measurement of the input is the large input capacitance. In order to combat this inaccuracy, the RAM block will not sample power data when the measured current is not within a small
deviation from the current setpoint. This ensures that the system is approaching steady state and helps to minimize measurement error. In addition, the RAM block always averages a new value with the old value to prevent high frequency noise from corrupting the RAM.

\[
\text{if } |I_{In} - I_{set}| \leq 10 \text{ then}
\]
\[
P_{new} \leftarrow I_{In} \times V_{In}
\]
\[
P_{old} \leftarrow \text{LookupTable}[I_{In}]
\]
\[
\text{LookupTable}[I_{In}] \leftarrow \frac{P_{old}}{2} + \frac{P_{new}}{2}
\]
\[
\text{end if}
\]

With the contents of the RAM constantly being streamed, it is possible to write decision making logic that works off of the RAM contents. Given the opportunity, a novel approach to maximum power point tracking is tested. Two Verilog blocks were written; the first block always watches the RAM contents to determine the maximum power point. The maximum power point is then fed into the current controller. The second block is switched in at startup to generate a sweep of the source and initially populate the RAM. The sweep generating block can also be switched in when major discontinuities occur in the RAM. A discontinuity in the RAM indicate stale system data and the source impedance has likely changed drastically.

The results with the RAM based method were excellent. Both a resistive load and an actual solar cell were tested and the controller had no problems quickly locating and locking onto the maximum power point. As a result of the natural dithering as the input current drifts slightly around the setpoint, it is unnecessary to put in an explicit dithering term to ensure maximum power is followed. However, an explicit oscillation term is intentionally introduced in order to speed closure on maximum power. Since data is collected constantly and there is always some dithering (if not just from ADC noise), any slight increase or decrease in power next to the setpoint
will be noticed and the controller will shift to follow it.

Figure 2.23 show a resistive source and figure 2.24 shows a solar source compared using the RAM-based MPPT algorithm. The plot shows increasing current along the X axis and power on the Y axis. The impulse on the blue trace indicates the controller setpoint.

Figure 2.23: Power curve and operation point for MPPT on a resistive source

Because of the difficulties in getting instrumentation into the field, efficiency was tested somewhat crudely. A 40Watt solar cell was taken to the roof of a campus building and fixed in place. Then, with a variable voltage dummy load, maximum power was located manually using well calibrated analog meters. The value that was read was 38.2Watts. Next the MPPT controller (running the RAM-based algorithm) was connected to the solar cell and the dummy load was connected to the output of the converter. A reading of 36Watts was observed. This works out to an overall efficiency of 94%.
2.5.4 Prototype Hardware Improvements

While the design of the prototype is nearly complete, there are still a few incremental improvements to be made. If designing this for the same power level, doubling the size of the magnetics would improve results even further. The reduced ripple current should translate to less core heating and higher efficiency. The lower ripple current will also translate to lower ESR based losses in the input and output capacitors.

While the efficiency of this design has been proven to be reasonably high (94%), the logic and gate-drive circuitry are still consuming too much power. The easiest location to grab some efficiency gains is the gate drive circuitry. When the converter is examined with a thermal camera during operation, the isolated DC-DC converters are noticeably warmer than the surrounding circuitry. A potential improvement is to go to a multiple output isolated supply for powering the gate-drive circuitry. With a coupled-magnetics multiple output topology, some of the efficiency losses can be
condensed and hopefully minimize total system losses. In addition, it makes sense to lower the gate-drive voltage. In the current design, the gates of each mosfet are driven with almost 18Volts. The excessive voltage is a result of poor light-load regulation in the isolated converters. Each mosfet would function identically well with only 12Volts of gate-drive and the gate drive power consumption could go down a little more than half.

Another improvement to be made is swapping the current opto-couplers for a lower power device. The forward current necessary to drive the LEDs in the current opto-couplers is 10mA. It should be reasonable to find a replacement device with one tenth the drive current. This change would grab another 50mW of efficiency.

The largest power consuming logic rail is the core voltage (1.2) for the FPGA. The 1.2Volt rail is being linearly regulated from 5Volts which implies a maximum possible efficiency of 24% in powering the high speed logic. Converting to a switching regulator will provide substantial efficiency gains in the logic powering circuitry.

2.6 Future Work

2.6.1 Magnetics Design

While the second prototype of the DC-DC converter produced quite satisfactory results, the magnetics were clearly still undersized at 100µH. The major benefit to moving to an much larger switching inductor is the increased controlability it offers. By reducing current ripple, the output noise will be greatly reduced and the demands on the controller are reduced because of the effective bandwidth limitation the extra inductance adds to the system. In addition, the extra inductance means higher voltage capability which is a clear necessity in order to scale up the design.
2.6.2 Re-evaluation of Switching Frequency

The original deciding factor for switching frequency was the ability to scale up the design. 50KHz was chosen as a general rule-of-thumb maximum IGBT switching frequency. However, this decision may have been somewhat ill-conceived. The benefit of the current source architecture is that many smaller sub-arrays of solar panels can be connected without losing power efficiency which implies that the majority of applications will not want much more than the 90Watts the converter is currently capable of. At this power level, mosfets will definitely be the switch of choice so it is quite possible to push the switching frequency up into the 100s of KHz. By pushing the frequency up, the current ripple will go down allowing a smaller inductor to be used. If the switching frequency is increased and the inductor is increased, not only will the current ripple decrease, but the amount of time the device spends in discontinuous mode is decreased. By minimizing discontinuous conduction operation, the overall device efficiency is improved.

2.6.3 Software Improvements

One of the important aspects of the design that is untested is the device communication architecture. In order to scale to a practical design, the physical layer needs to become some sort of bus topology. Currently the physical layer is implemented as dedicated full duplex serial links between master and slave. In order to make this feasible, software needs to be written for passing control messages back and forth. In addition, the operation of the DC-DC converter should be always subject to the master inverter’s discretion. Currently, the device will start up into maximum power point tracking immediately. In the final implementation, power up sequencing should be managed by the inverter via commands passed along the bus.
Chapter 3

DC-AC Conversion

3.1 Device Description

The primary goal with the design of the DC-AC inverter was to be capable of bi-directional power transfer from DC to AC and vice-versa. In addition, the inverter needs to be capable of supplying reactive power. In order for the device to supply reactive power it must be capable of operating as an AC current source with a large voltage compliance. Lastly, the inverter is supposed to charge and discharge a battery connected to the DC bus.

In order to meet the goals of the design, a switching architecture was chosen as follows. Coming in from the DC side the inverter will first use a boost converter to boost the DC bus from 12Volts to as much as 55Volts. The input also connects a battery to the DC bus in shunt configuration. The output of the boost converter is then connected to an H-Bridge. The H-Bridge will drive a transformer differentially through a filter in order to produce a ground-referenced sine wave (see figure 3.1).

The system design criteria are listed in table 3.1.

The control logic for the inverter will be attempting to simultaneously regulate two
outputs. First, the input converter is regulating battery current to charge or discharge the battery as desired. At the same time the H-Bridge is trying to regulate the output current to a perfect sine wave. This presents a challenge in terms of preventing one controller from interacting with the other too aggressively. To solve this, the high voltage bus capacitance in the design was chosen to store 10 times the energy in one half cycle of a 60Hz sine wave. By sizing the capacitor 10 times larger, the 60Hz voltage ripple on the bus is minimized. Assuming an output of 225VA:

\[
\frac{225VA}{120} = 1.875\text{Joules}
\]

\[
\frac{2 \times 1.875\text{Joules}}{45\text{Volts}^2} = 1.851\text{mF}
\]

Based on readily available parts, an 18.3mF 63V aluminum electrolytic capacitor was chosen for the bus capacitance.
3.1.1 System Modeling

Modeling of switching power-supplies is difficult, but becomes even more difficult when dealing with two supplies in series. In order to model this, we will analyze the system in each of the possible switch states and average them proportional to the time they are in that state and consider this as a discrete system operating at 50KHz.

In this model the two inputs are denoted: \( D_1 \) and \( D_2 \) where \( D_1 \) is the duty cycle for the boost converter and \( D_2 \) is the duty cycle for the H-Bridge. Both duty cycles are limited to be strictly between 0 and 1.

We have the following parameters: the parasitic series resistance of our inductors(\( R_{sL1} \) \( R_{sL2} \) \( R_{sT1} \)) and the parasitic on-state resistance of our mosfets: \( R_{dsOn} \)

To go about modeling this it is necessary to identify all the differential equations that define the evolution of the system in each state. For the sake of notation, use the subscripts a and b as the two states for the boost converter and c and d as the subscripts for the two H-bridge states. a is the state shorting inductor \( L_1 \) to ground, c is the state that increases the current through \( L_2 \) (see figure 3.2 and 3.3)

\[
\Delta V_{C_1} = \frac{I_{in} - i_{L_1}}{C_1} \tag{3.1}
\]

Figure 3.2: States of the boost converter in the DC-AC inverter
Figure 3.3: States of the output H-Bridge in the DC-AC inverter

\[
\Delta i_{L1a} = \frac{V_{C1} - (R_{sL1} + R_{dsOn}) i_{L1}}{L_1}
\]

(3.2a)

\[
\Delta i_{L1b} = \frac{V_{C1} - (R_{sL1} + R_{dsOn}) i_{L1} - V_{C2}}{L_1}
\]

(3.2b)

\[
\Delta i_L = \frac{V_{C1} - (R_{sL1} + R_{dsOn}) i_{L1}}{L_1} D_1 + \frac{V_{C1} - (R_{sL1} + R_{dsOn}) i_{L1} - V_{C2}}{L_1} (1 - D_1)
\]

(3.2c)

\[
\Delta V_{C2ac} = -\frac{i_{L2}}{C_2}
\]

(3.3a)

\[
\Delta V_{C2bc} = \frac{i_{L1} - i_{L2}}{C_2}
\]

(3.3b)

\[
\Delta V_{C2ad} = 0
\]

(3.3c)

\[
\Delta V_{C2bd} = \frac{i_{L1}}{C_2}
\]

(3.3d)

\[
\Delta V_{C2} = \frac{i_{L1} (1 - D_1) - i_{L2} D_2}{C_2}
\]

(3.3e)
\[ \Delta i_{L2c} = \frac{V_{C2} - (R_{sL2} + 2 * R_{dsOn}) i_{L2} - V_{C3}}{L_2} \]  
(3.4a)

\[ \Delta i_{L2d} = \frac{- (R_{sL2} + 2 * R_{dsOn}) i_{L2} - V_{C3}}{L_2} \]  
(3.4b)

\[ \Delta i_{L2} = \frac{V_{C2} - (R_{sL2} + 2 * R_{dsOn}) i_{L2} - V_{C3}}{L_2} D_2 + \frac{- (R_{sL2} + 2 * R_{dsOn}) i_{L2} - V_{C3}}{L_2} (1 - D_2) \]  
(3.4c)

\[ \Delta V_{C3} = \frac{i_{L2} - i_{T1}}{C_3} \]  
(3.5)

\[ \Delta i_{T1} = \frac{V_{C4}}{T_1} \]  
(3.6)

When combining these different state-evolution equations by multiplying by the duty cycle, the result is an averaged model that is valid at exactly 50KHz.

From these equations, the only term that couples the two systems is \( V_{C2} \). It can be seen that, with a sufficiently large \( C_2 \), the two systems become mostly decoupled. This makes sense intuitively if the extreme case of an infinite capacitance is considered - from \( I = C \frac{dV}{dt} \) an infinite capacitance can never have a voltage induced on it without infinite current. Given that the system is decoupled by the choice of \( C_2 \), analysis can proceed based on the original derivations in sections 1.3.3 and 1.3.2.
3.2 Design Decisions

When designing the DC-AC Converter, many lessons were taken from the DC-DC converter design. Just like the DC-DC converter, the layout of this device was especially critical in order to reduce EMI related problems. Most of the high-current layout was inspired by the layout of the DC-DC converter. All differential sense wires were routed completely parallel with ground pour and ground vias on all sides. In addition, the switching topology is constructed of exclusively half-bridges so the previously tested low parasitic half-bridge layout was able to be copied directly from the DC-DC design.

One of the design decisions that should be fixed was the choice to design the output filter to be asymmetric. The inductor in series with the transformer should be split to either side in order to make a symmetric filter. Although the asymmetric design does not appear to have problems, it makes for different amounts of noise on the positive and negative half cycles.

When the hardware was designed, it was decided that it would drive the output in a sign-magnitude scheme. However, during testing, fully differential driving was found to be more stable. The choice of driving method should be re-evaluated for the next revision and the magnetics should be sized for the extra ripple current that fully differential drive adds.

3.3 Hardware Bugs

When the time came to test the performance of the prototype, several errors appeared. The first problem to show up was the diode selection. The schottky diodes originally chosen for the design have a reverse breakdown voltage of 60Volts. Consequently as soon as the supply was brought anywhere close to 60Volts, diodes would blow-up,
thus shorting the bridge. The diodes were replaced with 200Volt schottkys and the problem disappeared.

Another problem discovered early in the debugging process was a misinterpretation of the differential mode on the ADC. The range pin on both ADCs was hardwired to "1". This was appropriate for some of the readings but on the differential reading of $V_{AC}$ the input range was losing one effective bit of resolution. In order to solve this the range pin was de-soldered and wired to a spare FPGA pin.

After the ADC range pin was repaired, it became apparent that there was a large amount of noise on the $V_{AC}$ measurement. From figure 3.4 it is apparent that the noise is related to the sampling input switching from track into hold. The input structure of the ADC is shown in figure 3.6. Every sampling interval, the sampling capacitor is switched in and out. This puts an AC load on the external circuitry resulting in noise at the input. Since the load the ADC presents is almost entirely AC, the most critical factor in designing the input conditioning is AC impedance. However DC resistance will introduce an extra noise component to the measured signal. Looking at figure 3.7 it can be seen that the initial design had a DC input impedance of roughly $5K\Omega$. More importantly, it had a single-ended AC impedance of $10K\Omega$. Although the differential AC impedance should be low because of the differential capacitor, any external resistor mismatch will result in reasonable differential error. In order to combat this, the magnitude of the resistive divider was divided by 10 in order to get the DC input impedance down to $500\Omega$. In addition, the differential capacitor was pushed up an order of magnitude to $0.1\mu F$ and a $0.1\mu F$ capacitor was added from the ADC’s negative input terminal to ground in order to lower the AC impedance. The result was dramatic (though still not perfect) and is visible in figure 3.5.

In order to improve on this, it is prudent to use op-amp buffered inputs on the next revision.
Figure 3.4: The analog signal measured at the ADC input, correlated with the SPI commands

Figure 3.5: The same analog signal with lower input impedance
Figure 3.6: ADC input structure equivalent circuit

Figure 3.7: The VAC divider with original component values
The final major mistake in the prototype design was the magnetic selection. During ripple current calculations, a power of ten error was made and 50uH was deemed suitable for the output filter inductor. Because the inverter is designed for delivering reactive power, it is hard to characterize worst case ripple (due to rarely operating in steady state) but an easy to calculate outer bound is simply calculating $\frac{di}{dt}$ from $V = L \frac{di}{dt}$ at 50KHz assuming a 40Volt rail.

$$\frac{40Volts}{50\mu H} = \frac{di}{dt} = \frac{800,000 \text{ Amps}}{S}$$ \hfill (3.7a)

$$\frac{di}{dt} / 50KHz = 16\text{ Amps}$$ \hfill (3.7b)

Figure 3.7 clearly shows that 50$\mu H$ is not enough reactance to reasonably manage the output stage. A replacement inductor was wound with $\approx 1mH$ of inductance. This results in a much more manageable bounding $\frac{di}{dt}$ limit of 0.8Amps per cycle.

3.4 Standalone Inverter Testing

To simplify the bring-up process, the two switching supplies were tested separately. The first step was controlling the boost converter as a current sink. Initial testing was done similar to the DC-DC converter using a $V_{be}$ multiplier as a shunt load.

3.4.1 Boost Controller Testing

The boost controller used the parameterized PI block from the DC-DC converter in order to tune the step response of the current waveform. After the current controller was operational, the more challenging problem was designing the voltage limiting controller. In the event that the output stage is not delivering as much power as the input stage is pulling in, the voltage on the high voltage DC bus will start to rise. At
some point the voltage will get too high and risks damaging the power electronics or the bus capacitor. In order to prevent potential damage, a voltage limiting controller is needed as well.

The first attempt to solve this utilized two separate PI controllers, one tuned as a voltage controller and one as a current controller. When the bus voltage surpasses the pre-set maximum voltage, the controller switches to the constant voltage controller and stays in that mode until input current exceeds the setpoint. The results were extremely unsatisfactory (figure 3.8) and the control was very poor at stabilizing and handling the mode transition. Several methods were attempted to solve the mode switching. First, an attempt was made at resetting the integral upon each mode switch but the time for the integral term to wind up resulted in lots of overshoot when switching modes. A second attempt was made where the integral term was frozen during the mode transition. This freezing resulted in less overshoot but the abrupt state change was nonetheless still a problem.

It is clear from the attempts with both the DC-DC converter and the input stage of the DC-AC inverter that abrupt changing of the controller will rarely work so a different approach is taken. Instead of attempting to swap controllers, a controller would modulate the current setpoint to limit voltage. The first attempt to do this worked reasonably well. A Verilog block was written as a saturation control and the current setpoint is passed through it. The upper limit of the setpoint was controlled by an instance of the modular PI controller feeding back on voltage. The integral term of the voltage PI controller saturates to the limit during normal operation. This in turn made the saturation control block pass the current setpoint unmodified. The results of the controller being fed a current step function that drives the voltage over the limit using this controller is shown in figure 3.9.

The drawback to this methodology is that the voltage controller is unable to imme-
Figure 3.8: Oscillation between voltage mode and current mode controllers

Figure 3.9: Saturation based current setpoint limiting. Yellow=Output Voltage, Blue=Measured system input current, Red=Input current-Original Setpoint
diately affect the output of the system when it crosses the voltage threshold. Since
the controller depends on a saturation block and the effect of integral windup, the
controller also takes time to remove the integral term. For example, if the input
setpoint was $\frac{1}{8}$ of full-scale and the output suddenly crosses the voltage threshold,
the integral will start to wind down. However, the winding down takes some time
and will not begin to affect the output until it has gone below $\frac{1}{8}$ of full-scale. The
resulting non-linearity can be seen in figure 3.9. The controller overshoots the set-
point and then after a delay abruptly changes direction and then settles as the voltage
controller takes over. This is simply a bad result of using a saturation based control
that will be difficult to avoid.

While the result of the saturation based controller was stable and functional, the
overshoot and sudden abrupt change is undesirable and might have difficulty with
disturbance rejection. In order to improve on this result, a slightly altered approach
was taken. Instead of a saturation based limiter, a fractional multiplication approach
was used. The saturation block is replaced with a gain block capable of multiplying
its input by any number from 0 to 1. The voltage controller was changed to output a
fractional number from 0 to 1 and by feeding that into the modulator and multiply-
ing the current setpoint the voltage controller is able to immediately respond to any
excursion above the voltage limit. Unlike the previous design, the voltage controller
immediately starts taking effect regardless of the input setpoint. This should drasti-
cally reduce any non-linear interactions from the saturation based design that could
de-stabilize the controller. The results of the controller tuning efforts are shown in
the following figures: initial parameters figure 3.10, with too much proportional gain
figure 3.11, too much integral figure 3.12, and the final choice figure 3.13 and up close
figure 3.14.

The result of the multiplying controller was quite satisfactory. In order to also demon-
strate disturbance rejection ability, a test was conducted where half of the resistive
Figure 3.10: Proportional Gain 8, Integral $\frac{1}{256}$

Figure 3.11: Too much proportional, Proportional Gain 32, Integral $\frac{1}{256}$
Figure 3.12: Too much integral, Proportional Gain 16, Integral $\frac{1}{128}$

Figure 3.13: Final gains, Proportional Gain 16, Integral $\frac{1}{256}$
load was rapidly shorted out repeatedly. The disturbance rejection of both controllers is quite good and can be seen in figure 3.15.

It is worth noting that the controller does end up with a small amount of undershoot when the controller rapidly changes the current setpoint but the error is short-lived. It is quite possible that an extra bit of logic looking for sudden setpoint derivatives that resets the voltage controller’s integral to full-scale might help solve this.

### 3.4.2 AC Controller Testing

In order to keep the device safe, initial testing was done with the line disconnected. A signal generator was used to trick the PLL into locking onto a simulated 60Hz. With the input coming from a DC bench supply, the output of the transformer was connected to a large resistive load. The resulting output voltage waveform is shown
in figure 3.16

The driving method for this plot is sign magnitude. It can be seen that at the zero crossings the system overshoots substantially. This has to do with the fact that the duty-cycle saturation starts to limit the ability of the system to decrease the output current. Because of the minimum on time for the top side mosfet, the duty cycle has a discontinuity below a duty cycle of roughly 10%. Figure 3.17 illustrates the problem of duty cycle limiting. Below a certain voltage it is clear that if the duty cycle is limited from going to 0 then the current will build up in the inductor until the voltage rises high enough to bring the system out of saturation.

3.4.2.1 Fully Differential

In order to combat the loss of control at zero crossing, fully differential bridge driving was implemented. If duty cycle limiting serves to limit the minimum duty cycle then
Figure 3.16: Sinusoidal output waveform

Figure 3.17: A plot showing $\frac{\text{di}}{\text{dt}}$ of the H-Bridge versus duty cycle with 45 volts in for various values of $V_{\text{Out}}$ demonstrating uncontrollability near 0 because of duty cycle limiting
sign magnitude will fail to control the output around 0. With differential bridge driving, 50% duty cycle is 0Volts so there is only a slight squashing of the peaks of the output instead of a loss of control around 0Volts. The results of this duty cycle limiting is illustrated in figure 3.18

![Figure 3.18](image)

Figure 3.18: A plot demonstrating the effects of duty cycle limiting in differential driven mode at the output

The pulse width modulating block for the H-bridge was modified to minimize duty cycle limitations. Scope captures were taken before (figure 3.19) and after (figure 3.20) with sample averaging on to illustrate the improvement.

The major difference was the removal of a max duty cycle. By removing the max duty cycle limit the system never hits a hard limit. Because the legs operate in anti-phase although at the extremes, one leg will hit a minimum on time limit (a result of the opto-coupler pulse width skew), the other leg will continue to be able to actuate linearly to full on. Effectively the gain of the system cuts in half at this point on the extreme ends when the output voltage is almost equal to the DC bus voltage.
Figure 3.19: A plot demonstrating the effects of duty cycle limiting in differently driven mode, Plot taken at half-bridge midpoints with time averaging to demonstrate average voltage
This performance could be improved even further with a better opto-coupler that introduces less pulse width skew. With less pulse width skew, smaller minimum on times could be achieved which would push the usable output voltage closer to the voltage on the DC bus.

The major problem encountered when this was implemented is that in fully differential mode, the inductor ripple current is at least twice the ripple current of sign magnitude. The voltage on the inductor is constantly reversing between $V_{Bus} - V_{Out}$ to $-(V_{Bus} + V_{Out})$. From equation 1.8, it can be seen that the worst case $I_{Ripple}$ that differential driving experiences is when $V_{Out} = 0$. Assuming a bus voltage of 45 volts and a filter inductance of $50\mu H$, $I_{Ripple} = 9\text{Amps}$
3.4.2.2 Common Mode Noise

One of the unforeseen consequences of the non-symmetric output filter is the high frequency common mode voltage at the output nodes induced by the h-bridge. With sign magnitude and the filter design in 1.8 there is a large difference between the positive half cycle and the negative half cycle. During the positive half cycle, the right hand side of the filter is tied to ground so the output has no common mode introduced. However, when the legs of the H-Bridge swap functionality then the output capacitor is suddenly swinging up and down at 50KHz. This common mode voltage couples through the transformer’s parasitic winding capacitance and shows up at the output. This is best illustrated in figure 3.21.

![Figure 3.21: Output voltage without any extra filtering](image)

In order to solve this problem, the leads of the transformer were fitted with some ferrite as a common mode choke. The incremental improvement of a single turn on the ferrite can be seen in figure 3.22. With an additional 4 turns, the noise was
brought to a reasonable level (see figure 3.23).

![Output voltage with a single turn of ferrite](image)

Figure 3.22: Output voltage with a single turn of ferrite

### 3.5 Software Design

The software for this design was substantially more complicated than the DC-DC converter as a result of the need to control a more complex system. Some of the following described design has in fact been left as future work although a road-map is laid out for predicted functionality.

#### 3.5.1 Boost Controller PWM

The PWM generating block for the boost controller was mostly straightforward with the only added complication being operation in discontinuous mode. The initial design for the PWM block simply assumed always operating in continuous mode and
therefore always switched synchronously. The problem comes when the load starts getting lighter. When the magnitude of the ripple current is larger than \( \frac{1}{2} \) the output current, the inductor current must return to zero within the switching cycle. The result if using an non-synchronously switched design is the current stops at 0. However, if the bridge is switched synchronously without regard to this, the inductor current will become negative and will start pumping current into and out of the input bus. This current reversal is highly undesirable for several reasons. First, this pumps reactive power needlessly through parasitic component resistances thereby multiplying system losses. This current reversal could also serve to de-stabilize the DC-DC controllers on the bus by subjecting them to unnecessary high frequency dynamics.

The PWM block was re-designed in order to take discontinuous conduction into account. By evaluating the output current sense value, the controller decides whether
to switch the top side mosfet in the design or not. Although asynchronous switching implies higher losses (diode drop vs extremely low $i^2R$ losses in the mosfet), since discontinuous mode occurs at the low end of power transfer, the loss is considered acceptable. It is conceivably possible to still do synchronous switching; however an additional set of high speed comparators would need to be connected to an inductor current sense resistor in order to allow for precise timing of mosfet turnoff.

3.5.2 H-Bridge Driving PWM

The H-Bridge PWM driving block generated a bit of trouble for the design. The initial filter design assumed much higher inductance in the transformer than was realistic. As a result, the ripple current in the filter inductor was extremely high. The first implementation of sign magnitude suffered from excessive duty cycle limitations. In figure 3.24, it can be seen that the controller loses the ability to control the output around 0, leading to massive overshoot.

In order to combat this instability, a fully differential PWM block was written in order to test with improved control. The results were drastically improved (see figure 3.25).

3.5.3 Sine Wave Generation and Phase Locking

In order for the controller to work, it needs to be capable of locking onto the phase and frequency of the AC-Line. In addition, once the converter has locked to the line, it needs to generate an independent pure sinusoid reference for the output controller to use as a setpoint. Since the design calls for the ability to deliver reactive power, the generated sinusoid must be capable of arbitrary phase shift.

To create the generic sinusoid, a numerically controlled oscillator was built. The prin-
Figure 3.24: Feed-forward output current overshoot

Figure 3.25: Fully differential output sine wave
ciple of operation of a numerically controlled oscillator is relatively simple algorithm. A sine wave lookup table is loaded into memory and then, on every clock cycle, the output skips a variable number of elements in the lookup table, thereby allowing an arbitrary frequency sinusoid to be output from a high frequency input clock. If the table skip size is restricted to integer values then it is easy to see that the output frequency is very coarsely restricted. In order to combat this, a fractional accumulator approach is taken. The basic mode of operation is as follows.

An integer variable corresponding to phase angle is incremented each cycle based on the specified phase-skip. The phase angle variable is used to determine the element to look up in the table. In the fractional accumulator approach, the operation is identical except only the MSBs of the phase angle variable are used to look up the output.

In the actual implementation this is taken one step further. While the results would be perfectly satisfactory with just truncating the LSBs, an additional level of smoothness can be accomplished by linear interpolation. In order to implement the look up and interpolation efficiently, the table size is restricted to be a power of two. By using a table size that is a power of two, wraparound logic is unnecessary. Instead, overflowing addition in the phase register handles jumping from the end to the beginning of the lookup table. Linear interpolation is performed using a single pipeline delay according to the following logic:

\[
\]

Because of the linear interpolation, the frequency precision achievable is limited really only by the number of fractional bits in the phase accumulator.

In order to phase lock with the line, a modified phase locked loop (PLL) was written to
lock the numerically controlled oscillator. A comparator on the line voltage feeds the PLL as the input clock. The comparator was intentionally designed with no hysteresis in order to provide the most accurate zero crossing detection. In order to prevent double transitions from tricking the PLL, the comparator is pre-conditioned with a de-bouncing circuit. The de-bouncing circuit is essentially time-based hysteresis; when the input changes, the output changes but cannot change again until a timer overflows. This serves to limit the maximum lockable line frequency. At the rising edge of the input, the PLL starts a counter and resets the phase of the numerically controlled oscillator. The counter is incremented by the fractional step size it is currently estimating. When the end of the cycle is reached (indicated by the next rising edge of the comparator), the error between the estimate and 360 degrees is calculated and added to an integral accumulator in order to form the next estimate. If the magnitude of the error is larger than 1, the lock counter is reset; otherwise, it is incremented. If the lock counter is greater than 6 (indicating 6 consecutive cycles of lock) and the frequency is within pre-defined limits, the block asserts the locked signal.

One of the benefits of this slightly modified PLL is that it resets the output phase every cycle. This helps to speed the locking process which is important with such a low frequency signal.

### 3.6 Future Work

This thesis has demonstrated that the current driven switchers function reasonably well. While it can’t be known for certain, it is reasonably expected that with some hardware revisions and software work, the system should work as intended. Given large enough magnetics and bus capacity, there is no foreseeable reason why the system would not scale well. Before more testing can be done, a hardware revision
needs to be done to both devices in order to fix magnetics and power supply concerns.

3.6.1 Low Power Circuitry

On the prototype there are a few unnecessary efficiency losses. The 3.3Volt supply and the 2.5Volt supply are both linearly regulated from 5Volts. If the linear regulators were swapped for switchers, some amount of extra efficiency could be gained back. The exact amount of power efficiency to be gained back by the migration to switchers is difficult to exactly calculate without directly measuring the consumption on the linear regulators but it will be non-negligible.

Similar to the DC-DC converter, there are some undesirable inefficiencies in the gate drive circuitry. The opto-couplers consume substantially more power than is necessary. A replacement part can be located that dissipates 1/10th the power. In addition, the gate drive supply rails would benefit greatly by being generated from a single multi-output magnetically coupled switching supply. The isolated switching supplies used in this design were at best 75% efficient and a properly designed multi-output supply might be able to push that towards 90%.

3.6.2 Battery Charging

The original design called for a contactor of some kind to disconnect the battery from the bus. In an effort to optimize board real estate, the original contactor was replaced with a P-Channel mosfet. This was a foolish decision because of the inherent body diode on the mosfet. The body diode prevents the battery from being fully disconnected from the bus which is especially hazardous in a short circuit condition.

In the next revision, the mosfet should be removed and replaced with a latching dual coil relay. With a latching dual coil relay, the switch will not consume any steady
state power. In addition, the relay will allow complete isolation from the bus.

In order to avoid a potentially uncontrollable situation, charging and discharging the battery was never tested. While the control should be relatively straightforward given that all the controllers have been implemented in current mode already, it still remains to test the charging controller.

The battery on the bus is the thing that mostly regulates the bus voltage. Thus it is important that the battery charge controller is well designed to prevent overcharging or completely draining the battery. In normal operation, the battery will be connected to the bus through a relay and a current shunt. Depending on the attached battery chemistry, the system will need to set its compliance values differently. Either the controller will function to regulate the bus voltage or alternately the battery current. Most battery chemistry’s can be charged and discharged quite accurately by using a constant current source with some preset voltage compliances. The target chemistry for this device would be some type of lithium chemistry although it should adapt well to lead acid among others. With rechargeable lithium chemistry’s, the mode of charge is generally constant current charging with a voltage compliance tail to charge the last several percent. This behavior is exactly the same as the design for the voltage-limiting boost current controller and seems likely to work. The only limiting factor will be the bus capacitance. With sufficiently large input bus capacitance, the systems should be thoroughly decoupled and the controller should be capable of regulating.

3.6.3 AC Bus Controller

Very similar to the battery charging controller the AC bus controller modulates the output magnitude of the sine wave stage to achieve high voltage bus regulation. The controller will function identically to the battery charge controller. However, it needs
to operate on a much slower timescale. If the controller has too much gain, it will make the output appear like DC instead of the desired sinusoid. Ideally, the controller will be designed with a sharp notch rejecting 50-70Hz in order to prevent the output from feeding back off of itself.

3.6.4 High Level Controller

While a suitable method has been proposed for managing the control of the inverter with a preset output and input current magnitude, a more sophisticated controller needs to be written that can handle high level functionality.

The high level controller needs to make decisions about the buying and selling of power. Depending on dynamic pricing strategies, it may be beneficial to buy power from the grid when power is cheap and then sell it back when power is expensive. The high level controller is constantly deciding the power balance in the system. The system can be in one of a few modes mostly indicated by the status of the battery. The system can be charging the battery strictly from solar, charging from solar and the grid, charging from the grid only, or bypassed and selling direct to the grid. It is the high level controller’s job to choose one of those states appropriately based on state of charge and pricing. In addition, when the battery is disconnected, the battery charge controller must be replaced with a controller that controls bus voltage. This bus voltage controller would regulate to the highest reasonable voltage possible to maximize bus efficiency by reducing $i^2R$ losses.

In the event that the grid goes down or the controller finds itself operating outside of compliance limits, the high level controller can also send shutdown or power limit commands to the DC-DC converters to prevent damage.

Lastly, the high level controller manages the phase angle of the output current. This can be decided based on utility preferences or time of day schedules. By controlling the
phase of the output, the controller affects the overall grid stability and power factor. One of the advantages to being able to increment the phase of the output continuously over 360 degrees is that the output can look like any power factor source or load. This means the device can operate in all four quadrants - the controller can simultaneously control the flow of real and reactive power in both directions independently. Even while charging its battery, the device can be contributing to grid stability. By varying itself to either lead or lag phase, the converter can manage reactive stability in the grid by applying a leading or lagging power factor to the line to oppose the grid power factor at that time.
Chapter 4

Conclusions and Future Work

4.1 Future Work

The next step for this architecture is to begin testing the devices as a whole system. While the individual parts have been proven to work in isolation, no testing has been done to prove how the components will interact with each other. In addition, the DC-AC inverter needs the remainder of its functionality proven. While the device has proven the ability to generate a sinusoidal output from a DC source, it is still has yet to prove line connected power transfer. In order to make the device line-capable, the output stage needs substantially larger magnetics. With much larger magnetics in the output filter, the 50KHz ripple current will be extremely small and will make the output smoother. This is essential for making the device easy to control.

The original challenge posed with this thesis is to build a grid-tie inverter system capable of working with any voltage and power rating. While this is a brilliant idea in theory, the wide specification range makes it difficult to optimize for all cases. A possible solution is to design the power magnetics off-board. A design could be architected with all the logic and safety circuitry on a small module with power
electronics on a separate board. In this way, the control is the same at all power levels with just different power electronics and magnetics.

One of the challenges faced with the prototype is that with such low output voltages (15Volts DC bus, 24VAC line) it is difficult to push large amounts of power because $i^2R$ losses become significant and start pushing efficiency down. By connecting to the actual AC line and operating at much higher voltages, the efficiency may be able to go up a few percent because of conduction losses being a lower percentage of output power in a higher voltage system.

In order for the bus topology to come together well, it is necessary to implement a communication protocol so that the DC-AC inverter can set power limits on all devices. Once the communication protocol is implemented, there is no foreseeable reason why more devices could not be designed to function as extra battery charge controllers. In this way, the storage capacity of the DC bus can be made as large as desired. By building the whole bus as low bandwidth current sources, it should be easy to have any number of consumers and generators of power on the DC Bus just like the national electric grid (except without the difficulties associated with balancing an AC grid)

One of the other things to be considered moving forward with this project is how the high-level decision making takes place. The DC-AC inverter needs to choose when to buy energy and when to sell it. This will require real-time and predictive pricing in order to optimize both the benefit to the grid and to maximize profit for the user.

4.2 Conclusions

The thesis presented here has proposed a novel topology for linking many small DC current sources into a single DC-AC inverter. The prototypes that were designed and
tested, demonstrated that the fundamental idea is solid. However, more testing is required to conclusively say whether the architecture will hold up on a large scale. Small scale isolated testing indicated that the switching topologies proposed do in fact work, and work efficiently.

The overwhelming takeaway from this thesis is that magnetics are the most crucial part of the supply. In both the DC-DC converter and the DC-AC inverter, ripple current was the largest problem. In future revisions, the magnetics will need to be custom ordered or hand-wound. Larger magnetics will serve to only improve the output ripple and therefore decrease the system inter coupling when assembled into the final system.

This thesis mostly addressed the practicality of implementing the components in this architecture. However, the systems were tested in isolation so the question still remains whether controlling and stabilizing the DC bus architecture is possible. Although the statement can not be made conclusively, it seems highly probable that this architecture will work once the required revisions are made.
Appendix A

DC-DC Schematics
Figure A.1: The top level schematic diagram
Figure A.2: Cyclone layout
Figure A.3: Power electronics schematic
Figure A.4: Logic Power Supplies
Figure A.5: ADC connections and safety circuitry
Appendix B

DC-DC Verilog Source Code
B.1 Top Level Design

This shows the top level interconnects and how all the sub-blocks fit together.

Figure B.1: The top level FPGA design
### B.2 ADC Sampling

This is the block that controls the ADC sampling.

```verilog
module ADCSample(
    input Reset, // Hold high to reset state machine
    input Clock, // Input clock should be 32 Mhz or less
    input DataInA, // ADC data lines
    input DataInB, // ADC data lines
    output reg ClockOut, // ADC clock generated
    output reg[2:0] Address, // The three address pins on the ADC
    output reg ChipSelect, // ADC Outputs
    output [47:0] SystemState, // The actual ADC values output
    output reg StrobeI, // Strobe outputs to trigger new data availability
    output reg StrobeV
);

// These are the addresses for our desired measurement channels
'define ADDRESSONE 3'b001
'define ADDRESSTWO 3'b011
// State variable mux for outputting
reg [11:0] IIn, IOut, VIn, VOut; // Registers for samples
StateVarMux outMux(IIn, IOut, VIn, VOut, SystemState);

reg [11:0] sampleInA, sampleInB; // Registers to shift into
reg [4:0] clockIndex = 0; // Always incr on rising edge

// Data is always valid on rising edge
// Shift in data. Data comes in as MSB first
always @(posedge ClockOut)
begin
    // This prevents shifting in a garbage bit at the end
    if (clockIndex < 28)
        begin
            sampleInA = sampleInA << 1;
            sampleInA[0] = DataInA;
            sampleInB = sampleInB << 1;
            sampleInB[0] = DataInB;
        end
    end

    // This block generates the adc clock and chip select
    always @(posedge ClockOut or posedge Reset)
    begin
        if (Reset)
            begin
                StrobeI = 0; // Init strobe low
                StrobeV = 0; // Init strobe low
                ChipSelect = 1; // CS starts high
                Address = 'ADDRESSONE; // Initialize to pseudodiff A1 and B1
                clockIndex = 0; // Restart the state machine counter
            end else begin
                clockIndex = clockIndex + 1'b1;
                // Generate the adc clock at 1/2 the input clock frequency
                if (clockIndex < 30 & & clockIndex > 1)
                    begin
                        ClockOut = clockIndex[0];
                    end
                case (clockIndex)
                    0:
                        ChipSelect = 0; // Trigger a sample
                    28:
                        begin
                            // Toggle which address we are reading, move it's value to
                            // an output register and trigger a strobe line
                            if (Address == 'ADDRESSONE) begin
                                Address = 'ADDRESSTWO;
                    end
```
IIn = sampleInA; // Read IIn from pseudo diff measurement A1-A2 (12 bit straight binary)
IOut = sampleInB; // Read IOut from pseudo diff measurement B1-B2 (12 bit straight binary)

end else begin
  Address = 'ADDRESSONE;
  VIn = sampleInA; // Read VIn from pseudo diff measurement A3-A4 (12 bit straight binary)
  VOut = sampleInB; // Read VOut from pseudo diff measurement B3-B4 (12 bit straight binary)
end

29: begin
  ChipSelect = 1; // Start quiet time
  if (Address == 'ADDRESSONE) begin
    StrobeV = 1; // Strobe to indicate new Voltage values
  end else begin
    StrobeI = 1; // Strobe to indicate new Current values
  end
end

30: begin
  StrobeV = 0; // Strobe low
  StrobeI = 0; // Strobe low
end
endcase
end
endmodule
B.3 Current Source Block

The current source logic consists of several sub-blocks as such:

Figure B.2: The current source controller block
B.4 Duty Cycle Calculator

This block calculates the output for the PWM block using a parameterized PI block see B.4.1

```verilog
module DutyCalc( // This should be clocked from the state variable inputs (i.e., the ADC)
    input Clock,
    input [11:0] Setpoint,
    input [47:0] SystemState,
    output [12:0] DutyOut,
    input AntiWindup, // Used to reset the integrator
    output reg OutOfRegulation,
    output Tracking // High when error is small
);

// State variable boilerplate
wire [11:0] IIn, IOut, VIn, VOut;
StateVarDeMux inMux (SystemState, IIn, IOut, VIn, VOut);

// To indicate whether we can capture power data
wire signed [12:0] Error;
assign Error = $signed({1'b0, Setpoint}) - $signed({1'b0, IIn});
assign Tracking = (Error > -20 && Error < 20);

wire signed [13:0] SignedDuty;
PILoop #(.INTMULT(1), .INTDIV(512),
    .INTSHIFT(0), .INTSKIP(0),
    .PMULT(1), .PDIV(2),
    .STATEVARWIDTH(13), .OUTVARWIDTH(14),
    .OUTSATLOW(0), .OUTSATHIGH(5120)) cur_control_inst(
    .Clock (Clock),
    .Setpoint(Setpoint),
    .ProcessVariable(IIn),
    .IntReset(AntiWindup),
    .IntFreeze(0),
    .PlantSetpoint(SignedDuty)
);

// This truncation is safe because the PILoop saturates to 0
assign DutyOut = SignedDuty[12:0];

// This is to indicate that the controller is unable to
// achieve the requested setpoint
assign OutOfRegulation = maxcount[4];

reg [5:0] maxcount;
always @(posedge Clock) begin
    if (DutyOut == 'MAXDUTY) begin
        maxcount <= maxcount + 1'b1;
    end else begin
        maxcount <= 0;
    end
end
endmodule
```

B.4.1 Parameterized PI Loop

This is the source code for the parameterized PI block. This block is designed for maximum re-usability and all data types are parameterized to support any data width for variables and calculations.
module PIloop(
  input Clock,
  input signed [STATEVARWIDTH-1:0] Setpoint,
  input signed [STATEVARWIDTH-1:0] ProcessVariable,
  input IntFreeze,
  input IntReset,
  output reg [OUTVARWIDTH-1:0] PlantSetpoint
);

parameter STATEVARWIDTH = 13;
parameter OUTVARWIDTH = 13;
parameter [7:0] INTSKIP = 0;
parameter INTSHIFT = 0;
parameter signed INTMULT = 1;
parameter signed INTDIV = 1024;
parameter signed PMULT = 1;
parameter signed PDIV = 2;
parameter signed [OUTVARWIDTH-1:0] OUTSATHIGH;
parameter signed [OUTVARWIDTH-1:0] OUTSATLOW = 0;

'define INTSATLOW (OUTSATLOW*INTDIV/INTMULT)
'define INTSATHIGH (OUTSATHIGH*INTDIV/INTMULT)

reg [7:0] IntegralSkipCount;
reg signed [31:0] Integral;
reg signed [31:0] Output;
wire signed [STATEVARWIDTH:0] Error;
assign Error = Setpoint - ProcessVariable;

// This block manages the integral skip clock
always @(posedge Clock) begin
  if (IntegralSkipCount == INTSKIP) begin
    IntegralSkipCount <= 0;
  end else begin
    IntegralSkipCount <= IntegralSkipCount + 1'b1;
  end
end

always @(posedge Clock) begin
  if (IntReset || INTMULT == 0) begin
    Integral <= 0;
  end else begin
    if (!IntFreeze) begin
      if (IntegralSkipCount == INTSKIP) begin
        Integral <= Integral + (Error >>> INTSHIFT);
      end
    end
    // Saturation control for the integral
    if (Integral < 'INTSATLOW) begin
      Integral <= 'INTSATLOW;
    end
    if (Integral > 'INTSATHIGH) begin
      Integral <= 'INTSATHIGH;
    end
  end
end

Output <= ((Integral*INTMULT)/INTDIV) + ((Error*PMULT)/PDIV);

// Saturation control on output
if ((Output < OUTSATHIGH) & (Output > OUTSATLOW)) begin
  PlantSetpoint <= Output[OUTVARWIDTH-1:0];
end else begin
  if (Output <= OUTSATLOW) begin
    PlantSetpoint <= OUTSATLOW;
  end else begin
    PlantSetpoint <= OUTSATHIGH;
  end
end

112
B.5 PWM Generation

This block manages the PWM generation for the H-Bridge. It takes in an offset (un)signed value for duty cycle. Sign of the duty cycle determines buck or boost mode operation and then magnitude of the duty cycle determines the actual duty cycle.

```verilog
module PWMGen (  
    input Reset, 
    input Enable,  
    input Clock,  // Expecting a system clock of 32x4 = 128Mhz  
    input [12:0] DutyInput,  
    output reg [3:0] GateOut,  
    output reg [3:0] State  
);  
// These counters are used to time the transition of the switched  
reg [11:0] CounterDeadTime; 
reg [14:0] DutyCounter;  // Latched the duty cycle  
reg [12:0] DutyLatched;  // Parameters  
parameter BOOSTENABLED = 1; // Set to 0 for disabling boost mode  
parameter DEADMILLISECONDS = 300;  // Number of deadtime ticks at 128Mhz  
parameter DUTYOVERFLOW = 2560 - DEADTIMEHL; // Number of ticks of a 128Mhz clock for 50Khz  
parameter MAXDUTY = DUTYOVERFLOW - 20; // This only applies to Boost mode  
// The defines for the state names  
'define STATEOFF 0  
'define STATEA 1  
'define STATEB 2  
'define STATEC 3  
'define STATEE 5  
'define STATEF 6  
'define STATEG 8  
'define STATEPRESET 9  
// Used for deciding buck or boost  
wire Boost;  
assign Boost = ( (DutyInput > DUTYOVERFLOW) & & BOOSTENABLED);  
always @ (posedge Clock or posedge Reset ) begin  
    if (Reset ) begin  
        State <= 'STATEPRESET;  
    end else begin  
        if (!Enable) begin  
            State <= 'STATEPRESET;  
        end else begin  
            // If it's a free-wheel state, increment the deadtime counter  
            // else increment DutyCounter  
```
if ( State == 'STATEB || State == 'STATEE ) begin
    CounterDeadTime <= CounterDeadTime + 1'b1;
end else begin
    DutyCounter <= DutyCounter + 1'b1;
end
// State transition logic:
case (State)
    'STATEA:
        begin
            // Count until overflow or threshold and goto StateD or StateB
            if ( DutyCounter >= DutyLatched || DutyCounter >= DUTYOVERFLOW )
                begin
                    State <= 'STATEB;
                end
        end
    'STATEB:
        begin
            // Check if enough time has elapsed then goto StateC
            if ( CounterDeadTime >= DEADTIMEHL )
                begin
                    CounterDeadTime <= 0; // Reset the Dead Time Counter
                    State <= 'STATEC; // We are in first half of buck mode, goto state C
                end
        end
    'STATEC:
        begin
            // Count until overflow then goto StateD
            if ( DutyCounter >= DUTYOVERFLOW )
                begin
                    DutyCounter <= 0;
                    // Start by latching in the Duty cycle
                    if ( Boost )
                        begin
                            DutyLatched <= DutyInput - DUTYOVERFLOW;
                            end
                    else begin
                        DutyLatched <= DutyInput;
                    end
                    // Reset the Dead Time Counter
                    CounterDeadTime <= 0;
                    // Transition to the next starting state (buck or boost)
                    if ( Boost )
                        begin
                            State <= 'STATEE;
                        end
                    else begin
                        State <= 'STATEA;
                    end
                end
        end
    'STATEE:
        begin
            // Check if enough time has elapsed then goto StateF
            if ( CounterDeadTime >= DEADTIMEHL )
                begin
                    CounterDeadTime <= 0; // Reset the Dead Time Counter
                    State <= 'STATEF;
                end
        end
    'STATEF:
        begin
            // Count until overflow or threshold and goto StateD, StateA or StateB
            if ( (DutyCounter >= MAXDUTY) || (DutyCounter >= DutyLatched) )
                begin
                    State <= 'STATEG; // Transition directly to StateH to prevent 100% duty cycle
                end
        end
endcase
/// End of State transition logic
if ( State == 'STATEA || State == 'STATEB ) begin
    DutyCounter <= DutyCounter + 1'b1;
end else begin
    DutyCounter <= DutyCounter + 1'b1;
end
DutyLatched <= DutyInput - DUTYOVERFLOW;
end else begin
  DutyLatched <= DutyInput;
end
//Reset the Duty Counter
DutyCounter <= 0;
//Transition to the next starting state (buck or boost)
if ( Boost ) begin
  State <= 'STATEE;
end else begin
  State <= 'STATEA;
end
end
$\text{default:}$
  State <= 'STATEOFF;  //This keeps any unwanted initialized values of
  state from wreaking havoc
endcase
end
end
//At state transitions, assign state to gates
always @(State)
  case (State)
    'STATEOFF:
      GateOut <= 4'b0000;
    'STATEPRESET:
      GateOut <= 4'b0000;
    'STATEA:
      GateOut <= 4'b0101;
    'STATEB:
      GateOut <= 4'b0100;
    'STATEC:
      GateOut <= 4'b0110;
    'STATEE:
      GateOut <= 4'b0001;
    'STATEF:
      GateOut <= 4'b1001;
    'STATEG:
      GateOut <= 4'b1011;
    $\text{default:}$
      GateOut <= 4'b0000;
  endcase
end
B.6 Compliance Monitoring

In order to keep the system within safe limits, a block was written to always keep track of what current setpoint will throw one of the important system variables outside of the desired compliance ranges. This also serves to protect the system in no-load conditions. Various blocks make use of this block’s output to maintain safe operating conditions.

```verilog
module Compliance (  
    input Clock,  
    input [47:0] SystemState,  
    input OutOfRegulation,  
    output reg [11:0] CompliancePoint,  
    output reg ComplianceHit  
);  
  // Standard state variable boilerplate  
  wire [11:0] IIn, IOut, VIn, VOut;  
  StateVarDeMux inMux (SystemState, IIn, IOut, VIn, VOut);  
  // Parameters for tweaking the trip point  
  parameter VINMINIMUM = 650;  
  parameter VOUTMAXIMUM = 3200;  
  parameter IINMAX = 2630;  
  parameter IOUTMAX = 2630;  
  wire TooFar;  
  assign TooFar = ( IIn > IINMAX || IOut > IOUTMAX || VIn < VINMINIMUM || VOut > VOUTMAXIMUM || OutOfRegulation );  
  // This always block manages updating the compliance point  
  always @(posedge Clock) begin  
    ComplianceHit <= TooFar; // Force this to be clocked  
    if ( (!ComplianceHit && IIn > CompliancePoint) || (ComplianceHit && IIn < CompliancePoint) ) begin  
      CompliancePoint <= IIn;  
    end  
  end  
endmodule
```

B.6.1 Perturb and Measure MPPT

This is the first implementation of maximum power point tracking. It consists of a fixed step size gradient ascent algorithm.

```verilog
// This module performs perturb and measure style MPPT  
module PowerTrack (  
    input Clock,  
    input Reset,  
    input [47:0] SystemState,  
    input ComplianceHit,  
    output reg [11:0] Setpoint
```

116
parameter STEPSIZE=10;

wire [11:0] IIn, IOut, VIn, VOut;
StateVarDeMux inMux (SystemState, IIn, IOut, VIn, VOut);

wire [11:0] IIn, IOut, VIn, VOut;
StateVarDeMux inMux (SystemState, IIn, IOut, VIn, VOut);

wire [23:0] PowerIn, PowerOut;
assign PowerIn = IIn * VIn;
assign PowerOut = IOut * VOut;

reg [24:0] Counter;
always @ (posedge Clock ) begin
Counter <= Counter + 1'b1;
end

always @ (posedge Clock ) begin
Setpoint <= 0;
end
else begin
if (ComplianceHit) begin
Setpoint <= Setpoint - STEPSIZE;
end else begin
if (PowerIncreasing) begin
if (SetpointIncreasing) begin
Setpoint <= Setpoint + STEPSIZE;
end else begin
// Keep us from going below 0
if (Setpoint >= STEPSIZE) begin
Setpoint <= Setpoint - STEPSIZE;
end else begin
Setpoint <= 0;
end
end
SetpointIncreasing <= SetpointIncreasing;
end else begin
if (SetpointIncreasing) begin
if (Setpoint >= STEPSIZE) begin
Setpoint <= Setpoint - STEPSIZE;
end else begin
Setpoint <= 0;
end
end
SetpointIncreasing <= ~SetpointIncreasing;
end else begin
end
PreviousPin <= PowerIn;
PreviousPOut <= PowerOut;
end
endmodule
B.7 Maximum Power Point Tracking

The RAM based MPPT logic consists of several sub-blocks as such:

Figure B.3: The RAM based MPPT block
B.7.1 RAM Based MPPT

B.7.1.1 Power RAM Storage block

This is the block that looks at the input power and constantly logs it. In addition, this performs a three element spatial average to prevent noise from corrupting the information. At the same time, this block streams the RAM contents and RAM address out for other blocks to read serially.

```verilog
module PowerRAMMaster(
    input RAMStreamClock, //Used to stream out the RAM (Clocks the output stream)
    input SampleClock,
    input [47:0] SystemState,
    input Tracking,
    input [11:0] CompliancePoint, //The current system max current allowed
    output [15:0] data_a,
    output [11:0] addr_a,
    output wren_a,
    output [15:0] data_b,
    output [11:0] addr_b,
    output wren_b,
    output [15:0] q_a, q_b,
    output reg [15:0] PowerVal,
    output reg [11:0] Address);

wire [11:0] IIn, IOut, VIn, VOut;
StateVarDeMux inMux (SystemState, IIn, IOut, VIn, VOut);

// Used to make a two state state machine
reg [2:0] State;
'define SAMPLEINC 3'b001
'define SAMPLEPREV 3'b010
'define SAMPLECUR 3'b011
'define SAMPLENEXT 3'b100
'define WRITE 3'b101

// Always calculate PowerIn
wire [23:0] PowerFullDepth;
wire [15:0] PowerIn;
reg [11:0] Address_A;
assign PowerFullDepth = IIn * VIn;
assign PowerIn = PowerFullDepth[23:8];
assign data_a = (ReadPower >> 1) + (PreviousPower >> 3) + (NextPower >> 3) + (CurrentPower >> 2);
assign wren_a = (State == 'WRITE);
assign addr_a = Address_A;
assign data_b = 0;
assign addr_b = copyIndex;
assign wren_b = 0;

// Temporaries for filtering
reg [23:0] ReadPower, PreviousPower, NextPower, CurrentPower;
reg [11:0] TempCurrent;
// Used for streaming out the ram
reg [11:0] copyIndex;
always @ (negedge SampleClock) begin
    // Only edit the ram if the controller is approaching steady state
    if (Tracking) begin
```
B.7.2 Maximum Tracker

This block reads the power RAM serial data stream and determines the maximum value continuously. This block also outputs a signal to trigger a re-sweep of the power
plot in the event it detects stale data.

```verilog
module MaximumTracker (   
  input RAMStreamClock,   
  input [15:0] PowerIn,   
  input [11:0] AddressIn, 
  input [11:0] CompliancePoint, 
  output reg [11:0] BestGuess, 
  output reg ReSweep   
); 

  // This parameter determines the required data discontinuity in order to 
  // trigger a re-sweep 
  parameter SWEEPTHRESHOLD=1024; 

  reg [11:0] TempGuess;   
  reg [15:0] TempPower;  
  reg [15:0] PrevPower;  
  reg [15:0] UnsignedDelta; 
  reg signed [25:0] SignedDelta; 

  always @ (posedge RAMStreamClock ) begin 
    if ( AddressIn == 0 ) begin 
      BestGuess <= TempGuess; 
      TempGuess <= 0; 
      TempPower <= 0; 
      ReSweep <= 0; 
    end 
    // Calculate sample to sample delta looking for discontinuities in the 
    // data stream 
    SignedDelta <= PrevPower - PowerIn; 
    if ( PrevPower > PowerIn ) begin 
      UnsignedDelta <= PrevPower - PowerIn; 
    end else begin 
      UnsignedDelta <= PowerIn - PrevPower; 
    end 
    if ( (UnsignedDelta > SWEEPTHRESHOLD && AddressIn < (CompliancePoint - 2)) ) begin 
      ReSweep <= 1; 
    end 
    // Store power for next iteration 
    PrevPower <= PowerIn; 
    // Store the maximum found value 
    if (TempPower < PowerIn) begin 
      TempPower <= PowerIn; 
      TempGuess <= AddressIn; 
    end 
  end 
endmodule
```

B.7.3 Power Sweep Generator

This block was written to sweep the input across its full range in order to populate the RAM with data.

```verilog
module PowerSweep (   
  input Clock, 
); 
```

122
input Reset,
input ComplianceHit,
input Enable,
output reg [11:0] Setpoint,
output reg SweepFinished
);

// This sets how much it increments by in each step of the sweep
parameter STEPSIZE = 3'b100;

// These variables and always block manage the divided clock
wire ClockDiv;
reg [23:0] ClockCounter;
GLOBAL divclock (ClockCounter[16], ClockDiv);
always @ (posedge Clock or posedge Reset)
begin
  if (Reset)
  begin
    ClockCounter <= 0;
  end else begin
    ClockCounter <= ClockCounter + 1'b1;
  end
end

// This block actually manages generating the sweep and stops when the
// system compliances are exceeded
always @ (posedge ClockDiv or posedge Reset)
begin
  if (Reset)
  begin
    Setpoint <= 0;
    SweepFinished <= 0;
  end else begin
    if (!Enable)
    begin
      Setpoint <= 0;
      SweepFinished <= 0;
    end
    if (ComplianceHit)
    begin
      Setpoint <= 0;
      SweepFinished <= 1;
    end else begin
      Setpoint <= Setpoint + STEPSIZE;
    end
  end
end
endmodule


B.7.4 Controller MUX

This block manages swapping between the sweep generating block and the maximum
tracker. In addition, this introduces a very small slow oscillation in the output to
ensure that the system keeps acquiring data on either side of the power peak.

module ControllerMux ( 
  input Clock,
  input Reset,
  input [11:0] BestGuess,
  input SweepTrigger,
  input [11:0] SweepSetpoint,
  input SweepFinished,
  output reg [11:0] Setpoint,
);
output reg SweepEnable;

parameter DEVIATION = 4;

reg [17:0] PerturbationCounter;
reg Sweeping;

always @ (posedge Clock or posedge Reset) begin
  // Reset logic
  if (Reset) begin
    Setpoint <= 0;
    SweepEnable <= 0;
    Sweeping <= 1;
  end else begin
    PerturbationCounter <= PerturbationCounter + 1;
    // This is the oneshot logic around the sweep generator block
    if (SweepFinished && !SweepTrigger) begin
      Sweeping <= 0;
    end else begin
      if (SweepTrigger) begin
        Sweeping <= 1;
      end
    end
  end
  // Switch between modes
  if (Sweeping) begin
    SweepEnable <= 1;
    Setpoint <= SweepSetpoint;
  end else begin
    SweepEnable <= 0;
    if (PerturbationCounter[17]) begin
      Setpoint <= BestGuess + DEVIATION;
    end else begin
      Setpoint <= BestGuess - DEVIATION;
    end
  end
endmodule
Appendix C

DC-AC Schematics
Figure C.1: The top level schematic diagram
Figure C.2: Cyclone layout
Figure C.3: Cyclone JTAG and Power Pins
Figure C.4: Power electronics schematic
Figure C.5: Signal Conditioning and ADC
Figure C.6: Safety Circuitry
Figure C.7: Differential RC Filter

Figure C.8: Differential Serial Link

Figure C.9: Non Isolated Gate Drive

Figure C.10: Floating Gate Drive
Figure C.11: Logic Power Supplies
Appendix D

DC-AC Verilog Source Code

D.1 ADC Sampling Block

This block was derived mostly from the DC-DC ADC block with a few differences. The ADCs sample 12 different values, four at a time. Operating within the speed limit of the ADCs (1Mps) these variables could be acquired at $1/3$ MHz. However this is no longer an even multiple of the 50KHz switching waveform. Therefore, it was decided to sample four values at 500KHz and the other eight at 250KHz. By using even multiples of the switching frequency, we will have better rejection of switching dynamics that we want to isolate the controller from feeding back on.

The four variables sampled at 50KHz were input bus voltage and current, and battery voltage and current. There are actually only five other sampled variables but the extra three samples are configured to measure offset in the ADCs. Although it has not been implemented, it could be used to more accurately calibrate out any offset.

```verilog
module ADCSample (  
  input Reset,  
  input Clock, //Input clock should be 32Mhz or less  
  input DataInA_A, DataInA_B, DataInB_A, DataInB_B,  
  output reg[2:0] AddressA,  
  output reg[2:0] AddressB,  
  output reg ClockOut,  
```

134
output reg sgldiff,
output reg ChipSelect,
SystemState StateVars);

//The address lines for both ADCs
'define ADDR1A 3'b000
'define ADDR2A 3'b010
'define ADDR3A 3'b100
'define ADDR1B 3'b000
'define ADDR2B 3'b001
'define ADDR3B 3'b010

reg signed [11:0] OffsetA, OffsetB; //This is a pseudo-diff that is measured shorted - for testing offset
reg [11:0] snglOffset;

// Variables and assignments for the output
reg signed [11:0] IIn, IBoost, IACOut, IBatt, ILFilt, VACOut; //Registers for samples
reg [11:0] VBus, VBoost, VBatt;
signals StateVars.IIn = IIn;
signals StateVars.IBoost = IBoost;
signals StateVars.IACOut = IACOut;
signals StateVars.IBatt = IBatt;
signals StateVars.ILFilt = ILFilt;
signals StateVars.VBus = VBus;
signals StateVars.VBoost = VBoost;
signals StateVars.VBatt = VBatt;
signals StateVars.VACOut = VACOut;

assign Range = sgldiff;

reg [11:0] sampleInA_A, sampleInA_B, sampleInB_A, sampleInB_B; // registers to shift into
reg [4:0] clockIndex = 4'b0000; //Always incr on rising edge
reg flipflop = 1'b0;

//Data is always valid on rising edge
//Shift in data. Data comes in as MSB first
always @ (posedge ClockOut)
begin
  //This prevents shifting in a garbage bit at the end
  if (clockIndex < 28)
    begin
      sampleInA_A = sampleInA_A << 1;
sampleInA_A[0] = DataInA_A;
sampleInB_A = sampleInB_A << 1;
sampleInB_A[0] = DataInB_A;
sampleInA_B = sampleInA_B << 1;
sampleInA_B[0] = DataInA_B;
sampleInB_B = sampleInB_B << 1;
sampleInB_B[0] = DataInB_B;
    end
  end

//This block generates the adc clock and chip select
always @ (posedge ClockOut or posedge Reset)
begin
  if (Reset) begin
    ClockOut = 1'b1;  //Clock idle state is high
    ChipSelect = 1'b1;  //CS starts high
    AddressA = 'ADDR1A;  //Initialize to pseudodiff A1 and B1
    AddressB = 'ADDR1B;
    clockIndex = 4'b0000;  //Restart the state machine counter
    sgldiff = 1'b1;  //Init in single ended mode
  end else begin
    clockIndex = clockIndex + 1'b1;
    //Generate the adc clock at 1/2 the input clock frequency
    if (clockIndex < 30 && clockIndex > 1)
begin
    ClockOut = clockIndex[0];
end

begin
    case (clockIndex)
        0:
            ChipSelect = 0; // Trigger a sample
        28:
            begin
                // Toggle which address we are reading, move its value to
                // an output register and trigger a strobe line
                case (AddressA)
                    'ADDR1A:
                        begin
                            flipflop = ~flipflop;
                            if (flipflop) begin
                                AddressA = 'ADDR2A;
                                AddressB = 'ADDR2B;
                            end else begin
                                AddressA = 'ADDR3A;
                                AddressB = 'ADDR3B;
                            end
                            // ADCA
                            IIn = sampleInA_A;
                            IBatt = sampleInA_B;
                        end
                    'ADDR2A:
                        begin
                            AddressA = 'ADDR1A;
                            AddressB = 'ADDR1B;
                            // ADCA
                            IBoost = sampleInA_A;
                            ILFilt = sampleInA_B;
                        end
                    'ADDR3A:
                        begin
                            AddressA = 'ADDR1A;
                            AddressB = 'ADDR1B;
                            // ADCA
                            IACOut = sampleInA_A;
                            OffsetA = sampleInA_B;
                        end
        29:
            begin
                ChipSelect = 1; // Start quiet time
                case (AddressB)
                    'ADDR1B:
                        sgldiff = 1'b1;
                    'ADDR2B:
                        sgldiff = 1'b1;
                    'ADDR3B:
                        sgldiff = 1'b0;
                    default:
                        sgldiff = 1'b1;
                endcase
            end
    endcase
end
D.2 Filtering Block

This is the parameterized rectangular filter block. In addition, an unsigned module was written that wraps the signed implementation with an extra bit.

```vhdl
module RectFilter (  
    input Reset,  
    input Clock,  
    input signed [WIDTH-1:0] SampleIn,  
    output reg signed [WIDTH-1:0] SampleOut  
);  

    parameter [7:0] WINDOWLEN = 10;  
    parameter WIDTH = 12;  
    reg signed [WIDTH-1:0] Samples [WINDOWLEN];  
    reg signed [WIDTH:0] Delta;  
    wire signed [24:0] Sum; // Intentionally oversized  
    wire signed [WIDTH-1:0] Result;  
    reg flipflop;  
    assign Delta = SampleIn - Samples[WINDOWLEN-1];  
    assign Result = Sum/$signed(WINDOWLEN);  
    int i;  
    always @(posedge Clock or posedge Reset) begin  
        if (Reset) begin  
            flipflop <= 1;  
            for (i=0; i < WINDOWLEN; i=i+1) begin  
                Samples[i] <= 0;  
                Sum <= 0;  
            end  
        end else begin  
            // The flipflop allows double clocking and reduces the pipeline delay  
            if (flipflop) begin  
                // Shift samples  
                for (i=1; i < WINDOWLEN; i = i + 1) begin  
                    Samples[i] <= Samples[i-1];  
                end  
                Samples[0] <= SampleIn;  
            end  
            Sum <= Sum + Delta;  
            flipflop <= ~flipflop;  
            // Clock previous sample  
            SampleOut <= Result;  
        end  
    end  
endmodule
```

```vhdl
module UnsignedRectFilter (  
    input Reset,  
    input Clock,  
    input [WIDTH-1:0] SampleIn,  
    output [WIDTH-1:0] SampleOut  
);  

    parameter [7:0] WINDOWLEN = 10;  
endmodule
```
parameter WIDTH = 12;

wire [WIDTH:0] signedout;
assign SampleOut = signedout[WIDTH-1:0];
RectFilter #( .WIDTH(WIDTH+1), .WINDOWLEN(WINDOWLEN)) signfilt_inst (
  .Reset(reset),
  .Clock(clock),
  .SampleIn(1'b0, samplein),
  .SampleOut(signedout)
);
endmodule

module FilterBlock (input reset, input clock1M, input clock500K, SystemState StateUnFilt, SystemState StateFilt);
RectFilter #( .WINDOWLEN(10) ) IInfilt(
  .Reset(reset),
  .Clock(clock1M),
  .SampleIn(StateUnFilt.IIn),
  .SampleOut(StateFilt.IIn)
);
RectFilter #( .WINDOWLEN(5) ) IBoostfilt(
  .Reset(reset),
  .Clock(clock500K),
  .SampleIn(StateUnFilt.IBoost),
  .SampleOut(StateFilt.IBoost)
);
RectFilter #( .WINDOWLEN(5) ) IACOutfilt(
  .Reset(reset),
  .Clock(clock500K),
  .SampleIn(StateUnFilt.IACOut),
  .SampleOut(StateFilt.IACOut)
);
RectFilter #( .WINDOWLEN(10) ) IBattfilt(
  .Reset(reset),
  .Clock(clock1M),
  .SampleIn(StateUnFilt.IBatt),
  .SampleOut(StateFilt.IBatt)
);
RectFilter #( .WINDOWLEN(5) ) ILFiltfilt(
  .Reset(reset),
  .Clock(clock500K),
  .SampleIn(StateUnFilt.ILFilt),
  .SampleOut(StateFilt.ILFilt)
);
RectFilter #( .WINDOWLEN(5) ) VACOutfilt(
  .Reset(reset),
  .Clock(clock500K),
  .SampleIn(StateUnFilt.VACOut),
  .SampleOut(StateFilt.VACOut)
);
UnsignedRectFilter #( .WINDOWLEN(5) ) VBusfilt(
  .Reset(reset),
  .Clock(clock500K),
  .SampleIn(StateUnFilt.VBus),
  .SampleOut(StateFilt.VBus)
);
UnsignedRectFilter #( .WINDOWLEN(10) ) VBattfilt(
  .Reset(reset),
  .Clock(clock1M),
  .SampleIn(StateUnFilt.VBatt),
  .SampleOut(StateFilt.VBatt)
);
D.3 Parameterized PI Loop

This is the source code for the parameterized PI block. This block is designed for maximum re-usability and all data types are parameterized to support any data width for variables and calculations.

```verilog
module PILoop(
    input Clock,
    input signed [STATEVARWIDTH-1:0] Setpoint,
    input signed [STATEVARWIDTH-1:0] ProcessVariable,
    input IntFreeze,
    input IntReset,
    output reg [OUTVARWIDTH-1:0] PlantSetpoint
);

    parameter STATEVARWIDTH = 13;
    parameter OUTVARWIDTH = 13;
    parameter INTSHIFT = 0;
    parameter signed INTMULT = 1;
    parameter signed INTDIV = 1024;
    parameter signed PMULT = 1;
    parameter signed PDIV = 2;
    parameter signed [OUTVARWIDTH-1:0] OUTSATHIGH;
    parameter signed [OUTVARWIDTH-1:0] OUTSATLOW = 0;

    `define INTSATLOW (OUTSATLOW*INTDIV/INTMULT)
    `define INTSATHIGH (OUTSATHIGH*INTDIV/INTMULT)

    reg [7:0] IntegralSkipCount;
    reg signed [31:0] Integral;
    reg signed [31:0] Output;

    wire signed [STATEVARWIDTH:0] Error;
    //wire signed [31:0] Error;
    assign Error = Setpoint - ProcessVariable;

    always @ (posedge Clock) begin
        if (IntegralSkipCount == INTSKIP) begin
            IntegralSkipCount <= 0;
        end else begin
            IntegralSkipCount <= IntegralSkipCount + 1'd1;
        end

    end

endmodule
```
D.4 Differential PWM Generator

This is the fully differential PWM generator. It takes in a signed duty cycle and outputs the gate-drive for fully differential PWM.

```verilog
module DifferentialPWMGen (  
  input Reset,  
  input Clock, //Expecting a system clock of 32x4 = 128Mhz  
  input signed [11:0] DutyInput,  
  input Enable,  
  output reg [3:0] GateOut  
);  
  //The variables for each half bridge  
  reg [2:0] StateLeft, StateRight;  
  reg [11:0] CounterDeadTimeLeft, CounterDeadTimeRight;  
  reg [14:0] DutyCounterLeft, DutyCounterRight;  
  reg signed [13:0] DutyLatched;  
  reg [3:0] PWMOut;  
  reg PhaseLatched;  
  assign GateOut[3:2] = RightBridge;  
  assign GateOut[1:0] = LeftBridge;  
endmodule
```
parameter DEADTIME = 300; // Number of deadtime ticks at 128 Mhz Measured 2.3 uS - use 2.35 for safety
parameter DUTYOVERFLOW = 2560 - DEADTIME; // Number of ticks of a 128 Mhz clock for 50 Khz
parameter DUTYMID = (DUTYOVERFLOW)/2;

// This is the defines for the states
'define STATEOFF 3'h0
'define STATEPRESET 3'h1
'define STAtea 3'h2
'define STATEB 3'h3
'define STATEC 3'h4
'define STATED 3'h5

always @ (posedge Clock or posedge Reset) begin
  if (Reset) begin
    StateLeft <= 'STATEPRESET;
    StateRight <= 'STATEPRESET;
    DutyCounterLeft <= 0;
    DutyCounterRight <= 0;
    CounterDeadTimeLeft <= 0;
    CounterDeadTimeRight <= 0;
  end else begin
    if (!Enable) begin
      StateLeft <= 'STATEPRESET;
      StateRight <= 'STATEPRESET;
      DutyCounterLeft <= 0;
      DutyCounterRight <= 0;
      CounterDeadTimeLeft <= 0;
      CounterDeadTimeRight <= 0;
    end else begin
      // If it's a free-wheel state, increment the deadtime counter
      // else increment
      if (StateLeft == 'STATEB) begin
        CounterDeadTimeLeft <= CounterDeadTimeLeft + 1'b1;
      end else begin
        DutyCounterLeft <= DutyCounterLeft + 1'b1;
      end
      if (StateRight == 'STATEC) begin
        CounterDeadTimeRight <= CounterDeadTimeRight + 1'b1;
      end else begin
        DutyCounterRight <= DutyCounterRight + 1'b1;
      end
      // State transition logic for the left side:
      case (StateLeft)
        'STATEA:
        begin
          if (DutyCounterLeft >= DutyLatched || DutyCounterLeft >= DUTYOVERFLOW ) begin
            StateLeft <= 'STATEB;
          end
        end
        'STATEB:
        begin
          if (CounterDeadTimeLeft >= DEADTIME) begin
            CounterDeadTimeLeft <= 0; // Reset the Dead Time Counter
          end
          if (DutyCounterLeft >= DUTYOVERFLOW) begin
            StateLeft <= 'STAtea;
          end else begin
            StateLeft <= 'STATEC;
          end
        end
        'STATEC:
        begin
          if (DutyCounterLeft >= DUTYOVERFLOW) begin
            DutyCounterLeft <= 0;
            StateLeft <= 'STAtea;
          end
        end
endcase
end
endcase

// At state transitions, assign state to gates
always @ ( StateLeft ) begin
    case ( StateLeft )
        'STATEOFF: LeftBridge <= 2'b00;
        'STATEPRESET: LeftBridge <= 2'b00;
        'STATEA: LeftBridge <= 2'b00;
        'STATEB: LeftBridge <= 2'b01;
        'STATEC: LeftBridge <= 2'b00;
        default: LeftBridge <= 2'b10;
    endcase
end

}
D.5 Phase Locked Loop

The PLL is composed of several sub-blocks detailed below.

D.5.1 Numerically Controlled Oscillator

The numerically controlled oscillator is a relatively simple block that just does linear interpolation on a variable step-size lookup table. By parameterizing the number of fractional bits and the lookup table size, it is possible to control how fine the frequency resolution is (at the expense of logic elements). Lastly, the oscillator has a phase offset value to independently control phase and frequency of the output.

```verilog
module DDSBlock (
    input Clock,
    input [TABLE_ADDR_WIDTH+FRAC_BITS-1:0] FracStepSize,
    input [TABLE_ADDR_WIDTH+FRAC_BITS-1:0] PhaseOffset,
    input PhaseSync,
    output signed [TABLE_DATA_WIDTH-1:0] Value
);

// These parameters allow for compile time definition
// of the block data widths
parameter TABLE_ADDR_WIDTH = 11;
parameter TABLE_DATA_WIDTH = 16;
parameter FRAC_BITS = 4;
wire [TABLE_ADDR_WIDTH-1:0] RomAddrA, RomAddrB;
```
// Stores a sample for pipelining
reg signed [TABLE_DATA_WIDTH-1:0] SampleDelayed;

// This is the phase accumulator (full width fractional number)
// These variables are used for breaking apart the fraction accumulator
wire [TABLE_ADDR_WIDTH-1:0] Phase;
wire [FRAC_BITS-1:0] PhaseFraction;
assign Phase = PhaseFraction[TABLE_ADDR_WIDTH+FRAC_BITS-1:FRAC_BITS];
assign PhaseFraction = PhaseFraction[FRAC_BITS-1:0];
assign PhaseFractionDelayed = PhaseFraction;

// Lookup this value and one ahead
assign RomAddrA = Phase;
assign RomAddrB = Phase + 1'b1;

// This is our lookup table
dprom #( . WIDTHAD (TABLE_ADDR_WIDTH) , . WIDTHDATA (TABLE_DATA_WIDTH) , . WORDCOUNT (2**TABLE_ADDR_WIDTH) , . INITFILE ("sine.hex") ) rom_inst (.
.address_a (RomAddrA),
.address_b (RomAddrB),
clock (Clock),
.q_a (RomDataA),
.q_b (RomDataB));

// This is the logic to move through the ROM
reg syncprevious; // Used for detecting rising edge
always @ (posedge Clock) begin
  if (syncprevious == 0 && PhaseSync == 1) begin
    PhaseFractional <= PhaseOffset;
  end else begin
    PhaseFractional <= PhaseFractional + FracStepSize;
  end
  syncprevious <= PhaseSync;
  // Store two values for the pipeline
  PhaseFractionDelayed <= PhaseFraction;
  SampleDelayed <= RomDataA;
end
endmodule

D.5.2 Hysteresis

This block pre-conditions the line comparator signal for the phase synchronizer.

module Hysteresis(
  input Signal,
  input Clock,
  output CleanSignal
);
D.5.3 Phase Synchronizer

The phase synchronizer block is the combination of a phase comparator and a low pass filter in a single block. The block functions by using a rollover counter with a variable step size. The size of the counter is sized to be larger than the NCO's phase accumulator. This is in order to allow the block to detect large frequency mismatch.

At each rising edge, the block integrates the error between the previous guess and the actual measured frequency. This then determines the counter increment size. The counter then free counts until the next rising edge where it subtracts the counter value from the desired $2^{\text{RolloverBit}}$ in order to calculate error. After several cycles of minimal phase error, the block asserts a locked signal in order to alert other blocks that the frequency is locked.

In this scheme, the Rollover Bit is chosen to be the same as the width of the NCO’s phase accumulator. This allows the counter step size to feed the NCO and generate our internal sine reference. Given how low frequency the line is, this technique could likely be made even more robust by utilizing a numerically controlled oscillator in this block and feeding back off the ADC voltage values.
module PhaseSynchro(
    input Clock,
    input ZeroCross,
    output [STEPWIDTH-1:0] StepSize,
    output PhaseStart,
    output reg Locked
);
reg [4:0] LockCount;
reg signed [STEPWIDTH+10:0] StepAccumulator;
// This register counts up with the NCO and then is used to calculate error
reg [STEPWIDTH:0] StepCount; // Intentionally one bit too large to prevent overflow except in massive frequency offset conditions
reg PrevCross; // Used to make the design synchronous (rising edge detection)
// This is the counter to count along with the numerically controlled oscillator
wire [STEPWIDTH+3:0] AbsolutePhase; // should follow the NCO phase value
assign AbsolutePhase = StepSize * StepCount;
// This is the error in the counted value versus the expected overflow point
wire signed [STEPWIDTH:0] PhaseError;
assign PhaseError = $signed(2**(STEPWIDTH-1)) - $signed(AbsolutePhase);
// This is the error term to be added to the accumulator for generating the new stepsize
wire signed [STEPWIDTH:0] StepError;
assign StepError = PhaseError/$signed({1' b0, StepCount});
assign Locked = LockCount > 6 && StepSize > 300 && StepSize < 600;
// The output stepsize
assign StepSize = StepAccumulator/$signed(16);
always @(posedge Clock) begin
    // This is to prevent harmonic locking
    if (AbsolutePhase >= 2**(STEPWIDTH + 1)) begin
        StepAccumulator <= StepAccumulator/2;
        StepCount <= 0;
        end else begin
            if (PrevCross == 0 && ZeroCross == 1) begin
                StepAccumulator <= StepAccumulator + StepError;
                StepCount <= 0;
                PhaseStart <= 1;
                if (!Locked) begin
                    LockCount <= LockCount + 1'b1;
                end
                end else begin
                    LockCount <= 1'b0;
                end
                end else begin
                    PhaseStart <= 0;
                end
                StepCount <= StepCount + 1'b1;
            end
            PrevCross <= ZeroCross;
        end
endmodule
D.6 Boost Converter Controller

This block is a combination of a current controller and a modulating voltage limiting controller. This makes use of the block in B.4.1.

```verilog
module BoostCalc(
    input Clock,
    input signed [11:0] Setpoint,
    SystemState StateVars,
    input AntiWindup, //Used to reset the integrator
    output [11:0] DutyOut,
);

//This is the high voltage bus voltage compliance
parameter signed [12:0] VMAX = 2000;

//Use a size of 13 because we aren’t inputing signed numbers
wire signed [11:0] VoltageGain;

PILoop #( .INTMULT (1) , .INTDIV (256) ,
    .INTSHIFT (0) , .INTSKIP (9) ,
    .PMULT (8) , .PDIV (1) ,
    .STATEVARWIDTH (13) , .OUTVARWIDTH (12) ,
    .OUTSATLOW (0) , .OUTSATHIGH (2047) ) volt_control_inst(
    .Clock ( Clock ) ,
    .Setpoint ( VMAX ) ,
    .ProcessVariable ( {1'b0, StateVars.VBoost} ) ,
    .IntReset ( AntiWindup ) ,
    .IntFreeze (0) ,
    .PlantSetpoint ( VoltageGain )
);

wire signed [11:0] SafeSetpoint; //This is the saturation controlled setpoint
Modulator #( .WIDTH (12) , .MODWIDTH (12) ) mod_inst (
    .Clock ( Clock ) ,
    .ModIn ( VoltageGain ) ,
    .InSignal ( Setpoint ) ,
    .OutSignal ( SafeSetpoint )
);

//This is the PI Loop that actually drives the pwm generation
wire signed [12:0] SignedCurDuty;
PILoop #( .INTMULT (1) , .INTDIV (64) ,
    .INTSHIFT (0) , .INTSKIP (9) ,
    .PMULT (1) , .PDIV (2) ,
    .STATEVARWIDTH (12) , .OUTVARWIDTH (13) ,
    .OUTSATLOW (0) , .OUTSATHIGH (2252) ) cur_control_inst(
    .Clock ( Clock ) ,
    .Setpoint ( SafeSetpoint ) ,
    .ProcessVariable ( StateVars.IIn ) ,
    .IntReset ( AntiWindup ) ,
    .IntFreeze (0) ,
    .PlantSetpoint ( SignedCurDuty )
);

//This is safe because the saturation logic will keep SignedCurDuty positive
assign DutyOut <= SignedCurDuty[11:0];
endmodule
```

147
D.6.1 Modulator

This is the fractional multiplier used to make the voltage limiting logic work.

```verilog
module Modulator(
    input Clock,
    input signed [MODWIDTH-1:0] ModIn,
    input signed [WIDTH-1:0] InSignal,
    output signed [WIDTH-1:0] OutSignal
);
// The size of the signal
parameter WIDTH = 12;
// The size of the gain term
parameter MODWIDTH = 12;
// The full result
reg [MODWIDTH + WIDTH -2:0] FullPrecision;
// Output is just the MSBs
assign OutSignal = FullPrecision [MODWIDTH + WIDTH -2: MODWIDTH -1];
// Clocked math
always @ (posedge Clock) begin
    FullPrecision <= InSignal * ModIn;
end
endmodule
```

D.7 Boost PWM Generator

This block is the gate drive signal generator for the boost controller.

```verilog
module BoostPWM(
    input Reset,
    input Clock, // Expecting a system clock of 32x4 = 128Mhz
    input [11:0] DutyInput,
    input Enable,
    SystemState StateVars,
    output reg [1:0] GateOut
);
reg [2:0] State;
reg [11:0] CounterDeadTime;
reg [14:0] DutyCounter;
reg [1:0] DutyLatched;
reg [1:0] PWMOut;
reg Discontinuous;

parameter DEADTIMEHL = 320; // Number of deadtime ticks at 128Mhz
parameter DEADTIMELH = 16;
parameter DUTYOVERFLOW = 2560 - DEADTIMELH - DEADTIMEHL; // Number of ticks of a 128Mhz clock
for 50Khz
parameter MAXDUTY = DUTYOVERFLOW - 10; // This only applies to Boost mode

'define STATE_OFF 3'h0
'define STATE_PRESET 3'h1
'define STATEE 3'h2
'define STATEF 3'h3
'define STATEH 3'h5
'define THRESH 5
```
'define HYSTERESIS 5

always @(posedge Clock or posedge Reset) begin
    if (Reset) begin
        State <= 'STATE_PRESET;
        if (!Enable) begin
            State <= 'STATE_PRESET;
        end else begin
            //If it’s a free-wheel state, increment the deadtime counter
            //else increment DutyCounter
            if (State == 'STATEE) begin
                CounterDeadTime <= CounterDeadTime + 1'b1;
            end else begin
                DutyCounter <= DutyCounter + 1'b1;
            end
        end
        //State transition logic:
        case (State)
            'STATEE:
                begin
                    if (StateVars.IIn < 'THRESH) begin
                        Discontinuous <= 1'b1;
                    end
                    //Check if enough time has elapsed then goto StateF
                    if (CounterDeadTime >= DEADTIMEHL) begin
                        CounterDeadTime <= 0; //Reset the Dead Time Counter
                        State <= 'STATEF;
                    end
                end
            'STATEF:
                begin
                    //Count until overflow or threshold and goto StateD, StateA or StateB
                    if ((DutyCounter == MAXDUTY) || (DutyCounter >= DutyLatched))
                        begin
                            State <= 'STATEH; //Transition directly to StateG to prevent
                                            100% duty cycle
                        end
                end
            'STATEH:
                begin
                    if (DutyCounter >= DUTYOVERFLOW) begin
                        //Start by latching in the Duty cycle
                        DutyLatched <= DutyInput;
                        //Reset the Duty Counter
                        DutyCounter <= 0;
                        //Transition to the starting state
                        State <= 'STATEE;
                        if (StateVars.IIn > ('THRESH + 'HYSTERESIS))
                            begin
                                Discontinuous <= 0;
                            end
                    end
                end
            'STATE_PRESET:
                begin
                    if (DutyCounter == DUTYOVERFLOW) begin
                        DutyCounter <= 0;
                        State <= 'STATEE;
                    end
                end
            default:
                begin
                    State <= 'STATE_OFF; //This keeps any unwanted initialized values of
                                            state from wreaking havoc
                end
        endcase
    end else begin
        //At state transitions, assign state to gates
        always @(State) begin
            case (State)
            end
        end
end
'STATE_OFF:
  GateOut <= 2'b00;
'STATE_PRESET:
  GateOut <= 2'b00;
'STATEE:
  GateOut <= 2'b00;
'STATEF:
  GateOut <= 2'b10;
'STATEH:
  if (Discontinuous) begin
    GateOut <= 2'b00;
  end else begin
    GateOut <= 2'b01;
  end
  default:
    GateOut <= 2'b00;
  endcase
end
Bibliography


