© 2014 Shibin Qin

SUB-MODULE DIFFERENTIAL POWER PROCESSING FOR PHOTOVOLTAIC APPLICATIONS

BY

SHIBIN QIN

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Adviser:

Assistant Professor Robert C. N. Pilawa-Podgurski

ABSTRACT

Photovoltaic energy offers great benefits over conventional energy resources, but its large scale deployment has long been hindered by the prohibitive cost and low energy output. Advanced power electronics design plays a critical role in reducing the cost of PV energy systems and improving the power output. Conventional distributed power electronics solutions in a PV system such as DC optimizer and micro-inverters have been developed, but still they still have certain limitations in terms of cost and efficiency. This thesis proposes the use of a technique known as differential power processing (DPP) to achieve substantial efficiency improvement and cost reduction over the conventional solutions and develops hardware and control design to apply the DPP technique to a PV system.

This thesis introduces the fundamental principle of differential power processing and highlights its advantages over the conventional solutions. Several hardware designs of the DPP converter are presented, and several prototypes have been built and tested. These hardware prototypes achieve high conversion efficiency and very small volume to the point that they can be easily fit into a PV module junction box (commonly referred to as PV module integration).

Moreover, a distributed algorithm for controlling DPP converters in a PV system is developed. This algorithm achieves *true* maximum power point tracking (MPPT) of series-connected PV sub-modules by relying only on local voltage measurements and *neighbor-to-neighbor* communication between the DPP converters. Compared to previous solutions, the proposed algorithm achieves a reduced number of perturbations at each step and potentially faster tracking without adding extra hardware; all these features make this algorithm well-suited for long sub-module strings. An indoor experimental setup was established and various experiments prove the effectiveness of the proposed algorithm.

Finally, the DPP technique is applied to a micro-inverter system. Micro-inverters typically perform only module level MPPT and do not address power losses due to uncompensated sub-module mismatch. The thesis provides a solution to seamlessly integrate DPP converters into the existing micro-inverter system to improve its energy capture by recovering power losses due to sub-module mismatch. To demonstrate the effectiveness of the proposed solution, a hardware prototype was built and tested with an off-the-shelf commercial micro-inverter to prove the concept. The improvement in energy capture with DPP converters is experimentally verified.

To Mengyao, and my parents.

ACKNOWLEDGMENTS

I want to thank Professor Robert C.N. Pilawa-Podgurski, my thesis advisor, for his help and guidance in the past two years. Two years ago, I was a novice in power electronics. Two years later, I feel like a serious researcher and feel prepared to enter the next stage in my academic study. He not only taught me how to do research in power electronics, but also showed me, by his own example, how to be a great educator and a caring advisor. I am grateful for all his help both in my research work and in many other aspects. It is always a pleasure to have an advisor with such knowledge, insights and patience.

I want to thank Professor Alejandro D. Domínguez-García for spending much of his precious time on my projects. The work in this thesis would not be possible without his mathematical insight and tremendous help. I learned a lot from working with him. There are many people who helped me a lot with the work in this thesis. Stanton Cady derived the mathematical foundation of the distributed control algorithm and wrote a very neat simulation code that I used through this work. Katherine Kim helped me with the PV emulator development that I used for all the experiments in this thesis. Andrew Morrison made an excellent demonstration for my project. Rodrigo Serna and Jonathan Ehlmann provided much help with my PV experiments. I also want to thank my fellow researchers in the research group, Enver Candan, Roy Bell, Aaron Ho, Yutian Lei, Josiah McClurg, Christopher Barth, Marcel Schuck, and many more, for all their help, which included proofreading my papers, preparing me for my qual, and sharing their knowledge and great ideas, just to name a few. I also want to thank all my office mates, who make it a fun place to work. There are many more people than I can list here.

I would like to thank the sponsors of my research. This work was funded in part by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Number DE-AR0000217. I would also like to thank Chiang Chen Industrial Charity Foundation, who generously provided support for my first year in graduate school.

I would like to thank my parents, who taught me the most important lessons in my life. They always firmly supported my decision to pursue further education in the U.S., even if this means that I will be far away from home most of the time.

Most of all, I want to thank my girlfriend, Mengyao. The delicious food she cooked and her constant love and comfort provided the strongest support in all my endeavors. I am always grateful for her joyful company.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Solar energy, as a promising alternative to the conventional resources such as fossil fuels, has drawn significant attention in recent years. Solar energy technology can be divided into two major categories: concentrated sun power (CSP) and photovoltaics (PV). CSP technology uses concentrated sunlight to produce heat and generate electricity via a thermal process similar to that of a conventional fossil fuel power plant. PV technology, on the other hand, converts solar radiation directly to DC electricity using semiconductors that exhibit the photovoltaic effect. Although both technologies have their own advantages and both are undergoing rapid development recently, CSP technology is often limited to large utility-scale applications while PV technology is more flexible in scale and can be found in residential, commercial and utility-scale applications. For the purpose of this thesis, the scope of discussion will be limited to PV technology.

Photovoltaic solar energy offers a series of benefits such as low pollution and carbon dioxide emission, energy independence and potentially low energy cost, given that sunlight is a free energy resource. However, solar energy historically remains at a cost disadvantage compared to conventional energy resources primarily due to the high manufacturing cost of solar devices and low conversion efficiency from solar radiation to electricity. Therefore, by 2010, nearly 60 years after the PV technology was commercialized, it still supplies less than 0.1% of the total electricity demand in the U.S. [3].

Despite the small share of PV technology in the energy market nowadays, it shows enormous technological potential due to the recent development. According to [4], over the past five years, new installations of PV systems has grown by 60% each year while the installed PV system price has dropped by about 6% each year, down to as low as approximately $\$5/W$. According to the “Sunshot” vision study [3], as a key enabler for the large-scale PV deployment, the price of electricity generated by PV needs to drop further by 75% percent by the end of this decade to reach grid parity without subsidy. To meet this great challenge, improvements are required in all aspects of PV systems. The work in this thesis focuses specifically on improving the power electronics technology in PV system.

The cost of PV energy is typically evaluated in terms of levelized cost of energy (LCOE), i.e.,

$$\text{LCOE} = \frac{\text{installation cost} + \text{operation cost} + \text{maintenance cost}}{\text{electricity generated over lifetime}} \tag{1.1}$$

To reduce PV energy price, measures should be taken on both reducing the manufacturing and maintenance cost and improving energy conversion to generate more electricity within the system lifetime. The work in this thesis addresses these two goals (reducing the cost of the entire PV system and boosting the energy production) through advanced power electronics designs.

The work in this thesis is built around an innovative technique known as differential power processing (DPP). DPP allows for significant improvement in terms of power conversion efficiency while reducing power converter size and cost. In this work, an “element-to-element” DPP architecture was developed and bidirectional buck-boost converter is designed and implemented for this architecture. A centralized control method and a distributed control method have been developed to perform maximum power point tracking (MPPT) in the DPP architecture. An experimental setup that allows for controllable and repeatable indoor solar experiment is developed. The effectiveness of the proposed DPP approach has been experimentally verified with this setup.

1.2 Organization of this Thesis

The thesis can be divided into three major parts.

The first part consists of Chapters 2, 3, and 4. The principle and benefit of DPP are introduced. The PV emulation setup and DPP hardware design are presented. This part lays the foundation of the platform to study DPP in PV systems and is constantly referred to throughout the entire thesis.

Chapter 2 introduces the conventional power electronics solutions in PV systems and proposes the new technique denoted as DPP. The advantages of DPP over conventional solutions are analyzed. The basic operation principle of DPP is explained. Previous works by various researchers on DPP are reviewed.

Chapter 3 reviews the model of PV module characteristics and based on that knowledge, a high fidelity, easy-to-implement PV module emulator is presented. The proposed emulator can replicate the electrical behavior of a sunlight illuminated PV module in an indoor environment. The construction of this emulator requires only a PV module and basic laboratory equipment, while still providing dynamic performance that closely matches that of an illuminated PV module in an outdoor environment. This emulator is used by all the following experiments.

Chapter 4 presents the hardware design for a DPP converter. The design requirement is analyzed and two versions of hardware design are proposed: one with integrated power stage and one with discrete power stage. The advantages of each design are highlighted. Hardware prototypes of both designs are built and tested. These prototypes are used in all the following experiments as well.

The second part consists of Chapter 5 and Chapter 6. A distributed algorithm to control DPP converters for maximum power point tracking (MPPT) is derived in theory and tested in experiment.

Chapter 5 formulates the control algorithm in the simplest setting of a 3-sub-module, 2-DPP system and generalizes the algorithm to a system of any size. The mathematical background of this algorithm is introduced. The properties of the algorithm and its implementation considerations are discussed.

Chapter 6 explains the experiment performed to verify the effectiveness of this algorithm and presents the experimental results.

The third part consists of Chapter 7 and Chapter 8. This part explores the application of DPP in a micro-inverter system. A centralized control algorithm is developed to provide a well-suited solution to control DPP converters in the micro-inverter system. This solution is tested with a commercial off-the-shelf micro-inverter to prove that DPP converters can be seamlessly integrated into the existing micro-inverter system.

Chapter 7 analyzes the special requirement in the setting of a micro-inverter system and proposes a centralized control algorithm for the DPP converters that best meet the specific situation of micro-inverter system. Practical considerations for implementing this control algorithm are discussed.

Chapter 8 presents the experiment with a commercial off-the-shelf micro-inverter. The experimental results are presented and analyzed to show the benefit of the DPP solutions.

CHAPTER 2

BACKGROUND

In this chapter some background information on PV energy system is introduced to reveal the rationale of applying DPP in PV system. The concept of DPP is introduced and previous power electronics solutions and the new DPP solutions are presented and compared to explain the advantage of using DPP in PV system.

2.1 PV System Basics

A PV system typically consists of PV modules, power electronics and other ancillary components. A PV module consists of strings of PV cells that convert solar radiation to DC electricity directly. Without going into the detailed principle of the photovoltaic effect, power electronics engineers are typically only concerned with the external current-voltage (I-V) characteristics of PV module, as plotted in Fig. 2.1. The power-voltage (P-V) curves of PV module resulting from this I-V curve are plotted in Fig. 2.2. The I-V curve of a PV module exhibits non-linear characteristics and changes dramatically with irradiance conditions. Therefore, a PV module can generate maximum power only when it is operating at a unique point which is often referred to as maximum power point (MPP), as illustrated in Fig. 2.2.

When connecting a PV module to a load, as shown in Fig. 2.3a, the load has to be adjusted to match the impedance of the PV module at its MPP to extract the maximum power from the PV module. This is illustrated by Fig. 2.4. However, the load, typically a battery, the power grid, etc., is often fixed and cannot be varied to match the MPP impedance as needed. Moreover, the load may require the electricity in a different form; e.g., in grid-connected applications, the DC electricity generated by the PV module needs to be converted into AC to feed it to the grid. Therefore, power elec-

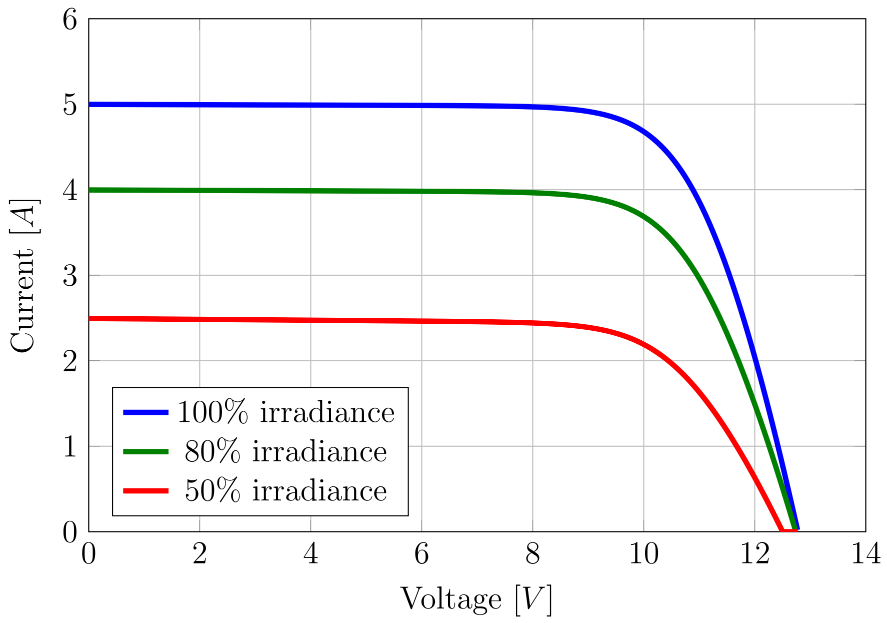


Figure 2.1: I-V characteristics of a PV module under $100\%, 80\%, 50\%$ normalized irradiance.

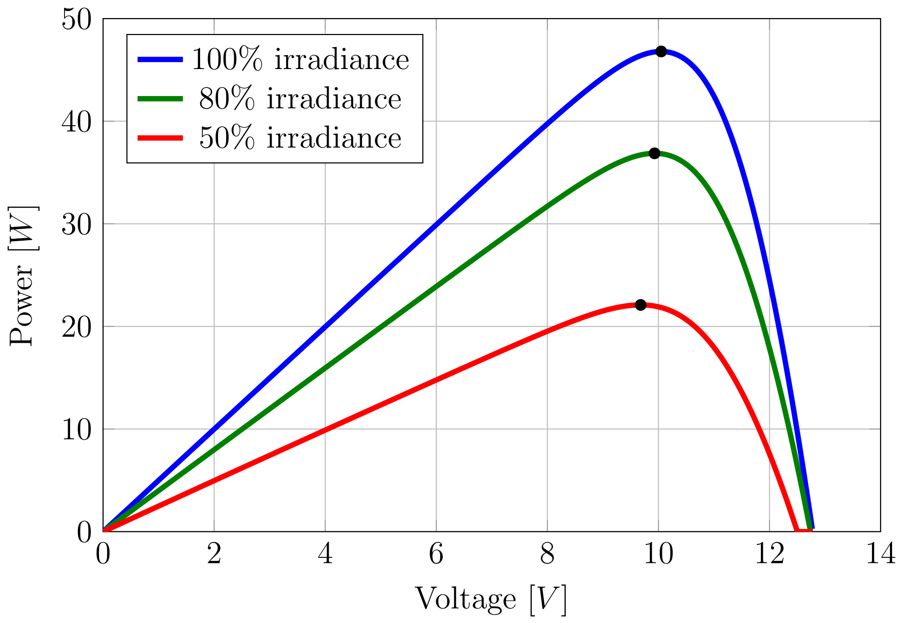
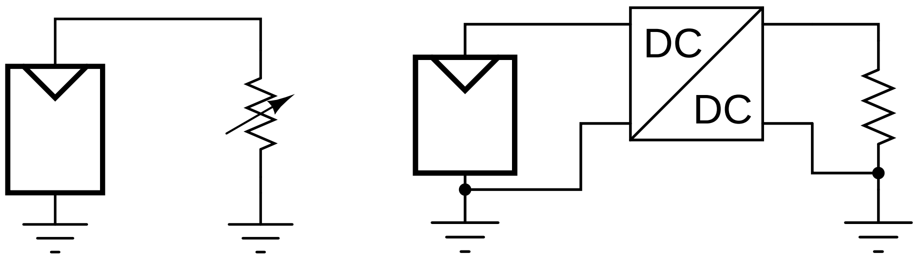


Figure 2.2: P-V characteristics of a PV module under $100\%, 80\%, 50\%$ normalized irradiance with MPPs marked with dots.

tronics converters are required in a PV system to perform MPPT (impedance matching) and power inversion if necessary.

To quantify the effectiveness of power electronics in PV systems, we define



(a) Variable load (b) Through DC/DC converter

Figure 2.3: PV module interfacing a load.

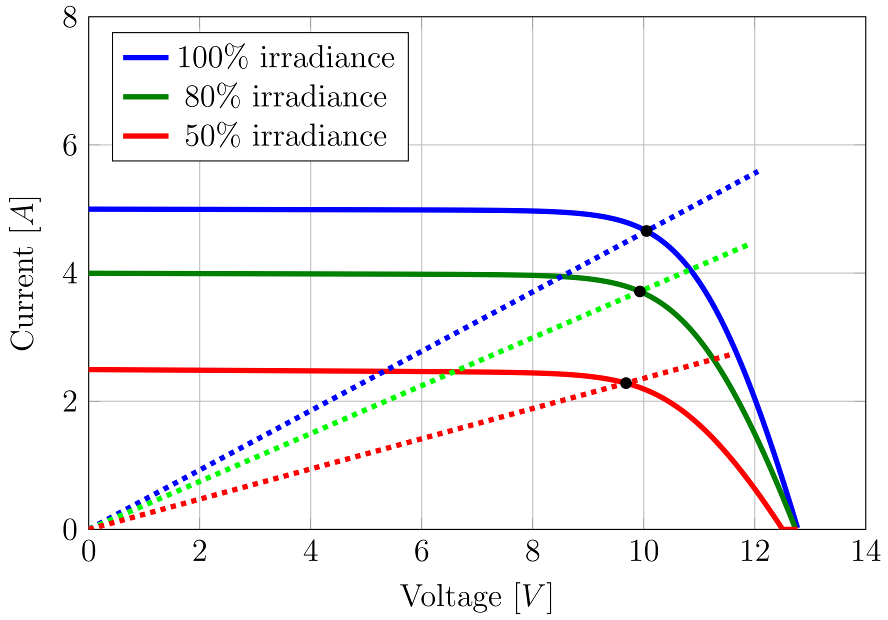


Figure 2.4: V-I characteristics of a PV module under 100%, 80%, 50% normalized irradiance with MPPs marked with dots. The load lines corresponding to MPP impedance matching are plotted with dotted lines.

tracking efficiency as

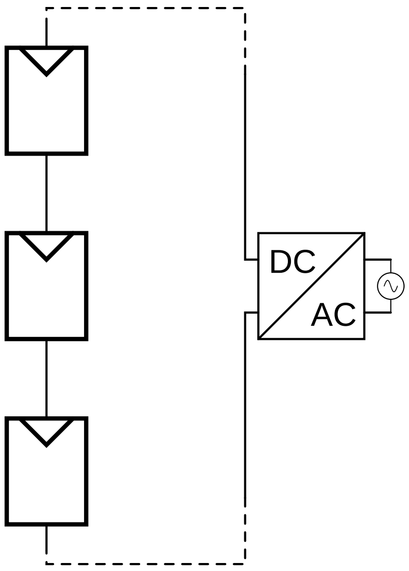
$$\xi = \frac{\sum_{i=1}^{n} p_i}{\sum_{i=1}^{n} p_{i,MPP}} \tag{2.1}$$

where $P_i, i = 1, ..., n$ is the actual power output of each PV module, and $P_{i,MPP}, i = 1, ..., n$ is the maximum power output of each PV module given certain irradiance condition. The tracking efficiency of a PV system depends on not only the power conversion efficiency of the power electronics circuit but also how accurately the power electronics converter matches the MPP impedance (i.e., the effectiveness of the MPPT). In other words, losses in PV system can result from the conversion losses in the power electronics circuits and from not operating PV modules exactly at their MPPs. The design goal of power electronics in PV system is therefore to achieve high conversion efficiency and precise MPPT while maintaining low cost.

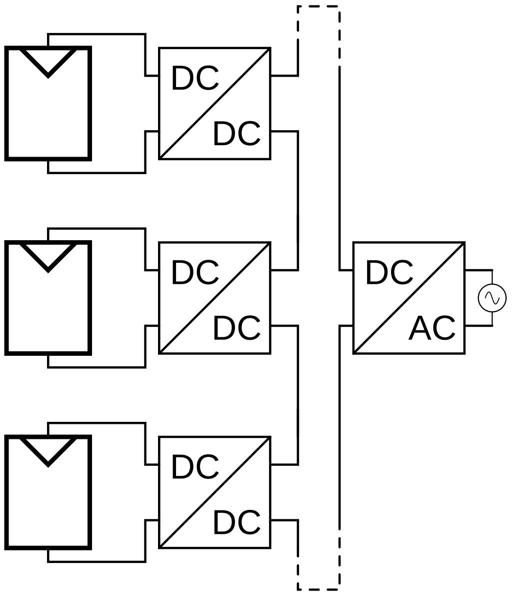
2.2 Conventional Power Electronics Solutions in PV System

In a PV system, the basic form of power electronics is a high-voltage central inverter that converts DC into AC, as shown in Fig. 2.5a. PV modules are connected in series to stack a high enough voltage for the efficient operation of the central inverter, and typically a MPPT algorithm is implemented on the central inverter to maximize the power output from the PV string. However, series connected PV modules often suffer from the well-known current mismatch problem [5,6]. That is, PV modules often have different I-V characteristics due to reasons like manufacturing variability, non-uniform aging, and mostly importantly, partial shading of the PV string (I-V curves can be quite different under different environment condition, as shown in Fig. 2.1). Since all PV modules in Fig. 2.5a are connected in series, they share exactly the same string current, and then the entire string is therefore limited by the weakest PV module (i.e., the module with the lowest I-V curve, such as the 50% irradiance curve in Fig. 2.2 is the weakest of the three) if there is a mismatch. Bypass diodes are often connected in parallel with PV modules to mitigate this limitation and to prevent thermal runaway. However, the power loss due to current mismatch is still very significant since all PV modules in the string cannot operate at their MPPs. Moreover, the conduction of bypass diode creates multiple local maxima in the PV string P-V characteristics, which might trap the MPPT of the central inverter. Therefore, although the central inverter has high power conversion efficiency (typically above 98%), the power losses due to current mismatch in the PV string can be very severe, and the tracking efficiency of the system can be low.

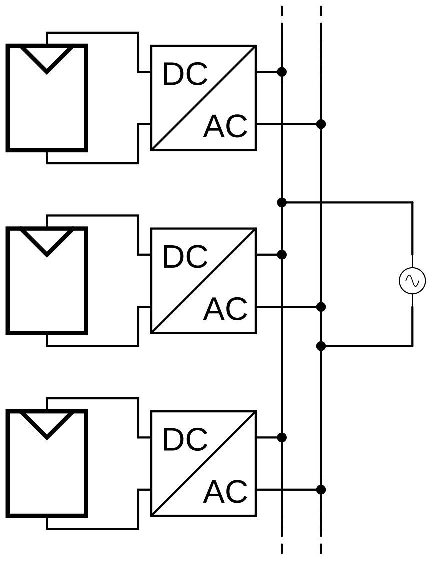
The current mismatch problem of PV string motivates the development of distributed power electronics that perform module level or even sub-module level MPPT. There are two mainstream solutions nowadays, namely micro-inverters [7,8], as in Fig. 2.5c, and DC optimizers [9–13], as in Fig. 2.5b. Micro-inverters are low-voltage inverters that interface directly with each PV



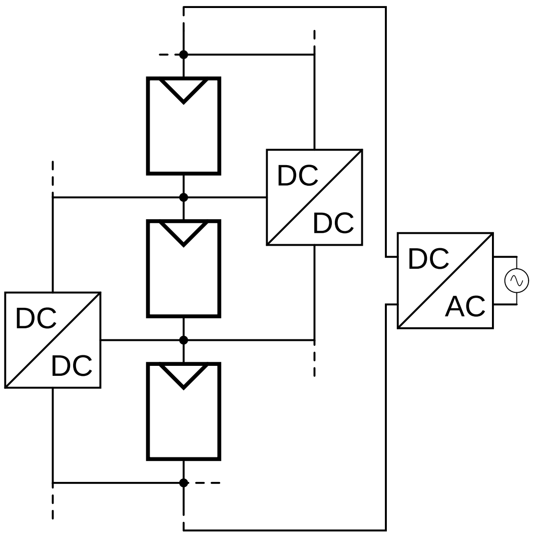
(a) Central inverter



(b) DC optimizer system



(c) Micro-inverter



(d) DPP system

Figure 2.5: Four types of power electronics solutions for PV systems.

module instead a PV string, and therefore can perform MPPT on each individual PV module. Micro-inverters offer great flexibility in terms of system scale, so they have become the dominant solution in residential PV system. Despite its capability to minimize loss due to PV module current mismatch, the primary problem with the micro-inverter solution is the relatively high cost and low conversion efficiency (typically $93\% - 95\%$) compared to a high-

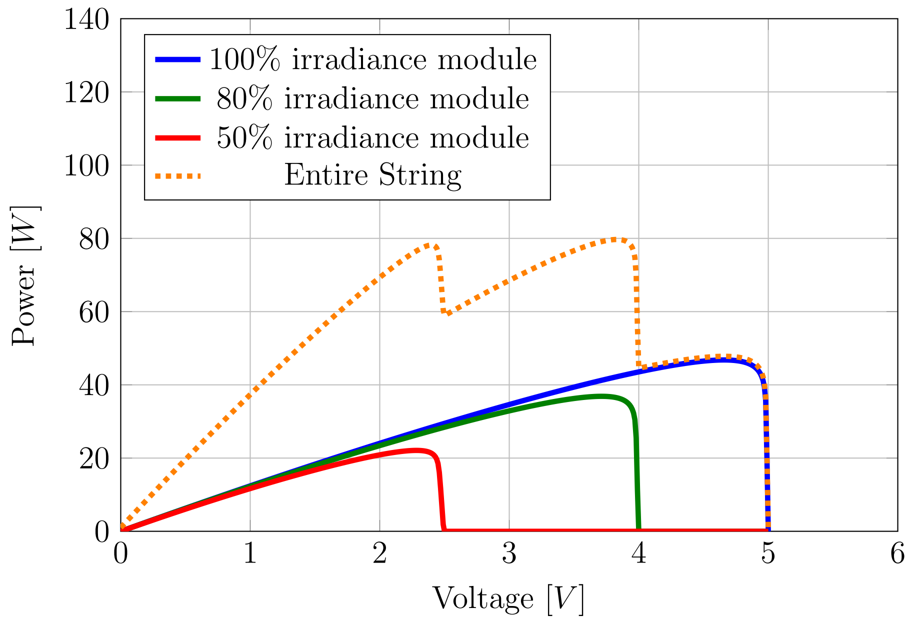


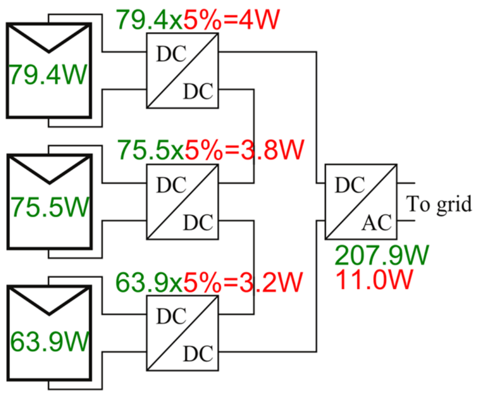
Figure 2.6: P-V characteristics of a PV string with non-uniform irradiance.

voltage central inverter.

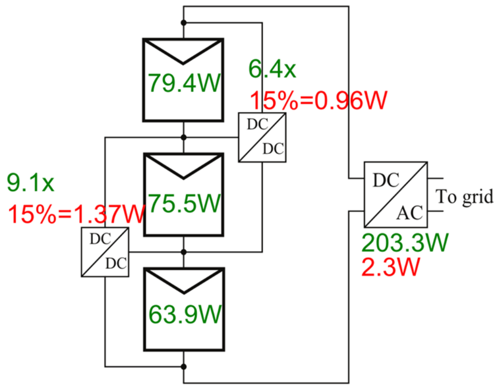
DC optimizer systems, on the other hand, are more suitable for commercial and utility scale systems. A DC optimizer system keeps the series PV module connection but inserts a DC/DC converter between each PV module and the string as shown in Fig. 2.5b. Each DC optimizer processes the power generated by the corresponding PV module and adjusts the converter duty ratio to perform MPPT. The entire string is still connected to a central inverter to convert DC power into AC. The major drawback of a DC optimizer system is that bulk energy is processed twice - first by the DC/DC stage and then by the central inverter. Even if the DC/DC stage is very efficient (typically $95\% - 97\%$), because it is processing the whole power of PV modules, the power loss in the DC/DC stage is still significant. In other words, the efficiency of DC optimizer system is limited by the DC/DC stage.

2.3 Advantage of Differential Power Processing Architecture

Since the converter efficiency limits the PV system efficiency, we would ask the question: is it possible to overcome this limitation, decouple the system



Loss: 11W
DC stage efficiency: 95%
(a) DC optimizer



Loss: 2.3W
DC stage efficiency: 98.9%
(b) DPP

Figure 2.7: Comparison of DC stage efficiency of DC optimizer system and DPP system. 95% efficiency is assumed for all DC/DC converters. The red text shows the total power processed by each converter. The violet text shows the power loss as a result of power conversion.

efficiency with converter efficiency and achieve even higher system efficiency? The answer is yes, and this can be achieved through the differential power processing structure as shown in Fig. 2.5d. An overview of the DPP concept can be found in [14]. In this structure, DC/DC converters are configured as bi-directional converters and connected in parallel with the PV string. The bulk power that is common to all PV modules goes to the central inverter directly without any intermediate conversion. The converters only shuffle the mismatch current to perform MPPT on each PV module, and thus process only the differential power, which is typically just a small fraction of the bulk power. Because the power processed by each converter is much smaller, with the same converter efficiency, power losses can be significantly reduced. In an ideal case, if no mismatch exists between PV modules, DC optimizers still need to process the whole power and will incur significant power losses, while in this case DPP converters will process zero mismatch power, and thus cause no loss at all (although in practice there will still be a small power overhead to control the converters). Therefore, the overall efficiency is no more limited by the DC/DC converters, and can approach the efficiency of central inverter (98%). Fig. 2.7 provides an example of a three element system to compare the efficiency of the DC stage in DC optimizer system and DPP system.

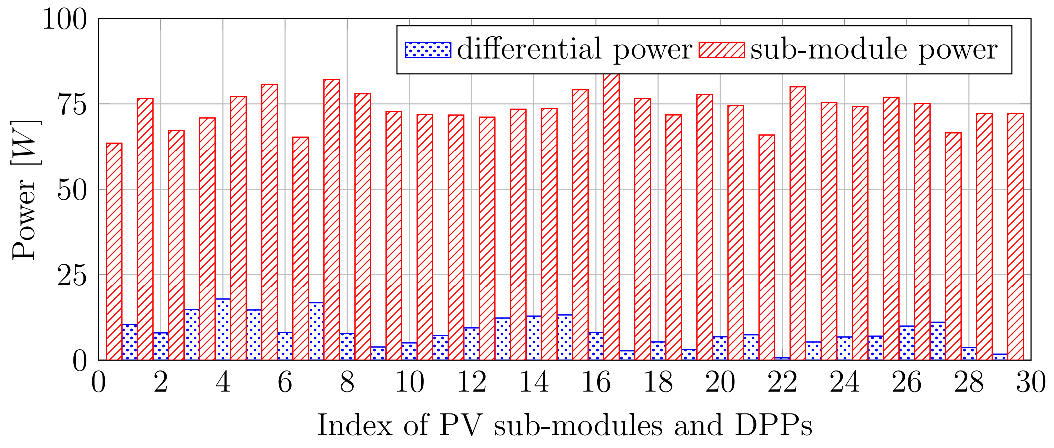


Figure 2.8: Maximum power of PV sub-modules and differential power of a 30 PV sub-module system with randomly generated irradiance condition.

In order to compare the efficiency of a DPP-based system with micro-inverters or DC optimizers in a more general case, we consider a system with 10 PV modules (i.e., 30 PV sub-modules) as an example. The irradiances on the PV sub-modules are set randomly by drawing from a Gaussian distribution with a standard deviation equal to 10% of the mean; the resulting maximum power of each PV sub-module is displayed in Fig. 2.8 using striped red bars. The differential power, i.e., the power difference between the maximum power of each PV module and the power common to all PV modules, is also shown in Fig. 2.8 using dotted blue bars. In a DC optimizer-based system, the DC optimizers have to process the full power of all PV sub-modules (represented by the striped red bars), which adds up to 2218 W. In a DPP-based system, each DPP only needs to process the differential power (represented by the dotted blue bars), which only adds up to 242.5 W. Both the DC optimizer-based system and the DPP-based system then use a string-level inverter to convert DC string voltage into AC voltage. In a micro-inverter system, the PV module voltage is directly converted to AC, but the efficiency of a micro-inverter is typically lower and its per-watt cost is typically higher than that of a string-level inverter or a central inverter. Table 2.1 summarizes the power losses of these three different solutions for the irradiance condition in this example. DC optimizers are assumed to have an average efficiency of 96%, while that of DPP converters is assumed to be 92% since they often operate in light-load conditions. For both the DC optimizer and DPP-based system, a string inverter efficiency of 98% was as-

Table 2.1: Comparison of System Efficiency of the DC Optimizer-Based System, the Micro-inverters and the DPP-Based System

Losses \ Solutions	DC optimizer	Micro-inverter	DPP
Power processed by DC stage [W]	2218	N/A	242.5
Intermediate power losses [W]	$2218 \times 4\% = 88.7$	N/A	$242.5 \times 8\% = 19.4$
Power processed by inverter(s) [W]	$2218 - 88.7 = 2129.3$	2218	$2218 - 19.4 = 2198.6$
Inverter power losses [W]	$2129.3 \times 2\% = 42.6$	$2218 \times 5.0\% = 110.9$	$2198.6 \times 2\% = 44.0$
Overall efficiency	94.08%	95.00%	97.14%

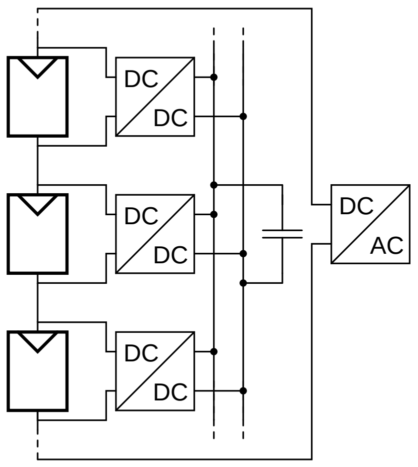
sumed. The micro-inverters are assumed to have an efficiency of 95%, and since, in practice, they only interface with PV modules, and therefore cannot compensate mismatch between sub-modules, the effective efficiency is typically even lower. As is evident from Table 2.1, the DPP-based system yields significant advantages in terms of overall system efficiency.

2.4 Variations of DPP Architecture

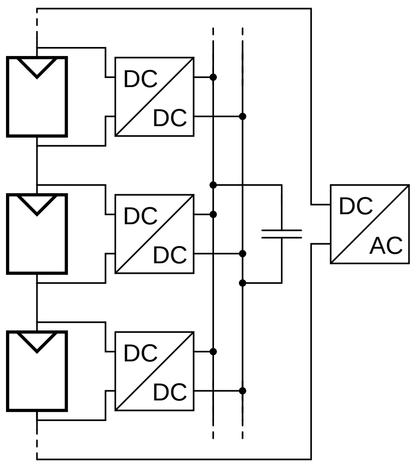
As mentioned in Section 2.6, various topologies exist under the framework of differential power processing. Figure 2.9 illustrates three topologies that have been studied the most. Element-to-bus topology and element-to-virtual-bus topology transfer energy from each PV module directly to the bus or to each other, while element-to-element topology shuffles energy between neighboring PV modules. As demonstrated in [14], element-to-bus topology and element-to-virtual-bus topology tend to process less power than the element-to-element topology. However, power converters in the former two topologies need to be isolated with a transformer, which will compromise the conversion efficiency and hardware size. In contrast, element-to-element topology allows for non-isolated converters with which high conversion efficiency and small converter footprint can be achieved. For the above reason, the focus of this thesis is going to be on element-to-element topology.

2.5 Operation of Element-to-Element Topology

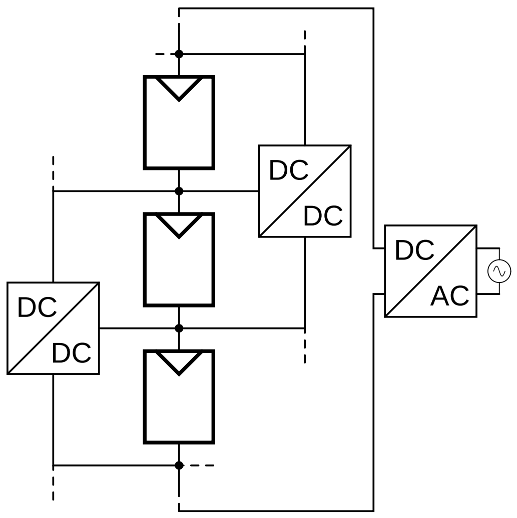
The operation of the element-to-element structure is complicated by the fact it has coupled states between neighboring converters. Fig. 2.10 depicts the



(a) Element-to-bus topology



(b) Element-to-virtual-bus topology



(c) Element-to-element topology

Figure 2.9: Three major topology variations of DPP architecture.

first four elements and three DPP converters in a N element, $N-1$ converter string of element-to-element structure. Each converter stage is implemented as a bidirectional buck-boost converter. The function of the converter stage is to enforce a desirable voltage ratio between neighboring elements and conduct the differential current (which is the current difference between the current of each element and the current common to the entire string, I_{string}). The duty ratio of each converter can be determined by the desirable voltage ratio

$$D_i = \frac{V_i}{V_i + V_{i+1}} \tag{2.2}$$

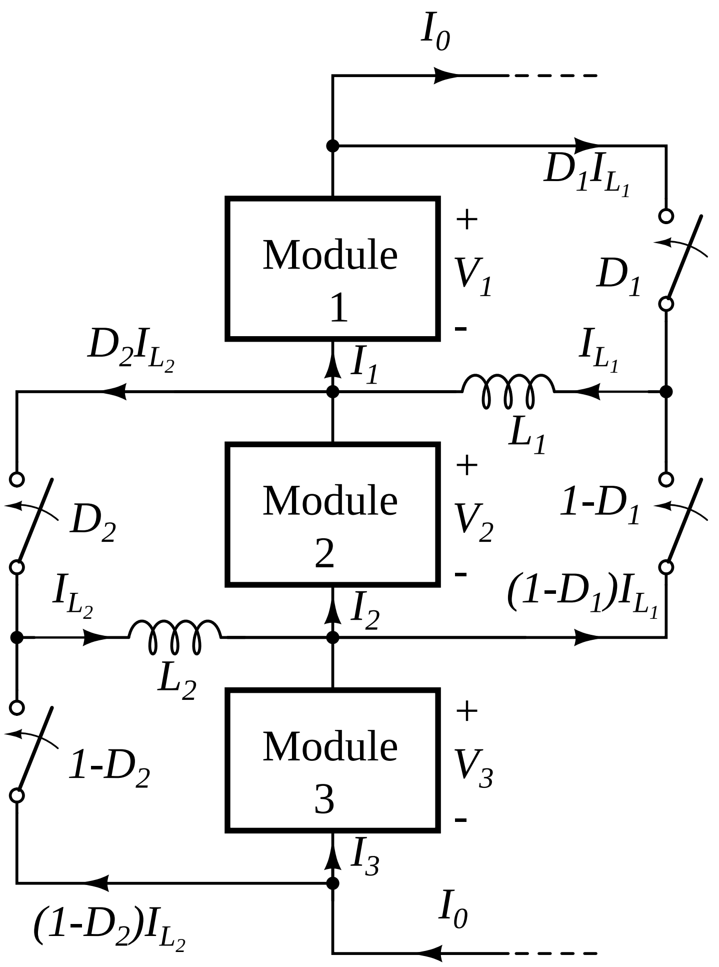


Figure 2.10: Steady state analysis of element-to-element buck-boost architecture.

assuming ideal operation of buck-boost converter. Apply KCL at each intermediate node between the i^{th} and the i^{i+1} element (e.g., node C between the 2^{nd} and the 3^{rd} element in Fig. 2.10) and we can get

$$I_{L_i} - (1 - D_{i-1})I_{L_{i-1}} - D_{i+1}I_{L_{i+1}} - I_i + I_{i+1} = 0 \tag{2.3}$$

Note that in this analysis we only consider a steady state average model that excludes all the circuit dynamics. The first and last intermediate node (i.e., node B in Fig. 2.10 and the node between the $N-1^{th}$ and N^{th} element that is not shown in Fig. 2.10) need is slightly different from other nodes since one less converter is connected to these two nodes.

$$I_{L_1} - D_2 I_{L_2} - I_1 + I_2 = 0 \tag{2.4}$$

Finally, the string current, which is the common mode current to all the elements, can be calculated by KCL at node A in Fig. 2.10.

$$-I_{string} - D_1 I_{L_1} + I_1 = 0 \tag{2.5}$$

By collecting all the node KCL equations in one matrix, the entire DPP system can be described by

$$\begin{bmatrix} 1 & D_1 & 0 & \cdots & \cdots & \cdots & 0 & 0 \\ 0 & 1 & -D_2 & 0 & \ddots & \ddots & \vdots & \vdots \\ 0 & -(1-D_1) & 1 & -D_3 & 0 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & -(1-D_{N-4}) & 1 & -D_{N-2} & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & -(1-D_{N-3}) & 1 & -D_{N-1} \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & -D_{N-2} & 1 \end{bmatrix} \begin{bmatrix} I_{string} \\ I_{L_1} \\ I_{L_2} \\ \vdots \\ I_{L_{N-3}} \\ I_{L_{N-2}} \\ I_{L_{N-1}} \end{bmatrix} = \begin{bmatrix} -I_1 \\ I_1 - I_2 \\ I_2 - I_3 \\ \vdots \\ I_{N-3} - I_{N-2} \\ I_{N-2} - I_{N-1} \\ I_{N-1} - I_N \end{bmatrix}. \tag{2.6}$$

The irradiance on each PV module can be considered random. Therefore, for each irradiance condition, as illustrated in Fig. 2.1, the PV module will have a unique output curve and maximum power point (V_{max}, I_{max}). With proper control algorithm (which will be introduced in the following two chapters), each element can operate at its maximum power point, which determines all V_i and I_i $(i \in [1, N])$ in Eq. 2.6. The duty ratios that correspond to these maximum power points are given by Eq. 2.2. Then Eq. 2.6 can be solved for a unique solution and the entire system is fully determined. The power processed by the i^{th} DPP converter can be calculated as

$$P_i = D_i I_{L_i} V_i \tag{2.7}$$

2.6 Previous Research on Differential Power Processing

To apply the concept of differential power processing to PV systems, especially at the sub-module-level, several architectures and corresponding control schemes have been proposed [2, 15–21]. For example, the work presented in [15, 18] uses a switched inductor topology; the work presented in [16] uses a resonant switched capacitor topology; the work presented in [17] uses a transformer coupled topology. All of these proposed solutions adopt a control method commonly known as voltage equalization or "virtual parallel" operation, in which all distributed converters strive to equalize the voltages of all PV sub-modules in a string. The distributed converters in the system run in open loop, simplifying the control requirements while obviating the need for communication among converters. However, the voltage equalization approach can only achieve *near* maximum power point (MPP) operation without truly seeking the MPP of each individual sub-module. The effectiveness of this approach relies on the fact that the MPP voltages of sub-modules in a string are very close even if their MPP currents differ significantly. This does require all the sub-modules to have very similar electrical characteristics, which in practice is typically guaranteed by a costly binning process performed by PV module manufacturers [22]. However, even modules that are carefully matched at installation will suffer from non-uniform degradation after several years of field exposure. As illustrated in [23], the standard deviation of PV module MPP voltages may increase by nearly four times over a twenty-year period. Furthermore, field operating conditions that result in thermal gradients or severe irradiance mismatch along the PV string can also cause the MPP voltages of sub-modules to drift apart. All of these factors limit the tracking efficiency of the voltage equalization approach; thus, in order to increase the total energy harvested over the entire lifetime of the PV system, it is necessary to develop a scheme capable of true MPPT.

On the other hand, the generation control circuit presented in [19, 20] achieves true MPPT without any local current sensing at each sub-module, but employs a control scheme that requires communication between all converters and a central control unit. The central control unit has to (i) command all the DPP converters to exhaust every possible combination of converter duty ratio perturbations, and (ii) measure the string voltage respectively in sequence before making the next tracking step. For a system with n DPP converters, the central control unit has to try 2^n duty ratio perturbations during each tracking step, rendering the algorithm slow and infeasible for a large system. Moreover, with this approach, the reliability of the system is fundamentally limited, as a single failure of the central control unit, or a communication link failure, would result in the malfunction or complete loss of the overall system. Recently, a distributed MPPT approach for a DPP-based system was presented in [21], which significantly reduces the communication requirements. However, this approach requires measurements of all PV sub-module currents; this results in additional power losses. Moreover, this approach requires synchronization between power converters,

Table 2.2: Comparison of DPP Control Approaches

Approaches / Features	Virtual Parallel [16–18]	Centralized P&O [2, 19, 20]	Multilevel PPT [21]	This work
Tracking	Near MPPT	True MPPT	True MPPT	True MPPT
Distributed algorithm	Yes	No	Yes	Yes
Requiring local current sensing	No	Yes	Yes	No
Requiring communication	No	Centralized	Synchronization	Neighbor-to-Neighbor

which still relies on some communication.

Table 2.2 provides a comparison of some key features of DPP control approaches presented in previous work and the one proposed in this paper. The information in this table implies that for DPP converters, either communication or local current measurements are needed to acquire adequate information to perform true MPPT operation. Implementing local current measurement impairs the system efficiency and increases the hardware cost, whereas implementing communication may result in a much smaller impact because communication hardware is often required for other purposes, including individual PV module diagnostics and on/off capability. Therefore, the suggested approach is to eliminate local current sensing while preserving communication. In this thesis, the approach of performing true MPPT utilizing communications between DPP converters is explored. A centralized communication scheme is developed. Moreover, a distributed approach that reduces the communication requirement to overcome the limitations of the centralized approaches is also developed.

CHAPTER 3

PV MODULE CHARACTERISTICS AND LABORATORY EMULATION

The foundation of developing a PV energy system is a good understanding of the PV module as well as an experimental setup that allows for controllable and repeatable PV experiments. In this chapter, the PV module characteristics are discussed in depth and based on that knowledge, a high fidelity, easy-to-implement photovoltaic (PV) module emulator is presented. The proposed emulator can replicate the electrical behavior of a sunlight illuminated PV module in an indoor environment. The construction of this emulator requires only a PV module and basic laboratory equipment, while still providing dynamic performance that closely matches that of an illuminated PV module in an outdoor environment. The output I-V characteristics of the PV module under real sunlight and that of the proposed emulator were experimentally obtained and compared, such that the functionality of the proposed emulator was verified. An in-depth analysis of the PV module output small-signal impedance is also presented to illustrate the dynamic performance of this emulator.

3.1 Motivation

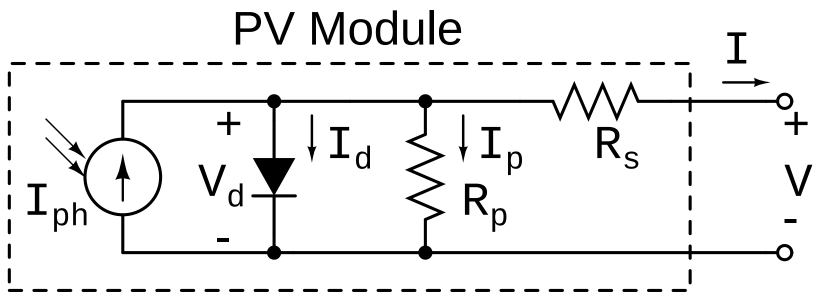
During the development of such power electronics systems, researchers often need to perform hardware experiments with real PV modules under sunlight to evaluate their circuit and control design. However, the output I-V characteristics of a PV module are a strong function of insolation and temperature on the module surface. Because insolation and temperature are constantly changing in outdoor environments, outdoor experiments of real PV modules are uncontrollable, laborious to conduct, and highly dependent on the weather conditions. It is more desirable to perform PV module experiments in a controllable manner and in an indoor environment. For this purpose, many hardware systems that reproduce the electrical characteristics of a PV module, referred to as a PV module emulator or simulator, have been proposed [24–28]. Similarly, there are a number of commercial laboratory instruments that aim to replicate PV module static I-V curves, often at a high cost [29, 30].

Reference [24] introduces an I-V magnifier, which magnifies the I-V output of a small pn photo-sensor and generates large output power based on analog circuitry. References [25–28] all use switching dc/dc converters to generate the output and use the I-V curves of the PV module as a signal reference to control the power converter. In [25], reference I-V curves were obtained through field measurements of entire I-V curves of real PV modules, while in [26, 27] I-V curves were obtained by computer simulation based on key parameters (open-circuit voltage, short-circuit current, maximum power point, etc.) from field measurement or manufacturer datasheet [31]. However, all these proposed solutions depend either on prior field measurements, in which case a certain insolation condition must be captured in the outdoor environment before one can emulate that condition, or on complicated computer simulations. Moreover, while all of these previous works as well as commercial instruments focus on a closely matched output I-V curve, few of them consider matching the dynamic performance. In fact, all of these solutions involve complex design and implementation, and the dynamic performance of these emulators is fundamentally limited by the switching converter and control circuitry.

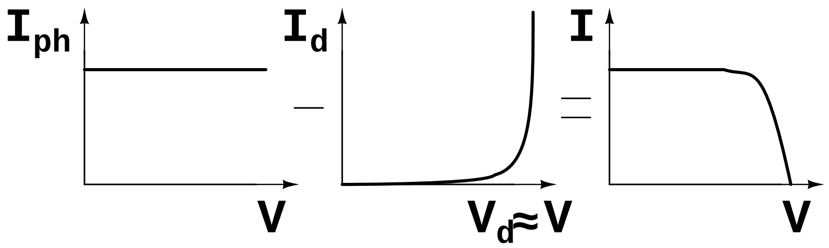
In this section, we present a PV module emulator solution that allows very simple hardware implementation and no prior simulation or field measurement, while still providing closely matched I-V curves and dynamic performance to a sunlight-illuminated PV module.

3.2 Model of PV Module

Although there are many variations of models for PV modules, the single-diode model with series and parallel resistance typically provides enough accuracy for power converter design and is adopted by many researchers [32]. The equivalent circuit of this model is shown in Fig. 3.1a. The output current



(a) Equivalent circuit schematic.



(b) Current relation.

Figure 3.1: Single-diode model of PV module.

I and output voltage V of a PV module can be derived as

$$\begin{aligned} I =& I_{ph} - I_d - I_p \\ =& I_{ph} - I_0 \left[exp\Big(\frac{V_d}{V_t}\Big) - 1 \right] - \frac{V_d}{R_p} \\ =& I_{ph} - I_0 \left[exp\Big(\frac{V + R_s I}{V_t}\Big) - 1 \right] - \frac{V + R_s I}{R_p} \end{aligned} \tag{3.1}$$

where I_{ph} is the photocurrent generated by light hitting the PV panel, I_d and V_d are the equivalent diode current and voltage, respectively, and I_p is the current through the equivalent parallel resistor. I_0 is the dark saturation current that depends on the temperature and V_t is the thermal voltage, which shows the dependency of the output characteristics on temperature. R_s and R_p are the equivalent series and parallel resistance, respectively. A detailed derivation of this equation can be found in reference [32].

The single-diode model provides a good intuition of the PV module I-V characteristics. The photocurrent I_{ph} is directly proportional to the insolation, and can be viewed as a constant current source under a stable insolation condition. The diode current I_d is the current through the equivalent diode,

Table 3.1: Test PV Module Specifications

Open Circuit Voltage	12 V
Short Circuit Current	2.5 A
Maximum Power Point Current	2.3 A
Maximum Power Point Voltage	9.5 V

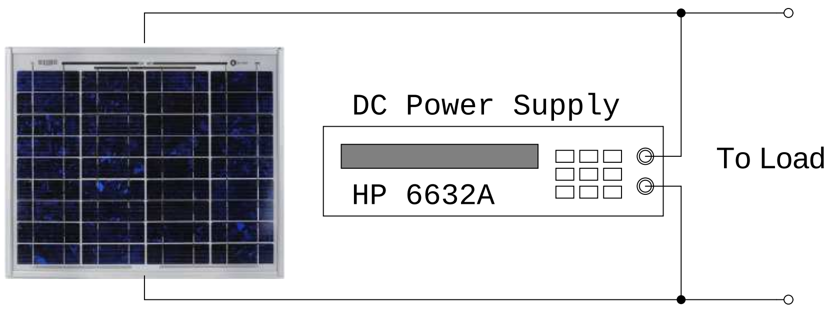
and its value is non-linearly dependent on the voltage V_d across it. At the same time, V_d depends on the output voltage as well as the voltage drop across R_s. This results in the overall PV module I-V characteristics being non-linear and lacking a closed-form expression.

The current sunk by the equivalent parallel resistor, I_p, is typically a small value, so it is ignored in this analysis for simplicity. The net output current of the PV module equals the difference between I_{ph} and I_d, as described in Eq. 3.1 and illustrated in Fig. 3.1b. The emulator is based on the idea that the PV module itself provides the non-linear characteristics, while the photocurrent can be provided using a constant current source that does not depend on illumination.

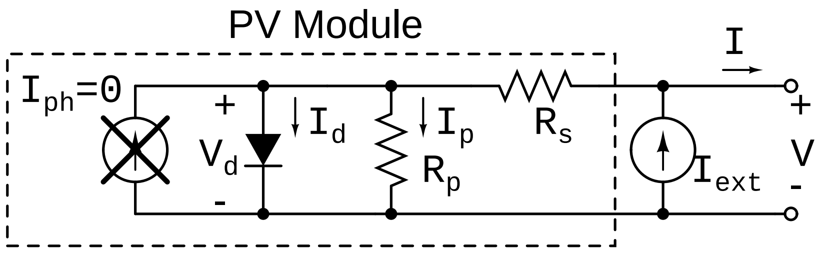
3.3 Proposed Emulator

Inspired by the above analysis of the model, we propose a method of emulating the illuminated PV module by connecting an external current source in parallel with the PV module to emulate the photovoltaic current I_{ph}, as shown in Fig. 3.2a. The accompanying equivalent circuit schematic is shown in Fig. 3.2b. When the PV module is completely covered to prevent any light contact, the photovoltaic current I_{ph} is zero. By externally supplying a current I_{ext} equivalent to I_{ph}, an I-V curve similar to an illuminated PV module can be obtained.

To verify this idea, an experiment was conducted on a SPI-020M-9.5 engineering evaluation prototype module from Solar Power Industries Inc. and its parameters obtained from the datasheet are listed in Table 3.1. A sourcemeter (Keithley 2420) was used as the load and performed a voltage sweep from 0 V to 13 V with 100 steps. The solar module was installed on the roof of Everitt Laboratory in Urbana, Illinois, under natural insolation and output sweeps were performed and recorded under different insolation levels



(a) Hardware connection.



(b) Equivalent circuit schematic.

Figure 3.2: Proposed simple emulator solution.

and partial shading conditions. For the controlled laboratory experiment, we completely covered the PV module and connected an external power supply (HP 6632A) operating in current limit mode as in Fig. 3.2a. The current limit of the power supply was set to the short-circuit currents previously measured during the output sweep, and the voltage limit was set to 15 V (voltage limit could be set to any value larger than the PV module's open circuit voltage). The same sweeps were performed by the sourcemeter on this emulator system in the same environment temperature. The I-V curves of the real PV module and the emulator were plotted and compared. Two examples are shown in Fig. 3.3 corresponding to a full insolation condition and a partial shading condition, respectively.

It can be seen from Fig. 3.3 that the emulator provides an I-V curve of very similar shape to that of the real PV module, except for a small offset (about 0.5V) in voltage when the output voltage is above a certain value. This voltage offset can be explained as follows: When the PV module is illuminated by stable sunlight, as in Fig. 3.1a, the photovoltaic current I_{ph} is the only current source and it feeds current through R_s to the output, such

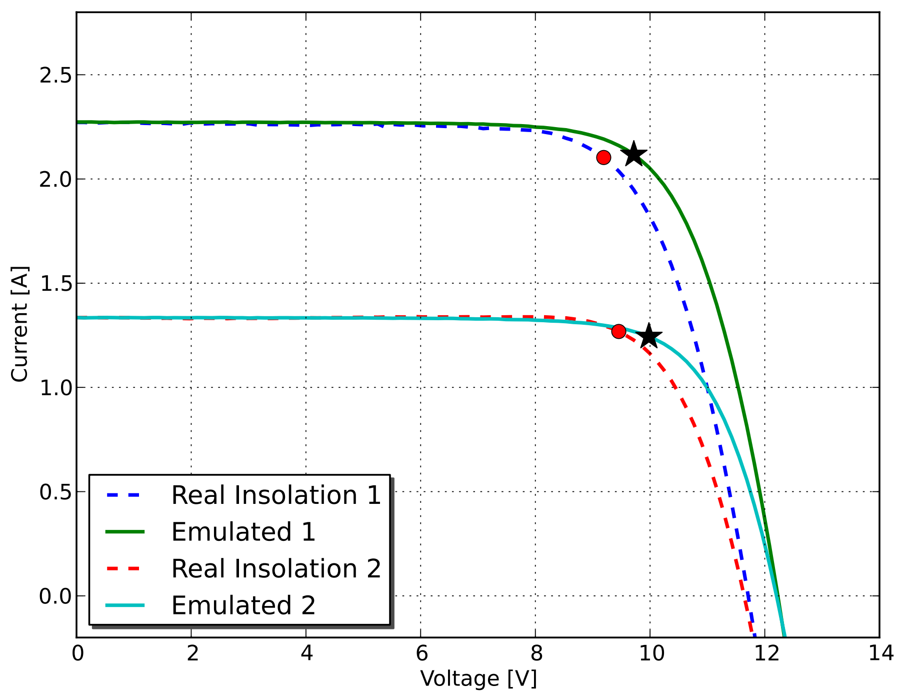


Figure 3.3: I-V curve of PV panel under natural insolation and with simple emulator for full insolation condition and partial shading condition. Maximum power points of real PV module are marked with dots and maximum power points of emulator are marked with stars.

that the voltage across the equivalent diode is given by

$$V_{d(ph)} = V + R_s I \tag{3.2}$$

However, when the current is supplied by the external current source, as in Fig. 3.2b, current goes through R_s into the panel, so the voltage across the equivalent diode in this case is given by

$$V_{d(ext)} = V - R_s I \tag{3.3}$$

The output current and voltage of this emulator, comparable to Eq. 3.1, can be described by

$$
\begin{aligned}
I =& I_{ph} - I_d - I_p \\
=& I_{ph} - I_0 \left[exp\Big(\frac{V_d}{V_t}\Big) - 1 \right] - \frac{V_d}{R_p} \\
=& I_{ph} - I_0 \left[exp\Big(\frac{V - R_s I}{V_t}\Big) - 1 \right] - \frac{V - R_s I}{R_p}
\end{aligned}
\tag{3.4}
$$

When the output voltage V is small, the diode voltage V_d is also small. It can be seen from Fig. 3.1b that the diode is in the "flat region" of its I-V curve. It sinks negligible current, which is only a fraction of the output current and is not sensitive to the change in V_d. As a result, the emulator I-V curve matches that of a real PV module very well at low output voltage range. As the output voltage V increases, so does the diode voltage V_d, and the diode will eventually enter the "steep region" of its I-V curve. In this region, the diode current I_d is very sensitive to the change of V_d. The difference between $V_{d(ph)}$ and $V_{d(ext)}$ causes a relatively large difference in I_d, and thus a large difference in output current. This explains the relatively large mismatch when the output voltage is above a certain value.

Despite this mismatch of the absolute value, this emulator provides an I-V curve of almost identical shape to that of the sunlight illuminated PV module. For many applications, the power converter is designed to work with different types of PV modules and the exact value of maximum power point is not critical, so this emulation result is adequate in many cases. Compared with other emulator solutions, the emulator proposed here is very easy to implement with just basic lab equipment, and no other complex circuit or simulation program implementation is needed. Overall, this emulator provides a good alternative to previous emulation solutions in many applications.

It should be noted that the proposed emulator is easily scalable. It can be used to simulate a PV array, an individual module, a sub-module or even a single cell. Compared with solar array simulator such as [29], the proposed emulator provides more flexibility in simulating a real-world scenario. For example, to simulate a partial shading condition between sub-modules of a PV module, sub-modules with bypass diodes can be connected in series as shown in Fig. 3.4. Power supplies are set to different current limit values to represent the insolation that each sub-module receives. The resulting I-V

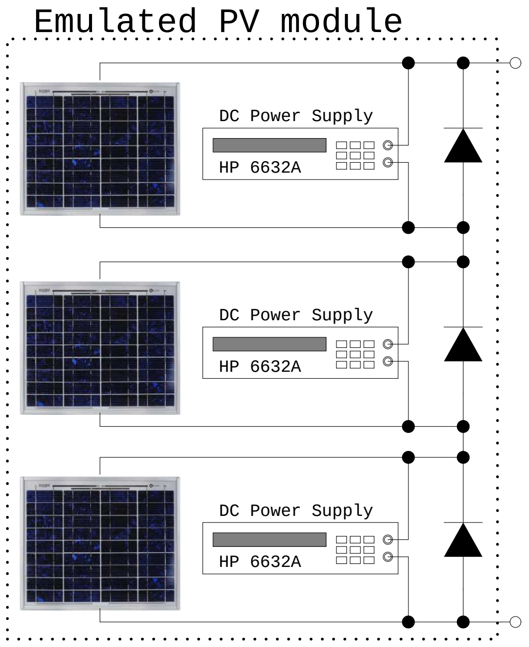


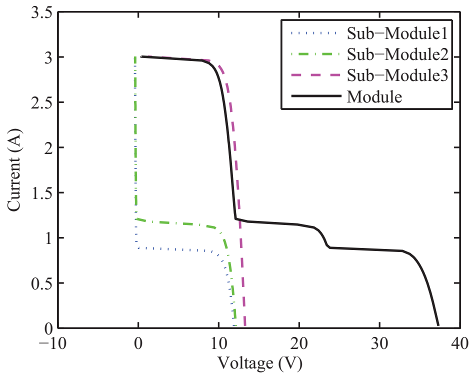
Figure 3.4: PV sub-module string connection to form a PV module.

characteristic is shown in Fig. 3.5a, which precisely captures the stair-like I-V curve caused by diode conducting. This method is also used in the experiments of [2] for sub-module differential power processing implementation.

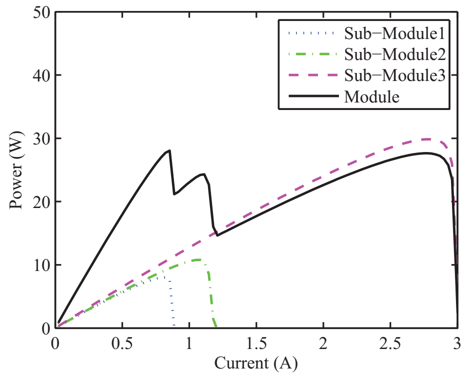
3.4 Dynamic Performance Analysis

Dynamics of the PV module are often an important factor in the design of power converters. For example, it is reported in [33] that the ripple-caused PV module power reduction predicted by first order small signal resistance approximation is typically an order of magnitude smaller than the actual power reduction due to the oversimplification of PV module dynamics by first order approximation. Previous PV emulator solutions, due to limited bandwidth, cannot replicate the dynamics of a PV module. However, a distinctive advantage of the proposed emulator over other solutions is its capability of preserving the original dynamics of a PV module. For research focusing on fast transient [34, 35], ripples and noise [33, 36, 37], etc., this emulator can be well suited.

To better understand the advantage of this emulator, a discussion of its dynamic performance through small-signal ac analysis is important. Ideally,



(a) I-V curve of the PV module in partial shading condition.



(b) P-I curve of the PV module in partial shading condition.

Figure 3.5: Partial shading condition simulation with the proposed emulator.

the dynamics of the emulated PV should be identical to that of the sunlight illuminated PV module; this would be true if the external current source is ideal so that it will exhibit infinite AC impedance, and thus connecting it in parallel with the PV module only provides a DC bias in place of the constant photovoltaic current but does not affect the AC impedance of the PV module at all. In practice, the quality of the constant current source, that is, the ability of the current source to hold a stable, constant current value in the presence of voltage fluctuation, determines how closely the emulator dynamics can be matched to an illuminated PV module.

To illustrate the effect of different current source implementations, Fig. 3.6 shows the measured small signal impedance magnitude over the frequency range for the illuminated PV module, emulated PV module with the HP 6632A as the current source, and emulated PV module with the Keithley 2420 as the current source, respectively. The small-signal magnitude is not well matched in the 100 Hz to 10 kHz range but is well matched above 10 kHz where the series inductance begins to dominate. This disparity between the illuminated and emulated PV modules is not surprising, since a power supply operating in constant current mode only approximates an ideal current source. Moreover, the large output capacitance typically present in power supplies affects the small signal impedance. As shown in Fig.3.6 the output capacitance varies according to design and manufacturer. The specifics of the power supply's internal feedback control may also have an effect on the

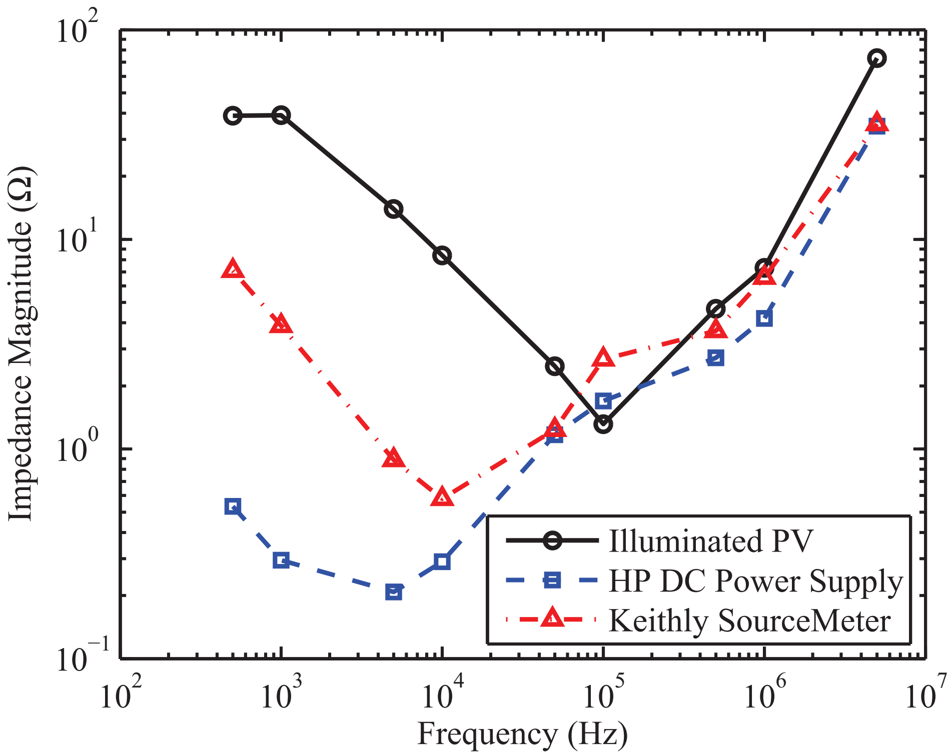


Figure 3.6: Ac characteristics for illuminated PV and emulated PV with two different supplies.

small-signal impedance, but it is difficult to characterize.

An inductor can be placed in series with the power supply to compensate for the disparate AC characteristics. The additional inductor increases the AC impedance of the parallel branch and helps the power supply act like a better constant current source. Fig. 3.7 illustrates how the impedance magnitude shifts up in the lower frequency range as the inductance is increased by increments of 75 mH inductors.

To further investigate how large of an inductance is needed to better match the illuminated PV module dynamics, we derive model parameter values from the measured data. Fig. 3.8 shows the models fit to the illuminated PV and emulated PV using the Keithley SourceMeter. The illuminated PV model has 1.3 Ω series resistance, 1.5 μH series inductance, 39 Ω equivalent parallel resistance, and 1.7 μF parallel capacitance. The power supply is modeled to have 50 μF output capacitance, 4 μH of inherent wire inductance, and 0.6 Ω wire resistance. Using these parameters, we model the effect of the inductance in series with the power supply current source. Fig. 3.9 shows

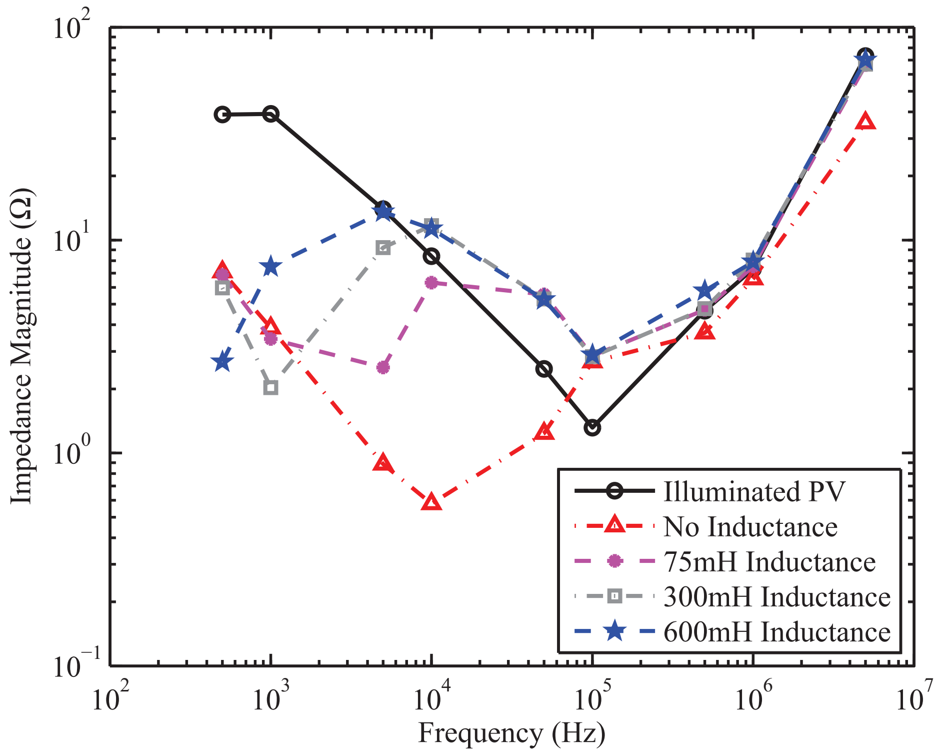


Figure 3.7: Ac characteristics for emulated PV as inductance is increased.

the emulated PV with 600 μH in series with the SourceMeter current supply and the model if the inductance were raised to 1 H; this large of an inductance provides a good match with the illuminated PV data. This result suggests that by adding more inductors, close match of dynamics can be achieved at even lower frequency than achieved in the previous experiment.

The inductance value required to match the illuminated PV module dynamics depends on the operating frequency of the application. For example, if a maximum power point tracking algorithm perturbs the system at 500 Hz, an inductance value should be chosen so that the small-signal ac characteristics are well matched at that frequency. For the Keithley SourceMeter setup, a 1 H inductor would provide well-matched characteristics at 500 Hz. Note that when selecting the inductor, care should be taken to make sure it does not saturate in the presence of the large DC current.

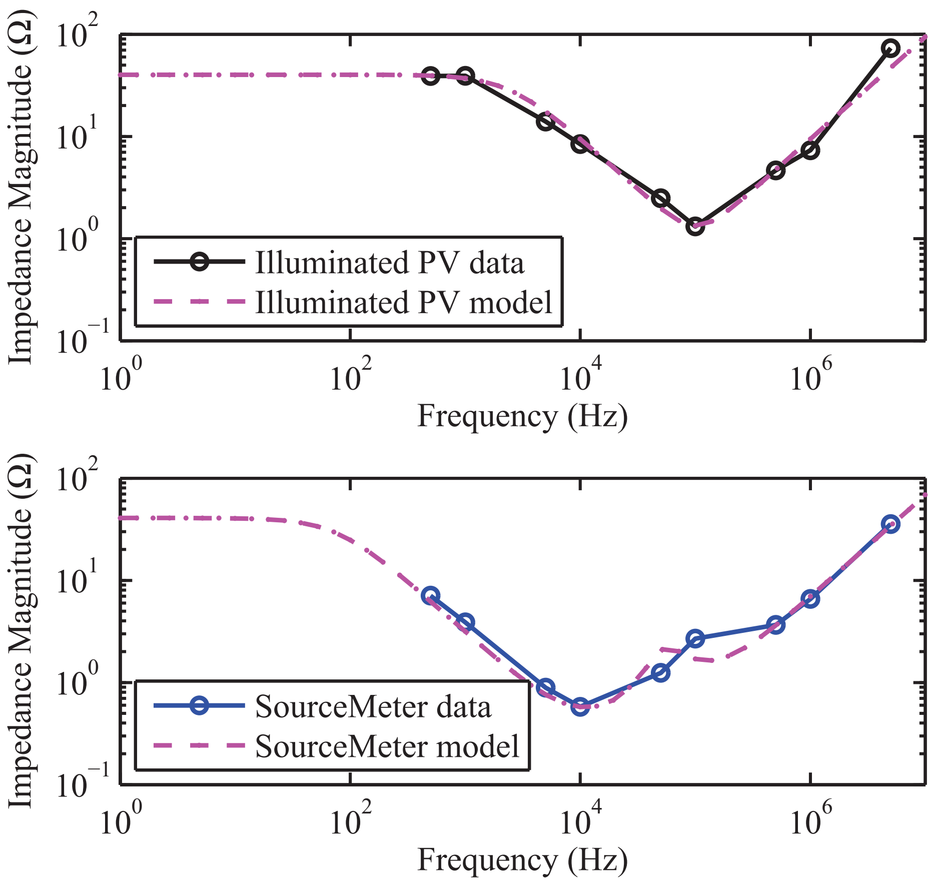


Figure 3.8: Model fit to illuminated PV, emulated PV with a power supply, and then in series with an inductor.

3.5 Modified Emulator

Although the emulator proposed above is suitable for many single module or sub-module level emulation scenarios, there are still a few applications, primarily micro-scale PV energy harvesting, that require very accurate emulation, especially around the maximum power point [38–40]. For better emulation accuracy, a modified emulator solution is proposed, as shown in Fig. 3.10. An external voltage source is inserted between the PV module and the external current source to increase the diode voltage V_d, and thus the diode will sink more current at a given output voltage, and the overall I-V curve of this emulator will be closer to that of a sunlight illuminated PV module. The value of V_{ext} can be set to a constant value, or controlled in real-time based on I_{ext} to better fit the desired I-V curve.

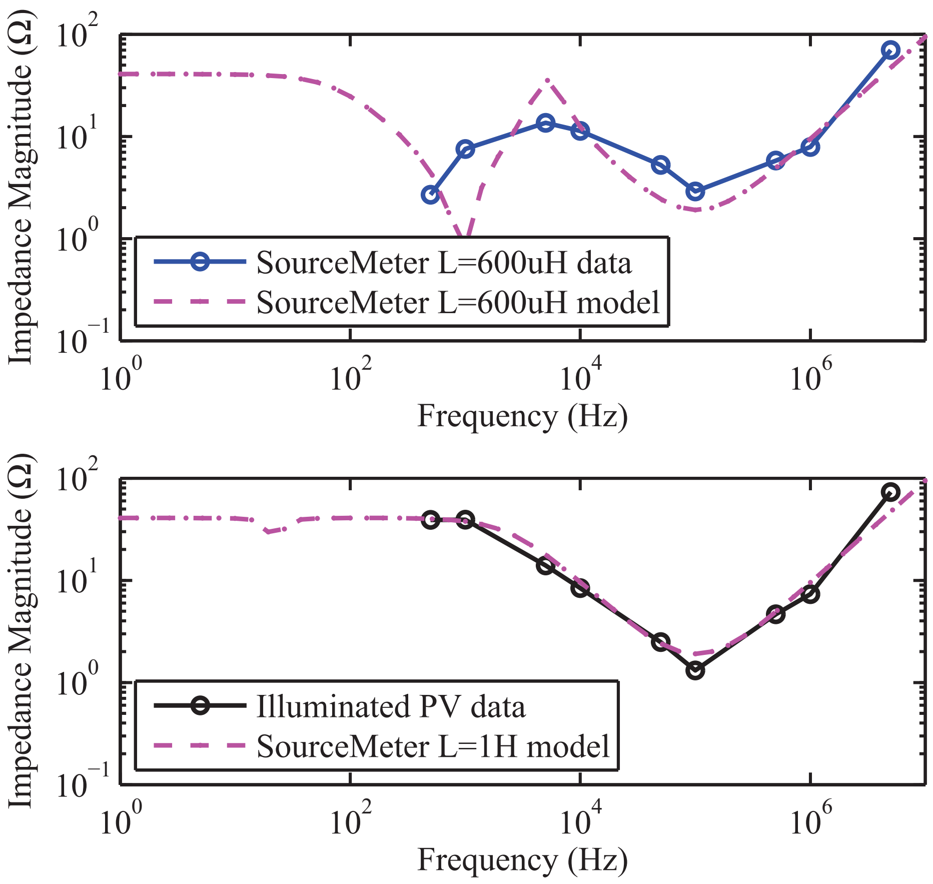


Figure 3.9: Model for emulated PV with very large inductance compared to illuminated PV characteristics.

The simplest method to acquire a good estimation of V_{ext} without field measurement is to use the datasheet. An I-V sweep of the simple emulator (as shown in Fig. 3.2a) can initially be performed with the parallel current source value set to the nominal short-circuit current in the datasheet. The MPP voltage of this sweep can be found and compared with the nominal MPP voltage in the datasheet, and V_{ext} in Fig. 3.10 equals the difference between these two voltages. This method still presents a small (approximately 0.2V) mismatch between the measured I-V curve of the sunlight illuminated PV module and the emulator in our experiment.

A more precise V_{ext} can be obtained as follows: first a field measurement is performed to capture an I-V curve of a sunlight illuminated module, and then the external current source value I_{ext} can be set to the measured short circuit current value to perform an emulation. The difference between the

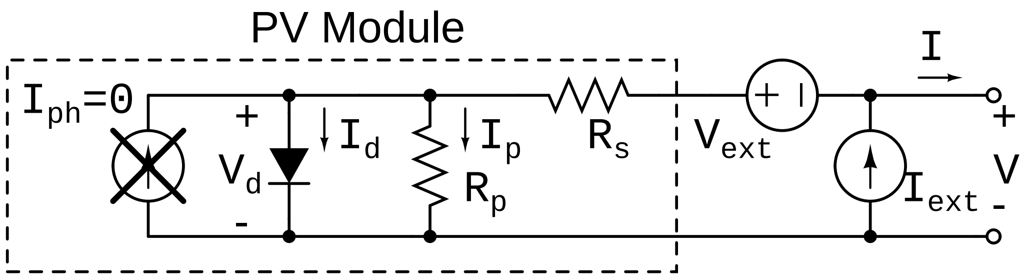


Figure 3.10: Proposed modified emulator solution.

MPP voltage of the field measured I-V curve and that of the emulated I-V curve can be calculated and used as the value of the voltage source. To verify this idea, the same experiment described in Section 3.3 was performed with an extra power supply (HP 6632A) to implement the voltage source. The value of V_{ext} is set to be constant as described above, and the resulting I-V curve is shown in Fig. 3.11, from which much improvement can be observed. The I-V curves as well as the maximum power point of the real PV module and the emulator are now very close to each other. Note that the difference between the illuminated module MPP voltage and the emulated module MPP voltage does not vary much with insolation level, and one measured voltage difference at a certain insolation level can be used to decide the value of V_{ext} for other insolation conditions, as was done for Fig. 3.11.

For even more precise I-V curve matching, more field measurements and comparison can be performed as described above for different insolation levels, and the values V_{ext} can be determined as a piecewise constant function of the current source I_{ext} and adjusted through a central control unit that controls both V_{ext} and I_{ext}. This can be implemented by a computer with communication link to both power supplies.

In practice, the voltage source V_{ext} is implemented by power supply in most cases. The introduction of the power supply will also impair the dynamic performance of the emulator due to the power supply's limited voltage regulating capability, or in other words, its finite response speed in the presence of the changing current through it. Similar to the method of adding series inductors to the current source, capacitors can be added in parallel to the voltage source. While this addition does not affect the DC characteristics, the small signal impedance of the voltage source can be made closer to zero

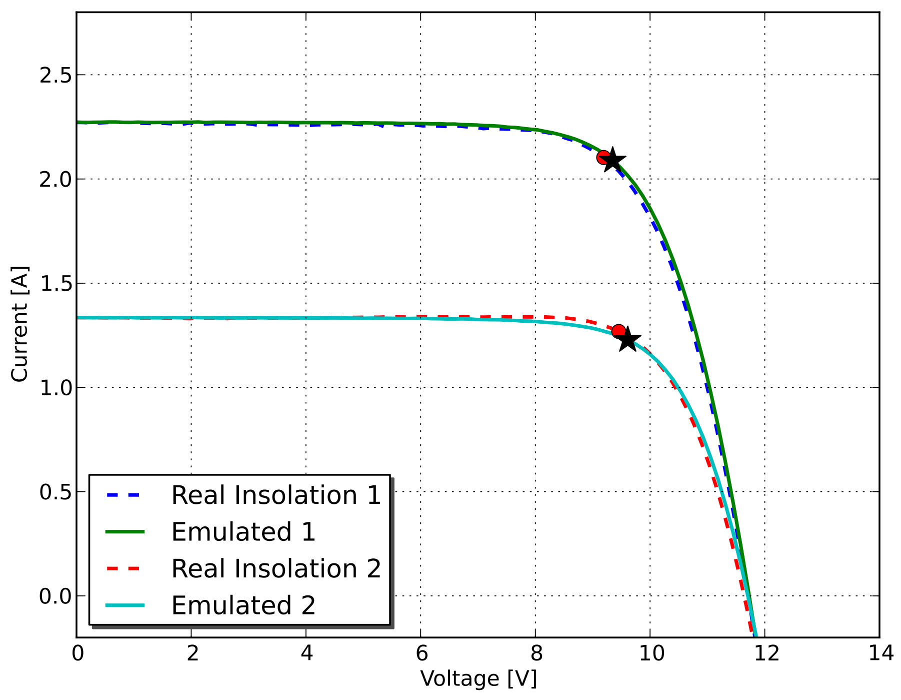


Figure 3.11: I-V curve of PV panel under natural insolation and with improve emulator for full insolation condition and partial shading condition. Maximum power points of real PV module are marked with dots and maximum power points of emulator are marked with stars.

to impose less change on the original PV module small signal impedance, and thus preserve the dynamic performance of the PV module.

CHAPTER 4

HARDWARE IMPLEMENTATION

In this chapter, the goals and considerations of DPP converter hardware implementation are introduced. Two DPP converter prototypes are described: one with DrMOS integrated power stage and one with discrete MOSFETs and gate drivers. Ancillary circuitry such as level shifter, current sensing circuitry, etc., is also described. Both hardware prototypes are characterized in terms of efficiency, volume, cost, etc. It should be noted that the two DPP converters are only different in implementation details and are interchangeable in terms of functionality.

4.1 DPP Hardware Design Requirement

One of the primary goals of distributed power electronics for PV systems, including DPP-based systems, is to achieve PV module integration [41]. Nowadays, the cost of separate enclosures for distributed power electronics represents a very significant portion of the total system cost, motivating efforts to reduce the converter footprint to fit into the existing weather-resistant junction box of an off-the-shelf PV module. Therefore, for sub-module DPP systems, the goal of the hardware design in this work is to achieve miniaturization of the DPPs for junction box integration, while maintaining high efficiency and capability to implement certain MPPT algorithms (as will be discussed in Chapter 5).

Figure 4.1 illustrates the wire connection of a DPP system; typically one PV module consists of three sub-modules. Therefore, each junction box needs to integrate three DPPs to perform module level mismatch balancing. Between adjacent junction boxes there are three wire connections: one series string connection in which the bulk power common to all sub-modules flows, one differential connection through which the DPPs shuffle the small power

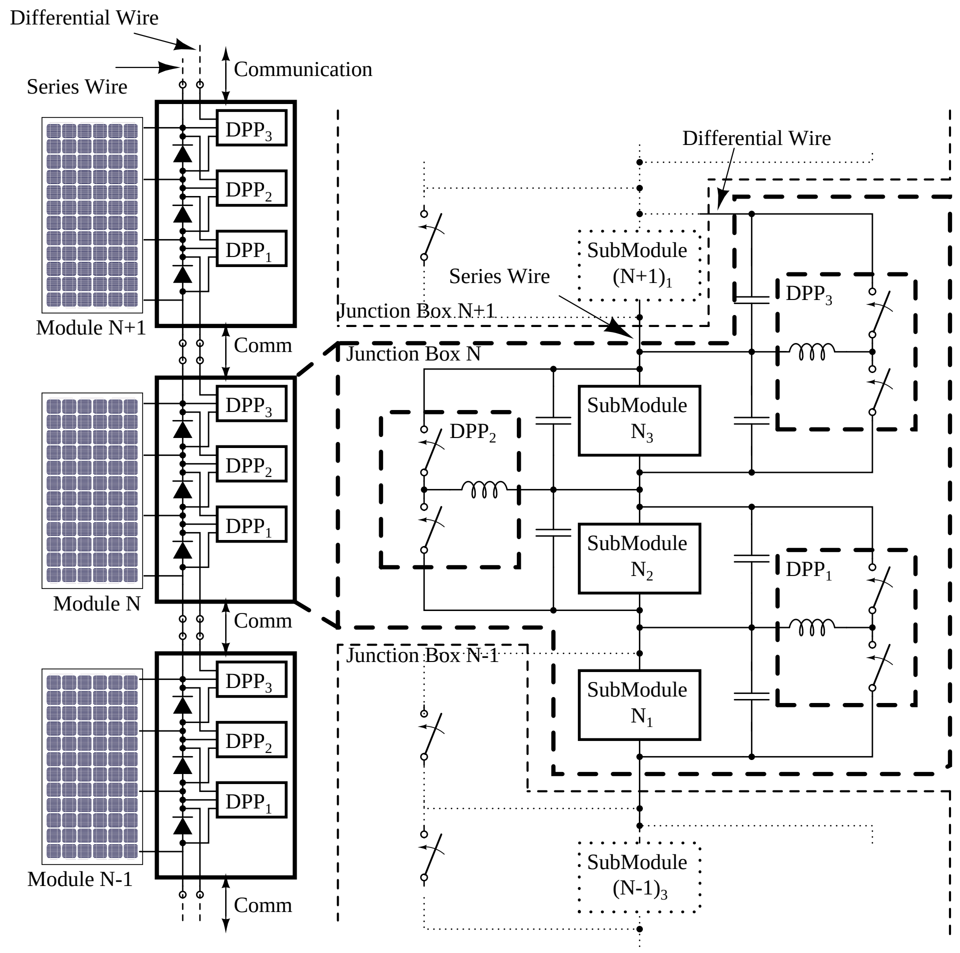


Figure 4.1: PV junction box connection for DPP system.

mismatch, and one or more wires for communication.

For the element-to-element topology studied in this work, DPP converters can be implemented as bidirectional synchronous buck-boost converters. In the element-to-element topology, each DPP converter needs only to be rated at twice the sub-module open circuit voltage (about 24V), which allows for the use of low-voltage, fast-switching transistors. By employing a high switching frequency in the order of hundreds of kHz, the size of the magnetic components, which typically dominate the converter size, can be significantly reduced. Furthermore, since the 3 DPPs in one junction box are physically very close, they can be integrated on one PCB and controlled by one micro-controller to save space and cost and to simplify wiring within one junction box. This also eliminates the need for communication between DPPs in one junction box and reduces the control overhead in terms of hardware cost and power consumption.

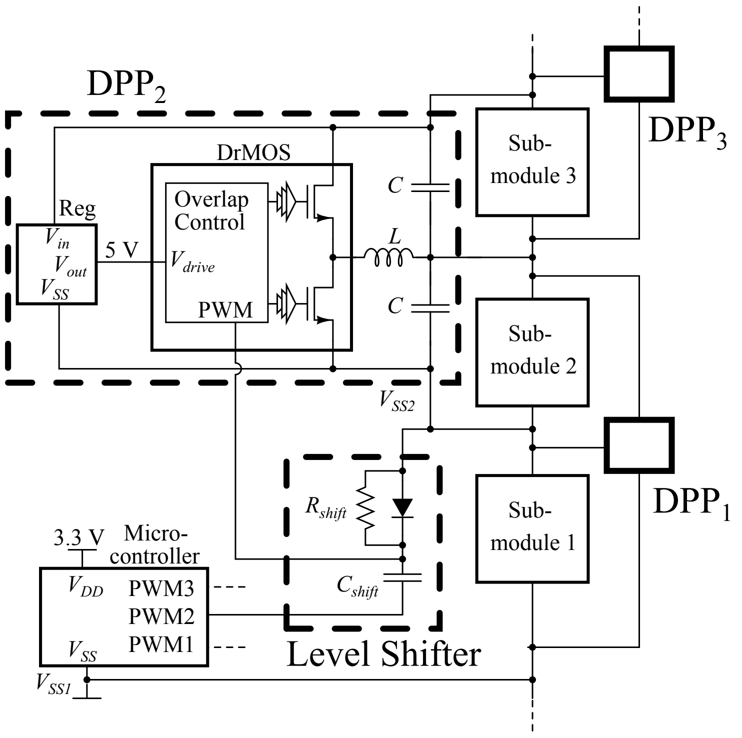


Figure 4.2: Schematic of hardware implementation with DrMOS integrated power stage (details shown for DPP$_2$ only).

4.2 Implementation with DrMOS Integrated Power Stage

4.2.1 Advantage of DrMOS Integrated Power Stage

Since the maximum voltage across each DPP converter is only twice the sub-module voltage, low-voltage, fast-switching transistors can be used. In particular, the voltage rating involved in this application enables us to adopt an integrated driver MOSFET module (DrMOS) initially designed for CPU power delivery applications. The DrMOS combines a PWM controller and two MOSFETs in a single QFN 6×6 package to achieve higher efficiency and smaller footprint than its discrete counterpart. DrMOS can take one PWM signal and add optimal deadtime automatically to generate gate driving signals for the synchronous operation of high-side and low-side switches. DrMOS was originally designed to operate as a synchronous buck converter, but in this particular application it is configured as a buck-boost converter, as shown in Fig. 4.2.

4.2.2 Integrated Power Stage

In theory, the synchronous buck topology and the synchronous buck-boost topology are interchangeable depending on which part to consider as input and output. Moreover, in theory, all synchronous topologies are inherently bidirectional (e.g., a synchronous buck converter can operate as a synchronous boost converter without any change to the circuit, if the input and output are interchanged; a synchronous buck-boost converter can have either side as input). However, in practice, depending on different manufactures, a DrMOS that is intended as synchronous buck converter may or may not be able to operate correctly as a synchronous buck-boost converter. This is because DrMOS from different manufactures have different deadtime generation mechanism.

For example, consider the deadtime generation mechanism of DrMOS from Fairchild, as illustrated by the timing diagram in Fig. 4.3. When the DrMOS is in buck configuration or buck-boost configuration when the current is flowing as shown in Fig. 4.4a, the deadtime generation mechanism works well as follows: When PWM rises from low to high, GL starts to go down. The control circuit will monitor GL, and when GL reaches 1.0V, a delay of t_{D_DEADON} (about 5ns) is inserted and then GH will start to go up. Conversely, when PWM goes from high to low, GH starts to go down. This time the control circuit will monitor VSWH instead of GH (so this deadtime generation mechanism is not symmetrical). When VSWH drops below 2.2V, it inserts a delay of $t_{D\_DEADOFF}$ (about 5ns) and turns on GL. The reason that VSWH can drop below 2.2V is that once GH is open, the inductor current will force the low side diode to conduct and pull PSWH low. Therefore, this mechanism works fine when DrMOS is used as buck, or buck-boost when energy is transferring in the same direction as buck converter. However, since DPP converters need to process power in both directions, a problem occurs when transferring energy backwards, as shown in Fig. 4.4b. The deadtime generation mechanism will work as follows: When PWM makes a high to low transition, the inductor current will force the high side diode to conduct (instead of the low side diode), and thus VSWH cannot be pulled down to below 2.2V before the inductor current reduces to 0A, and thus GL cannot be triggered on as intended. To prevent this problem, Fairchild DrMOS adds an additional mechanism. If GH goes below 1.2V, but VSWH is still not

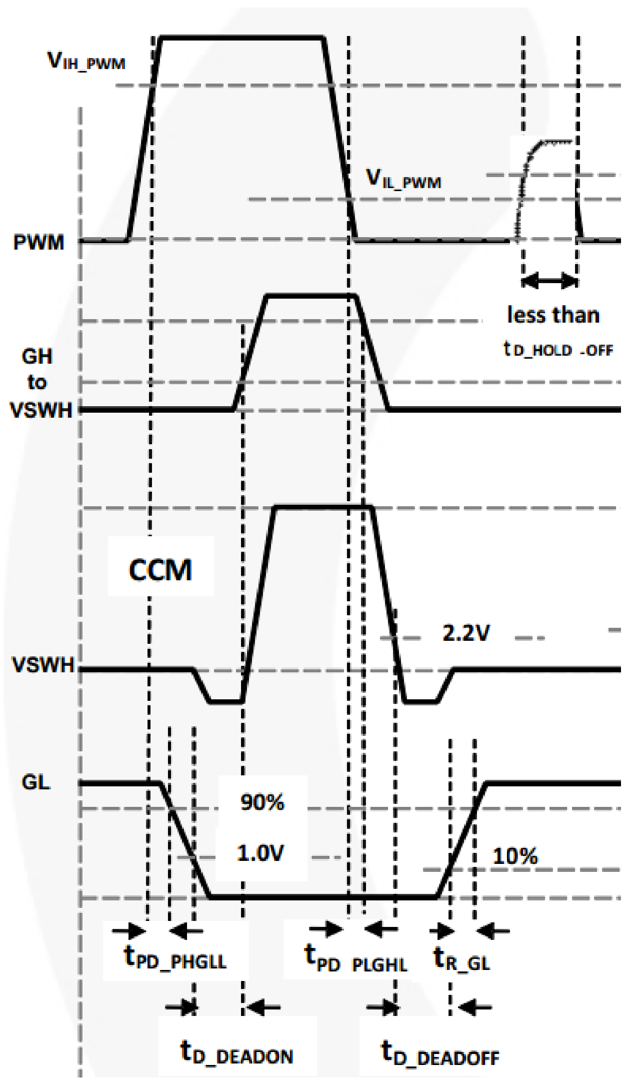
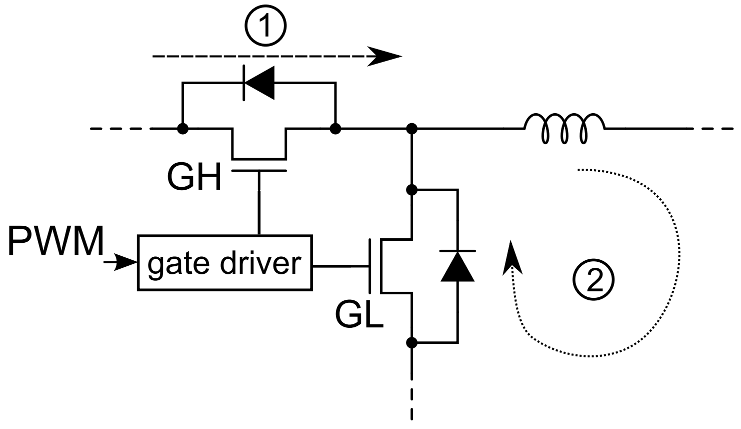
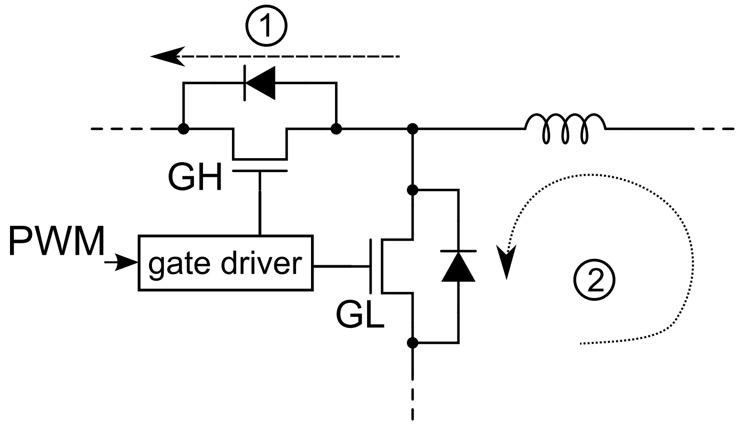


Figure 4.3: Timing diagram of Fairchild FDMF6707V DrMOS module (credit: [1])

below 2.2V, it will insert a delay $t_{D\_TIMEOUT}$ (about 250ns), and then turn on GL. Due to this mechanism, a Fairchild DrMOS can still be used as a bidirectional buck-boost converter, but this $t_{D\_TIMEOUT}$ delay is too long, as it is comparable to the switching cycle. When this DrMOS operates in the reverse direction, it will render a very different voltage ratio inconsistent with the duty ratio, and the conversion efficiency in the reverse direction is very low to the loss in the high side diode (inductor current flows through high side diode for at least 250ns each switching cycle). To summarize, the cause of this problem in Fairchild DrMOS is that when PWM makes a high to low transition, the deadtime generation mechanism monitors VSWH instead of



(a) Forward direction.



(b) Reverse direction.

Figure 4.4: The current flow when DrMOS is operating in different directions. When PWM is high, current is flowing as indicated by line 1. When PWM is low, current is flow as indicated by line 2.

GH. At the same time, DrMOS from Vishay uses a symmetrical deadtime generation mechanism [42]. It monitors GL when the PWM signal makes a low to high transition and monitors GH when the PWM signal makes a high to low transition. Therefore, the deadtime generation is independent of the converter topology. For this reason, Vishay SiC780 is selected to ensure the correct switching action in bi-directional operation. This DrMOS takes 3.3V PWM signal from a micro-controller to control the switching action.

4.2.3 Control Circuit

To implement all the control algorithms (as will be discussed in Chapter 5), a low-cost 32-bit ARM Cortex-M3 micro-controller (STM32F105) from STMicroelectronics is selected [43]. This micro-controller has 3.3V IOs and up to 36MHz clock frequency. A 64-pin package was selected to ensure enough ADC to measure circuit variables (mostly sub-module voltages) and enough IOs to control all the switches of three DPP converters integrated on the same PCB.

A potential limitation of the digital PWM signal generated by the micro-controller is that with limited clock frequency, a trade-off has to be made between PWM frequency and PWM resolution. The PWM frequency is determined by the requirement of high efficiency and small converter size, so it has to be maintained at a relatively high value (100 kHz in this work), rendering a low PWM resolution given the limited micro-controller frequency. In each iteration of the MPPT algorithm proposed in Chapter 5, the duty ratio updates that are calculated with high precision must be quantized to within the PWM resolution, impairing the effectiveness of the algorithm. To overcome this limitation, a PWM dithering technique was adapted from [44, 45]. By dithering periodically between two adjacent quantized duty ratio values available from the micro-controller, with proper filtering, the effective PWM resolution in the hardware prototype is increased by 10 times.

One unique challenge of controlling 3 DPPs with one micro-controller is sending the control signal across different voltage levels, as shown in Fig. 4.2. This figure illustrates the circuit details for DPP$_2$ only, while those for DPP$_1$ and DPP$_3$, which are similar, are omitted. In Fig. 4.2, while DDP$_1$ can be controlled directly by the micro-controller, the PWM control signals sent to DPP$_2$ and DPP$_3$ must be level shifted because they do not have the same voltage reference as the micro-controller. The level shifting circuitry employed in this work is implemented using low-cost passive devices (R_{shift}, C_{shift} and D_{shift} in Fig. 4.2). When the micro-controller PWM$_2$ signal is low, the level shifting capacitor gets charged to the voltage difference between the ground potential of DDP$_2$ and the ground potential of the micro-controller through the diode. The PWM input pin of DPP$_2$'s DRMOS is thus pulled to DPP$_2$'s ground potential. When the micro-controller PWM$_2$ signal reaches the high-voltage limit (3.3 V), the voltage of the DRMOS's PWM input pin is pushed to its high-voltage limit (3.3 V) with respect to DPP_2's ground reference, and the diode turns off. Since the PWM input pin of the DRMOS has a high impedance, any current that flows through the level shifting capacitor is very small, so the input voltage to the DRMOS's PWM input pin can remain high for a long enough time before PWM_2 goes low again. A resistor is placed in parallel with the diode to prevent over-voltage across the PWM input pin due to a sudden decrease of sub-module 1 voltage, i.e., the voltage difference between DPP_2's ground reference and the micro-controller's ground reference. The designer should be aware of certain considerations when selecting the value of level-shifting capacitors and resistors; factors to consider include the time constant of the level shifter compared to the PWM frequency, and the driving capability of micro-controller PWM pins. For protection of the micro-controller PWM pin, a small value resistor can be placed in series with the level-shifting capacitor to reduce current spike during start-up.

4.2.4 Communication

To implement certain control algorithms and to collect data during experiments, it is necessary for the micro-controllers to communicate with a central controller (typically a bench computer) or with each other (neighbor-to-neighbor communication, as will be discussed in Chapter 5). This communication is done through I^2C interface. Note that different micro-controllers operate in different voltage domains, so isolation is required in the communication link. For this purpose, an ADuM1250 I^2C isolator is used.

4.2.5 Complete Converter Circuit

In this prototype design, each DPP converter is constructed with a shielded power inductor and an integrated DRMOS power stage to achieve high efficiency and a very small footprint on the PCB. Each DrMOS is powered by a linear regulator, whose input is connected across the two PV sub-modules that the DrMOS is balancing. This connection slightly sacrifices the converter efficiency (a little bit more gating loss) to ensure that the DPP converter can turn even if the sub-modules are severely shaded. The micro-controller measures all sub-module voltages through a resistor voltage divider

Table 4.1: Main Component List for DPP Converter with Integrate Power Stage

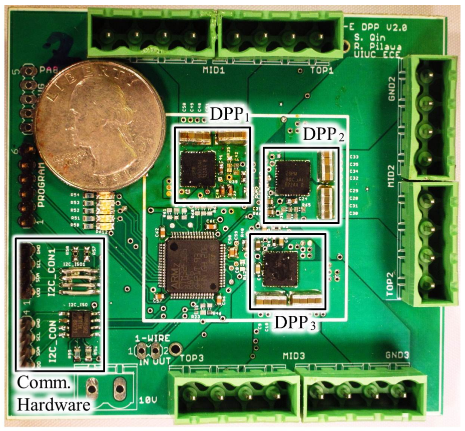
Device	Model	Value	Manufacturer
Microcontroller	STM32F105		STMicroelectronics
DRMOS	SiC780		Vishay
Linear Regulator	UA78L05CPK		Texas Instrument
L	SER1360-103KL	10 μH	Coilcraft
C	TMK212BBJ106KG-T	10 μF$\times$4	Taiyo Yuden
D_{shift}	1SS416CT(TPL3)		Toshiba
C_{shift}	0402	4700 pF	
R_{shift}	0402	100 kΩ	
I^2C Isolator	ADuM1250		Analog Devices

and controls all DPP converters through the capacitor-diode level shifter (except the DPP$_1$, which can be directly control by the micro-controller). While Fig. 4.2 shows a simplified schematic of the hardware prototype described above, a more detailed schematic can be found in the appendix. Table 4.1 contains a list of the main components used.

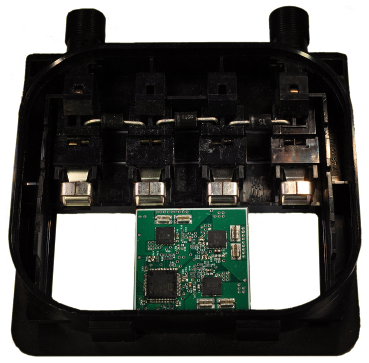
4.2.6 Hardware Prototype

According to the design introduced above, a hardware prototype was built and tested. Figure 4.5a shows an annotated photograph of the front side of the hardware prototype. Magnetic inductors are on the back side of the PCB. Note that a large portion of the board area is consumed by large connectors and ancillary circuitry to facilitate development and diagnosis, which can be eliminated in a final product. As shown in Fig. 4.5a, all essential components of the hardware, including the inductors on the back side of the PCB, only take up a 3.75 cm$\times$3.75 cm area encompassed by a white rectangle, and, as shown in Fig. 4.5b, can easily fit in a junction box.

Some important specifications of the hardware prototype are listed in Table 4.2. An efficiency characterization of the DPP converter across the entire load range is shown in Fig. 4.6. Details about the automatic efficiency measurement setup can be found in the appendix.



(a) The front side of the test prototype.



(b) The prototype fitting in a junction box.

Figure 4.5: Annotated photograph of the hardware prototype with integrated power stage.

Table 4.2: Hardware Prototype Specifications for DPP Converter with Integrate Power Stage

Sub-module Voltage Range	3 – 134 V
Converter Power Rating	60 W
Switching Frequency	100 kHz
Duty Ratio Resolution (with PWM dithering)	1/3600
Converter Peak Efficiency	95%
Weight	28 g
Volume	8.575 cm^3

4.2.7 Efficiency Considerations

Note the relatively low efficiency in light load condition (output current below 0.5A) in Fig. 4.6. This is partially because the converters are implemented as synchronous buck-boost circuits operating in fixed frequency pulse width modulation (PWM) mode. Synchronous buck-boost converters typically operate in continuous conduction mode (CCM). In light load condition, when the inductor current drops to zero and starts to flow reversely, the low side MOSFET will remain on and allow current to flow back to the input. As a result, energy will be transferred back and forth between converter input and output, while only a small amount of net energy is delivered to the output. Large loss will be incurred during the unnecessary back-and-forth transfer of energy. At the same time, as the converter operates at a fixed switching

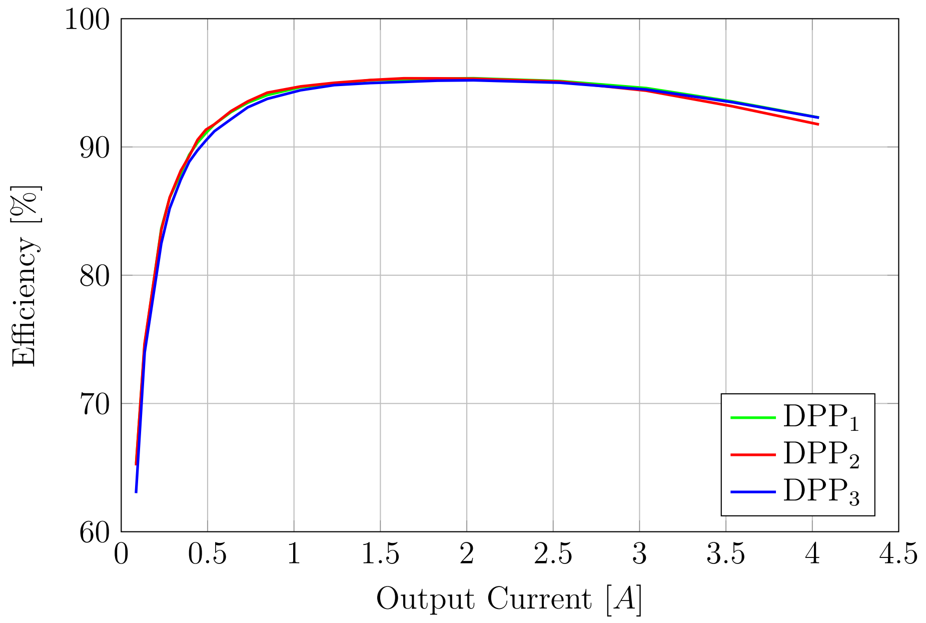


Figure 4.6: Measured efficiency of the 3 DPPs on one hardware prototype at 10 V input voltage, 50% duty ratio.

frequency, the switching loss of the converter is nearly constant over the entire output current range and does not scale down with decreasing load [46]. All these factors lead to an efficiency penalty of the CCM synchronous buck-boost circuit in light load operation. This is especially a problem in DPP structure: the converters in DPP structure only need to process the mismatch power, so they often operate in light load condition.

Many solutions for light load efficiency optimization have been proposed, including pulse frequency modulation (PFM) [47], burst mode operation [48, 49], etc. However, the application of existing light load operation techniques to DPP faces some unique challenges. Most of the existing light load solutions aim at regulating the output voltage to a fixed value, while in our proposed DPP structure, the converters should instead enforce a variable ratio between input and output voltage. Another challenge is that since the distribution of mismatch on a sub-module string is random, the direction in which each DPP converter needs to transfer energy cannot be decided beforehand, and may alternate. Thus, the DPP converter needs to be able to operate bidirectionally. Although a synchronous buck-boost converter running in PWM CCM readily lends itself to bidirectional operation, most of the light load operation techniques require a clearly defined operation direction to regulate the output voltage. Note that here it is assumed that inductor current measurement (which introduces extra power loss and hardware cost) is not performed; otherwise, the technique in [47] is still applicable. Nevertheless, it is desirable to develop a bidirectional light load control scheme that does not rely on any current sensing, which will be part of the future work.

4.3 Implementation with Discrete Power Stage

4.3.1 Advantage of Discrete Power Stage

While the DrMOS integrated power stage offers certain benefits such as high efficiency and small hardware footprint, it has certain limitations. First, at the time of this writing, the Vishay DrMOS chips suitable for bidirectional operation are rated for up to 27V input voltage, which means the open circuit voltage of each submodule should be less than 13.5V. This voltage range works for many types of commercial PV modules but could potentially limit the application to all types of PV modules. Second, the high side and low side MOSFET in DrMOS chips are typically optimized for low duty ratio (high side MOSFET has larger on resistance than low side MOSFET), while in this DPP application the converter duty ratio is always around 0.5. Last, compared to discrete gate drivers and MOSFETs, the choice for DrMOS is relatively limited, which leaves little flexibility in further design if we want to tradeoff performance for cost. All these reasons motivate the development of a DPP converter with discrete power stage components.

4.3.2 Discrete Power Stage

The schematic of the DPP converter with discrete power stage, as shown in Fig. 4.7 is very similar to the one with integrated power stage. The difference is that the DrMOS is replaced with a gate driver chip and two discrete MOSFETs. The CSD18504Q5A N-Channel Power MOSFET is selected to implement both switches as it offers a reasonable tradeoff between gate capacitance and on resistance. A FAN7390MX halfbridge gate driver from Fairchild is selected to drive both MOSFETs. It takes two separate

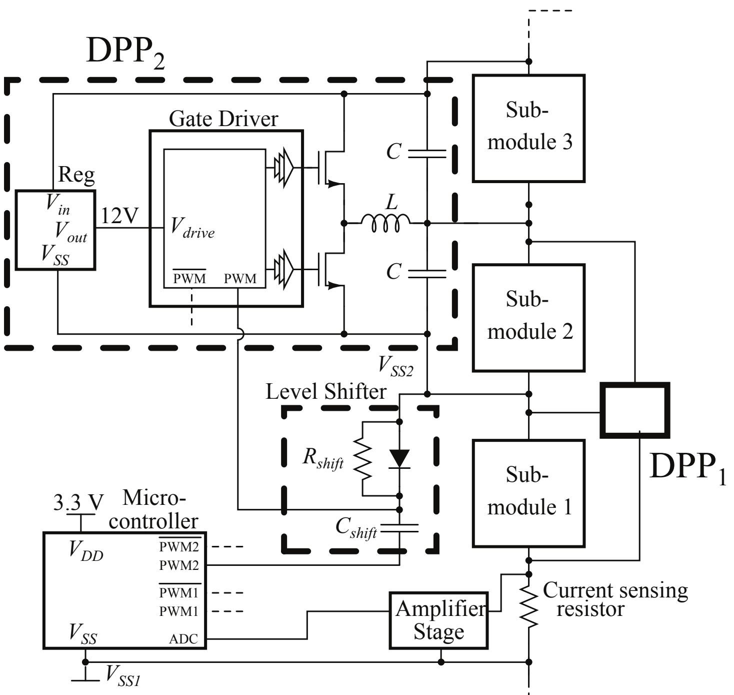


Figure 4.7: Schematic of hardware implementation with discrete power stage (details shown for DPP$_2$ only).

PWM control signals from the micro-controller for high side and low side switches; therefore, the deadtime generation is now part of the task of the micro-controller (please see the micro-controller code in the appendix for details). Compared to DrMOS design, the PCB layout is more critical in the discrete design as stray inductance may compromise the converter efficiency and performance. The gate driver is placed very close to the MOSFET to minimize the loop between gate drive and MOSFET gate. The layout of this design can be found in the appendix. Interested readers can also refer to Section 4.1 in [50] for a more detailed description on this matter.

4.3.3 Moving Window Current sensing

A potential challenge of implementing the control algorithm that will be introduced in Chapter 7 is precisely sensing the small current change resulting from the converter duty ratio perturbations. The module current typically ranges from 1A to 8A, while the small current change can be on the order of a few mA. In the presence of such a large average current, sensing a small change is difficult. A circuit consisting of a current sensing resistor and an amplifier that converts the module current linearly to a voltage signal for the micro-controller ADC cannot guarantee the effective operation of the algorithm: the small current change, after converted to a voltage signal, is typically too small for the ADC to distinguish from the noise in the circuit. While a stand-alone, high resolution ADC can alleviate this concern, it would add significant cost. To solve this problem, a digitally-assisted windowed sensing technique has been adopted from [51]. The basic concept of this technique is illustrated in Fig. 4.8. The current circuit implementation of this sensing technique is shown in Fig. 4.9. The first stage uses a conventional current sensing method: the module current signal is converted into a voltage signal by a 5 mΩ current sensing resistor and scaled up linearly by an amplifier with a fixed gain of 50. For a module current change of 10 mA, the output voltage signal, v_{sense}, only changes 5 mV, which is too small to be effectively captured by the micro-controller's ADC for any appreciable measurement noise in the system. However, any attempt to further increase the gain of this circuit will saturate the amplifier output since a large average value is also present in the module current. To decouple the relative change and absolute value of the signal and further amplify the relative change, the voltage signal v_{sense} is passed to a voltage subtractor (differential amplifier) stage. A voltage reference v_{bias} that is slightly smaller than v_{sense} is subtracted from v_{sense}, and the resulting small difference can then be further amplified by the differential amplifier to get $v_{windowed}$, i.e.,

$$v_{windowed} = \frac{R_2}{R_1}(v_{sense} - v_{bias}), \tag{4.1}$$

$$\text{given that} \quad R_1 = R_3,\ R_2 = R_4. \tag{4.2}$$

In this design, we select $R_1 = R_3 = 1\ k\Omega$, $R_2 = R_4 = 50\ k\Omega$, so a 10 mA change in module current results in a 5 mV change in v_{sense} and eventually a 250 mV change in $v_{windowed}$, which is large enough to be sensed by the micro-controller's ADC. The relative change of $v_{windowed}$ reflects the corresponding change of module current. Since the control algorithm is only concerned with the relative current change and not the absolute value, $v_{windowed}$ can provide enough information for the algorithm to work. v_{bias} is provided by the micro-controller's PWM output and can be adjusted by the micro-controller to make sure it is always slightly smaller than v_{sense} to avoid saturating the differen-

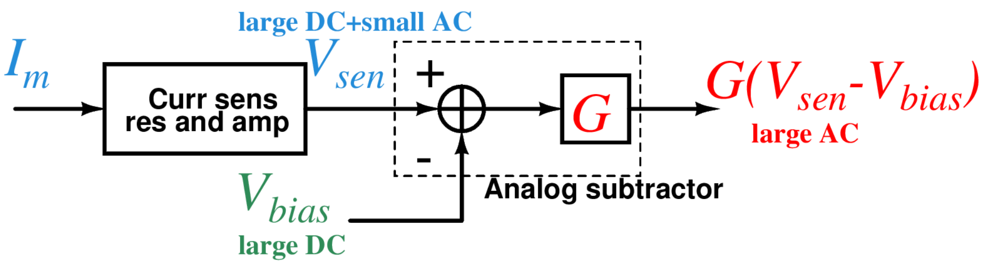


Figure 4.8: Concept of the digitally-assisted windowed sensing technique.

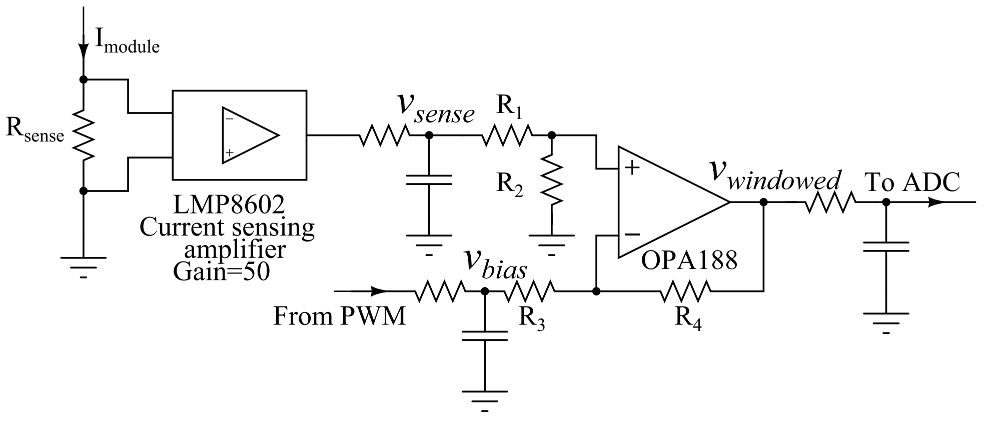


Figure 4.9: Schematic of the current sensing circuit using a measurement windowing technique.

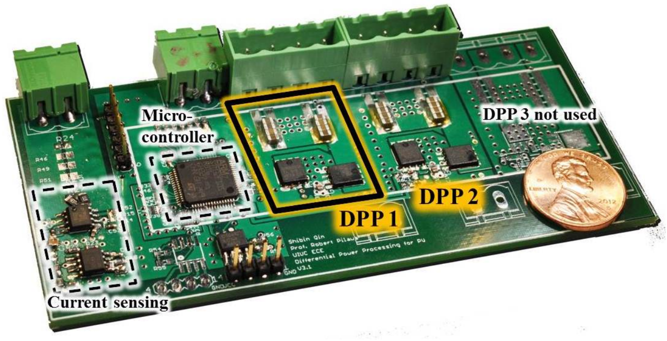
tial amplifier output. More details on the implementation of this windowed sensing technique can be found in [51].

4.3.4 Hardware Prototype

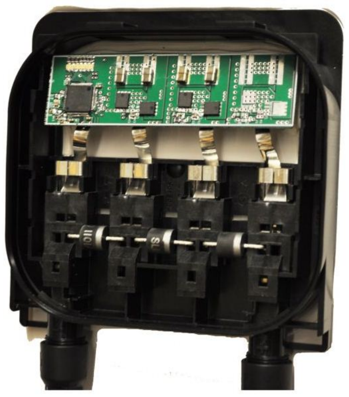
For this discrete design, most parts of the circuit other than the power stage remain the same as the design with integrated power stage. The component selection is summarized in Table 4.3. This prototype is built as shown in Fig. 4.10a. The size of the hardware prototype increases by more than 2 times due to the use of discrete power stage components, but all the essential components can still fit into the PV module junction box component, as shown in Fig. 4.10b.

Table 4.3: Main Component List for DPP Converter with Discrete Power Stage

Device	Model	Value	Manufacturer
Microcontroller	STM32F105		STMicroelectronics
MOSFET	CSD18504Q5A		Texas Instrument
Linear Regulator	KA78L12AMTF		Fairchild
Gate Driver	FAN7390MX		Fairchild
L	SER1360-103KL	10 μH	Coilcraft
C	TMK212BBJ106KG-T	10 μF$\times$4	Taiyo Yuden
D_{shift}	1SS416CT(TPL3)		Toshiba
C_{shift}	0402	1 nF	
R_{shift}	0402	40 kΩ	
Amplifier	LMP8602		Texas Instrument
Amplifier	OPA188		Texas Instrument



(a) The front side of the test prototype.



(b) The prototype fitting in a junction box.

Figure 4.10: Annotated photograph of the hardware prototype with discrete power stage.

Table 4.4: Hardware Prototype Specifications for DPP Converter with Discrete Power Stage

Sub-module Voltage Range	5-20 V
Power Rating (Each converter)	60 W
Switching Frequency	100 kHz
Converter Peak Efficiency	95%
Weight	25 g
Volume	17.29 cm^3

4.4 Future Work on Hardware Design

The purpose of hardware design in this research is to build a platform to prove the concept of differential power processing and the associated MPPT control algorithms. Although a very carefully optimized DPP converter is an essential part of DPP system, it is not the focus of this work at the current stage. Therefore, there is much space for improvement on the hardware design.

4.4.1 Power Stage Optimization

The integrated power stage is already optimized by the design of manufacturers, but is intended to operate with low duty ratio. An integrated power stage optimized for medium duty ratio will offer better performance if it becomes available from a manufacturer. For the discrete power stage, components selections are selected based on ease of use, and do not go through a careful optimization for efficiency. Moreover, it is difficult to precisely predict the converter loss based on calculation. In practice, several components should be tested to decide which gives the best performance. A careful redesign should include these steps and an increase in the converter efficiency is expected.

4.4.2 Power Line Communication

In current design, the communication between micro-controllers of different DPP converters is implemented with I^{2}C protocol through an ADuM 1250 isolators. A cost breakdown (please see the appendix) reveals that the isolators take up a very significant portion (about 25%) of the total cost. It also adds more wires between different PV modules and complicates the installation. Power line communication techniques can be explored to solve this problem.

4.4.3 Light Load Operation and Current Limiting

As mentioned above, one advantage of DPP arthitecture is that each DPP converter only needs to process differential power, which is typically very small. Therefore, the converters often operate in light load condition and suffer from low light load efficiency. A good bidirectional light load operation scheme (preferably without any need for current measurement) is yet to be developed. On the other hand, in some extreme case where severe mismatches exist between PV submodules, DPP converters may process more power than an entire submodule. These extreme cases are rare, but should be considered in the design for the reliable operation of the DPP system. While the converter can be designed for the worst case power rating, it is preferable to design the converter for lower rating (average case power rating) and add a current limiting functionality to handle the worse than average cases. One possible solution is to measure the inductor current and limit its peak when it is above a certain threshold. None of the control algorithms developed in this work consider the scenario when one or more DPP converters hit the current limit. Therefore, the impact of this potential current limiting scheme should also be analyzed in the future work.

CHAPTER 5

A DISTRIBUTED CONTROL ALGORITHM FOR DPP ARCHITECTURE

In this chapter, we formulate a distributed iterative algorithm which, through the exchange of information among local controllers, and for a given string current, maximizes the power extracted from an array of n series-connected PV sub-modules outfitted with $n-1$ DPP converters. We first formulate the algorithm for a 3-sub-module, 2-DPP system, and then generalize it to a system of any size.

5.1 Algorithm Formulation for a 3-sub-module, 2-DPP System

A system comprising two DPP converters and three sub-modules is illustrated in Fig. 5.1. This system is based on the architecture presented in [2]. Each DPP is implemented as a bidirectional buck-boost converter that enforces a voltage ratio between two adjacent PV sub-modules. The PV string is attached to a central converter (omitted in Fig. 5.1). The control objective of the MPPT algorithm is to maximize the power extracted from the PV string, P_o, i.e.,

$$\underset{D_1,D_2}{\text{maximize}} \quad P_o = V_o \times I_o, \tag{5.1}$$

where I_o is the string current, and $V_o := V_1 + V_2 + V_3$ is the string voltage. Given an irradiance condition, each PV sub-module has a unique maximum power point ($V_{i,max}$, $I_{i,max}$), $i = 1, 2, 3$. As derived in [14], when each PV sub-module is operating at its maximum power point, the entire system is operating at its maximum power point, and the corresponding string current and the string voltage can be uniquely determined through KCL and KVL as $V_{o,max}$ and $I_{o,max}$. Moreover, assume the DPP converters are ideal, then

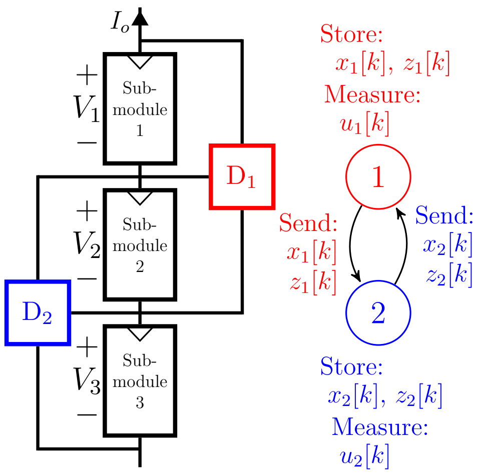


Figure 5.1: Block diagram of a 3-sub-module, 2-DPP system.

the duty ratios of all DPP converters, $D_{1,max}$ and $D_{2,max}$, can be uniquely determined as well. In fact, $I_{o,max}$, $D_{1,max}$ and $D_{2,max}$ form a group of variables that fully determine the entire system. Therefore, the control objective of maximizing P_o leads to the task of dynamically tracking the unique combination of $(I_{o,max}, D_{1,max}, D_{2,max})$ that corresponds to $P_{o,max}$.

To this end, we separate the setting of I_o and V_o into two control loops. In a relatively slow control loop, the central converter is configured as a controllable current sink, and it performs a conventional perturb and observe (P&O) algorithm to seek the string current I_o that gives the largest P_o. In the second, relatively fast control loop, all DPPs adjust their duty ratios in each DPP tracking iteration and it takes many iterations to maximize V_o. Since the perturbation interval of the central converter is much longer than the DPP control loop's time constant (because the DPP converters are switching at a much higher frequency, as discussed in Chapter 4), at any given time, the string current I_o can be considered temporarily fixed for the DPPs. Given a fixed I_o, it is easy to see that, regardless of the irradiance incident on each of the sub-modules, maximizing the string voltage V_o, is equivalent to maximizing P_o. As illustrated by the timeline in Fig. 5.2, in the slow control loop, the central first inverter updates and enforces ("perturb") a certain string current value, $I_{o,1}$. The DPP converters take many iterations in the fast control loop to maximum string voltage V_o and thus the string can provide the maximum power possible for this particular $I_{o,1}$, namely $P_{o,1}$. $P_{o,1}$ is measured ("observe") by the central inverter to use in its P&O

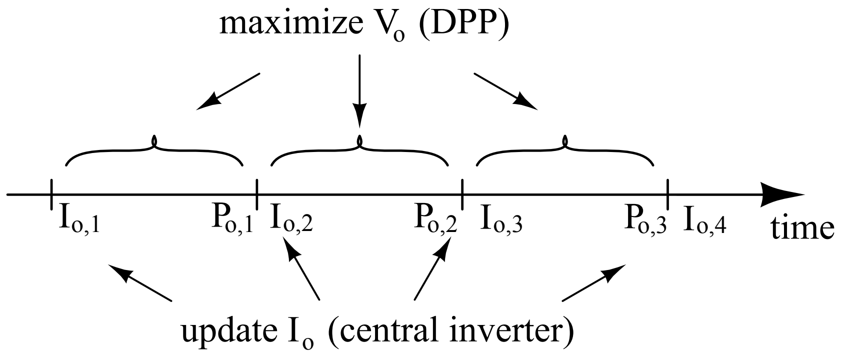


Figure 5.2: Timeline of the "slow" control loop (central inverter) and "fast" control loop (DPP).

algorithm. Then the central inverter updates ("perturb") the current to a to a new value $I_{o,2}$, the DPP converters maximize the string voltage V_o and produce the maximum power possible for $I_{o,2}$, namely $P_{o,2}$ to be measured ("observed") by the central inverter. The difference between $P_{o,1}$ and $P_{o,2}$ is used to determine the direction of next string current perturbation. The slow and fast control loop can repeat this pattern and find the optimal string current $I_{o,max}$. In this thesis, we assume that the slow loop implemented in the central inverter uses a conventional P&O algorithm to seek $I_{o,max}$, and we focus on the control algorithm for DPPs in the fast control loop.

In our system architecture, we observe that each DPP has access to measurements of only two of the three sub-module voltages instead of the entire string voltage. Thus, in order to determine the duty ratios for which V_o is maximized, we assume that the local controllers of DPP$_1$ and DPP$_2$ can share information through neighbor-to-neighbor communication; this can be described by a directed graph, as shown on the right of the block diagram in Fig. 5.1. Then, we tailor the distributed optimization algorithm in [52] to our setting. [The reader is referred to the Appendix for a brief overview on the mathematics of this algorithm.]

The control objective described above can be written as

$$\underset{D_1,D_2}{\text{maximize}} \;\; V_o(D_1,D_2) = V_1(D_1,D_2) + V_2(D_1,D_2) + V_3(D_1,D_2),$$

where D_1 and D_2 are the duty ratios of DPP$_1$ and DPP$_2$, respectively. Then, the maxima of the function $V_o(D_1,D_2)$ can be found by setting its gradient to zero, i.e.,

$$\nabla V(D_1^*, D_2^*) = \begin{bmatrix} \left.\frac{\partial V_1(D_1,D_2)}{\partial D_1}\right|_{D_1^*,D_2^*} + \left.\frac{\partial V_2(D_1,D_2)}{\partial D_1}\right|_{D_1^*,D_2^*} + \left.\frac{\partial V_3(D_1,D_2)}{\partial D_1}\right|_{D_1^*,D_2^*} \\ \left.\frac{\partial V_1(D_1,D_2)}{\partial D_2}\right|_{D_1^*,D_2^*} + \left.\frac{\partial V_2(D_1,D_2)}{\partial D_2}\right|_{D_1^*,D_2^*} + \left.\frac{\partial V_3(D_1,D_2)}{\partial D_2}\right|_{D_1^*,D_2^*} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{5.2}$$

where D_1^* and D_2^* are the respective duty ratios of DPP$_1$ and DPP$_2$ that correspond to the maximum power point. To find D_1^* and D_2^*, the local controller of each DPP iteratively adjusts its duty ratio based on (i) local voltage measurements, (ii) state variables maintained locally, and (iii) variables maintained by neighboring DPPs. Let $k = 1, 2, \dots$ index the iterations performed by every DPP, and let $x_1[k]$, $x_2[k]$ be state vectors maintained by the local controllers of DPP$_1$ and DPP$_2$, respectively. For DPP$_1$, the entries of $x_1[k]$ are $D_1[k]$—the actual duty ratio of DPP$_1$, and $\hat{D}_{1,2}[k]$—DPP$_1$'s estimate of the duty ratio of DPP$_2$. Similarly, for DPP$_2$, the entries of $x_2[k]$ are $\hat{D}_{2,1}[k]$—DPP$_2$'s estimate of the duty ratio of DPP$_1$, and $D_2[k]$—the actual duty ratio of DPP$_2$. Furthermore, let $z_1[k]$ and $z_2[k]$ be ancillary state vectors maintained by the local controllers respectively; unlike $x_1[k]$ and $x_2[k]$, these ancillary states do not correspond to any physical variable. For the entire system, we define

$$x[k] = \Big[\underbrace{D_1[k], \hat{D}_{1,2}[k]}_{x_1[k]}, \underbrace{\hat{D}_{2,1}[k], D_2[k]}_{x_2[k]}\Big]^{\mathrm{T}}, \tag{5.3}$$

$$z[k] = \Big[\underbrace{z_{1,1}[k], z_{1,2}[k]}_{z_1[k]}, \underbrace{z_{2,1}[k], z_{2,2}[k]}_{z_2[k]}\Big]^{\mathrm{T}}. \tag{5.4}$$

Then, at each iteration, the variables are updated as

$$x[k+1] = (\mathbb{I}_4 - \delta\tilde{L})x[k] - \delta\tilde{L}z[k] + \delta\gamma u[k], \tag{5.5}$$

$$z[k+1] = z[k] + \delta\tilde{L}x[k], \tag{5.6}$$

where $\mathbb{I}_4$ is the 4×4 identity matrix; $\tilde{L} = L \otimes \mathbb{I}_2$, where $\mathbb{I}_2$ is the 2×2 identity matrix,

$$L = \begin{bmatrix} 1 & -1 \\ -1 & 0 \end{bmatrix}$$

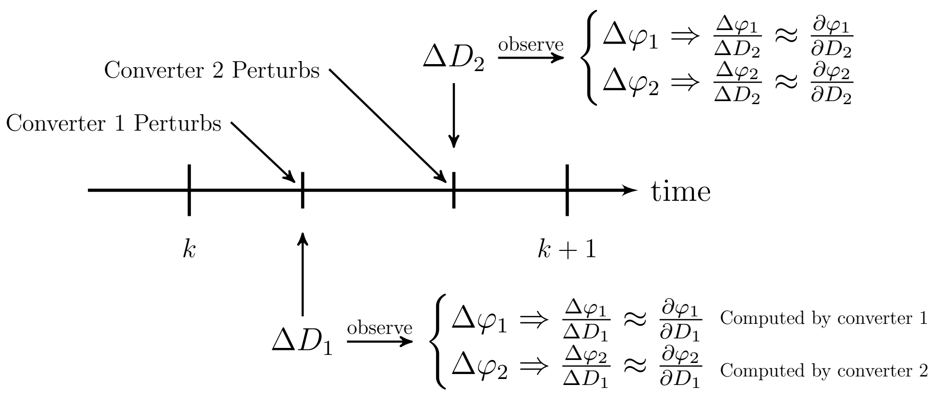
is the Laplacian matrix of the graph representing the exchange of information between local controllers as depicted in Fig. 5.1, and "$\otimes$" denotes the Kronecker product of L and $\mathbb{I}_2$ (see, e.g., [53]); δ and γ are parameters that can be used to tune the algorithm (we will discuss this matter in detail in Section 5.4); and $u[k]$ is defined as

$$u[k] := \Bigg[\underbrace{\frac{\partial\varphi_1}{\partial D_1}\bigg|_{D_1[k],D_2[k]}, \frac{\partial\varphi_1}{\partial D_2}\bigg|_{D_1[k],D_2[k]}}_{u_1[k]=\frac{\partial\varphi_1(D)}{\partial D}}, \underbrace{\frac{\partial\varphi_2}{\partial D_1}\bigg|_{D_1[k],D_2[k]}, \frac{\partial\varphi_2}{\partial D_2}\bigg|_{D_1[k],D_2[k]}}_{u_2[k]=\frac{\partial\varphi_2(D)}{\partial D}}\Bigg]^{\mathrm{T}}, \quad (5.7)$$

with φ_1 and φ_2 defined respectively as

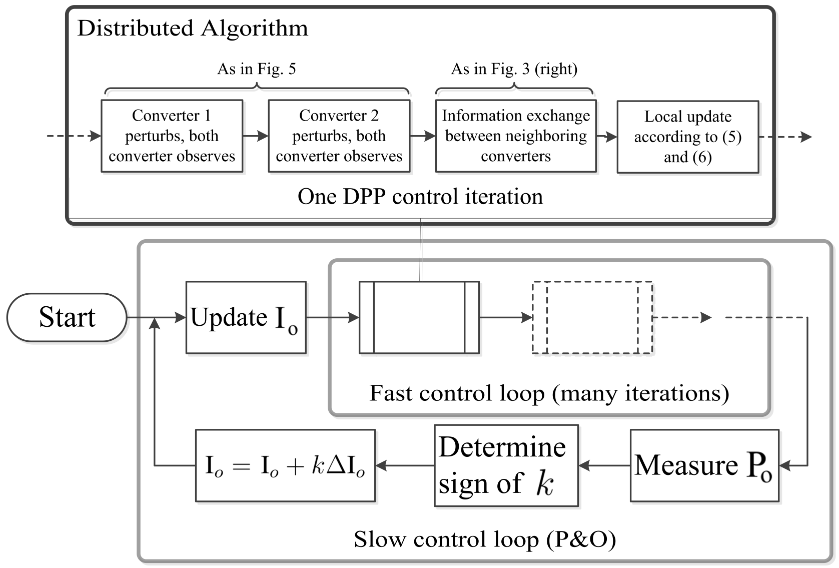
$$\varphi_1(D_1, D_2) := V_1(D_1, D_2) + \frac{1}{2}V_2(D_1, D_2),$$
$$\varphi_2(D_1, D_2) := \frac{1}{2}V_2(D_1, D_2) + V_3(D_1, D_2).$$

From the above discussion, we see that the update of DPP$_1$'s states, $x_1[k+1]$ and $z_1[k+1]$, depends on its own states, $x_1[k]$ and $z_1[k]$, the states of its neighbor, $x_2[k]$ and $z_2[k]$, and the partial derivatives of the sub-module voltages that DPP$_1$ is directly attached to, i.e., $u_1[k]$. In this regard, while the necessary information that each DPP converter needs for updating (5.5) and (5.6) can be acquired through neighbor-to-neighbor communication, in order to obtain $u_i[k]$, each DPP$_i$ must estimate the partial derivatives of the $\varphi_i(\cdot)$ function. To perform this estimation, at every iteration, each DPP$_i$ alternatively perturbs its duty ratio by a fixed small amount, ΔD_i, while both local controllers observe the sub-module voltages; the timeline of this process is depicted in Fig. 5.3. After both local controllers have observed the results of each perturbation, and upon receiving the necessary information from the neighboring DPP, they can approximate the partial derivatives as $\frac{\partial\varphi_i}{\partial D_j} \approx \frac{\Delta\varphi_i}{\Delta D_j}$, $i, j = 1, 2$, and update their respective state variables. After $k = m$ iterations, for m sufficiently large, we have that $D_1[m] \approx D_1^*$ and $D_2[m] \approx D_2^*$; thus the string voltage is maximized and the maximum power



Converter 2 Perturbs
Converter 1 Perturbs
$\Delta D_2 \xrightarrow{\text{observe}} \begin{cases} \Delta\varphi_1 \Rightarrow \frac{\Delta\varphi_1}{\Delta D_2} \approx \frac{\partial\varphi_1}{\partial D_2} \\ \Delta\varphi_2 \Rightarrow \frac{\Delta\varphi_2}{\Delta D_2} \approx \frac{\partial\varphi_2}{\partial D_2} \end{cases}$
time
k
$k+1$
$\Delta D_1 \xrightarrow{\text{observe}} \begin{cases} \Delta\varphi_1 \Rightarrow \frac{\Delta\varphi_1}{\Delta D_1} \approx \frac{\partial\varphi_1}{\partial D_1} \\ \Delta\varphi_2 \Rightarrow \frac{\Delta\varphi_2}{\Delta D_1} \approx \frac{\partial\varphi_2}{\partial D_1} \end{cases}$
Computed by converter 1
Computed by converter 2

Figure 5.3: Timeline of distributed perturb and observe for the 3-sub-module, 2-DPP system.



Distributed Algorithm
As in Fig. 5
As in Fig. 3 (right)
Converter 1 perturbs, both converter observes
Converter 2 perturbs, both converter observes
Information exchange between neighboring converters
Local update according to (5) and (6)
One DPP control iteration
Start
Update I_o
Fast control loop (many iterations)
$I_o = I_o + k\Delta I_o$
Determine sign of k
Measure P_o
Slow control loop (P&O)

Figure 5.4: Simplified flowchart of the proposed algorithm for the 3-sub-module, 2-DPP system.

is extracted from the system at a given string current I_o. The fast control loops continue tracking the maximum power and once the slow control loop update I_o to a new value, the fast control loop will track the new condition. Note that although it seems from Fig. 5.4 that the two control loops are synchronized, they are actually not. As long as the time interval of the slow control loop is much larger than that of the fast control loop such that the fast control loop can take a large number of iterations (more than 100) during each slow control loop P&O interval, the two control loops can operate independently with respect to each other.

5.2 Algorithm Formulation for an n-sub-module, (n-1)-DPP System

For a system of n sub-modules and $n-1$ DPP converters, the communication graph of which is shown in Fig. 5.5, we define the state vector, and the ancillary state vector of the entire system as

$$x[k] = \begin{bmatrix} x_1[k] & x_2[k] & \dots & x_i[k] & \dots & x_{n-1}[k] \end{bmatrix}^{\mathrm{T}}, \tag{5.8}$$

$$z[k] = \begin{bmatrix} z_1[k] & z_2[k] & \dots & z_i[k] & \dots & z_{n-1}[k] \end{bmatrix}^{\mathrm{T}}, \tag{5.9}$$

where $x_i[k]$ and $z_i[k]$, $i = 1, 2, \dots, n-1$, are state vectors maintained by the local controller of DPP$_i$. Each local controller maintains an estimate of the duty ratios of all other DPPs in addition to its own; thus, $x_i[k]$ consists of $\hat{D}_{i,j}[k]$, $i, j = 1, \dots, n-1$, $i \neq j$ and $D_i[k]$, i.e.,

$$x_i[k] = \begin{bmatrix} \hat{D}_{i,1}[k], \dots, \hat{D}_{i,i-1}[k], D_i[k], \hat{D}_{i,i+1}[k], \dots, \hat{D}_{i,n-1}[k] \end{bmatrix}^{\mathrm{T}},$$
$$z_i[k] = \begin{bmatrix} z_{i,1}[k], z_{i,2}[k], \dots, z_{i,n-1}[k] \end{bmatrix}^{\mathrm{T}}.$$

We also define the input vector as

$$u[k] = \begin{bmatrix} \frac{\partial \varphi_1(D)}{\partial D} & \frac{\partial \varphi_2(D)}{\partial D} & \dots & \frac{\partial \varphi_{n-1}(D)}{\partial D} \end{bmatrix}^{\mathrm{T}}, \tag{5.10}$$

where

$$\frac{\partial \varphi_i(D)}{\partial D} = \begin{bmatrix} \frac{\partial \varphi_i(D)}{\partial D_1}, \frac{\partial \varphi_i(D)}{\partial D_2}, \dots, \frac{\partial \varphi_i(D)}{\partial D_{n-1}} \end{bmatrix}^{\mathrm{T}}, \tag{5.11}$$

with $\varphi_i(D)$ given by

$$\varphi_i(D) = \begin{cases} V_i(D) + \frac{1}{2}V_{i+1}(D), & i = 1, \\ \frac{1}{2}V_i(D) + \frac{1}{2}V_{i+1}(D), & 1 < i < n-1, \\ \frac{1}{2}V_i(D) + V_{i+1}(D), & i = n-1. \end{cases} \tag{5.12}$$

Then, at each iteration, similar to (5.5) and (5.6), the variables are updated

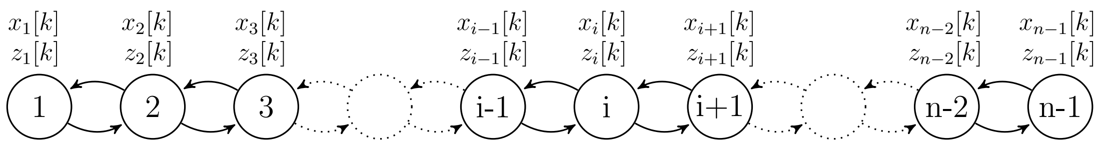


Figure 5.5: Graph of an n-sub-module, (n-1)-DPP system.

as

$$x[k+1] = (\mathbb{I}_{(n-1)^2} - \delta\tilde{L})x[k] - \delta\tilde{L}z[k] + \delta\gamma u[k], \tag{5.13}$$

$$z[k+1] = z[k] + \delta\tilde{L}x[k], \tag{5.14}$$

where $\mathbb{I}_{(n-1)^2}$ is the $(n-1)\times(n-1)$ identity matrix; $\tilde{L} = L \otimes \mathbb{I}_{n-1}$, where L is now the Laplacian matrix of the graph in Fig. 5.5:

$$L = \begin{bmatrix} 1 & -1 & 0 & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & 0 \\ 0 & -1 & 2 & -1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & \cdots & \cdots & -1 & 1 \end{bmatrix}.$$

By inspecting (5.13) and (5.14) for the n-sub-module case, it is easy to see that the state of DPP$_i$ at instant $k+1$, $x_i[k+1]$ and $z_i[k+1]$, depends on its own state at instant k, $x_i[k]$ and $z_i[k]$; the states of its neighboring DPPs at instant k, i.e., $x_{i-1}[k]$, $z_{i-1}[k]$, $x_{i+1}[k]$, $z_{i+1}[k]$, all of which can be acquired through the neighbor-to-neighbor communication; and $u_i[k]$, which is approximated by DPP$_i$ as $\frac{\Delta\varphi_i(D)}{\Delta D}$. Similar to the 2-DPP case, after $k = m$ iterations, for m sufficiently large, we have $D_i[m] \approx D_i^*, i = 1, 2, \dots, n-1$.

Note that although the algorithm update functions are compactly written in matrix form in (5.13) and (5.14), the actual data storage and computation is distributed among the DPPs. To understand how this algorithm is distributed, consider DPP$_i$ (node i) in Fig. 5.5 as an example. DPP$_i$ locally stores and updates vector $x_i[k]$ and $z_i[k]$ that are part of (5.8) and (5.9), respectively. For the computation performed by DPP$_i$, only the i^{th} row of Laplacian matrix L (representing the information exchange of DPP$_i$ with its closest neighbors) is relevant. If we expand (5.13) and (5.14), and write down only the part needed to compute $x_i[k+1]$, we obtain that

$$\begin{aligned} x_i[k+1] &= \delta x_{i-1}[k] + (1-2\delta)x_i[k] + \delta x_{i+1}[k] + \delta z_{i-1}[k] \\ &\quad - 2\delta z_i[k] + \delta z_{i+1}[k] + \delta\gamma u_i[k], \\ z_i[k+1] &= z_i[k] - \delta x_{i-1}[k] + 2\delta x_i[k] - \delta x_{i+1}[k], \end{aligned}$$

which illustrates that the computation of $x_i[k+1]$ can be performed independently by DPP$_i$ as long as this DPP converter has access to the state of its closest neighbors (this is achieved in practice through neighbor-to-neighbor communications). Other states of the system are irrelevant to DPP$_i$'s computation. A similar conclusion applies to other DPP converters; therefore, each DPP converter performs only part of the update computation relevant to itself, and thus the proposed algorithm is distributed.

5.3 Mathematical Background

In this section, a brief introduction to the mathematical background of our proposed algorithm is presented; the reader is referred to [52] for a more detailed account. While [52] derives the algorithm in the continuous-time framework, the algorithm proposed in this work presents the discrete-time counterpart.

To derive the algorithm, we will start in the continuous-time framework and discretize the resulting dynamics afterwards. Consider a network with n nodes, i.e., $v_1, v_2, \ldots, v_n$, whose communication topology can be described by a strongly connected digraph $\mathcal{G}$, and each node v_i, $i \in \{1, \ldots, n\}$ has a continuously differentiable and convex function f^i that is only available to this node; then as illustrated in [52], the unconstrained optimization problem

$$\min f(x^o) = \sum_{i=1}^{n} f^i\left(x^o\right) \tag{5.15}$$

can be equivalently formulated as a constrained optimization problem

$$\min \tilde{f}(x) = \sum_{i=1}^{n} f^i(x_i), \tag{5.16}$$

$$\text{subject to } \tilde{L}x = 0_n, \tag{5.17}$$

where $\tilde{L} = L \otimes \mathbb{I}_n$; L is the Laplacian Matrix representing $\mathcal{G}$, $\mathbb{I}_n$ is the $n \times n$ identity matrix, and "$\otimes$" represents Kronecker product of L and $\mathbb{I}_n$; 0_n is the n-dimensional all-zero vector; x^i is node v_i's estimate of the solution to (5.15), and x is a vector containing estimates from all nodes, i.e.,

$$x = [x_1^T, x_2^T, \ldots, x_n^T]^T. \tag{5.18}$$

Then this constrained optimization problem can be solved using augmented Lagrangian method, i.e.,

$$\min F(x, z) = \tilde{f}(x) + z^T \tilde{L}x + \frac{1}{2} x^T \tilde{L}x, \tag{5.19}$$

where z^T is the estimate of the Lagrange multiplier. To solve (5.19), we need to find the point (x^*,z^*) that sets the gradient of $F(x, z)$ to zero, which we can obtain through the following dynamics:

$$\dot{x} + \tilde{L}x + \tilde{L}z = -\nabla \tilde{f}(x), \tag{5.20}$$

$$\dot{z} = \tilde{L}x. \tag{5.21}$$

As (5.20) and (5.21) converge to the equilibrium point (x^*,z^*), x^* is the solution to (5.16), and $x_1 = x_2 = \cdots = x_n$ in (5.18) is the solution to the original problem in (5.15).

In our specific DPP system setting, the control objective is

$$\underset{D_1,\ldots,D_n}{\text{maximize}} \quad V(D_1, \ldots, D_n) = \sum_{i=1}^{n} V_i(D_i). \tag{5.22}$$

This control objective has the same separable structure as (5.15); thus we can tailor the algorithm in (5.20) and (5.21) to our setting. To this end, we discretize (5.20) and (5.21) using the forward rectangular rule, i.e.,

$$\dot{x} = \frac{x[k+1] - x[k]}{\delta}, \quad \dot{z} = \frac{z[k+1] - z[k]}{\delta}. \tag{5.23}$$

Then by defining $u = -\nabla \tilde{f}(x)$, and substituting it and the expression in (5.23) into (5.20) and (5.21), we obtain that

$$x[k+1] = (\mathbb{I}_{(n-1)^2} - \delta \tilde{L})x[k] - \delta \tilde{L} z[k] + \delta u[k], \tag{5.24}$$

$$z[k+1] = z[k] + \delta \tilde{L} x[k]. \tag{5.25}$$

After introducing the tuning parameter γ to (5.24) and (5.25), we arrive at (5.13) and (5.14) as presented above.

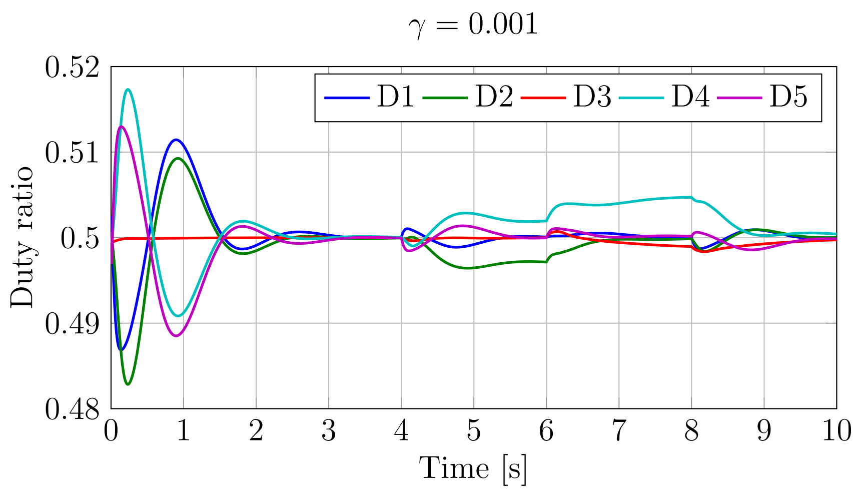
5.4 Discussion on Algorithm Characteristics

In this section, we consider a 6-sub-module, 5-DPP system as an example, and illustrate via computer simulations the characteristics of the proposed algorithm in terms of (i) update function parameter tuning, (ii) communication topology, and (iii) reconfiguration after communication failure.

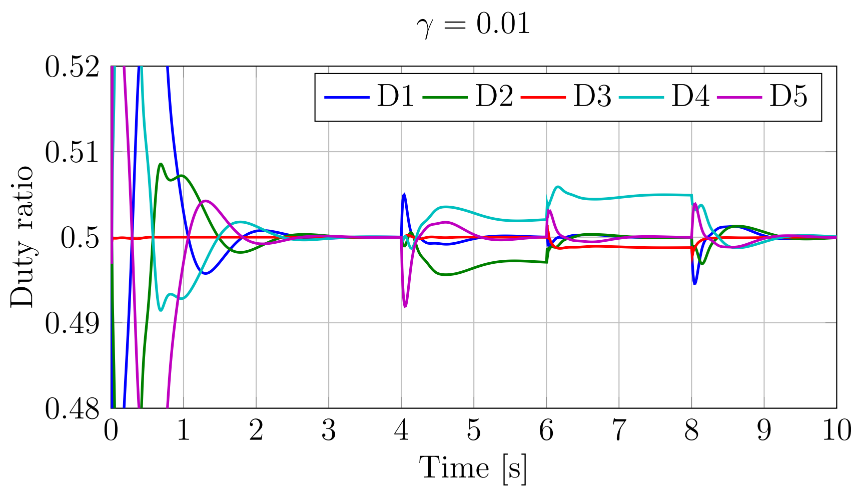
5.4.1 Choice of Tuning Parameter γ

In (5.5), the parameter γ determines the influence that the input vector $u[k]$ has on the state vector $x[k]$. Typically a relatively large value of γ renders a shorter response time but larger transient overshoot of duty ratios. As an example, Fig. 5.6a and Fig. 5.6b shows the duty ratio of a 6-sub-module, 5-DPP system evolving over time with different values of γ in the update function; the normalized irradiance profile used in this example is as follows

$$\begin{aligned} &\{100\%, 100\%, 100\%, 100\%, 100\%, 100\%\} \\ &\xrightarrow{t=4\text{ s}} \{100\%, 100\%, 80\%, 80\%, 50\%, 50\%\} \\ &\xrightarrow{t=6\text{ s}} \{100\%, 100\%, 100\%, 80\%, 50\%, , 50\%\} \\ &\xrightarrow{t=8\text{ s}} \{100\%, 100\%, 100\%, 100\%, 100\%, 100\%\}, \end{aligned} \tag{5.26}$$



(a) $\gamma = 0.001$.



(b) $\gamma = 0.01$.

Figure 5.6: Evolution of duty ratios computed by the distributed algorithm with fixed γ for a 6-sub-module, 5-DPP system.

with step changes occurring at $t = 4$ s, 6 s, 8 s as indicated. As shown in Fig. 5.6a and Fig. 5.6b, for different values of γ, the algorithm converges to the same steady-state value after each irradiance change. Compared to other transients, the duty ratio overshoot is very large at a "cold start", which corresponds to $t = 0$ s in Fig. 5.6b. [At "cold start", the algorithm is just activated without receiving any information of the system.] Although the entries of $x[k]$ can be initialized to 0.5, the entries of $z[k]$ can only be initialized to 0. Since $z[k]$ is intended to balance the input vector $u[k]$ in (5.5), the state vector $x[k]$ is very sensitive to $u[k]$ when $z[k]$ is small; in this case, a smaller γ is preferable to suppress the overshoot. Note that the slow convergence resulting from this small γ is not a problem since, in the field, "cold start" conditions correspond to turning on the DPP system, which typically happens only once per day. Once the system converges and $z[k]$ reaches steady state, a small γ is undesirable due to its slow response, and a larger γ can be used for faster tracking upon irradiance changes (also referred to as "hot start"). Duty ratio overshoot in a "hot start" condition is typically small despite a large γ since the $z[k]$ has already reached a steady state value. Therefore, some on-line tuning scheme for γ can be incorporated in the algorithm to improve its convergence properties; next, we discuss one such scheme.

To develop a method for smoothly changing the value of γ without introducing any disturbance to the duty ratio, we first note that after the algorithm converges to a steady state value, we have that

$$x[k+1] = x[k] = x^*, \tag{5.27}$$

$$z[k+1] = z[k] = z^*. \tag{5.28}$$

Substituting (5.27) and (5.28) in (5.13) and (5.14), respectively, yields

$$\tilde{L}z^* = \gamma u^*, \tag{5.29}$$

$$\tilde{L}x^* = 0, \tag{5.30}$$

which suggests that scaling γ and z^* by the same multiplicative constant does not affect the equilibrium point of the algorithm. Therefore, the on-line tuning scheme can modify γ without introducing a disturbance on the duty ratio as long as $z[k]$ is simultaneously changed by the same scaling factor; Fig. 5.7 illustrates the response of a 5-DPP system after such a tuning scheme is applied. The system "cold starts" with a small γ, and modifies the value of γ at $t = 3$ s without any noticeable disturbance to the duty ratios. The work in [54] presents the result of long-term measurement illustrating the transient effects of irradiance changes in solar PV applications, which can be used in the selection of γ for a suitable balance between speed and accuracy.

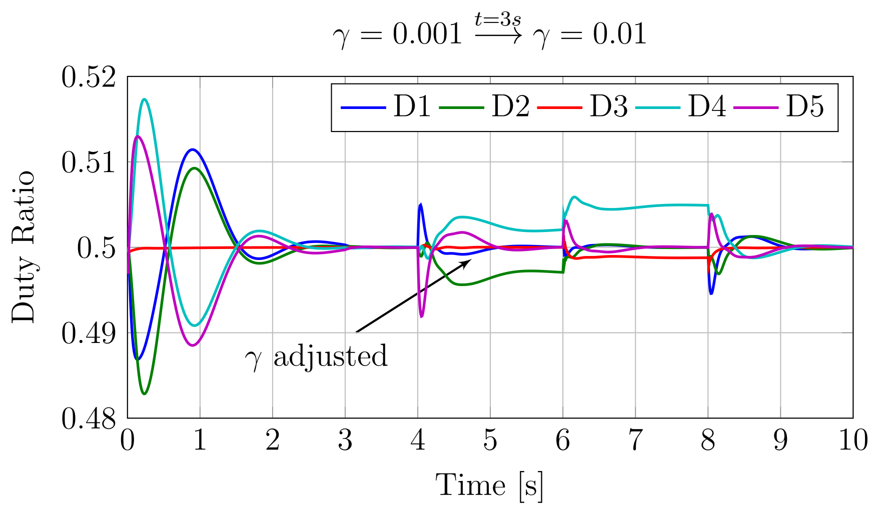


Figure 5.7: Evolution of duty ratios computed by distributed algorithm with adaptive γ for a 6-sub-module, 5-DPP system.

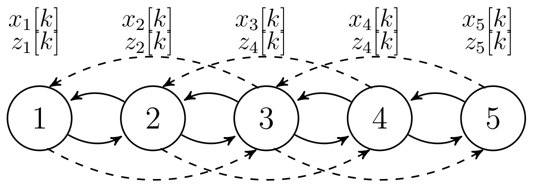


Figure 5.8: Graph of a 6-sub-module, 5-DPP system with redundant communications.

5.4.2 Impact of Communication Topology on Convergence Speed

The convergence speed of the algorithm can be improved by modifying the communication topology; Fig. 5.8 illustrates a directed graph representing the communication topology of a 6-sub-module, 5-DPP system. The same irradiance pattern in (5.26) is applied to the 6 sub-modules. By relying only on neighbor-to-neighbor communication (as captured by the solid line in Fig. 5.8), the evolution of duty ratios is relatively slow as has already been shown in Fig. 5.7. If redundancy is introduced to the communication topology, e.g., each DPP can communicate with its second closest neighbor(s) in addition to its direct neighbor(s) (as captured by the dashed line in Fig. 5.8), the convergence speed can be accelerated (as in Fig. 5.9) and the proposed algorithm can be easily adapted to accommodate this change. We only need to modify the Laplacian matrix L in the update functions in (5.13) and (5.14)

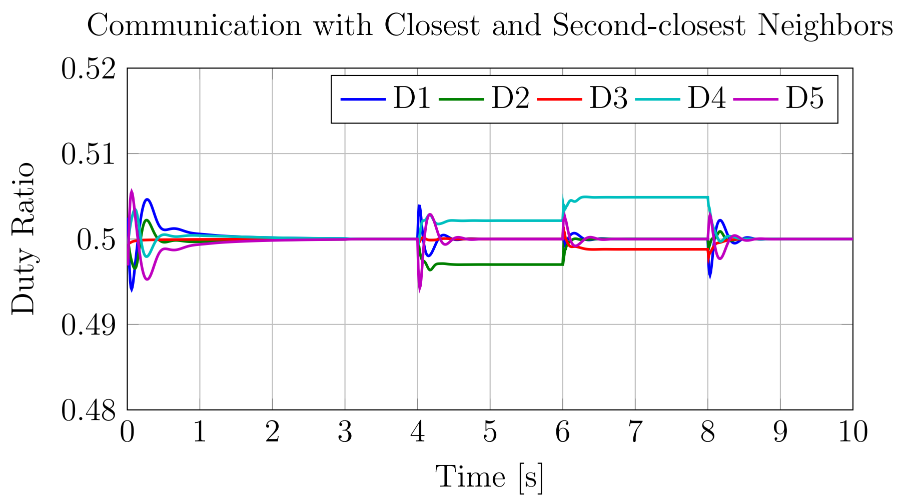


Figure 5.9: Evolution of duty ratios computed by distributed algorithm with redundant communications.

to reflect the new communication topology, which in this case is given by

$$L = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ -1 & -1 & 4 & -1 & -1 \\ 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{bmatrix}. \tag{5.31}$$

Besides L, no further modification to the algorithm is necessary. As illustrated in Fig. 5.9, the extra communication links significantly improve convergence speed. Further simulation indicates that as we add more communication links, the convergence speed of the algorithm further increases. Although it seems that additional communication hardware is required in the real system to introduce the modification as discussed above, this is not really the case. As introduced in Chapter 4, our hardware design, which integrates 3 DPPs on one printed circuit board (PCB), readily lends itself to the communication topology in which each DPP can communicate with at least its first, second, and third neighbors in both directions without extra hardware.

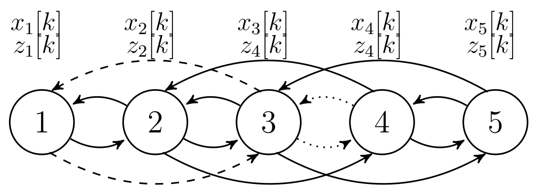


Figure 5.10: Graph of a 6-sub-module, 5-DPP system with communication failure.

5.4.3 Impact of Communication Topology on Reliability

As shown in [52], the proposed distributed algorithm converges to the optimal solution as long as the directed graph describing the communication topology remains strongly connected. Therefore, when the communication topology has redundancy, the algorithm can be reconfigured to work under communication link failures. Consider the 6-sub-module, 5-DPP system again with the communication topology shown in Fig. 5.10. Each DPP is communicating with its first and second closest neighbors. The Laplacian matrix of this graph is given in (5.31). If a communication link fails as illustrated by the dashed lines in Fig. 5.10, and this failure is detected by the local controllers of the DPPs on each end of the link, the two controllers involved can simply reconfigure the algorithm by modifying the corresponding row of the Laplacian matrix used in their update function to reflect this change in the communication topology. Thus, the new Laplacian matrix is given by

$$L = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 3 & -1 & -1 \\ 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{bmatrix}; \tag{5.32}$$

note that only the first and third row in the Laplacian matrix change. This change is completely local to the two DPPs on the ends of the failed communication link, i.e., DPP_1 and DPP_3, and affects only the update functions of these two DPPs. The update functions for other DPPs remain the same and continue to perform MPPT, since other rows of the Laplacian matrix do not change. If another communication failure occurs after the first one, e.g., as illustrated by the dotted line in Fig. 5.10, the Laplacian matrix can be

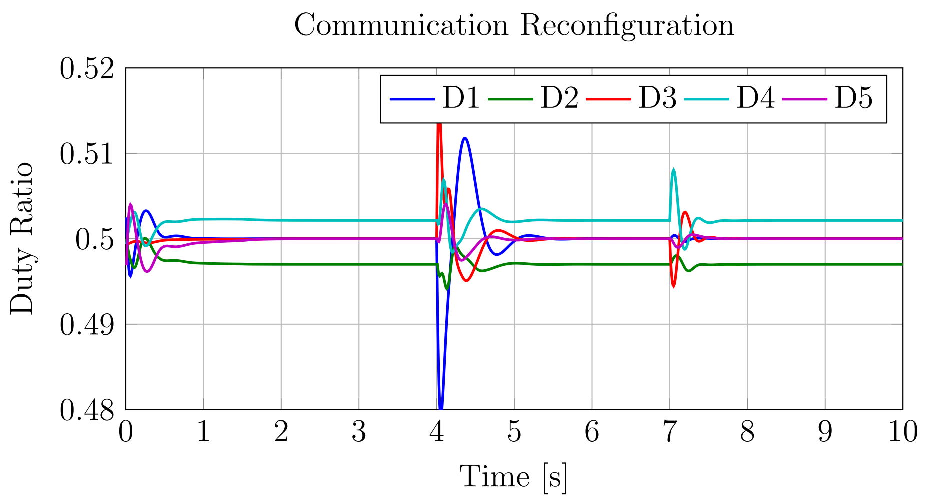


Figure 5.11: Evolution of duty ratios computed by distributed algorithm with communication failure reconfiguration.

modified again; this results in

$$L = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 2 & 0 & -1 \\ 0 & -1 & 0 & 2 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{bmatrix}. \tag{5.33}$$

Figure 5.11 illustrates the simulation result of this reconfiguration assuming unchanged irradiance pattern and the aforementioned communication link failures happening at $t = 4$ s and $t = 7$ s, respectively. As the simulation shows, after each communication failure, our algorithm is able to restore the duty ratio back to the optimal value after a short transient.

5.5 Practice Considerations for Algorithm Implementations

In simulation, the control algorithm can be directly implemented in Matlab in a relatively straightforward way. However, when implementing the algorithm in micro-controller, special care should be taken to avoid practical issues as highlighted below. The complete information on implementing the control algorithm can be found in the appendix.

5.5.1 Measurement Noise

The micro-controller uses an analog-to-digital converter (ADC) to sample the PV module voltage and converter it into a digital representation. In practice, the ADC will pick up some noise both from the power converter circuit and the micro-controller itself. To reduce noise in the measurement, in the hardware aspect, a low pass filter should be carefully designed to reduce the noise coupled from the power converter. The cut-off frequency of the low pass filter should be low enough to reject the switching noise from the converter, but also high enough to measure the voltage change due to MPPT control algorithm perturb action. The analog part of the micro-controller should be powered separately from the digital part (ideal from a dedicated linear regulator), and the analog ground should also be separated from the digital ground. In the software aspect, for one measurement, if the time allows, the program should take multiple ADC samples and average them.

5.5.2 Numerical Representation

Compared to a computer, the micro-controller has very limited computing capability. To perform the time-critical control task, the control algorithm has to be modified to reduce the computation requirement. In particular, the update equations developed in Chapter 5 are naturally represented in floating point numbers. However, the micro-controller computes floating point numbers much slower than fixed point numbers (integers). Moreover, the ADC measurements are represented by a 12-bit integer, and the duty ratio command written to the micro-controller PWM is also a 16-bit integer. For these reasons, it would be very desirable if all the calculation is done with integers. To do that, we can scale up the variables ($u[k]$, $x[k]$ and $z[k]$ in (5.13) and (5.14)) to a suitable, and scale the parameters in the update equations accordingly as well. The goal of the scaling is that all the variables and parameters are represented in integers with reasonable precision, such that they can be implemented as integer arithmetic in the micro-controller. In fact, with the scaling, multiple ADC samples (to reduce measurement noise) can be summed but not averaged (not divided by the number of samples). The update function will not be affected if the parameters in the update function are scaled accordingly. Since fixed number arithmetic can be carried out much faster than floating point arithmetic, this scaling technique can save much computation time of the micro-controller and speed up the iteration of the control algorithm.

CHAPTER 6

EXPERIMENT VERIFICATION OF THE DISTRIBUTED CONTROL ALGORITHM

In this chapter, the experiments performed to verify the proposed distributed control algorithm are presented. The experimental setup is introduced, followed by three case studies that compare the simulation result and experimental result to verify the effectiveness of the control algorithm.

6.1 Experimental Setup

To verify the proposed MPPT algorithm, the indoor experimental setup shown in Fig. 6.1 was developed. A power supply (HP 6631A) was connected in parallel with each PV sub-module to replicate the photo-generated current, providing an output I-V characteristic similar to the one that would result if the PV sub-module were to be illuminated by sunlight. Details about this method to conduct a repeatable indoor PV experiment can be found in [55]. Five DPPs implemented on 2 prototype boards (the prototypes with integrated power stage were used) were connected in parallel with a string of six PV sub-modules. Three of them are from a Sunmodule^{TM} SW 235 Poly PV modules, and the other three are from a SolarWorld Sunmodule^{TM} SW 245 Poly PV modules. These two modules are of the same series, which means they are manufactured in the same process, but they are rated for 235W and 245W respectively, which means their inherent characteristics are different due to the variation in the manufacturing process. The manufacturer bins them into different PV module models and sells these modules separately. In the field, typically only modules of the exact same model are used together, which may impose a logistics challenge in practice. In this experiment, we want to prove that DPP can compensate for the mismatch between PV submodules such that the PV manufacturers can perform a less stringent binning process or no binning at all, and thus reduce manufacturing cost of the PV modules. Therefore, these two different PV modules are chosen intentionally to emulate such a scenario. In theory, if we choose PV modules with even larger power rating difference, the power yield improvement with DPP will be even more significant. In practice, the mismatch produced by manufacturing variations is within a relatively small range, so the selection of PV modules in the setup can represent the general case. Besides, an electronic load (Agilent 6060B) acted as a central converter and controlled the string current. Two digital multimeters (Agilent 34401) were used to measure the string voltage and current. All power supplies, digital multimeters and the electronic load communicated with a computer through a GPIB interface. Micro-controllers on the two prototype boards communicated with each other and with the same computer through an I^2C interface. Note that for the following tests of the proposed distributed algorithm, the computer only received information from the micro-controllers for data logging and diagnostic purposes. The distributed algorithm, including data exchange and synchronization at each iteration, is fully implemented in the micro-controllers.

In the experiments, the current limits of the power supplies were set to certain percentages of the sub-modules' nominal short-circuit current to emulate the corresponding percentage of irradiance (e.g., a combination of 5 A, 4 A and 2.5 A was used to emulate an irradiance profile of 100%, 80% and 50% normalized value). For a given irradiance mismatch scenario, the string current was kept fixed at a certain value by the electronic load to emulate the behavior of a central inverter during its P&O interval. The evolution of duty ratios of all DPPs, along with the string current and voltage after the convergence of the algorithm, was recorded by the computer.

6.2 Case Studies

In this section, we verify the effectiveness of the proposed distributed algorithm through three case studies. We first present and compare simulation and experimental results from the simplest 2-DPP system to illustrate its basic features. Then, we focus on the experimental result of a six-sub-module 5-DPP system and analyze its tracking efficiency. Finally, we demonstrate the scalability of the algorithm by presenting simulation results for a system containing 31 DPPs and 32 sub-modules.

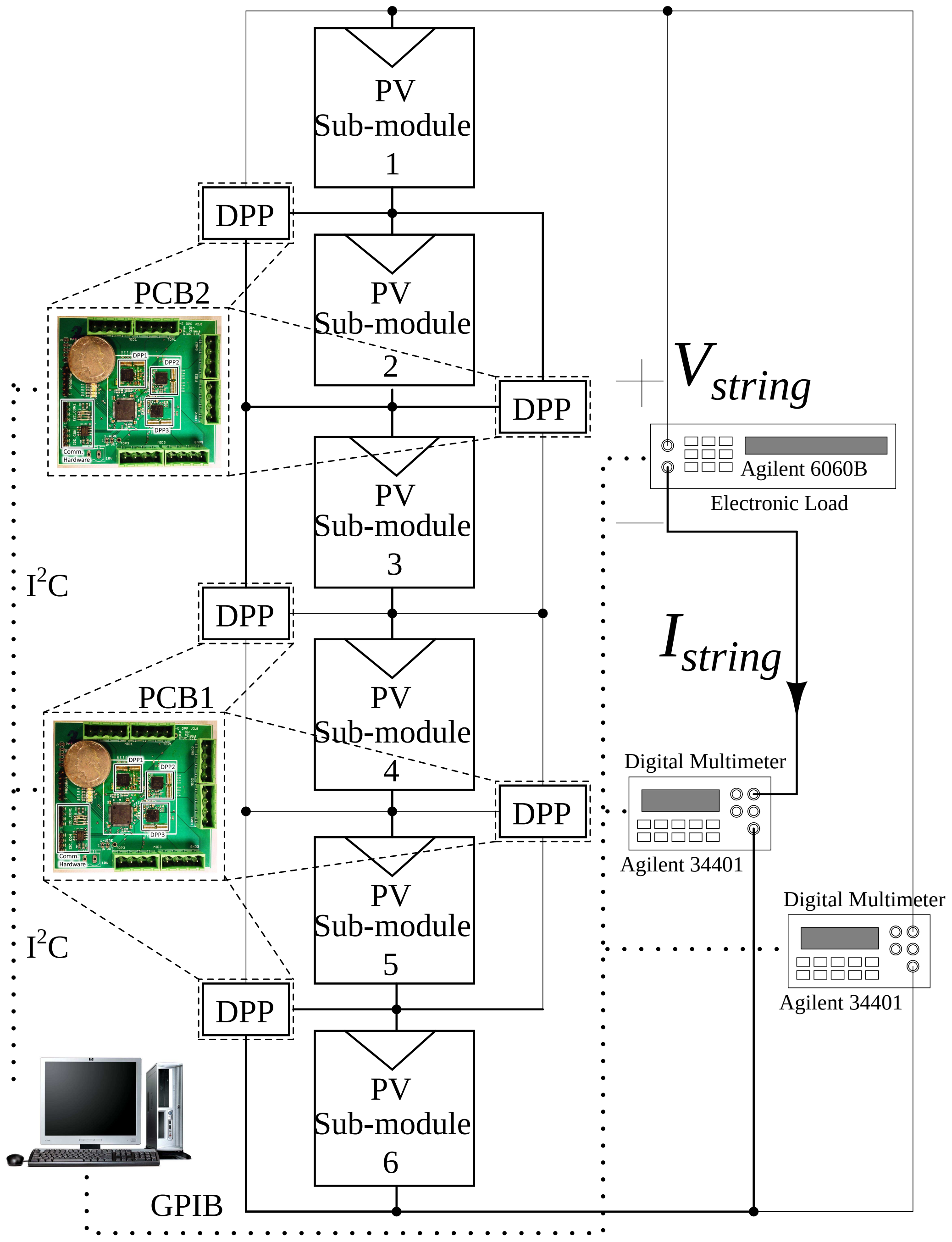


Figure 6.1: Experimental setup.

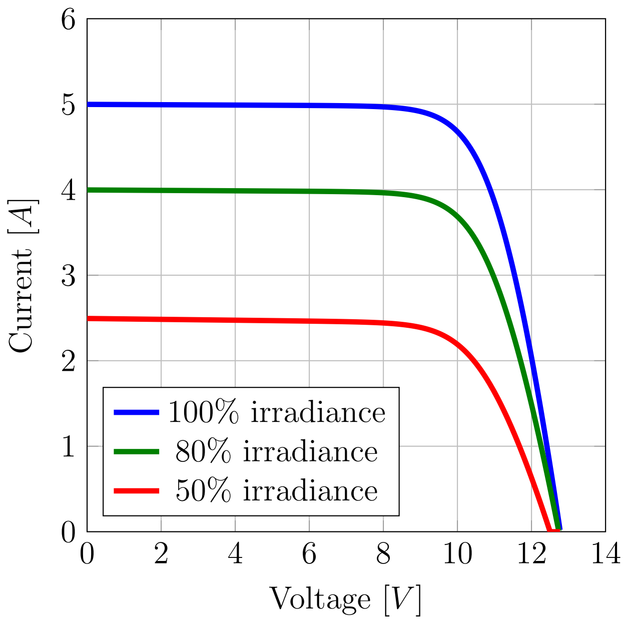
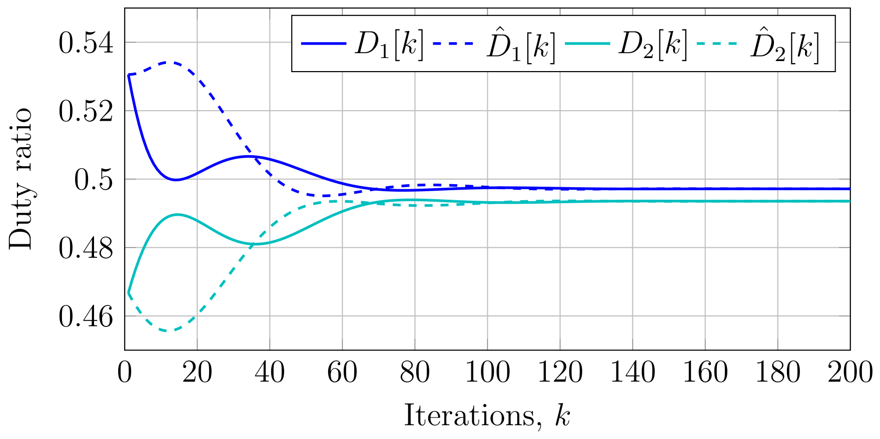


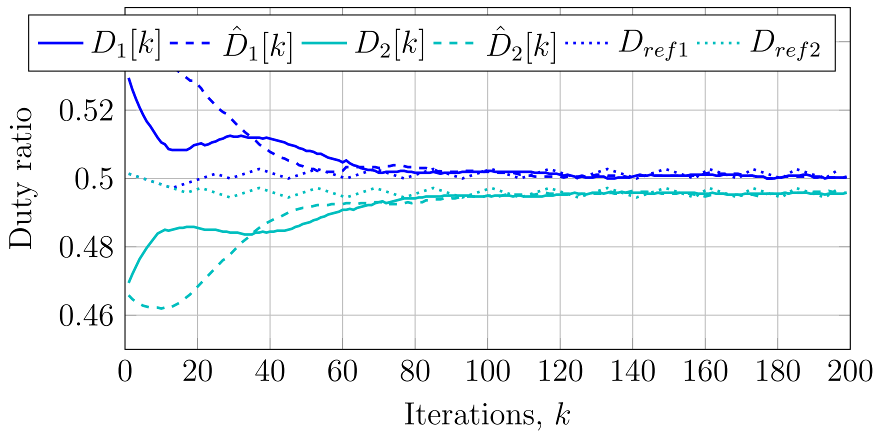
Figure 6.2: I-V characteristics of 3 sub-modules with $100\%, 80\%, 50\%$ normalized irradiance.

6.2.1 2-DPP system

To study the simplest 2-DPP case, we conducted experiments on the setup discussed above, but with only 3 sub-modules and compared the data with a MATLAB simulation. We set the irradiance pattern, string current, and initial duty ratios to be the same in the experiment and simulation, and recorded the evolution of the states of the algorithm. This test was performed for different conditions, and we found good matching between the simulation and the experimental data. For example, we set up a condition with $100\%, 80\%, 50\%$ normalized irradiance on sub-modules 1, 2 and 3, respectively (Fig. 6.2 shows the measured corresponding I-V curves); additionally, the string current was set to the current that corresponds to the global maximum power for this irradiance condition. Fig. 6.3a shows the evolution of the simulated duty ratios for this condition; Figure 6.3b shows the evolution of the duty ratios computed using the experimental setup with the same condition. As the figure shows, the state variables in both simulations and experiments quickly converge and roughly follow the same trajectory (also note that the duty ratios $D_1[k]$ and $D_2[k]$ settle to the same value as corresponding duty ratio estimates $\hat{D}_1[k]$ and $\hat{D}_2[k]$). To verify that the algorithm converges to the correct value, on the experimental setup, we also executed the centralized MPPT algorithm in [2], and recorded its duty ratio



(a) Simulation result.



(b) Experimental result.

Figure 6.3: Evolution of duty ratios computed by distributed algorithm for a 2-DPP system with irradiance pattern of 100%, 80%, 50%, $I_{string} = 3.3A$, $\delta = 0.1$ and $\gamma = 0.003$. Experimental result of the evolution of the centralized algorithm proposed in [2] is also plotted in dotted lines in as a benchmark reference for the final value after convergence.

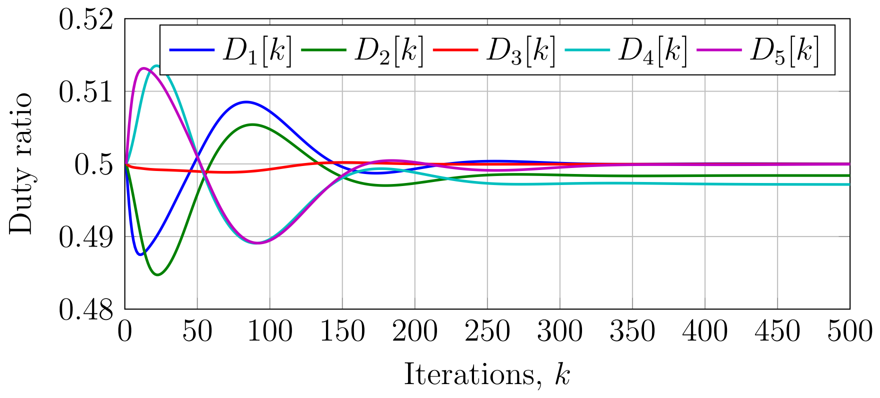
evolution as a benchmark (i.e., D_{ref1} and D_{ref2} in Fig. 6.3b). The final duty ratios of the proposed algorithm are almost identical to those of the centralized MPPT algorithm. Tests of other irradiance patterns and string current exhibited similar results.

6.2.2 5-DPP system

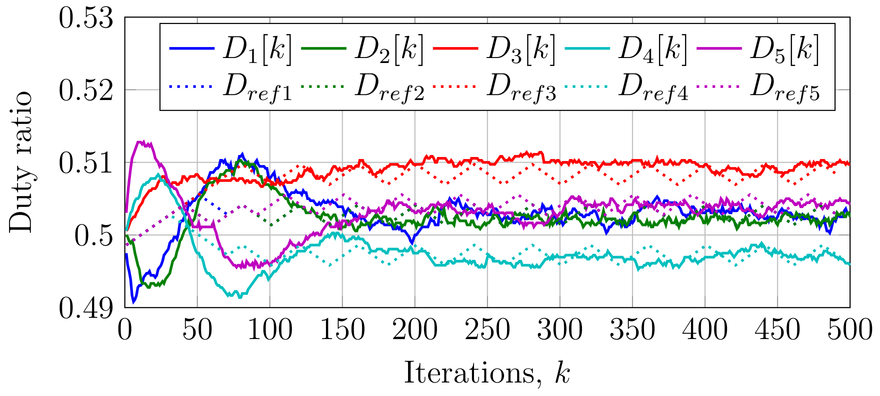
Similar to the 2-DPP case, we tested the proposed algorithm on a 5-DPP system for different conditions. Figures 6.4a and 6.4b show one example of the simulation and experimental result for the same conditions and initial duty ratios. Note that among all the state variables, only the actual duty ratios $D_i[k], i = 1, 2, \ldots, 5$, are plotted (the corresponding $\hat{D}_i[k]$s are omitted). Along with the distributed MPPT algorithm, the centralized MPPT algorithm proposed in [2] was also executed, and its evolution is plotted in Fig. 6.4b as a benchmark. Note that in both the 2-DPP and the 5-DPP systems, states in simulation and experiment follow very similar trajectories but do not converge to exactly the same final values; this is primarily due to two reasons. First, the simulation does not take into account the converter power losses. Second, as we used two types of PV modules in the experiment to emulate modules with less stringent binning, there is some variation in the I-V curves of the six sub-modules, which is not captured in the simulation.

At the same time, it is evident that the proposed algorithm and the centralized MPPT algorithm converge to the same final value. However, due to the noise in the voltage measurements, the duty ratio values still have small variations after convergence. With a smaller value of γ in (5.13), the variation around the steady-state value can be significantly reduced as $D_i[k]$ is less dependent on $u_i[k]$, but the algorithm will take more iterations to converge to the steady state value, as shown in Fig. 6.4c, which is consistent with the analysis in Section 5.4.1. In terms of hardware related issues, the noise in the voltage measurements can be reduced by averaging multiple reading, or filtering with larger capacitors at the ADC pins. However, both of these methods will increase the time needed for each iteration, and slow down the convergence of the algorithm. Similar to the choice of γ, the choice of δ in (5.13) and (5.14) can also be increased to speed up the convergence of the algorithm. However, δ represents a trade-off between speed and stability of the algorithm. Because the system is highly non-linear, δ can only be determined empirically, and we found δ should be kept smaller than 0.1 to guarantee stability in most scenarios.

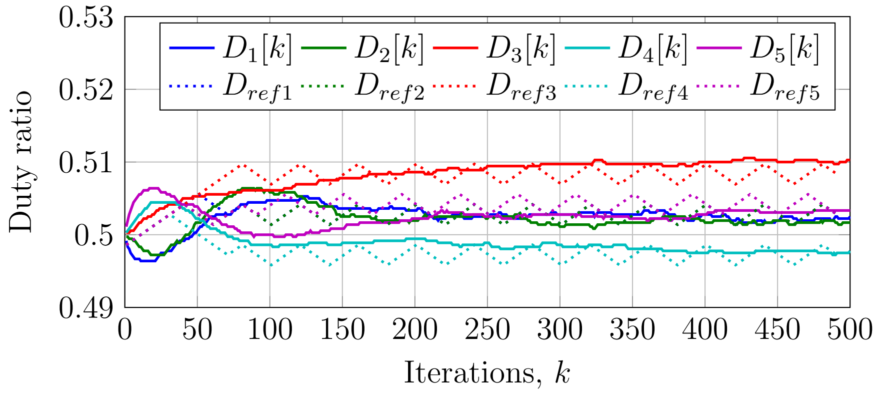
To further verify that the proposed algorithm effectively tracked the maximum power point, we measured the tracking efficiency, defined as the ratio of the actual power extracted to the maximum power of each sub-module for a given irradiance. To acquire the actual power extracted from each sub-module, we used two digital multi-meters (Agilent 34401) to perform synchronized, fast (0.006 NPLC) measurements on the current and voltage of each sub-module after the algorithm converged (the algorithm was kept running), so the tracking efficiency losses due to duty ratio perturbations,



(a) Simulation result, $\gamma = 0.002$.



(b) Experimental result, $\gamma = 0.002$.



(c) Experimental result, $\gamma = 0.0004$.

Figure 6.4: Evolution of duty ratios computed by distributed algorithm for a 5 -DPP system with irradiance pattern of 100%, 100%, 80%, 80%, 50%, 50%, $I_{string} = 3.3A$, $\delta = 0.1$ and different γ. Experimental result of the evolution of the centralized algorithm proposed in [2] is also plotted in dotted lines as a benchmark reference for the final value after convergence.

Table 6.1: Duty Ratio Comparison and Tracking Efficiency after Algorithm Convergence

Irradiance Pattern	100%, 100%, 80%, 80%, 50%, 50%	100%, 100%, 100%, 100%, 100%, 100%
Centralized MPPT Duty Ratio	0.503, 0.503, 0.508, 0.497, 0.503	0.499, 0.500, 0.502, 0.496, 0.501
Distributed MPPT Duty Ratio	0.502, 0.502, 0.510, 0.497, 0.504	0.499, 0.499, 0.502, 0.496, 0.501
Actual Sub-module Power $[W]$	45.42, N/A, 35.33, 36.59, N/A, 22.34	45.16, N/A, 44.96, 45.99, N/A, 46.54
Maximum Sub-module Power $[W]$	45.89, 45.63, 35.56, 36.67, 21.43, 22.35	45.89, 45.63, 45.39, 46.06, 45.00, 46.71
Tracking Efficiency	98.98%, N/A, 99.35%, 99.78%, N/A, 99.95%	98.41%, N/A, 99.05%, 99.85%, N/A, 99.63%

switching ripples and other transients could be taken into account in these measurements. The measurements were performed 100 times continuously. Sub-module power was calculated from each pair of current and voltage measurements, and averaged over the 100 times. On the other hand, to acquire the maximum sub-module power, an output voltage sweep was performed on each sub-module under the same irradiance, and the maximum output power was recorded. The resulting tracking efficiency was calculated as in in Table 6.1 for a few cases. Note that as introduced in Section 6.1, the six sub-modules used in the experiment were from two PV modules. Due to the internal wiring of the PV modules (three sub-modules connected in series in the junction box), the current of the sub-module in the middle of each PV module was inaccessible, i.e., the current of sub-module 2 and sub-module 4 could not be directly measured when the algorithm was running. Therefore, we only measured the power extracted from the other sub-modules.

6.2.3 31-DPP system

To demonstrate the scalability of the proposed distributed MPPT algorithm, we extended the simulation to model a system with 32 sub-modules and 31 DPPs. The evolution of the duty ratios found by each of the DPP local controllers using the distributed algorithm is shown in Fig. 6.5, for randomly chosen initial conditions, and for an irradiance pattern in which all sub-modules receive 100% irradiance. As Fig. 6.5 shows, the algorithm converges to the expected solution wherein the duty ratios are $D_i = 0.5$, $i = 1, \ldots, n-1$. While more iterations are required for a larger system to converge from a random initial point, some of the terms in $u_i[k] = \frac{\partial \varphi_i(D)}{\partial D_j} \approx \frac{\Delta \varphi_i(D)}{\Delta D_j}$, as defined in (5.11), are effectively zero when i and j are far apart. This allows parallelization of DPP duty ratio perturbations to speed up the convergence; we

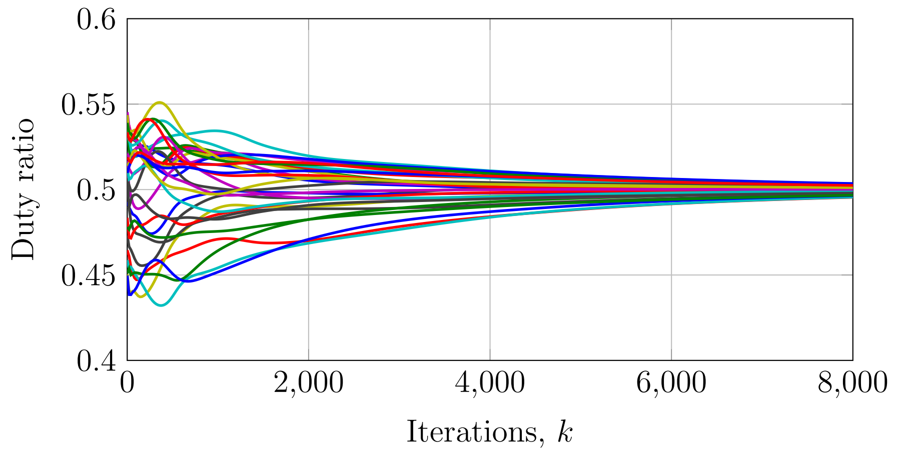


Figure 6.5: Evolution of duty ratios computed by distributed algorithm for 31-DPP simulation with 100% irradiance on all sub-modules and random initial duty ratios.

plan to explore this idea in future work.

6.3 Conclusion and Future Work

The simulation and experimental results for the 2-DPP system and the 5-DPP system match well with each other. The simulation result is validated and the experiment also proves the effectiveness of the proposed control algorithm. The simulation result of the 31-DPP system confirms the scalability of this approach. Although the experimental result seems promising, much work still needs to be done to improve this algorithm. Some of the important future work is summarized below.

6.3.1 Algorithm Parallelization

As mentioned above, the fact that converters far away from each other have limited effect on each other allows the possibility of speeding up the algorithm through parallelization. The update function (5.13) and (5.14) can simply drop far away turns or use some estimates. This can relax the requirement that all DPP converters have to perturb in sequence in a synchronous manner and allow converters far apart from each other to perturb in parallel. While this will speed up the algorithm, it will have a negative impact on the precision of maximum power point tracking. Therefore, the effect of parallelization has to be analyzed carefully before applying it in the system.

6.3.2 Field Test

Although the experimental result proves the effectiveness of the algorithm in a laboratory setting, the final test of the algorithm is to work in the real field and measure how much energy yield improvement is provided by the use of DPP. The field test condition poses great challenges to the hardware prototype. The hardware needs to be made reliable, easy to install and monitor, which has not been considered in the current version of the hardware design. Besides the converter hardware, a monitoring setup has to be established to collect data from the converters and PV modules under test. The setup has to be robust in the outdoor environment and easy to use and repair. All of these has not been established and requires much work in the future.

CHAPTER 7

A CENTRALIZED CONTROL ALGORITHM FOR DPP ARCHITECTURE IN A MICRO-INVERTER SYSTEM

In this chapter, the work on merging DPP into a micro-inverter system is presented. This work is different from the previous discussion where DPP is applied in a long PV string typically found in utility scale applications. Instead, this application typically involves only three PV submodules and a micro-inverter instead of a central inverter. This new application creates a very different set of constraints in the DPP system, and thus is specially considered in this chapter.

7.1 Application Background and Requirement

As introduced in Chapter 2, large PV arrays with DPP converters and a central inverter are well suited to utility scale PV systems. At the same time, micro-inverters are a popular solution for residential PV installations, where the environment is more complicated and thus the partial shadings and other types of mismatch are more frequent and severe. However, current micro-inverter configurations do not compensate for mismatch between sub-modules. When partial shading exists between submodules, the micro-inverter cannot exact maximum power from all sub-modules, and its maximum power point tracking algorithm might be trapped in one of the local maxima.

To compensate for mismatch between sub-modules, there are a few possible solutions. The first is to make a submodule level micro-inverter, which is never used as the dollar per watt cost of this solution can be prohibitively high. A more viable solution is a submodule DC optimizer, but a DC optimizer has to process the full power of each sub-module, and the shortcoming of this solution has been analyzed in Chapter 2. In a DC optimizer system, the system efficiency cannot be higher than the DC converter efficiency, so the overall system efficiency is limited.

To improve the power yield of micro-inverters, in this work, a DPP solution is proposed to seamlessly combine commercial micro-inverters with DPP converters to address sub-module mismatch while introducing minimal additional hardware cost. The control scheme required to interface DPP converters with a micro-inverter to perform sub-module level MPPT is also analyzed. The operation principle and control algorithm for DPP have been explored in previous chapters and by other literature [2,16,17,21] in the context of a long PV array with multiple modules. For such a system, as mentioned in Chapter 2, there always exists a trade-off between precise MPPT and different aspects of the control overhead. For example, [21] presents a distributed algorithm to perform MPPT but requires local current measurement at the sub-module level, which may introduce extra power losses. Alternatively, the distributed algorithm presented in Chapter 5 requires no local current measurement but relies on neighbor-to-neighbor communication between DPP converters. On the other hand, the control schemes presented in [16, 17] require minimal control overhead, but perform only *near* MPPT without seeking the true maximum power point. While the aforementioned trade-off is fundamental for a system with a large PV array, it is not the case for a micro-inverter system. As will be shown in this chapter, in a micro-inverter system where the length of the PV string is limited to only three sub-modules, the control algorithm for true MPPT can be implemented with very little additional overhead.

7.2 Control Algorithm Formulation

Figure 7.1 provides a schematic of our proposed solution for the micro-inverter system with DPP converters. In the implementation considered here, the DPP converters are bidirectional buck-boost converters that enforce a desired voltage ratio between neighboring sub-modules to adjust the operating point of sub-modules to their maximum power point voltage. When mismatch exists between sub-modules, the DPP converters manage the differential power during this process. At the same time, most micro-inverters run module-level MPPT based on a voltage-referenced perturb and observe (P&O) algorithm, in which the micro-inverter temporarily enforces a certain

PV module voltage V_m. Therefore, the micro-inverter can be considered a controlled voltage source. When the micro-inverter executes its P&O algorithm, it perturbs the module voltage periodically, observes the resulting PV module current and calculates the change in the PV module output power to decide the direction of the next voltage perturbation. While it may appear at first glance that significant coupling exists between the operation of the micro-inverter and that of the DPP converters, this is not the case in practice. As shown in Chapter 4, the DPP converters can be designed to operate at a high switching frequency (on the order of hundreds of kHz), and thus have a much smaller time constant than the perturbation interval of V_m (typically on the order of hundreds of Hz). For the DPP converters, the PV module voltage can be considered temporarily fixed by the micro-inverter at any given time. Since the power output of the PV module is

$$P_m = V_m \times I_m, \tag{7.1}$$

with a given module voltage, the power output of the PV module is maximized when the module current is maximized, regardless of the mismatch

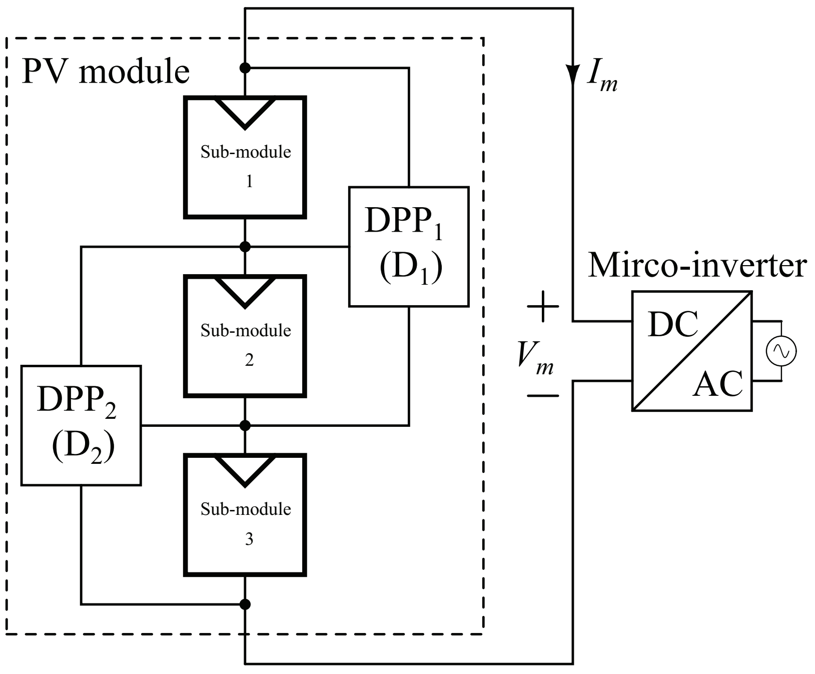


Figure 7.1: Schematic of the proposed micro-inverter system with DPP converters.

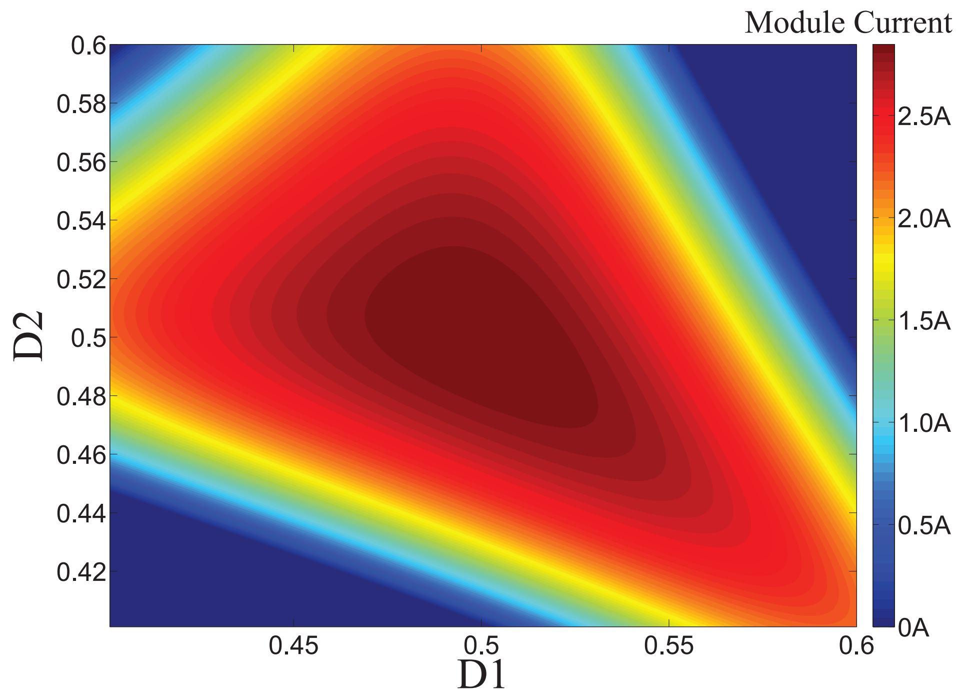


Figure 7.2: Visualization of the module current I_m as a function of DPP converter duty ratios $(D1, D2)$ with a normalized irradiance profile of 100%, 80%, 50% on three PV sub-modules and module voltage fixed at 30V.

between sub-modules. In this case, for given environmental conditions (irradiance, temperature, etc.), the module current is only a function of the duty ratios of the two DPP converters, (D_1, D_2), i.e.,

$$I_m = f(D_1, D_2). \tag{7.2}$$

Therefore, for a temporarily fixed V_m, the control problem for the DPP converters can be simplified to

$$\begin{aligned} \text{maximize} \quad & I_m(D_1, D_2), \\ \text{subject to} \quad & V_1 + V_2 + V_3 = V_m. \end{aligned} \tag{7.3}$$

As an example, Fig. 7.2 shows a simulation result that visualizes I_m as a function of (D_1, D_2) in a contour plot with a normalized irradiance profile of 100%, 80%, 50% on the three sub-modules of one PV module. It is clear from Fig. 7.2 that this function $I_m = f(D_1, D_2)$ has only one maximum. This property can also be verified by studying the concavity of the function, which holds as long as the I-V curve of each sub-module is concave.

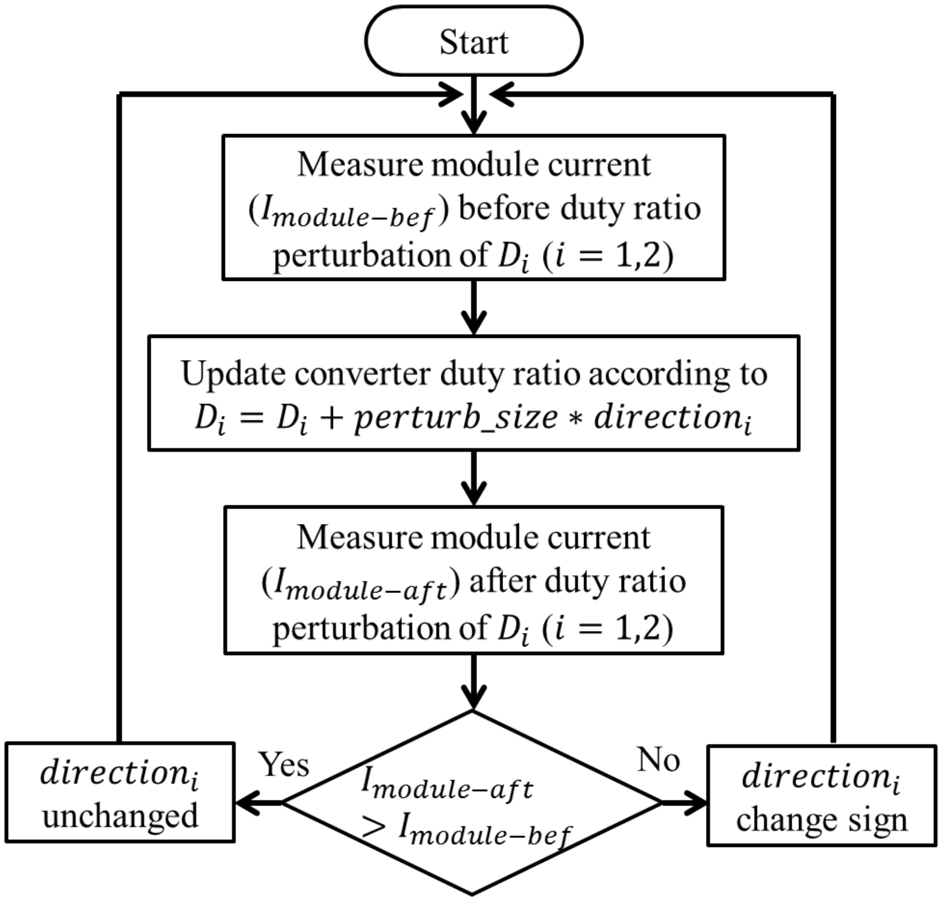

Start
Measure module current ($I_{module-bef}$) before duty ratio perturbation of D_i ($i = 1,2$)
Update converter duty ratio according to $D_i = D_i + perturb\_size * direction_i$
Measure module current ($I_{module-aft}$) after duty ratio perturbation of D_i ($i = 1,2$)
$I_{module-aft} > I_{module-bef}$
Yes
No
$direction_i$ unchanged
$direction_i$ change sign

Figure 7.3: Flow chart of the control algorithm for DPP converters.

From Fig. 7.2 it is clear that, similar to the control algorithm demonstrated in [2], the DPP converters can maximize the module current I_m by a DPP converter P&O algorithm that perturbs the duty ratios (D_1, D_2) and observes the change in I_m, as illustrated by Fig. 7.3. The DPP converter P&O algorithm is implemented in a relatively fast control loop while the micro-inverter P&O algorithm is implemented in a relatively slow control loop. These two control loops operate independently from each other. As long as the fast control loop is at least ten to twenty times faster than the slow control loop, the module voltage V_m can be treated as temporarily fixed for DPP converters, and thus the proposed DPP control algorithm is capable of sub-module MPPT as analyzed above. A more detailed example of how these two control loops interact with each other is provided by the experimental result presented in the next chapter.

7.3 Algorithm Implementation Considerations

It should be noted that the measurement of module current I_m is not directly available to the DPP converters without added current sensing circuitry. In a large PV system with multiple modules, this problem is a major constraint and certain communication is required between each DPP converter and the central inverter of the PV array. However, easier solutions exist for the micro-inverter system considered in this work. Typically, each micro-inverter interfaces with only one PV module and is often installed immediately on the backside of the PV module. The micro-inverter is thus physically close and accessible to the DPP converters. Since the micro-inverter needs to measure the module current I_m as well, the DPP converters can share the same current sensing device (such as a current sensing resistor) with the micro-inverter, though a separate amplification and filtering circuit may be needed for the DPP converters, since they need to sense the current change faster than the micro-inverter. In fact, the DPP converters can be merged into the micro-inverter circuit board and share the same control unit, since the control unit typically has enough surplus resources to control the DPP converters as well. In this way, DPP can be incorporated into the micro-inverter system with minimal additional hardware cost.

On the other hand, even if the micro-inverter is not accessible to DPP converters for some reason, there is a secondary control option. As can be seen from Fig. 7.2 and other similar simulations, when the mismatch between sub-modules is not severe, the maximum point is often close to $(0.5, 0.5)$ duty ratios. Therefore, voltage equalization (operating the DPP converters at a fixed 0.5 duty ratio) will often recover most of the power losses due to sub-module mismatch, as discussed in [16–18]. In this way, the DPP converters can still perform *near* MPPT as a stand-alone component without any current measurement.

CHAPTER 8

EXPERIMENT VERIFICATION OF THE CENTRALIZED CONTROL ALGORITHM IN A MICRO-INVERTER SYSTEM

In this chapter, the experimental result for the DPP enhanced micro-inverter system is presented. The experimental setup is first introduced, followed by experimental data and discussions.

8.1 Experimental Setup

We developed an experimental setup as shown in Fig. 8.1 to verify the effectiveness of the proposed method and to demonstrate the ability of the DPP converters to work with a commercial micro-inverter. In the setup, a Sunmodule^{TM} SW 235 Poly PV module consisting of 3 sub-modules was used. Each sub-module was connected in parallel with a HP6634A DC power supply (omitted in Fig. 8.1) in current-limited mode to emulate sunlight illumination. According to [55], the current-limit value (i.e., the amount of current sourced by each DC power supply) determines the intensity of the emulated irradiance on each sub-module. This method enables us to conduct the initial stage of our experiments in a more controllable and repeatable manner compared to outdoor PV experiments, where environmental factors constantly change. The PV module was connected to the DC side of a Pantheon^{TM} II micro-inverter from SolarBridge Technologies. The P&O algorithm of the micro-inverter was slowed down to about 1Hz for data capture purposes. A 112-AMX AC power supply that generated a 220V sinusoidal voltage to emulate the grid was connected to the AC side. The micro-inverter, together with the AC power supply, also powered a 370W load consisting of four light bulbs that were also connected to the AC side. Two Agilent 34401 multimeters were used to measure the PV module voltage and current on the DC side. During the entire experiment, the micro-inverter was set to operate autonomously.

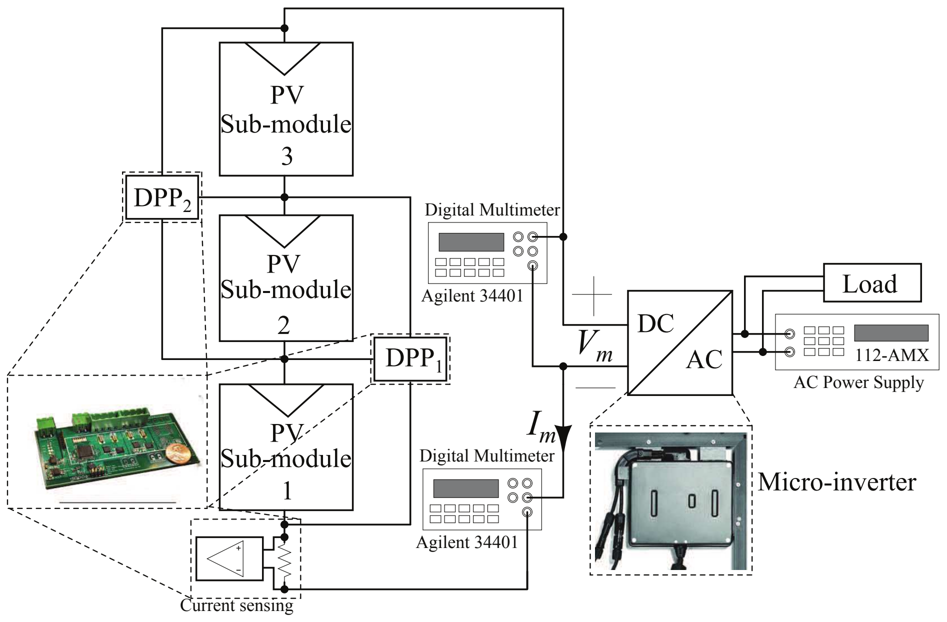


Figure 8.1: Experimental setup for the micro-inverter system.

8.2 Experiment Description and Results

In the experiment, mismatch between sub-modules (e.g., an irradiance profile of 100%, 50%, 100% normalized value) was emulated. For the first part of the experiment, an HP6060B electronic load was connected to the PV module instead of the micro-inverter to perform a DC voltage sweep to characterize the P-V curve of the PV module with and without the DPP converters running. The resulting P-V curve is plotted in Fig. 8.2. Note that without DPP, the bypass diode of the weak sub-module conducted and created a non-concave module P-V curve with local maxima, which can trap the MPPT operation of the micro-inverter. With the DPP converters running, the mismatch was compensated for by the DPP converters without turning on the bypass diode, and the resulting P-V curve was concave. When the DPP converters were operating, this concave P-V characteristic was the characteristic effectively seen by the micro-inverter. Then, the PV module under the same emulated irradiance condition was tested with the micro-inverter, and the DPP converters were turned on half-way through the test to show the improvement in power output. As shown in Fig. 8.3, without the DPP converters running, the micro-converter was trapped by the local maximum as expected, incurring significant power losses due to the mismatch. Once the DPP converters turned on, the micro-inverter quickly converged to the new operating

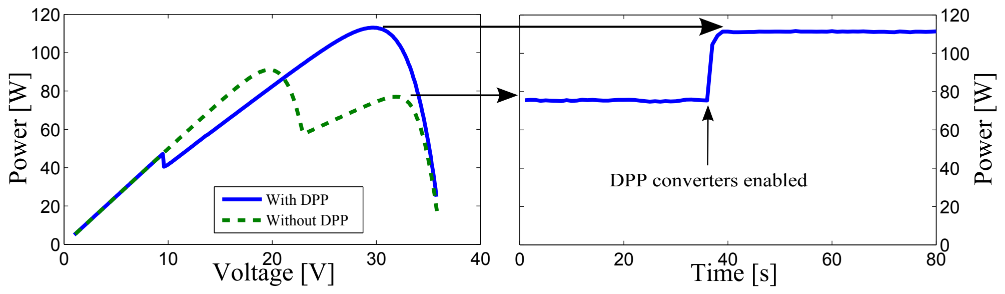
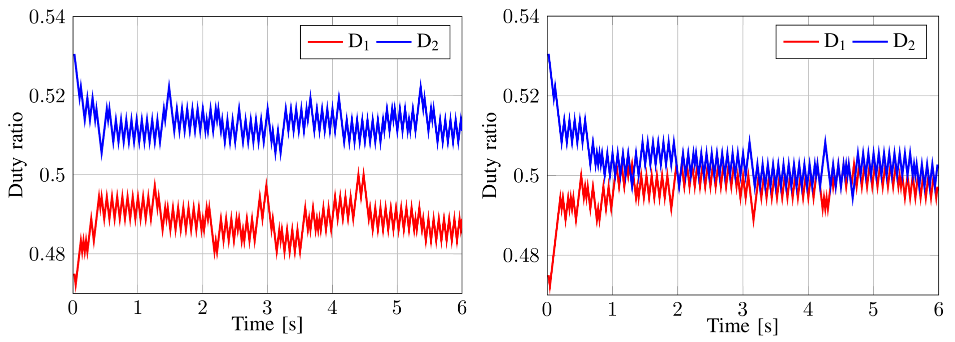


Figure 8.2: Plot of PV module power versus module voltage with and without DPP converters (using electronic load).

Figure 8.3: Plot of PV module power versus time (using micro-inverter). The DPP converters are enabled at about $t = 35$s.



(a) Irradiance condition: 60%, 100%, 60%

(b) Irradiance condition: 100%, 80%, 60%

Figure 8.4: Evolution of the DPP converter duty ratios (D_1, D_2) after turn-on. (D_1, D_2) was initialized at (0.52, 0.48) at turn-on.

point and the power losses were recovered. The evolution of the duty ratios of the DPP converters, (D_1, D_2), was recorded in Fig. 8.4. The duty ratios were initialized at a random point when the DPP converters were turned on, and they quickly converged to and oscillated around a steady state operating point. The steady state operating point changed approximately once per second since the P&O algorithm of the micro-inverter was slowed down to about 1Hz as previously mentioned.

APPENDIX A

SCHEMATIC DRAWING, PCB LAYOUT AND COMPONENT LISTING FOR THE DPP CONVERTER WITH INTEGRATED POWER STAGE

A.1 Schematic Drawing

To improve visibility, the schematic drawing of the converter is split into 4 parts that are shown separately in Figs. A.1-A.4.

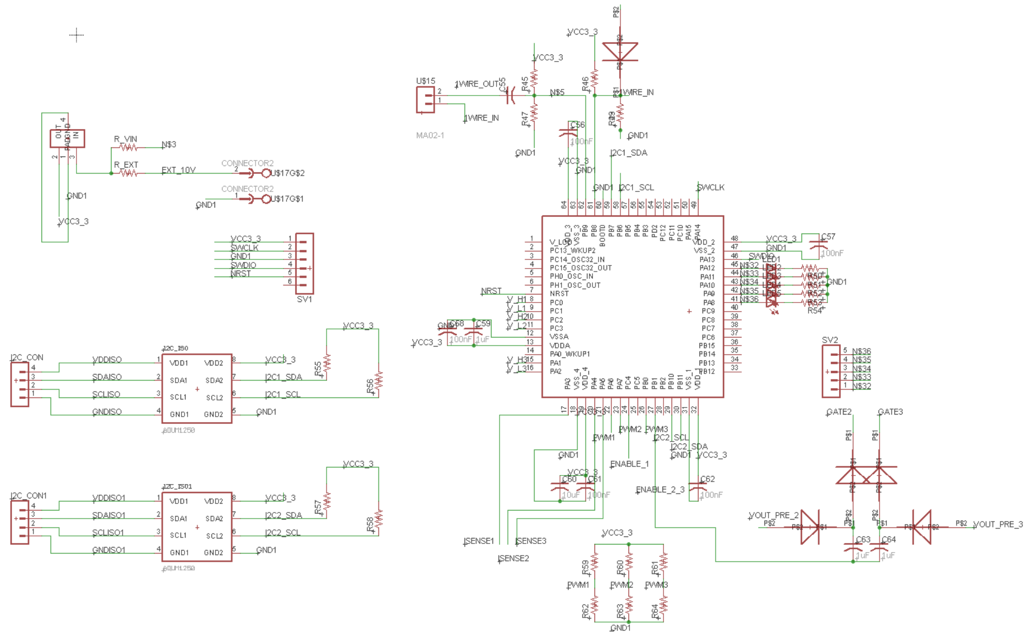


Figure A.1: Schematic drawing of the micro-controller with peripheral circuits.

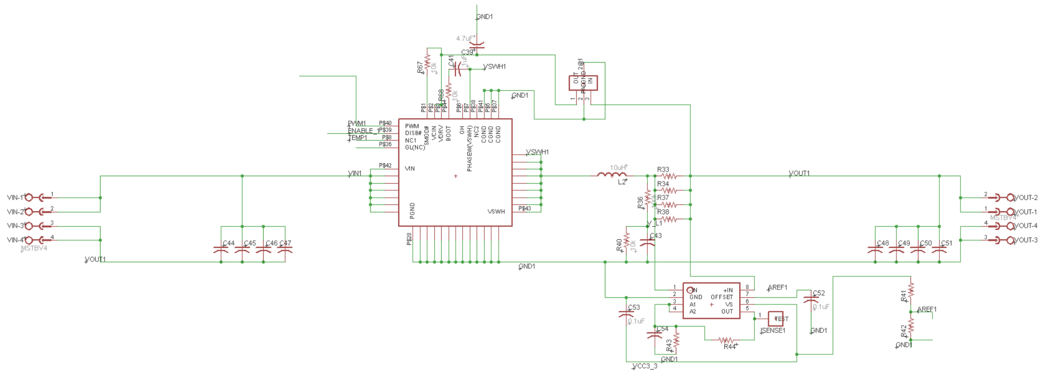


Figure A.2: Schematic drawing of the first converter (phase 1).

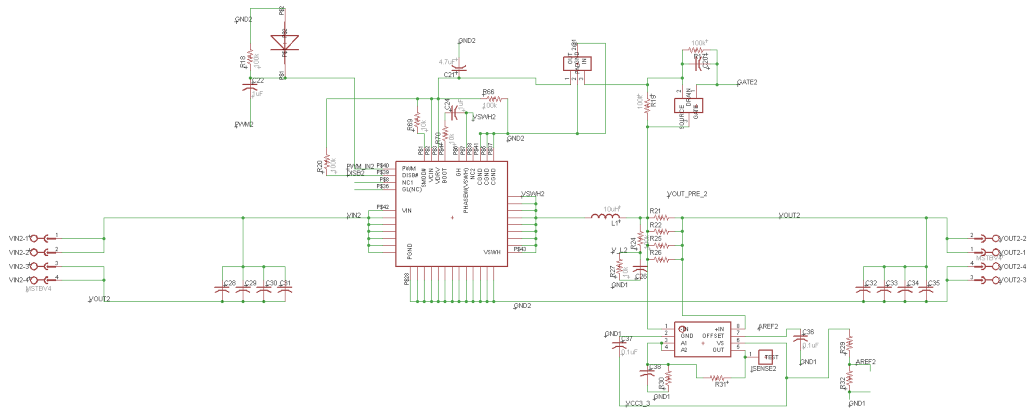


Figure A.3: Schematic drawing of the second converter (phase 2).

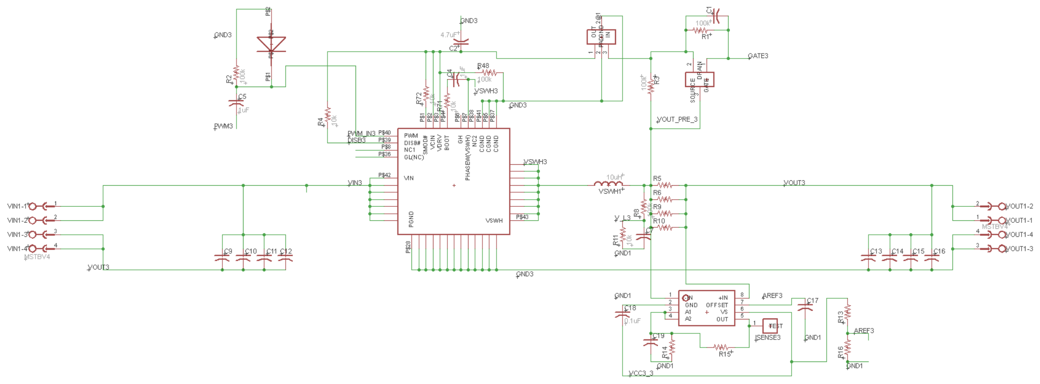


Figure A.4: Schematic drawing of the third converter (phase 3).

A.2 PCB Layout

The PCB layout is shown in Figs. A.5-A.12.

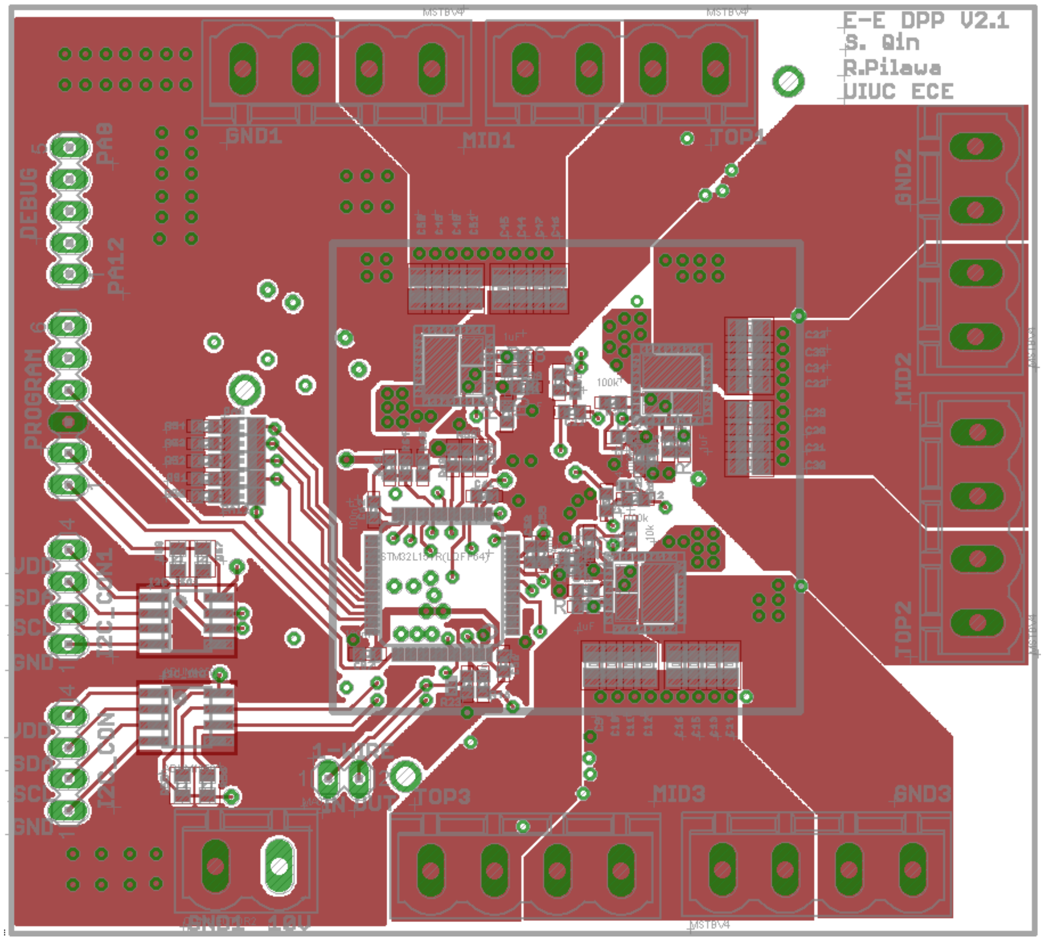


Figure A.5: PCB layout top view.

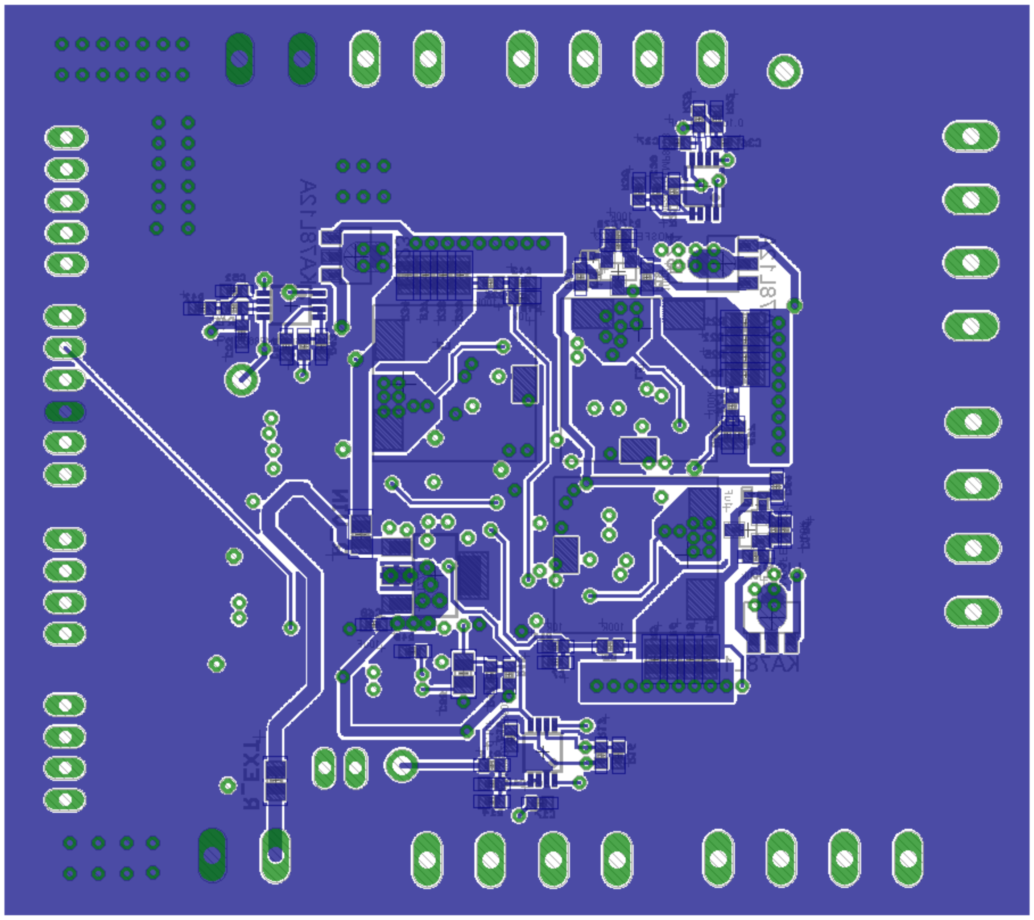


Figure A.6: PCB layout bottom view.

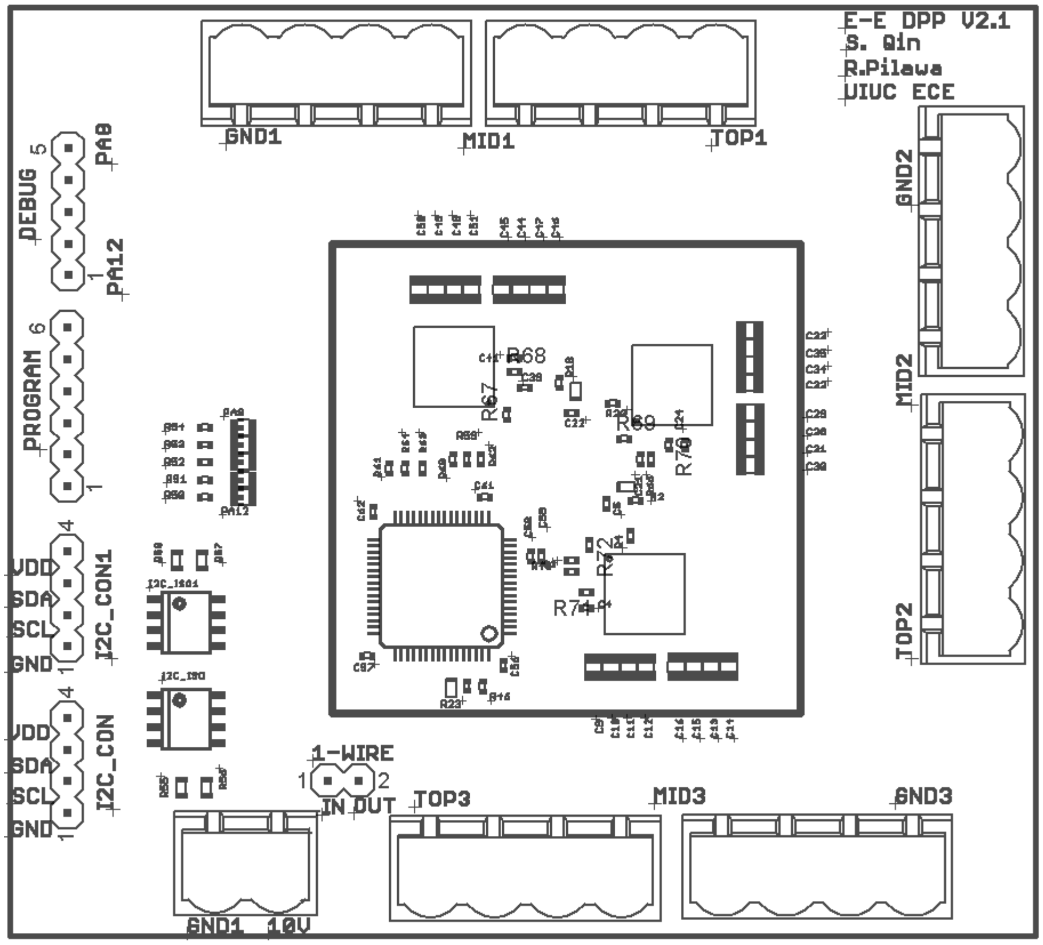


Figure A.7: Top silkscreen.

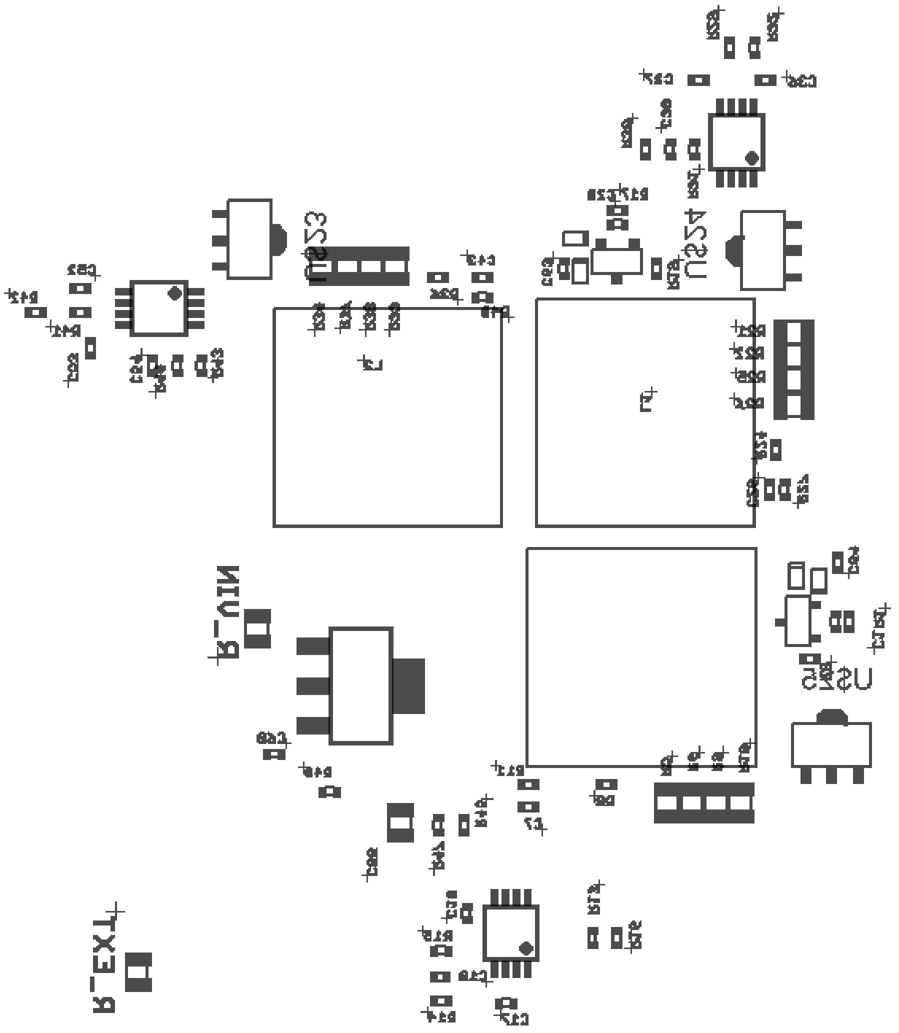


Figure A.8: Bottom silkscreen.

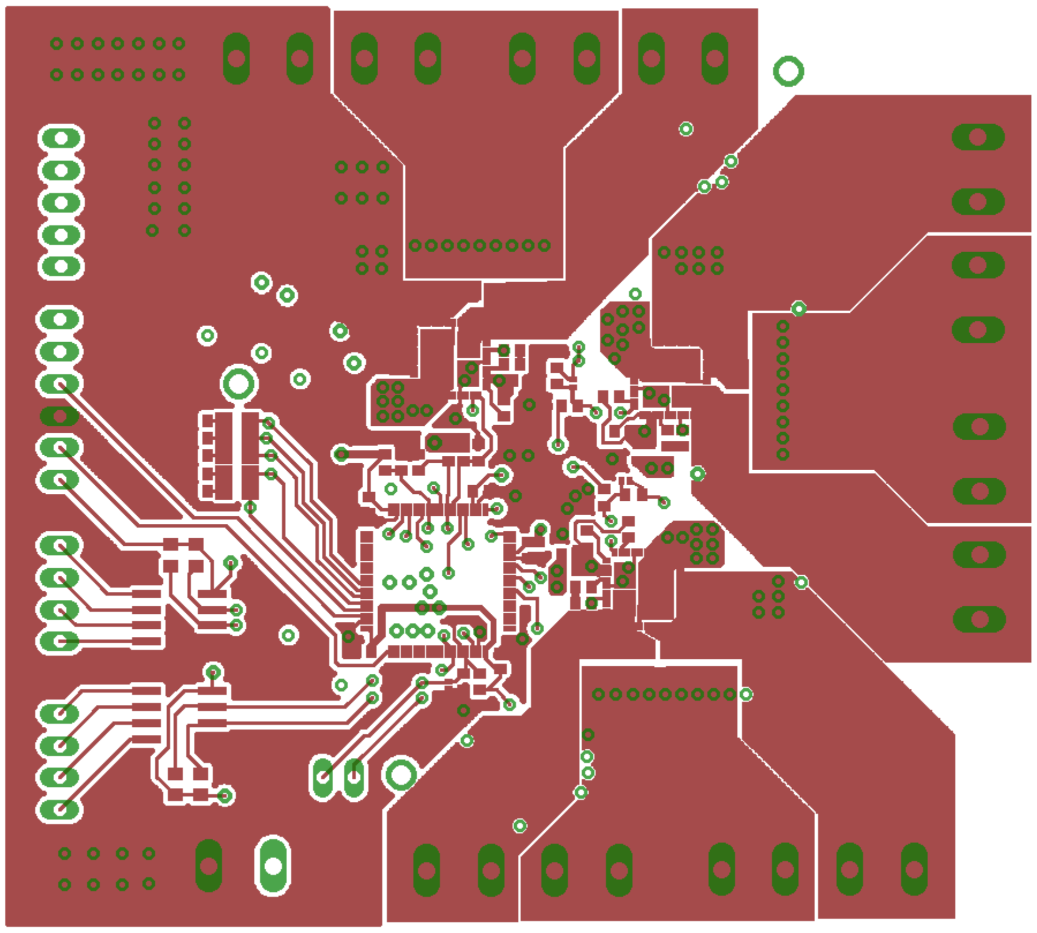


Figure A.9: Top-layer copper.

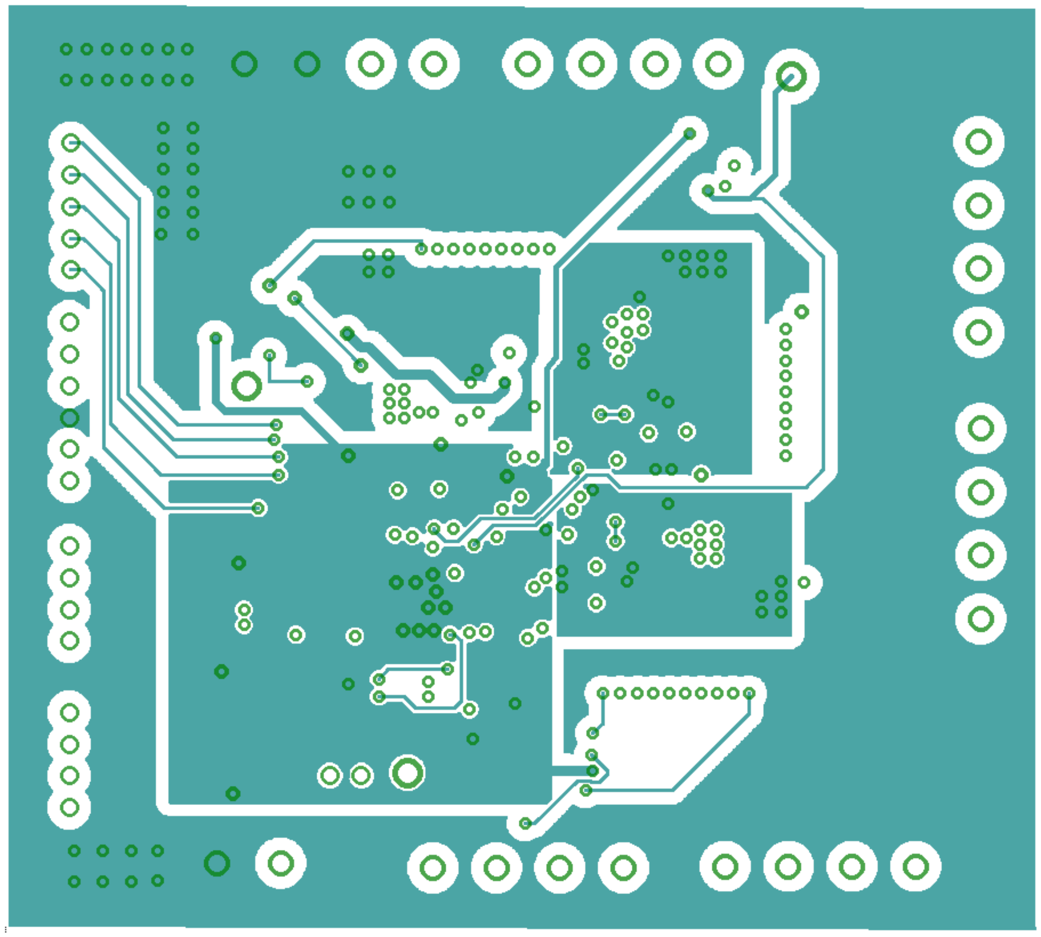


Figure A.10: PCB middle layer 1.

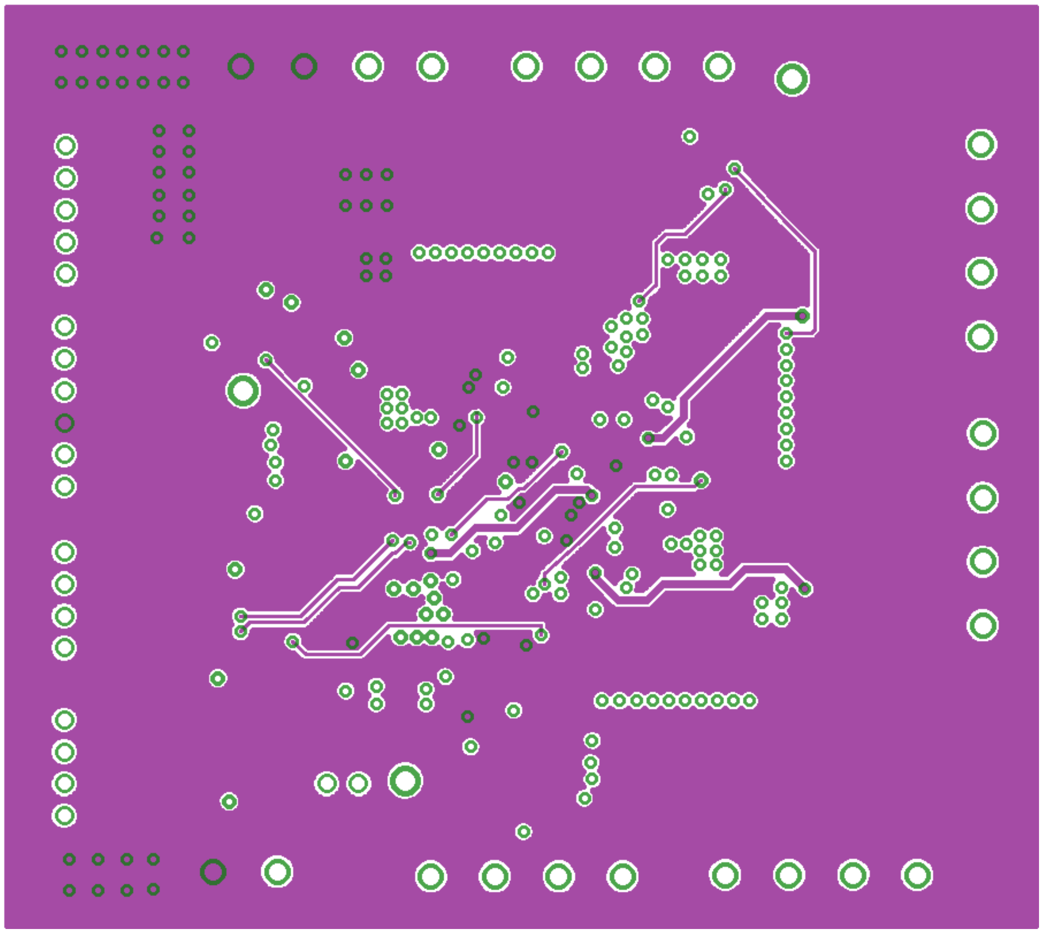


Figure A.11: PCB middle layer 2.

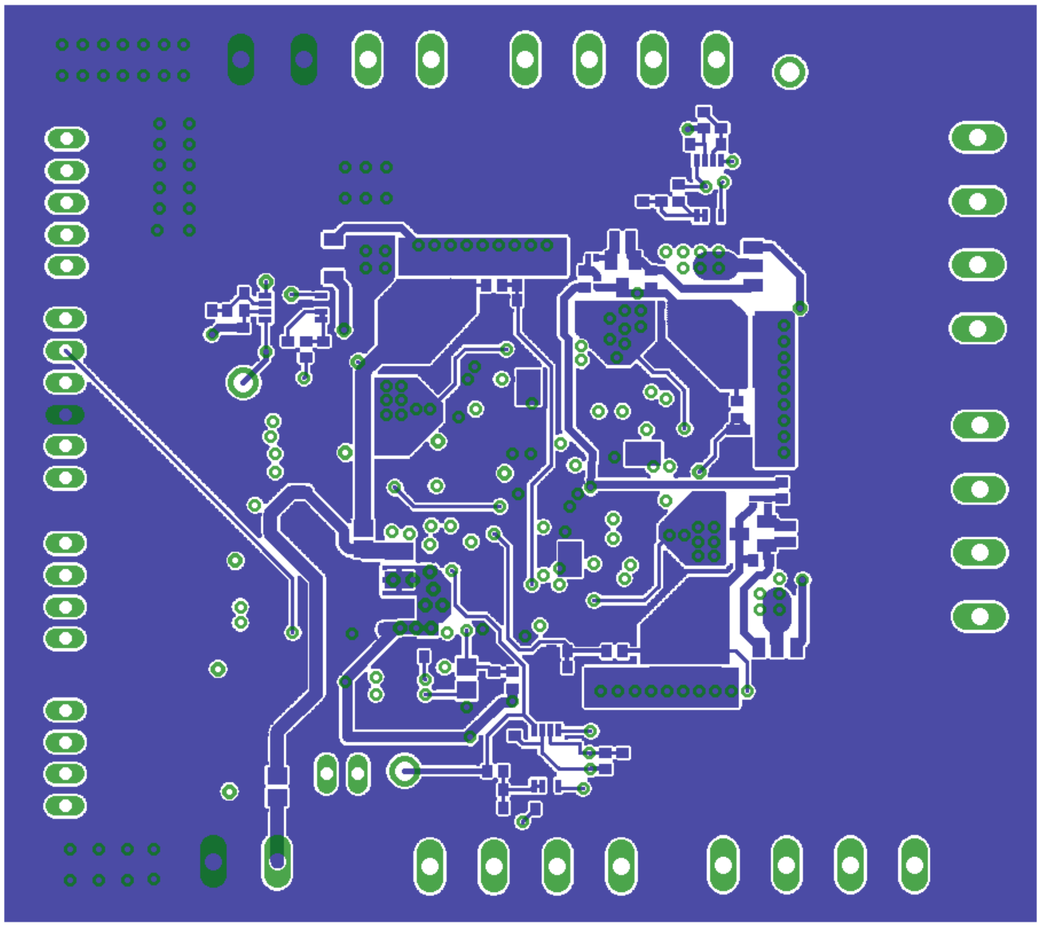


Figure A.12: Bottom-layer copper.

APPENDIX B

SCHEMATIC DRAWING, PCB LAYOUT AND COMPONENT LISTING FOR THE DPP CONVERTER WITH INTEGRATED POWER STAGE

B.1 Schematic Drawing

The schematic drawing of the converter is shown in Figs. B.1. .

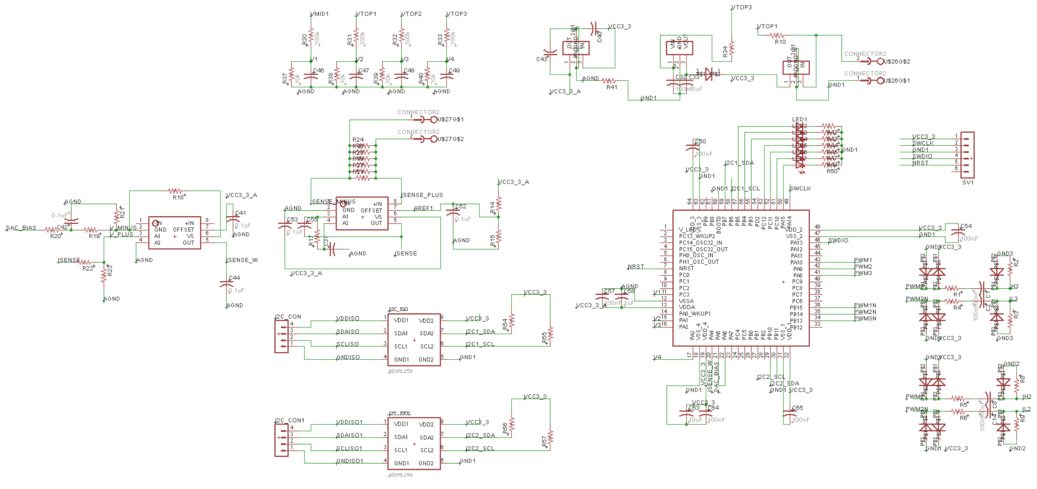


Figure B.1: Schematic drawing of the micro-controller with peripheral circuits.

B.2 PCB Layout

The PCB layout is shown in Figs. B.2-B.10.

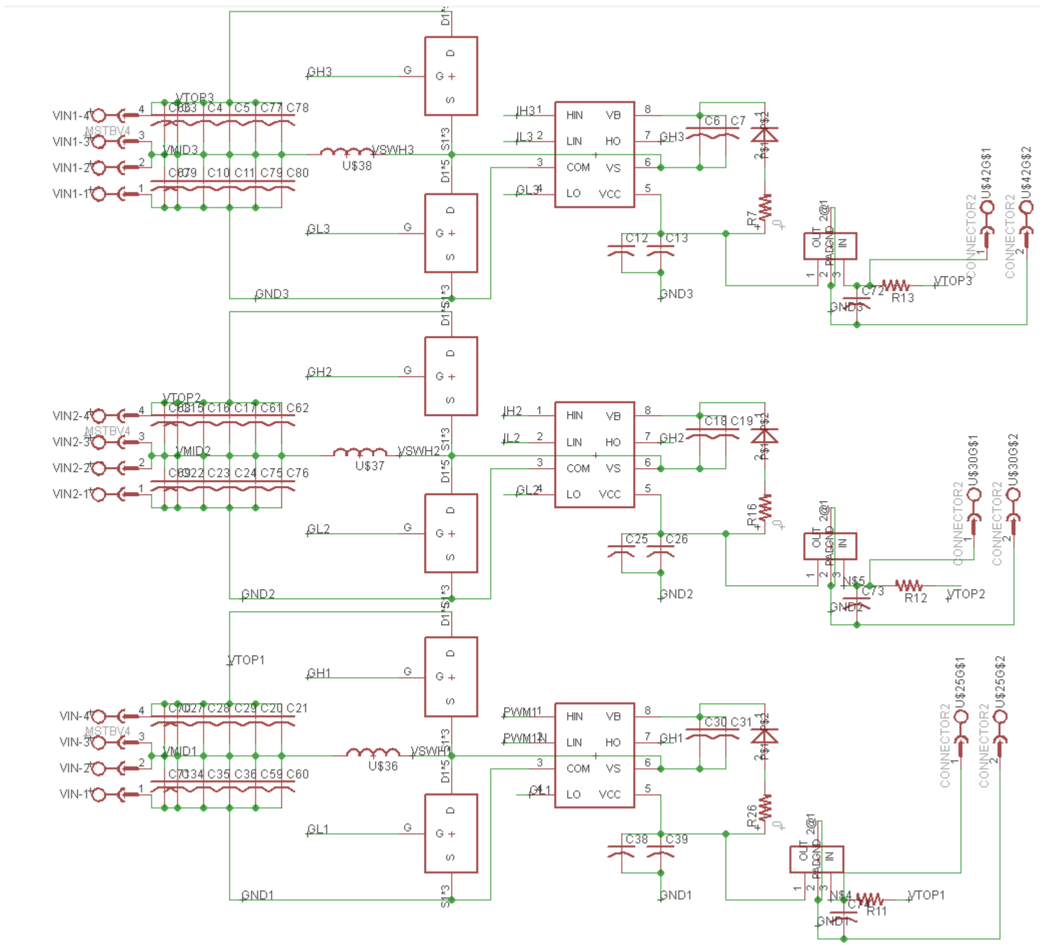


Figure B.2: Schematic drawing of the main power circuits.

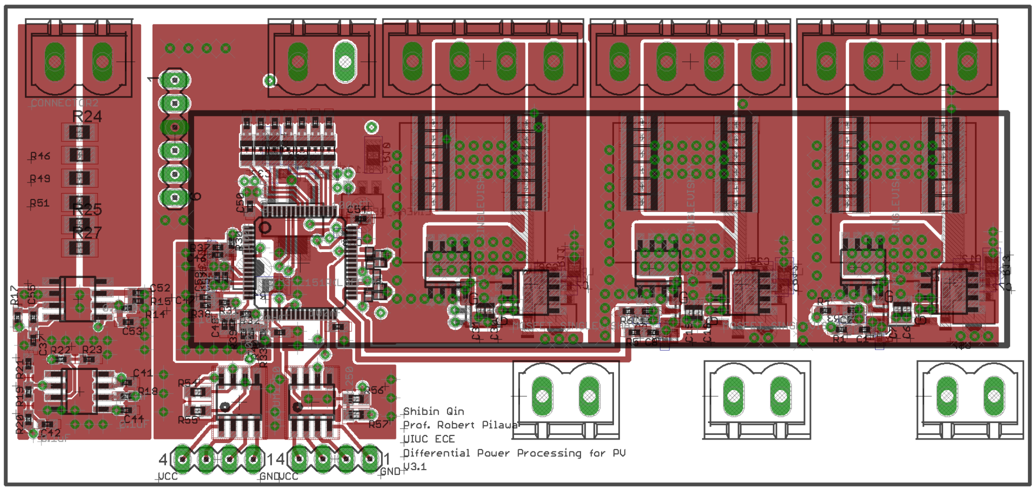


Figure B.3: PCB layout top view.

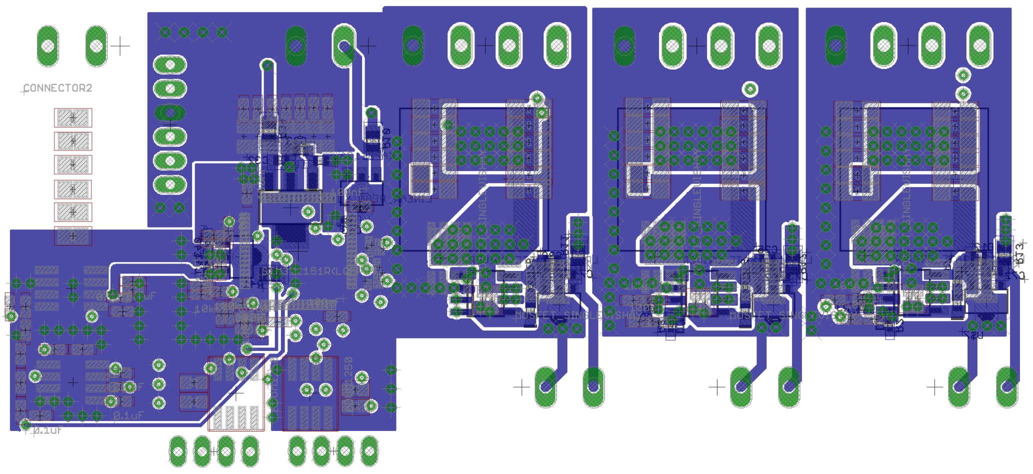


Figure B.4: PCB layout bottom view.

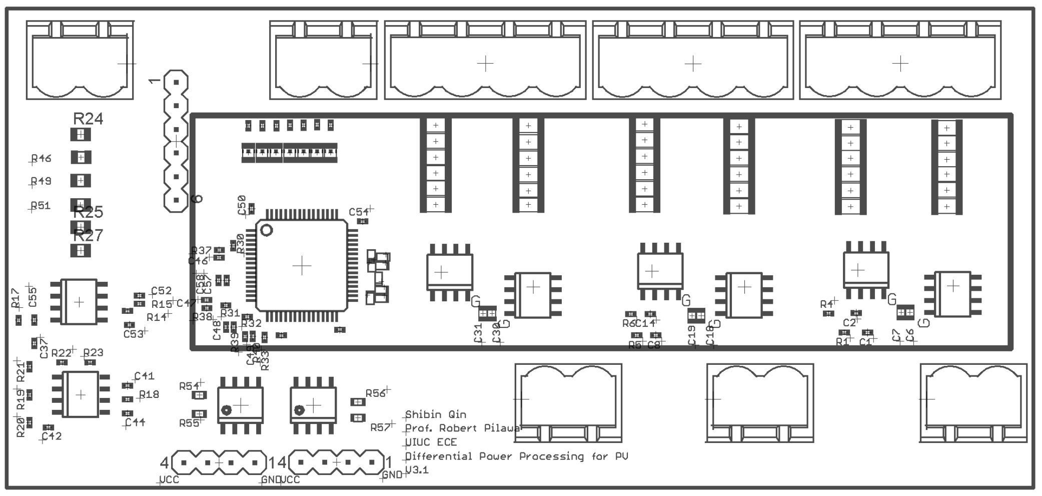


Figure B.5: Top silkscreen.

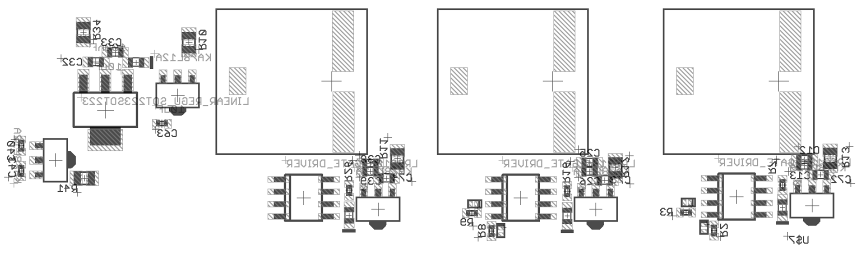


Figure B.6: Bottom silkscreen.

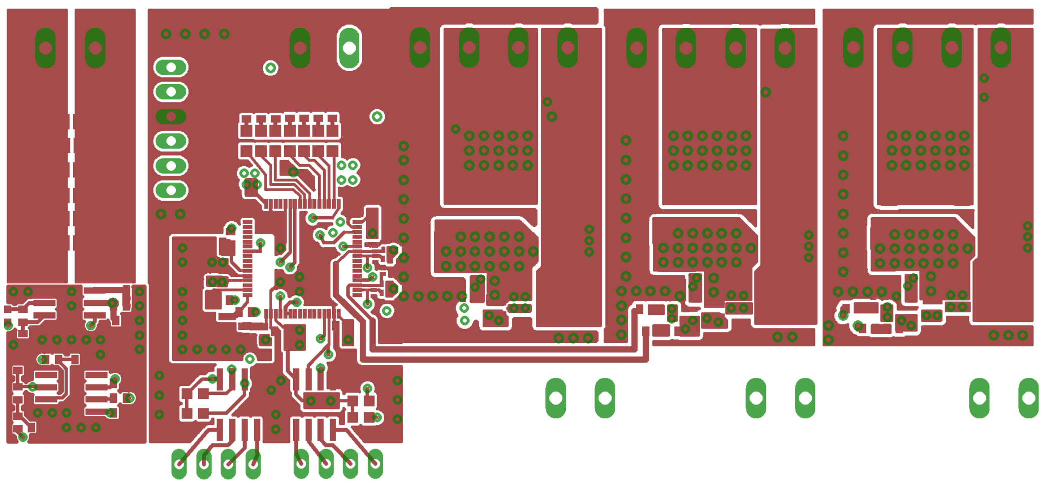


Figure B.7: Top-layer copper.

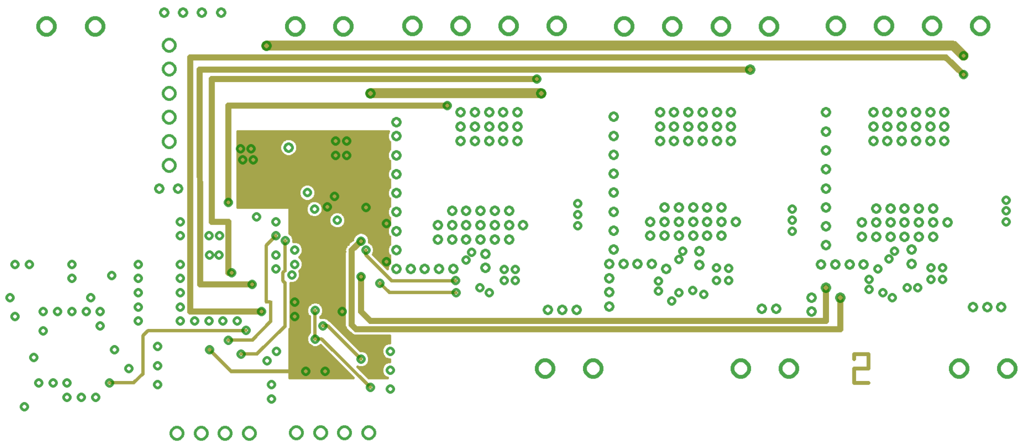


Figure B.8: PCB middle layer 1.

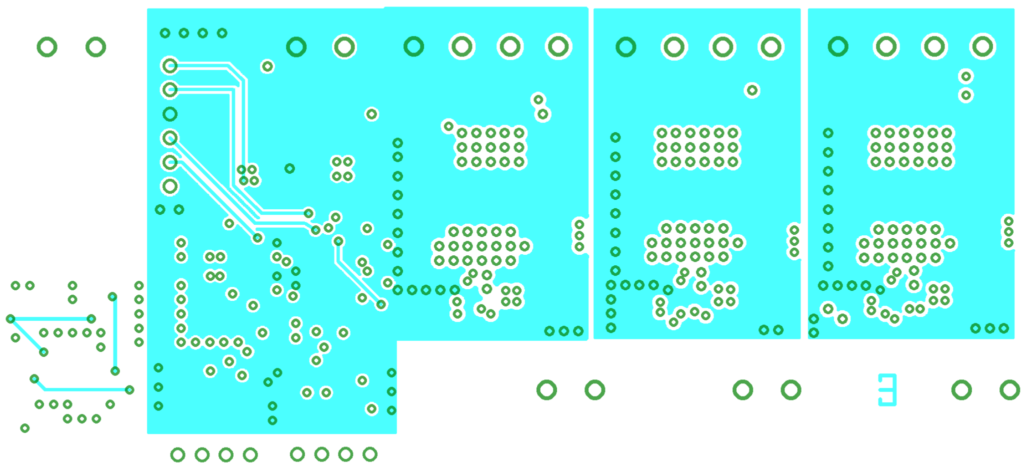


Figure B.9: PCB middle layer 2.

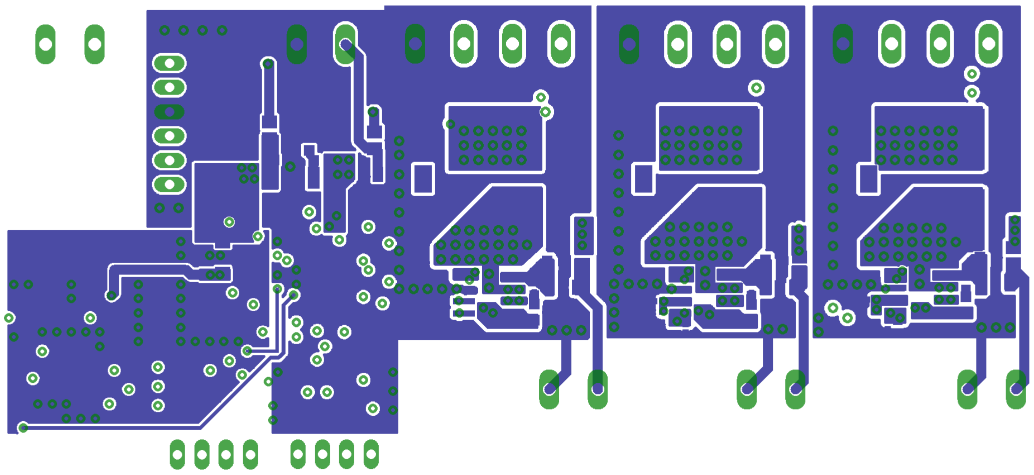


Figure B.10: Bottom-layer copper.

APPENDIX C

C CODE FOR MICRO-CONTROLLER

In this appendix, the C code for micro-controllers that implement the control algorithm is included.

C.1 Centralized Implementation

This part of the C code is used to configured DPP converters as a pure actuator. No control algorithm is embedded in these code. A bench computer runs the distributed control in a python code. See the Appendix D.1 for details of the python code. Here the control is developed as a distributed algorithm but run in a centralized way to facilitate debugging and data acquisition. For the fully distributed implantation in micro-controller C code, see Appendix C.2.

```
/********************************
 *
 * File: main.c
 *
 * Author:
 *  Shibin Qin
 *
 *
 *
 **********************************/
/**
 * @file Main file. Make the converter a pure actuator
 * @version 1.0
 * @date 2013/05/15
 */



```

```
#include "main.h"

// All initialization. etc. are omitted here to reduce document
   lenght. Full the full version, please contact sqin3@illinois.
   edu
......



int main(void)
{
  GPIO_InitTypeDef GPIO_InitStructure;

  /*right now SYSCLK is using HSI (8MHz), can be made up to 36
   MHz through PLL.
    Setting SYSCLK is important as all the other peripheral
   clocks are dependent on this one. */

  RCC_Configuration();
  GPIO_Configuration();
  NVIC_Configuration();
  I2C_Initialize(I2C1, I2C_Addr);
  I2C_Initialize(I2C2, I2C_Addr);
  Delay_Initialize();
  TIM3_PWM_Initialize();
  GPIO_SetBits(GPIOC,GPIO_Pin_4); //converter 1 enable
  //GPIO_ResetBits(GPIOC,GPIO_Pin_4); //converter 1 disable
  ADC_Initialize();

  I2C_Slave_ReadWrite(I2C1, Interrupt);
  I2C_Slave_ReadWrite(I2C2, Interrupt); //enable both I2C
   channel

  //Processor Idle Loop
 while(1)
  {
      if (LED_blink==0)
      {
        GPIO_SetBits(GPIOA,GPIO_Pin_9);
        LED_blink=1;
      }
      if (LED_blink==1)
      {
        GPIO_ResetBits(GPIOA, GPIO_Pin_9);
        LED_blink=0;
      }


      #ifdef controller1
      while (Buffer_Rx2[0]!=0)
        {
        Delay(150);
        mode1=Buffer_Rx2[0];
        tempdata1=Buffer_Rx2[1];
        tempdata2=Buffer_Rx2[2];
        datastring1=tempdata1*256+tempdata2;
        mode2=Buffer_Rx2[3];
        tempdata1=Buffer_Rx2[3];
        tempdata2=Buffer_Rx2[4];
        datastring2=tempdata1*256+tempdata2;
        for (k=0;k<=5;k++)
          Buffer_Rx2[k]=0;

      #else
      while (Buffer_Rx1[0]!=0)
        {
        Delay(150);
        mode1=Buffer_Rx1[0];
        tempdata1=Buffer_Rx1[1];
        tempdata2=Buffer_Rx1[2];
        datastring1=tempdata1*256+tempdata2;
        mode2=Buffer_Rx1[3];
        tempdata1=Buffer_Rx1[3];
        tempdata2=Buffer_Rx1[4];
        datastring2=tempdata1*256+tempdata2;
        for (k=0;k<=5;k++)
          Buffer_Rx1[k]=0;
      #endif
     if ((mode1!=mode2))//||(datastring1!=datastring2))
        {
          GPIO_SetBits(GPIOA, GPIO_Pin_8); //LED1 indicate a
   communication fault.
          }

     else
```

```
    {
    /* prevent extreme duty ratio*/
    if (datastring1 >1950)
        datastring1 =1950;
    if (datastring1 <1650)
        datastring1 =1650;

    if (mode1==1)   // PWM1 duty ratio change
      {
      PWM_Duty_Dither(PWM1, datastring1);
      //GPIO_SetBits(GPIOA, GPIO_Pin_9);
      }
    else if (mode1==2) // PWM2 duty ratio change
      {
      PWM_Duty_Dither(PWM2, datastring1);
      //GPIO_SetBits(GPIOA, GPIO_Pin_10);
      }
    else if (mode1==3)   // PWM3 duty ratio change
      {
      PWM_Duty_Dither(PWM3, datastring1);
      //GPIO_SetBits(GPIOA, GPIO_Pin_11);
      }
    else if (mode1==4)   //measure voltages
      {
      //GPIO_SetBits(GPIOA, GPIO_Pin_12);
      V1_measure=ADC_measure_single(ADC_Channel_11,100);
      V2_measure=ADC_measure_dual(ADC_Channel_11,
  ADC_Channel_13,100);
      V3_measure=ADC_measure_dual(ADC_Channel_13,
  ADC_Channel_2,100);
      #ifdef controller1
      Buffer_Tx2[0]=(uint8_t)(V1_measure>>8);
      Buffer_Tx2[1]=(uint8_t)(V1_measure);
      Buffer_Tx2[2]=(uint8_t)(V2_measure>>8);
      Buffer_Tx2[3]=(uint8_t)(V2_measure);
      Buffer_Tx2[4]=(uint8_t)(V3_measure>>8);
      Buffer_Tx2[5]=(uint8_t)(V3_measure);
      #else
      Buffer_Tx1[0]=(uint8_t)(V1_measure>>8);
      Buffer_Tx1[1]=(uint8_t)(V1_measure);
      Buffer_Tx1[2]=(uint8_t)(V2_measure>>8);
      Buffer_Tx1[3]=(uint8_t)(V2_measure);
      Buffer_Tx1[4]=(uint8_t)(V3_measure>>8);
```

```
          Buffer_Tx1[5]=(uint8_t)(V3_measure);
          #endif
          }
        else if (mode1==5)      //ping controller
          for(i=0;i<=5;i++)
              #ifdef controller1
              Buffer_Tx2[i]=I2C_Addr>>1;
              #else
              Buffer_Tx1[i]=I2C_Addr>>1;


        #endif
        }
      }
    }

} //End Main


/* End of file, must be followed by a blank line */
```

appendix/centralized/main.c

C.2 Distributed Implementation

This part of the C code is a fully distributed implementation of the distributed control algorithm.

```
/****************************
 *
 * File: main.c
 *
 * Author:
 *   Shibin Qin
 *
 *
 *
 ************************/
/**
 * @file Main file. Make the converter a pure actuator
 * @version 1.0
 * @date 2013/05/15
 */


#include "main.h"

// All initialization. etc. are omitted here to reduce document
    lenght. Full the full version, please contact sqin3@illinois.
    edu
//......


int main(void)
{
  GPIO_InitTypeDef GPIO_InitStructure;
  /*!< At this stage the microcontroller clock setting is
    already configured,
       this is done through SystemInit() function which is
    called from startup
       file (startup_stm32f10x_xx.s) before to branch to
    application main.
       To reconfigure the default setting of SystemInit()
    function, refer to
       system_stm32f10x.c file
     */
  /*right now SYSCLK is using HSI (8MHz), can be made up to 36
    MHz through PLL.
     Setting SYSCLK is important as all the other peripheral
    clocks are dependent on this one. */

  RCC_Configuration();
  GPIO_Configuration();
  NVIC_Configuration();
  I2C_Initialize(I2C1, Own_I2C_Addr);
  I2C_Initialize(I2C2, Own_I2C_Addr);
  Delay_Initialize();
  TIM3_PWM_Initialize();
  GPIO_SetBits(GPIOC,GPIO_Pin_4); //converter 1 enable
  //GPIO_ResetBits(GPIOC,GPIO_Pin_4); //converter 1 disable
  ADC_Initialize();

  I2C_Slave_ReadWrite(I2C1, Interrupt);
  I2C_Slave_ReadWrite(I2C2, Interrupt); //enable both I2C
    channel
  Delay(10000);


  /*initialize D and Z*/
  for (i=0;i<3;i++)
    for (j=0;j<6;j++)
    {
      D[i][j]=1800;
      Z[i][j]=0;
    }
  /*if want to start with specific D, change it here*/
  //D[0][0]=1730;
  //D[1][1]=1870;

  //Processor Idle Loop
 while(1)
  {
    GPIO_SetBits(GPIOA,GPIO_Pin_8); //LED1 indicates a new
   iteration starts
    /***************  synchronization
   *****************************/
    #ifdef controller1
    Buffer_Rx1[0]=0;
    Buffer_Rx1[1]=0;
    Buffer_Rx1[2]=0;
    Buffer_Tx1[0]='s';
    Buffer_Tx1[1]='y';
    Buffer_Tx1[2]='n';
    /*controller1 needs to read ack and send sync signal at the
   begining of each iterations*/
    i=0;
    while((Buffer_Rx1[0]!='a')||(Buffer_Rx1[1]!='c')||(
   Buffer_Rx1[2]!='k'))
    {
      while(I2C_Master_Read(I2C1, Buffer_Rx1, 3, Interrupt,
   Other_I2C_Addr)!=1); //read ack signal
    }
    while(I2C_Master_Write(I2C1, Buffer_Tx1, 3, Interrupt,
   Other_I2C_Addr)!=1);   //send syn signal


    #else   //in controller2
    Buffer_Rx1[0]=0;
    Buffer_Rx1[1]=0;
    Buffer_Rx1[2]=0;
    Buffer_Tx1[0]='a';
    Buffer_Tx1[1]='c';
    Buffer_Tx1[2]='k';

    /*controller2 needs to ack to controller1 and wait for sync
   signal at the begining of each iterations*/
    /*just wait for syn signal, interrupt will handle the
   communication in controller2*/
    while((Buffer_Rx1[0]!='s')||(Buffer_Rx1[1]!='y')||(
   Buffer_Rx1[2]!='n'));
    #endif

    GPIO_SetBits(GPIOA,GPIO_Pin_9); //LED2 indicates that
   perturb starts
    /******************* perturb D1 *******************/
    #ifdef controller1   //for controller1
    //Perturb D1 positively
    PWM_Duty_Dither(PWM1,D[0][0]+perturb_size);
    Delay(syn_delay);
    Vp[0][0]=ADC_measure_single(ADC_Channel_11,100);
    Vp[1][0]=ADC_measure_dual(ADC_Channel_11, ADC_Channel_13
   ,100);
    Vp[2][0]=ADC_measure_dual(ADC_Channel_13, ADC_Channel_2,100)
   ;
    Delay(syn_delay);
    //Perturb D1 negatively
    PWM_Duty_Dither(PWM1,D[0][0]-perturb_size);
    Delay(syn_delay);
    Vn[0][0]=ADC_measure_single(ADC_Channel_11,100);
    Vn[1][0]=ADC_measure_dual(ADC_Channel_11, ADC_Channel_13
   ,100);
    Vn[2][0]=ADC_measure_dual(ADC_Channel_13, ADC_Channel_2,100)
   ;
    Delay(syn_delay);
    //undo D1 perturb
    PWM_Duty_Dither(PWM1,D[0][0]);

    #else  //for controller2
    //pseudo perturb D4 for exact synchronization between two
   controller
    PWM_Duty_Dither(PWM1,D[0][3]);
```

```
    Delay(syn_delay);
    Vp[0][0]=ADC_measure_single(ADC_Channel_11,100);
    Vp[1][0]=ADC_measure_dual(ADC_Channel_11, ADC_Channel_13
    ,100);
    Vp[2][0]=ADC_measure_dual(ADC_Channel_13, ADC_Channel_2,100)
    ;
    Delay(syn_delay);
    //pseudo perturb D4 for exact synchronization between two
    controller
    PWM_Duty_Dither(PWM1,D[0][3]);
    Delay(syn_delay);
    Vn[0][0]=ADC_measure_single(ADC_Channel_11,100);
    Vn[1][0]=ADC_measure_dual(ADC_Channel_11, ADC_Channel_13
    ,100);
    Vn[2][0]=ADC_measure_dual(ADC_Channel_13, ADC_Channel_2,100)
    ;
    Delay(syn_delay);
    //pseudo undo D4 perturb
    PWM_Duty_Dither(PWM1,D[0][3]);
    #endif

    /******************* perturb D2 *******************/
    #ifdef controller1    //for controller1
    //Perturb D2 positively
    PWM_Duty_Dither(PWM2,D[1][1]+perturb_size);
    Delay(syn_delay);
    Vp[0][1]=ADC_measure_single(ADC_Channel_11,100);
    Vp[1][1]=ADC_measure_dual(ADC_Channel_11, ADC_Channel_13
    ,100);
    Vp[2][1]=ADC_measure_dual(ADC_Channel_13, ADC_Channel_2,100)
    ;
    Delay(syn_delay);
    //Perturb D2 negatively
    PWM_Duty_Dither(PWM2,D[1][1]-perturb_size);
    Delay(syn_delay);
    Vn[0][1]=ADC_measure_single(ADC_Channel_11,100);
    Vn[1][1]=ADC_measure_dual(ADC_Channel_11, ADC_Channel_13
    ,100);
    Vn[2][1]=ADC_measure_dual(ADC_Channel_13, ADC_Channel_2,100)
    ;
    Delay(syn_delay);
    //undo D2 perturb
    PWM_Duty_Dither(PWM2,D[1][1]);

    #else   //for controller2
    //pseudo perturb D5 for exact synchronization between two controller
    PWM_Duty_Dither(PWM2,D[1][4]);
    Delay(syn_delay);
    Vp[0][1]=ADC_measure_single(ADC_Channel_11,100);
    Vp[1][1]=ADC_measure_dual(ADC_Channel_11, ADC_Channel_13,100);
    Vp[2][1]=ADC_measure_dual(ADC_Channel_13, ADC_Channel_2,100);
    Delay(syn_delay);
    //pseudo perturb D5 for exact synchronization between two controller
    PWM_Duty_Dither(PWM2,D[1][4]);
    Delay(syn_delay);
    Vn[0][1]=ADC_measure_single(ADC_Channel_11,100);
    Vn[1][1]=ADC_measure_dual(ADC_Channel_11, ADC_Channel_13,100);
    Vn[2][1]=ADC_measure_dual(ADC_Channel_13, ADC_Channel_2,100);
    Delay(syn_delay);
    //pseudo undo D5 perturb
    PWM_Duty_Dither(PWM2,D[1][4]);
    #endif

        /******************* perturb D3 ******************/
    #ifdef controller1    //for controller1
    //Perturb D3 positively
    PWM_Duty_Dither(PWM3,D[2][2]+perturb_size);
    Delay(syn_delay);
    Vp[0][2]=ADC_measure_single(ADC_Channel_11,100);
    Vp[1][2]=ADC_measure_dual(ADC_Channel_11, ADC_Channel_13,100);
    Vp[2][2]=ADC_measure_dual(ADC_Channel_13, ADC_Channel_2,100);
    Delay(syn_delay);
    //Perturb D3 negatively
    PWM_Duty_Dither(PWM3,D[2][2]-perturb_size);
    Delay(syn_delay);
    Vn[0][2]=ADC_measure_single(ADC_Channel_11,100);
    Vn[1][2]=ADC_measure_dual(ADC_Channel_11, ADC_Channel_13,100);
```

```
Vn[2][2]=ADC_measure_dual(ADC_Channel_13, ADC_Channel_2,100)
;
Delay(syn_delay);
//undo D3 perturb
PWM_Duty_Dither(PWM3,D[2][2]);

#else   //for controller2
//pseudo perturb D6 for exact synchronization between two
controller
PWM_Duty_Dither(PWM3,D[2][2]);
Delay(syn_delay);
Vp[0][2]=ADC_measure_single(ADC_Channel_11,100);
Vp[1][2]=ADC_measure_dual(ADC_Channel_11, ADC_Channel_13
,100);
Vp[2][2]=ADC_measure_dual(ADC_Channel_13, ADC_Channel_2,100)
;
Delay(syn_delay);
//pseudo perturb D6 for exact synchronization between two
controller
PWM_Duty_Dither(PWM3,D[2][2]);
Delay(syn_delay);
Vn[0][2]=ADC_measure_single(ADC_Channel_11,100);
Vn[1][2]=ADC_measure_dual(ADC_Channel_11, ADC_Channel_13
,100);
Vn[2][2]=ADC_measure_dual(ADC_Channel_13, ADC_Channel_2,100)
;
Delay(syn_delay);
//pseudo undo D6 perturb
PWM_Duty_Dither(PWM3,D[2][2]);
#endif

        /*******************  perturb D4
*******************/
#ifdef controller1    //for controller1
//pseudo perturb D1 for exact synchronization between two
controller
PWM_Duty_Dither(PWM1,D[0][0]);
Delay(syn_delay);
Vp[0][3]=ADC_measure_single(ADC_Channel_11,100);
Vp[1][3]=ADC_measure_dual(ADC_Channel_11, ADC_Channel_13
,100);
Vp[2][3]=ADC_measure_dual(ADC_Channel_13, ADC_Channel_2,100)
;
```

```
Delay(syn_delay);
//pseudo perturb D1 for exact synchronization between two
controller
PWM_Duty_Dither(PWM1,D[0][0]);
Delay(syn_delay);
Vn[0][3]=ADC_measure_single(ADC_Channel_11,100);
Vn[1][3]=ADC_measure_dual(ADC_Channel_11, ADC_Channel_13
,100);
Vn[2][3]=ADC_measure_dual(ADC_Channel_13, ADC_Channel_2,100)
;
Delay(syn_delay);
//pseudo undo D1 perturb
PWM_Duty_Dither(PWM1,D[0][0]);

#else   //for controller2
//Perturb D4 positively
PWM_Duty_Dither(PWM1,D[0][3]+perturb_size);
Delay(syn_delay);
Vp[0][3]=ADC_measure_single(ADC_Channel_11,100);
Vp[1][3]=ADC_measure_dual(ADC_Channel_11, ADC_Channel_13
,100);
Vp[2][3]=ADC_measure_dual(ADC_Channel_13, ADC_Channel_2,100)
;
Delay(syn_delay);
//Perturb D4 negatively
PWM_Duty_Dither(PWM1,D[0][3]-perturb_size);
Delay(syn_delay);
Vn[0][3]=ADC_measure_single(ADC_Channel_11,100);
Vn[1][3]=ADC_measure_dual(ADC_Channel_11, ADC_Channel_13
,100);
Vn[2][3]=ADC_measure_dual(ADC_Channel_13, ADC_Channel_2,100)
;
Delay(syn_delay);
//undo D4 perturb
PWM_Duty_Dither(PWM1,D[0][3]);
#endif


        /*******************  perturb D5
*******************/
#ifdef controller1    //for controller1
//pseudo perturb D2 for exact synchronization between two
controller
```

```
    PWM_Duty_Dither(PWM2,D[1][1]);
    Delay(syn_delay);
    Vp[0][4]=ADC_measure_single(ADC_Channel_11,100);
    Vp[1][4]=ADC_measure_dual(ADC_Channel_11, ADC_Channel_13
   ,100);
    Vp[2][4]=ADC_measure_dual(ADC_Channel_13, ADC_Channel_2,100)
   ;
    Delay(syn_delay);
    //pseudo perturb D2 for exact synchronization between two
   controller
    PWM_Duty_Dither(PWM2,D[1][1]);
    Delay(syn_delay);
    Vn[0][4]=ADC_measure_single(ADC_Channel_11,100);
    Vn[1][4]=ADC_measure_dual(ADC_Channel_11, ADC_Channel_13
   ,100);
    Vn[2][4]=ADC_measure_dual(ADC_Channel_13, ADC_Channel_2,100)
   ;
    Delay(syn_delay);
    //pseudo undo D2 perturb
    PWM_Duty_Dither(PWM2,D[1][1]);

    #else   //for controller2
    //Perturb D5 positively
    PWM_Duty_Dither(PWM2,D[1][4]+perturb_size);
    Delay(syn_delay);
    Vp[0][4]=ADC_measure_single(ADC_Channel_11,100);
    Vp[1][4]=ADC_measure_dual(ADC_Channel_11, ADC_Channel_13
   ,100);
    Vp[2][4]=ADC_measure_dual(ADC_Channel_13, ADC_Channel_2,100)
   ;
    Delay(syn_delay);
    //Perturb D5 negatively
    PWM_Duty_Dither(PWM2,D[1][4]-perturb_size);
    Delay(syn_delay);
    Vn[0][4]=ADC_measure_single(ADC_Channel_11,100);
    Vn[1][4]=ADC_measure_dual(ADC_Channel_11, ADC_Channel_13
   ,100);
    Vn[2][4]=ADC_measure_dual(ADC_Channel_13, ADC_Channel_2,100)
   ;
    Delay(syn_delay);
    //undo D5 perturb
    PWM_Duty_Dither(PWM2,D[1][4]);
    #endif


    GPIO_SetBits(GPIOA,GPIO_Pin_10); //LED3 indicates exchange of information starts
    /******************* exchange information and calculate states *******************/


    // Communication codes are omitted here to reduce document lenght. Full the full version, please contact sqin3@illinois.edu
    //......
  }

} //End Main


/* End of file, must be followed by a blank line */
```

appendix/distributed/main.c

APPENDIX D

PYTHON CODE FOR DATA COLLECTION

The following Python scripts are to execute on a bench computer for control and data collection in automatic tests.

D.1 Centralized Implementation

This part of Python code works with the micro-controller C code in Appendix C.1

```
 #!/bin/env python

#==========================
# IMPORTS
#==========================
import sys
import time
import datetime
#from aardvark32.aardvark_py import *
from aardvark64.aardvark_py import *
from array import array
from pilawa_instruments import *
import os
import glob
import numpy as np
import matplotlib.pyplot as plt

#==========================
# CONSTANTS
#==========================
#debug=True

I2C_BITRATE =  400
PORT  = 0
```

```
ADDR1=1
ADDR2=2
PACKET_LENGTH=6
CLEAR_BUFFER_MESSAGE=99
MAX_RESENDS=3 #number of times to resend an i2c message if the
    count read is not correct
debug=False
NUMREADS = 20 # number of reads for the ADC on each sampling
    interval
DUTY_MIN = 1650
DUTY_MAX = 1950
insolation=[5,5,4,4,2.5,2.5]


READ_DELAY=0.2 #minimum seems to be 0.05 to prevent hangups
V_DIVIDER=(130.0+10.0)/10.0

VREF=3.3 #microcontroller vref value
ADC_MAX=4096.0

DUTY_COMMAND_PWM1=1
DUTY_COMMAND_PWM2=2
DUTY_COMMAND_PWM3=3
VOLTAGE_COMMAND=4
PING_COMMAND=5

command_string=("NO_COMMAND","DUTY_COMMAND_PWM1","
    DUTY_COMMAND_PWM2","DUTY_COMMAND_PWM3","VOLTAGE_COMMAND","
    PING_COMMAND")



MPPT_STEP_SIZE =5
CENTRALIZED_ITERATIONS=500
DISTRIBUTED_ITERATIONS=500

STARTUP_CURRENT=3.5
#sweep operation constants
STARTCURRENT=2
STOPCURRENT=5.0 #MUST HAVE THE .0 AT THE END!
NUMSTEPS=40.0

STARTCURRENT_VP=2.0
```

```
STOPCURRENT_VP=4.0
NUMSTEPS_VP=20.0

STARTCURRENT_MPPT=3.3
STOPCURRENT_MPPT=3.4
NUMSTEPS_MPPT=1.0

D_list=[1800,1800,1800,1800,1800]
#D_list=[1680,1920,1680,1920,1680]

#==============================
# CLASSES
#==============================

class controller:
    def __init__(self, handle, addr=1, duty=128, numreads=25,
    debug=False):
        self.handle=handle
        self.addr=addr
        self.numreads=numreads
        self.debug=debug
        self.direction=[1,1,1]
        self.v=[0, 0, 0]
        self.d=[1800, 1800, 1800]


    def sendMessage(self, command, parameter):
        command_byte=command
        transmit_byte_1=parameter>>8 #higher order bits
        transmit_byte_2=parameter & 0xFF
        send_array=array('B',[command_byte, transmit_byte_1,
    transmit_byte_2,command_byte, transmit_byte_1,
    transmit_byte_2])
        count = aa_i2c_write(self.handle, self.addr,
    AA_I2C_NO_FLAGS, send_array)
        if ( count != (len(send_array)) ):
            print "error sending, addr: %s, command %s, receive
    count: %d" % (self.addr,command_string[command_byte],count)
            return 0

        time.sleep(READ_DELAY)

        if (command==4):
            (count, data_in) = aa_i2c_read(self.handle, self.addr, AA_I2C_NO_FLAGS,PACKET_LENGTH)
            if ( count != (PACKET_LENGTH) ):
                print "error receiving, addr: %s, command %s, receive count: %d" % (self.addr,command_string[command_byte],count)
                return 0
            else:
                self.v[0]=(data_in[0]*256+data_in[1])#*3.3*V_DIVIDER/ADC_MAX
                self.v[1]=(data_in[2]*256+data_in[3])#*3.3*V_DIVIDER/ADC_MAX
                self.v[2]=(data_in[4]*256+data_in[5])#*3.3*V_DIVIDER/ADC_MAX
                return (1,data_in[0],data_in[1],data_in[2],data_in[3],data_in[4],data_in[5])
        if (command==5):
            (count, data_in) = aa_i2c_read(self.handle, self.addr, AA_I2C_NO_FLAGS,PACKET_LENGTH)
            if ( count != (PACKET_LENGTH) ):
                print "error receiving, addr: %s, command %s, receive count: %d" % (self.addr,command_string[command_byte],count)
                return 0
            elif (data_in[0]!=self.addr):
                return 0
        return 1


    def writeDuty(self, duty,PWM_channel):
        if (PWM_channel==1):
            return_value= self.sendMessage(DUTY_COMMAND_PWM1,duty)
            if (return_value == 1):
                self.d[0]=duty # update internal duty if message was successful
                return 1          #to indicate that the command was executed properly
            else:
                print "writeDuty error"
                return 0
        elif (PWM_channel==2):
            return_value= self.sendMessage(DUTY_COMMAND_PWM2,
duty)
            if (return_value == 1):
                self.d[1]=duty # update internal duty if message
 was successful
                return 1         #to indicate that the command
was executed properly
            else:
                print "writeDuty error"
                return 0
        elif (PWM_channel==3):
            return_value= self.sendMessage(DUTY_COMMAND_PWM3,
duty)
            if (return_value == 1):
                self.d[2]=duty # update internal duty if message
 was successful
                return 1         #to indicate that the command
was executed properly
            else:
                print "writeDuty error"
                return 0


    def ping(self):
        return self.sendMessage(PING_COMMAND, 0)


    def voltage_update(self):
        return self.sendMessage(VOLTAGE_COMMAND, 0)




#==============================
# FUNCTIONS
#==============================

def round_integer(number):
    return int(number+0.5)

def MPPTPing(converter_list):
    converter_alive=0
    while converter_alive < len(converter_list):
        converter_alive=0
        for x in converter_list:
            time.sleep(0.2)
            print "Establishing communication to addr %s......"
    %x.addr
            time.sleep(0.2)
            return_value = x.ping()
            if return_value == 1:
                converter_alive = converter_alive + 1
                print "addr %s responds" %x.addr
    print "all controllers have responded! :-)"

##############################
def IVSweep(startcurrent, stopcurrent,numstep,f,f_individual):
    print "doing IV current sweep"
    sweep_current=startcurrent
    step_size=(stopcurrent-startcurrent)/numstep #will be
    negative if we stop lower than we start
    eload.setMode("CURR")
    eload.setSlew(2000000)
    eload.setValue(sweep_current)
    while (sweep_current < stopcurrent): #change here if want to
     go different direction
        eload.setValue(sweep_current)
        for x in meterList:
            x.waitForTrigger()
        time.sleep(0.1)
        gpib.trigger_devices(meter_addrList)
        eload_voltage=str(meter1.readData())
        eload_current=str(float(meter2.readData())*1000.0)
        print "eload_current: %s" % eload_current
        print "eload_voltage: %s" % eload_current

        timestamp = datetime.datetime.now()
        f.write("%s%s%s%s%s\n" % ( timestamp.strftime("%H%M%S%f"
    ), delimiter, eload_current, delimiter, eload_voltage))
        sweep_current=sweep_current+step_size
    print ("IV current sweep done")
    eload.setMode("CURR")
    eload.setValue(STARTUP_CURRENT)

##############################
def IVSweep_VP(startcurrent, stopcurrent,numstep,f):
    #MPPTPing(controller_list)
    for controller in controller_list:
        controller.writeDuty(1800,1)
        controller.writeDuty(1800,2)
        controller.writeDuty(1800,3)
    print "doing IV current sweep (virtual parallel)"
    sweep_current=startcurrent
    step_size=(stopcurrent-startcurrent)/numstep #will be negative if we stop lower than we start
    eload.setMode("CURR")
    eload.setSlew(2000000)
    eload.setValue(sweep_current)
    while (sweep_current < stopcurrent): #change here if wanting to go different direction
        eload.setValue(sweep_current)
        for x in meterList:
            x.waitForTrigger()
        time.sleep(0.1)
        gpib.trigger_devices(meter_addrList)
        eload_voltage=str(meter1.readData())
        eload_current=str(float(meter2.readData())*1000.0)
        print "eload_current: %s" % eload_current
        print "eload_voltage: %s" % eload_voltage

        timestamp = datetime.datetime.now()
        f.write("%s%s%s%s%s\n" % ( timestamp.strftime("%H%M%S%f"), delimiter, eload_current, delimiter, eload_voltage))
        sweep_current=sweep_current+step_size
    print ("IV current sweep done (virtual parallel)")
    eload.setMode("CURR")
    eload.setValue(STARTUP_CURRENT)

##################################
def centralizedMPPT(startcurrent, stopcurrent,numstep, f):
    step_size=(stopcurrent-startcurrent)/numstep #will be negative if we stop lower than we start
    MPPTPing(controller_list)
    print "doing IV current sweep (centralized MPPT)"
    eload.setMode("CURR")
    eload.setSlew(2000000)
    sweep_current=startcurrent
    while (sweep_current < stopcurrent): #change here is wanting to go different direction
        eload.setValue(sweep_current)
        time.sleep(0.1)
        filename="%s_sweep3_duty_I%s.dat" % (root_filename, int(sweep_current*100))
        f_duty=open(foldername + "/" + filename,"w")
        for controller in controller_list:
            controller.writeDuty(1800,1)
            controller.writeDuty(1800,2)
            controller.writeDuty(1800,3)
        mppt_iterations=1
        while (mppt_iterations <= CENTRALIZED_ITERATIONS):
            print "No.%s" % mppt_iterations
            for controller in controller_list:
                for channel in range(1,4):
                    for x in meterList:
                        x.waitForTrigger()
                    gpib.trigger_devices(meter_addrList)
                    string_voltage_old=float(meter1.readData())
                    print "controller:%s  channel:%s  duty:%s  old_Vstring: %s  current direc:%s  " %(controller.addr, channel, controller.d[channel-1], string_voltage_old, controller.direction[channel-1])
                    controller.d[channel-1]=controller.d[channel-1]+controller.direction[channel-1]*MPPT_STEP_SIZE
                    controller.writeDuty(controller.d[channel-1],channel)
                    time.sleep(0.05)
                    for x in meterList:
                        x.waitForTrigger()
                    gpib.trigger_devices(meter_addrList)
                    string_voltage_new=float(meter1.readData())
                    if (string_voltage_new<=string_voltage_old):
                        controller.direction[channel-1]=-1*controller.direction[channel-1]
                    print "controller:%s  channel:%s  duty:%s  new_Vstring: %s  new direc:%s  " %(controller.addr, channel, controller.d[channel-1], string_voltage_new, controller.direction[channel-1])
            f_duty.write("%s%s%s%s%s%s%s%s%s%s%s\n" % (mppt_iterations, delimiter, controller1.d[0], delimiter, controller1.d[1], delimiter, controller1.d[2], delimiter, controller2.d[0], delimiter, controller2.d[1]))
            mppt_iterations=mppt_iterations+1

        for x in meterList:
            x.waitForTrigger()
        gpib.trigger_devices(meter_addrList)
        eload_voltage=str(meter1.readData())
        eload_current=str(float(meter2.readData())*1000.0)
        print "eload_current: %s" % eload_current
        print "eload_voltage: %s" % eload_voltage
        timestamp = datetime.datetime.now()
        f.write("%s%s%s%s%s\n" % ( timestamp.strftime("%H%M%S%f"
    ), delimiter, eload_current, delimiter, eload_voltage))
        sweep_current=sweep_current+step_size
        f_duty.close()

    print ("MPPT IV current sweep done")
    eload.setMode("CURR")
    eload.setValue(STARTUP_CURRENT)

##############################

def distributedMPPT(startcurrent, stopcurrent,numstep, f):
    step_size=(stopcurrent-startcurrent)/numstep #will be
    negative if we stop lower than we start
    MPPTPing(controller_list)
    print "doing IV current sweep (distributed MPPT)"
    eload.setMode("CURR")
    eload.setSlew(2000000)
    sweep_current=startcurrent
    while (sweep_current < stopcurrent): #change here is wanting
     to go different direction
        eload.setValue(sweep_current)
        time.sleep(0.1)
        filename="%s_duty_I%s.dat" % (root_filename, int(
    sweep_current*100))
        f_duty=open(foldername + "/" + filename,"w")
        filename="%s_duty_I%s_power.dat" % (root_filename, int(
    sweep_current*100))
        f_power=open(foldername + "/" + filename,"w")
        find_mpp(controller_list,D_list,f_duty,f_power,
    DISTRIBUTED_ITERATIONS)

        for x in meterList:
            x.waitForTrigger()
        gpib.trigger_devices(meter_addrList)
```

```python
        eload_voltage=str(meter1.readData())
        eload_current=str(float(meter2.readData())*1000.0)
        print "eload_current: %s" % eload_current
        print "eload_voltage: %s" % eload_voltage
        timestamp = datetime.datetime.now()
        f.write("%s%s%s%s%s\n" % ( timestamp.strftime("%H%M%S%f"), delimiter, eload_current, delimiter, eload_voltage))
        sweep_current=sweep_current+step_size
        f_duty.close()
    print "MPPT IV current sweep done"
    eload.setMode("CURR")
    eload.setValue(STARTUP_CURRENT)

############Distributed algorithm################
def find_mpp(controller_list,D_list, f_duty,f_power,num_iter):
    D1= np.array([[D_list[0]],[D_list[1]],[D_list[2]],[D_list[3]],[D_list[4]]])
    D2= np.array([[D_list[0]],[D_list[1]],[D_list[2]],[D_list[3]],[D_list[4]]])
    D3= np.array([[D_list[0]],[D_list[1]],[D_list[2]],[D_list[3]],[D_list[4]]])
    D4= np.array([[D_list[0]],[D_list[1]],[D_list[2]],[D_list[3]],[D_list[4]]])
    D5= np.array([[D_list[0]],[D_list[1]],[D_list[2]],[D_list[3]],[D_list[4]]])
    D= np.vstack([D1,D2,D3,D4,D5])
    # number of converters
    Nc = 5
    # graph laplacian representing exchange of info between DPPs
    L = np.array([[1,-1,0,0,0], [-1,2,-1,0,0], [0,-1,2,-1,0], [0,0,-1,2,-1], [0,0,0,-1,1]])
    Ltilde = np.kron(L,np.identity(Nc))
    z = np.zeros((Nc*Nc,1))
    for k in range(1,num_iter):
        (D,z,u) = update_controller_states(Ltilde,D,z)
        print "Iternation:%s" %k
        print "D:"
        print "D1:%s, D2:%s, D3:%s, D4:%s, D5:%s" %(D.item(0),D.item(6),D.item(12),D.item(18),D.item(24))
        D = np.array([[round_integer(D.item(0))],[round_integer(D.item(1))],[round_integer(D.item(2))],[round_integer(D.item(3))],[round_integer(D.item(4))],[round_integer(D.item(5))],[round_integer(D.item(6))],[round_integer(D.item(7))],[
round_integer(D.item(8))],[round_integer(D.item(9))],[
round_integer(D.item(10))],[round_integer(D.item(11))],[
round_integer(D.item(12))],[round_integer(D.item(13))],[
round_integer(D.item(14))],[round_integer(D.item(15))],[
round_integer(D.item(16))],[round_integer(D.item(17))],[
round_integer(D.item(18))],[round_integer(D.item(19))],[
round_integer(D.item(20))],[round_integer(D.item(21))],[
round_integer(D.item(22))],[round_integer(D.item(23))],[
round_integer(D.item(24))]])
        print "Quantized D:"
        print "D1:%s, D2:%s, D3:%s, D4:%s, D5:%s" %(D.item(0),D.
item(6),D.item(12),D.item(18),D.item(24))
        controller1.writeDuty(D.item(0),1)
        controller1.writeDuty(D.item(6),2)
        controller1.writeDuty(D.item(12),3)
        controller2.writeDuty(D.item(18),1)
        controller2.writeDuty(D.item(24),2)

        f_duty.write("%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s
%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%
s%s%s%s%s%s%s%s%s\n" % ( k, delimiter, D.item(0),delimiter,
 D.item(6),delimiter, D.item(12),delimiter, D.item(18),
delimiter, D.item(24),delimiter,u.item(0),delimiter,u.item(1)
,delimiter,u.item(2),delimiter,u.item(3),delimiter,u.item(4),
delimiter,u.item(5),delimiter,u.item(6),delimiter,u.item(7),
delimiter,u.item(8),delimiter,u.item(9),delimiter,u.item(10),
delimiter,u.item(11),delimiter,u.item(12),delimiter,u.item
(13),delimiter,u.item(14),delimiter,u.item(15),delimiter,u.
item(16),delimiter,u.item(17),delimiter,u.item(18),delimiter,
u.item(19),delimiter,u.item(20),delimiter,u.item(21),
delimiter,u.item(22),delimiter,u.item(23),delimiter,u.item
(24)))
        #measure power after steady state (for tracking
efficiency)
        if (k>0.8*DISTRIBUTED_ITERATIONS):
            submodule_power_sum=0
            for i in range(0,20):
                for x in meterList:
                    x.waitForTrigger()
                gpib.trigger_devices(meter_addrList)
                submodule_voltage=float(meter1.readData())
                submodule_current=float(meter2.readData())
*1000.0
```

```
                submodule_power= submodule_current*submodule_voltage
                submodule_power_sum=submodule_power_sum+submodule_power
            submodule_power_sum=str(submodule_power_sum/20.0)
            timestamp = datetime.datetime.now()
            f.write("%s%s%s\n" % ( k, delimiter, submodule_power_sum))

def get_value_phi_functions():
    # get all three voltages
    return_number=0
    while (return_number!=1):
        [return_number,i,j,k,l,m,n]=controller1.voltage_update()
    return_number=0
    while (return_number!=1):
        [return_number,i,j,k,l,m,n]=controller2.voltage_update()
    return (controller1.v[0] + 0.5*controller1.v[1],0.5*controller1.v[1] + 0.5*controller1.v[2],0.5*controller1.v[2] + 0.5*controller2.v[0],0.5*controller2.v[0] + 0.5*controller2.v[1], 0.5*controller2.v[1] + controller2.v[2])

def update_controller_states(Ltilde,D,z):
    Nc=5
    # resolution of duty ratio command (~1/1800)
    Dres = 5
    # tuning factor for algorithm
    gamma = 15.0
    delta = 0.1
    # amount of time in seconds to wait for steady state
    delay=0.002
    # perturb DPP 1 positivey
    controller1.writeDuty(D.item(0) + Dres,1)
    # wait for steady state?
    time.sleep(delay)
    # get value of phi functions for DPP1 positive perturb
    (phi11p,phi21p,phi31p,phi41p,phi51p) = get_value_phi_functions()
    #print "phi11p:%s, phi21p:%s, phi31p:%s, phi41p:%s, phi51p:%s" %(phi11p,phi21p,phi31p,phi41p,phi51p)
    # perturb DPP 1 negatively
    controller1.writeDuty(D.item(0) - Dres,1)
    # wait for steady state?
    time.sleep(delay)
    # get value of phi functions for DPP1 negative perturb
    (phi11n,phi21n,phi31n,phi41n,phi51n) =
    get_value_phi_functions()
    #print "phi11n:%s, phi21n:%s, phi31n:%s, phi41n:%s, phi51n:%
    s" %(phi11n,phi21n,phi31n,phi41n,phi51n)
    #undo perturb to DPP1
    controller1.writeDuty(D.item(0) ,1)

    # perturb DPP 2 positively
    controller1.writeDuty(D.item(6) + Dres,2)
    # wait for steady state?
    time.sleep(delay)
    # get value of phi functions for DPP2 positive perturb
    (phi12p,phi22p,phi32p,phi42p,phi52p) =
    get_value_phi_functions()
    #print "phi12p:%s, phi22p:%s, phi32p:%s, phi42p:%s, phi52p:%
    s" %(phi12p,phi22p,phi32p,phi42p,phi52p)
    # perturb DPP 2 negatively
    controller1.writeDuty(D.item(6) - Dres,2)
    # wait for steady state?
    time.sleep(delay)
    # get value of phi functions for DPP2 negative perturb
    (phi12n,phi22n,phi32n,phi42n,phi52n) =
    get_value_phi_functions()
    #print "phi12n:%s, phi22n:%s, phi32n:%s, phi42n:%s, phi52n:%
    s" %(phi12n,phi22n,phi32n,phi42n,phi52n)
    #undo perturb to DPP2
    controller1.writeDuty(D.item(6) ,2)

    # perturb DPP 3 positively
    controller1.writeDuty(D.item(12) + Dres,3)
    # wait for steady state?
    time.sleep(delay)
    # get value of phi functions for DPP3 positive perturb
    (phi13p,phi23p,phi33p,phi43p,phi53p) =
    get_value_phi_functions()
    #print "phi13p:%s, phi23p:%s, phi33p:%s, phi43p:%s, phi53p:%
    s" %(phi13p,phi23p,phi33p,phi43p,phi53p)
    # perturb DPP 3 negatively
    controller1.writeDuty(D.item(12) - Dres,3)
    # wait for steady state?
    time.sleep(delay)
    # get value of phi functions for DPP3 negative perturb
    (phi13n ,phi23n ,phi33n ,phi43n ,phi53n ) =
   get_value_phi_functions()
    #print "phi13n:%s, phi23n:%s, phi33n:%s, phi43n:%s, phi53n:%
   s" %(phi13n ,phi23n ,phi33n ,phi43n ,phi53n)
    #undo perturb to DPP3
    controller1.writeDuty(D.item(12) ,3)

    # perturb DPP 4 positively
    controller2.writeDuty(D.item(18) + Dres,1)
    # wait for steady state?
    time.sleep(delay)
    # get value of phi functions for DPP4 positive perturb
    (phi14p ,phi24p ,phi34p ,phi44p ,phi54p ) =
   get_value_phi_functions()
    #print "phi14p:%s, phi24p:%s, phi34p:%s, phi44p:%s, phi54p:%
   s" %(phi14p ,phi24p ,phi34p ,phi44p ,phi54p)
    # perturb DPP 4 negatively
    controller2.writeDuty(D.item(18) - Dres,1)
    # wait for steady state?
    time.sleep(delay)
    # get value of phi functions for DPP4 negative perturb
    (phi14n ,phi24n ,phi34n ,phi44n ,phi54n ) =
   get_value_phi_functions()
    #print "phi14n:%s, phi24n:%s, phi34n:%s, phi44n:%s, phi54n:%
   s" %(phi14n ,phi24n ,phi34n ,phi44n ,phi54n)
    #undo perturb to DPP4
    controller2.writeDuty(D.item(18), 1)

    # perturb DPP 5 positively
    controller2.writeDuty(D.item(24) + Dres,2)
    # wait for steady state?
    time.sleep(delay)
    # get value of phi functions for DPP5 positive perturb
    (phi15p ,phi25p ,phi35p ,phi45p ,phi55p ) =
   get_value_phi_functions()
    #print "phi15p:%s, phi25p:%s, phi35p:%s, phi45p:%s, phi55p:%
   s" %(phi15p ,phi25p ,phi35p ,phi45p ,phi55p)
    # perturb DPP 5 negatively
    controller2.writeDuty(D.item(24) - Dres,2)
    # wait for steady state?
    time.sleep(delay)
    # get value of phi functions for DPP5 negative perturb
    (phi15n, phi25n, phi35n, phi45n, phi55n) = get_value_phi_functions()
    #print "phi15n:%s, phi25n:%s, phi35n:%s, phi45n:%s, phi55n:%s" %(phi15n, phi25n, phi35n, phi45n, phi55n)
    #undo perturb to DPP5
    controller2.writeDuty(D.item(24) ,2)


    # compute input vector (~ partials of phi functions)
    u = np.array([[(phi11p-phi11n)/(2.0*Dres)],[(phi12p-phi12n)/(2.0*Dres)],[(phi13p-phi13n)/(2.0*Dres)],[(phi14p-phi14n)/(2.0*Dres)],[(phi15p-phi15n)/(2.0*Dres)],[(phi21p-phi21n)/(2.0*Dres)],[(phi22p-phi22n)/(2.0*Dres)],[(phi23p-phi23n)/(2.0*Dres)],[(phi24p-phi24n)/(2.0*Dres)],[(phi25p-phi25n)/(2.0*Dres)],[(phi31p-phi31n)/(2.0*Dres)],[(phi32p-phi32n)/(2.0*Dres)],[(phi33p-phi33n)/(2.0*Dres)],[(phi34p-phi34n)/(2.0*Dres)],[(phi35p-phi35n)/(2.0*Dres)],[(phi41p-phi41n)/(2.0*Dres)],[(phi42p-phi42n)/(2.0*Dres)],[(phi43p-phi43n)/(2.0*Dres)],[(phi44p-phi44n)/(2.0*Dres)],[(phi45p-phi45n)/(2.0*Dres)],[(phi51p-phi51n)/(2.0*Dres)],[(phi52p-phi52n)/(2.0*Dres)],[(phi53p-phi53n)/(2.0*Dres)],[(phi54p-phi54n)/(2.0*Dres)],[(phi55p-phi55n)/(2.0*Dres)]])
    #print "u:"
    #print u
    # compute part a of update
    a = np.dot((np.identity(Nc*Nc) - delta*Ltilde),D)
    # compute part b of update
    b = np.dot(delta*Ltilde,z)
    # compute part c of update
    c = gamma*delta*u
    # compute part d of upate
    d = np.dot(delta*Ltilde,D)

    # find update for D and z states
    dNext = a - b + c
    zNext = z + d
    return (dNext,zNext,u)

#==================================
# MAIN PROGRAM
#==================================

root_filename = sys.argv[1]
```

```

handle = aa_open(PORT)
if (handle <= 0):
    print "Unable to open Aardvark device on port %d" % PORT
    print "Error code = %d" % handle
    sys.exit()

# Ensure that the I2C subsystem is enabled, also do I2C
#aa_configure(handle,  AA_CONFIG_SPI_I2C)
aa_configure(handle,  AA_CONFIG_GPIO_I2C)

# Enable the I2C bus pullup resistors (2.2k resistors).
# This command is only effective on v2.0 hardware or greater.
# The pullup resistors on the v1.02 hardware are enabled by
    default.
aa_i2c_pullup(handle, AA_I2C_PULLUP_BOTH)

# Enable the Aardvark adapter's power supply.
# This command is only effective on v2.0 hardware or greater.
# The power pins on the v1.02 hardware are not enabled by
    default.
aa_target_power(handle, AA_TARGET_POWER_BOTH)

# Set the bitrate
bitrate = aa_i2c_bitrate(handle, I2C_BITRATE)
print "Bitrate set to %d kHz" % bitrate

delimiter=', '
t=time.strftime('%Y%m%d')
current_directory=os.curdir
print "Current directory: %s" % current_directory
foldername=current_directory + "/data/" + t
print foldername
if not os.path.exists(foldername):
    os.makedirs(foldername)

header_string=("time", "iload", "vload\n")
header_string_VP=("addr","time","Vin","Vout","duty","
    sweep_current\n")
header_string_mppt=("addr", "time", "vout", "vin", "vin_old", "
    duty", "direction", "current\n")

addrList = [ADDR1,ADDR2]
```

```

#duty is 0-360
controller1=controller(handle, ADDR1, 1800, NUMREADS, debug)
controller2=controller(handle, ADDR2, 1800, NUMREADS, debug)

controller_list=[controller1,controller2]


gpib  = prologix_serial(port="/dev/ttyUSB0", baud=115200, debug=
    False, timeout=5)
eload= prologix_6060b(prologix=gpib, addr=4, mode="VOLT", rang="
    20",  debug=False)
power_supply_5=prologix_6632a(prologix=gpib, addr=5,debug=False)
power_supply_6=prologix_6632a(prologix=gpib, addr=6,debug=False)
power_supply_7=prologix_6632a(prologix=gpib, addr=7,debug=False)
power_supply_8=prologix_6632a(prologix=gpib, addr=8,debug=False)
power_supply_9=prologix_6632a(prologix=gpib, addr=9,debug=False)
power_supply_10=prologix_6632a(prologix=gpib, addr=10,debug=
    False)
power_supply_11=prologix_6632a(prologix=gpib, addr=11,debug=
    False)
power_supply_12=prologix_6632a(prologix=gpib, addr=12,debug=
    False)


meter1 = prologix_34401a(prologix=gpib, addr=21, mode="VOLT",
    maxrange="25", nplc="1", debug=False)
meter2 = prologix_34401a(prologix=gpib, addr=22, mode="VOLT",
    maxrange="1",  nplc="1", debug=False)

meterList = [meter1,meter2]
meter_addrList=[21,22]
scaleList = [1, 1000]
for x in meterList:
    x.waitForTrigger()
gpib.trigger_devices(meter_addrList)
eload_voltage=float(meter1.readData())
eload_current=float(meter2.readData())*1000.0
print "eload_current: %s" % eload_current
print "eload_voltage: %s" % eload_voltage

#always make sure turn on power supply to controller first
power_supply_11.setVoltage(10)
```

```
power_supply_11.setCurrent(1)
power_supply_11.activate()
time.sleep(1)
power_supply_12.setVoltage(10)
power_supply_12.setCurrent(1)
power_supply_12.activate()
time.sleep(1)

eload.setMode("CURR")
eload.setValue(STARTUP_CURRENT)

power_supply_5.setVoltage(14)
power_supply_5.setCurrent(insolation[0])
power_supply_6.setVoltage(14)
power_supply_6.setCurrent(insolation[1])
power_supply_7.setVoltage(14)
power_supply_7.setCurrent(insolation[2])
power_supply_8.setVoltage(14)
power_supply_8.setCurrent(insolation[3])
power_supply_9.setVoltage(14)
power_supply_9.setCurrent(insolation[4])
power_supply_10.setVoltage(14)
power_supply_10.setCurrent(insolation[5])

power_supply_6.activate()
power_supply_5.activate()
power_supply_7.activate()
power_supply_8.activate()
power_supply_9.activate()
power_supply_10.activate()
time.sleep(1)


#sweep 1
filename_panel= "%s_data%s.dat" % (root_filename, "panel")
f_panel=open(foldername + "/" + filename_panel,"w")
filename= "%s_data%s.dat" % (root_filename, "sweep1")
f=open(foldername + "/" + filename,"w")
IVSweep(STARTCURRENT, STOPCURRENT,NUMSTEPS, f,f_panel)
f.close()
f_panel.close()


```

```
power_supply_10.deactivate()
power_supply_9.deactivate()
power_supply_8.deactivate()
power_supply_12.deactivate()
power_supply_7.deactivate()
power_supply_6.deactivate()
power_supply_5.deactivate()
power_supply_11.deactivate()


raw_input('Make sure converter is all connected and press enter'
    )

power_supply_11.deactivate()
power_supply_12.deactivate()
power_supply_6.activate()
power_supply_5.activate()
power_supply_7.activate()
power_supply_8.activate()
power_supply_9.activate()
power_supply_10.activate()
time.sleep(1)

#time.sleep(60*50)
#sweep2
filename= "%s_data%s.dat" % (root_filename, "sweep2")
f=open(foldername + "/" + filename,"w")
IVSweep_VP(STARTCURRENT_VP, STOPCURRENT_VP,NUMSTEPS_VP, f)
f.close()

#sweep3
filename= "%s_data%s.dat" % (root_filename, "sweep3")
f=open(foldername + "/" + filename,"a")
centralizedMPPT(STARTCURRENT_MPPT, STOPCURRENT_MPPT,
    NUMSTEPS_MPPT, f)
f.close()

#sweep4
filename= "%s_data%s.dat" % (root_filename, "sweep4")
f=open(foldername + "/" + filename,"a")
distributedMPPT(STARTCURRENT_MPPT, STOPCURRENT_MPPT,
    NUMSTEPS_MPPT, f)
f.close()
```

```



eload.setMode("CURR")
eload.setValue(STARTUP_CURRENT)
aa_close(handle)
power_supply_10.deactivate()
power_supply_9.deactivate()
power_supply_8.deactivate()
power_supply_12.deactivate()
power_supply_7.deactivate()
power_supply_6.deactivate()
power_supply_5.deactivate()
power_supply_11.deactivate()
time.sleep(1)
print "program ends"

plt.show()
```

appendix/centralized/5converter.py

D.2 Distributed Implementation

This part of python code works with the mirco-controller C code in Appendix C.2

```
 #!/bin/env python

#==============================
# IMPORTS
#==============================
import sys
import time
import datetime
#from aardvark32.aardvark_py import *
from aardvark64.aardvark_py import *
from array import array
from pilawa_instruments import *
import os
import glob
import numpy as np
import matplotlib.pyplot as plt
```

```

#=====================================
# CONSTANTS
#=====================================
#debug=True
BUFFER_SIZE        = 65535
SLAVE_RESP_SIZE   =     26
INTERVAL_TIMEOUT =    100000
I2C_BITRATE =   400
PORT   = 0
ADDR=3
ADDR1=1
ADDR2=2
PACKET_LENGTH=6
CLEAR_BUFFER_MESSAGE=99
MAX_RESENDS=3 #number of times to resend an i2c message if the
    count read is not correct
debug=False
NUMREADS = 20 # number of reads for the ADC on each sampling
    interval
DUTY_MIN = 1650
DUTY_MAX = 1950
insolation=[5.0,5.0,4.0,4.0,2.5,2.5]


READ_DELAY=0.2 #minimum seems to be 0.05 to prevent hangups
V_DIVIDER=(130.0+10.0)/10.0

VREF=3.3 #microcontroller vref value
ADC_MAX=4096.0

DUTY_COMMAND_PWM1=1
DUTY_COMMAND_PWM2=2
DUTY_COMMAND_PWM3=3
VOLTAGE_COMMAND=4
PING_COMMAND=5

command_string=("NO_COMMAND","DUTY_COMMAND_PWM1","
    DUTY_COMMAND_PWM2","DUTY_COMMAND_PWM3","VOLTAGE_COMMAND","
    PING_COMMAND")


STARTCURRENT=0
```

```
STOPCURRENT=5
NUMSTEPS=10.0

DISTRIBUTED_ITERATIONS=50
STARTCURRENT_MPPT=2.0
STOPCURRENT_MPPT=3.5
NUMSTEPS_MPPT=5.0



 D_list=[1800,1800,1800,1800,1800]
#D_list=[1680,1920,1680,1920,1680]



#==============================================
# FUNCTIONS
#==============================================

def round_integer(number):
    return int(number+0.5)

def MPPTPing(converter_list):
    converter_alive=0
    while converter_alive < len(converter_list):
        converter_alive=0
        for x in converter_list:
            time.sleep(0.2)
            print "Establishing communication to addr %s......" %x.addr
            time.sleep(0.2)
            return_value = x.ping()
            if return_value == 1:
                converter_alive = converter_alive + 1
                print "addr %s responds" %x.addr
    print "all controllers have responded! :-)"



#==============================================
# MAIN PROGRAM
#==============================================

root_filename = sys.argv[1]

handle = aa_open(PORT)
if (handle <= 0):
    print "Unable to open Aardvark device on port %d" % PORT
    print "Error code = %d" % handle
    sys.exit()

# Ensure that the I2C subsystem is enabled, also do I2C
aa_configure(handle,  AA_CONFIG_GPIO_I2C)

# Enable the I2C bus pullup resistors (2.2k resistors).
# This command is only effective on v2.0 hardware or greater.
# The pullup resistors on the v1.02 hardware are enabled by
    default.
aa_i2c_pullup(handle, AA_I2C_PULLUP_BOTH)

# Enable the Aardvark adapter's power supply.
# This command is only effective on v2.0 hardware or greater.
# The power pins on the v1.02 hardware are not enabled by
    default.
aa_target_power(handle, AA_TARGET_POWER_BOTH)

# free the I2C from a held bus condition, if any
aa_i2c_free_bus(handle)

# Set the slave response
slave_resp = array('B', [ 0 for i in range(SLAVE_RESP_SIZE) ])
for i in range(SLAVE_RESP_SIZE):
    slave_resp[i] = ord('A') + i
aa_i2c_slave_set_response(handle, slave_resp)



delimiter=', '
t=time.strftime('%Y%m%d')
current_directory=os.curdir
print "Current directory: %s" % current_directory
foldername=current_directory + "/data/" + t
print foldername
if not os.path.exists(foldername):
    os.makedirs(foldername)
gpib  = prologix_serial(port="/dev/ttyUSB0", baud=115200, debug=
    False, timeout=5)
eload= prologix_6060b(prologix=gpib, addr=4, mode="VOLT", rang="
    20",  debug=False)
power_supply_5=prologix_6632a(prologix=gpib, addr=5,debug=False)
power_supply_6=prologix_6632a(prologix=gpib, addr=6,debug=False)
power_supply_7=prologix_6632a(prologix=gpib, addr=7,debug=False)
power_supply_8=prologix_6632a(prologix=gpib, addr=8,debug=False)
power_supply_9=prologix_6632a(prologix=gpib, addr=9,debug=False)
power_supply_10=prologix_6632a(prologix=gpib, addr=10,debug=
    False)
power_supply_11=prologix_6632a(prologix=gpib, addr=11,debug=
    False)
power_supply_12=prologix_6632a(prologix=gpib, addr=12,debug=
    False)
meter1 = prologix_34401a(prologix=gpib, addr=21, mode="VOLT",
    maxrange="25", nplc="1", debug=False)
meter2 = prologix_34401a(prologix=gpib, addr=22, mode="VOLT",
    maxrange="1",  nplc="1", debug=False)
meterList = [meter1,meter2]
meter_addrList=[21,22]
scaleList = [1, 1000]
for x in meterList:
    x.waitForTrigger()
gpib.trigger_devices(meter_addrList)
eload_voltage=float(meter1.readData())
eload_current=float(meter2.readData())*1000.0
print "eload_current: %s" % eload_current
print "eload_voltage: %s" % eload_voltage



# doing bare string sweep
filename= "%s_string_sweep.dat" %root_filename
f=open(foldername + "/" + filename,"w")
print "doing IV current sweep"
sweep_current=STARTCURRENT
step_size=(STOPCURRENT-STARTCURRENT)/NUMSTEPS
eload.setMode("CURR")
eload.setSlew(2000000)
eload.setValue(sweep_current)
while (sweep_current < stopcurrent):
    eload.setValue(sweep_current)
    for x in meterList:
        x.waitForTrigger()
    time.sleep(0.1)
    gpib.trigger_devices(meter_addrList)
    eload_voltage=str(meter1.readData())
    eload_current=str(float(meter2.readData())*1000.0)
    print "eload_current: %s" % eload_current
    print "eload_voltage: %s" % eload_current

    timestamp = datetime.datetime.now()
    f.write("%s%s%s%s%s\n" % ( timestamp.strftime("%H%M%S%f"), delimiter, eload_current, delimiter, eload_voltage))
    sweep_current=sweep_current+step_size
print ("IV current sweep done")
eload.setMode("CURR")
eload.setValue(STARTUP_CURRENT)


print ("Connect DPP converters and press any key to continue")


# doing dpp sweep
filename= "%s_dpp_sweep.dat" %root_filename
f=open(foldername + "/" + filename,"w")
current_step=(STOPCURRENT_MPPT-STARTCURRENT_MPPT)/NUMSTEPS_MPPT
current=STARTCURRENT_MPPT
while (current <=STOPCURRENT_MPPT):
    #always make sure turn on power supply to controller first
    power_supply_11.setVoltage(10)
    power_supply_11.setCurrent(1)
    power_supply_11.activate()
    power_supply_12.setVoltage(10)
    power_supply_12.setCurrent(1)
    power_supply_12.activate()
    time.sleep(1)

    eload.setMode("CURR")
    eload.setValue(3.3)
    power_supply_5.setVoltage(14)
    power_supply_5.setCurrent(insolation[0])
    power_supply_6.setVoltage(14)
    power_supply_6.setCurrent(insolation[1])
    power_supply_7.setVoltage(14)
    power_supply_7.setCurrent(insolation[2])
    power_supply_8.setVoltage(14)
    power_supply_8.setCurrent(insolation[3])
    power_supply_9.setVoltage(14)
    power_supply_9.setCurrent(insolation[4])
    power_supply_10.setVoltage(14)
    power_supply_10.setCurrent(insolation[5])
    power_supply_6.activate()
    power_supply_5.activate()
    power_supply_7.activate()
    power_supply_8.activate()
    power_supply_9.activate()
    power_supply_10.activate()
    time.sleep(2)
   # Enable the slave
    aa_i2c_slave_enable(handle, ADDR, 0, 0)

    filename= "%s_current_%s.dat" % (root_filename,int(current
   *100))
    f_d=open(foldername + "/" + filename,"w")
    for i in range(1,DISTRIBUTED_ITERATIONS+1):
        result = aa_async_poll(handle, 100000000) #100s timeout
        if (result == AA_ASYNC_I2C_READ):
            # Get data written by master
            print "Receiving information...\n Iteration %s" %i
            (num_bytes, addr, data_in) = aa_i2c_slave_read(
   handle, BUFFER_SIZE)
            if (num_bytes < 0):
                print "error: %s" % aa_status_string(num_read)
            elif (num_bytes==10): #sent from controller1
                D1=(data_in[0]<<8)+data_in[1]
                D2=(data_in[2]<<8)+data_in[3]
                D3=(data_in[4]<<8)+data_in[5]
                D4=(data_in[6]<<8)+data_in[7]
                D5=(data_in[8]<<8)+data_in[9]
                print "D1:%s, D2:%s, D3:%s, D4:%s, D5:%s" %(D1,
   D2,D3,D4,D5)

            else:
                print "????receive %s bytes" %num_bytes
        else :
            print "No information received..."
        f_d.write("%s%s%s%s%s%s%s%s%s%s%s\n" % ( i, delimiter,
   D1, delimiter, D2, delimiter, D3, delimiter, D4, delimiter,
    D5))
    f_d.close()
    aa_i2c_slave_disable(handle)
    eload_voltage=0
    eload_current=0
    for i in range(0,100):
        for x in meterList:
            x.waitForTrigger()
        gpib.trigger_devices(meter_addrList)
        eload_voltage=eload_voltage+float(meter1.readData())
        eload_current=eload_current+float(meter2.readData())
    *1000.0
    eload_voltage=str(eload_voltage/100.0)
    eload_current=str(eload_current/100.0)
    print "eload_current: %s" % eload_current
    print "eload_voltage: %s" % eload_voltage
    timestamp = datetime.datetime.now()
    f.write("%s%s%s%s%s\n" % ( timestamp.strftime("%H%M%S%f"),
    delimiter, eload_current, delimiter, eload_voltage))

    power_supply_10.deactivate()
    power_supply_9.deactivate()
    power_supply_8.deactivate()
    power_supply_12.deactivate()
    power_supply_7.deactivate()
    power_supply_6.deactivate()
    power_supply_5.deactivate()
    power_supply_11.deactivate()
    time.sleep(1)
    current=current+current_step


filename= "%s_sweep.dat" %root_filename
f.close()
aa_close(handle)
print "program ends"
```

appendix/distributed/5converter.py

REFERENCES

[1] *FDMF6704v Datasheet*, Fairchild Semiconductor International, Inc., 2011, [Online] Available at www.fairchild.com.

[2] S. Qin and R. C. Pilawa-Podgurski, "Sub-module differential power processing for photovoltaic applications," in *Proc. of the Applied Power Electronics Conference and Exposition*, 2013, pp. 101–108.

[3] U. D. of Energy, "Sunshot initative," 2011. [Online]. Available: http://www1.eere.energy.gov/solar/sunshot/index.html

[4] U. D. of Energy, "Photovoltaic pricing trends: Historical, recent and near-term projections," Nov. 2012.

[5] P. Bakas, A. Marinopoulos, and B. Stridh, "Impact of pv module mismatch on the pv array energy yield and comparison of module, string and central mppt," in *Photovoltaic Specialists Conference (PVSC), 2012 38th IEEE*, June 2012, pp. 001 393–001 398.

[6] S. MacAlpine, M. Brandemuehl, and R. Erickson, "Beyond the module model and into the array: Mismatch in series strings," in *Photovoltaic Specialists Conference (PVSC), 2012 38th IEEE*, June 2012, pp. 003 392–003 396.

[7] R. Erickson and A. Rogers, "A microinverter for building-integrated photovoltaics," in *Proc. of the Applied Power Electronics Conference and Exposition*, 2009, pp. 911–917.

[8] A. Trubitsyn, B. Pierquet, A. Hayman, G. Gamache, C. Sullivan, and D. Perreault, "High-efficiency inverter for photovoltaic applications," in *Proc. of the Energy Conversion Congress and Exposition*, 2010, pp. 2803–2810.

[9] G. R. Walker and P. C. Sernia, "Cascaded dc-dc converter connection of photovoltaic modules," *IEEE Transaction on Power Electronics*, vol. 19, no. 4, pp. 1130–1139, 2004.

[10] N. Femia, G. Lisi, G. Petrone, G. Spagnuolo, and M. Vitelli, "Distributed maximum power point tracking of photovoltaic arrays: Novel approach and system analysis," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 7, pp. 2610–2621, 2008.

[11] L. Linares, R. Erickson, S. MacAlpine, and M. Brandemuehl, "Improved energy capture in series string photovoltaics via smart distributed power electronics," in *Proc. of the Applied Power Electronics Conference and Exposition*, 2009, pp. 904–910.

[12] R. Pilawa-Podgurski and D. Perreault, "Submodule integrated distributed maximum power point tracking for solar photovoltaic applications," *IEEE Transactions on Power Electronics*, vol. 28, no. 6, pp. 2957–2967, 2013.

[13] M. Schuck and R. Pilawa-Podgurski, "Ripple minimization in asymmetric multiphase interleaved dc-dc switching converters," in *Energy Conversion Congress and Exposition (ECCE), 2013 IEEE*, Sept 2013, pp. 133–139.

[14] P. Shenoy, K. Kim, B. Johnson, and P. Krein, "Differential power processing for increased energy production and reliability of photovoltaic systems," *IEEE Transactions on Power Electronics*, vol. 28, no. 6, pp. 2968–2979, 2013.

[15] H. Bergveld, D. Buthker, C. Castello, T. Doorn, A. de Jong, R. van Otten, and K. de Waal, "Module-level dc/dc conversion for photovoltaic systems: The delta-conversion concept," *IEEE Transactions on Power Electronics*, vol. 28, no. 4, pp. 2005–2013, 2013.

[16] J. Stauth, M. Seeman, and K. Kesarwani, "Resonant switched-capacitor converters for sub-module distributed photovoltaic power management," *IEEE Transactions on Power Electronics*, vol. 28, no. 3, pp. 1189–1198, 2013.

[17] C. Olalla, D. Clement, M. Rodriguez, and D. Maksimovic, "Architectures and control of submodule integrated dc-dc converters for photovoltaic applications," *IEEE Transactions on Power Electronics*, vol. 28, no. 6, pp. 2980–2997, 2013.

[18] S. Poshtkouhi, A. Biswas, and O. Trescases, "Dc-dc converter for high granularity, sub-string mppt in photovoltaic applications using a virtual-parallel connection," in *Proc. of the Applied Power Electronics Conference and Exposition*, 2012, pp. 86–92.

[19] T. Shimizu, M. Hirakata, T. Kamezawa, and H. Watanabe, "Generation control circuit for photovoltaic modules," *IEEE Transactions on Power Electronics*, vol. 16, no. 3, pp. 293 –300, May 2001.

[20] T. Shimizu, O. Hashimoto, and G. Kimura, “A novel high-performance utility-interactive photovoltaic inverter system,” *IEEE Transaction on Power Electronics*, vol. 18, no. 2, pp. 704–711, 2003.

[21] C. Schaef and J. T. Stauth, “Multilevel power-point-tracking for variable-conversion-ratio photovoltaic ladder converters,” in *Proc. of the Workshop on Control and Modeling for Power Electronics*, 2013, pp. 1–7.

[22] H. Field and A. Gabor, “Cell binning method analysis to minimize mismatch losses and performance variation in silicon-based modules,” in *Proc. of the Photovoltaic Specialists Conference*, 2002, pp. 418–421.

[23] C. E. Chamberlin, M. A. Rocheleau, M. W. Marshall, A. M. Reis, N. T. Coleman, and P. Lehman, “Comparison of pv module performance before and after 11 and 20 years of field exposure,” in *Proc. of the Photovoltaic Specialists Conference*, 2011, pp. 101–105.

[24] H. Nagayoshi, S. Orio, Y. Kono, and H. Nakajima, “Novel pv array/module i-v curve simulator circuit,” in *Proc. Conf Photovoltaic Specialists Conf. Record of the Twenty-Ninth IEEE*, 2002, pp. 1535–1538.

[25] P. Sanchis, I. Echeverria, A. Ursua, O. Alonso, E. Gubia, and L. Marroyo, “Electronic converter for the analysis of photovoltaic arrays and inverters,” in *Proc. IEEE 34th Annual Power Electronics Specialist Conf. PESC ’03*, vol. 4, 2003, pp. 1748–1753.

[26] M. Cirrincione, M. C. Di Piazza, G. Marsala, M. Pucci, and G. Vitale, “Real time simulation of renewable sources by model-based control of dc/dc converters,” in *Proc. IEEE Int. Symp. Industrial Electronics ISIE 2008*, 2008, pp. 1548–1555.

[27] W. Lee, Y. Kim, Y. Wang, N. Chang, M. Pedram, and S. Han, “Versatile high-fidelity photovoltaic module emulation system,” in *Proc. Low Power Electronics and Design (ISLPED) 2011 Int. Symp*, 2011, pp. 91–96.

[28] C.-H. Chang, E.-C. Chang, and H.-L. Cheng, “A high-efficiency solar array simulator implemented by an llc resonant dc-dc converter,” *IEEE Transactions on Power Electronics*, vol. 28, no. 6, pp. 3039 –3046, June 2013.

[29] “Generating i-v curves with the agilent e4360a solar array simulator using the parameters v_{oc}, i_{sc}, n, and r_s, application note,” *Agilent*, 2009.

[30] “Photovoltaic power profile emulation (pppe),” *Magna-Power Electronics*, [online] www.magna-power.com.

[31] D. Sera, R. Teodorescu, and P. Rodriguez, "Pv panel model based on datasheet values," in *Proc. IEEE Int. Symp. Industrial Electronics ISIE 2007*, 2007, pp. 2392–2396.

[32] M. Villalva, J. Gazoli, and E. Filho, "Comprehensive approach to modeling and simulation of photovoltaic arrays," *Power Electronics, IEEE Transactions on*, vol. 24, no. 5, pp. 1198–1208, May 2009.

[33] C. Sullivan, J. Awerbuch, and A. Latham, "Decrease in photovoltaic power output from ripple: Simple general calculation and the effect of partial shading," *Power Electronics, IEEE Transactions on*, vol. 28, no. 2, pp. 740 –747, Feb. 2013.

[34] D. Sera, R. Teodorescu, J. Hantschel, and M. Knoll, "Optimized maximum power point tracker for fast-changing environmental conditions," vol. 55, no. 7, pp. 2629–2637, 2008.

[35] A. Bratcu, I. Munteanu, S. Bacha, D. Picault, and B. Raison, "Cascaded dc-dc converter photovoltaic systems: Power optimization issues," *Industrial Electronics, IEEE Transactions on*, vol. 58, no. 2, pp. 403–411, Feb 2011.

[36] N. D. Benavides, T. Esram, and P. L. Chapman, "Ripple correlation control of a multiple-input dc-dc converter," in *Proc. IEEE 36th Power Electronics Specialists Conf. PESC '05*, 2005, pp. 160–164.

[37] A. Latham, R. Pilawa-Podgurski, K. Odame, and C. Sullivan, "Analysis and optimization of maximum power point tracking algorithms in the presence of noise," *Power Electronics, IEEE Transactions on*, vol. 28, no. 7, pp. 3479 –3494, july 2013.

[38] C. Lu, V. Raghunathan, and K. Roy, "Maximum power point considerations in micro-scale solar energy harvesting systems," in *Proc. IEEE Int Circuits and Systems (ISCAS) Symp*, 2010, pp. 273–276.

[39] R. C. N. Pilawa-Podgurski, W. Li, I. Celanovic, and D. J. Perreault, "Integrated CMOS dc-dc converter with maximum power point tracking for a portable thermophotovoltaic power generator," in *Third Annual IEEE Energy Conversion Congress and Exposition*, 2011.

[40] R. C. N. Pilawa-Podgurski, "Architectures and circuits for low-voltage energy conversion and applications in renewable energy and power management," Ph.D. dissertation, (http://dspace.mit.edu/handle/1721.1/71485) Department of Electrical Engineering and Computer Science Massachusetts Institute of Technology, 2012. [Online]. Available: http://dspace.mit.edu/handle/1721.1/71485

[41] C. Deline, B. Marion, J. Granata, and S. G. Sigifredo, "A performance and economic analysis of distributed power electronics in photovoltaic systems," Tech. Rep. NREL/TP-5200-50003, January 2011.

[42] *SiC780 Datasheet*, Vishay Intertechnology, Inc., 2013, [Online] Available at www.vishay.com.

[43] *STM32F1 Reference Manual*, STMicroelectronics, 2011, [Online] Available at www.st.com.

[44] A. Peterchev and S. Sanders, "Quantization resolution and limit cycling in digitally controlled pwm converters," *IEEE Transactions on Power Electronics*, vol. 18, no. 1, pp. 301–308, 2003.

[45] C. Barth and R. Pilawa-Podgurski, "Implementation of dithering digital ripple correlation control," in *Proc. of the Workshop on Control, Modeling and Simulation in Power Electronics*, 2013, pp. 1–7.

[46] B. Arbetter, R. Erickson, and D. Maksimovic, "Dc-dc converter design for battery-operated systems," in *Power Electronics Specialists Conference, 1995. PESC '95 Record., 26th Annual IEEE*, vol. 1, June 1995, pp. 103 –109 vol.1.

[47] B. Arbetter and D. Maksimovic, "Control method for low-voltage dc power supply in battery-powered systems with power management," in *Power Electronics Specialists Conference, 1997. PESC '97 Record., 28th Annual IEEE*, vol. 2, June 1997, pp. 1198 –1204 vol.2.

[48] J. Choi, D. Huh, and Y. Kim, "The improved burst mode in the stand-by operation of power supply," in *Applied Power Electronics Conference and Exposition, 2004. APEC '04. Nineteenth Annual IEEE*, vol. 1, 2004, pp. 426–432 vol.1.

[49] Y. Jang and M. Jovanovic, "Light-load efficiency optimization method," *Power Electronics, IEEE Transactions on*, vol. 25, no. 1, pp. 67 –74, Jan. 2010.

[50] M. Schuck, "Implementation and control of distributed maximum power point tracking in solar photovoltaic applications," M.S. thesis, (http://hdl.handle.net/2142/44124) Department of Electrical Engineering and Computer Science, University of Illinois at Urbana-Champaign, 2013. [Online]. Available: http://hdl.handle.net/2142/44124

[51] C. Barth and R. Pilawa-Podgurski, "Dithering digital ripple correlation control with digitally-assisted windowed sensing for solar photovoltaic mppt," in *Proc. of the Applied Power Electronics Conference and Exposition*, 2014.

[52] B. Gharesifard and J. Cortes, "Distributed continuous-time convex optimization on weight-balanced digraphs," *IEEE Transactions on Automatic Control*, to appear, 2014.

[53] R. A. Horn and C. R. Johnson, *Matrix Analysis.* Cambridge University Press, 1990.

[54] R. J. Serna, B. J. Pierquet, J. Santiago, and R. Pilawa-Podgurski, "Field measurements of transient effects in photovoltaic panels and its importance in the design of maximum power point trackers," in *Proc. of the Applied Power Electronics Conference and Exposition*, 2013, pp. 3005–3010.

[55] S. Qin, K. Kim, and R. Pilawa-Podgurski, "Laboratory emulation of a photovoltaic module for controllable insolation and realistic dynamic performance," in *Proc. of the Power and Energy Conference at Illinois*, 2013, pp. 23–29.