DAY 1: Introduction

Tuesday, January 10, 2023

As part of my IAP Adventures, I'm taking The Art and Science of PCB Design!

I've decided that I want to make an MPPT solar charge controller for my project. I made an attempt at this in my junior fall, but course constraints prevented me from using a microcontroller, which made the design a lot harder. I did make it much farther than I anticipated, but on my second attempt, I hope to make it even farther!

What is an MPPT?

Before we can build an MPPT, we need to know what an MPPT is!

"MPPT" is an abbreviation for "Maximum Power Point Tracker." Its job is to make sure that some input power source, a solar cell in this case, is supplying the maximum possible power to a battery.

If you look at a solar cell's IV curve for a given irradiance (amount of incoming light), you'll get a curve that looks like this. If you plot the PV curve on top of it, you can see that there is a given voltage at which the solar array puts out the most power. This point is called the **maximum power point.**

But what if we want to connect our solar array to a battery to charge it? Since the battery is a constant voltage source, it will force the solar array to operate at its own voltage. But this voltage might drag the solar array away from the maximum power point.

To mitigate this, we can place a DC-DC converter in between the solar array and the battery. Now the solar array is free to operate at its optimal voltage, and that power can be converted down to voltage and current levels that the battery can actually accept.

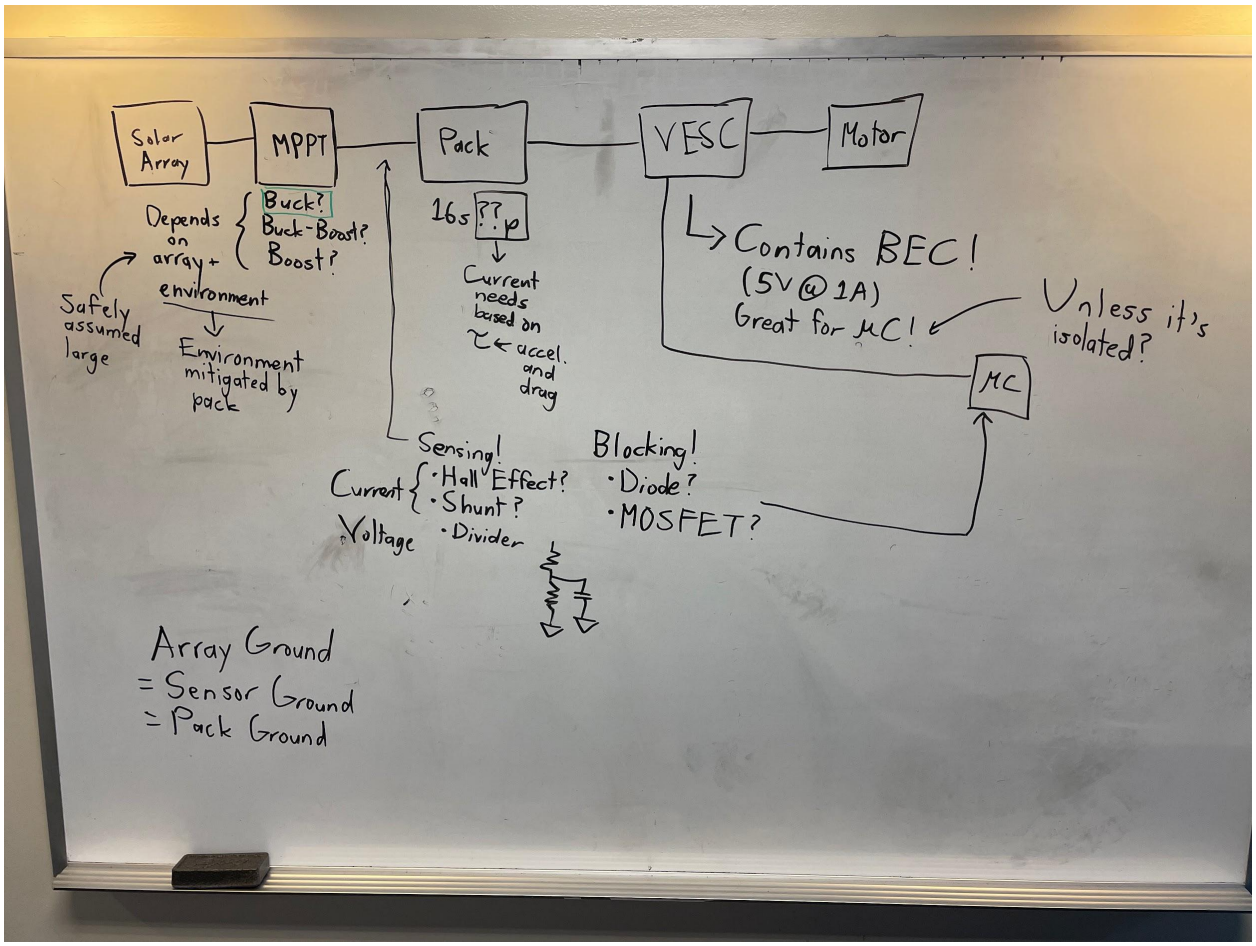
DAY 2: System Overview

Thursday, January 12, 2023

Alright, today is the day we buckle down and actually do some designing! We'll start by pinning down the specifics of how this system is going to work on the whole.

The System

Here's a system diagram on a whiteboard! We'll break this down, as well as clean this up a bit at the end of this log, but the rawness and messiness of the original diagram shows the joy of the process.



This architecture is basically the same as the one that we've been using on the Solar Electric Vehicle Team for some time now, but our focus this time around is on the MPPT in particular. However, to size the MPPT, we do need to know what we're building for.

The input to the MPPT is the solar array, and the output of the MPPT is power to a battery pack. In between, we also have some sensing performed by a microcontroller. Finally, there's a safety feature or two we can include between the MPPT and the pack, but we'll worry about those later.

If we know what we ultimately want to drive, we can come up with reasonable estimates for how much input power we need. So, let's start by considering the MPPT output!

Output: The Battery Pack

The battery pack will receive all of the power that our MPPT outputs. It will also source all of the power that the motor controller requests.

Our MPPT needs to be able to put out enough voltage and current to charge this battery pack. I say "voltage and current" instead of just "power" because even if our MPPT defied physics and became 500% efficient in terms of power, if that power is delivered at one volt, we won't be able to charge our battery at all.

This is because current flows from higher voltage to lower voltage. So for our MPPT to be able to push current into our battery, it needs to be capable of an output voltage of at least the maximum voltage of our battery pack. **The battery pack gives us a lower bound for our maximum MPPT output voltage.**

So, how big might our battery pack be? Well, the battery pack will be driving a motor through a Flipsky motor controller (which is based on the VESC architecture). The specifications listed it as accepting 14V-84V as its input voltage. We want this design to work for any battery pack that the user might want to use, so we'll design around a battery pack that gives us **84V at maximum charge.**

Implications on MPPT DC Converter Architecture

Alright, so now we know that we need to be able to put out at least 84V to be able to fully charge our battery. This is actually a very valuable piece of information; most commercial solar panels are nominally 12V or 24V. For the average consumer trying to charge their motorcycle battery, constructing a solar array that can put out 84V on its own is a nontrivial task.

But what about people using MPPTs for solar-powered vehicles? These sorts of vehicles tend to have very large solar arrays, and their output voltage can easily exceed that of an 84V battery.

In the name of compatibility, then, we'll use a **buck-boost DC converter** for our MPPT. A buck-boost converter is capable of generating an output voltage that is either lower or higher than its input voltage, whereas a buck converter can only output a lower voltage and a boost converter can only output a higher voltage. With that, our commercial user can use the boost converter functionality to charge their motorcycle battery, whereas a solar vehicle racer can use the buck converter functionality to charge their battery.

Feedback Control

The last major system component to consider is how we'll do the feedback control. My current plan is to use a microcontroller and have it do analog sensing to determine the current and voltage outputs of the MPPT. It can then use this to determine the current power output and decide if the solar array is operating at the maximum power point.

There is one problem, however. In my initial block diagram on the whiteboard, I put the microcontroller on the BEC from the VESC. This works perfectly fine, but it does force the MPPT to be coupled to the VESC, which we probably don't want.

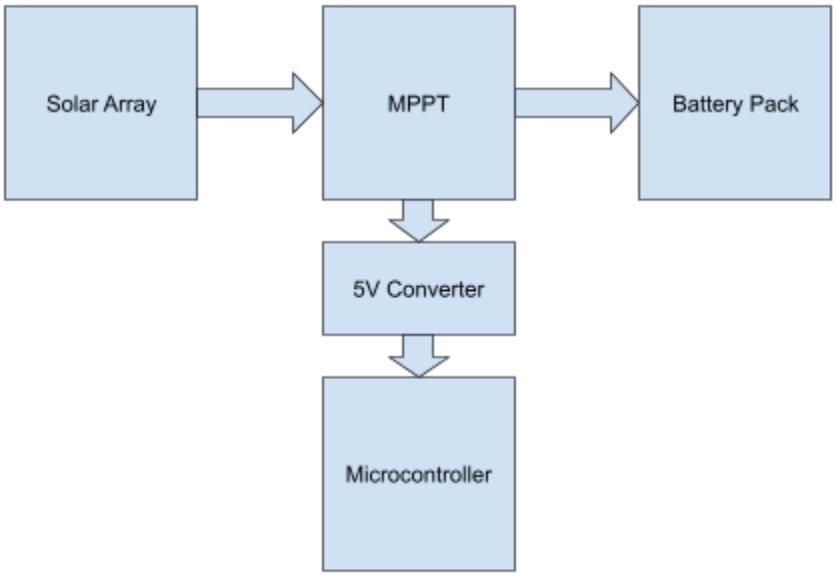
As such, I'll probably include a 5V buck converter on the MPPT. Some of the solar power will be diverted to power this converter, but better to take it from the source than to eat all of the losses from the MPPT, battery pack, AND the VESC before routing it back to the microcontroller.

I'll worry about exactly how to do voltage and current sensing later, but I'll probably just use passive components routed into the microcontroller's ADC.

Final Diagram

That leaves us with this final diagram for our MPPT!

I suppose next time I'll do something like analysis on either the different MPPT algorithms or considering ways to maximize DC converter efficiency.



Day 3: MPPT Architecture

Tuesday, January 17, 2023

Julian was naughty and didn't really do any actual work on his design for the entire weekend. He also didn't do any work yesterday, which was Martin Luther King Jr. Day. But today, he's back and ready to rumble!

Turning On: Converters All the Way Down

OK, so here's the thing… I know that I was originally planning to start today with either MPPT algorithms or DC converter efficiency and design. But I realized that there's a much more pressing issue in our block diagram from yesterday. Specifically… how does this thing turn on?!

The MPPT is effectively a glorified DC converter. DC converters do their DC converting by flipping switches (MOSFETs) on and off really fast. We can't do this without powering the microcontroller, which will be handling all of the feedback control.

Yesterday, the plan was to divert some of the solar energy to a 5V converter, which would in turn power the microcontroller. But… a 5V converter, being a DC converter, would ALSO have switches that need to be turned on, which also requires power. We have a recursive power problem–our converters need converters! It's turtles all the way down!

To break the chain, we need a first mover. We have a couple of options.

Supplemental Battery as the First Mover

We could use a supplemental battery, which would supply power to both the microcontroller and to the gate drivers so they can get the ball rolling. Then, the microcontroller could drive a 5V converter and start drawing power from it instead of from the battery. The whole thing also won't just stop if the solar array goes under a cloud.

But the problem with batteries is that they discharge. If a supplemental battery is necessary to run our charge controller, then we'd have to replace its battery every thirty days. And for a charger that's allegedly solar powered, that seems really, really backwards. Not inherently problematic. Just really backwards.

Linear Regulator as the First Mover

To avoid this, we could also use a linear regulator instead of a 5V buck converter, eliminating the need for a first mover. The benefit of a linear regulator is that we don't need to do any toggling to get it to function, so as soon as we get enough current from the solar cells, everything just turns on immediately. That way we don't have to do any switching from one power source to another.

Gate Drive Power

There's also the issue of powering the gate drive circuitry. Since the gate drive will almost certainly operate at a higher voltage than the microcontroller, we can actually place the linear regulator here, then use a buck converter to drop the power down to the 5V for the microcontroller.

Gate drive draws a non-trivial amount of power, so using a linear regulator to power it will probably prove to be a bit lossy. However, getting a functional design is my focus for now; we can iterate on this part of the design next.

New Target Audience, New Specifications

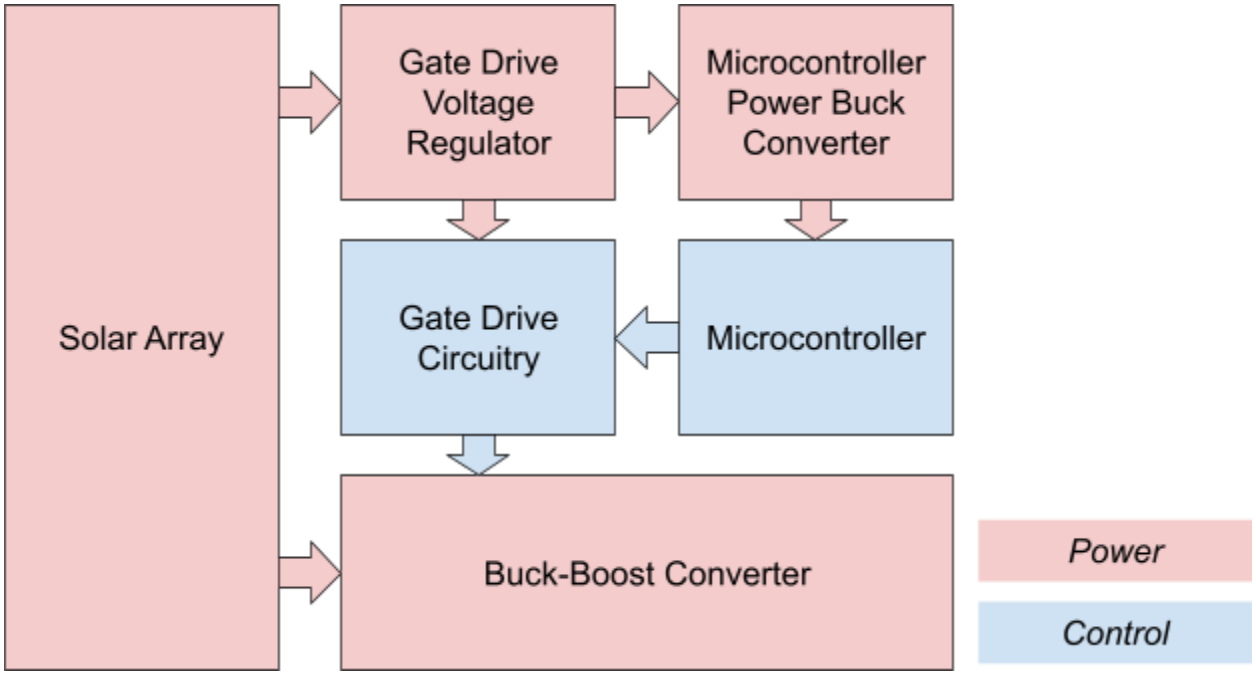
There's also the question of who we're designing for. On Day 2, I decided to design for the arbitrary Flipsky user, which made my target output voltage 84 volts.

However, given my involvement with MIT SEVT, it would make a lot more sense to design for batteries of that scale. MIT SEVT uses a 32s13p battery pack made up of 18650 lithium-ion cells, which would put its operating voltage range at **80V-135V.** With respect to the solar array, it puts out just under **1 kW** of power, with its longest string of solar cells being just over 115 volts. Thus, we're switching our specifications to match.

SPECIFICATIONS	
Input Voltage (Maximum)	120 V
Input Power	1 kW
Output Voltage (Minimum)	80 V
Output Voltage (Maximum)	135 V

New Topology

With these choices in mind, we'd better pin down our topology before the day's end, since we need to start sifting through components to see if this design is viable.



Sounds like a plan! Tomorrow, we can select parts!

DAYS 4-7: Component Selection

Wednesday, January 18, 2023
Sunday, January 22, 2023
Monday, January 23, 2023
Thursday, January 26, 2023

Welp, I can't put it off any longer. Time to start picking components!

What to Pick When

To pin down the order we need to select components, we should get a sense of what our dependencies are. Specifically, it makes the most sense to select components that inform us of the specifications of other parts of the board.

1. **Power.** Since they're the heftiest parts of the system, we'll start with the components carrying the highest amounts of power. This will be things like our DC converter modules, or if we can't find a chip to do it for us, our inductors, capacitors, and, most importantly, power MOSFETs.
2. **Gate Drive.** Once we at least have power MOSFETs selected, we'll know threshold voltages and gate charge characteristics. This will be key information in how we select our gate drivers.
3. **Voltage Regulator.** And then, once we know what our gate drive modules need to run at our desired switching frequencies, we'll know what our voltage regulator needs to be able to handle.
4. **Microcontroller.** This thing draws very little power and can honestly be made to work with just about anything we pick, so we can leave it for last.

Power

Alright, let's start with the meat of the system! Given that we're designing an MPPT, our number one design priority for power transfer is **efficiency.** We'll do this by minimizing losses. With respect to our power component selection, there are two main sides of this we need to consider at this stage:

- **Parasitics.** Our inductors and capacitors aren't going to be perfect. Some of the energy we ask them to store will be lost as heat. When we select components, we need to be aware of their efficiency and make choices accordingly.
- **Switching Losses.** The more we have to switch our MOSFETs on and off, the more losses we'll experience at the gates. We can mitigate this by using a lower switching frequency, but the trade-off is that we'll have to use larger inductors and capacitors,

which will make layout harder. We could also use MOSFETs with lower threshold voltages to decrease power needed to charge the gates, but this would come at the cost of more leakage power. As such, we'll let switching frequency be our focus for this first iteration.

Specifications

Recall what I'm calling our "input specifications," or the specifications we've established that will inform our first component selections. (I also added short-circuit current for the solar array.) The component selections we make from here will give us new input specifications for our next selections.

INPUT SPECIFICATIONS	
Input Voltage (Maximum)	120 V
Input Current (Maximum)	6.2 A
Input Power (Maximum)	744 W
Output Voltage (Minimum)	80 V
Output Voltage (Maximum)	135 V

With a few simple equations, we derive our buck-boost converter specifications. The components we select will need to be able to handle the following:

MPPT BUCK-BOOST CONVERTER SPECIFICATIONS	
Input Voltage (Maximum)	120 V
Input Current (Maximum)	6.2 A
Output Voltage (Minimum)	80 V
Output Voltage (Maximum)	135 V
Output Current (Maximum)	12.5 A
Output Power (Maximum)	744 W

Alright, we've got our specs! Let's poke around for parts! We'll start by seeing if we can find a chip to do all of the buck-boost converting for us. Otherwise, we'll resort to DIY.

Candidate Components: Buck-Boost IC

I'll list the suppliers and what we find from them as we go.

- **Mouser:** I found two isolated DC converters that could handle 1 kW, and were in the ballpark of our input voltage specs, but their output voltage was limited to a maximum of 24 V. Also, they're chassis mount, and although I'm not certain what that means, I'm reasonably sure that it won't work for a PCB.
- **Analog Devices:** I placed our specifications through the buck-boost regulator search engine. Came back with zero results.
- **Texas Instruments:** Inputting our output voltage requirements immediately sends the number of results to zero. I'm sensing a pattern here…
- **Digikey:** Anything close to our voltage requirements was too wimpy in terms of current.

Yeah, I don't think we'll find much that's of use. Let's design this sucker ourselves!

Buck-Boost Converter Topology

There are a number of DC converter topologies that enable buck-boost design. There are a number of ways we could evaluate them, but I have two main concerns:

- **Efficiency.** As mentioned previously, we shouldn't be designing any element of this MPPT without considering this.
- **Non-Inverting-ness.** The vanilla buck-boost topology you see everywhere is inverting (output voltage polarity is negative with respect to input voltage ground), but that leads to a bunch of headache with getting the control components to be able to manage the power components. To minimize headache (and ensure that I can actually finish this design on time), I'm going to implement a non-inverting design.

The problem is that non-inverting buck-boost converters are used less often because of the tradeoffs they make in efficiency. To explore how to mitigate these tradeoffs, I hopped into this article from Texas Instruments's *Analog Applications Journal*.

The article suggests a topology that is essentially just the concatenation of a standalone buck converter and a standalone boost converter. Given that it turns switches on and off to enter entirely separate modes of operation, analyzing it to size components shouldn't be too difficult. For my background experience, then, I think this is a reasonable option to explore.

Finding Power MOSFETs

We're going to start with the power MOSFETs. The reason for this is because sizing the passive components (inductor and capacitor) requires us to commit to a switching frequency. In principle, a smaller switching frequency leads to smaller switching losses at the cost of using larger components, so we want as large a switching frequency as possible. However, larger components are also heavier, which is a negative for weight-conscious designs (e.g. solar

vehicles). Larger components are also just much more difficult to find, and are often much more expensive.

However, if our power MOSFETs have a particularly low power consumption to begin with, the switching losses may be relatively small, allowing us to accept slightly higher switching losses in exchange for a much more accessible and viable design.

To locate power MOSFETs, I just hopped onto the internet and started browsing. After taking a look at this YouTube video, I was reminded of gallium nitride (GaN) MOSFETs, which have low on resistance and low threshold voltages. I decided to take a look at what I could find, and located a decent selection on Mouser.

Power MOSFETs					
Part	**Material**	**$V_{Threshold}$**	**Gate Charge**	**$R_{DS(On)}$**	**Cost**
TP65H070LSG-TR	GaN	4V	9.3nC	148mΩ	$12.51
GS66504B-TR	GaN	2.6V	3.3nC	130mΩ	$13.86

The GS66504B-TR is very compelling. It has a logic-level threshold voltage and takes very little charge to turn on–little enough such that the 5V regulator is actually a viable option to drive this thing. It also has a low on resistance, which is compelling for an efficiency-focused design like an MPPT. What's more–they're actually in stock, and have SPICE models for simulation and footprint for layout!

So, for now, we will design around this GaN MOSFET.

Sizing Passive Components: The Leeb Method

Let's size components for this topology. We'll use the method I remember from Power Electronics (formerly 6.131, now 6.2220), taught by Steven Leeb, where we size the inductor and capacitor based on switching frequency and ripple specifications.

Let's start by pinning down a switching frequency. Again, the advantage of using a higher switching frequency is that we can use smaller capacitors and inductors, but the drawback is that the increased switching activity means more overall loss. Since we're trying to make this MPPT as efficient as possible, we want to try to push our switching frequency down as low as possible, but our choice of GaN FETs enables us to toy with higher switching frequencies. After iterating on with the calculations below, I settled at ?!? kHz.

Now, let's see what the implications are for our passive components in both the buck converter and the boost converter.

Buck Converter

We can come up with a size for our capacitor based on our desired maximum ripple voltage. I chose a relatively small maximum ripple of 0.1V, since we want voltage to remain fairly steadily at the maximum power point. We can then size our capacitor with our dear friend, Math:

$$Q = CV$$
$$I = \frac{dQ}{dT} = C\frac{dV}{dt} \approx C\frac{\Delta V}{\Delta t}$$
$$C \approx I\frac{\Delta t}{\Delta V} = I\frac{1/f_{sw}}{V_{ripple}}$$
$$C \approx 12.5A * \frac{1/20000s^{-1}}{0.1V}$$
$$C \approx 6.25mF$$

We can do a similar thing to size our inductor. We'll similarly choose a ripple specification of 0.1A.

$$V = L\frac{dI}{dt} \approx L\frac{\Delta I}{\Delta t}$$
$$L \approx V\frac{\Delta t}{\Delta I} = V_{out} * \frac{1/f_{sw}}{I_{ripple}}$$
$$L \approx 135V * \frac{1/20000s^{-1}}{0.1A}$$
$$L \approx 67.5mH$$

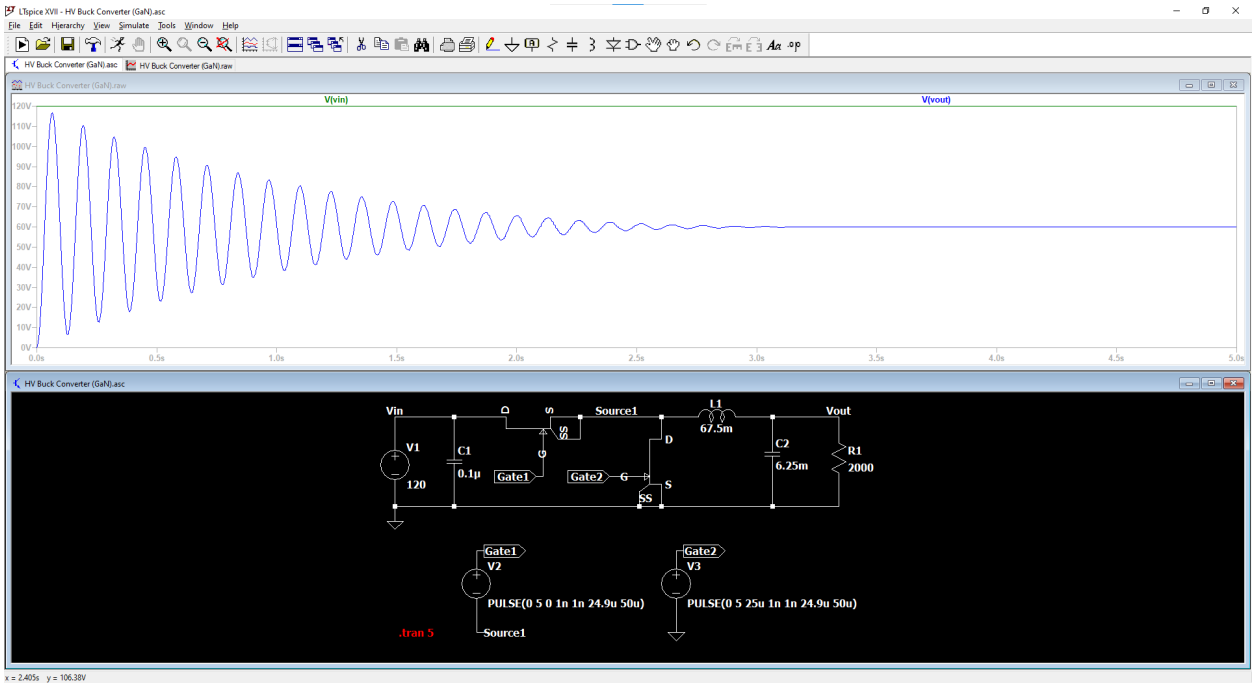
Last, we just need to make sure that the resonant frequency of this circuit will be lower than our switching frequency, lest we get massively boosted noise at our output. The general rule of thumb is to make sure that our resonant frequency is at most a tenth of our actual target switching frequency, so we need to verify that our resonant frequency is 2kHz or lower. Let's see where our resonant frequency lies with our selected capacitor and inductor values:

$$\omega_0 = 2\pi f_0 = \frac{1}{\sqrt{LC}}$$
$$f_0 = \frac{1}{2\pi\sqrt{LC}} = \frac{1}{2\pi\sqrt{67.5e{-}3 * 6.25e{-}3}}$$
$$f_0 = 7.75Hz$$

That is… very extremely low. Doing the math, to put our resonant frequency at a tenth of our switching frequency given our ripple spec, we need an inductor that's at least 1μH, which is… a lot smaller than what we get from the ripple spec. So, it makes sense that our resonant frequency

is super low, but… it is kinda terrifying in a way. Somehow I feel like there is some larger principle at work here, of which I am entirely ignorant, that will come back to bite me. Oh, well. We'll find out when it blows up in the testing phase.

When we try this out in LTSpice, we find that we have a pretty severely high current swing on startup–it actually exceeds our output specification. This suggests that we need to include some sort of precharge circuitry on our output.



Boost Converter

We can do a similar set of calculations to determine sizing for the passive components for the boost converter. The main difference is going to be the voltage value we use for our inductor calculation; given the change in topology between buck mode and boost mode, the voltage across the inductor will now be the difference between the input and output voltages, and not just the output voltage value. Previously, we actually didn't enforce a minimum voltage value for our solar array, and we technically can't prevent it from falling all the way down to zero in no light. For now, we'll limit the boost converter to a duty cycle of no less than **50%**, which means that our minimum input voltage in boost mode should be set to 67.5V, or half the maximum output voltage.

$$Q = CV$$
$$I = \frac{dQ}{dT} = C\frac{dV}{dt} \approx C\frac{\Delta V}{\Delta t}$$

$$C \approx I\frac{\Delta t}{\Delta V} = I\frac{1/f_{sw}}{V_{ripple}}$$
$$C \approx 12.5A * \frac{1/20000s^{-1}}{0.1V}$$
$$C \approx 6.25mF$$

$$V = V_{out} - V_{in} = L\frac{dI}{dt} \approx L\frac{\Delta I}{\Delta t}$$
$$L \approx V\frac{\Delta t}{\Delta I} = (V_{out} - V_{in}) * \frac{1/f_{sw}}{I_{ripple}}$$
$$L \approx 67.5V * \frac{1/20000s^{-1}}{0.1A}$$
$$L \approx 33.75mH$$

By these calculations, the buck converter passives will work perfectly fine for the boost converter.

Gate Drive

Since our GaN FETs have a threshold voltage of under 5V, it would be nice to also find a gate driver that can operate off of a 5V source. In the past, I've used IR2125 gate drivers, but they have a listed minimum input voltage of 12V, which would force us to introduce multiple power stages.

So, after shopping around on Mouser, I tracked down the SIC533CD-T1-GE3, a 5V gate driver. This is great, but the question is: Can it supply the current we need?

The GaN FETs we chose need 3.3nC of gate charge. We'll be driving up to four MOSFETs, so we can calculate our worst-case current draw. (In expected operation, we won't be turning all four MOSFETs on at once, but we'll assume that something goes wrong for extra safety.) Let's assume a switching frequency of 200kHz, and estimate that we want to go from all the way off to all the way on within 10% of a gate's on time. Also recall that we have to calculate based on half the period, since switching frequency includes both on and off time.

$$I = \frac{dQ}{dt}$$
$$I = \frac{3.3nC \cdot 4}{1/2 \cdot 1/10 \cdot 1/200000s^{-1}}$$
$$I = 52.8mA$$

We need a minimum of 52.8 mA from our gate driver, and we're in luck–this one has a maximum output current of 35A! It seems that we're well above the necessary specifications for this one.

Voltage Regulator

It's nice to know that we can afford to drive our gates at 5V, as it means that we'll only need a single voltage regulator. However, we do need to make sure that our voltage regulator can supply the amount of current that we need. Generally, microcontrollers draw negligible current, so we'll ignore that for now and focus on our gate drive current needs, which we calculated in the previous part to be roughly 53mA.

After browsing through possibilities on Mouser, I concluded that the MIC5283-5.0YME is a great candidate. Not only does it support three times as much output current as we may need at the given voltage, but it also sports a high PSSR–in other words, fluctuations in the input voltage will have a minimal effect on the output voltage. With a power supply as volatile as a solar array, anything to ensure steady power output on the microcontroller side will be a great addition.

Microcontroller

MIT SEVT has a long history with the Atmel family of microcontrollers, specifically the AT90CAN128, but given that the world is going the way of the STM, I suppose I could take a look at forging that new path.

I wish to begin with the minimum viable microcontroller, so I've got my eye on the STM32G431K6. It has ADC support, which will be necessary for reading input voltage and current values and giving a report back, and otherwise supports a 32-bit architecture in case we need any sort of precise calculations in our algorithm.

Bill of Materials

That's every major component of the baseline, rudimentary design considered! Here's the full list:

Major Components		
Component	**Part Number**	**Count**
N-Channel GaN FETs	GS66504B-TR	4
Gate Driver	SIC533CD-T1-GE3	4
Capacitor	–	1
Inductor	–	1
Voltage Regulator	MIC5283-5.0YME	1

Microcontroller	STM32G431K6	1

DAY 8: Simulation and Design Review

January 30, 2023

The other day, I had a very helpful design review with Adi and Fischer. I was having some trouble getting my simulations to run properly, so they helped me resolve those problems. In the name of documentation, I want to survey those fixes before calculating some of the implications on components. From there, we'll update the bill of materials before taking a large crack at the schematic and layout.

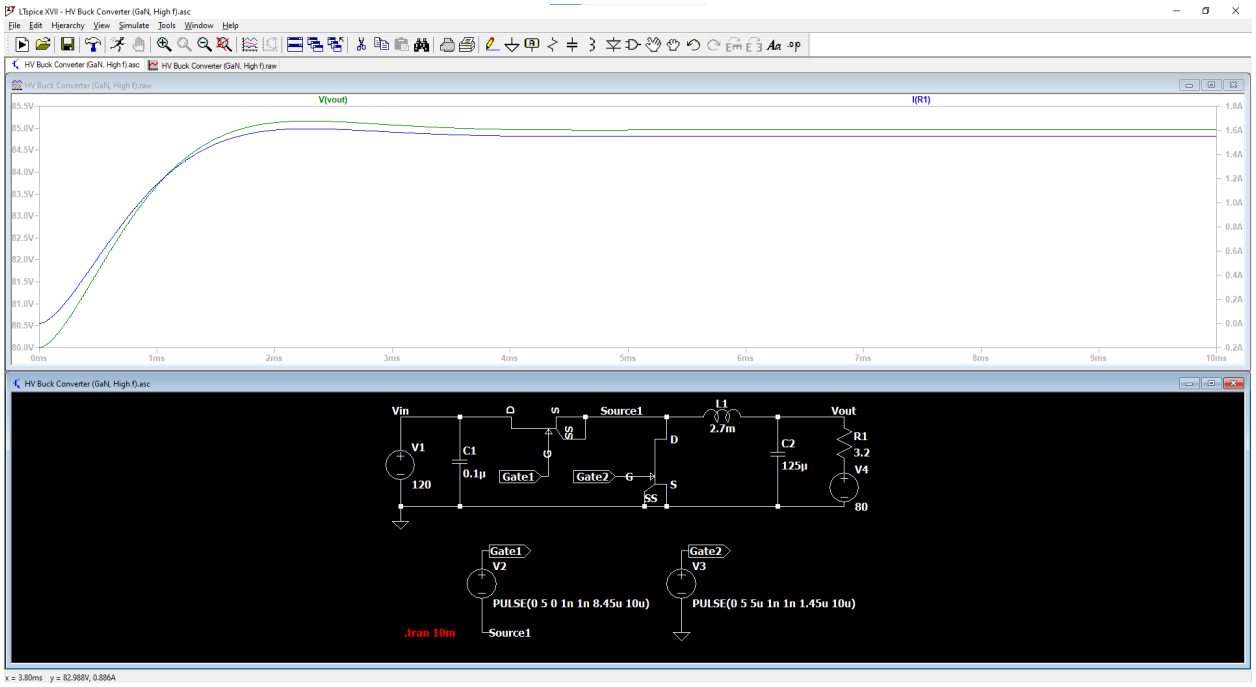
Simulation

Although the topology we're using has both a boost and a buck converter in it, they function such that I could simulate my buck and my boost converters separately. That's what I did.

Buck Converter

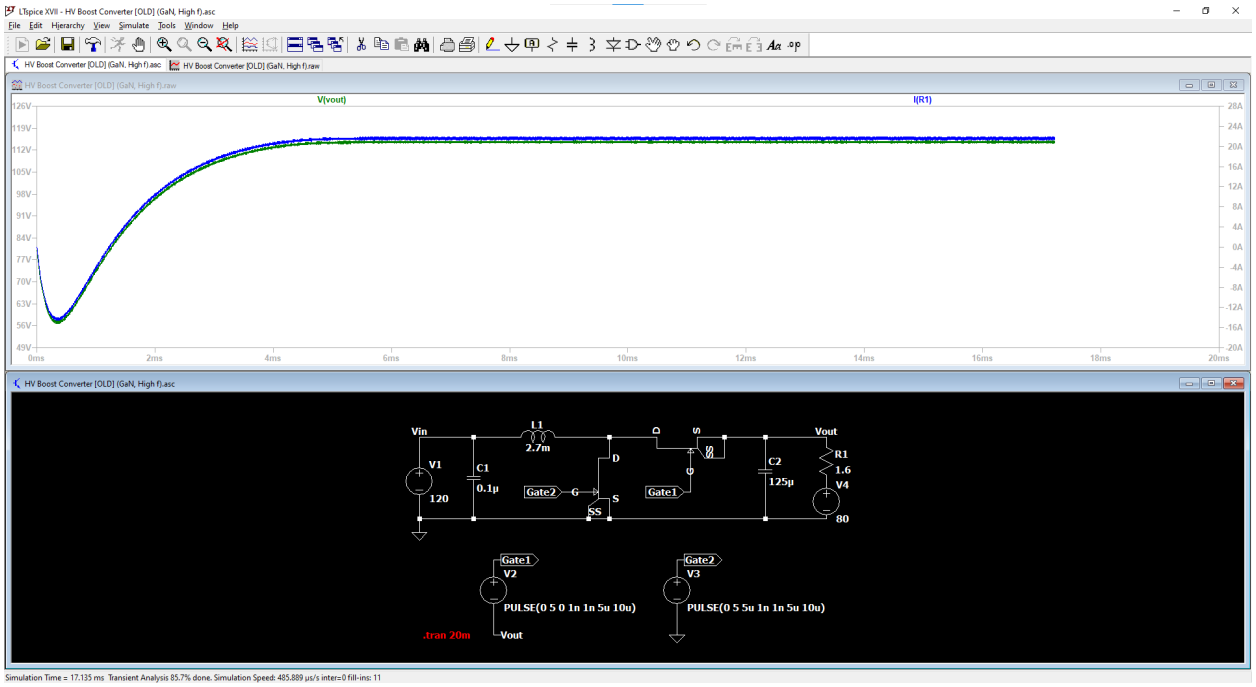
In a previous section, I mentioned that massive current ripple at the output. Well, that's because I was modeling my load stupidly. I chose an arbitrary load because I was focused on the converter itself, but the load–especially the high-impedance load I threw in–dhas a non-trivial effect on circuit operation.

An 18650 cell has an internal voltage and an internal resistance. On SEVT, we allow our cells to fluctuate between 2.5V and 4.2V, and they have an internal resistance of 100mΩ. When I model things accordingly, the buck converter functions much more reasonably.



Boost Converter

The boost converter was a lot trickier. The first problem was that it was operating as a buck converter somehow, and not a boost converter. This is even after correcting the stupid load as discussed in the previous section.

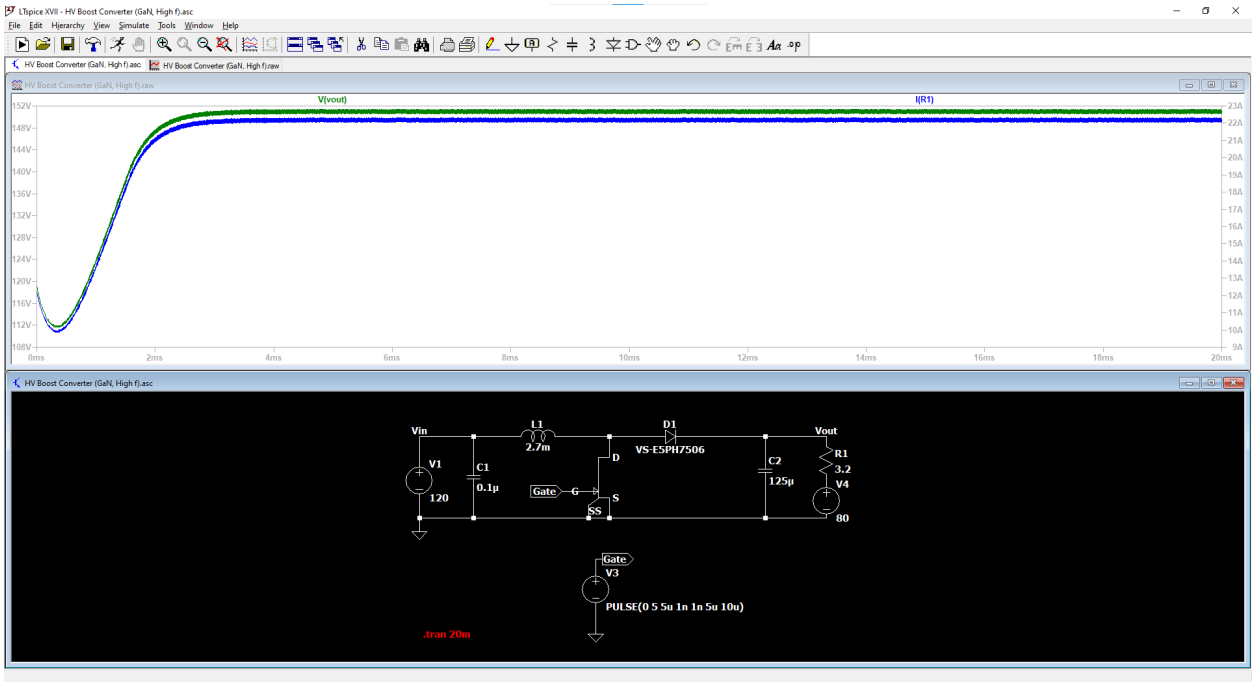


There are two problems here. First of all, there's the dead time in the gate switches. There isn't any explicit dead time in the screenshotted setup, but I had it in previous iterations of the design. The problem with dead time in a boost converter is that it disconnects the inductor, which is supposed to be handling a non-trivial amount of current. And if we know anything about inductors, it's that they strongly resent being disconnected while handling non-trivial amounts of current.

So, I threw a flyback diode across the inductor to see how it would function. But with the dead time in between, a lot of current–multiple amps, in fact–were diverted through that diode. This once again manifested as an inability to build voltage across the output.

So, we abandoned a synchronous design. Although I replaced the diode from the original topology with a GaN FET in the hopes of avoiding the efficiency loss imposed by a diode voltage drop, a diode is the easiest solution to the shoot-through problem, and it prevents the flyback current from blowing anything. What's more, it keeps us from sending current through non-trivial paths.

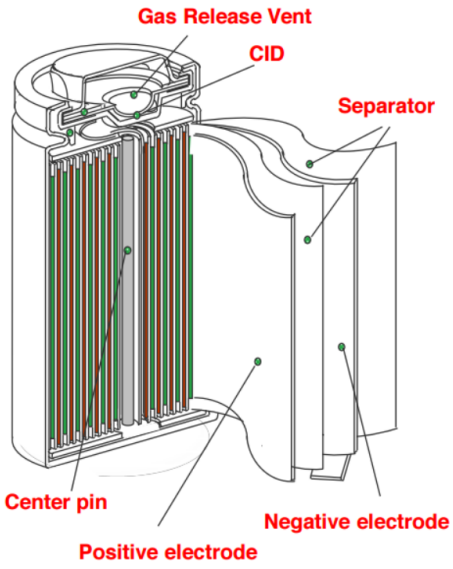
With all of these fixes in place, the boost converter was boosting properly!



It hasn't escaped my attention that our current output is way above specification, but I'm not too concerned about it because our input power source is being modeled as an ideal voltage source, which a solar array is not.

Revisiting the Ripple Specifications

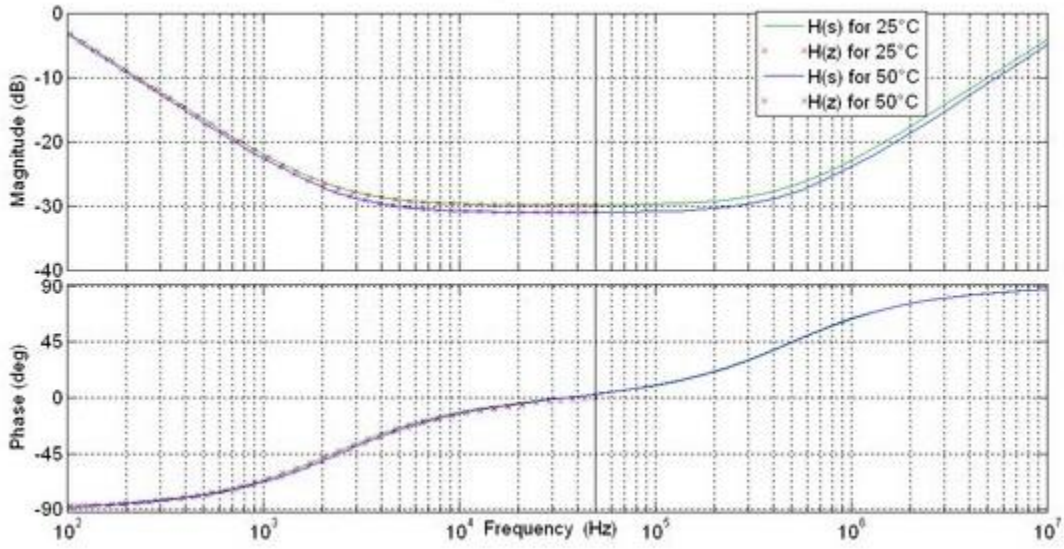
The other main challenge to this current design is just how large my passive components are. I've been driving the switching frequency up to try to get them down, but one other way to get the numbers down is to relax my ripple specification. I spoke with Adi and Fischer about this, and they gave me Good Advice™ that I'll summarize now.



Previously, my ripple specifications were nearly arbitrary–I just knew that I wanted them to be low so I could track my maximum power point closely. However, we can make a more informed decision about our ripple specifications.

To start, let's consider our load–in this case, the battery pack. As the article providing the figure on the left points out, the 18650 cell's positive and negative electrodes are plates rolled up together into a cylinder. In other words, each 18650 cell is basically an electrolytic capacitor. This means that each cell, in addition to has a large inductance in series with some parasitic capacitance.

This research paper took a bode plot of an electrolytic capacitor. Their capacitor wasn't an 18650 cell, but the shape here is still instructive. (We're not too concerned about phase, but rather magnitude.)



We see that, at low frequencies, we get very little attenuation given the series capacitance. Similarly, at high frequencies, we get very little attenuation given the series inductance. But there's a "sweet spot" of frequencies that are attenuated by a large margin.

The implication here is that, if our ripple frequencies tend to fall into the "sweet spot" of frequencies, then our load itself will largely eliminate the appearance of that ripple on our output. So, what sorts of frequencies can we expect in our ripply DC output?

We can approximate a DC signal with ripple as a triangle wave. If you take the FFT of a triangle wave, you'll find that you get a signal on the odd harmonics, which are monotonically decreasing in magnitude. Thus, the high-frequency harmonics are going to have little effect on the signal anyway, so we can set them aside.

But the larger harmonics? We care where they fall on the bode plot for the 18650 cell. Unfortunately, I wasn't able to find any metrics for the exact LC characteristics of the cells. But maybe I could test them?

If I inject a DC signal into an 18650 cell and measure the amount of time it takes for

https://github.com/TjitteS/ReboostV2-Hardware/blob/main/Outputs/Reboost%20V0.2.1/Schematic%20Prints.PDF

https://www.youtube.com/watch?v=ShXNJM6uHLM
https://drive.google.com/drive/folders/1Sd2jWAb-F8NAXlJ6PLdhcnPDQV0alD15