Allison Korczynski
Alvaro Aguilar
Daniel Sohn
11/28/12
EE152

Final Report

Summary:

We designed, simulated and built a Lithium Polymer battery charger based on an intermittent power source. Although we designed for a DC motor as input, another power source with similar voltage and current ranges (-16V to 16V, and 0 to 250mA respectively) could be used. We researched many parts to be used, keeping efficiency in mind as the number one factor. The design iteration occurred several times, always trying to improve the overall efficiency of the power conversion.

Given the polarity of our input stage, a rectifier was necessary. Since the beginning we planned for a high-efficiency synchronous rectifier which utilized MOSFETs rather than diodes to eliminate the usual 1.1V drop associated with bridge rectifiers. We designed and simulated the rectifier stage using LTSpice.

The initial power path design consisted of a buck converter which regulated anything above 4.0V down to 3.8V (an appropriate battery-charging voltage). However, we quickly realized that this would be useless for the periods of time when the motor output was less than 4.0V. For this reason, we decided to change our topology to a buck/boost topology, which would allow us to either regulate the input voltage up or down. The buck/boost topology was not studied in depth in class, so a lot of design hours went into analyzing the circuit. A buck/boost assuming rectified input voltage was simulated using LTSpice, more details can be found under “Key Results” section. As we learned in class, steady-state is not the problem when it comes to DC/DC converters, rather its the transients that are important and must be controlled. In order to develop a control loop around the buck/boost topology, we had to find the transfer function relating small changes in duty cycle to small changes in output voltage. Given the nature of the transfer function, we decided to measure the input and output voltages to the regulator, the output current into the battery, and finally the battery voltage. The control loop consisting of an integrator was validated using MATLAB. Additional details on this can be found under the “Control” section.

Finally, we chose a low-power microcontroller to implement our feedback loop and drive the two switches used for the buck/boost power path. The Atmega168V microprocessor was chosen and code was written from scratch to implement the necessary functions for our application. Namely, we developed code to perform analog to digital conversions, drive two independent PWM outputs and communicate with the computer via UART for debugging. Code was written to implement the control-loop (basically an integrator) and prevent wind-up error. Additional details about each one of these libraries can be found under the “Microcontroller and software” section.

What you did and what you learned:

Parts characterization:

In order to choose a topology for the charging circuit, the input and output stages had to be characterized. We found the motor voltage to vary between -12 to 12V depending on which way it was cranked, which meant a full-bridge rectifier was necessary at the input. Additionally, we learned that the motor outputs no more than 500mA when cranked at full speed. Our initial idea was to use a synchronous FET rectifier to avoid the ~1.2V diode drop. We successfully simulated a circuit which utilized a quad comparator in conjunction with 2 N-channel MOSFETs and 2 P-channel MOSFETs to rectify the output voltage. Details of this simulation and design decisions can be found under the “Key Results” section. Due to time constraints, we decided to use a diode rectifier with a 1.1V drop .

We also studied the battery datasheet to understand the output requirements of our power stage. We used an 860mAh Lithium Polymer battery model number 063058. The charging limit for the voltage and current on this battery are: 4.25V and 860mA. For safety reasons, we decided to charge up to 3.8V with a current of no more than 300mA. Since the motor is an intermittent power source, our design trickle-charges the battery. Given the fact that we need a steady output voltage of 3.8V with an input voltage that could range anywhere from 0V to approximately 14V, we chose to use a buck/boost topology for our power stage.

Knowing the Atmega168 clock frequency limit of 8MHz (using the internal oscillator), a maximum switching frequency of 40kHz can be achieved without losing too much on resolution. Our design uses Timer1 to count up to 100, meaning that our resolution in duty cycle is of 1 timer count per 1 duty cycle percent point.

$$f_{sw} = \frac{f_{clk}}{2 \cdot N \cdot TOP} = \frac{8MHz}{2 \cdot 1 \cdot 100} = 40kHz$$

Knowing the switching frequency, and the input and output voltage ranges, we designed our power stage and analyzed the control-to-output-voltage transfer function of the buck/boost topology. Given that our design goal is to trickle-charge the battery, we chose to run the circuit in discontinuous conduction mode. One of the added benefits of operating in DCM is the size of the magnetics can be reduced significantly, at the expense of increasing ripple currents and voltages. With this in mind, a 22μH inductor was selected. This means that the critical current that is needed to enter CCM must be:

$$I_o(crit) \geq \frac{-V_o T_s}{2L} \frac{V_{i(max)}}{V_o - V_{i(max)}} = 2.96\,A$$

The battery however, needs a tight control on the voltage at its terminals. For that reason, a big output capacitor is necessary. The following formula was used to calculate the size of the output capacitance given an output ripple of 50mV:

$$C_o \geq \frac{I_{o(max)} D_{(max)}}{f_s \Delta V_o} = \frac{0.3 \cdot 0.9}{40kHz \cdot 0.050} = 135\mu F$$

given this constraint, we decided to use a 300μF capacitor at the output stage.

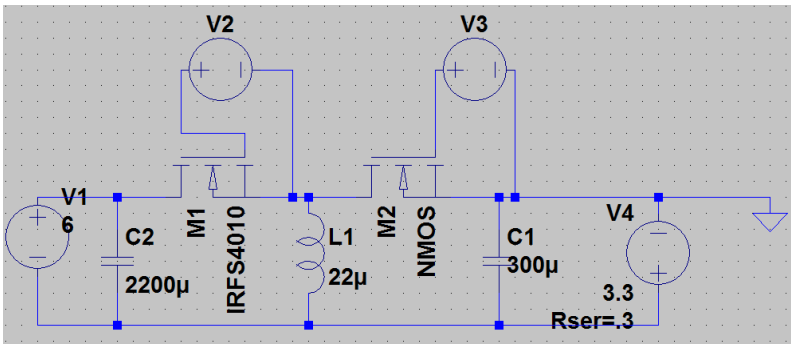
Since we explicitly limit the charging current to less than 300mA, we know we are well into the DCM region of operation. Under steady-state, the following formula relates the output voltage to duty cycle and input voltage:

$$V_o = \frac{-D}{\sqrt{K}} V_i, where\ K = \frac{2L}{RT_s}$$

In order to build a control loop around our system however, we were interested in a transfer function that related small variations in duty cycle to small variations in output voltage. We found the control-to-output-voltage transfer function for the Buck/Boost under DCM operation to be:

$$\frac{\widehat{v_o}}{\widehat{d}} = -V_i\sqrt{\frac{RT_{sw}}{2L}}\frac{\omega_p}{\omega_p + s}, where\ \omega_p = \frac{2}{RC}$$

This transfer function was used to come up with a control scheme which is detailed under the “Control” section. A subtle detail surrounding the Buck/Boost topology is which node to call the reference node. Our microcontroller must perform measurements with respect to some point on the circuit, and drive the switches with respect to that point as well. For ease of measurement (avoiding negative voltages), and ease of gate-driving, we chose the negative terminal of the battery to be GND. The final power path schematic (excluding the rectifier) can be found below:



This circuit was validated using LTSpice to ensure that we were in the DCM region of operation and that the steady-state voltage response was in accordance with the formula stated above. More information on the simulation details can be found under the section “Key Results”.

The microcontroller is powered straight from the battery, which means that measurements for the output voltage were positive rather than negative. The rectified input voltage was measured differentially. Additionally, the current sense resistor was installed between the source of the low-side transistor and the negative terminal of the battery, giving us a distinction between Vout and Vbat. More information of how this helped us regulate current at the output can be found under the “Control” section.

Analog circuitry used for ADC measurements require a stable supply. Due to time constraints, we decided to use an external DC supply to power these circuits. We decide to use two differential amplifiers, one to measure the output current through a sense resistor, and the other to measure the input voltage. The other two ADC measurements (voltage directly at the DC/DC converter output and battery voltage) can be found using simple voltage dividers. These two voltage will differ by the voltage drop across the sense resistor used to measure output current, which gives us a way to regulate the output current. Again, additional details on this particular choice can be found under the “Control” section.

Microcontroller and software:

Based on what we learned in lab, we decided to move further and implement our charging controller on a different microcontroller. We were able to set up and build the Atmega168V board from scratch, initialize its factory-state chip, program through ISP, and create our own library to bring up ADC, PWM, UART, filtering and calibration. We chose Atmega168V for its relatively low power consumption of 250µA (at 1MHz) and its wide range of operating voltage (1.8V - 5.5V).

For PWM, phase correct PWM with two outputs was implemented with software programmed dead time to prevent shoot-through conditions. From N-channel MOSFET datasheet (IRLI540N), tOff (39ns) + tRise (81ns) + (safety margin) gives dead time of 125ns.

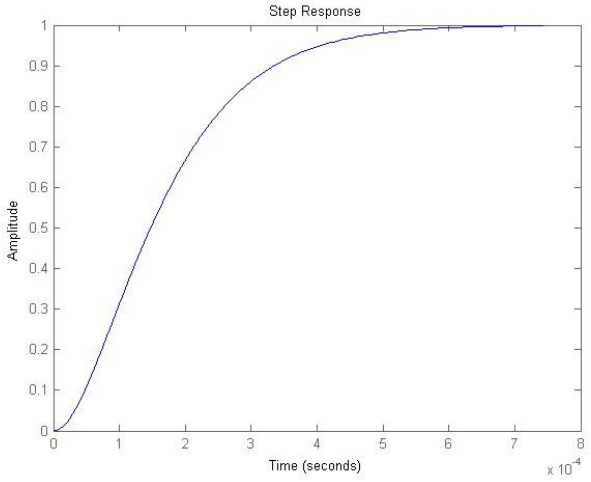
Given 8MHz clock frequency and prescaler of 1 used for Timer1, each timer tick is 125ns. This means the required dead time can be achieved by having a one tick difference between the two PWM registers. Phase correct PWM allows the timer to set the output when the timer reaches the register value counting up, and clear it when counting down. Having two separate registers with opposite actions (one clears counting up, and the other sets when counting up), means we can drive two separate switches with any programmable dead time (in multiples of 125ns). Additionally, the output pulse is always centered about TOP (the maximum value that the counter counts to). Again, we chose TOP to be 100, which sets our switching frequency to 40 kHz and gives us a resolution of 1 timer count per 1 duty cycle percent point.

For power monitoring and debugging, a polling ADC function was created to sequentially start conversion and store 4 channel results to a designated array. The three channels measure input voltage, output voltage, battery voltage and output current. A subtle interrupt flag error eluded us for some time, but was solved after a detailed debug session. The filtering and calibration library of the original lab were adapted to Atmega168V, with UART and PWM libraries implemented from scratch to support receiving and transmitting debugging messages using interrupt method.

The main code accepts commands via the UART port to enter calibration stage or manual duty-cycle stage. The calibration stage allows us to calibrate the ADC and record the values in EEPROM to retain after a power-off cycle. The manual duty-cycle stage allows us to manually change the duty cycle in order to ensure the power path is behaving as expected under steady-state conditions. Under normal operation, the main code polls the ADC inputs every millisecond and calculates the required duty cycle to ensure correct charging of the battery (controls the buck/boost to maintain a constant current at the output).

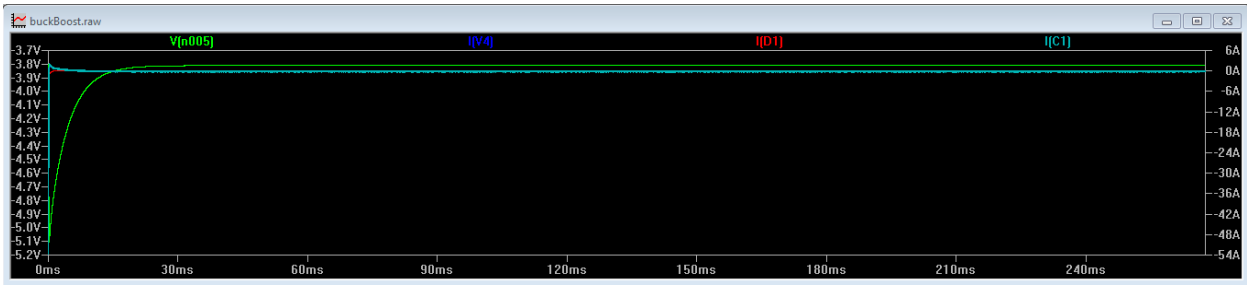
Control:

To achieve near constant current charging of 300mA, we inserted a 0.01Ω shunt resistor between the output of the buck/boost and the battery. This way, we can calculate the desired voltage on the converter's side based on the shifting battery voltage over the charging cycle (the desired converter output voltage will always be one voltage-drop across the shunt resistor above the battery voltage. The plant described by the small-signal transfer function relating output voltage to duty cycle was simulated under MATLAB. For stabilizing the plant with varying buck/boost input voltage from 2V to 12V , a single integrator approach proved most robust. Manual tuning using rltools was used to stabilize the system and find the appropriate integral gain. The wind up was prevented by adding a clamping algorithm to the integral action.

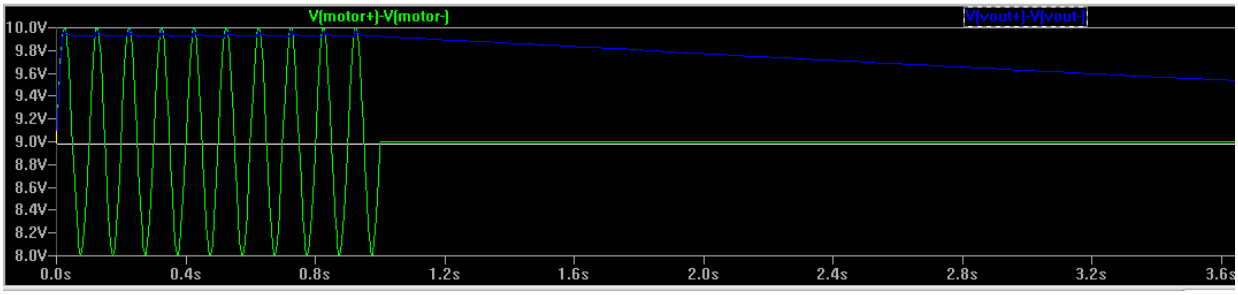


Key results:

The following picture shows simulation results for the buck/boost converter shown in the schematic above. This simulation allowed us to validate that we were in DCM and that the output current was below the expected limit. The spikes shown at the beginning of the simulation are suppressed by our feedback loop.



The following picture shows the simulation of our active rectifier circuit with 1Vpp source oscillating at 10Hz around 9V. We originally wanted to opt away from the standard full bridge diode rectifier because we wanted to achieve better efficiency from our rectifier to supply our buck boost. Since our simulation requires a steady supply voltage at the TL084 quad, we simulated with a direct 10V and GND supply. The simulation did give us the correct rectified output as shown below (~10V DC). In this picture, we do see some ripple varying between 9.93 and 9.95 volts, but mostly a DC voltage source.



When we built up active rectifier circuit on a breadboard, we noticed a 1.2 voltage drop from the input, which did not match our expected behavior. This circuit also introduced some unexpected load on our motor which made it harder to turn and generate voltage. Due to our time constraints, we ended up hacking our flashlight to find a DB107 rectifier chip. We added a 2200uF capacitor to the rectified output to help stabilize the voltage. We hooked our rectifier circuit with the motor to the oscilloscope and saw expected behavior. The maximum DC voltage seen at the output is 16V with an average of 10V.

Next steps:

1. Powering the microcontroller and analog circuits more efficiently (all powered from the same supply, probably the battery itself)
2. Achieving synchronous rectification using a FET bridge rather than a diode bridge
3. Characterize the power path to account for the efficiency losses and see how each source of loss could be addressed