EE152 Final Report

Harry Johnson

Summary:

For my final project, I chose to begin implementation of a prototype DC link input for the Stanford Littlebox Converter. While I was not able to accomplish what I originally intended, I hope that through my efforts I have brought the project several steps closer to physical reality. The DC-link input is implemented as a continuous-mode boost converter running at 1Mhz, with 400-450V input voltage and 450V-900V output voltage. The output voltage range is large because the capacitor charges directly and without load, since the cap is drained in a separate section of the 120Hz cycle than it is charged.

Statement of Work:

I began by writing some MATLAB code to generate waveforms for the DC-link input current setpoint and simulate its behavior relative to the rest of the system. I wanted to abstract away as much as I could to make my life easier once I got to circuit-level simulation. I generated a circuit model of an average current mode controlled boost converter in LTSpice, first using ideal switches and diodes, and fed the waveforms generated by the MATLAB code in as setpoints. After I verified that the average current mode analog controller I implemented was at least stable enough for simulation purposes (I intend to implement the controller in digital when it's built for real), I started adding in non-idealities, specifically focusing on switching losses.

Matlab:

This code models the converter on a cycle-by-cycle basis. The critical system-level variables passed into this program are RMS output power (up to 2KW), DC input voltage and resistance (450V, 10 ohms), and RMS output voltage (240V). To simplify system behavior, I chose to only model unity power factor output. Using these variables, I generated the desired output voltage and current waveforms. For any given RMS output power, the program solves for the average DC input current that will produce that power level (since input current also changes the effective input voltage), after this point the input voltage and current are assumed to have to be constant. As such, the main buck converter is modeled as input-current limited to said DC input current level. Any time the buck ratio and output current results in a buck input current that is less than the DC average current, the difference is used as the setpoint of the DC-link input boost converter. Any time the buck on its own cannot supply the requested output current, the difference is fed as the output-current setpoint to the DC-link output buck. From this, the storage capacitor output current waveforms are also generated.

LTSpice:

To start with, I modeled the power supply and surrounding littlebox components in simplified form, with the addition of a 10mR shunt resistor on the low side. I exported the buck input current waveform as a CSV, then imported it into the schematic as a piece-wise linear current source on the boost side of the shunt resistor. A second piece-wise linear current

source was used to simulate the current drained from the storage capacitor in the high-current sections of the 120Hz waveform.

A control loop was set up to regulate the average total DC supply current current (buck + boost) to 5A (I was modelling full-load). I used an analog-implemented average current mode control loop and a simple (and standard) pulse-width modulator based on a sawtooth wave generator. To start with, I used ideal switching components, namely voltage controlled switch models and no-capacitance diodes. Inductor and storage capacitor values (100uH and 22uF, respectively) were chosen based on professor Dally's straw man design, but seemed to be verified, with the possible exception that the inductor value may need to be increased if lower ripple current is desired (currently ~1.6A pk-pk at peak load and boost ratio). In particular, the storage capacitor is capable of storing 5J of energy in a swing that starts at over 450V, which allows all of the needed energy to be stored in the capacitor using a boost-only circuit.

Once I had verified that the system worked properly using ideal elements, I began introducing non-idealities. Because the switching losses are one of the more inevitable sources of losses when compared to inductor or capacitor losses, I chose to simulate them first. To do this I downloaded the Cree-provided LTSpice models of the selected FET (a Cree C2M0280120D) and diode (Cree C4D05120A). To simulate a gate driver, the last gate that drives the FET is modeled as a 20Voh, -5Vol 1ps rise and fall time gate in series with a 2.5 ohm resistor. I also added series inductors on the drain and source to more correctly model packaging effects (as recommended by Cree in their information packet included with the LTSpice libraries).

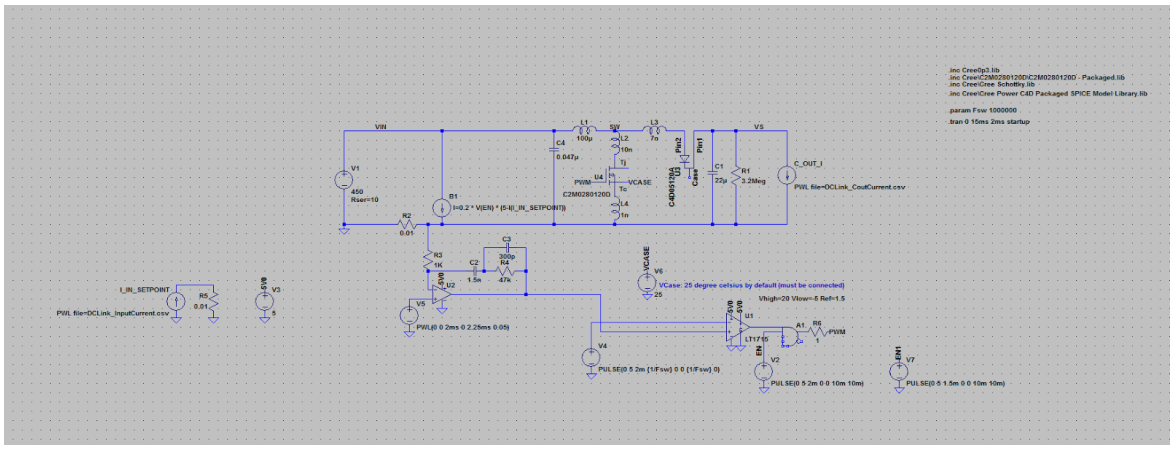
Using LTspice's built-in integration tool, I integrated the input power and output power waveforms over the amount of time that the boost regulator is active (not over the whole 120Hz cycle), then used that to find the average input and output power, directly using those to calculate average efficiency.

Results

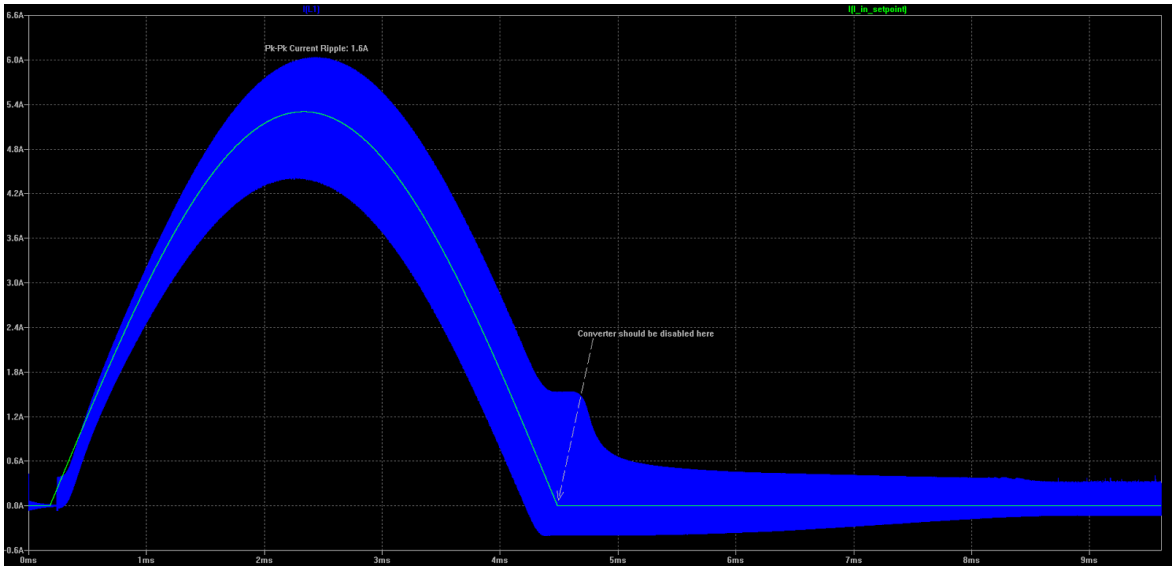
Input Power	1.3195 KW
Output Power	1.2888 KW
Switching Power Loss	30.7 Watts
Efficiency (only counting switching loss)	97.6%

Future Work:

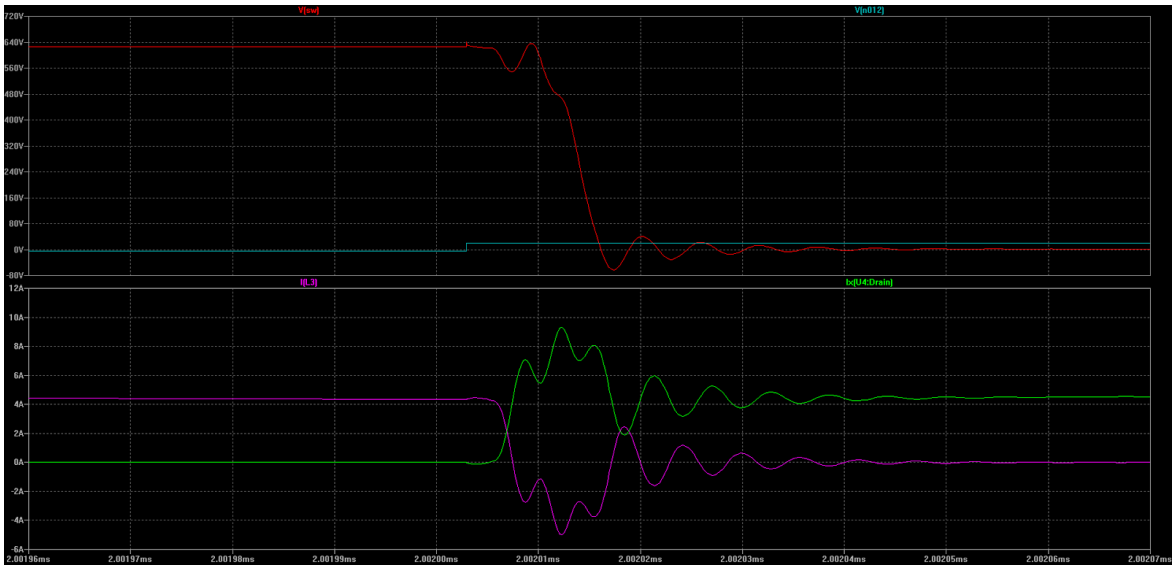
I would like to continue future work by building further non-idealities into the spice modeling. In particular, I would like to model the behavior of the storage capacitor, given that the real "capacitor" will in fact be many smaller capacitors in a series/parallel configuration. After that, it would be good to actually do schematic capture, layout, and bringup for a first-pass converter using off the shelf magnetics, followed by a second rev using custom optimized magnetics.



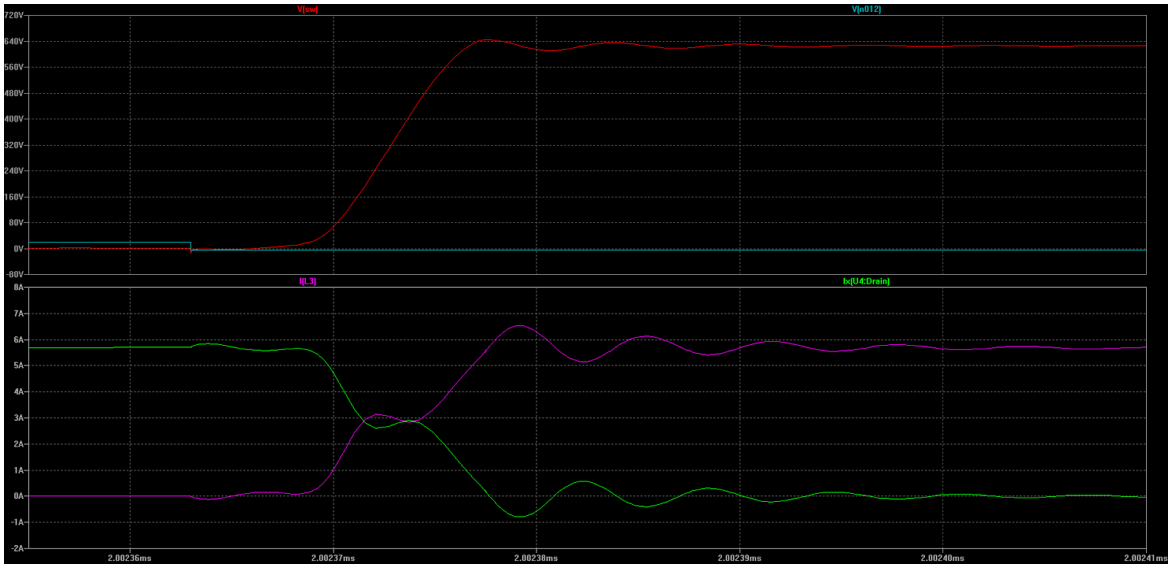
Final LTSpice Schematic



Boost Input Current Setpoint & Inductor Current vs time



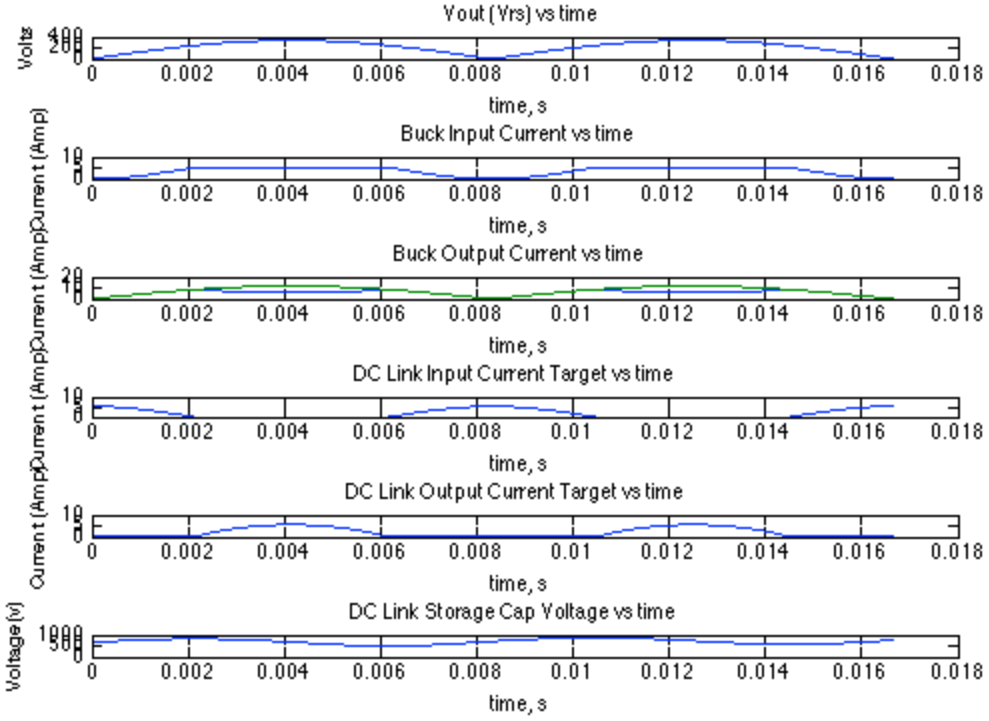
FET current, Diode current, and Switch Node voltage vs time for FET turn-on.



Same as above, but for FET turn-off.

```
DUT_IN_V = 450; %450Vin
DUT_IN_R = 10; %in series with a 10 ohm resistor.

Csc = 22E-6; %Storage Cap is 22uF

RMS_AC_P = 2E3; %Set output power (RMS AC)
LB_DC_IN_P = RMS_AC_P / 0.95;  %set input power based on 95% efficiency.

%Find correct input current to give the proper DC input power after drop
%across resistor.
syms x;
LB_DC_IN_I = min(eval(solve('LB_DC_IN_P = (x*(DUT_IN_V - x*DUT_IN_R))', 'x'))); %for

LB_DC_IN_V = 450 - LB_DC_IN_I * DUT_IN_R; %Littlebox DC input voltage.


AC_f = 60; %60Hz output
RMS_AC_V = 240; %120VAC RMS output.
PK_AC_V = RMS_AC_V * sqrt(2); %Peak DC voltage based on RMS AC voltage.

RMS_AC_I = RMS_AC_P / RMS_AC_V; %RMS Current output.
PK_AC_I =  RMS_AC_I * sqrt(2); %Peak current.

AC_I_PS = 0; %degrees phase shift (other than 0 not supported right now).

ti = 0/AC_f;
tf = 1/AC_f;
dt = 1E-6; %1/1Mhz
t = ti:dt:tf;
tlen = size(t,2);

AC_Vout = abs(PK_AC_V * sin(2*pi*AC_f*t)); %VRs(t)
AC_Iout = abs(PK_AC_I * sin(2*pi*AC_f*t - AC_I_PS * pi / 180)); %IRS(t)
AC_Pout = AC_Vout .* AC_Iout; %Pout(t)

BUCK_IN_I = min(LB_DC_IN_I, (AC_Pout) / LB_DC_IN_V);
BUCK_OUT_I = BUCK_IN_I .* LB_DC_IN_V ./ (AC_Vout+0.000001); %buck output current, li

DCL_IN_I = max(0, LB_DC_IN_I - BUCK_IN_I); %assuming DCL input can take up all the s
DCL_OUT_I = AC_Iout - BUCK_OUT_I; %assuming DCL output can take up all the slack.
DCL_I = DCL_IN_I - DCL_OUT_I;

VSC = zeros(1,tlen);
VSC(1) = 650; %say we got to this point at some time before. Still working on contro

%Time Domain simulation for VSC. There must be a more efficient solution to
%this...
DCL_C_I_IN = zeros(1,size(t,2));
DCL_C_I_OUT = zeros(1,size(t,2));
DCL_C_I = zeros(1,size(t,2));
for i = 2:tlen
    DCL_C_I_IN(i-1) = (DCL_IN_I(i-1)*LB_DC_IN_V / VSC(i-1)); %input current to cap h
    DCL_C_I_OUT(i-1) = (DCL_OUT_I(i-1) * AC_Vout(i-1) / VSC(i-1)); %Output current f
    DCL_C_I(i-1) = DCL_C_I_IN(i-1) - DCL_C_I_OUT(i-1);
```

```
    VSC(i) = VSC(i-1) + (DCL_C_I(i-1)/Csc)*dt;
end


subplot(6,1,1);
plot(t, AC_Vout);
xlabel('time, s');
ylabel('Volts');
title('Vout (Vrs) vs time');

subplot(6,1,2);
plot(t, BUCK_IN_I);
xlabel('time, s');
ylabel('Current (Amp)');
title('Buck Input Current vs time');

subplot(6,1,3);
plot(t, BUCK_OUT_I, t, AC_Iout);
xlabel('time, s');
ylabel('Current (Amp)');
title('Buck Output Current vs time');


subplot(6,1,4);
plot(t, DCL_IN_I);
xlabel('time, s');
ylabel('Current (Amp)');
title('DC Link Input Current Target vs time');

subplot(6,1,5);
plot(t, DCL_OUT_I);
xlabel('time, s');
ylabel('Current (Amp)');
title('DC Link Output Current Target vs time');

subplot(6,1,6);
plot(t, VSC);
xlabel('time, s');
ylabel('Voltage(v)');
title('DC Link Storage Cap Voltage vs time');
```



Published with MATLAB® 7.12