Single-Inductor, Multiple-Output Buck Converter with Parallel Source Transient Recovery

by

Charles Jackson King III

S.B. E.E., M.I.T., 2007

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

February 2009

Author_____
Department of Electrical Engineering and Computer Science
February 3, 2009

Certified by_____
Ken Mok
Senior Power Systems Engineer, Qualcomm
VI-A Company Thesis Supervisor

Certified by_____
David J. Perreault
Associate Professor
M.I.T. Thesis Supervisor

Accepted by_____
Arthur C. Smith
Professor of Electrical Engineering
Chairman, Department Committee on Graduate Theses

Single-Inductor, Multiple-Output Buck Converter with Parallel Source Transient Recovery

by

Charles Jackson King III

Submitted to the
Department of Electrical Engineering and Computer Science

February 3, 2009

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

To address the need for multiple regulated voltage supplies in electronic devices, this thesis presents a modeling and design study of a single-inductor, multiple-output (SIMO) DC-DC buck converter with parallel source transient recovery. This converter would provide substantial cost and space savings over traditional options for producing multiple supply voltages. Operating in pseudo-continuous conduction mode (PCCM), it can supply heavy loads while not suffering from cross-regulation problems. The parallel current source circuitry at each output will greatly dampen any voltage spikes that may occur due to sudden load changes, thus improving transient performance. While the entire converter could not be nicely simulated as envisioned, the initial steps and accomplishments outlined in this thesis show definite promise. The proposed converter certainly merits further research, as the problems encountered here most likely stem from implementation and control issues rather than fundamental flaws in the idea.

VI-A Company Supervisor: Ken Mok
Title: Senior Power Systems Engineer, Qualcomm

Thesis Supervisor: David J. Perreault
Title: Associate Professor

Acknowledgments

I would like to thank my parents, Mary and Charley, and my little sister, Kristen, for all their love and support during my years at MIT. Their encouragement and comforting words helped preserve my sanity during stressful periods and bouts of homesickness, enabling me to appreciate my time here and create memories that will last a lifetime. They truly made me the man I am today, and I am incredibly thankful to have such a loving home to which I can escape.

I would also like to thank all my friends, both in Georgia and at MIT, for all the wonderful memories and much-needed distractions from schoolwork. Particularly Amanda Doane, who has shared my experiences, both good and bad, over countless hours on the phone, and whose loving heart compares only to my mother's in its warmth and compassion. My friends at school have provided a family away from home since almost the first moment I walked on campus, and they are truly some of the most remarkable people I know. I could not have made it through my years here without their support and the many laughs we've shared.

Finally, I would like to thank Ken Mok for all his guidance and patience during my thesis work at Qualcomm, and Prof. David Perreault for proofreading my thesis and for inspiring my entrance into power electronics through one of my favorite classes at MIT, 6.334.

Contents

List of Figures and Tables

Chapter 1

Introduction

1.1 Motivation

As battery-operated portable devices such as mobile phones and music players have gained popularity, the extension of battery life – and thus the minimization of power consumption – has become one of the most important design criteria for these devices. In many of the devices, different circuit modules require different supply voltages, and providing multiple regulated voltage supplies can greatly reduce the power consumption of such systems. As outlined in [9], one conventional DC-DC converter implementation for providing N output voltages would be to simply construct N independent converters, while another would be to use a transformer with N secondary windings to distribute energy into the different outputs. However, the first method of N independent converters can violate size and cost constraints, as it requires many power devices and controllers. Indeed, both methods require N inductors or transformer windings, which can be quite bulky and costly since inductors and transformers are typically the largest off-chip components. The second method of using a transformer also does not allow the individual outputs to be independently controlled. Often only one output is regulated through tight closed-loop control, while the other outputs are generated through coupling of the secondary windings. Serious cross-regulation problems can also occur as a result of leakage inductance and cross-coupling among windings.

A DC-DC switching converter that uses only one inductor to produce multiple output voltages is thus highly desirable, since each individual output can be independently regulated

while also minimizing system cost and size. This thesis presents a system-level modeling and design study of a single-inductor, multiple-output (SIMO) buck converter with parallel source current injection at each output, which could feature better cross-regulation suppression and transient voltage recovery than previously proposed SIMO converters. The topology and timing diagram for such a converter with two outputs is shown in Figure 1.

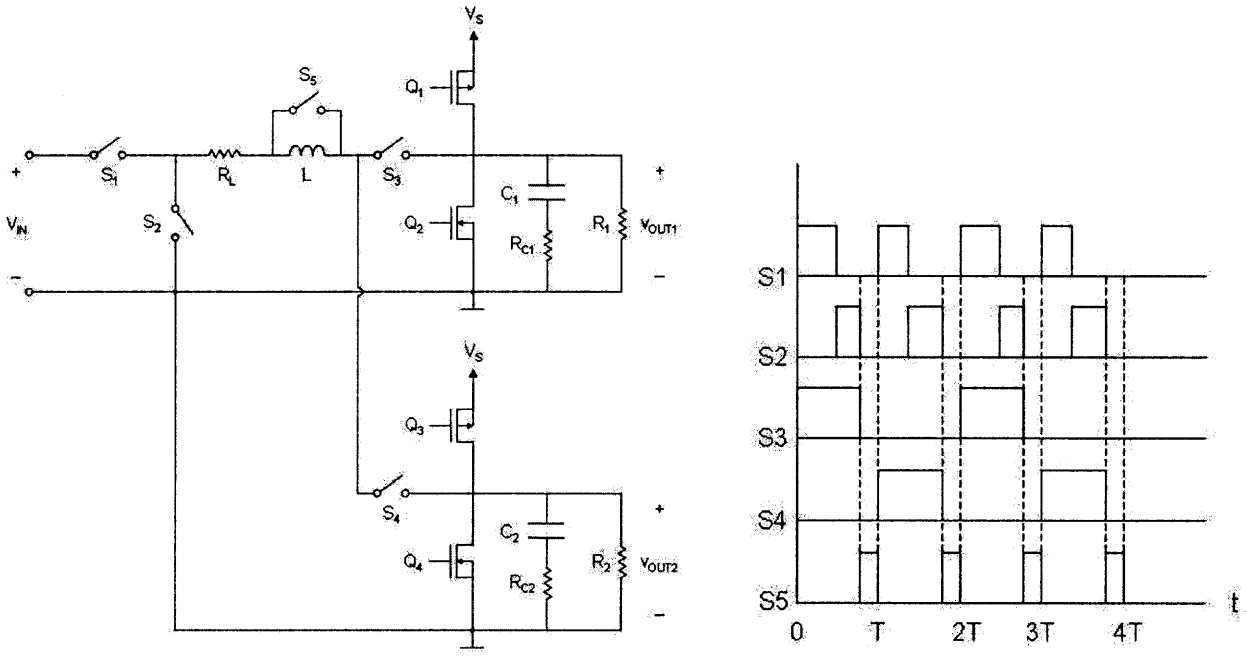


Figure 1: SIMO Buck Converter Topology and Timing Diagram

1.2 Pseudo-Continuous Conduction Mode (PCCM)

This converter will ultimately operate in pseudo-continuous conduction mode (PCCM), meaning the inductor current will fall to, and stay constant at, a DC current level at the end of each switching cycle. Using a time-multiplexing control strategy similar to those in [6] and [9], switches S_3 and S_4 in Figure 1 will channel the inductor current into the appropriate output. When S_3 is closed, a feedback loop will determine the duty cycles at which switches S_1 and S_2 must be driven in order to keep v_{OUT1} regulated at the desired level. When the inductor current is flowing through S_2 and falls to a certain level, S_2 and S_3 open and S_5 closes, thereby keeping the

inductor current constant at that level by providing a freewheeling path and holding the voltage across the inductor to zero (since $v_L = L\frac{di_L}{dt}$). Analogous operation occurs when S_4 is closed, except the inductor current flows into Output 2.

PCCM combines the advantages of continuous conduction mode (CCM) and discontinuous conduction mode (DCM), while suffering from neither of their major disadvantages. In DCM, the inductor current reaches and stays constant at zero at the end of each switching cycle. Cross-regulation is thus not a problem since each output is isolated, but the converter cannot support heavy loads. A smaller inductor could be used for heavier loads, but this would lead to a larger peak inductor current since the inductor current's slope $\frac{di_L}{dt} = \frac{v_L}{L}$. A larger current ripple would result, as well, which would lead to a larger voltage ripple since $\tilde{v}_{out} = \tilde{i}_L R_C$ (where $\tilde{v}_{out}$ and $\tilde{i}_L$ are the AC components of the output voltage and the inductor current, respectively). A larger filter capacitor could be used to mitigate this large ripple, but this would lead to a slower transient response. In CCM, the converter can support heavy loads since the inductor current always stays above zero and is not required to fall to any particular DC level. However, since the current is nonzero when switching between outputs, the outputs are not isolated from each other and serious cross-regulation problems can be introduced. The outputs' duty cycles are interdependent, and a sudden load change at one output may produce a change in the other output's voltage. If both loads change at the same time, the converter may become unstable.

In PCCM, however, the inductor current reaches and stays constant at a nonzero current level (determined by the load requirement) at the end of each switching cycle, with both loads being disconnected from the inductor during this constant-current stage. For unbalanced loads,

each output can even have a different DC current level, as shown in Figure 2. PCCM can therefore support much heavier loads than DCM and should be able to solve the cross-regulation problems of CCM. The current ripple can also be reduced, since a larger inductor can be used in PCCM converters.

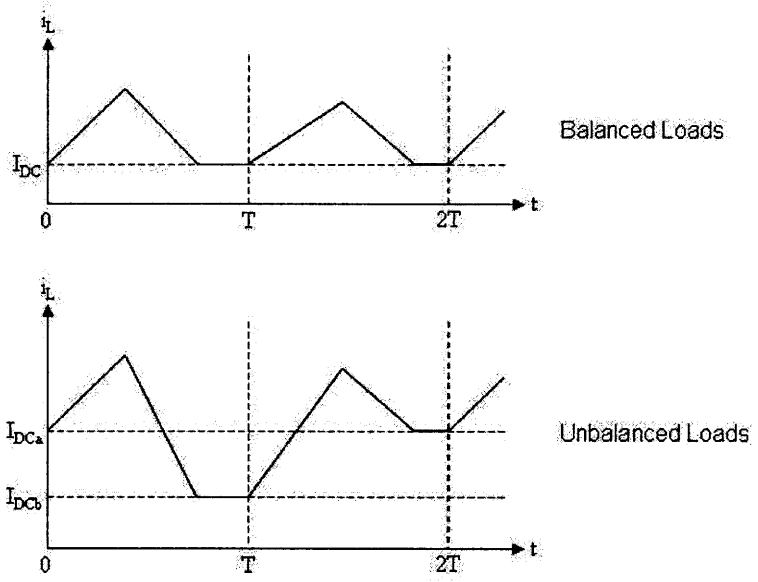


Figure 2: Inductor Current in PCCM

The authors in [7] and [8] tested the PCCM operational concept in both single- and dual-output boost converters and verified its aforementioned advantages over CCM and DCM operation, while also maintaining small output voltage ripples and efficiencies of approximately 89%. The converter presented in this thesis uses a buck topology, not boost, but [7] and [8] nonetheless illustrate the promise of PCCM operation.

1.3 Parallel Source Transient Recovery

Transient performance has become a significant concern in supplying power to low-voltage digital systems with dynamic loads because of the fast, high-current load steps in such

systems. In DC-DC power converters, these transients are primarily caused by the energy stored in the inductor. A surplus of energy could cause a voltage overshoot upon being transferred to the load and output capacitor, if the load current requirement decreases. Conversely, a shortage of energy could lead to a voltage undershoot, if the load suddenly demands more current. A common solution to this transient problem is to simply increase the output capacitance, either at the power stage or with a passive output filter. However, this solution decreases the system's closed-loop bandwidth, and is shown by [12] to produce voltage spikes because of the resonant loops between the parasitic components of the filter capacitor and the various interconnections around the power supply.

Active transient suppression techniques are therefore needed and have been examined a great deal in recent research. Since all energy must pass through the inductor and exit through the capacitor and load in DC-DC converters, thus creating a performance bottleneck, many proposed solutions involve bypassing these energy storage elements using additional conduction paths. These paths need only handle transient power, and do not increase the number or size of energy storage elements. An augmented single-output buck converter topology is suggested in [11], in which the traditional buck is modified with alternate resistive current routes around both the inductor and the capacitor. These routes can be connected or disconnected via switches. The transient performance improvement with this augmentation is measured for several levels of knowledge about the magnitude and timing of load changes. For the situation in which both the magnitude and the timing are unknown, the authors use hysteretic voltage thresholds to trigger the switching of the alternate current paths, which is similar to the control method proposed in this thesis. The buck converter controlled in this way in [11] had 45% and 31% less peak undershoot and overshoot, respectively, compared to a normal PI controlled buck for load steps

of magnitude 50Ω. The rise time was also much better. The greater the knowledge that is available about the magnitude and timing of the load changes, the better the transient suppression. In practical applications, the magnitude of load changes could be known from prior measurements, while the timing could be indicated by clock and data signals.

An active clamp scheme in [1] proposes the use of auxiliary capacitors at the output of a voltage regulator, which would be kept charged at a higher voltage and could deliver in a single-shot manner the extra charge needed by the load during step-up transients. Conversely, the clamp could sink excess charge during step-down transients. Different RC networks with different time constants could be used together in the clamp circuit to closely match the current needed by the load during a transient. Though auxiliary capacitors would be used in the active clamp circuit, this solution would actually result in a net decrease in energy storage elements, as the number and size of decoupling and filter capacitors at the voltage regulator's output would be significantly reduced. A different active clamp design is presented in [13], which is based upon the same fundamental idea as the parallel source transient recovery scheme presented in this thesis. This clamp circuit works in parallel with the output of a switching regulator, behaving as a linear regulator and either sourcing or sinking current for the load during transients. The authors in [13] designed and fabricated an integrated circuit that used this active clamp, which yielded promising results in suppressing transient voltage spikes. This active clamp allows the use of a smaller output capacitor for the switching regulator, without simultaneously requiring the regulator's inductor and switching frequency to be designed for extremely fast transient response.

Similar to the active clamp design in [13], the parallel current sources Q_1, Q_2, Q_3, and Q_4 on the outputs in Figure 1 help minimize transient effects during sudden load changes by acting

as linear regulators and sourcing or sinking extra current. When an output voltage goes above a certain threshold voltage, the bottom source turns on and provides an alternate path for some of the inductor current. Since the current flowing into the load is decreased, the output voltage falls back to its steady-state level more quickly. The source does not turn back off until the output voltage has dropped below a threshold that is lower than the one used to turn it on, as this hysteretic effect prevents the voltage from simply oscillating back and forth across the original threshold. Analogous behavior occurs when an output voltage drops below a certain threshold voltage – the top source turns on and injects additional current into the load.

The parallel sources can thus greatly dampen voltage spikes caused by sudden load changes. These sources can turn on and off regardless of the states of S_3 and S_4 – i.e. regardless of whether or not a particular output is connected to the inductor at the time. And since they only turn on for short periods of time during transients, the sources will not dissipate much power and will therefore not have too harmful an effect on the converter's efficiency, so long as transients are infrequent.

The remainder of this thesis will detail the progress I have made towards achieving the ultimate goal of a SIMO buck converter operating in PCCM with parallel source transient recovery. Chapter 2 demonstrates a SIMO buck converter operating in DCM, thus illustrating our modeling and simulation techniques while exposing the limitations of DCM operation. Chapter 3 addresses these limitations by exploring PCCM operation, highlighting the progress made and problems faced thus far. Chapter 4 discusses the implementation of the parallel source idea in a single-output buck converter, demonstrating its potential usefulness in a multiple-output converter. Finally, Chapter 5 discusses my conclusions and future work to be done.

Chapter 2

SIMO Buck Converter Operating in DCM

2.1 Mathematical Model

The first step towards the ultimate converter implementation was to design and simulate a SIMO buck converter operating in DCM, without the parallel sources on the outputs. The circuit topology and timing diagram are shown in Figure 3 and include resistors that model the parasitic resistances of the inductor and capacitors.

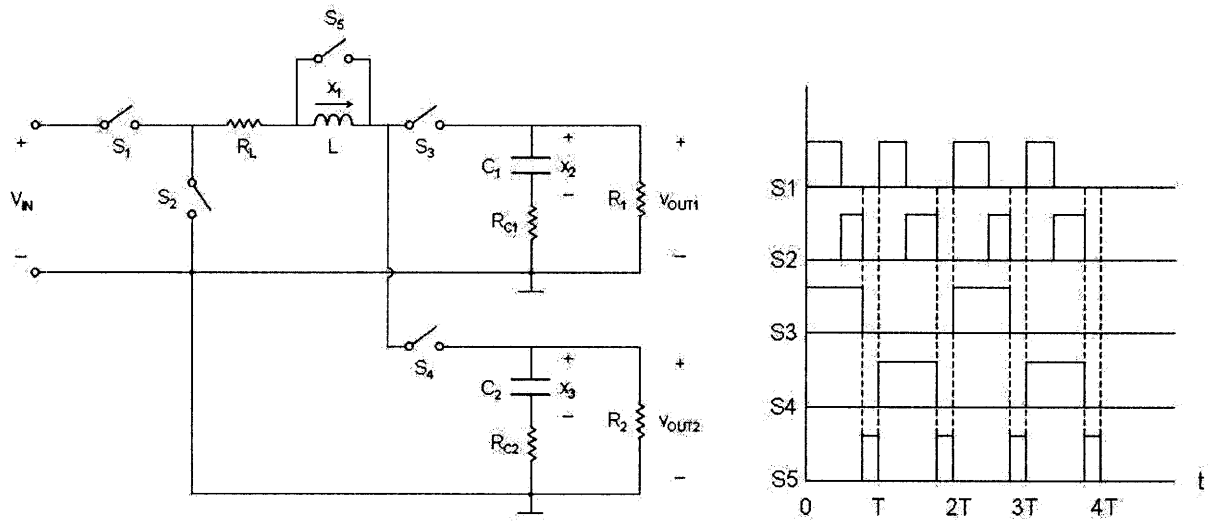


Figure 3: SIMO Buck Converter Topology and Timing Diagram

The converter was simulated using state space methods by tracking and updating three circuit state variables (inductor current, output capacitor 1 voltage, and output capacitor 2 voltage) and two outputs (output voltage 1 and output voltage 2):

3 state variables: x_1 = inductor current
x_2 = output capacitor 1 voltage
x_3 = output capacitor 2 voltage

2 outputs: y_1 = output voltage 1
y_2 = output voltage 2

For each different circuit state (i.e. each different combination of switch states), we define state matrices A, B, and C that are used to determine the values of the state variables and output voltages for the next cycle, given a control input *u*. We can then derive a discrete time model that can be used to simulate the converter's operation:

Define state matrices **A**, **B**, and **C**: $\mathbf{dx/dt = Ax + Bu}$
$\mathbf{y = Cx}$

Model in discrete time (for simulation):
$\mathbf{x}(n+1) = \mathbf{\Phi x}(n) + \mathbf{\Gamma u}(n)$
$\mathbf{y}(n+1) = \mathbf{C x}(n+1)$
where $\mathbf{\Phi} = e^{AT}$
$\mathbf{\Gamma = B^*}(e^{AT} - e^{A^*0})/\mathbf{A}$

The state matrices used for simulation are shown on the next several pages, one for each equivalent circuit state in the converter's operation cycle. A time-multiplexing control strategy is employed, which is similar to those used in [6] and [9]. In State 1, switches S_1 and S_3 are closed, connecting V_{IN} to the circuit and causing the inductor current to ramp up and flow into Output 1. In State 2, S_1 opens and S_2 closes, and the inductor current ramps down as it freewheels through Output 1 while energy is dissipated in the resistors. In State 3, switches S_1, S_2, S_3, and S_4 open as the inductor current reaches zero and switch S_5 closes, thus keeping the current constant at zero as defined by its DCM operation. Since state variable x_1 (inductor current) is not changing during State 3, the state matrices are simplified and have reduced dimensions. States 4, 5, and 6 are analogous to the first three, but applied to Output 2 instead of Output 1.

State 1: Switches S_1 and S_3 Closed

$$A=\begin{bmatrix} \dfrac{R_1R_{C1}+R_1R_L+R_LR_{C1}}{L(R_1+R_{C1})} & \dfrac{R_1}{L(R_1+R_{C1})} & 0 \\ \dfrac{R_1}{C_1(R_1+R_{C1})} & \dfrac{1}{C_1(R_1+R_{C1})} & 0 \\ 0 & 0 & \dfrac{1}{C_2(R_2+R_{C2})} \end{bmatrix}$$

$$B=\begin{bmatrix} 1/L \\ 0 \\ 0 \end{bmatrix} \qquad C=\begin{bmatrix} \dfrac{R_1R_{C1}}{R_1+R_{C1}} & \dfrac{R_1}{R_1+R_{C1}} & 0 \\ 0 & 0 & \dfrac{R_2}{R_2+R_{C2}} \end{bmatrix}$$

State 2: Switches S_2 and S_3 Closed

$$A=\begin{bmatrix} \dfrac{R_1R_{C1}+R_1R_L+R_LR_{C1}}{L(R_1+R_{C1})} & \dfrac{R_1}{L(R_1+R_{C1})} & 0 \\ \dfrac{R_1}{C_1(R_1+R_{C1})} & \dfrac{1}{C_1(R_1+R_{C1})} & 0 \\ 0 & 0 & \dfrac{1}{C_2(R_2+R_{C2})} \end{bmatrix}$$

$$B=\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad C=\begin{bmatrix} \dfrac{R_1R_{C1}}{R_1+R_{C1}} & \dfrac{R_1}{R_1+R_{C1}} & 0 \\ 0 & 0 & \dfrac{R_2}{R_2+R_{C2}} \end{bmatrix}$$

State 3: Switch S_5 Closed ($I_L = 0$)

$$A = \begin{bmatrix} -\dfrac{1}{C_1(R_1 + R_{C1})} & 0 \\ 0 & -\dfrac{1}{C_2(R_2 + R_{C2})} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad C = \begin{bmatrix} \dfrac{R_1}{R_1 + R_{C1}} & 0 \\ 0 & \dfrac{R_2}{R_2 + R_{C2}} \end{bmatrix}$$

State 4: Switches S_1 and S_4 Closed

$$A = \begin{bmatrix} -\dfrac{R_2R_{C2} + R_2R_L + R_LR_{C2}}{L(R_2 + R_{C2})} & 0 & -\dfrac{R_2}{L(R_2 + R_{C2})} \\ 0 & -\dfrac{1}{C_1(R_1 + R_{C1})} & 0 \\ \dfrac{R_2}{C_2(R_2 + R_{C2})} & 0 & -\dfrac{1}{C_2(R_2 + R_{C2})} \end{bmatrix}$$

$$B = \begin{bmatrix} 1/L \\ 0 \\ 0 \end{bmatrix} \qquad C = \begin{bmatrix} 0 & \dfrac{R_1}{R_1 + R_{C1}} & 0 \\ \dfrac{R_2R_{C2}}{R_2 + R_{C2}} & 0 & \dfrac{R_2}{R_2 + R_{C2}} \end{bmatrix}$$

State 5: Switches S_2 and S_4 Closed

$$A = \begin{bmatrix} -\dfrac{R_2R_{C2} + R_2R_L + R_LR_{C2}}{L(R_2 + R_{C2})} & 0 & -\dfrac{R_2}{L(R_2 + R_{C2})} \\ 0 & -\dfrac{1}{C_1(R_1 + R_{C1})} & 0 \\ \dfrac{R_2}{C_2(R_2 + R_{C2})} & 0 & -\dfrac{1}{C_2(R_2 + R_{C2})} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad C = \begin{bmatrix} 0 & \dfrac{R_1}{R_1 + R_{C1}} & 0 \\ \dfrac{R_2R_{C2}}{R_2 + R_{C2}} & 0 & \dfrac{R_2}{R_2 + R_{C2}} \end{bmatrix}$$

State 6: Switch S_5 Closed ($I_L = 0$)

$$A = \begin{bmatrix} -\dfrac{1}{C_1(R_1 + R_{C1})} & 0 \\ 0 & -\dfrac{1}{C_2(R_2 + R_{C2})} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad C = \begin{bmatrix} \dfrac{R_1}{R_1 + R_{C1}} & 0 \\ 0 & \dfrac{R_2}{R_2 + R_{C2}} \end{bmatrix}$$

The maximum average load current for DCM operation can be calculated for each load by studying the boundary between DCM and CCM operation, which is depicted in Figure 4. For each output, we can calculate:

$$i_{Lpeak} = \frac{(V_{IN} - i_{Lpeak}R_L - v_{OUT})DT}{L} \rightarrow i_{Lpeak} = \frac{(V_{IN} - v_{OUT})DT}{L + R_L DT}$$

$$\langle i_{OUT} \rangle = I_{OUT} = \frac{i_{Lpeak}T}{2T} = \frac{i_{Lpeak}}{2}$$

For the circuit parameters listed in Figure 5, and with $D_1 = 0.5$ and $D_2 = 0.25$ being the average duty cycles at which S_1 and S_2 are operated for each output, we can determine approximate maximum average load currents of $I_{OUT1} = 70mA$ and $I_{OUT2} = 53mA$. If the loads demand more current than this, the converter will slip into continuous conduction operation, with its associated cross-regulation problems.

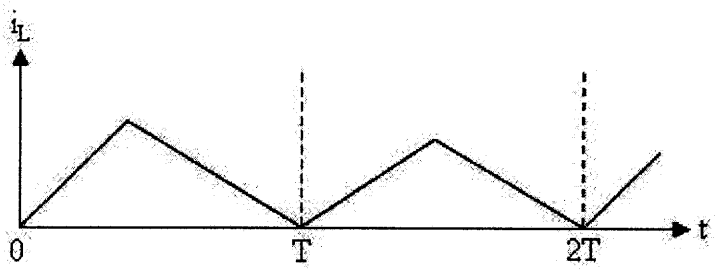


Figure 4: Inductor Current at DCM/CCM Boundary

2.2 Simulation Results

The SIMO buck converter was simulated in MATLAB using the aforementioned state space methods, the code for which can be seen in Appendix A. The simulation utilizes PID control in the feedback loop that determines the duty cycles for switches S_1 and S_2. Figure 5 shows a plot of the two output voltages and the inductor current, as well as a list of the circuit parameters used. Each output voltage reaches steady state in approximately 40μs, and the

inductor current can be seen to decrease and reach its DCM operating point at about that same instant. Unfortunately, as calculated in Section 2.1 for DCM operation, the maximum average load current for each output is only about 50mA for the balanced load case of $I_{OUT1} = I_{OUT2}$. This weak load-serving capability clearly demonstrates the need for PCCM operation, as most applications require more current than this. PCCM control of the SIMO converter will be explored in Chapter 3.

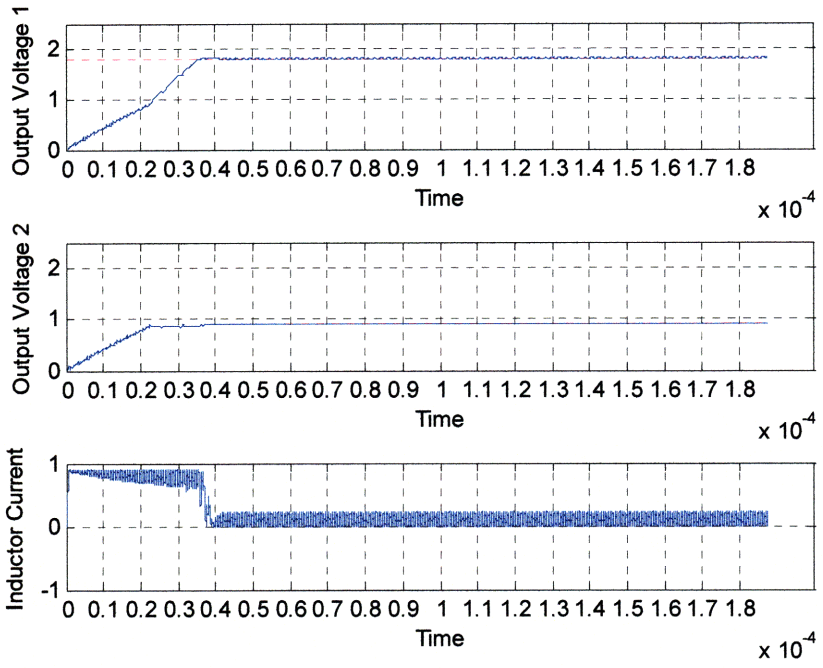


Figure 5: SIMO Simulation Results and Circuit Parameters

The converter that generated the plots in Figure 5 implemented a correction for an initialization problem that was encountered. Figure 6 shows the uncorrected simulation results, in which v_{OUT2} (since it is the lower output voltage of the two) overshoots a great deal and does not begin to settle toward steady state until v_{OUT1} has reached its steady state level. This happens because the inductor current stays fairly high while v_{OUT1} is ramping up to its steady state voltage, and this high current gets switched into Output 2 every other cycle. Thus v_{OUT2} keeps

increasing, with an overshoot of approximately 700mV and a settling time of 140μs. The following correction was therefore implemented: at start-up, after v_{OUT2} surpasses 95% of V_{REF2}, all of the inductor current is channeled into Output 1 until v_{OUT1} reaches V_{REF1}. This eliminates v_{OUT2}'s overshoot and greatly reduces both v_{OUT1}'s and v_{OUT2}'s settling times to about 40μs.

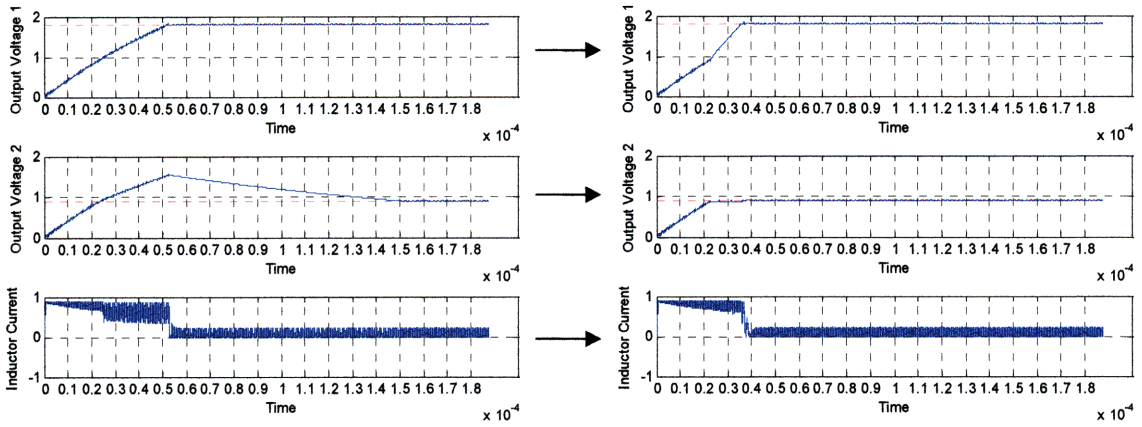


Figure 6: SIMO Initialization Not Corrected → Initialization Corrected

The SIMO converter's transient performance is also quite good. Figure 7 and Figure 8 show simulation results for load current steps from 50mA to 1mA to 50mA, with the transient voltage spikes having magnitudes between 10mV and 20mV. Unfortunately, a 50mA load current step is the largest that can be simulated because of the DCM restriction.

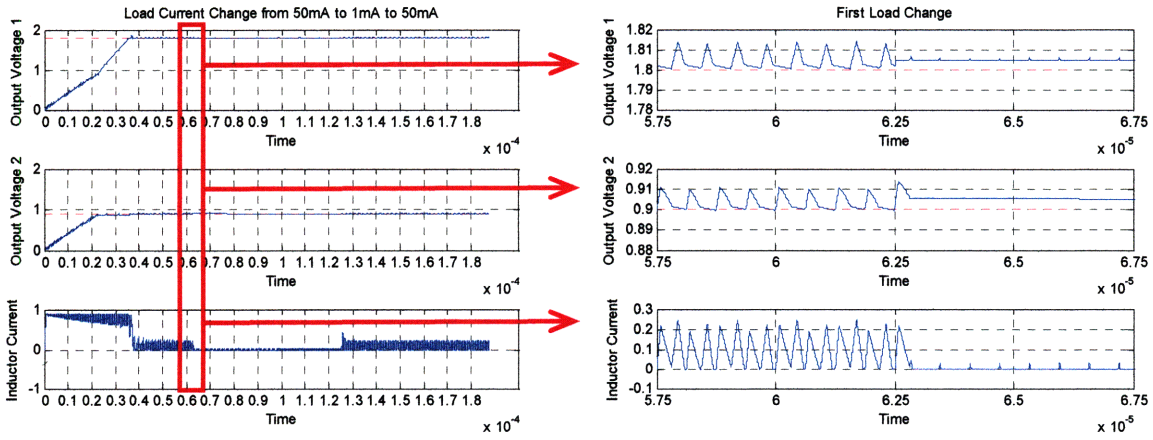


Figure 7: Transient for Load Current Change from 50mA to 1mA

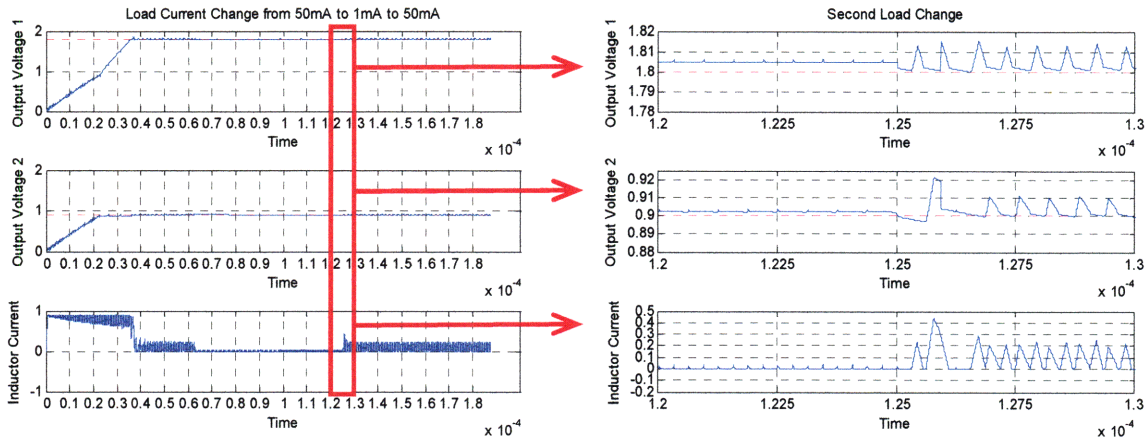


Figure 8: Transient for Load Current Change from 1mA to 50mA

In addition to the preceding simulations performed with L = 1μH and $C_1 = C_2 = 10\mu F$, the converter's performance was also tested for different inductor and capacitor values. Keeping the inductor value constant at 1μH, along with current loads of 50mA for both outputs, the settling times and ripple voltages were measured for different capacitor values commonly used in Qualcomm's integrated circuits. This data is displayed in Table 1.

Table 1: Converter Performance for Different Capacitor Values, with $I_{OUT1} = I_{OUT2} = 50mA$

Inductor Value L	Capacitor Value $C_1 = C_2$	Settling Time	Output 1 Voltage Ripple	Output 2 Voltage Ripple
1μH	4.7μF	22μs	30mV	27.9mV
1μH	10μF	40μs	13mV	10.7mV
1μH	22μF	82μs	14mV	11mV

The load current of 50mA for each output proved to be too high when testing larger inductor values, as increasing L above 1μH caused the converter to pull out of DCM into CCM in order to supply the required load current. As previously mentioned, this continuous-conduction operation could lead to cross-regulation problems. For L = 2.2μH, the maximum current the converter can supply to each load while staying in DCM is 20mA, while it is an even lower 10mA for L = 4.7μH. In order to provide a constant basis for comparison, the converter

was tested with the 10mA load requirement for different inductor values, with the results displayed in Table 2.

Table 2: Converter Performance for Different Inductor Values, with $I_{OUT1} = I_{OUT2} = 10mA$

Inductor Value L	Capacitor Value $C_1 = C_2$	Settling Time	Output 1 Voltage Ripple	Output 2 Voltage Ripple
1μH	10μF	36μs	5mV	4.8mV
2.2μH	10μF	45μs	4mV	3.1mV
4.7μH	10μF	80μs	2mV	2.2mV

Chapter 3

SIMO Buck Converter Operating in PCCM

One of the goals of this thesis was to exploit the advantages of PCCM operation in the SIMO buck converter. Unfortunately, however, PCCM control was unable to be nicely implemented in the SIMO converter, as excessive ripple and other problems could not quite be corrected. The following sections detail the partial mathematical model and simulation results that were obtained, while discussing the problems that were encountered.

3.1 Mathematical Model

The converter's topology is shown in Figure 9 and is the same as in the DCM case. The converter's timing diagram is also the same, reflecting the PCCM switching methodology described in Section 1.2.

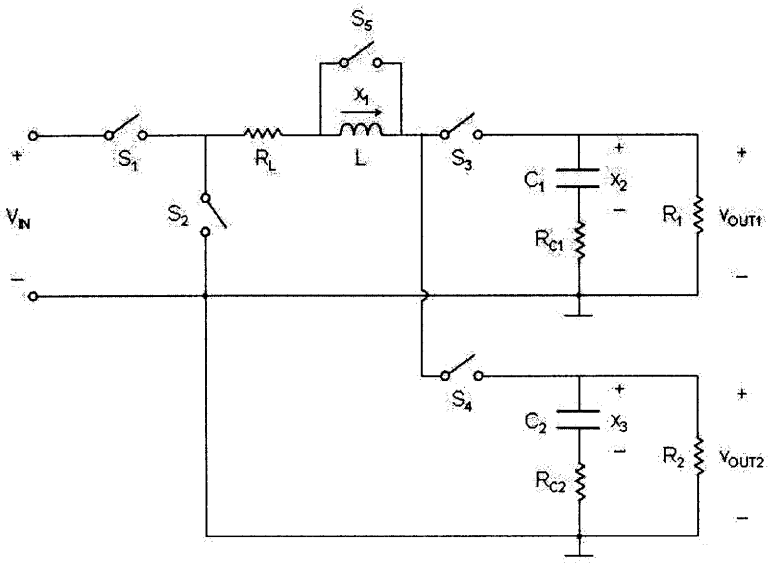


Figure 9: SIMO Buck Converter Topology

The basic mathematical state-space model also does not change for the PCCM case, with all the same state matrices. However, in States 3 and 6 from Section 2.1, when only S_5 is closed, the inductor current is no longer held constant at zero as in DCM. In PCCM, the freewheeling path provided by S_5 holds the inductor current constant at a nonzero value. The inductor current thus has the form shown in Figure 10 for the unbalanced load case, in which each output requires a different DC inductor current level.

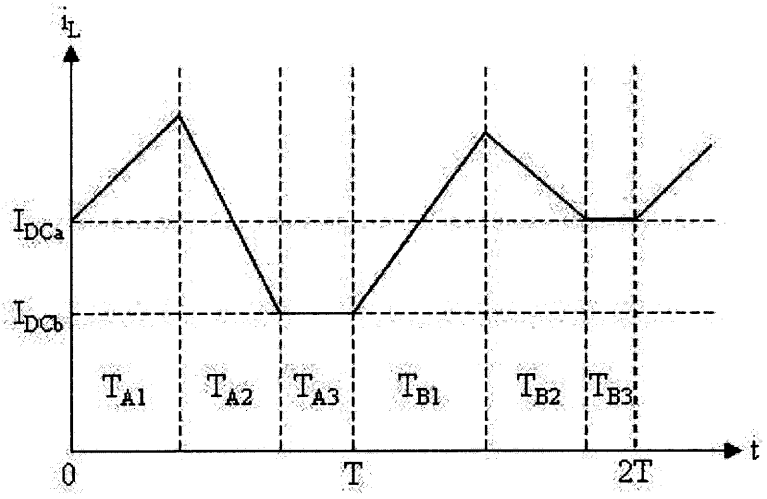


Figure 10: Inductor Current in PCCM Operation for Unbalanced Loads

The difficulty comes in determining each output's appropriate DC inductor current level for each cycle, I_{DCa} and I_{DCb}. In calculating these, we face six unknowns:

1) I_{DCa} – output 1's "freewheeling" DC inductor current level for PCCM operation
2) I_{DCb} – output 2's "freewheeling" DC inductor current level for PCCM operation
3) T_{A1} – period of time for which switches S_1 and S_3 are closed (all others open)
4) T_{A2} – period of time for which switches S_2 and S_3 are closed (all others open)
5) T_{B1} – period of time for which switches S_1 and S_4 are closed (all others open)
6) T_{B2} – period of time for which switches S_2 and S_4 are closed (all others open)

Four equations relating these variables have been found, but these are insufficient since there are six unknowns. Equations 1 and 2 provide expressions for I_{Oa} and I_{Ob}, the average output currents, where V_{Oa} and V_{Ob} denote the output voltages. For each output, the average current is just the integral of the inductor current over the time that the inductor is connected to that output, divided by the period (2T in Figure 10). These integrals are then simplified by finding the area under the appropriate parts of the curve in Figure 10.

$$I_{Oa} = \frac{1}{2T}\int_0^{T-T_{A3}} I_L(t)dt \qquad \text{Eq. 1}$$

$$= \frac{1}{2T}\left[I_{DCb}(T_{A1}+T_{A2}) + T_{A1}(I_{DCa}-I_{DCb}) + \frac{1}{2}T_{A1}\left(\frac{V_I - V_{Oa}}{L}T_{A1}\right) + \frac{1}{2}T_{A2}\left(\frac{V_{Oa}}{L}T_{A2}\right)\right]$$

$$= \frac{1}{2T}\left[I_{DCb}T_{A2} + I_{DCa}T_{A1} + \frac{V_I - V_{Oa}}{2L}T_{A1}^2 + \frac{V_{Oa}}{2L}T_{A2}^2\right]$$

$$I_{Ob} = \frac{1}{2T}\int_0^{T-T_{B3}} I_L(t)dt \qquad \text{Eq. 2}$$

$$= \frac{1}{2T}\left[I_{DCb}(T_{B1}+T_{B2}) + T_{B2}(I_{DCa}-I_{DCb}) + \frac{1}{2}T_{B1}\left(\frac{V_I - V_{Ob}}{L}T_{B1}\right) + \frac{1}{2}T_{B2}\left(\frac{V_{Ob}}{L}T_{B2}\right)\right]$$

$$= \frac{1}{2T}\left[I_{DCb}T_{B1} + I_{DCa}T_{B2} + \frac{V_I - V_{Ob}}{2L}T_{B1}^2 + \frac{V_{Ob}}{2L}T_{B2}^2\right]$$

Equations 3 and 4 simply follow the inductor current in Figure 10 through one cycle, from one DC level to the next, without involving the average output current.

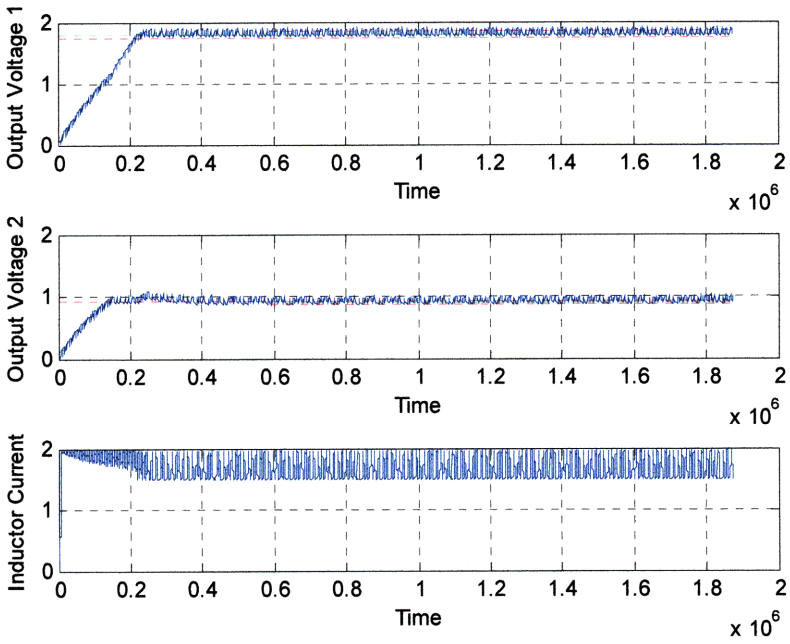
$$I_{DCa} + \frac{V_I - V_{Oa}}{L}T_{A1} - \frac{V_{Oa}}{L}T_{A2} = I_{DCb} \qquad \text{Eq. 3}$$

$$I_{DCb} + \frac{V_I - V_{Ob}}{L}T_{B1} - \frac{V_{Ob}}{L}T_{B2} = I_{DCa} \qquad \text{Eq. 4}$$

This under-constrained control problem indicates that time-multiplexing control may not be appropriate for a SIMO converter operating in PCCM, but my time constraints prevented much research into alternate control methods.

3.2 Simulation Results

The SIMO converter operating in PCCM was simulated in MATLAB, the code for which can be seen in Appendix B. In light of the incomplete mathematical model discussed in Section 3.1, the DC inductor current level for each output was chosen arbitrarily and verified graphically. Ideally, the converter itself should determine the necessary current levels, possibly even dynamically adjusting them each cycle through a feedback loop, but this is impossible here given the current state of the mathematical model. Nonetheless, the simulation results show promise, as can be seen in Figure 11. Both load currents were set at 500mA, and both DC current levels were set at 1.5A, while the inductor current was limited to a maximum of 2A.



$V_{IN} = 3.6V$
$V_{REF1} = 1.8V$
$V_{REF2} = 0.9V$
$I_{OUT1} = I_{OUT2} = 500mA$
$L = 1\mu H$
$C_1 = C_2 = 10\mu F$
$R_L = 50m\Omega$
$R_{C1} = R_{C2} = 50m\Omega$
$f_{sw} = 3.2MHz$

$I_{DCa} = 1.5A$
$I_{DCb} = 1.5A$

Figure 11: SIMO Converter in PCCM Simulation Results and Circuit Parameters

The converter behavior is not ideal, with a large ripple voltage of approximately 130mV and ugly steady state behavior, as can be seen in the magnified plots in Figure 12. The PID control parameters were re-tuned in an attempt to correct these issues, but to no avail.

Additionally, the DC current levels had to be set surprisingly high in order to avoid continuous conduction, and even then the converter occasionally slipped into CCM. Higher DC current levels and a higher maximum inductor current were tried, but this CCM problem continued.

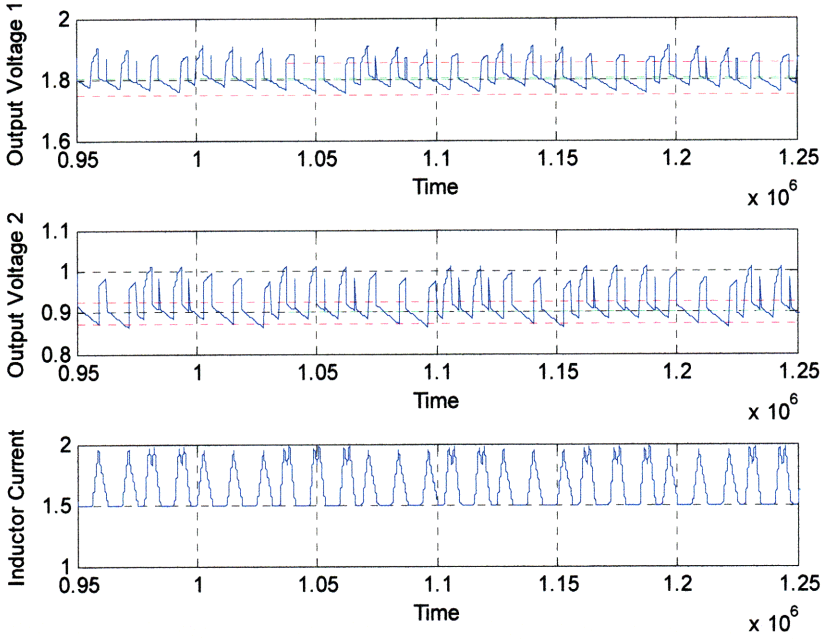


Figure 12: SIMO Converter in PCCM Magnified Steady State Behavior

The converter's transient behavior was also tested, in order to determine if the parallel source transient recovery system could be useful despite the converter's somewhat messy performance. Figure 13 shows the converter's behavior for load current steps from 500mA to 50mA to 500mA, with the DC current levels set to 150mA for the lighter 50mA load. The inductor current keeps spiking even during steady state with the light load, and this problem was not fixed by returning the converter to DCM operation during this stage, thus probably indicating a simulation coding error. Even more troubling is the fact that the transient voltages are of approximately the same magnitude as the steady state ripple voltages for the heavy load, thus rendering the parallel sources useless since they should not be activated by the ripple.

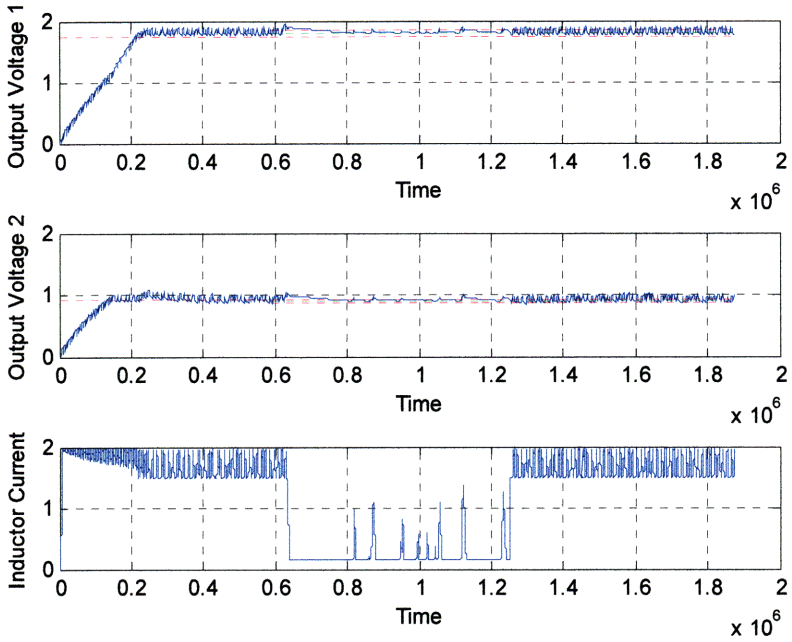


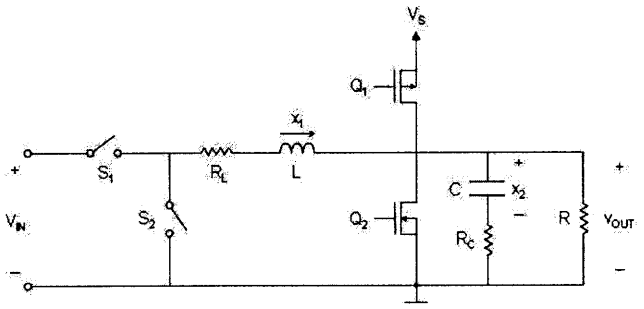
Figure 13: Transient Behavior for Load Current Change from 500mA to 50mA to 500mA

Chapter 4

Single-Output Buck Converter with Parallel Source Transient Recovery

4.1 Mathematical Model

Initial testing of the parallel source transient recovery idea focused on just a single-output buck converter. The topology and mathematical model are shown in Figure 14, and the math is quite similar to the aforementioned model for the SIMO converter. When the output voltage goes above or below a certain threshold voltage, either the top or bottom source turns on to source or sink current, whichever is appropriate. This simply adds another input *i* to the state equations – either positive or negative current, depending on which source is on – with the new state matrices F and G now being introduced. This input is then just set back to zero when the current injection is no longer needed and the source turns off.



$\mathbf{dx/dt} = \mathbf{Ax} + \mathbf{Bu} \pm \mathbf{Gi}$

$\mathbf{y} = \mathbf{Cx} \pm \mathbf{Fi}$

Model in discrete time (for simulation):

$\mathbf{x}(n+1) = \mathbf{\Phi x}(n) + \mathbf{\Gamma}_1 u(n) \pm \mathbf{\Gamma}_2 i(n)$

$\mathbf{y}(n+1) = \mathbf{Cx}(n+1) \pm \mathbf{F}i(n)$

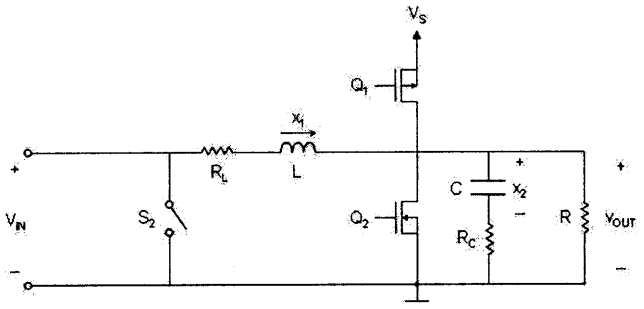
where $\mathbf{\Phi} = e^{AT}$

$\mathbf{\Gamma}_1 = \mathbf{B}^*(e^{AT} - e^{A^*0})/\mathbf{A}$

$\mathbf{\Gamma}_2 = \mathbf{G}^*(e^{AT} - e^{A^*0})/\mathbf{A}$

Figure 14: Buck Converter with Parallel Sources Topology and Mathematical Model

One of the switching states and its state matrices are shown in Figure 15, in which S_1 is closed and either the top or bottom parallel source is on (depending on whether the output voltage is too high or too low). The inductor current is thus ramping up and flowing into the output, and current is either being added or subtracted through one of the parallel sources.



$$A = \begin{bmatrix} -\dfrac{RR_C + RR_L + R_L R_C}{L(R+R_C)} & -\dfrac{R}{L(R+R_C)} \\ \dfrac{R}{C(R+R_C)} & -\dfrac{1}{C(R+R_C)} \end{bmatrix}$$

$$B = \begin{bmatrix} 1/L \\ 0 \end{bmatrix} \qquad C = \begin{bmatrix} \dfrac{RR_C}{R+R_C} & \dfrac{R}{R+R_C} \end{bmatrix}$$

$$F = \begin{bmatrix} \dfrac{RR_C}{R+R_C} \end{bmatrix} \qquad G = \begin{bmatrix} -\dfrac{RR_C}{L(R+R_C)} \\ \dfrac{R}{C(R+R_C)} \end{bmatrix}$$

Figure 15: Example Switching State with Current Being Injected

4.2 Simulation Results

The parallel source current injection makes a significant difference even in just this single-output converter, as did the similar active clamp design in [13], which bodes well for the effect it could have in a SIMO converter (though the control issues would be more complex). Using voltage thresholds of ±1% for the parallel sources, a maximum injected current level of

300mA, and load current steps from 500mA to 50mA to 500mA, transients were simulated in this single-output buck converter both with and without the parallel sources. This MATLAB code can be seen in Appendix C. The injected current (whether positive or negative) is modeled as ramping up to the maximum 300mA over 10ns, and then remaining constant at 300mA, assuming the sources stay on for that long. Figure 16 shows the standard buck converter transient response without the parallel sources, while Figure 17 shows the improved transient response with the sources. The red dashed lines in Figure 17 are the ±1% thresholds, while the green lines are the hysteretic lower thresholds used for turning the sources off. There is some oscillation and ugly behavior, since the two control loops may not be well-integrated with each other, but the transient voltage overshoot and settling time are nonetheless improved.

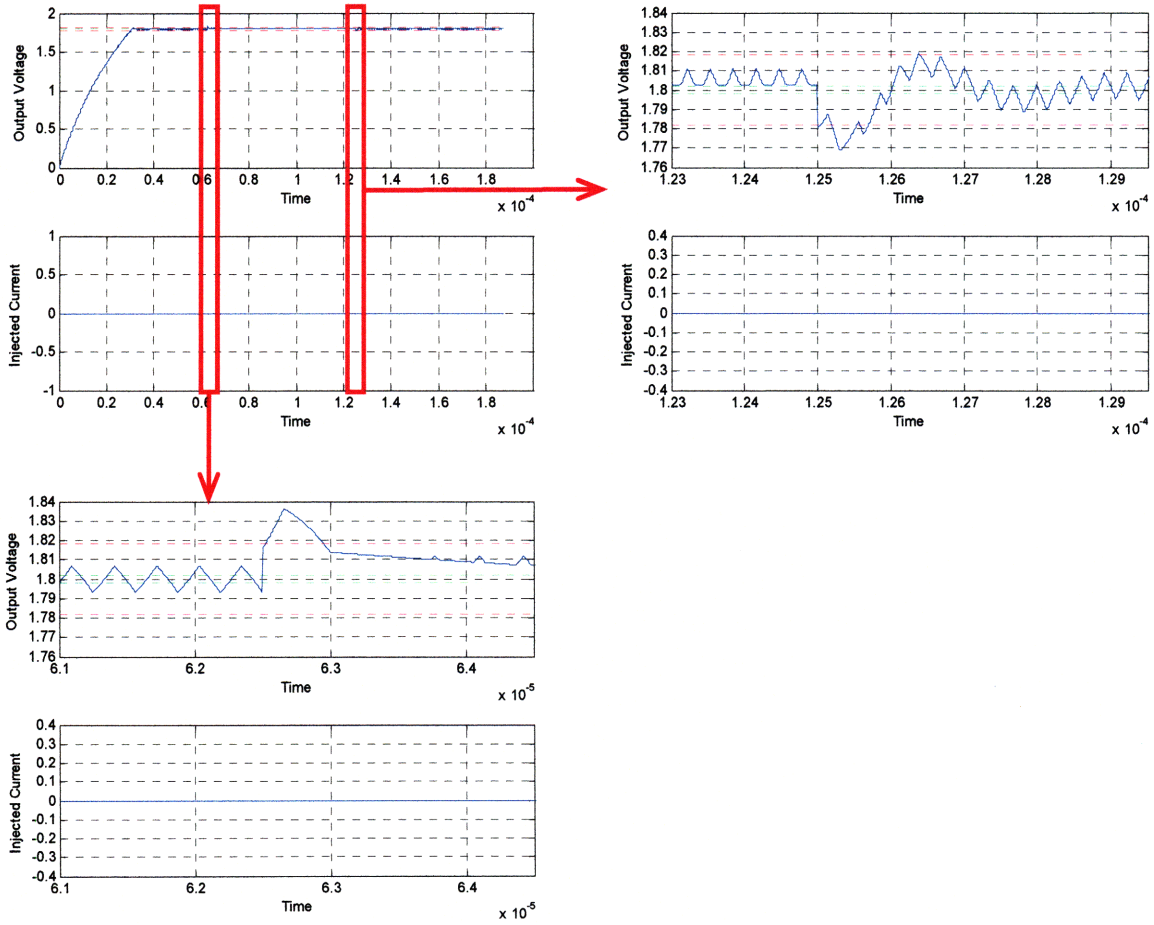


Figure 16: Transient Response *without* Parallel Sources

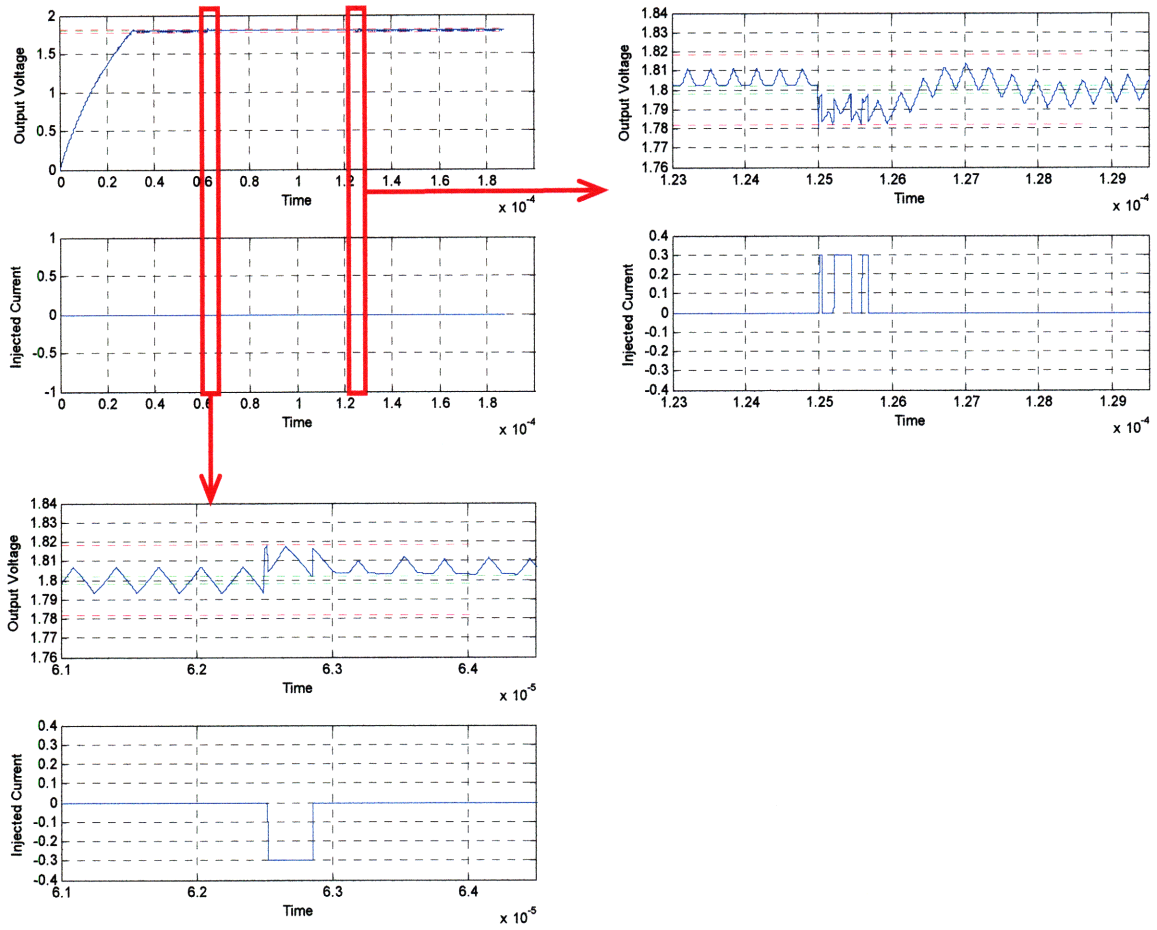


Figure 17: Transient Response *with* Parallel Sources

The relationship between the maximum injected current level and the transient voltage overshoot was also tested for a 500mA load current step, which produced the surprisingly linear result shown in Figure 18 (though only six data points were used). Although a higher maximum injected current generally means a lower voltage overshoot and a faster settling time, in practice it would also require larger and more expensive power devices, so there is a design trade-off for which an engineer could use the plot in Figure 18. Beyond the 300mA level for I_{max}, high frequency ringing was introduced, again possibly the result of having two competing control loops. This problem is certainly something that should be further researched.

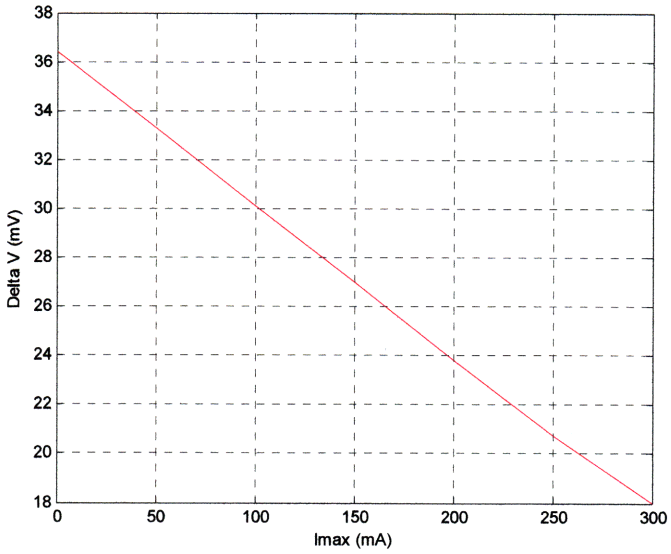


Figure 18: Voltage Overshoot (ΔV) vs. Maximum Current Injected by a Parallel Source (I_{max})

Chapter 5

Conclusions and Future Work

The results in the previous chapters demonstrate much promise for the SIMO buck converter operating in PCCM with parallel source transient recovery, though the design has not yet been perfected. The strange PCCM behavior in our simulations, described in Chapter 3, prevents our testing of the parallel source idea in a SIMO converter, since it is only necessary when supplying heavy loads and thus cannot be tested with a DCM converter. Given the PCCM behavior obtained thus far, any reasonably chosen voltage thresholds for the sources would be surpassed every cycle by the large steady state voltage ripple, and the transient voltages are no greater than the ripple amplitudes. Indeed, testing of the parallel sources with this imperfect PCCM operation was attempted, but they continually turned on and off during steady state operation. In addition to just being senseless, this sort of operation would dissipate a great deal of power since the sources would be on during much of the converter's operation.

It was therefore concluded that PCCM operation must first be mastered before a viable proof-of-concept can be provided for the parallel source idea. The two control loops may be difficult to integrate together, especially in a real-world setting with all its non-idealities, but at this stage the system-level idea seems promising. Admittedly, too much time was spent debugging code and attempting to solve the under-constrained control problem discussed in Section 3.1, rather than rethinking time-multiplexing control altogether and exploring other options such as current-mode control. The problems encountered here therefore probably result mostly from implementation and control issues in the simulations, rather than fundamental flaws in the idea.

Once the SIMO buck converter is operating nicely in PCCM with the parallel sources, more in-depth performance analysis can begin. This will include analyzing the converter's efficiency and testing its behavior with different size inductors and capacitors, balanced and unbalanced loads, different input and output voltages, and other variations. The improved transient performance afforded by the parallel sources would allow greater flexibility in sizing the inductor and output capacitors, though one must also study the delay inherent in the parallel sources' devices and control loops. The more distant future could see the expansion of the SIMO converter to three or more outputs. This may be theoretically possible, but getting it to work in practice could be rather difficult because of non-idealities, and the control issues would be monumental.

Appendix A

MATLAB code for SIMO buck converter operating in DCM

```
% Circuit parameters
Vin = 3.6;       % input voltage
L = 1e-6;        % inductor
rL = 50e-3 ;     % inductor parasitic resistance

Cf1 = 10e-6;     % output capacitor 1
rCf1 = 50e-3;    % output capacitor 1 ESR
Vref1 = 1.8;     % reference voltage 1
Iref1 = 0.050;   % load current 1
R1 = Vref1/Iref1; % output load resistor 1

Cf2 = 10e-6;     % output capacitor 2
rCf2 = 50e-3;    % output capacitor 2 ESR
Vref2 = 0.9;     % reference voltage 2
Iref2 = 0.050;   % load current 2
R2 = Vref2/Iref2; % output load resistor 2

Ilimit = 0.5;        % inductor current limit during startup
changeLoad = 0;      % 1 or 0 -- change load or don't change load for
                       transient simulation

numcycle = 200; % number of simulation cycles
fs = 3.2e6; % switching frequency
Ts = 1/fs; % sampling period;

% Continuous time model
A1(1,1) = -(rL*R1+rCf1*R1+rL*rCf1)/(L*(rCf1+R1));   % Matrices used when
                                                      inductor current is
                                                      positive (state 1)
A1(1,2) = -R1/(L*(R1+rCf1));
A1(1,3) = 0;
A1(2,1) = R1/(Cf1*(rCf1+R1));
A1(2,2) = -1/(Cf1*(rCf1+R1));
A1(2,3) = 0;
A1(3,1) = 0;
A1(3,2) = 0;
A1(3,3) = -1/(Cf2*(rCf2+R2));

B1 = [1/L ; 0 ; 0];

C1(1,1) = rCf1*R1/(rCf1+R1);
C1(1,2) = R1/(R1+rCf1);
C1(1,3) = 0;
C1(2,1) = 0;
C1(2,2) = 0;
C1(2,3) = R2/(R2+rCf2);

D1 = 0;


A2(1,1) = -(rL*R2+rCf2*R2+rL*rCf2)/(L*(rCf2+R2));   % Matrices used when
                                                      inductor current is
                                                      positive (state 2)
A2(1,2) = 0;
A2(1,3) = -R2/(L*(R2+rCf2));
```

```
A2(2,1) = 0;
A2(2,2) = -1/(Cf1*(rCf1+R1));
A2(2,3) = 0;
A2(3,1) = R2/(Cf2*(rCf2+R2));
A2(3,2) = 0;
A2(3,3) = -1/(Cf2*(rCf2+R2));

B2 = [1/L ; 0 ; 0];

C2(1,1) = 0;
C2(1,2) = R1/(R1+rCf1);
C2(1,3) = 0;
C2(2,1) = rCf2*R2/(rCf2+R2);
C2(2,2) = 0;
C2(2,3) = R2/(R2+rCf2);

D2 = 0;


A3(1,1) = -1/(Cf1*(rCf1+R1));          % Matrices used when inductor current
                                         is zero (for DCM)
A3(1,2) = 0;
A3(2,1) = 0;
A3(2,2) = -1/(Cf2*(rCf2+R2));

B3 = [0 ; 0];

C3(1,1) = R1/(R1+rCf1);
C3(1,2) = 0;
C3(2,1) = 0;
C3(2,2) = R2/(R2+rCf2);

D3 = 0;

% Discrete time model
td = 0.1e-9;  % sampling time;

Phi1 = expm(A1*td);     % State transition matrix in discrete time
Tau1 = (expm(A1*td)-expm(A1*0))/A1*B1;

Phi2 = expm(A2*td);     % State transition matrix in discrete time
Tau2 = (expm(A2*td)-expm(A2*0))/A2*B2;

Phi3 = expm(A3*td);     % State transition matrix in discrete time
Tau3 = (expm(A3*td)-expm(A3*0))/A3*B3;
%Tau3 = [0 ; 0 ; 0];     % Since B3 = [0 ; 0 ; 0]


Kp1 = 30;           % PID coefficients for Output 1
Ki1 = 0.0001;
Kd1 = 10;
df1 = Vref1/Vin;

Kp2 = 30;           % PID coefficients for Output 2
Ki2 = 0.0001;
Kd2 = 10;
df2 = Vref2/Vin;

% loop simulation

tend = numcycle*Ts;
t = td:td:tend;              % Time vector stepping by sample size
t_lr = Ts:Ts:tend;           % Time vector stepping by cycle size
x = zeros(3,3*length(t));
```

```
y = zeros(2,3*length(t));
iref1 = zeros(1,3*length(t));
iref2 = zeros(1,3*length(t));

Ve1 = zeros(1,3*numcycle);  % Error signal 1
DVe1 = zeros(1,3*numcycle); % Diff. error 1
IVe1 = zeros(1,3*numcycle); % Integral error 1
Ve2 = zeros(1,3*numcycle);  % Error signal 2
DVe2 = zeros(1,3*numcycle); % Diff. error 2
IVe2 = zeros(1,3*numcycle); % Integral error 2
dc = zeros(1,3*numcycle);
iref1_lr = zeros(1,3*numcycle);
iref2_lr = zeros(1,3*numcycle);

x0 = [0;0;0];
x_seg = zeros(3,Ts/td);
y_seg = zeros(2,Ts/td);
dc_seg = zeros(1,Ts/td);


dc0 = 0;
y(1:2,1) = C1*x0;
x(1:3,1) = Phi1*x0 + Tau1*Vin*dc0;
dc_seg_last = 0;

selector = 1;
notStartup = 0;
currentLimited = 0;

counter = 0;        % Counts the number of steps the inductor current takes
                      to rise and then fall to zero in each cycle, for
                      plotting duty cycles
dc_plot = zeros(1,3*numcycle);  % Keeps track of Vout/Vin = T1/(T1+T2) in DCM

for q = 2:1:3*numcycle
    if (q == numcycle+1) && (changeLoad == 1)   % Load current change from
                                                  50mA to 1mA
        Iref1 = 0.001;
        Iref2 = 0.001;
        R1 = Vref1/Iref1;
        R2 = Vref2/Iref2;

        A1(1,1) = -(rL*R1+rCf1*R1+rL*rCf1)/(L*(rCf1+R1));
        A1(1,2) = -R1/(L*(R1+rCf1));
        A1(1,3) = 0;
        A1(2,1) = R1/(Cf1*(rCf1+R1));
        A1(2,2) = -1/(Cf1*(rCf1+R1));
        A1(2,3) = 0;
        A1(3,1) = 0;
        A1(3,2) = 0;
        A1(3,3) = -1/(Cf2*(rCf2+R2));

        B1 = [1/L ; 0 ; 0];

        C1(1,1) = rCf1*R1/(rCf1+R1);
        C1(1,2) = R1/(R1+rCf1);
        C1(1,3) = 0;
        C1(2,1) = 0;
        C1(2,2) = 0;
        C1(2,3) = R2/(R2+rCf2);

        D1 = 0;
```

```
    A2(1,1) = -(rL*R2+rCf2*R2+rL*rCf2)/(L*(rCf2+R2));
    A2(1,2) = 0;
    A2(1,3) = -R2/(L*(R2+rCf2));
    A2(2,1) = 0;
    A2(2,2) = -1/(Cf1*(rCf1+R1));
    A2(2,3) = 0;
    A2(3,1) = R2/(Cf2*(rCf2+R2));
    A2(3,2) = 0;
    A2(3,3) = -1/(Cf2*(rCf2+R2));

    B2 = [1/L ; 0 ; 0];

    C2(1,1) = 0;
    C2(1,2) = R1/(R1+rCf1);
    C2(1,3) = 0;
    C2(2,1) = rCf2*R2/(rCf2+R2);
    C2(2,2) = 0;
    C2(2,3) = R2/(R2+rCf2);

    D2 = 0;


    A3(1,1) = -1/(Cf1*(rCf1+R1));
    A3(1,2) = 0;
    A3(2,1) = 0;
    A3(2,2) = -1/(Cf2*(rCf2+R2));

    B3 = [0 ; 0];

    C3(1,1) = R1/(R1+rCf1);
    C3(1,2) = 0;
    C3(2,1) = 0;
    C3(2,2) = R2/(R2+rCf2);

    D3 = 0;

    Phi1 = expm(A1*td);   % State transition matrix in discrete time
    Tau1 = (expm(A1*td)-expm(A1*0))/A1*B1;

    Phi2 = expm(A2*td);   % State transition matrix in discrete time
    Tau2 = (expm(A2*td)-expm(A2*0))/A2*B2;

    Phi3 = expm(A3*td);   % State transition matrix in discrete time
    Tau3 = (expm(A3*td)-expm(A3*0))/A3*B3;
    % Tau3 = [0 ; 0 ; 0];     % Since B3 = [0 ; 0 ; 0]
end

if (q == 2*numcycle+1) && (changeLoad == 1)   % Load current change from
                                                1mA to 50mA
    Iref1 = 0.050;
    Iref2 = 0.050;
    R1 = Vref1/Iref1;
    R2 = Vref2/Iref2;

    A1(1,1) = -(rL*R1+rCf1*R1+rL*rCf1)/(L*(rCf1+R1));
    A1(1,2) = -R1/(L*(R1+rCf1));
    A1(1,3) = 0;
    A1(2,1) = R1/(Cf1*(rCf1+R1));
    A1(2,2) = -1/(Cf1*(rCf1+R1));
    A1(2,3) = 0;
    A1(3,1) = 0;
    A1(3,2) = 0;
    A1(3,3) = -1/(Cf2*(rCf2+R2));
```

```
    B1 = [1/L ; 0 ; 0];

    C1(1,1) = rCf1*R1/(rCf1+R1);
    C1(1,2) = R1/(R1+rCf1);
    C1(1,3) = 0;
    C1(2,1) = 0;
    C1(2,2) = 0;
    C1(2,3) = R2/(R2+rCf2);

    D1 = 0;


    A2(1,1) = -(rL*R2+rCf2*R2+rL*rCf2)/(L*(rCf2+R2));
    A2(1,2) = 0;
    A2(1,3) = -R2/(L*(R2+rCf2));
    A2(2,1) = 0;
    A2(2,2) = -1/(Cf1*(rCf1+R1));
    A2(2,3) = 0;
    A2(3,1) = R2/(Cf2*(rCf2+R2));
    A2(3,2) = 0;
    A2(3,3) = -1/(Cf2*(rCf2+R2));

    B2 = [1/L ; 0 ; 0];

    C2(1,1) = 0;
    C2(1,2) = R1/(R1+rCf1);
    C2(1,3) = 0;
    C2(2,1) = rCf2*R2/(rCf2+R2);
    C2(2,2) = 0;
    C2(2,3) = R2/(R2+rCf2);

    D2 = 0;


    A3(1,1) = -1/(Cf1*(rCf1+R1));
    A3(1,2) = 0;
    A3(2,1) = 0;
    A3(2,2) = -1/(Cf2*(rCf2+R2));

    B3 = [0 ; 0];

    C3(1,1) = R1/(R1+rCf1);
    C3(1,2) = 0;
    C3(2,1) = 0;
    C3(2,2) = R2/(R2+rCf2);

    D3 = 0;

    Phi1 = expm(A1*td);   % State transition matrix in discrete time
    Tau1 = (expm(A1*td)-expm(A1*0))/A1*B1;

    Phi2 = expm(A2*td);   % State transition matrix in discrete time
    Tau2 = (expm(A2*td)-expm(A2*0))/A2*B2;

    Phi3 = expm(A3*td);   % State transition matrix in discrete time
    Tau3 = (expm(A3*td)-expm(A3*0))/A3*B3;
    % Tau3 = [0 ; 0 ; 0];     % Since B3 = [0 ; 0 ; 0]
end

if (y_seg(2,Ts/td) >= (0.95*Vref2)) && (y_seg(1,Ts/td) < (0.99*Vref1)) &&
notStartup == 0
    selector = 1;       % Initialization correction -- channels inductor
                          current into Output 1
elseif y_seg(1,Ts/td) >= (0.99*Vref1)
```

```
    notStartup = 1;
end

if selector == 1            % IF INDUCTOR CONNECTED TO OUTPUT 1
    y_avg1(q) = mean(y_seg(1,1:end));
    Ve1(q) = Vref1-y_avg1(q);  % error signal 1
    DVe1(q) = Ve1(q)-Ve1(q-1);
    IVe1(q) = IVe1(q-1)+ Ve1(q);

    y_avg2(q) = mean(y_seg(2,1:end));
    Ve2(q) = Vref2-y_avg2(q);  % error signal 2
    DVe2(q) = Ve2(q)-Ve2(q-1);
    IVe2(q) = IVe2(q-1)+ Ve2(q);

    dc(q) = Kp1*Ve1(q)+Ki1*IVe1(q)+Kd1*DVe1(q)+df1;     % Duty cycle
    iref1_lr(q) = Vref1/R1;

    if dc(q)> 1;
        dc(q) = 1;
    elseif dc(q) <0;
        dc(q)=0;
    end
    dc_max(q) = L*(0.9-x_seg(1,Ts/td))/(Vin-y_seg(1,Ts/td))/Ts; % Limit
        inductor current to 0.9A
    if dc(q) > dc_max(q)
        dc(q) = dc_max(q);
    end

    if (y_seg(1,Ts/td) >= (0.99*Vref1)) % Inductor current limit during
                                        startup
        dc_max(q) = L*(Ilimit-x_seg(1,Ts/td))/(Vin-y_seg(1,Ts/td))/Ts;
        if dc(q) > dc_max(q)
            if dc_max >= 0
                dc(q) = dc_max(q);
            else
                dc(q) = 0;
            end
        end
    end

    x_seg(1:3,1) = Phi1*x_seg(1:3,Ts/td)+Tau1*Vin*dc_seg_last;  % For
        first step of each cycle, must use values from last step of last
        cycle
    y_seg(1:2,1) = C1*x_seg(1:3,1);

    dc_seg(1:round(Ts/td*(dc(q)))) = 1;
    dc_seg(round(Ts/td*(dc(q))+1):Ts/td) = 0;

    for n = 1:1:Ts/td-1
        if dc_seg(n) == 1
            x_seg(1:3,n+1) = Phi1*x_seg(1:3,n) + Tau1*Vin*dc_seg(n);
            y_seg(1:2,n+1) = C1*x_seg(1:3,n+1);
            counter = counter + 1;
        else
            if (x_seg(1,n) < 0.0001) && (y_seg(1,n) > (0.25*Vref1))
                    % Keeps inductor current at zero for DCM
                x_seg(1,n+1) = 0;
                x_seg(2:3,n+1) = Phi3*x_seg(2:3,n) + Tau3*Vin*dc_seg(n);
                y_seg(1:2,n+1) = C3*x_seg(2:3,n+1);
            else
                x_seg(1:3,n+1) = Phi1*x_seg(1:3,n) + Tau1*Vin*dc_seg(n);
                y_seg(1:2,n+1) = C1*x_seg(1:3,n+1);
                counter = counter + 1;
            end
```

```
        end
    end

    dc_plot(q) = round(dc(q)*Ts/td)/counter;    % Duty cycle for plotting
    counter = 0;

    dc_seg_last = dc_seg(n+1);
    x(1,round((q-1)*Ts/td)+1:round(q*Ts/td))=x_seg(1,:);
    x(2,round((q-1)*Ts/td)+1:round(q*Ts/td))=x_seg(2,:);
    x(3,round((q-1)*Ts/td)+1:round(q*Ts/td))=x_seg(3,:);
    y(1,round((q-1)*Ts/td)+1:round(q*Ts/td))=y_seg(1,:);
    y(2,round((q-1)*Ts/td)+1:round(q*Ts/td))=y_seg(2,:);
    iref1(round((q-1)*Ts/td)+1:round(q*Ts/td))=Vref1/R1;

    selector = 0;   % Switch outputs

else            % IF INDUCTOR CONNECTED TO OUTPUT 2
    y_avg2(q) = mean(y_seg(2,1:end));
    Ve2(q) = Vref2-y_avg2(q);  % error signal 2
    DVe2(q) = Ve2(q)-Ve2(q-1);
    IVe2(q) = IVe2(q-1)+ Ve2(q);

    y_avg1(q) = mean(y_seg(1,1:end));
    Ve1(q) = Vref1-y_avg1(q);  % error signal 1
    DVe1(q) = Ve1(q)-Ve1(q-1);
    IVe1(q) = IVe1(q-1)+ Ve1(q);

    dc(q) = Kp2*Ve2(q)+Ki2*IVe2(q)+Kd2*DVe2(q)+df2;     % Duty cycle
    iref2_lr(q) = Vref2/R2;

    if dc(q)> 1;
        dc(q) = 1;
    elseif dc(q) <0;
        dc(q)=0;
    end
    dc_max(q) = L*(0.9-x_seg(1,Ts/td))/(Vin-y_seg(2,Ts/td))/Ts; % Limit
        inductor current to 0.9A
    if dc(q) > dc_max(q)
        dc(q) = dc_max(q);
    end

    if (y_seg(1,Ts/td) >= (0.99*Vref1)) % Inductor current limit during
                                            startup
        dc_max(q) = L*(Ilimit-x_seg(1,Ts/td))/(Vin-y_seg(2,Ts/td))/Ts;
        if dc(q) > dc_max(q)
            if dc_max >= 0
                dc(q) = dc_max(q);
            else
                dc(q) = 0;
            end
        end
    end

    x_seg(1:3,1) = Phi2*x_seg(1:3,Ts/td)+Tau2*Vin*dc_seg_last;  % For
        first step of each cycle, must use values from last step of last
        cycle
    y_seg(1:2,1) = C2*x_seg(1:3,1);

    dc_seg(1:round(Ts/td*(dc(q)))) = 1;
    dc_seg(round(Ts/td*(dc(q))+1):Ts/td) = 0;

    for n = 1:1:Ts/td-1
        if dc_seg(n) == 1
            x_seg(1:3,n+1) = Phi2*x_seg(1:3,n) + Tau2*Vin*dc_seg(n);
```

```
                y_seg(1:2,n+1) = C2*x_seg(1:3,n+1);
                counter = counter + 1;
            else
                if (x_seg(1,n) < 0.0001) && (y_seg(2,n) > (0.25*Vref2))
                        % Keeps inductor current at zero for DCM
                    x_seg(1,n+1) = 0;
                    x_seg(2:3,n+1) = Phi3*x_seg(2:3,n) + Tau3*Vin*dc_seg(n);
                    y_seg(1:2,n+1) = C3*x_seg(2:3,n+1);
                else
                    x_seg(1:3,n+1) = Phi2*x_seg(1:3,n) + Tau2*Vin*dc_seg(n);
                    y_seg(1:2,n+1) = C2*x_seg(1:3,n+1);
                    counter = counter + 1;
                end
            end
        end

        dc_plot(q) = round(dc(q)*Ts/td)/counter;    % Duty cycle for plotting
        counter = 0;

        dc_seg_last = dc_seg(n+1);
        x(1,round((q-1)*Ts/td)+1:round(q*Ts/td))=x_seg(1,:);
        x(2,round((q-1)*Ts/td)+1:round(q*Ts/td))=x_seg(2,:);
        x(3,round((q-1)*Ts/td)+1:round(q*Ts/td))=x_seg(3,:);
        y(1,round((q-1)*Ts/td)+1:round(q*Ts/td))=y_seg(1,:);
        y(2,round((q-1)*Ts/td)+1:round(q*Ts/td))=y_seg(2,:);
        iref2(round((q-1)*Ts/td)+1:round(q*Ts/td))=Vref2/R2;

        selector = 1;   % Switch outputs
    end
end


t_long = td:td:3*tend;
t_lr_long = Ts:Ts:3*tend;


ref1 = Vref1*ones(1,length(t_long));
ref2 = Vref2*ones(1,length(t_long));
xscale = [0:0.1e-4:tend*3];

figure
subplot(3,1,1), plot(t_long,y(1,:),t_long,ref1,':r'), xlabel('Time'),
                ylabel('Output Voltage 1'), grid
set(gca,'XTick',xscale), ylim([0 2.5])
subplot(3,1,2), plot(t_long,y(2,:),t_long,ref2,':r'), xlabel('Time'),
                ylabel('Output Voltage 2'), grid
set(gca,'XTick',xscale), ylim([0 2.5])
subplot(3,1,3), plot(t_lr_long,dc_plot), xlabel('Time'), ylabel('Control
                Input (Duty Cycle)'), grid
set(gca,'XTick',xscale), ylim([0 1]) %axis([0 tend*3 0 1])

clear grid_l grid_h iref1 iref2 ref1 ref2 t y_avg1_mem y_avg2_mem u x_seg
      y_seg y_avg1 y_avg2 tab status iref1_lr iref2_lr;

figure
subplot(3,1,1), plot(t_long,x(1,:)), xlabel('Time'), ylabel('Inductor
                Current'), grid
set(gca,'XTick',xscale)
subplot(3,1,2), plot(t_long,x(2,:)), xlabel('Time'), ylabel('Capacitor
                Voltage 1'), grid
subplot(3,1,3), plot(t_long,x(3,:)), xlabel('Time'), ylabel('Capacitor
                Voltage 2'), grid
```

Appendix B

MATLAB code for SIMO buck converter operating in PCCM with optional parallel source current injection

```
% Circuit parameters
Vin = 3.6; % input voltage
L = 1e-6; % inductor
rL = 50e-3; % inductor parasitic resistance

Cf1 = 10e-6; % output capacitor 1
rCf1 = 50e-3; % output capacitor 1 ESR
Vref1 = 1.8; % reference voltage 1
Iref1 = 0.500; % load current 1
R1 = Vref1/Iref1; % output load resistor 1

Cf2 = 10e-6; % output capacitor 2
rCf2 = 50e-3; % output capacitor 2 ESR
Vref2 = 0.9; % reference voltage 2
Iref2 = 0.500; % load current 2
R2 = Vref2/Iref2; % output load resistor 2

Idc1 = Iref1+1.000;     % PCCM DC inductor current levels
Idc2 = Iref2+1.000;

Ilimit = 1.5;     % inductor current limit during startup

changeLoad = 0;   % 1 or 0 -- change load or don't change load for transient
                    simulation

Imax = 0.500;   % Maximum injected current level

Vmin1 = 0.97*Vref1;          % Current injection lower voltage threshold 1
Vmin_high1 = 0.999*Vref1;    % Current injection hysteretic lower turn-off
                               voltage threshold 1
Vmax1 = 1.03*Vref1;          % Current injection upper voltage threshold 1
Vmax_low1 = 1.001*Vref1;     % Current injection hysteretic upper turn-off
                               voltage threshold 1

Vmin2 = 0.97*Vref2;          % Current injection lower voltage threshold 2
Vmin_high2 = 0.999*Vref2;    % Current injection hysteretic lower turn-off
                               voltage threshold 2
Vmax2 = 1.03*Vref2;          % Current injection upper voltage threshold 2
Vmax_low2 = 1.001*Vref2;     % Current injection hysteretic upper turn-off
                               voltage threshold 2

numcycle = 200; % number of simulation cycles
fs = 3.2e6; % switching frequency
Ts = 1/fs; % sampling period;

% Continuous time model
A1(1,1) = -(rL*R1+rCf1*R1+rL*rCf1)/(L*(rCf1+R1));   % Matrices used when
                                                      inductor current is
                                                      positive (state 1)
A1(1,2) = -R1/(L*(R1+rCf1));
A1(1,3) = 0;
A1(2,1) = R1/(Cf1*(rCf1+R1));
A1(2,2) = -1/(Cf1*(rCf1+R1));
```

```
A1(2,3) = 0;
A1(3,1) = 0;
A1(3,2) = 0;
A1(3,3) = -1/(Cf2*(rCf2+R2));

B1 = [1/L ; 0 ; 0];

C1(1,1) = rCf1*R1/(rCf1+R1);
C1(1,2) = R1/(R1+rCf1);
C1(1,3) = 0;
C1(2,1) = 0;
C1(2,2) = 0;
C1(2,3) = R2/(R2+rCf2);

D1 = 0;

G1(1,1) = -R1*rCf1/(L*(R1+rCf1));
G1(1,2) = 0;
G1(2,1) = R1/(Cf1*(R1+rCf1));
G1(2,2) = 0;
G1(3,1) = 0;
G1(3,2) = R2/(Cf2*(R2+rCf2));

F1(1,1) = R1*rCf1/(R1+rCf1);
F1(1,2) = 0;
F1(2,1) = 0;
F1(2,2) = R2-(R2*R2/(R2+rCf2));


A2(1,1) = -(rL*R2+rCf2*R2+rL*rCf2)/(L*(rCf2+R2));    % Matrices used when
                                                       inductor current is
                                                       positive (state 2)
A2(1,2) = 0;
A2(1,3) = -R2/(L*(R2+rCf2));
A2(2,1) = 0;
A2(2,2) = -1/(Cf1*(rCf1+R1));
A2(2,3) = 0;
A2(3,1) = R2/(Cf2*(rCf2+R2));
A2(3,2) = 0;
A2(3,3) = -1/(Cf2*(rCf2+R2));

B2 = [1/L ; 0 ; 0];

C2(1,1) = 0;
C2(1,2) = R1/(R1+rCf1);
C2(1,3) = 0;
C2(2,1) = rCf2*R2/(rCf2+R2);
C2(2,2) = 0;
C2(2,3) = R2/(R2+rCf2);

D2 = 0;

G2(1,1) = 0;
G2(1,2) = -R2*rCf2/(L*(R2+rCf2));
G2(2,1) = R1/(Cf1*(R1+rCf1));
G2(2,2) = 0;
G2(3,1) = 0;
G2(3,2) = R2/(Cf2*(R2+rCf2));

F2(1,1) = R1-(R1*R1/(R1+rCf1));
F2(1,2) = 0;
F2(2,1) = 0;
F2(2,2) = R2*rCf2/(R2+rCf2);
```

```
A3(1,1) = -1/(Cf1*(rCf1+R1));        % Matrices used when inductor current is
                                       constant (for PCCM)
A3(1,2) = 0;
A3(2,1) = 0;
A3(2,2) = -1/(Cf2*(rCf2+R2));

B3 = [0 ; 0];

C3(1,1) = R1/(R1+rCf1);
C3(1,2) = 0;
C3(2,1) = 0;
C3(2,2) = R2/(R2+rCf2);

D3 = 0;

G3(1,1) = R1/(Cf1*(R1+rCf1));
G3(1,2) = 0;
G3(2,1) = 0;
G3(2,2) = R2/(Cf2*(R2+rCf2));

F3(1,1) = R1-(R1*R1/(R1+rCf1));
F3(1,2) = 0;
F3(2,1) = 0;
F3(2,2) = R2-(R2*R2/(R2+rCf2));


% Discrete time model
td = 0.1e-9;  %sampling time;

Phi1 = expm(A1*td);      % State transition matrix in discrete time
Tau1 = (expm(A1*td)-expm(A1*0))/A1*B1;
Tau_I1 = (expm(A1*td)-expm(A1*0))/A1*G1;

Phi2 = expm(A2*td);      % State transition matrix in discrete time
Tau2 = (expm(A2*td)-expm(A2*0))/A2*B2;
Tau_I2 = (expm(A2*td)-expm(A2*0))/A2*G2;

Phi3 = expm(A3*td);      % State transition matrix in discrete time
Tau3 = (expm(A3*td)-expm(A3*0))/A3*B3;
Tau_I3 = (expm(A3*td)-expm(A3*0))/A3*G3;



Kp1 = 500;           % PID coefficients for Output 1
Ki1 = 0.0001;
Kd1 = 10;
df1 = Vref1/Vin;

Kp2 = 500;           % PID coefficients for Output 2
Ki2 = 0.0001;
Kd2 = 10;
df2 = Vref2/Vin;

% loop simulation

tend = numcycle*Ts;
t = td:td:tend;                % Time vector stepping by sample size
t_Ir = Ts:Ts:tend;             % Time vector stepping by cycle size
x = zeros(3,3*length(t));
y = zeros(2,3*length(t));

Ve1 = zeros(1,3*numcycle);  % Error signal 1
DVe1 = zeros(1,3*numcycle); % Diff. error 1
```

```
IVe1 = zeros(1,3*numcycle); % Integral error 1
Ve2 = zeros(1,3*numcycle);  % Error signal 2
DVe2 = zeros(1,3*numcycle); % Diff. error 2
IVe2 = zeros(1,3*numcycle); % Integral error 2
dc = zeros(1,3*numcycle);
iref1_lr = zeros(1,3*numcycle);
iref2_lr = zeros(1,3*numcycle);

x0 = [0;0;0];
x_seg = zeros(3,Ts/td);
y_seg = zeros(2,Ts/td);
dc_seg = zeros(1,Ts/td);


dc0 = 0;
y(1:2,1) = C1*x0;
x(1:3,1) = Phi1*x0 + Tau1*Vin*dc0;
dc_seg_last1 = 0;
dc_seg_last2 = 0;

selector = 1;
notStartup = 0;

current_count1 = 1;              % keeps track of how long a parallel source
                                   has been on
current_injected_low1 = 0;       % 1 or 0
current_injected_high1 = 0;      % 1 or 0
current_count2 = 1;              % keeps track of how long a parallel source
                                   has been on
current_injected_low2 = 0;       % 1 or 0
current_injected_high2 = 0;      % 1 or 0
i = zeros(2,3*length(t));
i_seg = zeros(2,Ts/td);

for q = 2:1:3*numcycle
    if (q == numcycle+1) && (changeLoad == 1)   % Load current change from
                                                  500mA to 50mA

        Iref1 = 0.050;
        Iref2 = 0.050;
        R1 = Vref1/Iref1;
        R2 = Vref2/Iref2;

        Idc1 = Iref1+.100;
        Idc2 = Iref2+.100;

        A1(1,1) = -(rL*R1+rCf1*R1+rL*rCf1)/(L*(rCf1+R1));
        A1(1,2) = -R1/(L*(R1+rCf1));
        A1(1,3) = 0;
        A1(2,1) = R1/(Cf1*(rCf1+R1));
        A1(2,2) = -1/(Cf1*(rCf1+R1));
        A1(2,3) = 0;
        A1(3,1) = 0;
        A1(3,2) = 0;
        A1(3,3) = -1/(Cf2*(rCf2+R2));

        B1 = [1/L ; 0 ; 0];

        C1(1,1) = rCf1*R1/(rCf1+R1);
        C1(1,2) = R1/(R1+rCf1);
        C1(1,3) = 0;
        C1(2,1) = 0;
        C1(2,2) = 0;
        C1(2,3) = R2/(R2+rCf2);
```

```
D1 = 0;

G1(1,1) = -R1*rCf1/(L*(R1+rCf1));
G1(1,2) = 0;
G1(2,1) = R1/(Cf1*(R1+rCf1));
G1(2,2) = 0;
G1(3,1) = 0;
G1(3,2) = R2/(Cf2*(R2+rCf2));

F1(1,1) = R1*rCf1/(R1+rCf1);
F1(1,2) = 0;
F1(2,1) = 0;
F1(2,2) = R2-(R2*R2/(R2+rCf2));


A2(1,1) = -(rL*R2+rCf2*R2+rL*rCf2)/(L*(rCf2+R2));
A2(1,2) = 0;
A2(1,3) = -R2/(L*(R2+rCf2));
A2(2,1) = 0;
A2(2,2) = -1/(Cf1*(rCf1+R1));
A2(2,3) = 0;
A2(3,1) = R2/(Cf2*(rCf2+R2));
A2(3,2) = 0;
A2(3,3) = -1/(Cf2*(rCf2+R2));

B2 = [1/L ; 0 ; 0];

C2(1,1) = 0;
C2(1,2) = R1/(R1+rCf1);
C2(1,3) = 0;
C2(2,1) = rCf2*R2/(rCf2+R2);
C2(2,2) = 0;
C2(2,3) = R2/(R2+rCf2);

D2 = 0;

G2(1,1) = 0;
G2(1,2) = -R2*rCf2/(L*(R2+rCf2));
G2(2,1) = R1/(Cf1*(R1+rCf1));
G2(2,2) = 0;
G2(3,1) = 0;
G2(3,2) = R2/(Cf2*(R2+rCf2));

F2(1,1) = R1-(R1*R1/(R1+rCf1));
F2(1,2) = 0;
F2(2,1) = 0;
F2(2,2) = R2*rCf2/(R2+rCf2);


A3(1,1) = -1/(Cf1*(rCf1+R1));
A3(1,2) = 0;
A3(2,1) = 0;
A3(2,2) = -1/(Cf2*(rCf2+R2));

B3 = [0 ; 0];

C3(1,1) = R1/(R1+rCf1);
C3(1,2) = 0;
C3(2,1) = 0;
C3(2,2) = R2/(R2+rCf2);

D3 = 0;
```

```
    G3(1,1) = R1/(Cf1*(R1+rCf1));
    G3(1,2) = 0;
    G3(2,1) = 0;
    G3(2,2) = R2/(Cf2*(R2+rCf2));

    F3(1,1) = R1-(R1*R1/(R1+rCf1));
    F3(1,2) = 0;
    F3(2,1) = 0;
    F3(2,2) = R2-(R2*R2/(R2+rCf2));


    Phi1 = expm(A1*td);     % State transition matrix in discrete time
    Tau1 = (expm(A1*td)-expm(A1*0))/A1*B1;
    Tau_I1 = (expm(A1*td)-expm(A1*0))/A1*G1;

    Phi2 = expm(A2*td);     % State transition matrix in discrete time
    Tau2 = (expm(A2*td)-expm(A2*0))/A2*B2;
    Tau_I2 = (expm(A2*td)-expm(A2*0))/A2*G2;

    Phi3 = expm(A3*td);     % State transition matrix in discrete time
    Tau3 = (expm(A3*td)-expm(A3*0))/A3*B3;
    Tau_I3 = (expm(A3*td)-expm(A3*0))/A3*G3;
end

if (q == 2*numcycle+1) && (changeLoad == 1)     % Load current change
                                                  from 50mA to 500mA

    Iref1 = 0.500;
    Iref2 = 0.500;
    R1 = Vref1/Iref1;
    R2 = Vref2/Iref2;

    Idc1 = Iref1+1.000;
    Idc2 = Iref2+1.000;

    A1(1,1) = -(rL*R1+rCf1*R1+rL*rCf1)/(L*(rCf1+R1));
    A1(1,2) = -R1/(L*(R1+rCf1));
    A1(1,3) = 0;
    A1(2,1) = R1/(Cf1*(rCf1+R1));
    A1(2,2) = -1/(Cf1*(rCf1+R1));
    A1(2,3) = 0;
    A1(3,1) = 0;
    A1(3,2) = 0;
    A1(3,3) = -1/(Cf2*(rCf2+R2));

    B1 = [1/L ; 0 ; 0];

    C1(1,1) = rCf1*R1/(rCf1+R1);
    C1(1,2) = R1/(R1+rCf1);
    C1(1,3) = 0;
    C1(2,1) = 0;
    C1(2,2) = 0;
    C1(2,3) = R2/(R2+rCf2);

    D1 = 0;

    G1(1,1) = -R1*rCf1/(L*(R1+rCf1));
    G1(1,2) = 0;
    G1(2,1) = R1/(Cf1*(R1+rCf1));
    G1(2,2) = 0;
    G1(3,1) = 0;
    G1(3,2) = R2/(Cf2*(R2+rCf2));

    F1(1,1) = R1*rCf1/(R1+rCf1);
```

```
F1(1,2) = 0;
F1(2,1) = 0;
F1(2,2) = R2-(R2*R2/(R2+rCf2));


A2(1,1) = -(rL*R2+rCf2*R2+rL*rCf2)/(L*(rCf2+R2));
A2(1,2) = 0;
A2(1,3) = -R2/(L*(R2+rCf2));
A2(2,1) = 0;
A2(2,2) = -1/(Cf1*(rCf1+R1));
A2(2,3) = 0;
A2(3,1) = R2/(Cf2*(rCf2+R2));
A2(3,2) = 0;
A2(3,3) = -1/(Cf2*(rCf2+R2));

B2 = [1/L ; 0 ; 0];

C2(1,1) = 0;
C2(1,2) = R1/(R1+rCf1);
C2(1,3) = 0;
C2(2,1) = rCf2*R2/(rCf2+R2);
C2(2,2) = 0;
C2(2,3) = R2/(R2+rCf2);

D2 = 0;

G2(1,1) = 0;
G2(1,2) = -R2*rCf2/(L*(R2+rCf2));
G2(2,1) = R1/(Cf1*(R1+rCf1));
G2(2,2) = 0;
G2(3,1) = 0;
G2(3,2) = R2/(Cf2*(R2+rCf2));

F2(1,1) = R1-(R1*R1/(R1+rCf1));
F2(1,2) = 0;
F2(2,1) = 0;
F2(2,2) = R2*rCf2/(R2+rCf2);


A3(1,1) = -1/(Cf1*(rCf1+R1));
A3(1,2) = 0;
A3(2,1) = 0;
A3(2,2) = -1/(Cf2*(rCf2+R2));

B3 = [0 ; 0];

C3(1,1) = R1/(R1+rCf1);
C3(1,2) = 0;
C3(2,1) = 0;
C3(2,2) = R2/(R2+rCf2);

D3 = 0;

G3(1,1) = R1/(Cf1*(R1+rCf1));
G3(1,2) = 0;
G3(2,1) = 0;
G3(2,2) = R2/(Cf2*(R2+rCf2));

F3(1,1) = R1-(R1*R1/(R1+rCf1));
F3(1,2) = 0;
F3(2,1) = 0;
F3(2,2) = R2-(R2*R2/(R2+rCf2));
```

```
    Phi1 = expm(A1*td);     % State transition matrix in discrete time
    Tau1 = (expm(A1*td)-expm(A1*0))/A1*B1;
    Tau_I1 = (expm(A1*td)-expm(A1*0))/A1*G1;

    Phi2 = expm(A2*td);     % State transition matrix in discrete time
    Tau2 = (expm(A2*td)-expm(A2*0))/A2*B2;
    Tau_I2 = (expm(A2*td)-expm(A2*0))/A2*G2;

    Phi3 = expm(A3*td);     % State transition matrix in discrete time
    Tau3 = (expm(A3*td)-expm(A3*0))/A3*B3;
    Tau_I3 = (expm(A3*td)-expm(A3*0))/A3*G3;
end

if (y_seg(2,Ts/td) >= (0.99*Vref2)) && (y_seg(1,Ts/td) < (0.99*Vref1)) &&
notStartup == 0
    selector = 1;       % Initialization correction -- channels inductor
                          current into Output 1
elseif y_seg(1,Ts/td) >= (0.99*Vref1)
    notStartup = 1;
end

if selector == 1        % IF INDUCTOR CONNECTED TO OUTPUT 1
    y_avg1(q) = mean(y_seg(1,1:end));
    Ve1(q) = Vref1-y_avg1(q);  % error signal 1
    DVe1(q) = Ve1(q)-Ve1(q-1);
    IVe1(q) = IVe1(q-1)+ Ve1(q);

    y_avg2(q) = mean(y_seg(2,1:end));
    Ve2(q) = Vref2-y_avg2(q);  % error signal 2
    DVe2(q) = Ve2(q)-Ve2(q-1);
    IVe2(q) = IVe2(q-1)+ Ve2(q);

    dc(q) = Kp1*Ve1(q)+Ki1*IVe1(q)+Kd1*DVe1(q)+df1;
    iref1_Ir(q) = Vref1/R1;

    if dc(q)> 1;
        dc(q) = 1;
    elseif dc(q) <0;
        dc(q)=0;
    end

    % Limit inductor current to 2A
    dc_max(q) = L*(2-x_seg(1,Ts/td))/(Vin-y_seg(1,Ts/td))/Ts;
    if dc(q) > dc_max(q)
        dc(q) = dc_max(q);
    end

    if (y_seg(1,Ts/td) >= (0.99*Vref1))     % Inductor current limit
                                              during startup
        dc_max(q) = L*(Ilimit-x_seg(1,Ts/td))/(Vin-y_seg(1,Ts/td))/Ts;
        if dc(q) > dc_max(q)
            if dc_max >= 0
                dc(q) = dc_max(q);
            else
                dc(q) = 0;
            end
        end
        currentLimited = 1;
    end

    dc_seg(1:round(Ts/td*(dc(q)))) = 1;
    dc_seg(round(Ts/td*(dc(q))+1):Ts/td) = 0;

    for n = 0:1:Ts/td-1
```

```
if n == 0
    if dc_seg_last1 == 1
        x_seg(1:3,n+1) = Phi1*x_seg(1:3,Ts/td) +
        Tau1*Vin*dc_seg_last1 + Tau_I1*i_seg(1:2,Ts/td);
        y_seg(1:2,n+1) = C1*x_seg(1:3,n+1) + F1*i_seg(1:2,Ts/td);
    else
        if (x_seg(1,Ts/td) < (Idc1+0.0001)) && (y_seg(1,Ts/td) >
        (0.25*Vref1))
            x_seg(1,n+1) = Idc1;    % Keeps inductor current
                                      constant for PCCM
            x_seg(2:3,n+1) = Phi3*x_seg(2:3,Ts/td) +
            Tau3*Vin*dc_seg_last1 + Tau_I3*i_seg(1:2,Ts/td);
            y_seg(1:2,n+1) = C3*x_seg(2:3,n+1) +
            F3*i_seg(1:2,Ts/td);
        else
            x_seg(1:3,n+1) = Phi1*x_seg(1:3,Ts/td) +
            Tau1*Vin*dc_seg_last1 + Tau_I1*i_seg(1:2,Ts/td);
            y_seg(1:2,n+1) = C1*x_seg(1:3,n+1) +
            F1*i_seg(1:2,Ts/td);
        end
    end
else
    if dc_seg(n) == 1
        x_seg(1:3,n+1) = Phi1*x_seg(1:3,n) + Tau1*Vin*dc_seg(n) +
        Tau_I1*i_seg(1:2,n);
        y_seg(1:2,n+1) = C1*x_seg(1:3,n+1) + F1*i_seg(1:2,n);
    else
        if (x_seg(1,n) < (Idc1+0.0001)) && (y_seg(1,n) >
        (0.25*Vref1))
            x_seg(1,n+1) = Idc1;    % Keeps inductor current
                                      constant for PCCM
            x_seg(2:3,n+1) = Phi3*x_seg(2:3,n) +
            Tau3*Vin*dc_seg(n) + Tau_I3*i_seg(1:2,n);
            y_seg(1:2,n+1) = C3*x_seg(2:3,n+1) + F3*i_seg(1:2,n);
        else
            x_seg(1:3,n+1) = Phi1*x_seg(1:3,n) +
            Tau1*Vin*dc_seg(n) + Tau_I1*i_seg(1:2,n);
            y_seg(1:2,n+1) = C1*x_seg(1:3,n+1) + F1*i_seg(1:2,n);
        end
    end
end

if (y_seg(1,n+1) > Vmin1 && y_seg(1,n+1) < Vmax1 &&
current_injected_low1 == 0 && current_injected_high1 == 0) ||
(y_seg(1,n+1) > Vmin_high1 && current_injected_high1 == 1) ||
(y_seg(1,n+1) < Vmax_low1 && current_injected_low1 == 1) || (q <
0)
    i_seg(1,n+1) = 0;
    current_injected_low1 = 0;
    current_injected_high1 = 0;
    current_count1 = 1;

    if (y_seg(2,n+1) > Vmin2 && y_seg(2,n+1) < Vmax2 &&
    current_injected_low2 == 0 && current_injected_high2 == 0) ||
    (y_seg(2,n+1) > Vmin_high2 && current_injected_high2 == 1) ||
    (y_seg(2,n+1) < Vmax_low2 && current_injected_low2 == 1) ||
    (q < 0)
        i_seg(2,n+1) = 0;
        current_injected_low2 = 0;
        current_injected_high2 = 0;
        current_count2 = 1;
    elseif (y_seg(2,n+1) < Vmin_high2)
```

```
            if (current_count2*td) < 10e-9      % injected current
                                                ramps up for 10ns as long as
                                                source stays on
                i_seg(2,n+1) = current_count2*td*Imax/10e-9;
            else
                i_seg(2,n+1) = Imax;
            end
            current_injected_low2 = 0;
            current_injected_high2 = 1;
            current_count2 = current_count2 + 1;
        else
            if (current_count2*td) < 10e-9      % injected current
                                                ramps down for 10ns as long
                                                as source stays on
                i_seg(2,n+1) = -1*current_count2*td*Imax/10e-9;
            else
                i_seg(2,n+1) = -1*Imax;
            end
            current_injected_low2 = 1;
            current_injected_high2 = 0;
            current_count2 = current_count2 + 1;
        end
    elseif (y_seg(1,n+1) < Vmin_high1)
        if (current_count1*td) < 10e-9
            i_seg(1,n+1) = current_count1*td*Imax/10e-9;
        else
            i_seg(1,n+1) = Imax;
        end
        current_injected_low1 = 0;
        current_injected_high1 = 1;
        current_count1 = current_count1 + 1;

        if (y_seg(2,n+1) > Vmin2 && y_seg(2,n+1) < Vmax2 &&
        current_injected_low2 == 0 && current_injected_high2 == 0) ||
        (y_seg(2,n+1) > Vmin_high2 && current_injected_high2 == 1) ||
        (y_seg(2,n+1) < Vmax_low2 && current_injected_low2 == 1) ||
        (q < 0)
            i_seg(2,n+1) = 0;
            current_injected_low2 = 0;
            current_injected_high2 = 0;
            current_count2 = 1;
        elseif (y_seg(2,n+1) < Vmin_high2)
            if (current_count2*td) < 10e-9
                i_seg(2,n+1) = current_count2*td*Imax/10e-9;
            else
                i_seg(2,n+1) = Imax;
            end
            current_injected_low2 = 0;
            current_injected_high2 = 1;
            current_count2 = current_count2 + 1;
        else
            if (current_count2*td) < 10e-9
                i_seg(2,n+1) = -1*current_count2*td*Imax/10e-9;
            else
                i_seg(2,n+1) = -1*Imax;
            end
            current_injected_low2 = 1;
            current_injected_high2 = 0;
            current_count2 = current_count2 + 1;
        end
    else
        if (current_count1*td) < 10e-9
            i_seg(1,n+1) = -1*current_count1*td*Imax/10e-9;
        else
```

```
                i_seg(1,n+1) = -1*Imax;
            end
            current_injected_low1 = 1;
            current_injected_high1 = 0;
            current_count1 = current_count1 + 1;

            if (y_seg(2,n+1) > Vmin2 && y_seg(2,n+1) < Vmax2 &&
            current_injected_low2 == 0 && current_injected_high2 == 0) ||
            (y_seg(2,n+1) > Vmin_high2 && current_injected_high2 == 1) ||
            (y_seg(2,n+1) < Vmax_low2 && current_injected_low2 == 1) ||
            (q < 0)
                i_seg(2,n+1) = 0;
                current_injected_low2 = 0;
                current_injected_high2 = 0;
                current_count2 = 1;
            elseif (y_seg(2,n+1) < Vmin_high2)
                if (current_count2*td) < 10e-9
                    i_seg(2,n+1) = current_count2*td*Imax/10e-9;
                else
                    i_seg(2,n+1) = Imax;
                end
                current_injected_low2 = 0;
                current_injected_high2 = 1;
                current_count2 = current_count2 + 1;
            else
                if (current_count2*td) < 10e-9
                    i_seg(2,n+1) = -1*current_count2*td*Imax/10e-9;
                else
                    i_seg(2,n+1) = -1*Imax;
                end
                current_injected_low2 = 1;
                current_injected_high2 = 0;
                current_count2 = current_count2 + 1;
            end
        end
    end

    dc_seg_last1 = dc_seg(Ts/td);
    i(1,round((q-1)*Ts/td)+1:round(q*Ts/td))=i_seg(1,:);
    i(2,round((q-1)*Ts/td)+1:round(q*Ts/td))=i_seg(2,:);
    x(1,round((q-1)*Ts/td)+1:round(q*Ts/td))=x_seg(1,:);
    x(2,round((q-1)*Ts/td)+1:round(q*Ts/td))=x_seg(2,:);
    x(3,round((q-1)*Ts/td)+1:round(q*Ts/td))=x_seg(3,:);
    y(1,round((q-1)*Ts/td)+1:round(q*Ts/td))=y_seg(1,:);
    y(2,round((q-1)*Ts/td)+1:round(q*Ts/td))=y_seg(2,:);

    selector = 0;

else                        % IF INDUCTOR CONNECTED TO OUTPUT 2
    y_avg2(q) = mean(y_seg(2,1:end));
    Ve2(q) = Vref2-y_avg2(q);  % error signal 2
    DVe2(q) = Ve2(q)-Ve2(q-1);
    IVe2(q) = IVe2(q-1)+ Ve2(q);

    y_avg1(q) = mean(y_seg(1,1:end));
    Ve1(q) = Vref1-y_avg1(q);  % error signal 1
    DVe1(q) = Ve1(q)-Ve1(q-1);
    IVe1(q) = IVe1(q-1)+ Ve1(q);

    dc(q) = Kp2*Ve2(q)+Ki2*IVe2(q)+Kd2*DVe2(q)+df2;
    iref2_lr(q) = Vref2/R2;

    if dc(q)> 1;
        dc(q) = 1;
```

```
elseif dc(q) <0;
    dc(q)=0;
end

% Limit inductor current to 2A
dc_max(q) = L*(2-x_seg(1,Ts/td))/(Vin-y_seg(2,Ts/td))/Ts;
if dc(q) > dc_max(q)
    dc(q) = dc_max(q);
end

if (y_seg(1,Ts/td) >= (0.99*Vref1))     % Inductor current limit
                                          during startup
    dc_max(q) = L*(Ilimit-x_seg(1,Ts/td))/(Vin-y_seg(2,Ts/td))/Ts;
    if dc(q) > dc_max(q)
        if dc_max >= 0
            dc(q) = dc_max(q);
        else
            dc(q) = 0;
        end
    end
    currentLimited = 1;
end

dc_seg(1:round(Ts/td*(dc(q)))) = 1;
dc_seg(round(Ts/td*(dc(q))+1):Ts/td) = 0;

for n = 0:1:Ts/td-1
    if n == 0
        if dc_seg_last2 == 1
            x_seg(1:3,n+1) = Phi2*x_seg(1:3,Ts/td) +
            Tau2*Vin*dc_seg_last2 + Tau_I2*i_seg(1:2,Ts/td);
            y_seg(1:2,n+1) = C2*x_seg(1:3,n+1) + F2*i_seg(1:2,Ts/td);
        else
            if (x_seg(1,Ts/td) < (Idc2+0.0001)) && (y_seg(2,Ts/td) >
            (0.25*Vref2))
                x_seg(1,n+1) = Idc2;    % Keeps inductor current
                                          constant for PCCM
                x_seg(2:3,n+1) = Phi3*x_seg(2:3,Ts/td) +
                Tau3*Vin*dc_seg_last2 + Tau_I3*i_seg(1:2,Ts/td);
                y_seg(1:2,n+1) = C3*x_seg(2:3,n+1) +
                F3*i_seg(1:2,Ts/td);
            else
                x_seg(1:3,n+1) = Phi2*x_seg(1:3,Ts/td) +
                Tau2*Vin*dc_seg_last2 + Tau_I2*i_seg(1:2,Ts/td);
                y_seg(1:2,n+1) = C2*x_seg(1:3,n+1) +
                F2*i_seg(1:2,Ts/td);
            end
        end
    else
        if dc_seg(n) == 1
            x_seg(1:3,n+1) = Phi2*x_seg(1:3,n) + Tau2*Vin*dc_seg(n) +
            Tau_I2*i_seg(1:2,n);
            y_seg(1:2,n+1) = C2*x_seg(1:3,n+1) + F2*i_seg(1:2,n);
        else
            if (x_seg(1,n) < (Idc2+0.0001)) && (y_seg(2,n) >
            (0.25*Vref2))
                x_seg(1,n+1) = Idc2;    % Keeps inductor current
                                          constant for PCCM
                x_seg(2:3,n+1) = Phi3*x_seg(2:3,n) +
                Tau3*Vin*dc_seg(n) + Tau_I3*i_seg(1:2,n);
                y_seg(1:2,n+1) = C3*x_seg(2:3,n+1) + F3*i_seg(1:2,n);
            else
                x_seg(1:3,n+1) = Phi2*x_seg(1:3,n) +
                Tau2*Vin*dc_seg(n) + Tau_I2*i_seg(1:2,n);
```

```
            y_seg(1:2,n+1) = C2*x_seg(1:3,n+1) + F2*i_seg(1:2,n);
        end
    end
end

if (y_seg(2,n+1) > Vmin2 && y_seg(2,n+1) < Vmax2 &&
current_injected_low2 == 0 && current_injected_high2 == 0) ||
(y_seg(2,n+1) > Vmin_high2 && current_injected_high2 == 1) ||
(y_seg(2,n+1) < Vmax_low2 && current_injected_low2 == 1) || (q <
0)
    i_seg(2,n+1) = 0;
    current_injected_low2 = 0;
    current_injected_high2 = 0;
    current_count2 = 1;

    if (y_seg(1,n+1) > Vmin1 && y_seg(1,n+1) < Vmax1 &&
    current_injected_low1 == 0 && current_injected_high1 == 0) ||
    (y_seg(1,n+1) > Vmin_high1 && current_injected_high1 == 1) ||
    (y_seg(1,n+1) < Vmax_low1 && current_injected_low1 == 1) ||
    (q < 0)
        i_seg(1,n+1) = 0;
        current_injected_low1 = 0;
        current_injected_high1 = 0;
        current_count1 = 1;
    elseif (y_seg(1,n+1) < Vmin_high1)
        if (current_count1*td) < 10e-9      % injected current
                                    ramps up for 10ns as long as
                                    source stays on
            i_seg(1,n+1) = current_count1*td*Imax/10e-9;
        else
            i_seg(1,n+1) = Imax;
        end
        current_injected_low1 = 0;
        current_injected_high1 = 1;
        current_count1 = current_count1 + 1;
    else
        if (current_count1*td) < 10e-9      % injected current
                                    ramps down for 10ns as long
                                    as source stays on
            i_seg(1,n+1) = -1*current_count1*td*Imax/10e-9;
        else
            i_seg(1,n+1) = -1*Imax;
        end
        current_injected_low1 = 1;
        current_injected_high1 = 0;
        current_count1 = current_count1 + 1;
    end
elseif (y_seg(2,n+1) < Vmin_high2)
    if (current_count2*td) < 10e-9
        i_seg(2,n+1) = current_count2*td*Imax/10e-9;
    else
        i_seg(2,n+1) = Imax;
    end
    current_injected_low2 = 0;
    current_injected_high2 = 1;
    current_count2 = current_count2 + 1;

    if (y_seg(1,n+1) > Vmin1 && y_seg(1,n+1) < Vmax1 &&
    current_injected_low1 == 0 && current_injected_high1 == 0) ||
    (y_seg(1,n+1) > Vmin_high1 && current_injected_high1 == 1) ||
    (y_seg(1,n+1) < Vmax_low1 && current_injected_low1 == 1) ||
    (q < 0)
        i_seg(1,n+1) = 0;
        current_injected_low1 = 0;
```

```
            current_injected_high1 = 0;
            current_count1 = 1;
        elseif (y_seg(1,n+1) < Vmin_high1)
            if (current_count1*td) < 10e-9
                i_seg(1,n+1) = current_count1*td*Imax/10e-9;
            else
                i_seg(1,n+1) = Imax;
            end
            current_injected_low1 = 0;
            current_injected_high1 = 1;
            current_count1 = current_count1 + 1;
        else
            if (current_count1*td) < 10e-9
                i_seg(1,n+1) = -1*current_count1*td*Imax/10e-9;
            else
                i_seg(1,n+1) = -1*Imax;
            end
            current_injected_low1 = 1;
            current_injected_high1 = 0;
            current_count1 = current_count1 + 1;
        end
    else
        if (current_count2*td) < 10e-9
            i_seg(2,n+1) = -1*current_count2*td*Imax/10e-9;
        else
            i_seg(2,n+1) = -1*Imax;
        end
        current_injected_low2 = 1;
        current_injected_high2 = 0;
        current_count2 = current_count2 + 1;

        if (y_seg(1,n+1) > Vmin1 && y_seg(1,n+1) < Vmax1 &&
        current_injected_low1 == 0 && current_injected_high1 == 0) ||
        (y_seg(1,n+1) > Vmin_high1 && current_injected_high1 == 1) ||
        (y_seg(1,n+1) < Vmax_low1 && current_injected_low1 == 1) ||
        (q < 0)
            i_seg(1,n+1) = 0;
            current_injected_low1 = 0;
            current_injected_high1 = 0;
            current_count1 = 1;
        elseif (y_seg(1,n+1) < Vmin_high1)
            if (current_count1*td) < 10e-9
                i_seg(1,n+1) = current_count1*td*Imax/10e-9;
            else
                i_seg(1,n+1) = Imax;
            end
            current_injected_low1 = 0;
            current_injected_high1 = 1;
            current_count1 = current_count1 + 1;
        else
            if (current_count1*td) < 10e-9
                i_seg(1,n+1) = -1*current_count1*td*Imax/10e-9;
            else
                i_seg(1,n+1) = -1*Imax;
            end
            current_injected_low1 = 1;
            current_injected_high1 = 0;
            current_count1 = current_count1 + 1;
        end
    end
end

dc_seg_last2 = dc_seg(Ts/td);
i(1,round((q-1)*Ts/td)+1:round(q*Ts/td))=i_seg(1,:);
```

```
        i(2,round((q-1)*Ts/td)+1:round(q*Ts/td))=i_seg(2,:);
        x(1,round((q-1)*Ts/td)+1:round(q*Ts/td))=x_seg(1,:);
        x(2,round((q-1)*Ts/td)+1:round(q*Ts/td))=x_seg(2,:);
        x(3,round((q-1)*Ts/td)+1:round(q*Ts/td))=x_seg(3,:);
        y(1,round((q-1)*Ts/td)+1:round(q*Ts/td))=y_seg(1,:);
        y(2,round((q-1)*Ts/td)+1:round(q*Ts/td))=y_seg(2,:);

        selector = 1;
    end
end



t_long = 1:(3*numcycle*Ts/td);
t_lr_long = Ts:Ts:3*tend;

ref1 = Vref1*ones(1,length(t_long));
ref2 = Vref2*ones(1,length(t_long));

grid_l1 = Vmin1+zeros(1,length(t_long));
grid_l2 = Vmin_high1+zeros(1,length(t_long));
grid_h1 = Vmax1+zeros(1,length(t_long));
grid_h2 = Vmax_low1+zeros(1,length(t_long));

grid_l3 = Vmin2+zeros(1,length(t_long));
grid_l4 = Vmin_high2+zeros(1,length(t_long));
grid_h3 = Vmax2+zeros(1,length(t_long));
grid_h4 = Vmax_low2+zeros(1,length(t_long));

xscale = 0 : 0.2e-4 : tend*3;
yscale = -0.5 : 0.1 : 0.5;

figure
subplot(3,1,1), plot(t_long,y(1,:), t_long,grid_l1,':r', t_long,grid_l2,':g',
                t_long,grid_h1,':r', t_long,grid_h2,':g'), xlabel('Time'),
                ylabel('Output Voltage 1'), grid
subplot(3,1,2), plot(t_long,y(2,:), t_long,grid_l3,':r', t_long,grid_l4,':g',
                t_long,grid_h3,':r', t_long,grid_h4,':g'), xlabel('Time'),
                ylabel('Output Voltage 2'), grid
subplot(3,1,3), plot(t_long,x(1,:)), xlabel('Time'), ylabel('Inductor
                Current'), grid
```

Appendix C

MATLAB code for single-output buck converter with parallel source current injection

```
% Circuit parameters
Vin = 3.6;       % input voltage
L = 1e-6;        % inductor
rL = 50e-3 ;     % inductor parasitic resistance
Cf = 10e-6;      % output capacitor
rCf = 50e-3;     % output capacitor ESR
Vref = 1.8;      % reference voltage
Iref = 0.500;    % load current
R = Vref/Iref;   % output load resistor

Vmin = 0.99*Vref;           % Current injection lower voltage threshold
Vmin_high = 0.999*Vref;     % Current injection hysteretic lower turn-off
                              voltage threshold
Vmax = 1.01*Vref;           % Current injection upper voltage threshold
Vmax_low = 1.001*Vref;      % Current injection hysteretic upper turn-off
                              voltage threshold
Imax = 0.300;               % Maximum current injection level

numcycle = 200; % number of simulation cycles
fs = 3.2e6; % switching frequency
Ts = 1/fs; % sampling period

% Continuous time model
A1(1,1) = -(rL*R+rCf*R+rL*rCf)/(L*(rCf+R));     % Matrices used when inductor
                                                  current is positive
A1(1,2) = -R/(L*(R+rCf));
A1(2,1) = R/(Cf*(rCf+R));
A1(2,2) = -1/(Cf*(rCf+R));

B1 = [1/L ; 0];

C1 = [rCf*R/(rCf+R) R/(R+rCf)];

D1 = 0;

G1 = [-R*rCf/(L*(R+rCf)) ; R/(Cf*(R+rCf))];

F1 = R*rCf/(R+rCf);


A3 = -1/(Cf*(rCf+R));   % Matrices used when inductor current is held to zero
                          (not allowed to go negative)

B3 = 0;

C3 = R/(R+rCf);

D3 = 0;

G3 = R/(Cf*(R+rCf));

F3 = R*rCf/(R+rCf);


% Discrete time model
```

```
td = 0.1e-9;  % sampling time
Phi1 = expm(A1*td);     % State transition matrix in discrete time
Tau1 = (expm(A1*td)-expm(A1*0))/A1*B1;
Tau_I1 = (expm(A1*td)-expm(A1*0))/A1*G1;

Phi3 = expm(A3*td);     % State transition matrix in discrete time
Tau3 = (expm(A3*td)-expm(A3*0))/A3*B3;
Tau_I3 = (expm(A3*td)-expm(A3*0))/A3*G3;


Kp = 10;        % PID coefficients
Ki = 0.0001;
Kd = 3;

df = Vref/Vin;

tend = numcycle*Ts;
t = td:td:tend;             % Time vector stepping by sample size
t_lr = Ts:Ts:tend;          % Time vector stepping by cycle size
x = zeros(2,3*length(t));
y = zeros(1,3*length(t));
u = zeros(1,3*length(t));
iref = zeros(1,3*length(t));


Ve = zeros(1,3*numcycle);  % Error signal
DVe = zeros(1,3*numcycle); % Diff. error
IVe = zeros(1,3*numcycle); % Integral error
dc = zeros(1,3*numcycle);
iref_lr = zeros(1,3*numcycle);
dc_org = zeros(1,3*numcycle);
tab = zeros(1,3*numcycle);
status = zeros(1,3*numcycle);

x0 = [0;0];
x_seg = zeros(2,Ts/td);
y_seg = zeros(1,Ts/td);
dc_seg = zeros(1,Ts/td);


dc0 = 0;
y(1) = C1*x0;
x(1:2,1) = Phi1*x0 + Tau1*Vin*dc0;
dc_seg_last = 0;

current_count = 1;          % Keeps track of how long the current has been
                              injected, which determines how much is injected
current_injected = 0;       % 1 or 0, keeping track of whether or not current
                              is being injected
i = zeros(1,3*length(t));   % Injected current vector
i_seg = zeros(1,Ts/td);


for q = 2:1:3*numcycle

    % Change load current from 500mA to 50mA
    if q == numcycle+1;
        Kp = 40;          % Changing PID coefficients helps with transient
                            behavior
        Ki = 0.0001;
        Kd = 3;

        Iref = 0.050;   % load current
        R = Vref/Iref;  % output load resistor
```

```
    % Continuous time model
    A1(1,1) = -(rL*R+rCf*R+rL*rCf)/(L*(rCf+R));     % Matrices used when
                                                      inductor current is
                                                      positive
    A1(1,2) = -R/(L*(R+rCf));
    A1(2,1) = R/(Cf*(rCf+R));
    A1(2,2) = -1/(Cf*(rCf+R));

    B1 = [1/L ; 0];

    C1 = [rCf*R/(rCf+R) R/(R+rCf)];

    D1 = 0;

    G1 = [-R*rCf/(L*(R+rCf)) ; R/(Cf*(R+rCf))];

    F1 = R*rCf/(R+rCf);


    A3 = -1/(Cf*(rCf+R));   % Matrices used when inductor current is held
                              to zero (not allowed to go negative)

    B3 = 0;

    C3 = R/(R+rCf);

    D3 = 0;

    G3 = R/(Cf*(R+rCf));

    F3 = R*rCf/(R+rCf);


    % Discrete time model
    td = 0.1e-9;  % sampling time
    Phi1 = expm(A1*td);     % State transition matrix in discrete time
    Tau1 = (expm(A1*td)-expm(A1*0))/A1*B1;
    Tau_I1 = (expm(A1*td)-expm(A1*0))/A1*G1;

    Phi3 = expm(A3*td);     % State transition matrix in discrete time
    Tau3 = (expm(A3*td)-expm(A3*0))/A3*B3;
    Tau_I3 = (expm(A3*td)-expm(A3*0))/A3*G3;
end

% Change load current from 50mA to 500mA
if q == 2*numcycle+1;
    Kp = 10;        % Changing PID coefficients back to original values
    Ki = 0.0001;
    Kd = 3;

    Iref = 0.500;   % load current
    R = Vref/Iref;  % output load resistor

    % Continuous time model
    A1(1,1) = -(rL*R+rCf*R+rL*rCf)/(L*(rCf+R));     % Matrices used when
                                                      inductor current is
                                                      positive
    A1(1,2) = -R/(L*(R+rCf));
    A1(2,1) = R/(Cf*(rCf+R));
    A1(2,2) = -1/(Cf*(rCf+R));

    B1 = [1/L ; 0];
```

```
    C1 = [rCf*R/(rCf+R) R/(R+rCf)];

    D1 = 0;

    G1 = [-R*rCf/(L*(R+rCf)) ; R/(Cf*(R+rCf))];

    F1 = R*rCf/(R+rCf);


    A3 = -1/(Cf*(rCf+R));   % Matrices used when inductor current is held
                              to zero (not allowed to go negative)

    B3 = 0;

    C3 = R/(R+rCf);

    D3 = 0;

    G3 = R/(Cf*(R+rCf));

    F3 = R*rCf/(R+rCf);


    % Discrete time model
    td = 0.1e-9;  % sampling time
    Phi1 = expm(A1*td);     % State transition matrix in discrete time
    Tau1 = (expm(A1*td)-expm(A1*0))/A1*B1;
    Tau_I1 = (expm(A1*td)-expm(A1*0))/A1*G1;

    Phi3 = expm(A3*td);     % State transition matrix in discrete time
    Tau3 = (expm(A3*td)-expm(A3*0))/A3*B3;
    Tau_I3 = (expm(A3*td)-expm(A3*0))/A3*G3;
end

y_avg(q) = mean(y_seg);
Ve(q) = Vref-y_avg(q);      % Error signal
DVe(q) = Ve(q)-Ve(q-1);     % Diff. error
IVe(q) = IVe(q-1)+ Ve(q);   % Integral error

dc(q) = Kp*Ve(q)+Ki*IVe(q)+Kd*DVe(q)+df;    % Duty cycle

if dc(q)> 1;
    dc(q) = 1;
elseif dc(q) <0;
    dc(q)=0;
end

% Limit inductor current to 1A
dc_max(q) = L*(1-x_seg(1,Ts/td))/(Vin-y_seg(Ts/td))/Ts;
if dc(q) > dc_max(q)
    dc(q) = dc_max(q);
end

dc_seg(1:round(Ts/td*(dc(q)))) = 1;
dc_seg(round(Ts/td*(dc(q))+1):Ts/td) = 0;

for n = 0:1:Ts/td-1
    if n == 0       % For first step of each cycle, must use values from
                      last step of last cycle
        if dc_seg(Ts/td) == 1
            x_seg(1:2,n+1) = Phi1*x_seg(1:2,Ts/td) +
            Tau1*Vin*dc_seg(Ts/td) + Tau_I1*i_seg(Ts/td);
            y_seg(n+1) = C1*x_seg(1:2,n+1) + F1*i_seg(Ts/td);
        else
```

```
            % Prevents inductor current from going negative
            if (x_seg(1,Ts/td) < 0.0001) && (y_seg(Ts/td) > (0.25*Vref))
                x_seg(1,n+1) = 0;
                x_seg(2,n+1) = Phi3*x_seg(2,Ts/td) +
                Tau3*Vin*dc_seg(Ts/td) + Tau_l3*i_seg(Ts/td);
                y_seg(n+1) = C3*x_seg(2,n+1) + F3*i_seg(Ts/td);
            else
                x_seg(1:2,n+1) = Phi1*x_seg(1:2,Ts/td) +
                Tau1*Vin*dc_seg(Ts/td) + Tau_l1*i_seg(Ts/td);
                y_seg(n+1) = C1*x_seg(1:2,n+1) + F1*i_seg(Ts/td);
            end
        end
    else
        if dc_seg(n) == 1
            x_seg(1:2,n+1) = Phi1*x_seg(1:2,n) + Tau1*Vin*dc_seg(n) +
            Tau_l1*i_seg(n);
            y_seg(n+1) = C1*x_seg(1:2,n+1) + F1*i_seg(n);
        else
            % Prevents inductor current from going negative
            if (x_seg(1,n) < 0.0001) && (y_seg(n) > (0.25*Vref))
                x_seg(1,n+1) = 0;
                x_seg(2,n+1) = Phi3*x_seg(2,n) + Tau3*Vin*dc_seg(n) +
                Tau_l3*i_seg(n);
                y_seg(n+1) = C3*x_seg(2,n+1) + F3*i_seg(n);
            else
                x_seg(1:2,n+1) = Phi1*x_seg(1:2,n) + Tau1*Vin*dc_seg(n) +
                Tau_l1*i_seg(n);
                y_seg(n+1) = C1*x_seg(1:2,n+1) + F1*i_seg(n);
            end
        end
    end

    if (y_seg(n+1) > Vmin && y_seg(n+1) < Vmax && current_injected == 0)
    || (y_seg(n+1) > Vmin_high && y_seg(n+1) < Vmax_low) || (q < 200)
        i_seg(n+1) = 0;          % No current injected
        current_count = 1;
        current_injected = 0;
    elseif (y_seg(n+1) < Vmin_high)
        if (current_count*td) < 10e-9       % Positive current injected
                                              (top parallel source is on)
            % Injected current ramping up
            i_seg(n+1) = current_count*td*Imax/10e-9;
        else
            % Injected current constant at maximum value
            i_seg(n+1) = Imax;
        end
        current_count = current_count + 1;
        current_injected = 1;
    else
        if (current_count*td) < 10e-9       % Negative current injected
                                              (bottom parallel source is
                                              on)
            % Injected current ramping up
            i_seg(n+1) = -1*current_count*td*Imax/10e-9;
        else
            i_seg(n+1) = -1*Imax;     % Injected current constant at
                                        maximum value
        end
        current_count = current_count + 1;
        current_injected = 1;
    end
end
dc_seg_last = dc_seg(n+1);
i(round((q-1)*Ts/td)+1:round(q*Ts/td))=i_seg;
```

```
    x(1,round((q-1)*Ts/td)+1:round(q*Ts/td))=x_seg(1,:);
    x(2,round((q-1)*Ts/td)+1:round(q*Ts/td))=x_seg(2,:);
    y(round((q-1)*Ts/td)+1:round(q*Ts/td))=y_seg;
    iref(round((q-1)*Ts/td)+1:round(q*Ts/td))=Vref/R;
end


t_long = td:td:3*tend;      % For plotting in terms of time
t_lr_long = Ts:Ts:3*tend;

grid_l1 = Vmin*ones(1,length(t_long));        % Current injection thresholds
grid_l2 = Vmin_high*ones(1,length(t_long));
grid_h1 = Vmax*ones(1,length(t_long));
grid_h2 = Vmax_low*ones(1,length(t_long));

xscale = 0 : 0.2e-4 : tend*3;   % Different scales for looking at entire
                                  plot, just the 1st transient, or just the
                                  2nd transient
xscale1 = 6.1e-5 : 1e-6 : 6.45e-5;
xscale2 = 1.23e-4 : 1e-6 : 1.295e-4;
yscale = -0.5 : 0.1 : 0.5;
yscale1 = 1.76 : 0.001 : 1.84;
yscale2 = -0.4 : 0.1 : 0.4;

figure
subplot(3,1,1), plot(t_long,y, t_long,grid_l1,':r', t_long,grid_l2,':g',
                t_long,grid_h1,':r', t_long,grid_h2,':g'), xlabel('Time'),
                ylabel('Output Voltage'), title(['Imax =', num2str(Imax)]),
                grid
set(gca,'XTick',xscale)
%axis([6.1e-5 6.45e-5 1.76 1.84]), set(gca,'XTick',xscale1),
      set(gca,'YTick',yscale1)
%axis([1.23e-4 1.295e-4 1.76 1.84]), set(gca,'XTick',xscale2),
      set(gca,'YTick',yscale1)
%axis([0 tend*3 0 2])
subplot(3,1,2),plot(t_long,i), xlabel('Time'), ylabel('Injected Current'),
               grid
set(gca,'XTick',xscale)
%axis([6.1e-5 6.45e-5 -0.4 0.4]), set(gca,'XTick',xscale1),
      set(gca,'YTick',yscale2)
%axis([1.23e-4 1.295e-4 -0.4 0.4]), set(gca,'XTick',xscale2),
      set(gca,'YTick',yscale2)
%axis([0 tend*3 -0.5 0.5]), set(gca,'YTick',yscale)
subplot(3,1,3),plot(t_lr_long,dc), xlabel('Time'), ylabel('Control Input
               (Duty Cycle)'), grid
axis([0 tend*3 0 1])

figure
subplot(3,1,1), plot(t_long,x(1,:)), xlabel('Time'), ylabel('Inductor
                Current'), grid
set(gca,'XTick',xscale)
subplot(3,1,2), plot(t_long,x(2,:)), xlabel('Time'), ylabel('Capacitor
                Voltage'), grid
subplot(3,1,3), plot(t_long,iref), xlabel('Time'), ylabel('Set Load
                Current'), grid
```

Bibliography

[1] Amoroso, Luca, Mauro Donati, Xunwei Zhou, and Fred C. Lee. "Single Shot Transient Suppressor (SSTS) for High Current High Slew Rate Microprocessor." *Applied Power Electronics Conference and Exposition.* Dallas, Texas: IEEE, 1999. 284-288 v1.

[2] C., Sreekumar, and Vivek Agarwal. "A Circuit Theoretical Approach to Hybrid Mode Switching Control of a Pseudo CCM Boost Converter." *IEEE International Conference on Industrial Technology.* Mumbai, India: IEEE, 2006. 791-795.

[3] Franklin, Gene F., J. David Powell, and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems.* Prentice Hall, 2002.

[4] Kassakian, John G., Martin F. Schlecht, and George C. Verghese. *Principles of Power Electronics.* Reading, MA: Addison-Wesley, 1992.

[5] Lam, Yat-Hei, Wing-Hung Ki, Chi-Ying Tsui, and Philip K. T. Mok. "Single-Inductor Dual-Input Dual-Output Switching Converter for Integrated Battery Charging and Power Regulation." *Proceedings of the 2003 International Symposium on Circuits and Systems.* Bangkok, Thailand: ISCAS, 2003. 447-450 v3.

[6] Ma, Dongsheng. "Integrated Single-Inductor Multiple-Output and Adaptive DC-DC Conversion for Efficient Power Management." Diss. Hong Kong University of Science and Technology, 2003.

[7] Ma, Dongsheng, and Wing-Hung Ki. "Fast-Transient PCCM Switching Converter with Freewheel Switching Control." *IEEE Transactions on Circuits and Systems II: Express Briefs* 54.9 (2007): 825-829.

[8] Ma, Dongsheng, Wing-Hung Ki, and Chi-Ying Tsui. "A Pseudo-CCM/DCM SIMO Switching Converter with Freewheeling Switching." *IEEE Journal of Solid-State Circuits* 38.6 (2003): 1007-1014.

[9] Ma, Dongsheng, Wing-Hung Ki, Chi-Ying Tsui, and Philip K. T. Mok. "Single-Inductor Multiple-Output Switching Converters with Time-Multiplexing Control in Discontinuous Conduction Mode." *IEEE Journal of Solid-State Circuits* 38.1 (2003): 89-100.

[10] Patra, Pradipta, Susovon Samanta, Amit Patra, and Souvik Chattopadhyay. "A Novel Control Technique for Single-Inductor Multiple-Output DC-DC Buck Converters." *IEEE International Conference on Industrial Technology.* Mumbai, India: IEEE, 2006. 807-811.

[11] Pitel, Grant E., and Philip T. Krein. "Transient Reduction of DC-DC Converters Via Augmentation and Geometric Control." *IEEE Power Electronics Specialists Conference.* Orlando, Florida: IEEE, 2007. 1652-1657.

[12] Wong, Pit-Leong, Fred C. Lee, Xunwei Zhou, and Jiabin Chen. "VRM Transient Study and Output Filter Design for Future Processors." *Proceedings of the 24th Annual IEEE Industrial Electronics Society Conference.* Aachen, Germany: IEEE, 1998. 410-415 v1.

[13] Wu, Albert M., and Seth R. Sanders. "An Active Clamp Circuit for Voltage Regulation Module (VRM) Applications." *IEEE Transactions on Power Electronics* 16.5 (2001): 623-634.