

© 2020 Sergio Silva

MODELING OF A BUCK CONVERTER WITH FAST LOAD
TRANSIENT AUXILIARY CIRCUIT FOR CPUS

BY

SERGIO SILVA

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2020

Urbana, Illinois

Adviser:

Professor Pavan Kumar Hanumolu

ABSTRACT

In a buck converter, the output capacitance is the dominant factor that determines the magnitude of overshoot/undershoot in the supply voltage during a load transient. In applications such as powering a CPU, the CPU can make computational predictions for its future supply current levels which can be used to inform the power converter of future load transients. With information about future load transients, an auxiliary circuit can be used to deliver the needed energy to the load at the time of the informed load transient. With a communication scheme between the CPU and power supply, the power supply load transient response can be optimized allowing for a reduction in the total output capacitance, board space, and cost.

To my parents, for their love and support.

ACKNOWLEDGMENTS

I would like to thank Professor Hanumolu for giving me exposure to the field of analog circuit design and providing me a supportive working environment. I would also like to thank the members of the Integrated Circuits and Systems Group for the great discussions regarding work and life.

TABLE OF CONTENTS

CHAPTER 1	INTRODUCTION	1
1.1	Motivation	1
1.2	Outline	1
CHAPTER 2	BUCK CONVERTER DESIGN	2
2.1	Power Stage	2
2.2	Control Scheme	5
CHAPTER 3	CPU POWER SUPPLY REQUIREMENTS	7
3.1	Voltage and Current	7
3.2	Output Voltage Tolerance	8
3.3	Impedance Model	9
CHAPTER 4	AUXILIARY CIRCUIT	11
4.1	Description	11
CHAPTER 5	SIMULATION RESULTS	14
5.1	Model	14
5.2	Operation	16
5.3	Error Analysis	18
CHAPTER 6	FUTURE WORK	22
REFERENCES		23
APPENDIX A	VERILOG-AMS MODELS	24

CHAPTER 1

INTRODUCTION

1.1 Motivation

Buck converters are widely used for supplying CPUs. Usually, there is no communication between the power converter and the CPU, so the power converter is designed standalone. Since the CPU can make computational predictions of the magnitude of its future supply current, it can inform the power converter of the predicted load transients. In a typical fast load transient event, the required energy from the load comes from the output capacitor of the power converter which causes voltage overshoot/undershoot. By introducing an auxiliary circuit that is controlled by the CPU, the auxiliary circuit can immediately provide the needed energy to the load to keep the same supply voltage during a predicted load transient event. Therefore, the power converter's output capacitance can be lowered to achieve the same specifications as without the auxiliary circuit. Achieving a lower output capacitance saves board space and cost. In this thesis, we focus on modeling the operation of a specific auxiliary circuit and the interaction between power converter and CPU.

1.2 Outline

In Chapter 2, a review of buck converter design is given. Power supply design specifications for specific CPUs are given in Chapter 3. Chapter 4 covers the auxiliary circuit details and operation. Chapter 5 presents simulation results of the power converter implemented in Verilog-AMS. Chapter 6 addresses future work.

CHAPTER 2

BUCK CONVERTER DESIGN

2.1 Power Stage

2.1.1 Introduction

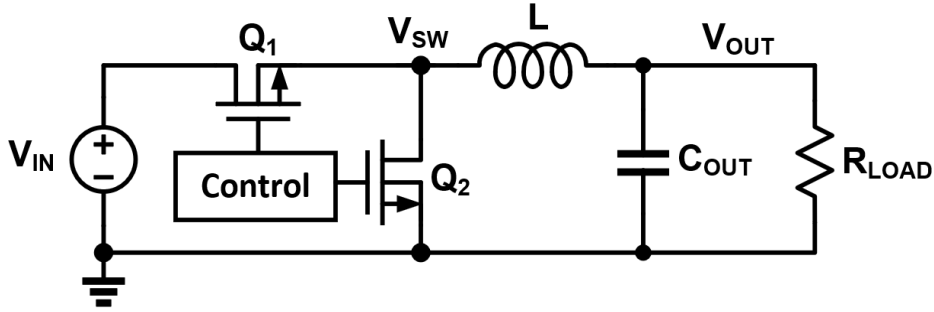


Figure 2.1: Buck converter

A buck converter is a switch-mode power converter that steps down the input voltage to a lower output voltage. A standard buck converter design is shown in Fig. 2.1. A buck converter is typically used in high current applications such as powering a CPU.

The power stage of a buck converter consists of the electrical components that see the main power flow from the input to the output: the inductor, L , the output capacitor, C_{OUT} , and the two MOSFET switches, Q_1 and Q_2 . The design of the power stage is dictated by the steady state and transient requirements of a power supply. The steady state requirements consist of the inductor current ripple and output voltage ripple. The transient requirements consist of the output voltage overshoot/undershoot and recovery time. This section will highlight the main design equations related to this thesis. For more details, the basic calculations for designing the power stage of a buck converter are mainly covered in [1].

2.1.2 Steady State Operation

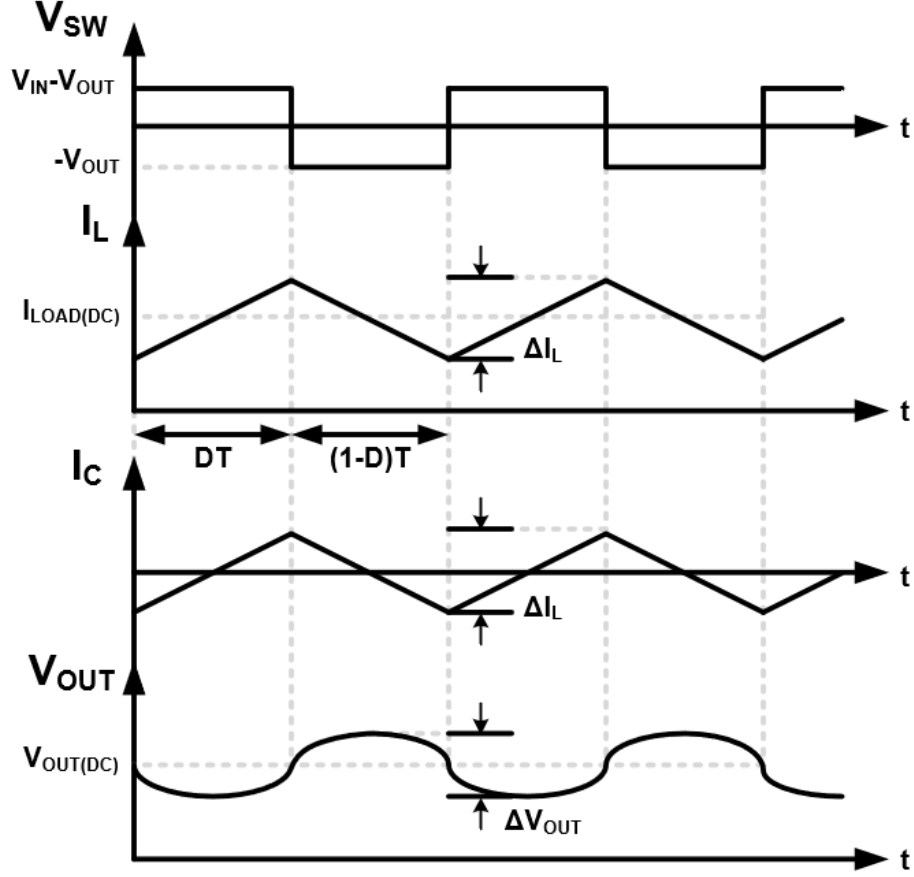


Figure 2.2: Buck converter steady state waveforms

The buck converter has two operation modes: on and off. The waveforms in Fig. 2.2 show the two modes. During the on state, Q_1 is on and Q_2 is off, so the input is connected to the inductor which charges the inductor. During the off state, Q_1 is off and Q_2 is on, so the input is disconnected from the inductor which discharges the inductor. By switching between the on and off states with a fixed frequency, F_{SW} , and duty cycle, D , the average current through the inductor can be controlled. The output capacitor works to filter out the high-frequency ripple, ΔI_L , of the inductor current. Therefore, the load current is given by the average current through the inductor current. The output voltage of a buck converter is given by

$$V_{OUT(DC)} = D \times V_{IN} \quad (2.1)$$

Since the output capacitor takes the AC current of the inductor, this generates voltage ripple across the output capacitor given by

$$\Delta V_{OUT} = \frac{\Delta I_L}{8 \times F_{SW} \times C_{OUT}} \quad (2.2)$$

2.1.3 Transient Response

The load will draw different amounts of current at different times and these current changes affect the performance the power converter. When the load current changes immediately by I_{STEP} amount, either positive or negative, the output voltage either undershoots or overshoots as shown in Fig. 2.3.

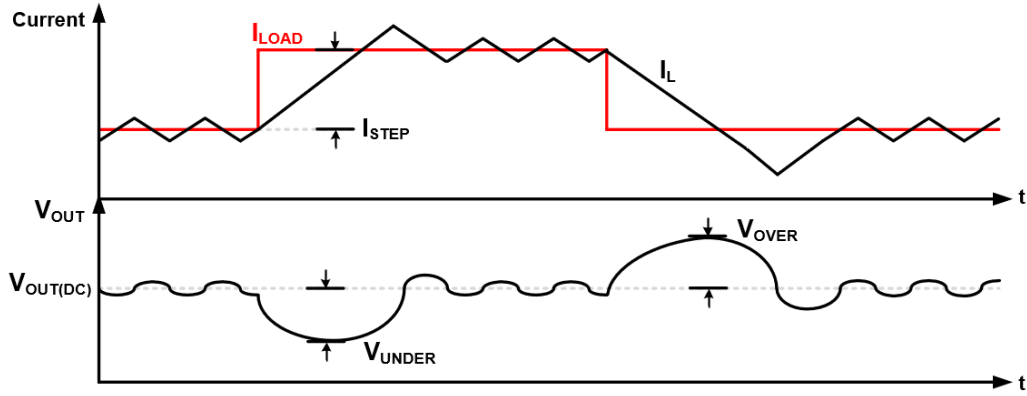


Figure 2.3: Buck converter load transient response

The current through the inductor cannot instantly change to the new load current value, so the change in current is provided by the output capacitor. During a positive load step, the output capacitor is discharging which causes undershoot, and the duty cycle is set to max, D_{MAX} , to quickly charge the inductor. During a negative load step, the output capacitor is charging which causes overshoot, and the duty cycle is set to zero to quickly discharge the inductor. Calculations for the output voltage overshoot and undershoot are approximately given as

$$V_{UNDER} = \frac{L \times I_{STEP}^2}{2 \times C_{OUT} \times D_{MAX} \times (V_{IN} - V_{OUT(DC)})} \quad (2.3)$$

$$V_{OVER} = \frac{L \times I_{STEP}^2}{2 \times C_{OUT} \times (V_{IN} - V_{OUT(DC)})} \quad (2.4)$$

2.2 Control Scheme

2.2.1 Digital Controller

A buck converter incorporates a feedback system in order to regulate the output voltage to a reference voltage. Compared to an analog controller, a digital controller has the advantages of being less sensitive to process technology and more easily programmable [2]. A buck converter with a digital controller is shown in Fig. 2.4.

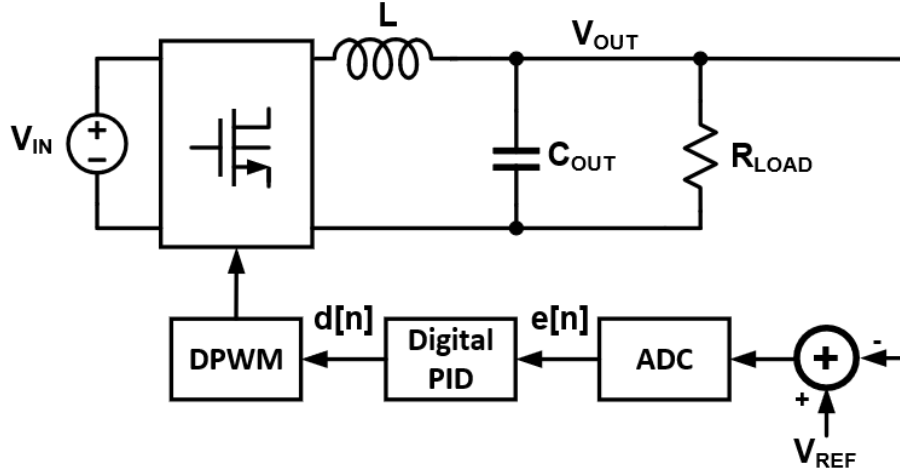


Figure 2.4: Buck converter with digital controller

The A/D converter samples the output voltage at the rate of the switching frequency and compares it to a reference voltage to generate a quantized error signal with quantization level V_Q . The error signal, $e[n]$, is processed by a digital PID controller to generate duty ratio values, $d[n]$, such that the error is minimized. The controller has the control law in the form of

$$d[n] = d[n-1] + a \cdot e[n] + b \cdot e[n-1] + c \cdot e[n-1] \quad (2.5)$$

where $e[n-i]$ and $d[n-i]$ represent the current and past error and duty ratio values, and where a , b , and c determine the characteristics of the controller. The DPWM converts the current duty ratio value, $d[n]$, into a PWM voltage signal with the corresponding duty ratio at the set switching frequency. The generated PWM signal drives the power switches, regulating the output voltage to $V_{OUT(DC)} = V_{REF}$.

2.2.2 Small Signal Analysis

The system is modeled as a continuous-time system. The block diagram of the system is shown in Fig. 2.5.

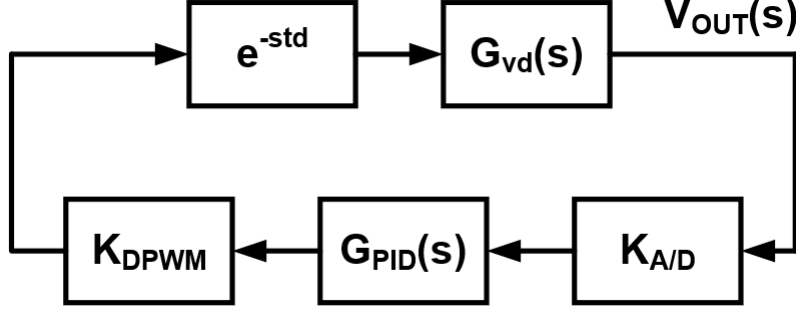


Figure 2.5: Block diagram of a buck converter with digital controller

The power stage is represented by the continuous-time transfer $G_{vd}(s)$ given as:

$$G_{vd}(s) = V_{IN} \frac{1}{1 + (L/R)s + (LC)s^2} \quad (2.6)$$

The power stage transfer function describes a two-pole system which is inherently unstable. Therefore, the controller is designed to make the system stable. The loop gain of the system with the digital controller is given as:

$$T_u(s) = K_{A/D} \cdot K_{DPWM} \cdot G_{PID}(s) \cdot G_{vd}(s) \cdot e^{-std} \quad (2.7)$$

where $K_{A/D} = 1/V_Q$ is the A/D converter gain, $K_{DPWM} = 1/n_{DPWM}$ is the DPWM gain, $G_{PID}(s)$ is the PID controller transfer function, and $t_d = 1/F_{SW}$ is the processing delay of the controller. Since we are modeling the system in continuous-time, $G_{PID}(s)$ is obtained by taking the inverse bilinear transformation of the discrete-time controller transfer function $G_{PID}(z)$. Direct Z -transform of (2.5) leads to

$$G_{PID}(z) = \frac{a + b \cdot z^{-1} + c \cdot z^{-2}}{1 - z^{-1}} \quad (2.8)$$

The PID controller coefficients are chosen to guarantee stability, and the selection method is described in [3].

CHAPTER 3

CPU POWER SUPPLY REQUIREMENTS

3.1 Voltage and Current

Each CPU has certain requirements on the design of its power supply in order for the CPU to function properly. This chapter covers certain specifications related to the power supply output voltage which is based on the Intel power supply design guide [4]. The power supply for a CPU typically comes in two forms: a voltage regulator module that plugs into the motherboard via connection pins or a voltage regulator that is permanently embedded onto the motherboard. The voltage regulator must support an 8-bit voltage code (VID) to control the output voltage with a usable range of 0.5 V to 1.6 V. The voltage regulator must be required to support a load with the following limits:

- A maximum continuous load current (I_{CTDC}) of 130 A.
- A maximum load current (I_{CCMAX}) of 150 A peak.
- A maximum load current step (I_{CCSTEP}) of 120 A.
- A maximum current slew rate (dI_{CC}/dt) of 300 A/ μ s.

The continuous load current (I_{CTDC}) is the DC load current value that the CPU can maintain indefinitely. The maximum load current (I_{CCMAX}) represents the maximum peak current that the CPU can draw. The maximum load current step (I_{CCSTEP}) represents the maximum change in current that the CPU can demand. The maximum current slew rate (dI_{CC}/dt) represents how fast the load current can change.

3.2 Output Voltage Tolerance

To ensure CPU safety and performance, the die voltage must always stay within a tolerance voltage band for a given output voltage code (VID). This requirement is specified for both steady state and transient voltages at the CPU die voltage. Operating at a die voltage below or above the specified tolerance band may cause damage to the CPU and results in reduced CPU performance and reliability issues. Fig. 3.1 shows the normalized load line regulation for voltage regulator with a resistance of $0.8\text{ m}\Omega$ from the path of the voltage regulator to the CPU die voltage. For the given load currents, the die voltage must stay within the specified lines of $V_{cc\text{ max}}$ and $V_{cc\text{ min}}$.

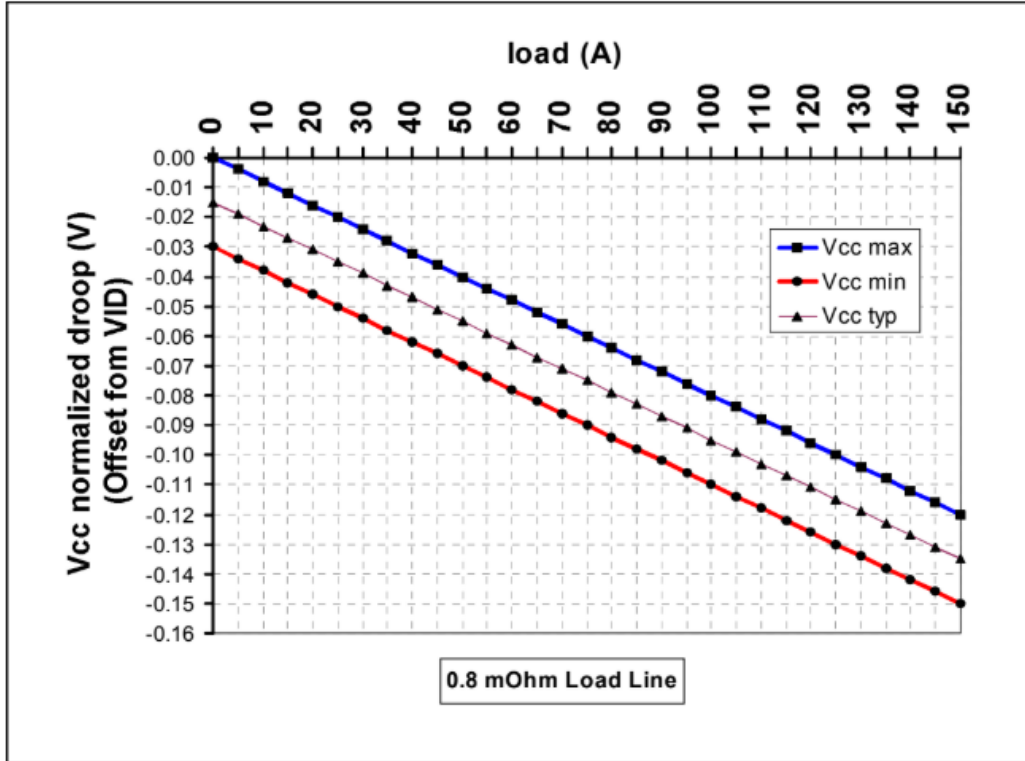


Figure 3.1: Load line tolerance band example

In order to validate the voltage regulator performance based on the output voltage tolerance, the following maximum load step size vs. load step rep rate is used for bench validation as shown in Fig. 3.2. The validation test tries to model the worst case behavior of the CPU for different load current step sizes and rep rate. This testing method allows for the voltage regulator to be designed standalone from the CPU.

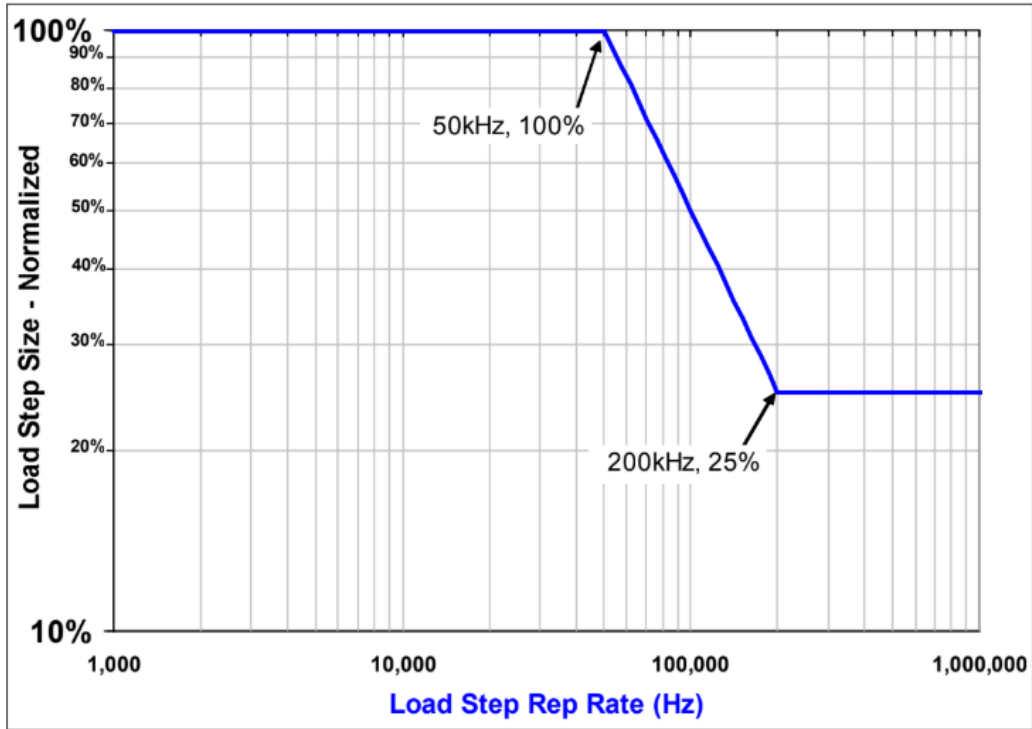


Figure 3.2: Load step size vs. load step rep rate

3.3 Impedance Model

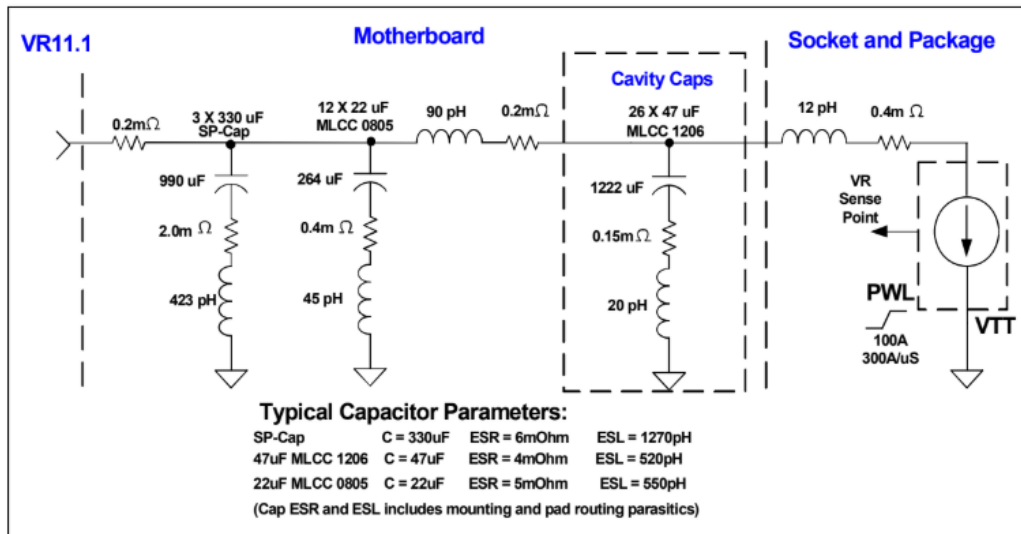


Figure 3.3: Power delivery impedance model path example

The validation test is performed by a test current load that models the output current as a PWM waveform with the current step given as the load

step size and the frequency given as the rep rate with a 50% duty cycle and current transition rates set as the maximum dI_{CC}/dt . The 100% load step size corresponds to the max value of I_{CCSTEP} . The voltage regulator must be stable for up to a rep rate of 2 MHz.

The impedance model represents the output capacitors of the voltage regulator that are located on the motherboard near the CPU socket, the cavity capacitors that are on the CPU socket, and the trace inductance and resistance from the voltage regulator output to the CPU die voltage. Fig. 3.3 shows an example of the Intel Xeon 5500 Platform CRB motherboard model. The type and value of the output capacitors depend on the voltage regulator design and load currents. The load line regulation test shown in Fig. 3.1 must be done using the power delivery path for a given design.

CHAPTER 4

AUXILIARY CIRCUIT

4.1 Description

In order to prevent overshoot/undershoot at the output of the voltage regulator, the output capacitors must not be charged/discharged. This is normally not the case in a standard buck converter during a load transient because the output capacitor charges/discharges to provide the instantaneous current change due to the load. Therefore, we introduce an auxiliary circuit that is controlled by the CPU to provide the instantaneous current change as shown in Fig. 4.1. Multiple auxiliary circuit schemes have been classified in [5] where this thesis covers a particular example of the load-informed scheme.

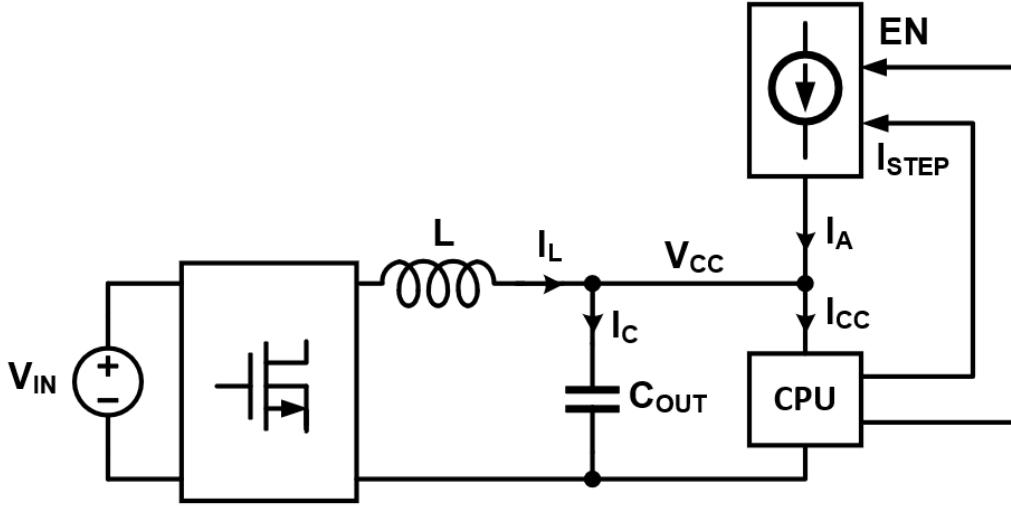


Figure 4.1: Buck converter with auxiliary circuit

During a load transient, I_{CC} changes from I_1 to I_2 by a change of I_{STEP} at time t_n as shown in Fig. 4.2. The CPU informs the auxiliary circuit of the predicted I_{STEP} magnitude and the predicted time t_n of the load transient. We are modeling prediction with no errors in the predicted current step

magnitude and time. The auxiliary circuit provides a current $I_A(t)$ such that $I_C(t) = 0$ during the time t_n to $t_n + \Delta t$ in order for V_{CC} to remain constant as shown in Fig. 4.1. The auxiliary circuit current follows the following relationship:

$$I_A(t) = \begin{cases} I_{STEP} - m \cdot (t - t_n) & \text{for } t \in [t_n, t_n + \Delta t] \\ 0 & \text{else} \end{cases} \quad (4.1)$$

where m is the slope at which the inductor current, $I_L(t)$, ramps up which is given as

$$m = \frac{dI_L}{dt} = \frac{V_{IN} - V_{CC}}{L} \quad (4.2)$$

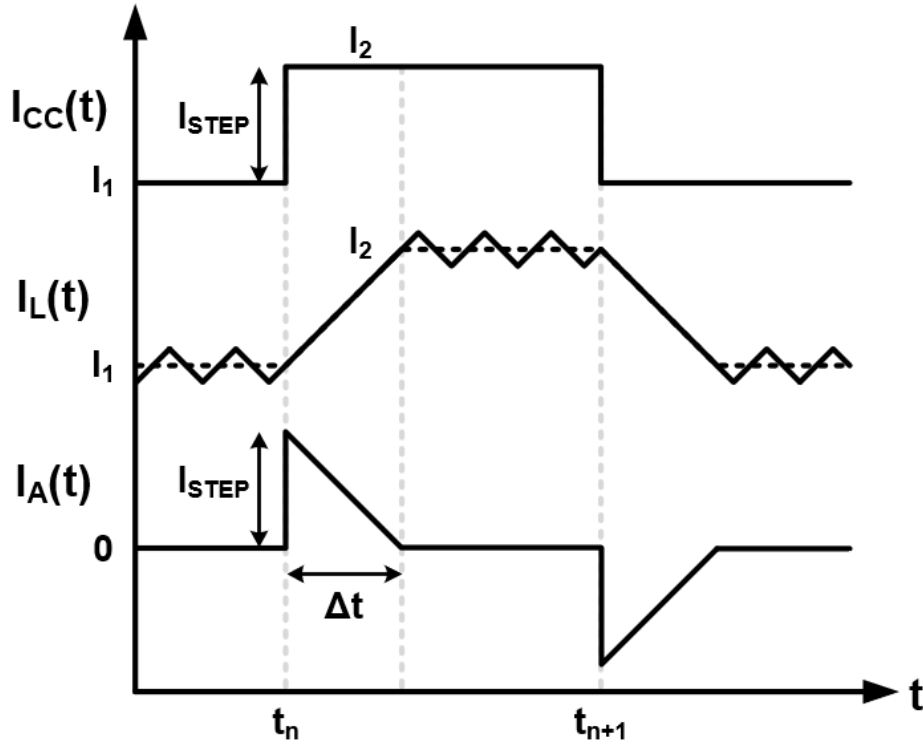


Figure 4.2: Operation during load transient

With the provided auxiliary current, we show that $I_C(t) = 0$. KCL at the V_{CC} node is given as

$$I_L(t) - I_C(t) + I_A(t) = I_{CC}(t) \quad (4.3)$$

$I_A(t)$ is given in (4.1) such that $I_L(t) + I_A(t) = I_{CC}(t)$ as shown in Fig. 4.2, and thus $I_C(t) = 0$.

The generation of the auxiliary circuit current depends on multiple variables; thus, this load-informed scheme relies on the accuracy of that information. In terms of the slope of the current, it depends on V_{IN} , V_{CC} , and L . V_{IN} and V_{CC} are accurately set in the power stage design. However, the inductor value experiences process variations; therefore, the inductor value must be continuously or frequently sensed in order to generate an accurate slope. In terms of the current magnitude and timing, it depends on the accuracy of the prediction scheme used by the CPU. Miscalculations of I_{STEP} or t_n result in a nonzero $I_C(t)$ during transition time causing overshoot/undershoot. The magnitude of that overshoot/undershoot depends on the error due to miscalculations.

CHAPTER 5

SIMULATION RESULTS

5.1 Model

5.1.1 Parameters

This chapter demonstrates a digitally controlled buck converter using an auxiliary circuit. Table 5.1 shows the chosen power stage and controller parameters that are used for all the presented simulation results.

Table 5.1: Digitally controlled buck converter parameters

Parameter	Value
V_{IN}	5 V
V_{CC}	1 V
F_{SW}	250 kHz
L	1 μ H
C_{OUT}	1 mF
E_{SR}	2 m Ω
Load	10 A — 40 A
Slew rate	≤ 300 A/ μ s
ADC resolution	9 bit
DPWM resolution	9 bit
Controller	Digital
Control mode	Voltage mode

The parameters are chosen to represent a buck converter used as an off-chip voltage regulator. Typically, voltage regulators for CPU applications are multi-phase buck converters; however we are only modeling a single phase buck converter in order to demonstrate the effects of the auxiliary circuit.

5.1.2 Modules

The test-bench for the model is shown in Fig. 5.1 where the modules are implemented in Verilog-AMS, and the test-bench is simulated using Spectre. The modules are as follows:

1. BUCK: represents the inductor, output capacitor, and switches of a buck converter
2. AUX: implements the auxiliary circuit
3. LOAD: models the CPU current and load transients by using an effective resistance. The resistance is computed using the nominal V_{CC} value of 1 V.
4. ADC: outputs an error code based on the difference of V_{REF} and V_{CC}
5. PID: generates an N -bit duty cycle code based on the error code history and PID algorithm
6. DPWM: converts an N -bit duty cycle code into a PWM waveform with the corresponding duty cycle
7. CLK: sets the switching frequency of the converter and clock frequency of the digital circuits

The Verilog-AMS code for each module can be found in the appendix.

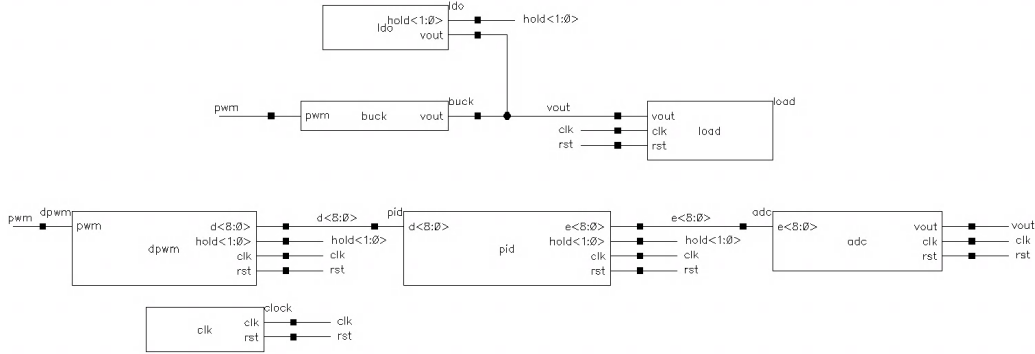


Figure 5.1: Schematic test-bench of Verilog-AMS modules

5.2 Operation

The following simulation results are for a load transient from 10 A to 40 A at $t = 2$ ms with a slew rate of $300 \text{ A}/\mu\text{s}$. The output voltage, load current, inductor current, and auxiliary circuit are shown in Fig. 5.2.

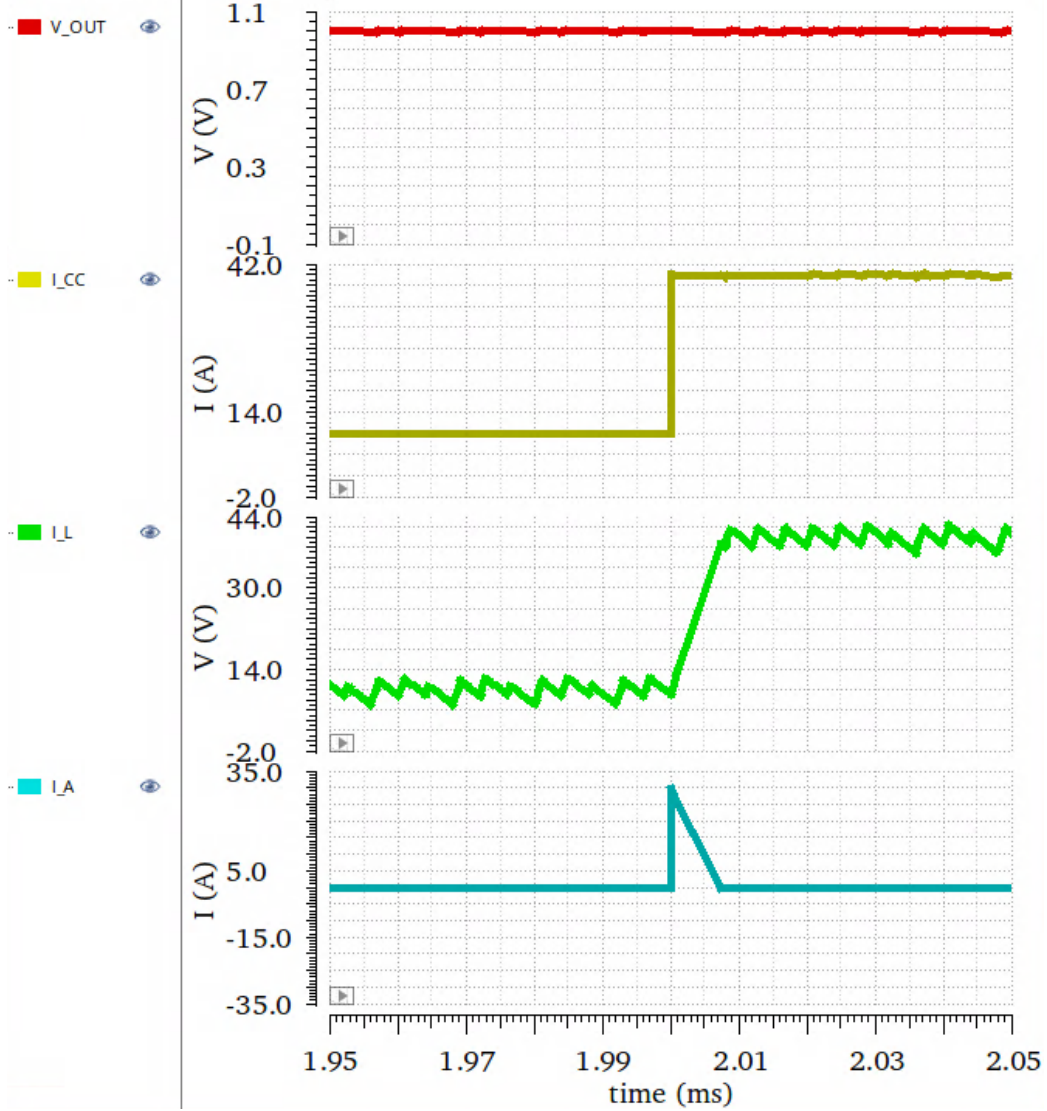


Figure 5.2: Buck converter with auxiliary circuit transient waveforms

The auxiliary circuit is provided with the the exact load transient current magnitude of 30 A and the exact load transient time of 2 ms. Therefore, the generated auxiliary circuit current guarantees there is no voltage droop. As expected, the output voltage remains constant during the load transient due to the auxiliary circuit current.

The auxiliary circuit is implemented as an ideal current source that is controlled by an enable signal. The enable signal is positive edge triggered and corresponds to the time at which the load transient happens. When the enable signal is triggered then the auxiliary provides the specific current in order to keep the output voltage stable. During the time the auxiliary circuit is providing current, the feedback controller is bypassed and the max duty cycle is set in order to ramp up the inductor current as shown in Fig. 5.2. After the auxiliary circuit is done, the control is returned back to the feedback controller.

For the same load transient, the output voltage is plotted in Fig. 5.3 for a buck converter with and without the auxiliary circuit.

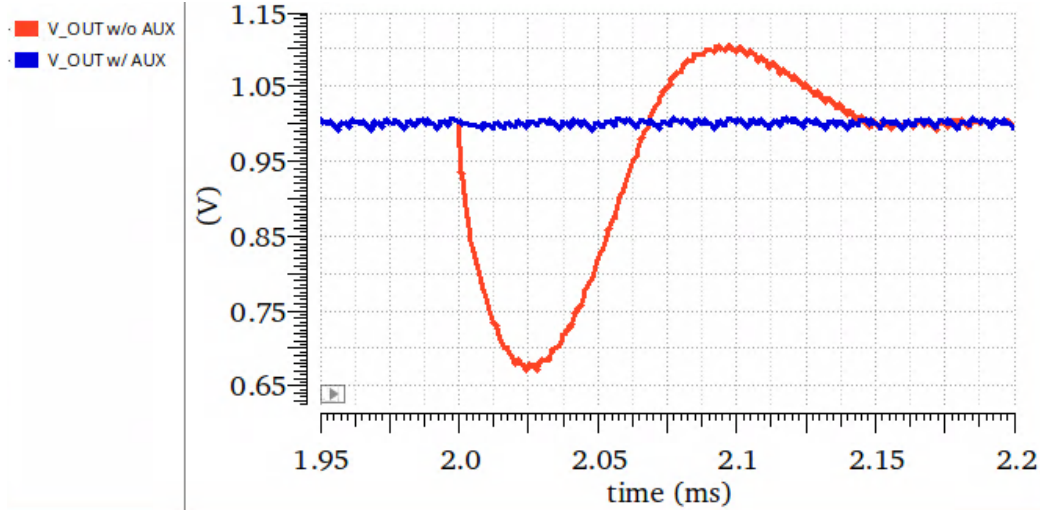


Figure 5.3: Output voltage transient waveforms

The output voltage for the buck converter without the auxiliary circuit follows the typical voltage transient waveform. The voltage undershoot is approximately 70 mV which is a 30% error. To achieve less than 5% voltage error, the output capacitance must be increased by a factor of approximately 6, which would increase the cost of the capacitor by a factor of 6. By having the auxiliary circuit instead of the output capacitor provide the current to the load, the output capacitance does not need to be unnecessarily large to meet output voltage tolerance specifications.

5.3 Error Analysis

5.3.1 Load Transient Current Magnitude Error

The following simulation results are for a load transient from 10 A to 40 A at $t = 2$ ms with a slew rate of $300 \text{ A}/\mu\text{s}$. We examine the voltage error due to inaccuracy in the predicted load transient current magnitude. The output voltage is plotted in Fig. 5.4 for predicted I_{STEP} magnitude values of 0 A to 45 A.

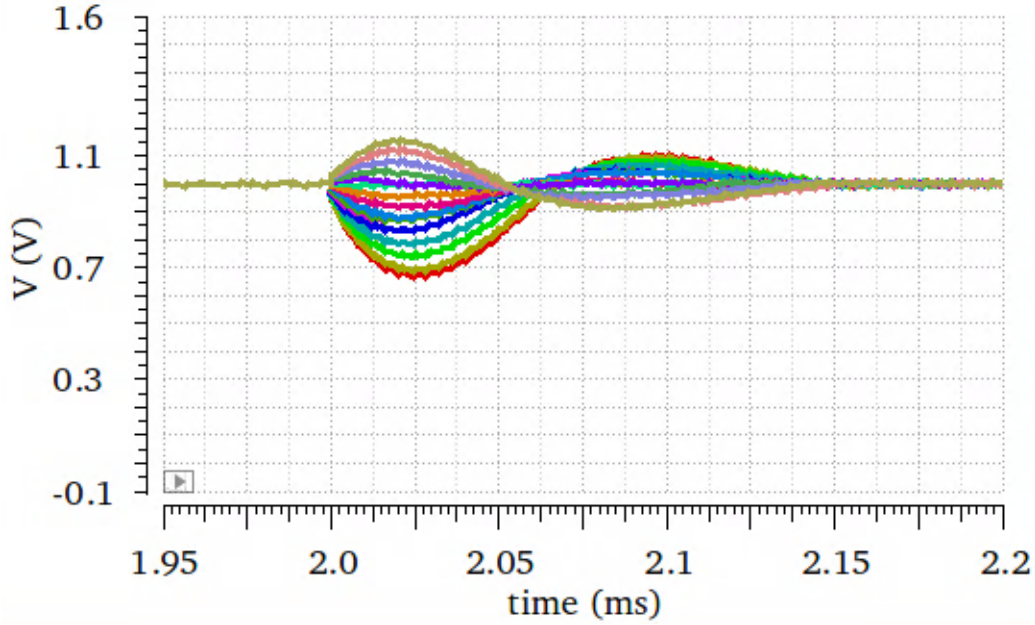


Figure 5.4: Output voltage for various predicted I_{STEP} magnitudes

An ideal predictor would give the predicted magnitude to be 30 A. However, we do not require predicted values to be ideal since if the prediction values are accurate within a percentage of the ideal value, then the voltage error would be bounded.

For some error in the predicted I_{STEP} magnitude, the auxiliary circuit current and inductor current do not add up exactly to the load current. Therefore, capacitor current sees the I_{STEP} error, the difference between $I_A(t)$ and $I_{CC}(t)$. However, this technique does not require the prediction to be ideal as long as the prediction error range is within the output voltage error margin. Fig. 5.5 shows the auxiliary circuit current for the various predicted I_{STEP} magnitude values.

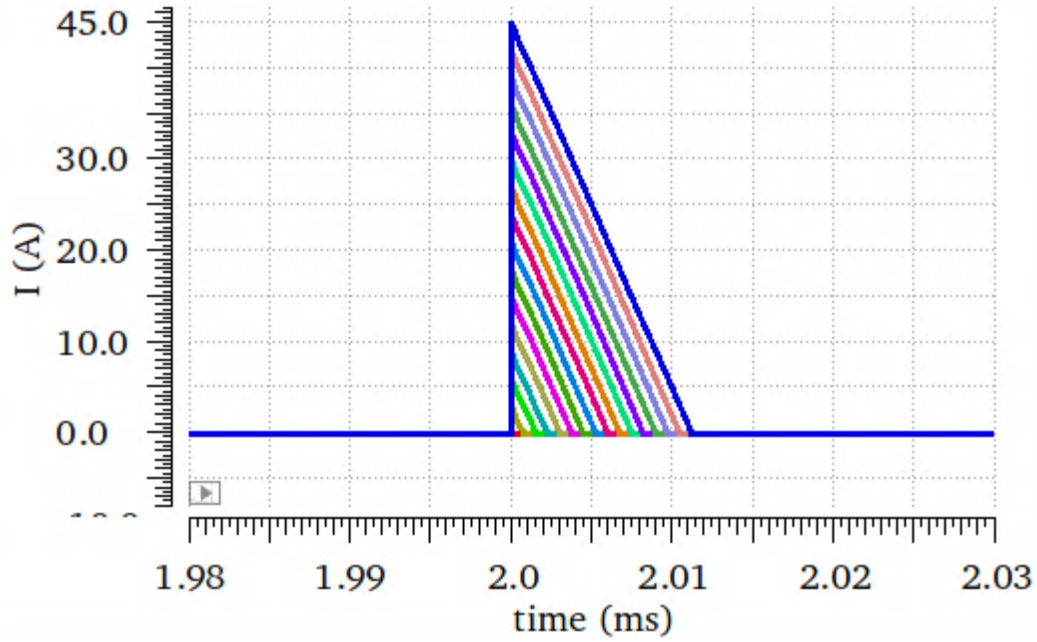


Figure 5.5: Auxiliary circuit current for various predicted I_{STEP} magnitudes

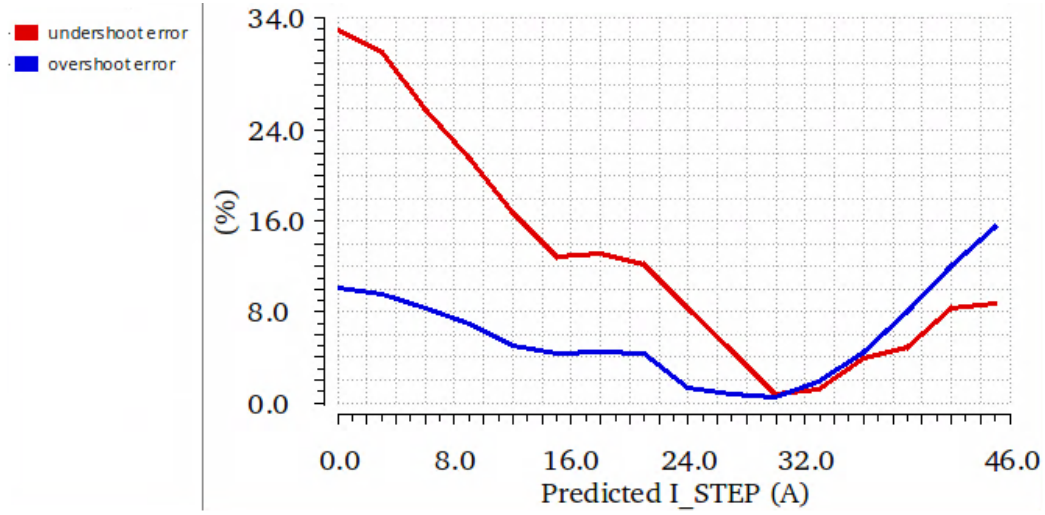


Figure 5.6: Voltage error vs. predicted I_{STEP} magnitude

For the given load transient, the overshoot and undershoot error percentages are plotted against the predicted I_{STEP} magnitude in Fig. 5.6. As expected, the minimum voltage error is when the predicted I_{STEP} magnitude corresponds to the actual I_{STEP} magnitude of 30 A. We see that the voltage error is bounded based on the accuracy of the prediction. For example, for predicted I_{STEP} magnitude values of 27 A — 36 A, the voltage error is $\leq 4\%$

5.3.2 Load Transient Time Error

The following simulation results are for a load transient from 10 A to 40 A at $t = 2$ ms with a slew rate of $300 \text{ A}/\mu\text{s}$. We examine the voltage error due to the inaccuracy in the predicted load transient time. The time error is defined as the error centered around the correct load transient time of 2 ms. In this case, the predicted I_{STEP} magnitude is set to the actual magnitude of 30 A. The output voltage is plotted in Fig. 5.7 for time errors from $-4 \mu\text{s}$ to $4 \mu\text{s}$.

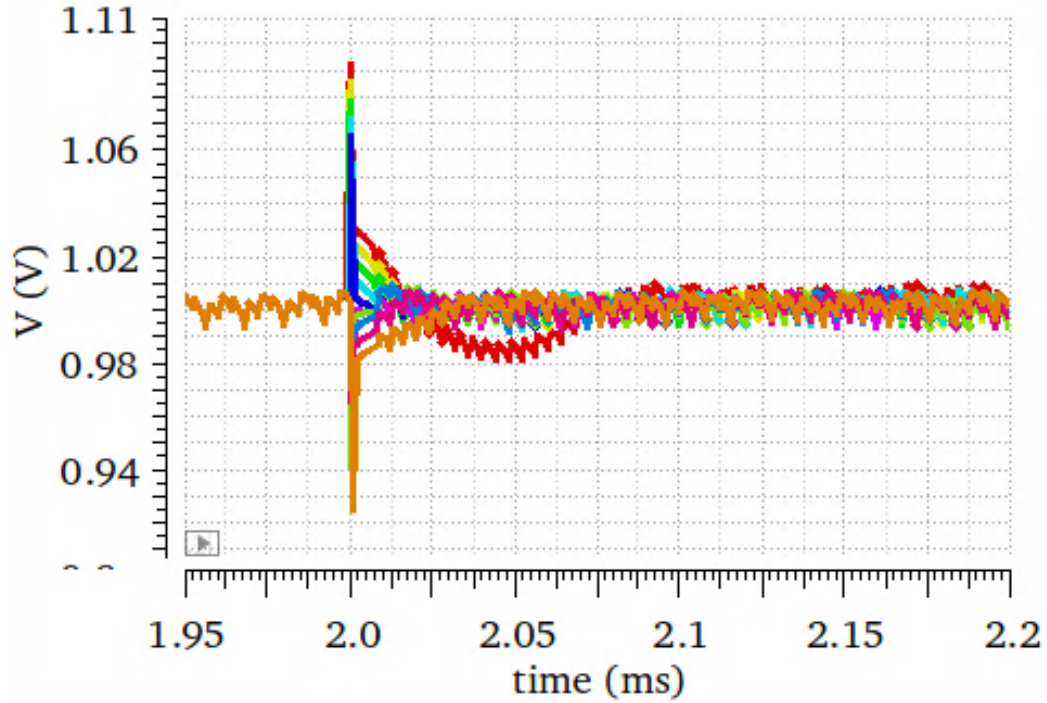


Figure 5.7: Output voltage for various predicted load transient times

A negative time error happens if the auxiliary circuit is enabled before the load transient occurs. In this case the output capacitor must provide the current change due to the auxiliary circuit since the load current has not changed yet. A positive time error happens if the auxiliary circuit is enabled after the load transient occurs. In this case the output capacitor must provide the current change due to the load current before the auxiliary circuit is enabled. In both cases, the capacitor sees a current spike with the width given as the time error. Fig. 5.8 shows the auxiliary circuit current for various predicted load transient times.

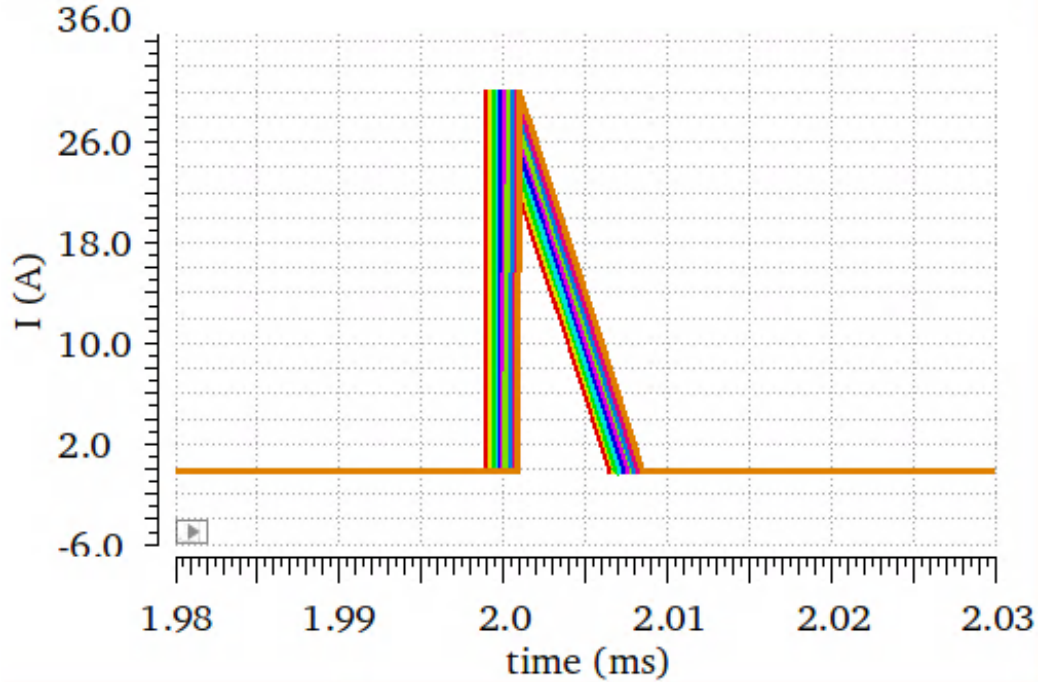


Figure 5.8: Auxiliary circuit current for various predicted load transient times

From the plots of Fig. 5.9, we see that the minimum voltage error is when the time error is 0. Compared to Fig. 5.6, it seems the voltage error is more sensitive from timing errors. Intuitively, this makes sense since for any time error, the output capacitor sees a large sudden change in current that it must provide for a duration given as the time error.

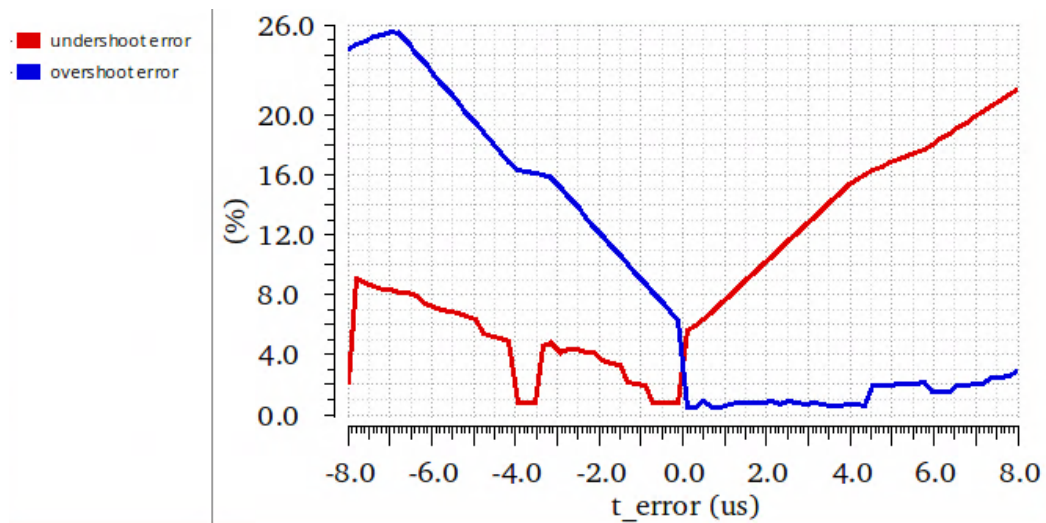


Figure 5.9: Voltage error vs. time error

CHAPTER 6

FUTURE WORK

In this thesis, we explored the functionality and limitations of the load-informed scheme with an auxiliary circuit for buck converters. We saw that the voltage transient can be significantly improved using this technique; however, there are problems that must be considered to make this technique practical.

The main limitations of the load-informed scheme come from the accuracy of the predicted current and time values. More development needs to be done on the power prediction algorithms used by CPUs in order to guarantee a high accuracy.

Furthermore, this thesis mainly focused on off-chip voltage regulators. However, with newer designs emerging for both on-chip and off-chip voltage regulators, the load slew rates are getting increasingly large such that the auxiliary circuit may not be sufficient to mitigate the voltage transients. Therefore, there is the need to validate the auxiliary circuit model with newer designs and reconsider any principle ideas to meet the newer specifications.

Lastly, in order to verify the design of the load-informed scheme, there is the need to simulate both the voltage regulator and CPU to guarantee proper functionality and results. Typically, the verification of the voltage regulator is done standalone without taking into account the CPU information. However, with the load informed scheme, the CPU communicates with the voltage regulator, and that communication channel must be properly verified. Therefore, a simulation tool must be built that models both the voltage regulator and CPU.

REFERENCES

- [1] B. Hauke, “Basic calculation of a buck converter’s power stage,” Texas Instruments, Aug. 2015.
- [2] A. Prodic and D. Maksimovic, “Design of a digital PID regulator based on look-up tables for control of high-frequency dc-dc converters,” in *2002 IEEE Workshop on Computers in Power Electronics, 2002. Proceedings.*, 2002, pp. 18–22.
- [3] L. Corradini, D. Maksimovic, P. Mattavelli, and R. Zane, *Digital Control of High-Frequency Switched-Mode Power Converters*. Hoboken, New Jersey: John Wiley Sons, Inc., 2015.
- [4] “Voltage regulator module (VRM) and enterprise voltage regulator-down (EVRD 11.1),” Intel, Sep. 2009.
- [5] Z. Shan, C. K. Tse, and S. Tan, “Classification of auxiliary circuit schemes for feeding fast load transients in switching power supplies,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 3, pp. 930–942, 2014.

APPENDIX A

VERILOG-AMS MODELS

```

`include "constants.vams"
`include "disciplines.vams"

module buck (pwm,vout);
    input pwm;
    output vout;
    electrical vout;
    electrical vin;
    electrical n1, n2;
    electrical IL, IC;

    parameter real VIN = 5;
    parameter real L = 1u;
    parameter real C = 2m;
    parameter real ESR = 2m;
    parameter real DCR = 0;
    parameter real tr = 100p;
    real switch;

    always @(pwm) begin
        if (pwm == 1'b1) switch = 1;
        else switch = 0;
    end

    analog begin
        if (analysis("ic"))
            I(n1,vout) <+ 0;
        else
            V(n1,vout) <+ L*ddt(I(n1,vout));

        if (analysis("ic"))
            V(n2) <+ 0;
        else
            I(n2) <+ C*ddt(V(n2));

        V(vin) <+ transition(VIN*switch,0,tr,tr);
        V(vin,n1) <+ DCR*I(vin,n1);
        V(vout,n2) <+ ESR*I(vout,n2);

        V(IL) <+ I(n1,vout);
        V(IC) <+ I(vout,n2);
    end
endmodule

```

Figure A.1: Buck converter power stage

```

`include "constants.vams"
`include "disciplines.vams"

`timescale 1ns/1ps
module adc (vout, e, clk, rst);

    parameter integer bits = 9;          // resolution (bits)
    parameter real vref = 1;             // reference voltage (V)
    parameter real vlsb = 40m;
    parameter real levels = 9;           // number of output voltage thresholds (odd)
    parameter real td = 0;               // delay from clock to output (ns)

    input vout, clk, rst;
    electrical vout;
    output signed [bits-1:0] e;
    reg signed [bits-1:0] e;
    reg over;
    real sample;
    real vq;
    real min_e;
    real i;
    real vrange;

    initial begin
        e = 0;
        over = 0;
    end
    always @(posedge clk) begin
        sample = V(vout);
        min_e = -1*(levels-1)/2;
        e = $rtoi(min_e);
        vrange = (levels-3)*vlsb;
        for (i=0; i<=levels-3; i=i+1) begin
            over = (vref - vrange/2 + i*vlsb - sample > 0);
            if (over) begin
                e=e+1;
                if (e == 0) e=1;
            end
        end
    end
endmodule

```

Figure A.2: ADC

```

`include "constants.vams"
`include "disciplines.vams"

`timescale 1ns/1ps
module pid (e, d, hold, clk, rst);

    parameter integer bits = 9;          // resolution (bits)
    parameter real kp = 30.54;           // proportional gain
    parameter real ki = 52.62;           // integral gain
    parameter real kd = 22.59;           // derivative gain

    parameter real a1 = 0;
    parameter real a2 = 0;

    input signed [bits-1:0] e;
    input [1:0] hold;
    input clk, rst;
    output signed [bits-1:0] d;

    reg signed [bits-1:0] d_prev;
    reg signed [bits-1:0] d_prev2;
    reg signed [bits-1:0] e_prev;
    reg signed [bits-1:0] e_prev2;

    integer max_d, min_d;
    reg en;
    wire signed [bits-1:0] comp;
    wire signed [bits:0] comp_next;
    wire signed [bits-1:0] d_test;
    wire signed [bits:0] d_next;

    initial begin
        e_prev = 0;
        e_prev2 = 0;
        max_d = 2**(bits-1)-1;
        min_d = -2**(bits-1);
        d_prev = min_d;
        d_prev2 = min_d;
        en = 0;
        en = #(1) 1;
    end

    always @ (posedge clk) begin
        if (hold == 0 && en == 1) begin
            e_prev2 <= e_prev;
            e_prev <= e;
            d_prev2 <= d_prev;
            d_prev <= d;
        end
    end

    assign comp_next = kp*e - ki*e_prev + kd*e_prev2 + a1*d_prev + a2*d_prev2;
    assign comp = (comp_next < -256) ? min_d:
        (comp_next > 255) ? max_d:
        comp_next[bits-1:0];
    assign d_next = {{d_prev[bits-1],d_prev[bits-1]},d_prev} + {{comp[bits-1],comp[bits-1]},comp};
    assign d = (d_next < -256) ? min_d:
        (d_next > 255) ? max_d:
        d_prev + comp;

endmodule

```

Figure A.3: Digital PID

```

`include "constants.vams"
`include "disciplines.vams"

`timescale 1s/1fs

module dpwm (d, pwm, hold, clk, rst);

    parameter integer bits = 9;
    parameter real FSW = 250K;

    input signed [bits-1:0] d;
    input [1:0] hold;
    output pwm;
    input clk, rst;

    reg pwm_d;
    reg signed [bits-1:0] counter;
    reg clk_count;
    real count_freq, count_period;

    initial begin
        pwm_d = 0;
        clk_count = 0;
        counter = -2**(bits-1);
        count_freq = 2**(bits)*FSW;
        count_period = 1/count_freq;
    end

    always begin
        #(count_period/2) clk_count = ~clk_count;
    end

    always @(posedge clk_count) begin
        if (d > counter) pwm_d <= 1;
        else pwm_d <= 0;
        counter <= counter + 1;
    end

    assign pwm = (hold==1)? 1: (hold==2)? 0 : pwm_d;

endmodule

```

Figure A.4: DPWM

```

`include "constants.vams"
`include "disciplines.vams"

`timescale 1s/1ps
module load (vout, clk, rst);
    input vout, clk, rst;
    electrical vout;

    parameter real VOUT = 1;
    parameter real I1 = 5;
    parameter real I2 = 20;
    parameter real tr = 15n;

    real R1, R2;
    integer SEL;

    initial begin
        R1 = VOUT/I1;
        R2 = VOUT/I2;
        SEL = 0;

        SEL = #(2m) 1;
        SEL = #(2m) 0;
    end

    analog begin
        I(vout) <+ V(vout)*transition((SEL==0) ? 1/R1 : 1/R2, 0, tr);
    end
endmodule

```

Figure A.5: Resistor load

```

`include "constants.vams"
`include "disciplines.vams"

`timescale 1s/1ps
module aux (vout, hold);

parameter real VIN = 12;
parameter real VOUT = 1;
parameter real L = 1u;
parameter real en_ldo = 1;

parameter real current_step = 5;
parameter real t_step = 2m;
parameter real t_error = -0.5u;

inout vout;
output reg [1:0] hold;
electrical vout;

real slew_rate = 350; // A/us
real trise, tfall, istep, sign;
reg SEL, EN;

initial begin
    hold = 0;
    SEL = 0;
    EN = 0;
    istep = 0;

    if(en_ldo) begin
        EN = #(t_step+t_error) 1;
        istep = current_step;
        EN = #(t_step-t_error) 1;
        istep = -1*current_step;
    end
end

always @ (posedge EN) begin
    trise = 15n;

    if (istep >= 0) begin
        tfall = istep*L/(VIN-VOUT);
        sign = -1;
        hold = 1;
    end
    else begin
        tfall = abs(istep)*L/VOUT;
        sign = 1;
        hold = 2;
    end

    SEL <= 1;
    SEL <= #(trise) 0;
    hold <= #(tfall) 0;

    EN <= #(1n) 0;
end

analog begin
    I(vout) <+ sign*transition((SEL==0)? 0 : abs(istep), 0, trise, tfall);
end

endmodule

```

Figure A.6: Auxiliary circuit

```

`include "constants.vams"
`include "disciplines.vams"

`timescale 1s/1ps
module clk (clk, rst);
    output clk, rst;
    reg clk, rst;

    parameter real FSW = 250K;
    real period;

    initial begin
        clk = 1;
        rst = 0;
        period = 1/FSW;
    end

    always #(period/2) clk = ~clk;
endmodule

```

Figure A.7: CLK