2017

MINIMUM TIME CONTROL OF PARALLELED BOOST CONVERTERS

Shishir Patel
Michigan Technological University, sjpatel2@mtu.edu

MINIMUM TIME CONTROL OF PARALLELED BOOST CONVERTERS

By

Shishir J. Patel

A THESIS

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Electrical Engineering

MICHIGAN TECHNOLOGICAL UNIVERSITY

2017

This thesis has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE in Electrical Engineering.

Department of Electrical and Computer Engineering

Thesis Advisor : *Dr. Wayne W. Weaver*

Committee Member : *Dr. Gordon G. Parker*

Committee Member : *Prof. John T. Lukowski*

Department Chair : *Dr. Daniel R. Fuhrmann*

Contents

List of Figures

List of Tables

To my grandfather, Mohan Patel.

Acknowledgments

Firstly, I would like to express my sincere gratitude to my advisor Dr. Wayne Weaver for the continuous support, for his immense patience, motivation, and knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for Masters study. It was truely a previledge for me to have an access to the laboratory and research facilities. It not only helped me to conduct the research but also provided an excellent opportunity to learn and experiment even beyond the scope of this thesis.

Besides my advisor, I would like to thank the rest of my thesis committee, Dr. Gordon Parker and Prof. John Lukowski for their insightful comments and encouragement.

Last but not the least, I would like to thank my family, especially my parents for encouragment and unconditional support that always inspired me pursue this path. I will alway be debtful for their sacrifices throuout my life.

Nomenclature

MTC	Minimum Time Control
SMC	Sliding Mode Control
IGBT	Insulated Gate Bipolar Transistor
MOSFET	Metal Oxide Surface Field Effect Transistor
MBC	Multiple Boost Converter
VCS	Voltage Controlled Source
CCS	Current Controlled Sources
CCM	Continuous Conduction Mode
DCM	Discontinuous Conduction Mode
SBC	Single Board Computer
ESR	Equivalent Series Resistance
PWM	Pulse Width Modulation
CPU	Central Processing Unit
GPU	Graphical Processing Unit

Abstract

Demand for electrification is booming in both, traditional and upcoming generations of technological advancements. One of the constituent blocks of these electrified systems is Power conversion. Power conversion systems are often constructed by paralleling multiple power converter blocks for high performance and reliability of overall system. An advanced control technique is developed with an aim to optimize system states of heterogeneous power converters within minimum time while maintaining feasible stress level on individual power converter blocks. Practical implementation of real-time controller and performance improvement strategies are addressed. Experimental results validating high performance control scheme, and sensitivity analysis of system states as measure of system robustness are also presented.

1 Introduction

1.1 Thesis objective

Switching mode power supplies are a fundamental component of modern power system, providing most efficient and controllable energy conversion solution. Over many decades of improvements made this technology even superior for all range of power applications. Along with the advancements in computational resources and wide range of control system, power electronics converters have dominated almost every aspect of power systems whether it be a power supply for a low-power microprocessor[1] or high-power wind-turbines[2][3] to electric vehicles[4][5]. Sophisticated control systems are replacing the role of conventional energy sources by incorporating electrical energy sources along with the power converters.

Significant interest in the range of medium to high power applications are, pulsed power systems where energy is supplied in form of pulses to the load, and quick response systems such as transient stabilization[3]. Often times, high power applications are implemented through combination of several medium power modules to share the overall load[6] [7]. Increasing interest in electrification of naval vessels, amphibious vehicles and motor vehicles demand high speed performance in their operations. A pulsed radar system that conventionally operates through high-frequency oscillators and resonators is also in need for more efficient and less bulky solution for low to mid range detection applications.

To implement above mentioned applications through power converters, a sophisticated control approach is needed. High power applications consist large number of

medium power converters that share power load depending on their capabilities. It is also important to note that these modules may or may not be identical in nature. So, the control system should also consider non-symmetric nature of the system. What makes a pulsed power application most challenging is the power level and time duration of pulse. So, it makes overall system very vulnerable to withstand pulses of power that force system to operate at its extreme level. While pulse power applications take minimum time for state transitions, it is also important to facilitate that performance in least amount of time possible so that the overall speed of operation can be increased.

To fulfill the demands for high-power high-speed application, prime objective of this thesis is to develop a control method/architecture that can implement minimum time control for multiple paralleled boost converters in real-time.

1.2 Previous art

Numerous research endeavors have been made in providing a feasible solution for the minimum time control problem. Varying from low to high power applications, those solutions fit into a well defined criteria for which the control system/approach was designed. Some of those previous works are briefly discussed in this section.

One of the implementations [8] focusing on minimum time control of single stage boost converter is through finding a solution for a trajectory with boundary conditions and storing the calculated trajectory in a static memory. This kind of approach using pre-calculated trajectories take less time for computation while

running the system but the limitation of the approach is that it can not be extended to n-number of parallel boost converter based system. As the number of stages grow, the complexity to find an analytical solution for an optimal control trajectory grows rapidly and hence it can not be a viable option for quick real-time transition. Another important factor to be considered for an off-line solution method is that the accuracy of real-time transition depend on the available memory storage. So, the available on-board memory becomes an important factor in hardware selection.

Another research [9][10][11] focused on multiple power converter stages provide minimum time control solution through mixed-signal compensatory design. One prominent advantage of this method is that the complete control system is implemented through hardware and does not require an on-line computational burden. The control method primarily designed for low-power applications does not consider non-homogeneity of the system. So, all of the power converter stages are assumed identical. Due to the homogeneity in system, a control can be designed merely by considering only one stage and expanding to whole system of n number of stages. It assumes equal sharing of current among the stages and hence makes it very simplified for low power applications. Although, this method provides least computationally expensive solution, it do not fit to most of practical power system application.

Other relevant works [12][13][14][15] that address the minimum time control or similar approach for a specific application and assumption. But most of the methods either consider a simplified topology or non real time solution. There are no solutions available for multiple converters/interleaved converters to have

lightweight, scalable and real-time approach that consider non-ideal system parameters. Hence, it is prime focus of this thesis to develop a control method that address all of the short-comings of the methods discussed here.

1.3 Thesis organization

A detailed mathematical modeling of a boost converter is derived for different modes of operation, Continuous Conduction Mode (CCM) and Discontinuous Conduction Mode (DCM). The discrete modeling approach is extended for multiple/paralleled Boost converters interfaced to a resistive load. The mathematical implementation derived in the chapter 2 is used for iteration of paralleled boost converter in Minimum Time Control (MTC) operation.

Chapter 3 illustrates the implementation of Sliding Mode Control (SMC) as one of the robust control technique for paralleled boost converter. A concept of contribution and non-identical sharing of inductor currents is introduced as key aspect of MTC. The core content of the thesis, MTC algorithm, is elaborated in this chapter and the simulation results are also illustrated to compare the performance of MTC with SMC.

A real-time implementation of the control system is described in chapter 4. The architecture of the discrete control system for Hardware-in-loop (HIL) emulation of paralleled boost converters under MTC is described. Real-time experimental results for SMC and MTC are also juxtaposed for time domain and phase trajectory point of view. Sensitivity analysis and the experimental results are also demonstrated to highlight robustness of the MTC operation.

Final chapter 5 summarize the reported work with conclusive remarks and focuses on several key aspects of the proposed method that can significantly improve the performance (speed of response) of the overall system.

2 Overview of Boost converters in and as DC Microgrid

2.1 Mathematical model of a Boost converter

Boost converter/regulator is one of the constituent blocks of a DC microgrid that translate input voltage to a higher level output voltage [16]. The voltage source boost converter shown in Figure 2.1, has voltage source V_s, ideal controlled electronic switch (IGBT/MOSFET) S, and ideal diode D as an uncontrolled switch. Duty cycle(d) of the PWM signal applied at the gate terminal determine ON time dT, and OFF time $(1-d)T$ of the switch. Where, T represents the time period of applied PWM signal. Energy storing passive elements, inductance L and capacitance C, have their equivalent series resistances (ESR) represented as R_L and R_c respectively. Similarly, electronic switch and diode have their ON resistance modeled as R_{SW} and R_D accordingly.

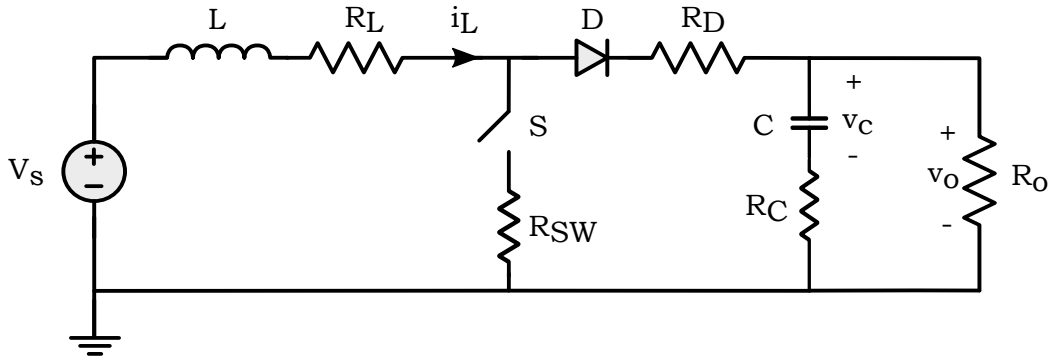


Figure 2.1: Voltage source boost converter with constant resistance

Resistances of electronic devices depends on hardware specifications and include losses in the system model. However, one of the fundamental parameters at-

tributed to electronic devices, cut-off voltage, is omitted from the model Boost converter. For high power applications the effect of any voltage drop of scale of several hundreds of millivolts tend to alter overall system performance at no significant level. One another reason, elaborated in detail in section (3.2.1), to eliminate the terms with non-significant impact on accuracy of system model is to reduce simulation time while running in real-time.

Assuming strictly non-linear and ideal switching of a transistor S and diode D, the boost converter circuit operates in a different manner according to the discrete state of the switch and diode. In general, boost converter has two modes of operations [17] [18], namely,

1. Continuous conduction mode (CCM)
2. Discontinuous conduction mode (DCM)

CCM is the mode of operation in which, inductor current i_L remains always non-zero and positive for any switch period; whereas, during DCM inductor i_L can reduce to zero during OFF period of switch. Depending upon switch state and inductor current there exist multiple combinations of the boost converter circuit. In order to simplify discrete modeling of the boost converter circuit, these combinations are lumped with the linear model of individual circuit combination through binary variables designated as A and B. Where,

$$A = \begin{cases} 1, & Switch\ S\ is\ ON \\ 0, & Switch\ S\ is\ OFF \end{cases} \tag{2.1}$$

$$B = \begin{cases} 1, & \textit{inductor current } i_L \in (0, \infty) \\ 0, & \textit{inductor current } i_L = 0 \end{cases} \tag{2.2}$$

2.1.1 Discrete modeling of Boost converter in CCM

According to equations (2.1) and (2.2), binary variable $B = 1$ and $A = 1/0$ represents the state of switch S. Two different circuit typologies resulting from this combination are illustrated in Figure 2.2-2.3.

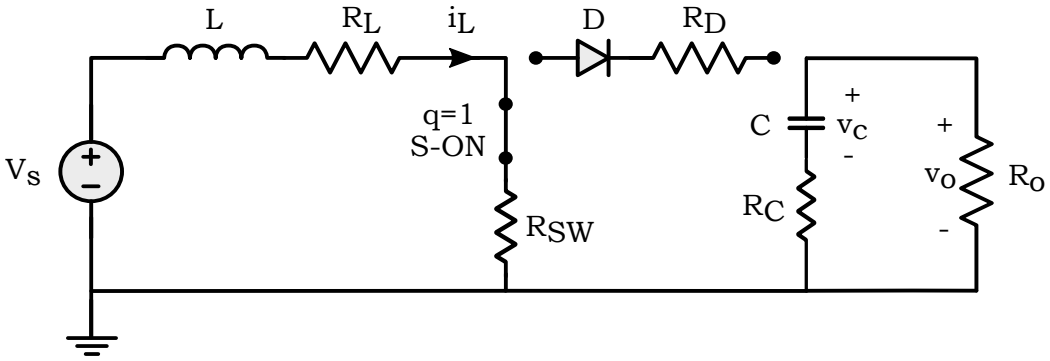


Figure 2.2: Boost converter when $A = 1$,$B = 1$

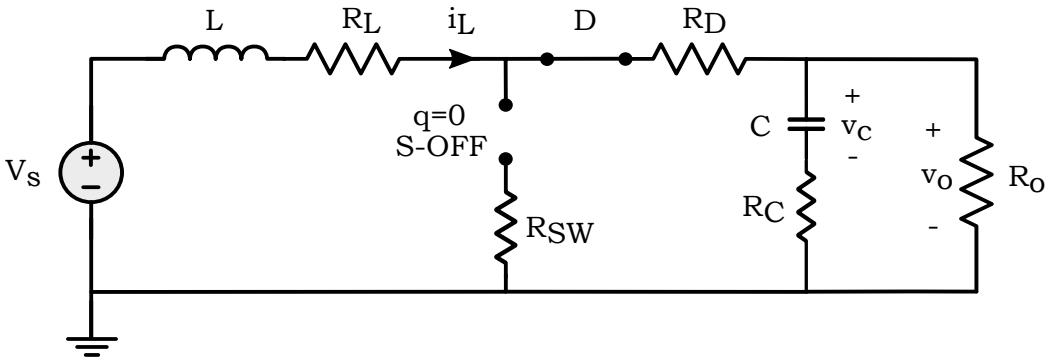


Figure 2.3: Boost converter when $A = 0$,$B = 1$

Using KVL and KCL for the circuit shown in Figure 2.2, differential equations of

system states, current through inductor i_L and voltage across capacitor v_C, and output voltage across load resistor v_O are:

$$\frac{di_L}{dt} = \frac{(R_L + R_{SW})}{L} i_L + 0v_C + \frac{1}{L} V_S \tag{2.3}$$

$$\frac{dv_C}{dt} = 0i_L - \frac{1}{(R_o + R_C)C} v_C + 0V_S \tag{2.4}$$

$$v_O = 0i_L + \frac{R_o}{R_o + R_C} v_C + 0V_S \tag{2.5}$$

Similarly, the differential system of equations for circuit shown in Figure 2.3 can be represented as:

$$\frac{di_L}{dt} = -\frac{R_o(R_L + R_D) + R_c(R_o + R_L + R_D)}{L(R_o + R_c)} i_L - \frac{R_o}{L(R_o + R_c)} v_C + \frac{1}{L} V_S \tag{2.6}$$

$$\frac{dv_C}{dt} = \frac{R_o}{(R_o + R_c)C} i_L - \frac{1}{(R_o + R_C)C} v_C + 0V_S \tag{2.7}$$

$$v_O = \frac{R_o R_c}{R_o + R_C} i_L + \frac{R_o}{R_o + R_C} v_C + 0V_S \tag{2.8}$$

Combining both aforementioned discrete systems, using Boolean arithmetic for variables A and B, yields into following set of equations:

$$\frac{di_L}{dt} = \frac{(R_L + R_{SW})}{L} i_L A - \frac{R_o(R_L + R_D) + R_c(R_o + R_L + R_D)}{L(R_o + R_c)} i_L \overline{A} - \frac{R_o}{L(R_o + R_c)} v_C \overline{A} + \frac{1}{L} V_S \tag{2.9}$$

$$\frac{dv_C}{dt} = \frac{R_o}{(R_o + R_c)C} i_L \overline{A} - \frac{1}{(R_o + R_C)C} v_C + 0V_S \tag{2.10}$$

$$v_o = \frac{R_o R_c}{R_o + R_C} i_L \overline{A} + \frac{R_o}{R_o + R_C} v_C + 0 V_S \tag{2.11}$$

Since the boost converter is operating in CCM mode and hence $B = 1$, as per the Boolean redundancy law, none of the equations (2.9)-(2.11) contain Boolean variable B. These set of equations represents all combinations of system. There exist few oms states for CCM mode based on state of electronic switch.

2.1.2 Discrete modeling of Boost converter in DCM

Discontinuous mode of boost converter attributes to the zero inductor current $i_L = 0$. In contrast to CCM, where inductor current remains always non-zero due to closed electric path either through switch S or diode D; in DCM there exist an additional circuit configuration. When $A = 0$ and $B = 0$, electronic switch S and diode D remain in OFF mode, and, hence leave no electric path for inductor current i_L to flow. The circuit configuration of this mode is illustrated in Figure (2.4).

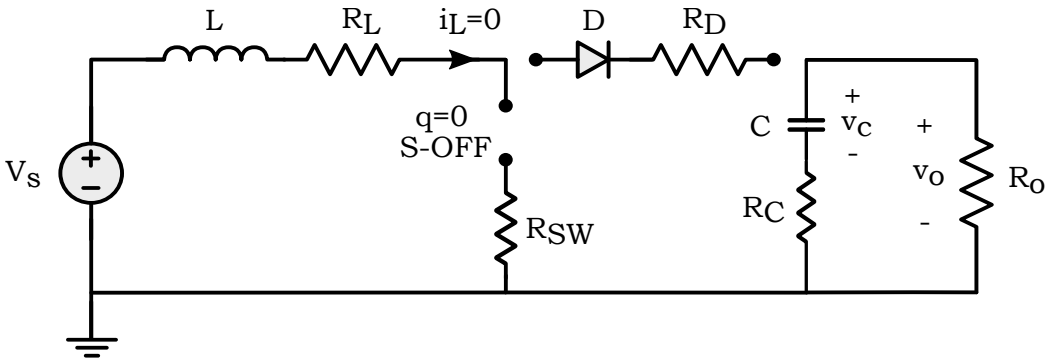


Figure 2.4: Boost converter during DCM when $A = 0$, $B = 0$

Differential system of equations for the circuit configuration ($A = 0$ and $B = 0$)

shown in Figure (2.4) is represented as:

$$\frac{di_L}{dt} = 0i_L + 0v_C + 0V_S \tag{2.12}$$

$$\frac{dv_C}{dt} = 0i_L - \frac{1}{(R_o + R_C)C}v_C + 0V_S \tag{2.13}$$

$$v_O = 0i_L + \frac{R_o}{R_o + R_C}v_C + 0V_S \tag{2.14}$$

Apart from this DCM specific circuit configuration, rest of the circuit combinations from CCM, Equations (2.9)-(2.11), remain unchanged for $i_L > 0$. Hence, complete discrete model with CCM and DCM modes can be represented in combined from as follows:

$$\frac{di_L}{dt} = \frac{(R_L + R_{SW})}{L}i_L AB - \frac{R_o(R_L + R_D) + R_c(R_o + R_L + R_D)}{L(R_o + R_c)}i_L \overline{A}B \tag{2.15}$$

$$- \frac{R_o}{L(R_o + R_c)}v_C \overline{A}B + \frac{1}{L}V_S B$$

$$\frac{dv_C}{dt} = \frac{R_o}{(R_o + R_c)C}i_L \overline{A}B - \frac{1}{(R_o + R_C)C}v_C + 0V_S \tag{2.16}$$

$$v_O = \frac{R_o R_c}{R_o + R_C}i_L \overline{A}B + \frac{R_o}{R_o + R_C}v_C + 0V_S \tag{2.17}$$

The aforementioned discrete model of boost converter will provide strong foundation in extension of single stage boost converter to multiple boost converter system with minimal changes. In addition, the extended multiple/paralleled boost converter model based on discrete modeling approach will be incorporated crucially in implementation of MTC algorithm elaborated in section (3.2).

2.1.3 Simulation of Boost converter using discrete model

In general, average modeling of boost converter [19] is a very versatile method for simulation and implementation of controls. The average modeling approach emphasis on average of ON- and OFF- state of system and connects duty cycle d of PWM signal, applied at the gate of switch S , with the system states i_L and v_C. With identical system parameters, there is no functional difference in operation boost converter, but discrete modeling approach provides full access to system model at each switch transitions, from $1(ON) \rightarrow 0(OFF)$ and $0(OFF) \rightarrow 1(ON)$, hence implementation of discrete controls becomes much easier. In order to exemplify how to simulate discrete model, a functional block diagram of simulation process is illustrated in Figure 2.5. It is important to note that the step-time (model update time) for simulation is significantly lower than time period of PWM signal.

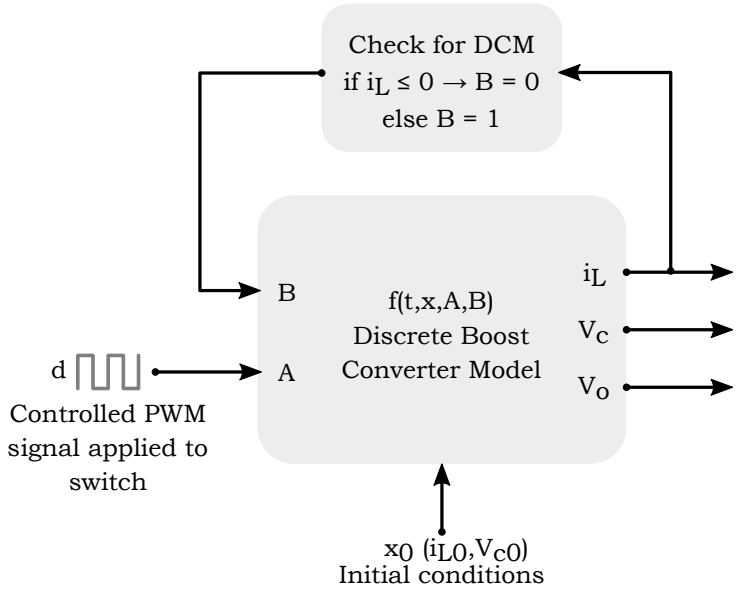


Figure 2.5: Functional representation for simulation of discrete boost converter

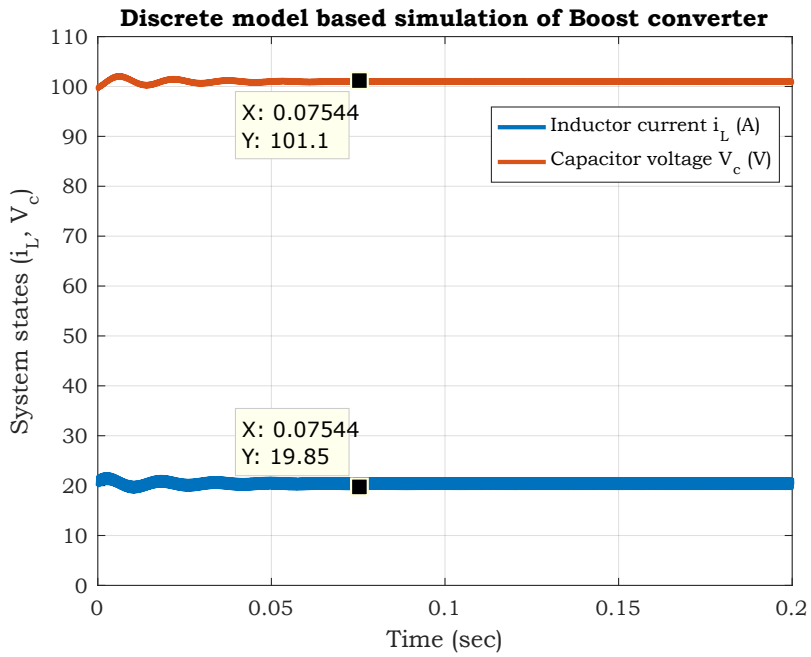


Figure 2.6: Discrete model based simulation of boost converter

To validate system response with discrete model, a sample boost converter simulation starting from steady state and operating under duty cycle d based open-loop control is illustrated in Figure 2.6. Note that the system states i_L and V_C begins with initial conditions close to the steady state values. The deviations at the beginning, due to switching of system states, settle down very quickly and continue with steady state response. Besides from time based depiction of system states, phase-plane representation of system states provide more insight into the state transitions per switching event instead of time. This method of representing system states, phase plane trajectories, will be used extensively in later sections to evaluate and analyze performance of SMC (in (3.1)) and MTC (in (3.2)) based controls algorithms. A phase plane trajectory for a time-domain simulation result shown in Figure 2.6 is given in Figure 2.7.

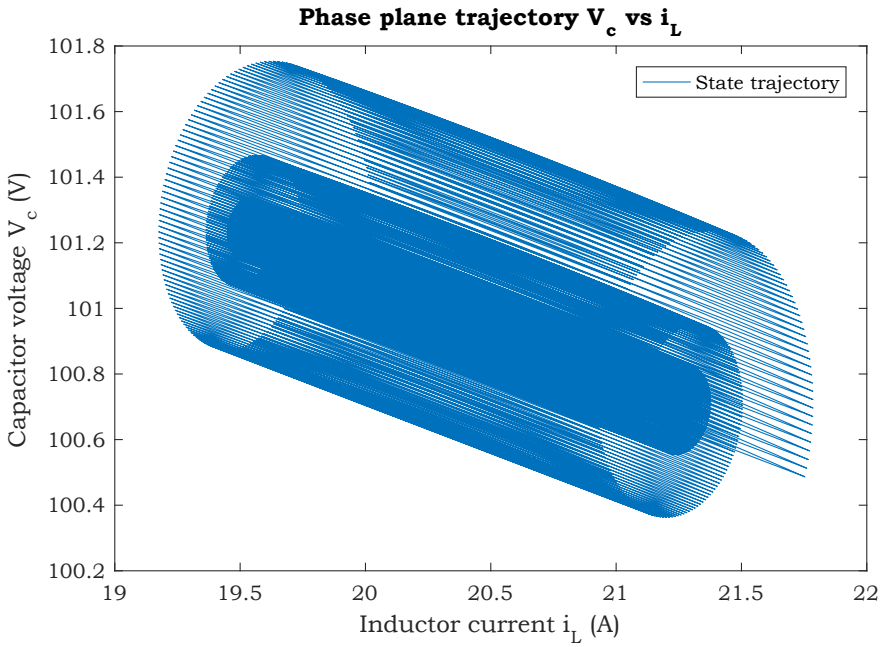


Figure 2.7: Phase plane trajectory $V_C \rightarrow i_L$ from simulation shown in Figure 2.6

From the phase trajectory, it is evident that the system states converge circularly to the steady state within finite time. Additionally, each switch transition provide more insight to variance and steady state error of the inductor current and capacitor voltage.

2.2 Mathematical model of multiple/paralleled Boost converters connected to a resistive load

The discrete modeling approach developed in section (2.1.2) for a single boost converter can be easily extended for multiple boost converters connected to a load. A simplified electrical network of multiple boost converter is illustrated in Figure 2.8, where n number of boost converter low-side stages are connected a

bus. All of the individual boost converters are assumed to be connected to an independent energy source and hence form a parallel network of boost converters. In the literature, similar typologies developed for low-power applications emphasis on identical system parameters, i.e. $L_1 = L_2 = \ldots = L_n = L$. The assumption of identical system parameters simplifies the control system architecture by a large extent and reduces the computational complexity for optimal control of system states. This is accomplished by reducing system complexity to one of the stages and then sharing/dividing control law among the stages equally.

While this approach of sharing system states, i.e inductor current i_L, among all stages simplifies the computational burden of optimization problem, but do not provide similar performance to the system that has heterogeneous system parameters. Practical systems, specifically for high-power applications, are not suitable to be considered as identical. Hence, the prime focus of this study is to include non-identical system parameters into the system modeling. An added advantage of this approach is that it covers the system with identical parameters as one of its subset cases.

By extending the approach of discrete modeling of a boost converter in CCM and DCM modes for multiple/paralleled boost converters network, each stage has has binary variable $B_n = \{0, 1\}$ and $A_n = \{0, 1\}$ to represents state of switch S_n. The combination resulting from these Boolean variables convert each converter stage into one of the configurations as illustrated in Figure 2.2,2.3 and 2.4. Irrespective of mode of operation (CCM/DCM) for an individual boost converter, the flow of energy (inductor current I_L) remains unidirectional, from source V_{Sn} to load.

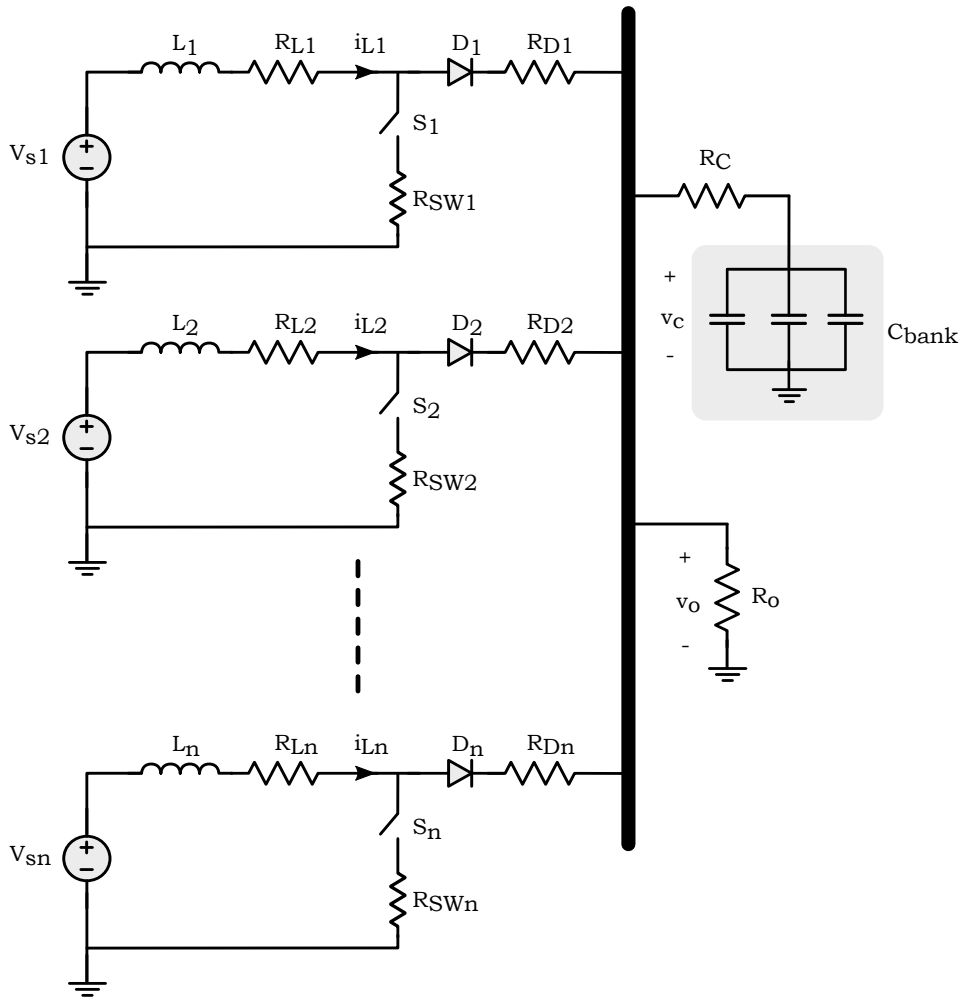


Figure 2.8: n- paralleled boost converters connected to a non-variable resistive load

Mathematical model of $n-$paralleled boost converters is :

$$\frac{di_{L1}}{dt} = \frac{(R_{L1} + R_{SW1})}{L_1} i_{L1} A_1 B_1 - \frac{R_o}{L_1(R_o + R_c)} v_C \overline{A}_1 B_1 \tag{2.18}$$

$$- \frac{R_o(R_{L1} + R_{D1}) + R_c(R_o + R_{L1} + R_{D1})}{L_1(R_o + R_c)} i_{L1} \overline{A}_1 B_1 + \frac{1}{L_1} V_{S1} B_1$$

$$\frac{di_{L2}}{dt} = \frac{(R_{L2} + R_{SW2})}{L_2} i_{L2} A_2 B_2 - \frac{R_o}{L_2(R_o + R_c)} v_C \overline{A}_2 B_2$$

$$- \frac{R_o(R_{L2}+R_{D2})+R_c(R_o+R_{L2}+R_{D2})}{L_2(R_o+R_c)} i_{L2}\overline{A}_2 B_2 + \frac{1}{L_2} V_{S2} B_2 \tag{2.19}$$

$$\vdots$$

$$\frac{di_{Ln}}{dt} = \frac{(R_{Ln}+R_{SWn})}{L_n} i_{Ln} A_n B_n - \frac{R_o}{L_n(R_o+R_c)} v_C \overline{A}_n B_n$$

$$- \frac{R_o(R_{Ln}+R_{Dn})+R_c(R_o+R_{Ln}+R_{Dn})}{L_n(R_o+R_c)} i_{Ln}\overline{A}_n B_n + \frac{1}{L_n} V_{Sn} B_n \tag{2.20}$$

$$\frac{dv_C}{dt} = \left\{ \frac{R_o}{(R_o+R_c)C} i_{L1}\overline{A}_1 B_1 + \frac{R_o}{(R_o+R_c)C} i_{L2}\overline{A}_2 B_2 + \cdots \right.$$

$$\left. + \frac{R_o}{(R_o+R_c)C} i_{Ln}\overline{A}_n B_n \right\} - \frac{1}{(R_o+R_C)C} v_C \tag{2.21}$$

$$v_O = \left\{ \frac{R_o R_c}{R_o+R_C} i_{L1}\overline{A}_1 B_1 + \frac{R_o R_c}{R_o+R_C} i_{L2}\overline{A}_2 B_2 + \cdots \right.$$

$$\left. + \frac{R_o R_c}{R_o+R_C} i_{Ln}\overline{A}_n B_n \right\} + \frac{R_o}{R_o+R_C} v_C \tag{2.22}$$

Hereafter, a simplified version of 3-paralled boost converters shown in Figure 2.9 will be used to design control methods for the multiple/paralleled boost converter network. The simplified network will also be considered as reference to demonstrate simulation and experimental results in following sections of this thesis. Reduced order mathematical model for the microgrid network shown in Figure 2.9 can be described by Equations (2.23)-(2.27). As noted earlier, parasitic components of inductor, capacitors and electronic switches are taken into account to replicate identical system behavior as much as feasible. Here, it is important to

note that all inductor currents i_{L1}, i_{L2}, i_{L3}, and capacitor voltage v_C are dependent states and linked together. However, inductor currents of individual stages are separate; indicates that there is no direct electrical connection between two boost converter stages and they transfer energy individually from sources to load.

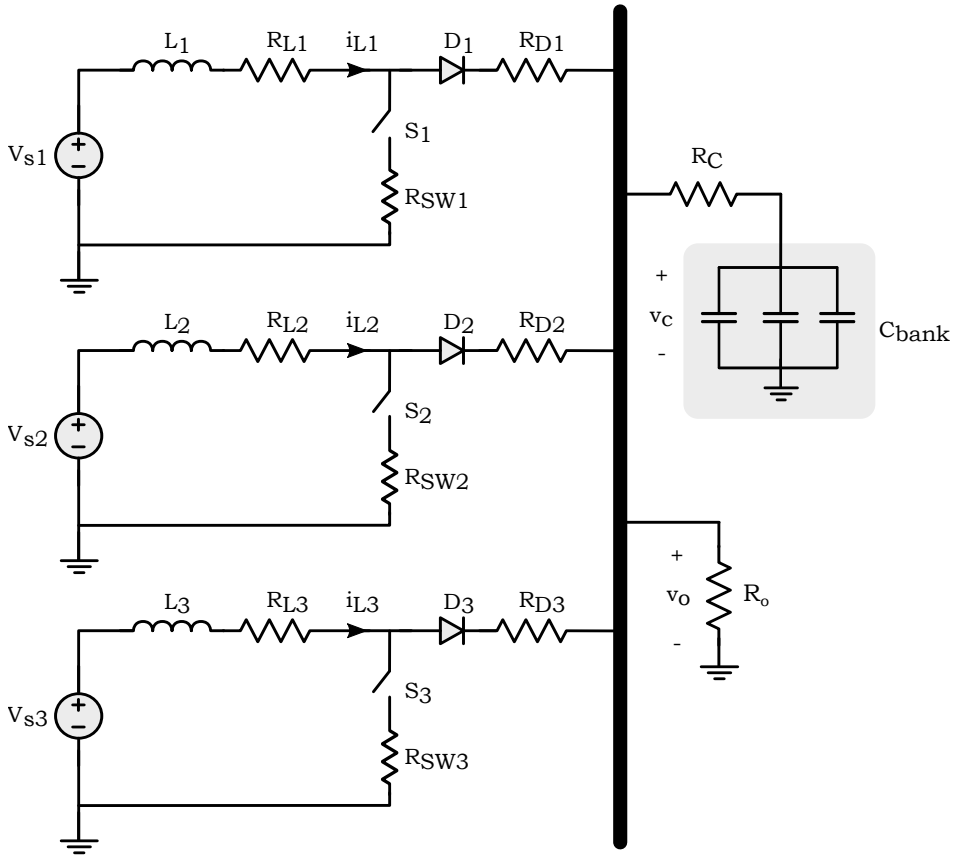


Figure 2.9: 3- paralleled boost converters connected to a non-variable resistive load

$$\frac{di_{L1}}{dt} = \frac{(R_{L1} + R_{SW1})}{L_1} i_{L1} A_1 B_1 - \frac{R_o}{L_1(R_o + R_c)} v_C \overline{A}_1 B_1$$

$$- \frac{R_o(R_{L1} + R_{D1}) + R_c(R_o + R_{L1} + R_{D1})}{L_1(R_o + R_c)} i_{L1} \overline{A}_1 B_1 + \frac{1}{L_1} V_{S1} B_1 \qquad (2.23)$$

$$\frac{di_{L2}}{dt} = \frac{(R_{L2}+R_{SW2})}{L_2} i_{L2} A_2 B_2 - \frac{R_o}{L_2(R_o+R_c)} v_C \overline{A}_2 B_2$$

$$- \frac{R_o(R_{L2}+R_{D2}) + R_c(R_o+R_{L2}+R_{D2})}{L_2(R_o+R_c)} i_{L2} \overline{A}_2 B_2 + \frac{1}{L_2} V_{S2} B_2 \quad (2.24)$$

$$\frac{di_{L3}}{dt} = \frac{(R_{L3}+R_{SW3})}{L_n} i_{L3} A_3 B_3 - \frac{R_o}{L_3(R_o+R_c)} v_C \overline{A}_3 B_3$$

$$- \frac{R_o(R_{L3}+R_{D3}) + R_c(R_o+R_{L3}+R_{D3})}{L_3(R_o+R_c)} i_{L3} \overline{A}_3 B_3 + \frac{1}{L_3} V_{S3} B_3 \quad (2.25)$$

$$\frac{dv_C}{dt} = \left\{ \frac{R_o}{(R_o+R_c)C} i_{L1} \overline{A}_1 B_1 + \frac{R_o}{(R_o+R_c)C} i_{L2} \overline{A}_2 B_2 + \right.$$

$$\left. + \frac{R_o}{(R_o+R_c)C} i_{L3} \overline{A}_3 B_3 \right\} - \frac{1}{(R_o+R_C)C} v_C \quad (2.26)$$

$$v_O = \left\{ \frac{R_o R_c}{R_o+R_C} i_{L1} \overline{A}_1 B_1 + \frac{R_o R_c}{R_o+R_C} i_{L2} \overline{A}_2 B_2 + \right.$$

$$\left. + \frac{R_o R_c}{R_o+R_C} i_{L3} \overline{A}_3 B_3 \right\} + \frac{R_o}{R_o+R_C} v_C \quad (2.27)$$

Equation (2.27) represents actual voltage across load R_o which is almost equivalent to bus/capacitor voltage v_C, given $R_c \lll R_o$.

3 Sliding Mode Control (SMC) and Minimum Time Control(MTC) of paralleled boost converters

Designing the control system for the DC-DC boost converter is a complex process due the non-linear characteristic of converter. The non-linearity of a boost or any other type of power converter is studied under specific category of systems, known as, Variable Structure System(VSS) [20]. The variable structure of system, here boost converter, attributes to the different system configuration due to condition of switches. Depending on the state of power electronic switch, an overall system will reduce to a subsystem and will have specific characteristic according to the mode of operation. Mathematical model and equivalent electrical circuit for a boost converter is elaborated in section (2.1) during possible switch configurations.

In this chapter, a robust control system, Sliding Mode Control, for paralleled boost converters is developed. Simulation results for this implementation is also demonstrated to validate the controller operation. In following sections, a novel control method for same system is developed in order to achieve minimum state transition time. It is important to note that the control method developed under the name, Minimum Time Control (MTC), may not be a primary control method for regular application of paralleled boost converter. It can be invoked as a secondary control system to achieve desired performance. However, it does not restrict the usage of the MTC as in role of primary control system.

3.1 Sliding Mode Control(SMC) of Multiple Boost Converters

Sliding Mode Control is very a suitable control technique for intrinsically non-linear, and system with non-modeled external perturbation [21]. Boost converters fit very closely within the definition of variable structure system, and SMC is a most viable method to implement robust control system. There exists various types of SMC based solutions for VSS and one of the most commonly used with boost converters is current-mode SMC where Inductor current is considered as reference surface and control law is implemented to follow the trajectory. In addition, voltage based control can also be implemented by translating voltage to current[20][22]. In the literature, there are numerous techniques developed for single stage or interleaved boost converter [23]. In this section, a SMC scheme is developed for paralleled boost converter topology. Simulation results are illustrated to exemplify the operation of SMC for paralleled boost converters.

3.1.1 Sliding Mode Controller design

From the fundamental theory of SMC, the sliding function/surface (S) is represented by linear difference between actual system state and its reference value. For current-mode control, the sliding surface S can be represented as is,

$$S = i_{Ln_{actual}} - i_{Ln_{ref}} \tag{3.1}$$

where, $i_{Ln_{actual}}$and $i_{Ln_{ref}}$ are actual and reference values of inductor current for a boost converter stage. $i_{Ln_{actual}}$ represents a feedback signal measured from the circuit, whereas $i_{Ln_{ref}}$is command signal translated from $V_{bus_{ref}}$ and can be represented as,

$$Power\ from\ source = Power\ to\ system$$

$$= \{Power\ consumed\ by\ load\} + \{losses\ in\ system\} \quad (3.2)$$

$$V_{s_i} i_{Ln_{ref}} = \left\{ CF_i \frac{V^2_{bus_{ref}}}{R_{load}} \right\} + \{ i^2_{Ln_{actual}} (R_{L_i} + R_{sw_i}) + i_{Ln_{actual}} V_{d_i} \} \quad (3.3)$$

$$i_{Ln_{ref}} = CF_i \frac{V^2_{bus_{ref}}}{R_{load} V_{s_i}} + \frac{i^2_{Ln_{actual}} (R_{L_i} + R_{sw_i})}{V_{s_i}} + \frac{i_{Ln_{actual}} V_{d_i}}{V_{s_i}} \quad (3.4)$$

Hence, any desired bus voltage can be achieved by setting equivalent current reference for particular stage. Factor CF_i represents a contribution of a boost converter stage, since overall current flowing towards the load is total of the individual stage currents. Normally, $CF_i = \frac{1}{n}$, where n is number of boost converter stages in paralleled topology. It can also be set in different proportion where it properly shares the contribution among the stages depending on their current/power carrying capabilities. Last two terms in equation (3.4) account for the current that contribute to the losses through the resistance of system components

and voltage drop across the diodes. To maintain sliding mode in manifold $S = 0$,

$$u = \begin{cases} 0, & S \geq 0 \\ 1, & S < 0 \end{cases}$$

$$\Rightarrow u = \frac{1}{2}(1 - sign(S)) \tag{3.5}$$

In order to maintain the manifold $S = 0$, the reaching condition can be achieved with a constraint $V_{bus} > V_{s_i}$. In the following section, simulation results for an example topology are illustrated.

3.1.2 Simulation of Multiple Boost Converters with SMC

Simulation results for a specific case, voltage transition from $V_{bus_{init}} = 90\,V$ to final voltage $V_{bus_{final}} = 150\,V$, is represented in this section to exemplify operation of SMC.

Table 1: Paralleled Boost Converter system parameters

Parameter	**Stage 1**	**Stage 2**	**Stage 3**
Source Voltage V_S(V)	72.00	65.00	58.00
Inductance L(mH)	3.000	3.200	2.000
ESR of Inductor $R_L(\Omega)$	0.200	0.200	0.200
Maximum Inductor current $I_{L_{max}}$(A)	60.00	60.00	60.00
Switch ON resistance $R_{SW}(\Omega)$	0.020	0.020	0.020
Diode cut-off voltage $V_{D_{ON}}$(V)	0.100	0.15	0.20
Forward resistance of Diode $R_{D_{ON}}(\Omega)$	0.020	0.020	0.020

System parameters for 3 paralleled boost converter stages selected for the simulation are tabulated in Table 1. Capacitance of the bank C_{bank} and load resistance R_{load} is considered as $1.5\,mF$ and $10\,\Omega$ respectively. The ESR for the capacitor bank is considered $0\,\Omega$ for this simulation. According to the equation (3.4), the reference current for each stage can be given as,

$$i_{L1_{ref}} = 0.33\frac{V^2_{bus_{ref}}}{720} + \frac{i^2_{Ln_{actual}}0.220}{72} + \frac{i_{Ln_{actual}}0.10}{72}$$

$$i_{L2_{ref}} = 0.33\frac{V^2_{bus_{ref}}}{650} + \frac{i^2_{Ln_{actual}}0.220}{65} + \frac{i_{Ln_{actual}}0.15}{65}$$

$$i_{L3_{ref}} = 0.33\frac{V^2_{bus_{ref}}}{580} + \frac{i^2_{Ln_{actual}}0.220}{58} + \frac{i_{Ln_{actual}}0.20}{58}$$

and in simplified form,

$$i_{L1_{ref}} = 0.000458V^2_{bus_{ref}} + 0.00305i^2_{L1_{actual}} + 0.00138i_{L1_{actual}} \tag{3.6}$$

$$i_{L2_{ref}} = 0.000507V^2_{bus_{ref}} + 0.00338i^2_{L2_{actual}} + 0.00230i_{L2_{actual}} \tag{3.7}$$

$$i_{L3_{ref}} = 0.000568V^2_{bus_{ref}} + 0.00379i^2_{L3_{actual}} + 0.00344i_{L3_{actual}} \tag{3.8}$$

Current for each boost converter stage is controlled through the switching law derived in equation (3.5) and substituting results from sliding surface S(3.1) and corresponding equations (3.6)-(3.8). Relation between inductor currents and load voltage is equated as,

$$\begin{aligned} i_{load_{actual}} &= i_{L1_{actual}} + i_{L1_{actual}} + i_{L1_{actual}} && (\because \textit{inpendant stages}) \\ V_{bus_{actual}} &= \tfrac{i^2_{Load_{actual}}}{R_{load}} && (\because C_{ESR} = 0\,\Omega) \end{aligned}$$

Figure 3.1 represents system states during SMC operation. Throughout the simulation, system is performing under SMC. Overall time span of the simulation is set to $t = 80\,ms$ and a step change in $V_{bus_{ref}}$ is triggered at time $t = 30\,ms$. When step change in $V_{bus_{ref}}$ is triggered, inductor currents swiftly moves toward next steady state point. Whereas, V_{bus} takes relatively slow transition and reaches to steady state around $t = 65\,ms$. Even though, inductor currents set very quickly compared to V_{bus}, slower response of V_{bus} keeps overall system under transition for longer time.

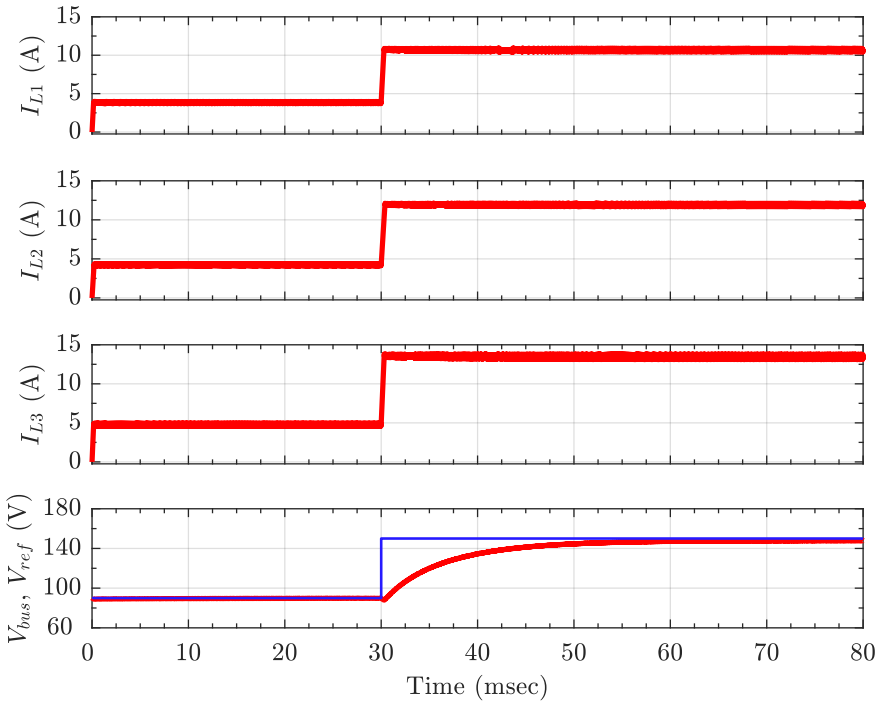


Figure 3.1: System states under sliding mode control (SMC); step change in V_{ref} at $t = 30\,ms$

Instantaneous energy stored in all of the passive elements, E_{L_1},E_{L_2},E_{L_3} and $E_{C_{bank}}$ shown in equations (3.9)-(3.10), are shown in Figure 3.2. With step change in

V_{bus}, energy in inductors E_{L_n}also steps up following sliding surface S. Whereas, the capacitor bank energy E_C remains unchanged at the moment but gradually increases and reaches to the steady level.

$$E_{L_n}(t) = \frac{1}{2} L_i i^2_{Ln_{actual}}(t) \tag{3.9}$$

$$E_C(t) = \frac{1}{2} C_{bank} v^2_C(t) = \frac{1}{2} C_{bank} V^2_{bus_{actual}}(t) \tag{3.10}$$

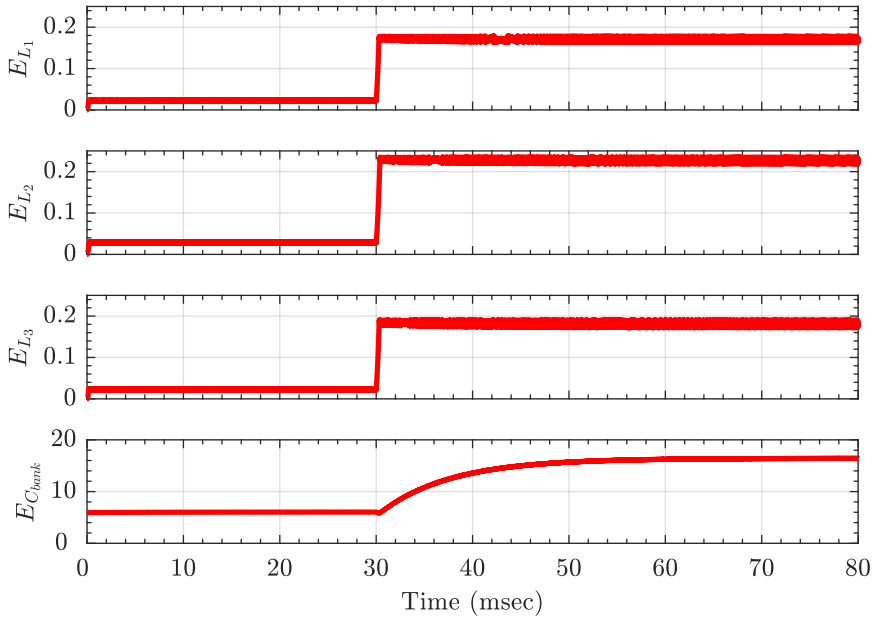


Figure 3.2: Instantaneous Energy (in Joules) stored in passive elements during SMC; (a) E_{L_1},(b) E_{L_2}, (c) E_{L_3}, (d) $E_{C_{bank}}$

Phase plane representation for the system states $i_{L_n} \rightarrow V_{bus}$ over complete duration of the simulation(from $t = 0\,msec$ to $t = 80\,msec$) is illustrated in Figure 3.3. As noted earlier, quick propagation of inductor currents is seen as a single transition while bus voltage remains at $90\,V$. From this point, current in inductors

maintain a steady level while bus voltage moves to the next steady state. Bus voltage prorogation can be seen as horizontal line in all of three trajectories.

Sliding mode control (SMC) developed and illustrated for 3-paralleled boost converters in this section can be extended for n-level configuration of paralleled boost converters. The SMC method will be used in following sections as primary control system along with the Minimum Time Control (MTC).

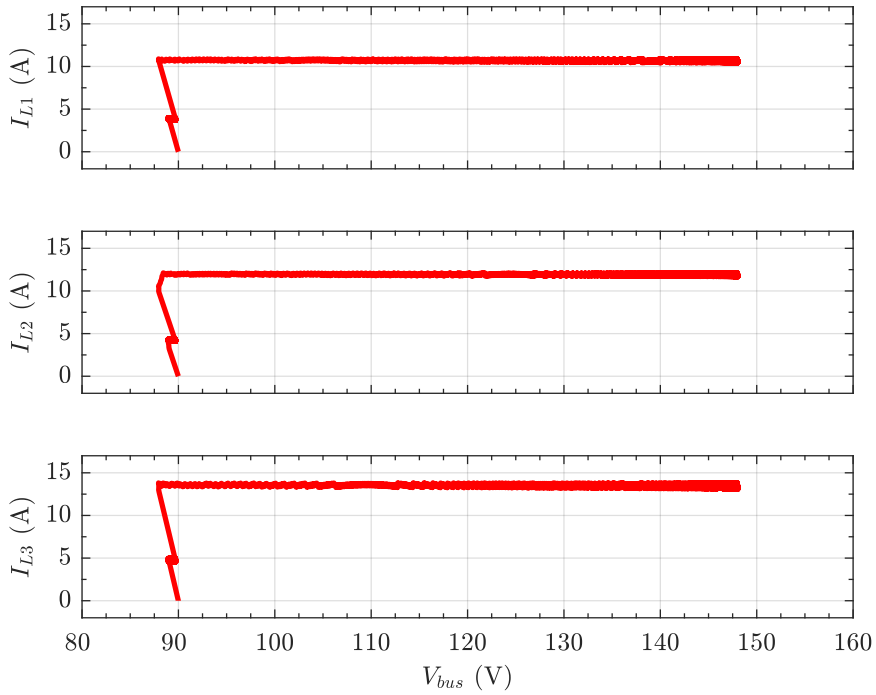


Figure 3.3: Phase plane trajectories $I_{L_n} \rightarrow V_{bus}$ for all boost converter stages over time $t = [0, 80]\, ms$

3.2 Minimum Time Control (MTC) of Paralleled Boost Converters

In previous section (3.1), the robust control system (SMC) for paralleled boost converter was developed. Since SMC is one of the most robust control systems for VSS and can sustain uncertainty in the system and/or external disturbances, it is often considered one of the reliable control system architecture. The system states gradually approach to the steady state while maintaining switching law to follow sliding surface. For many time critical applications, where speed of response is very important factor; SMC fails to perform in robust manner as it does for regular operation. With the goal to achieve steady state as quick as possible, a novel control method is proposed, minimum time control (MTC). In the following sections, the concept of MTC and its implementation is thoroughly explained. Simulation results are shown to illustrate the step-by-step implementation of MTC, as well as for performance comparison with SMC results.

3.2.1 MTC concept and objectives

Minimum time control is about the control of systems states in the minimum time. As previously noted with a sliding mode control where system states slide over the sliding surface and gradually reach to the steady state. In contrast, in a minimum time control, system states are forced to reach steady state within a minimum time. A similar concept of achieving steady state in minimum time is also known as bang-bang control in control systems literature, where system

states are forced to drive in a specific manner such that desire performance is achieved in list amount of time.

In the case of paralleled boost converters, controlling time of ON/OFF state of power electronic switch will determine the states of system (i_L, v_C). While switch S is ON, the inductor L will accumulate the energy from voltage source and current through the inductor i_L will continue to rise and the capacitor bank C voltage v_C will release stored energy through the load resistor R_{load}. In contrast, when switch S is OFF, the inductor L will release the energy to the capacitor bank C and hence, $i_L$will decline, while voltage v_C will increase due to energy transferred from inductor. During this single ON-OFF cycle energy coming from voltage sources passes through the inductor and discharged into load via capacitor bank.

A single ON-OFF cycle, also named as single switch operation, can transit system state from one steady-state operating point $(i_{L1}, i_{L2}, i_{L3}, \cdots, i_{Ln}, v_C)_{t_1}$ to another steady-state operating point $(i_{L1}, i_{L2}, i_{L3}, \cdots, i_{Ln}, v_C)_{t_2}$, provided proper(optimal) time instance of switching is known. There exist no operating condition, other than single switch operation, to achieve system states transition for paralleled boost converter in minimum time. In order to implement the single switch operation, a unique set of switching instances of individual boost converter stage must be known. Mathematical modeling of this optimization problem to find out the switching instances can be described as follows:

Optimization problem definition :

Cost of the optimization function is the time of transition,

$$transition\ time\ J(t) = t_f - t_0 \tag{3.11}$$

Several constraints are imposed on this cost function are,

$$\dot{v}_C(t_f) \to 0 \tag{3.12}$$

$$i_{Ln}(t) \geq 0 \tag{3.13}$$

$$i_{Ln}(t) \leq i_{Ln(max)} \tag{3.14}$$

where, $t_0 \leq t \leq t_f$.

The aforementioned fundamental representation of optimization problem is also referred as maximum effort control, or minimum time control or much familiar name bang-bang control. The goal of this control method is to do a state transition in minimum time with maximum effort. A mathematical representation of a cost/objective function $J(t)$ combined with system constraints/co-states for 3-boost converters system is,

$$J(t) = \{minimum\ transition\ time\} + \ldots \tag{3.15}$$

$$\{minimum\ steady\ state\ inductor\ error\ in\ currents\} + \ldots$$

$$\{minimum\ steady\ state\ error\ in\ bus\ voltage\}$$

$$J(t) = (t_f - t_0)^2 + \dots \tag{3.16}$$

$$(i_{L_{1f}} - i_{L_{10}})^2 + (i_{L_{2f}} - i_{L_{20}})^2 + (i_{L_{3f}} - i_{L_{30}})^2 + \dots$$

$$(V_f - V_0)^2$$

In addition to the the overall cost function in Equation 3.16, there are three more constrains on optimization problem through the discrete nature of switches. While system states are continuous in nature, the switching signals are digital and have only states, ON and OFF.

$$Switch\, Q_n = \begin{cases} 0(OFF), & inductor\ L_n\ discharge\ energy\ into\ capcitor \\ 1(ON), & inductor\ L_n\ accumulate\ energy\ through\ current \end{cases}$$

$here\ n = 1, 2, 3.$

The developed mathematical model of the optimization problem is about minimizing time that is the independent variable and poses several challenges to solve this problem with guaranteed optimal solution. In addition, the discrete behavior of the electronic switches make whole optimization problem into a hybrid domain optimization. In this case, not just several continuous and time-dependent system states but discrete variables like state of Switch S and instance of switching also become more important. For higher number of converter stages complexity of solving the hybrid problem increases enormously. It is important to point out that the objective of this hybrid optimization problem is to obtain optimal switching instances such that overall transition time for all states remain mini-

mum/optimal. To solve this problem with traditional optimization techniques is nearly impossible/impractical for reasons listed below,

1. One of the most challenging aspect of this optimization problem is to minimize the independent variable, time(t), itself. Conventional optimization methods are suitable to optimize dependent quantities, i.e. K(t) or P(t), with/without time dependent constraints.

2. Hybrid optimization problems do not guarantee convergence or let alone unique solution.[24] There exist several hybrid optimization techniques, however they address very restricted domain of linear problems [25].

3. Implementation of traditional optimization solution is computationally very demanding and will grow drastically with every single increased order of optimization problem. Addition of a one more boost converter stage will impose two added constraints and additional time dimension (through the switch time of the corresponding boost converter) on the overall problem. Hence, expand-ability of the optimization problem comes with the cost of a time-intensive optimization process.

4. It is very unlikely to have conventional optimization routine that can be fit into real-time system workflow where the optimization routine has to be extremely lightweight in order to respond effectively in real-time control system.

Considering these limitations of the traditional optimization approach; a novel optimal-like algorithm is proposed, minimum time control(MTC). The aim of

this algorithm is to overcome all the shortcomings of the traditional optimization approach, and yet to be so light-weight that it can be expandable and integrated in real-time workflow without adding significant overhead in operation of real-time control system. A detailed MTC algorithm is elaborated and illustrated in the following section.

3.2.2 Minimum Time Control (MTC) algorithm

In this section, a MTC algorithm is explained and individual steps to calculate switching time t_{switch} for all boost converter stages are listed as below:

Step 1 Derive initial and final values of inductor currents i_{Ln} through $V_{C_{init}}$ and $V_{C_{final}}$.

Step 2 Calculate maximum ON-time for individual boost converter:

$$t_{ON_{max}} = Q_i \rightarrow ON \mid \{I_L < \min(I_{Q_{max}}, I_{bus_{max}})$$

Assuming $Q_i \rightarrow ON$, $t_{ON_{max}}$ can be derived from a known boundary conditions. $I_{Q_{max}}$ represent maximum allowed continuous current through power electronic switch and $I_{bus_{max}}$ denote current threshold for bus depending upon the gauge of the wire.

Step 3 Find equivalent contribution time(τ) for each stage:

$$\boldsymbol{\tau_i} = \frac{I_{L_{max}} - I_{init}}{\alpha} \text{ , where } \boldsymbol{\alpha} = \sum_{i=1}^{n} \frac{V_{s_i}}{L_i}$$

Step 4 Calculate absolute and relative contribution factor(CF):

$$CF_i = \frac{\tau_i}{\sum_{i=1}^{n} \tau_i}$$

$$CF_{rel} = \frac{CF_i}{\min(CF_i)}$$

Absolute contribution factor for each boost converter stage is a unitless quantity that loosely translates to the amount of energy from source to inductors with respect to time. Relative contribution factor is a normalized representation of CF_i.

Step 5 Derive set of switch time(t_{switch}) over full range of β:

$$t_{switch} = \beta \times CF_{rel} \times t_{ON_{max}},\ \beta \in (0, 1)$$

Here, t_{switch} denote set of values of switching instances for all boost converters. Every β value corresponds to a unique set of switching instances.

Step 6 Iterate discrete model of paralleled Boost converter system for any two distinct values of $\beta \in (0, 1)$, generally mid points of the range, β_1 and β_2, to find slope and y-intercept of characteristic line.

$$slope\, m = \frac{V_{C_{\beta 2}} - V_{C_{\beta 1}}}{\beta_2 - \beta_1}$$

$$y - intercept\, C = V_{C_{\beta 2}} - m\beta_2 \quad OR$$

$$= V_{C_{\beta 1}} - m\beta_1$$

There are two distinct methods implemented for system iteration depending upon the mode of operation of boost converters. One is through the Discontinuous Conductance Mode(DCM) (in (3.2.3)) and another is with Continuous Conduction Mode(CCM) (in (3.2.4)). Each of these modes and their implementation is further discussed in the following sections. Moreover, characteristic line for both methods, DCM and CCM, are illustrated in Figure 3.4 and Figure 3.6 respectively.

Step 7 Calculate $t_{switch} = t_{MTC}$ using β_{MTC} for the desired end point $v_{C_{final}}$:

$$\beta_{MTC} = \frac{v_{C_{final}} - C}{m}$$

$$t_{MTC} = \beta_{MTC} \times CF_{rel} \times t_{ON_{max}}$$

here, t_{MTC} denote switching instances corresponding to minimum time transition for all boost converter. t_{MTC} has dimension equivalent of number of boost converter.

3.2.3 DCM based iteration of paralleled boost converters for MTC

Discontinuous conduction mode(DCM) of the boost converter represents a specific state of the system in which the current flowing through the inductor reduces to the lowest possible level, 0 A, and the capacitor voltage continue to discharge through the load. Generally, DCM is not a preferred mode of operation for most of

power supply applications and not employed unless it serves a specific requirement. No steady operation of real time power supply require sources side inductor to be completely emptied of energy. However, in implementation of the MTC for paralleled boost converters the DCM can be very crucial. A DCM for each one of the stage is used as reference point to synchronize single switch operation between multiple stages. A simple algorithm to iterate the paralleled boost converters for Step 6 of MTC can be described as follows:

Step 1 Time for $i_{Ln} = 0\,A$ to $i_{Ln} = i_{Ln_{final}}$, $t_{switchback}$, is calculated by solving $n-$single order boundary value problems.

Step 2 Simulation of paralleled boost converters begin with the initial values of $i_{Ln_{init}}$ and $v_{C_{init}}$ and all switches being ON.

Step 3 Switch states are flipped from ON to OFF at the time instances t_{switch} found for a particular value of $\beta \in (0, 1)$, i.e. switching instances are $t_{switch} = \{0.00223, 0.00205, 0.0015\}$ for an MTC operation shown in Figure 3.5

Step 4 Every boost converter stage reach to the DCM and $i_{Ln} = 0A$ and waits for a specific amount of time until a unique point, called as Point of action (POA), beginning of last phase in Figure 3.5.

Step 5 Switch states are flipped again from OFF to ON at offset time such that each stage stays ON for a time duration of $t_{switchback}$. The additional procedure required to offset the switchback time for each stage is to synchronize the end point. Offset time, time between beginning

of DCM and POA, can be calculated by substantiating $\max(t_{switchback})$ from $(t_{switchback})$ of all stages. Time duration $(t_{switchback})$ denote the time duration for which a boost converter stage has to persist DCM until POA.

Step 6 All boost converter stages reach to the final state $v_{C_{final}}$ and $i_{Ln_{final}}$ for a particular value of $\beta \in (0, 1)$, i.e. β_1

A detailed state representation for 3 paralleled boost converters is presented in Figure 3.5 for a unique value of $\beta \in (0, 1)$. The change in operation of individual stage achieved through the discrete modeling of paralleled boost converters. Note the minor overshoot in V_{bus} due to the DCM mode. It is due to the transfer of all of the energy stored into inductor and then forcing switches to stay ON until a final state is not achieved, resulting capacitor bank to discharge.

When V_{bus} is plotted for almost full range $\beta \in [0.1, 0.8]$, the resulting set of transitions look as per shown in Figure 3.4. Here, it is important to differentiate the theoretical and practical range of β. The reason why the boundaries of β are shrunken is to avoid infeasible transition cases. From the range $[0.1, 0.8]$ the characteristic line (dotted line in Figure 3.4) covers almost all feasible cases. It is recommended to select two mid points, β_1 and β_2, of the specified range to get accurate line parameters and hence closest $V_{bus_{final}}$ point.

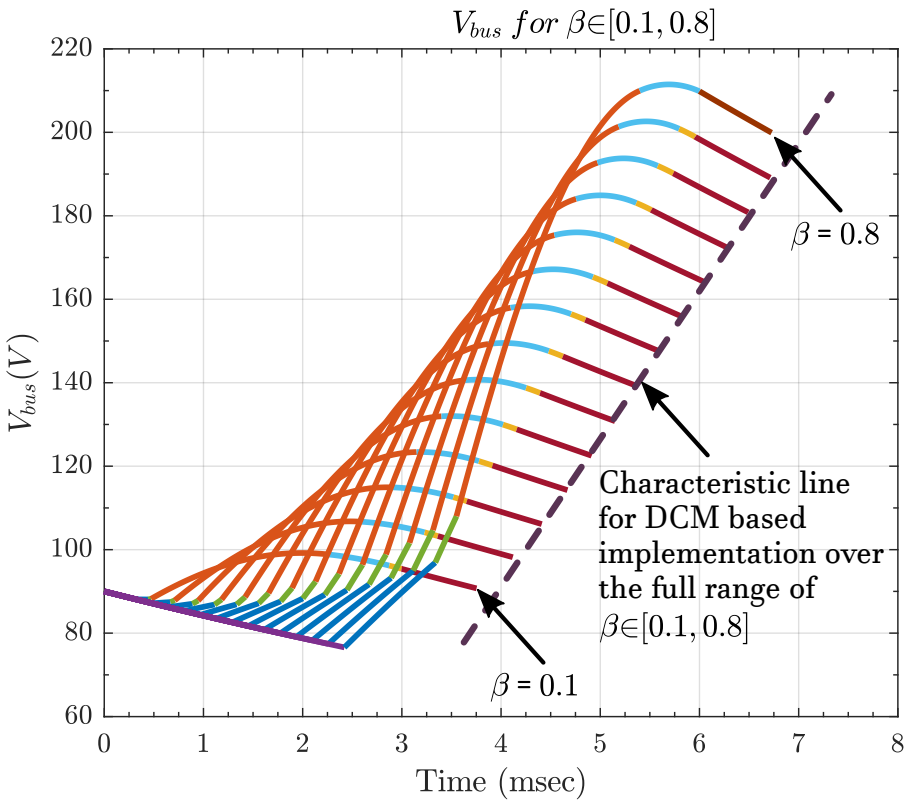


Figure 3.4: V_{bus} over $\beta \in [0.1, 0.8]$ for DCM based implementation of MTC

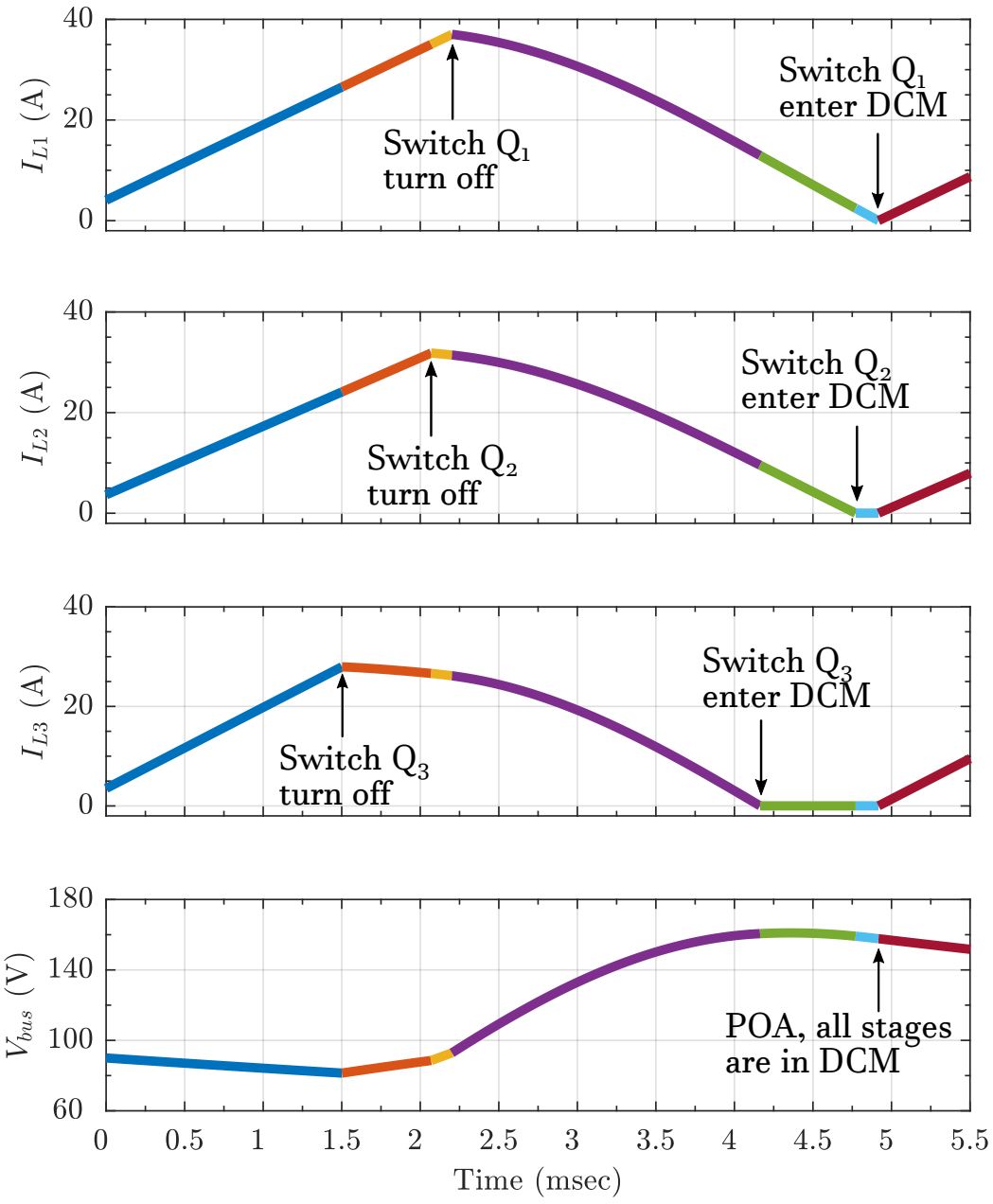


Figure 3.5: System states during DCM based implementation of MTC

3.2.4 CCM based iteration of paralleled boost converters for MTC

Continuous conduction mode (CCM) of the boost converter represents a specific state of the system in which the current flowing through the inductor always remains above the lowest possible level, 0 A. Generally, CCM is a preferred mode of operation for most of power supply applications. During switch mode power supply operation, energy is transferred from inductor to capacitor in consecutive switching cycles and maintains CCM. In contrast with the DCM based MTC, CCM based MTC approach do not require the inductor currents reduced to 0 A. A simple algorithm to iterate the paralleled boost converters for Step 6 of MTC can be described as follows:

Step 1 Simulation of paralleled boost converters begin with the initial values of $i_{Ln_{init}}$ and $V_{C_{init}}$ and all switches being ON.

Step 2 Switch states are flipped from ON to OFF at the time instances t_{switch} found for a particular value of $\beta \in (0, 1)$, i.e. $t_{switch} = \{0.00223, 0.00205, 0.0015\}$ for a MTC operation shown in Figure 3.7

Step 3 MTC control is released after t_{switch}, and each stage is now under auxiliary control, i.e. current-controlled SMC as described in (3.1.2).

Step 4 All boost converter stages reach to the final state $v_{C_{final}}$ and $i_{Ln_{final}}$ for a particular value of $\beta \in (0, 1)$, i.e. β_1

A detailed state representation for 3 paralleled boost converters is presented in Figure 3.7 for a unique value of $\beta \in (0, 1)$. The change in operation of individual

stage achieved through the discrete modeling of paralleled boost converters. Note the elimination of overshoot in V_{bus} due to the CCM mode. Once the switch states are flipped from ON to OFF, each stage is controlled by its separate current-mode control. The only possible transition from here is to the steady states current $i_{Ln_{final}}$. Hence, in CCM based MTC switches are only forced until the time instance t_{switch}. Then after, control is switched back to the auxiliary/routine control system that drives system states to the desired levels.

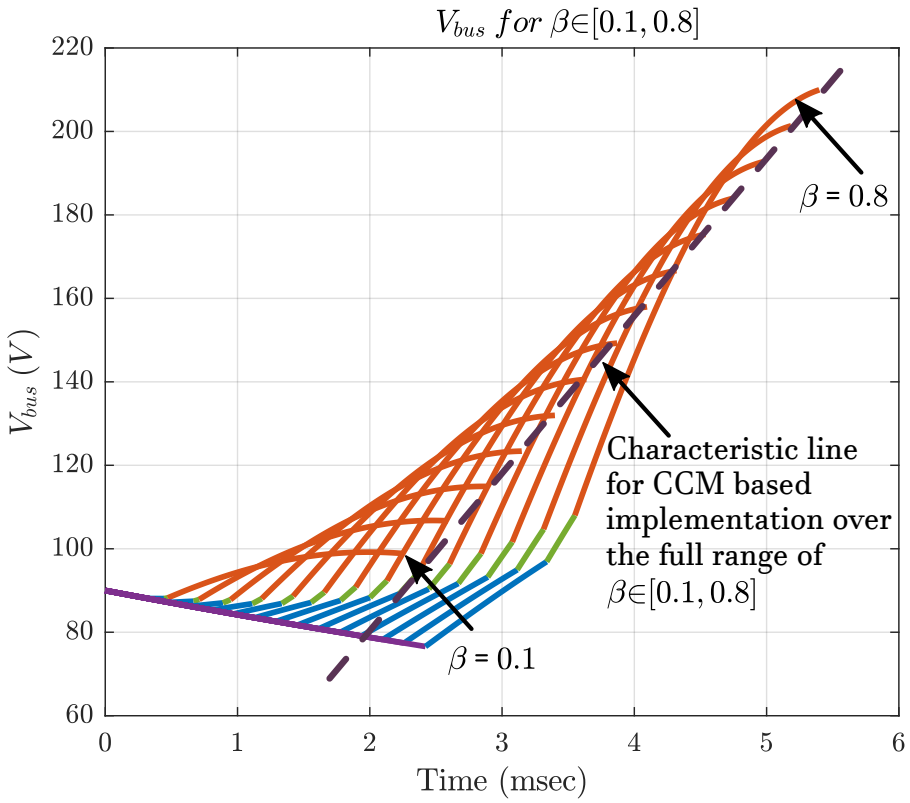


Figure 3.6: V_{bus} over $\beta \in [0.1, 0.8]$ for CCM based implementation of MTC

When V_{bus} is plotted for almost full range $\beta \in [0.1, 0.8]$, the resulting set of transitions look as per shown in Figure 3.6. Here, it is important to differentiate the theoretical and practical range of β. The reason why the boundaries of β

are shrunken is to avoid infeasible transition cases. From the range $[0.1, 0.8]$ the characteristic line (dotted line in Figure 3.4) covers almost all feasible cases. It is recommended to select two mid points, β_1 and β_2, of the specified range to get accurate line parameters and hence closest $V_{bus_{final}}$ point. Except forcing inductor currents to DCM($0\ A$), for synchronization, there is no functional difference in both approaches to implement MTC. However, one of the methods has several crucial aspects that can be very beneficial in implementation of MTC in real-time framework. Some of these differences between these two approaches are highlighted in following section.

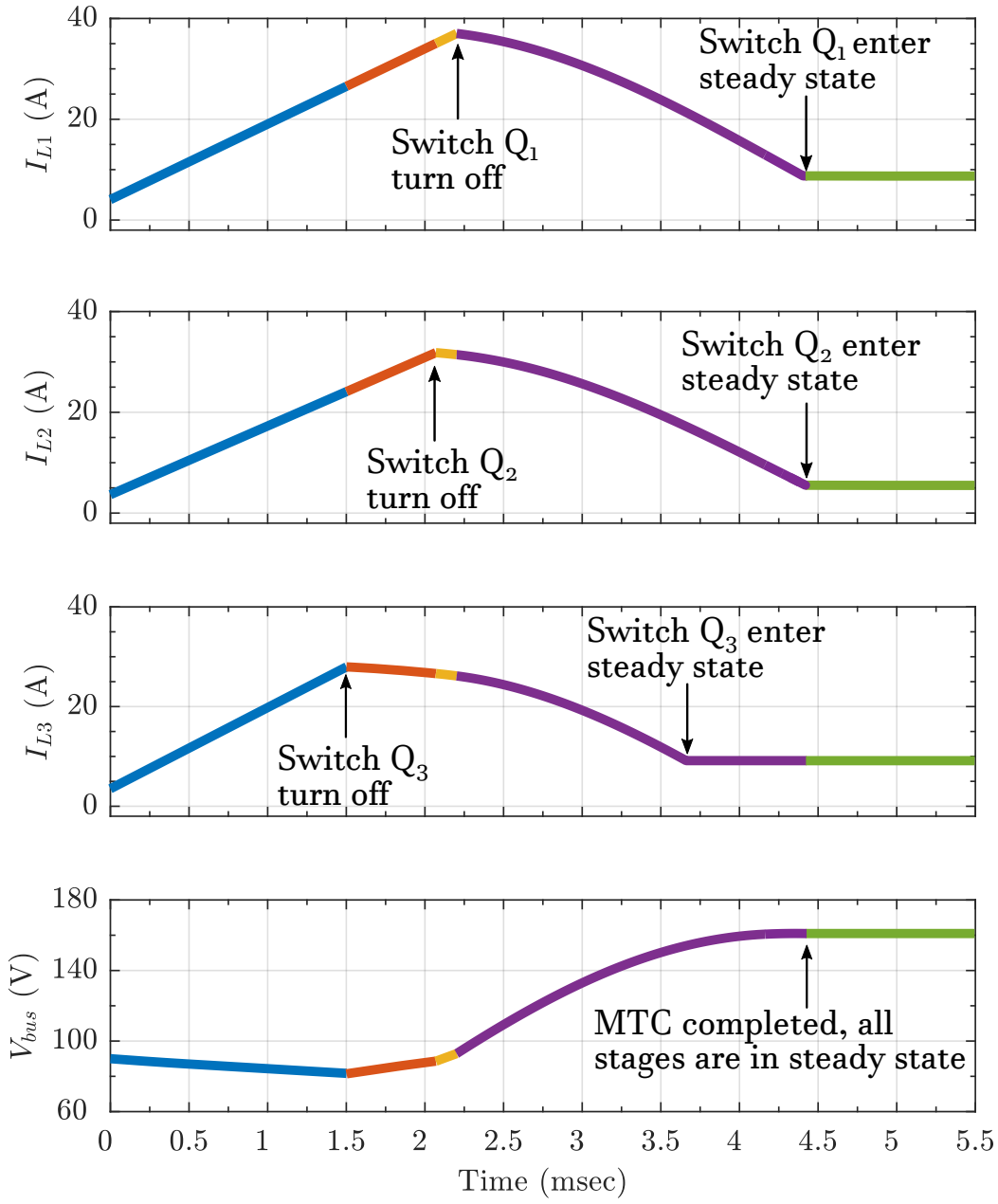


Figure 3.7: System states during CCM based implementation of MTC

3.2.5 Comparison between CCM and DCM based implementation of MTC

In previous sections 3.2.3-3.2.4, both DCM and CCM based MTC algorithm were thoroughly discussed and the simulation of each method was presented. Although, the approaches attain the same results, there is a very significant difference in both methods. Some of the very important differences are highlighted as below:

- CCM based MTC approach is computationally less expensive that of DCM based approach. The amount of calculation during each MTC operation is considerable higher for the DCM based approach due to the calculation for synchronization time between boost converter stages. In case of CCM based method this computational overhead is completely eliminated since each stage will be released from MTC routine once the inductor current reaches the cut-off level and the switch turns off. From real-time MTC implementation point of view this is an important aspect that keeps the computer less occupied for computation purposes and hence can complete time sensitive tasks, i.e. updating IO ports, supervisory control.

- Apart from the occupation of a computer for on-line computations of MTC, CCM based approach complete the transition process in less time than of DCM based approach. The time overall time to complete the transition differ between both approaches due the complete elimination of synchronization of individual stages. Uncoupled control of each staged during CCM enables the faster state transition and hence the steady state can be achieved relatively faster than DCM mode. Refer Figure 3.5 and 3.7 for overall transition time

of DCM ($\sim 5.5\,ms$) and CCM($\sim 4.5\,ms$) based approach respectively. CCM based state transition takes almost 20% less time than that of DCM.

- Another notable difference between both of these methods is the overshoot in bus voltage $V_{bus_{init}}$. Since the inductor current of each stage has to reach $0\,A$ in DCM based method, all the energy is transferred to capacitor bank resulting in an overshoot in bus voltage. In case of CCM based method, inductor current of each stage will never go below its final steady states value. Hence, the overall transition do not have an overshoot in bus voltage during CCM based method.

3.2.6 Simulation of paralleled boost converter with MTC

As explained in previous section, CCM based approach is more appropriate compared to DCM from computational overhead and complexity of overall MTC operation point of views. Hence, CCM based MTC is the prime focus for the upcoming content of the thesis. Simulation results for a specific case, in which bus voltage transition from $V_{bus_{init}} = 90\,V$ to final voltage $V_{bus_{final}} = 150\,V$, is represented in this section to exemplify the CCM based MTC algorithm, the underlying steps for its implementation in detail. System parameters for 3 boost converter stages selected for the simulation purpose are tabulated in Table 2. Capacitance of the bank C_{bank} and load resistance R_{load} is considered as $1.5\,mF$ and $10\,\Omega$ respectively. The ESR for the Capacitor bank is considered as $0\,\Omega$ to simplify the illustration of algorithm and simulation. However, ESR is an important parameter for an overall impedance of the capacitor, it can also be considered non-zero. Mathematical

model developed in section (2.2) accounts for ESR of capacitor for state space representation of paralleled boost converters.

Table 2: Paralleled boost converter system parameters

Parameter	**Stage 1**	**Stage 2**	**Stage 3**
Source Voltage V_S(V)	72.00	65.00	58.00
Inductance L(mH)	3.000	3.200	2.000
ESR of Inductor $R_L(\Omega)$	0.200	0.200	0.200
Maximum Inductor current $I_{L_{max}}$(A)	60.00	60.00	60.00
Switch ON resistance $R_{SW}(\Omega)$	0.020	0.020	0.020
Diode cut-off voltage $V_{D_{ON}}$(V)	0.100	0.15	0.20
Forward resistance of Diode $R_{D_{ON}}(\Omega)$	0.020	0.020	0.020

Three paralleled boost converter system are assumed to be operating in steady state condition under a primary control system, i.e. Sliding mode control illustrated in section (3.1). MTC operation is triggered at time $t = 30\,ms$. For simulation purpose only, MTC results can either be directly loaded from switching instance values t_{switch} or by halting simulation and executing MTC. It is important to note that real-time implementation requires the MTC algorithm to be executed concurrently with the primary control. Step-by-step implementation of MTC algorithm for assumed system is as follows.

Step 1: Initial and Final values of state variables

Initial and final values of inductor currents i_{Ln} calculated from steady state conditions of primary control $V_{C_{init}}$ and $V_{C_{final}}$ as explained in section (3.1) are:

Table 3: Initial and final system states for 3-paralleled boost converters during MTC

Initial Value		**Final value**	
$V_{bus_{inti}}$	90 V	$V_{bus_{final}}$	150 V
$i_{L1_{init}}$	3.7125 A	$i_{L1_{final}}$	10.313 A
$i_{L2_{init}}$	4.1123 A	$i_{L2_{final}}$	11.423 A
$i_{L3_{init}}$	4.6086 A	$i_{L3_{final}}$	12.802 A

Step 2: Calculate maximum ON-time for individual boost converter

Assuming $Q_i \rightarrow ON$, $t_{ON_{max}}$ can be calculated using known boundary conditions.

For assumed parameters of system, $t_{ON_{max}}$for all stages are:

Table 4: Maximum ON time for boost converter switches during MTC

Maximum ON time $t_{ON_{max}}$(sec)	
$t_{ON_{max}}$ for Stage 1	0.0025775
$t_{ON_{max}}$ for Stage 2	0.0030578
$t_{ON_{max}}$ for Stage 3	0.0021525

Step 3: Equivalent contribution time(τ) for each stage:

$$\boldsymbol{\tau_i} = \frac{I_{L_{max}} - I_{init}}{\alpha},\ where\ \alpha = \sum_{i=1}^{3} \frac{V_{s_i}}{L_i}$$

$$\alpha = \sum_{i=1}^{3} \left\{ \frac{72}{0.003} + \frac{65}{0.0032} + \frac{58}{0.002} \right\}$$

$$\Rightarrow \alpha = 73312.5$$

Table 5: Equivalent contribution of each boost converter stage for MTC

Equivalent contribution time τ	
τ_1	0.0007696
τ_2	0.0007641
τ_3	0.0007574

Step 4: Absolute and relative contribution factor(CF)**:**

Contribution factor CF translates the contribution time τ_i into a meaningful quantity that represents proportional share of each stage in the transfer of energy.

$$CF_i = \frac{\tau_i}{\sum_{i=1}^{3} \tau_i}$$

$$CF_{rel_i} = \frac{CF_i}{\min(CF_i)}$$

Table 6: Absolute and relative contribution factor for each boost converter during MTC

Absolute contribution factor		**Relative contribution factor**	
CF_1	0.33591	CF_{rel_1}	1.0162
CF_2	0.33353	CF_{rel_2}	1.0090
CF_3	0.33056	CF_{rel_3}	1.0000

Step 5: Switch time(t_{switch}) over full range of β:

$$\{t_{switch}\} = \beta \times CF_{rel} \times t_{ON_{max}}, \beta \in (0, 1)$$

$\{t_{switch}\}$ denote set of values of switching instances for all boost converters. Both, CF_{rel} and $t_{ON_{max}}$ are of same dimension and the multiplication is performed element-wise. Every β value(scalar) corresponds to a unique set of switching instances. For the assumed system, $\{t_{switch}\}$ can be defined as,

$$\{t_{switch}\} = \beta \begin{bmatrix} 1.0162 \\ 1.0090 \\ 1.0000 \end{bmatrix} \begin{bmatrix} 0.0025775 \\ 0.0030578 \\ 0.0021525 \end{bmatrix}$$

$$\Rightarrow \{t_{switch}\} = \beta \begin{bmatrix} 0.0026192 \\ 0.0030852 \\ 0.0021525 \end{bmatrix} \tag{3.17}$$

Now,$\{t_{switch}\}$ is only dependent on the scalar parameter β, irrespective of number of paralleled boost converter stages.

Step 6: Parameters of characteristic line

Iterate discrete model of paralleled boost converter system for two distinct values of $\beta \in (0, 1)$, $\beta_1 = 0.35$ and $\beta_2 = 0.65$, to find slope and y-intercept of characteristic line.

$$slope\ m = \frac{V_{C_{\beta 2}} - V_{C_{\beta 1}}}{\beta_2 - \beta_1}$$

$$slope\, m = \frac{144.33 - 111.58}{0.65 - 0.35}$$

$$\Rightarrow slope\, m = 109.18$$

$$y - intercept\, C = V_{C_{\beta 2}} - m\beta_2$$

$$\Longrightarrow\ y - intercept\, C = 144.33 - (109.18 * 0.65) = 73.336$$

Step 7: Switch time for MTC t_{MTC}

Calculation of $t_{switch} = t_{MTC}$ using β_{MTC} for the desired end point $V_{C_{final}}$is as follows,

$$\beta_{MTC} = \frac{V_{C_{final}} - C}{m}$$

$$\beta_{MTC} = \frac{150 - 73.336}{109.18}$$

$$\Rightarrow \beta_{MTC} = 0.70189$$

From equation (3.17),

$$t_{MTC} = \beta_{MTC} C F_{rel} t_{ON_{max}}$$

$$t_{MTC} = 0.70189 \begin{bmatrix} 0.0026192 \\ 0.0030852 \\ 0.0021525 \end{bmatrix}$$

$$\Rightarrow t_{MTC} = \begin{bmatrix} 0.0018384 \\ 0.0021655 \\ 0.0015108 \end{bmatrix}$$

here, t_{MTC} denotes switching instances corresponding to minimum time transition from $V_{bus} = 90\,V$ to $V_{bus} = 150\,V$. Switching time t_{MTC} is always referenced to the time when MTC operation execution begins. Simulation results demonstrating SMC+MTC operation are illustrated in Figure 3.8-3.12.

System states during the SMC+MTC operation are shown in Figure 3.8 for time span of $80\,ms$. Minimum time transition is scheduled to be triggered at time $t = 30\,ms$ through step change in V_{ref}, and all the necessary calculations for the MTC operation described previously are completed in advance. Hence, control of all the boost converter switches are transferred from SMC to MTC at $t = 30\,ms$ and continues until time $t = t_{switch}$. The time to revert controls back to SMC depends on the t_{switch} for each boost converter stage. As explained in section (3.2.4), after $t = t_{switch}$, the only possible case for inductor currents i_{L_n} is to return towards the next steady state condition since switch remains OFF until the current reaches to the $i_{L_{final}}$. Rise time of inductor currents depends on inductance(L_i) and ESR(R_{L_i}).

To closely examine the transition results, a time close-up of the MTC operation is shown in Figure 3.9. In order to simplify the representation of switch time and corresponding system states, the time axis of actual simulation is shifted such that MTC begins at time $t = 0\,s$. From this moment all of the boost converter switches are turned ON and the Inductor starts accumulating energy in form of magnetic

field . While the capacitor is detached from all of the stages, discharges through the load resistor and bus voltage starts dropping from the previous steady state level. At time $t = t_{MTC}$, switches are turned OFF and corresponding inductor starts discharging and transfer energy to the capacitor bank. Note the change in slope of the capacitor/bus voltage V_{bus} when switch status are flipped. Since, the assumed topology of paralleled boost converter is non-homogeneous, t_{switch} is different for all stages. Once, the switches are turned OFF, the SMC operation is completed for the corresponding stage and now it is under primary control scheme, here SMC. Once MTC is completed, the only possible state transition is to converge to the steady state current that is $i_{L_{final}}$. Average Duty cycle D_i is derived from the switch signal and shown in Figure 3.10.

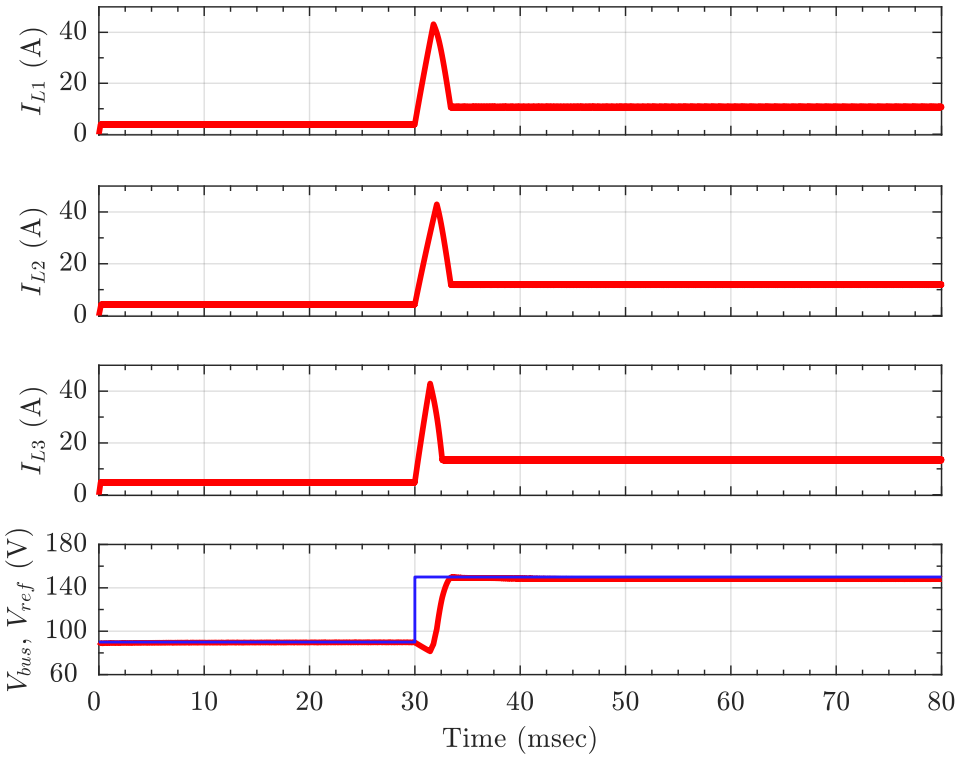


Figure 3.8: System states operating under SMC+MTC operation during simulation; MTC triggered at $t = 30$ ms

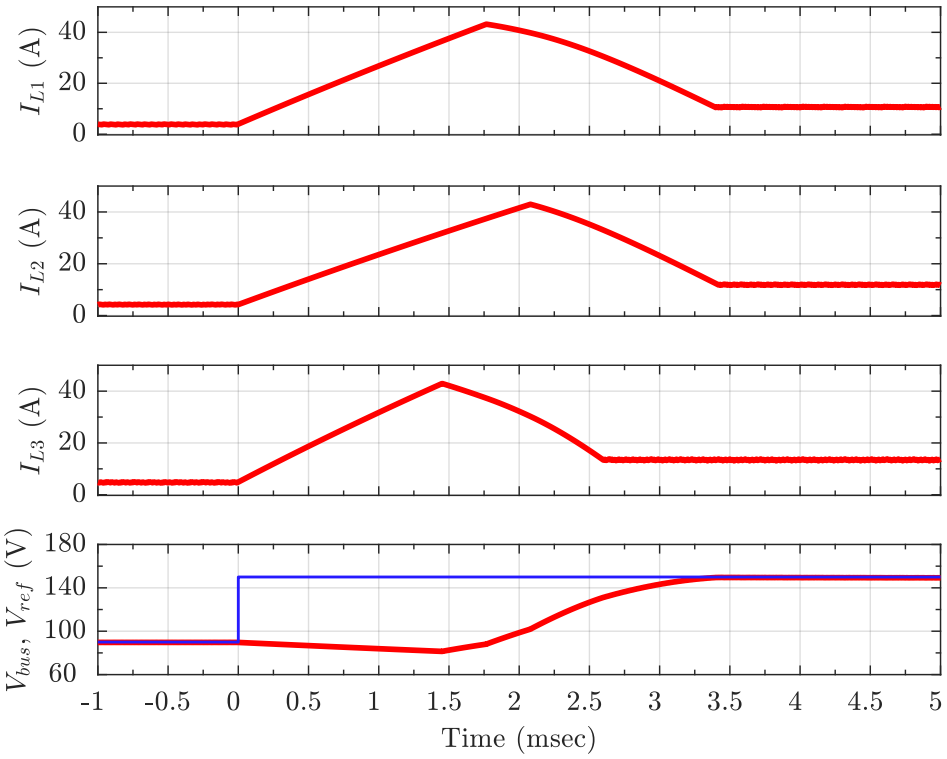


Figure 3.9: System states during simulation of MTC with 3-boost converter stages, plotted with trigger point at time $t = 0$ ms

Since minimum time transition operation transfer optimum amount of energy from source to load, the transaction of energy into the passive elements provides clear insight about the operation. The instantaneous energy into the passive elements during the complete operation is illustrated in Figure 3.11. During steady state execution under SMC, all passive elements maintains a steady level of energy, since average ripple in system states remains zero during steady state. During MTC operation, switches are forced to turn ON for specific duration and a large

portion of energy is transferred to inductors from voltage sources. When these switches turn OFF, the stored energy is released and deposited into load side passive element, capacitor bank. During MTC operation, switches are operated at enormously high currents for a very small amount of time while capacitor voltage remains almost steady and then undergoes through the quick transition of bus voltage.

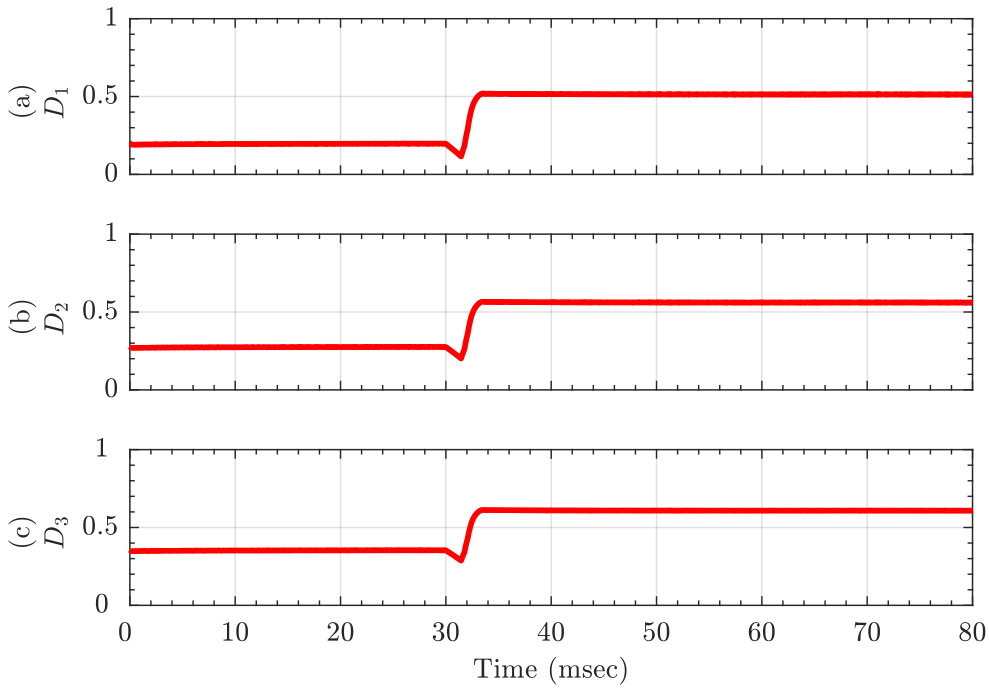


Figure 3.10: Duty cycle (D) for all boost converter stages during SMC and MTC operation, (a) D_1, (b) D_2, (c) D_3

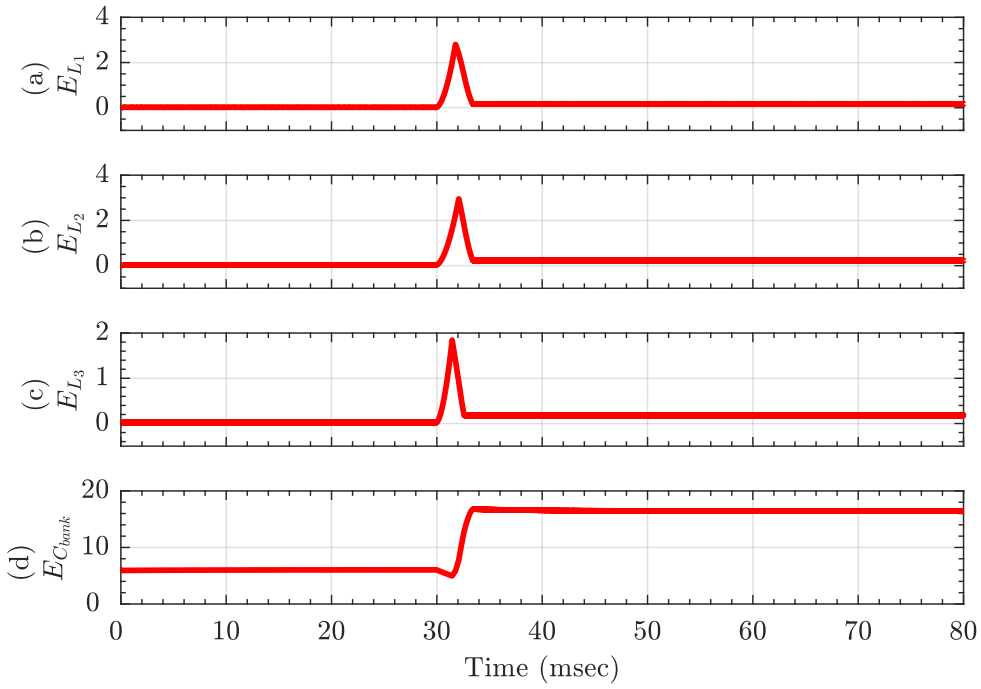


Figure 3.11: Instantaneous Energy (in Joules) stored in passive elements during SMC+MTC operation; (a) E_{L_1},(b) E_{L_2}, (c) E_{L_3}, (d) $E_{C_{bank}}$

An alternative way to represent the minimum time transition operation is through the Phase plane trajectories. Figure 3.12 represents the phase plane trajectories for individual boost converter $i_L \rightarrow V_{bus}$. From the phase plane representation, it is clearly evident that all boost converters operate under complete electrical isolation. Stage 3 inductor current happens to be achieving the steady state well before the remaining stages, but it has no impact on the operation of other converter stages. Stages 1 and 2 still follow their expected trajectory to attain steady state. Due to this functionality, CCM based approach is far superior in terms of implementation complexity. There is no need for scheduling of states to

drive them back to steady state at a same time. Comparing the MTC simulation results demonstrated in this section with the SMC simulation results in section (3.1.2) shows improvement in the transition time(speed of response) by a large extent.

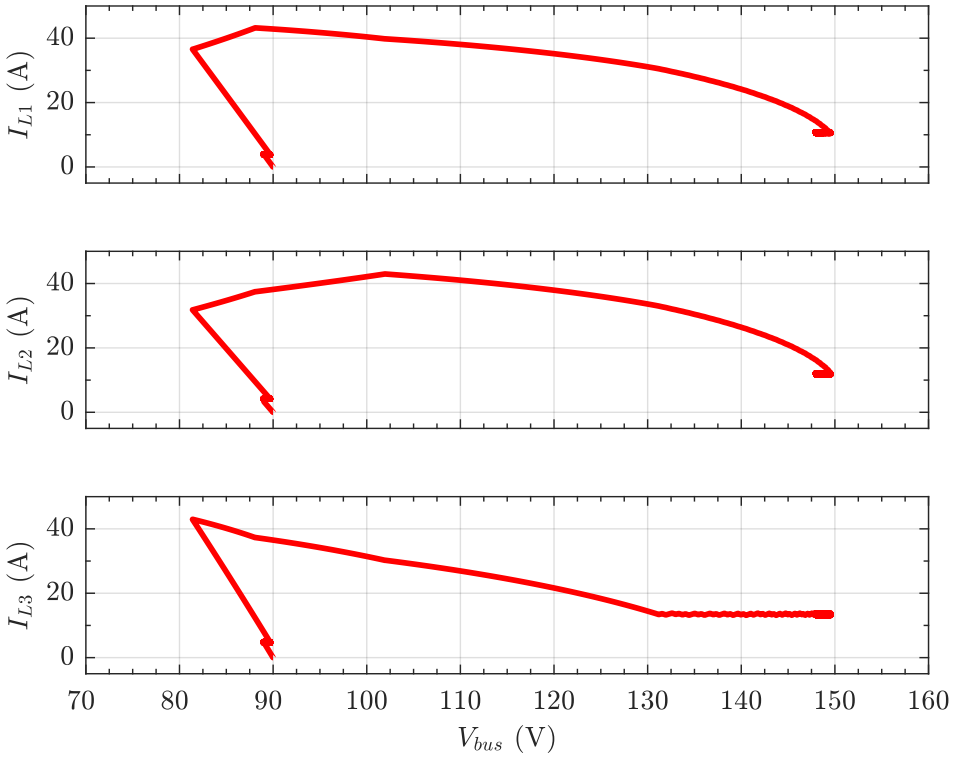


Figure 3.12: Phase plane trajectories $I_{L_n} \rightarrow V_{bus}$ for all boost converter stages over time $t = [0, 80]$ms

There are many fundamental differences when it comes to implementing the translate same control architecture from simulation to real-Time (emulation) environment and it poses several challenges pertaining to hardware (computational resources) limitations. In the following chapter, CCM based algorithm is proposed and implemented for real-time framework. Several design related challenges are

also highlighted in the following chapters.

4 Real-time control system implementation and validation of MTC

The CCM based MTC algorithm was thoroughly represented in chapter 3 where the complete focus was to illustrate the MTC algorithm and analyze the performance of the control system through the simulation results. This chapter focus on the real-time implementation of the CCM based MTC algorithm. Compared to the simulation, the real-time implementation/emulation is very different. While there is no limitation on the amount of computational time during a simulation process, a real-time implementation has very hard bound on the amount of time allotted to execute an update of the system. The restricted time frame to execute the control while updating system IO ports in real-time is a major concern for an high-speed control system. A detailed real-time implementation of MTC from hardware and software perspective is illustrated in this chapter. Emulation results for CCM based MTC are represented and compared with the SMC to highlight the improvement in speed of response of the system. Finally, parameter sensitivity for the MTC is highlighted through the analytical and experimental results to analyze the robustness of MTC algorithm.

4.1 Minimum time control (MTC) system architecture for emulation

In contrast to the simulation, the real-time emulation has two separate components, hardware side and software side, that have to work in combination while

maintaining synchronization. Real time update of time for both components is also a major difference with the simulation only approach. During emulation, the complete system is updated at a constant duration to maintain synchronization and hence create real-time execution of system. Both of these components are discussed in this section.

4.1.1 Hardware implementation for real time MTC

The hardware side of the implementation is comprised of the three components as follows,

1. Real-time emulator to emulate power electronics circuits
2. Target computer
3. Host computer

The real-time emulator is the fundamental component of the system that emulates 3-boost converter based network as shown in Figure 2.9. The emulator used for the real-time experiments is the Typhoon HIL(Hardware-in-loop) 400 that has integrated analog and digital IO ports. The analog signals generated from the HIL system carry system states such as bus voltage and inductor currents that are interfaced with the Target computer. The analog signals primarily serve as the feedback signals fro the MTC or SMC based control. It is also possible to have additional channels carrying intermediate states for supervisory control purposes.

The target computer is the most important component of the real-time system and is responsible for all the computational tasks related to the MTC, primary control routine and supervisory control. The real-time control is executed on target computer with constant model update rate, i.e. $0.1\,msec.$ Within this time frame, the whole control system is updated once and the feedback signals are revised accordingly. Apart from computation, the target computer is also responsible for collecting the system states/feedback signals and storing into the static on-board memory. The time required to acquire all the feedback signals through continuous scanning of IO ports add tremendous amount of workload on target computer. Hence, there is a trade-off between the bandwidth of control system and resolution of the data acquired during real-time emulation.

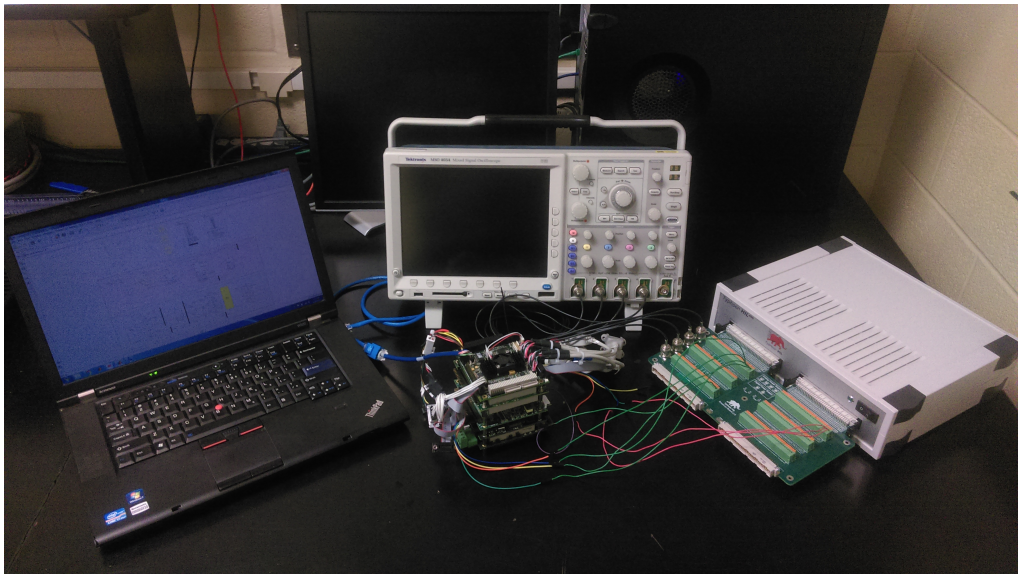


Figure 4.1: Experimental setup for HIL Simulation

The host computer is also a development platform where all components of related to control system and filtering of IO signals are developed which then transferred onto the target computer via Ethernet link. In addition to the development,

the host computer provides a low bandwidth feedback for debugging purposes. Refer to Appendix A.3 for detailed model of the control system developed on host computer for real-time emulation. The MATLAB model is later translated and compiled into the equivalent C code using code-generation tool-chain to be executed on target computer.

Figure 4.1 represents the experimental setup created for the real-time simulation of MTC with SMC as primary control. As mentioned earlier, all the three major components of the setup: the host computer, the target computer and the Typhoon HIL system, are illustrated from left to right. A high-speed digital oscilloscope is used to monitor the major feedback signals during the real-time execution.

4.1.2 Software implementation for real-time MTC

The simulation of MTC illustrated in chapter 3 was implemented through a MATLAB code that executes in serial fashion line-by-line. Whereas the real-time implementation of same MTC algorithm is drastically different because each component of the system has to execute within the model update time duration. So, when the target computer is updating control loop, the IO ports are also collecting feedback signals. During this routine execution of high-priority tasks, inclusion of MTC algorithm requires adequate control of a scheduler. The computational routine for MTC takes longer than a usual model update duration. So, the proper scheduling of the computational overhead is required to divide the low-priority computation over a few cycles of high-priority execution. The multi-threaded/multi-tasking

capability of the target computer can reduce this complexity to a great extent and can improve the bandwidth for high-priority control.

During the real-time execution, the primary control (SMC) is considered as high-priority task and all of the boost converter stages are operating under SMC. The moment MTC is triggered, the sequence of execution is as follows:

1. Approve for MTC algorithm by validating systems states current values
    - The validation of MTC is done through the current bus voltage and desired bus voltage at the end of the transition. All requests for final bus voltage of higher than 110% of, and less than 220% of initial bus voltage are considered as feasible transitions and approved for further calculations
    - The system states captured at this moment are considered as the initial conditions for the rest of the calculation
2. Begin the execution of MTC algorithm for CCM based approach
    - Set the "Simulation enable" signal to indicate the status of MTC execution
3. Begin simulation of multiple boost converter system for β_1 and β_2 consecutively
    - An internal strobe signal is used to handle consecutive execution so that the computation process do not exceed the model update time
4. Calculate final beta value β_{MTC} and switch timings

5. Trigger the MTC operation and validate the current system states

    - If there is more than 5% of deviation from initial system states, the MTC trigger is disabled and the calculations for MTC algorithm will be repeated with updated system states

6. The switching signals (PWM signals to the power switches) are transferred from SMC to MTC based scheduler that keeps each switch ON for duration t_{sw_n}

    - As soon as a boost converter stage crosses the MTC switching duration t_{sw_n}, the scheduler will handover the stage to primary control

7. After all of the boost converter stages are moved to primary control, the real-time execution of the MTC is considered as completed

    - All the internal strobe and flag signals are reset and new MTC execution cycle is enabled

The primary control is active throughout the execution of these steps. During the execution of these steps, the target computer has to take maximum burden and if the model update rate is not sufficiently high enough to take computational burden then the target computer might get overloaded and terminate the complete execution. This situation can be avoided by multi-tasking/multi-threading capability of the target computer by spiting time sensitive operations and low-priority calculations. The multi-threading can also improve the overall execution time by reducing the calculations per thread. So, with single core dual threaded target

computer, the execution of two β cases can be split between threads and hence improve the overall calculation time.

4.2 HIL simulation results and analysis

In this section, a real-time MTC execution of 3-paralled boost converter stages is illustrated for a specific case, in which bus voltage transition from $V_{bus_{init}} = 90\,V$ to final voltage $V_{bus_{final}} = 150\,V$. System parameters for 3 boost converter stages selected for the simulation purpose are tabulated in Table 2. Capacitance of the bank C_{bank} and load resistance R_{load} is considered as $1.5\,mF$ and $10\,\Omega$ respectively. The ESR for the Capacitor bank is considered as $0\,\Omega$. For comparison purpose the circuit and its components are assumed as identical to those of used during simulation.

Figure 4.2 represents the set of internal signals used for scheduling purpose during MTC execution. As shown in Figure 4.2(a), the MTC algorithm is triggered at time $0\,ms$.Along with the trigger, a "simulation enable" signal (in Figure 4.2(b)) is also set to indicate that the MTC execution is active. The use of these signals ensure that only one instance of algorithm can be active at a time and the execution remains active until its over, so that any MTC trigger in between the process can not interrupt. Figure 4.2(c) depicts a digital signal that handles sequential execution of two separate β cases., by issuing new trigger at the of calculation for one of the β. The first simulation case for β_1 begin at $0\,ms$ and the remaining case for β_2 begin at $4\,ms$ .The computation part after that takes very cycles to determine the switch timings t_{MTC} . After a slight delay of around $1\,ms$, real-time

execution of MTC is triggered and the scheduler changes from primary control (here SMC) to MTC. As shown in Figure 4.2(d), the MTC execution begin at 10 ms. A simulation enable signal is reset at the same moment that clear all of the internal signals so that next computation can begin immediately.

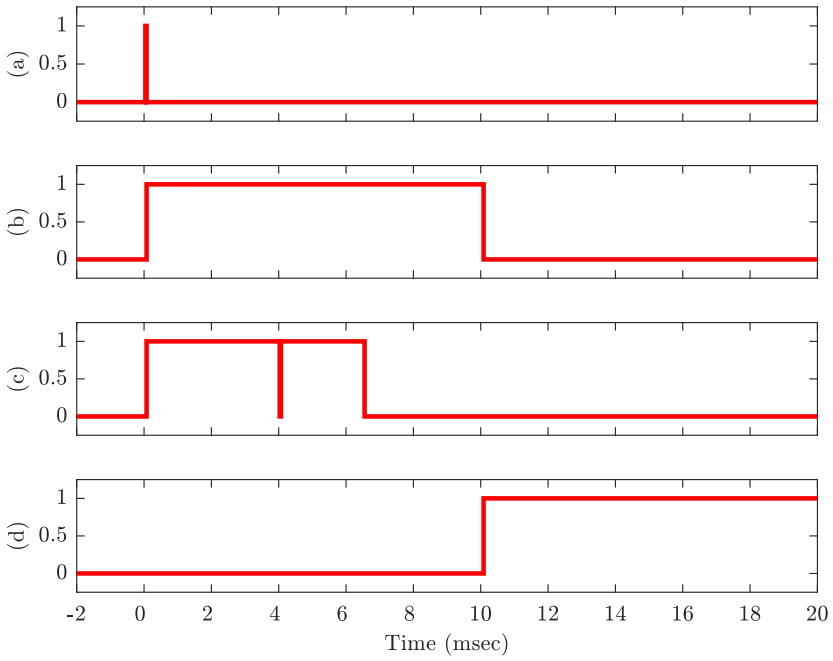


Figure 4.2: Trigger signals during concurrent execution of MTC; (a) Simulation for MTC algorithm triggered, (b) Simulation enable signal, (c) sequential strobe signals indicating execution of MTC algorithm for two β values, (d) MTC trigger for Real-time execution

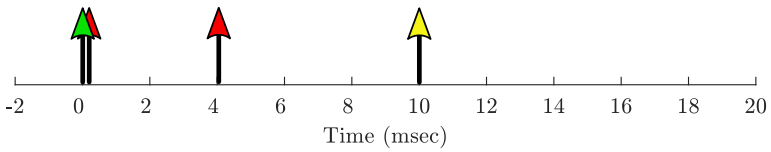


Figure 4.3: Timeline of trigger signals for execution of MTC algorithm in real-time

Timeline of events during execution of MTC is illustrated in Figure 4.3, where

the execution is triggered at time $t = 0\,ms$. Next immediate action is to start simulation for one of the two β cases. Note that the arrow (red pointed) right after the trigger signal indicates consecutive execution. At time, second β case simulation begins and terminates around $t \sim 6.1\,ms$. From this point onward, remaining calculations are carried out to find switching instances and the real-time MTC execution is triggered at time $t = 10\,ms$. The series of events will repeat every time MTC execution is trigger. Here importance of sequential switching of tasks is to distribute the computation load uniformly over a time span so that the simulation process, which can be accomplished in non-real-time priority, while high-bandwidth control loop executed primary control. Once ready, the MTC trigger will force the switches to operate in MTC mode and will return back to primary control after steady state is achieved from minimum-time transition.

Figure 4.4 depicts the system states for both β cases. The final values of system states derived from these simulations are used to calculate the switch time t_{MTC}. To finish the calculation, the target computer takes around $7\,ms$ while executing primary control simultaneously. Here, it is important to note that the time required for the simulation process depends on the final value(desired steady state value) of bus voltage. Higher the final value of bus voltage, higher the final value of inductor currents and hence higher the value of τ. So, higher bus voltage transitions take more time compared to the one with lower transition voltage.

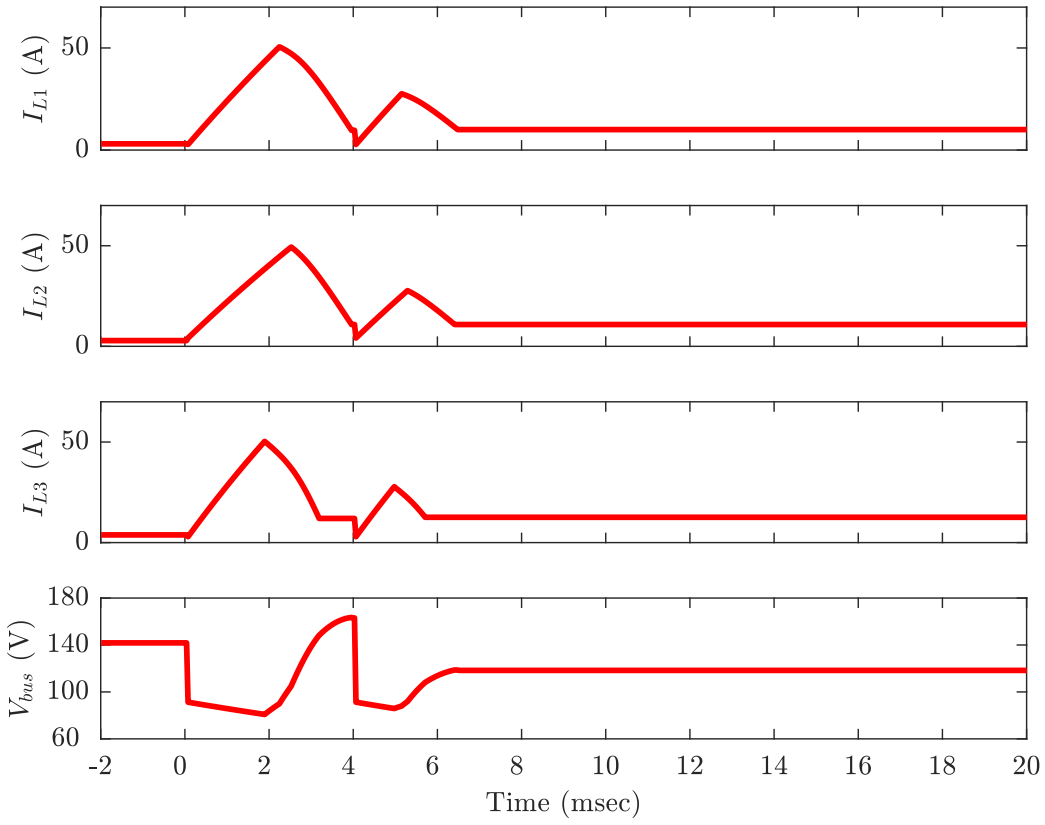


Figure 4.4: System states during execution of MTC algorithm

Figure 4.5 show the switching signals applied to each power converter stage. Until time $t = 10\,ms$, the switching signal of low duty cycle belong to the primary SMC control routine. Then after timing control is moved to the MTC based scheduler. Each boost converter stage is kept is ON for its respective t_{SMC} during. Note switching instances for each stage according to calculated value of $t_{MTC} = \begin{bmatrix} 0.00183 & 0.00216 & 0.00151 \end{bmatrix}\ ms$. Switches are turned OFF after this duration and the scheduler control is moved back to the primary control. Since, the inductor current values are reducing during OFF period from its peak, the primary controller will continue operating in steady state.

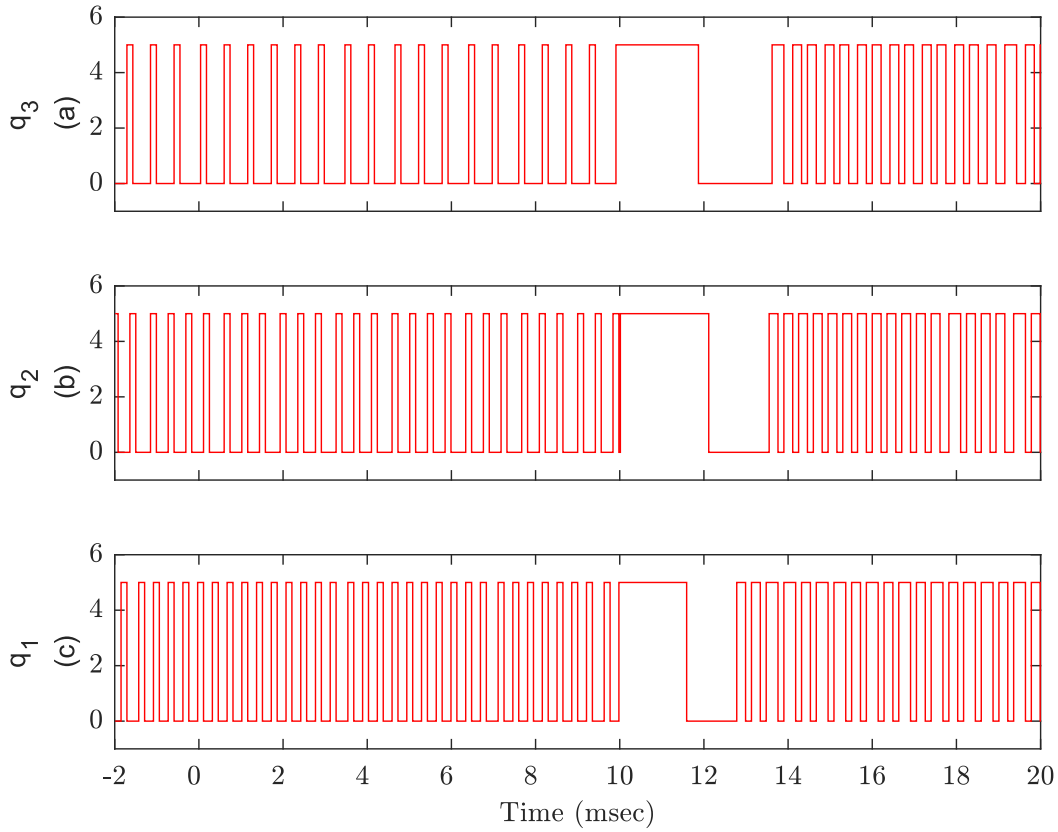


Figure 4.5: Switching signals applied at gates of power switches

Figure 4.4 and Figure 4.6 represent system states in time domain and in form of phase plane trajectories respectively. Till 10 msc, the system states are operating in steady state under primary control (SMC). Then the SMC transition begin and new steady state is achieved by all of the boost converter stages in minimum possible amount of time. Then after the primary control takes charge of the emulation and continue to operate in new steady state. Note smooth transition of bus voltage without an overshoot. Another important observation can be drawn from the experimental results is that not all of the stages return to steady state at a specific instance, rather they return to the steady state with respect to the

contribution of each stage. Since the parameters of boost converters are non-identical, it takes different time duration to return to steady state. If all of boost converter stages are identical then contribution factor for all of them will also be identical and the system response would like almost identical for all stages. In that case, all of the boost converter stages return to steady state at the same instance.

From phase plane trajectories, it can be concluded that the MTC transition achieve steady state with just a single switch cycle (from ON→OFF), resulting in minimum possible time. Initially, the bus voltage starts plummeting due to capacitor discharging through load. Note the switching instances represented by sudden change in phase trajectory.

The experimental results for MTC emulation results are recorded on to the target computer with time resolution of $10\mu s$. Following are the results captured through digital oscilloscope. Figure 4.7 illustrates result for time division of $1s$. It is almost infeasible to identify the gradual decrease and then rapid rise in bus voltage (in green). The inductor current (lower three graphs in sequence with boost converter stages) spikes are just noticeable. Figure 4.8 represent horizontally expanded version of same results with time division $t = 10\,ms$, where transition in system states can be identified as close as of emulation.

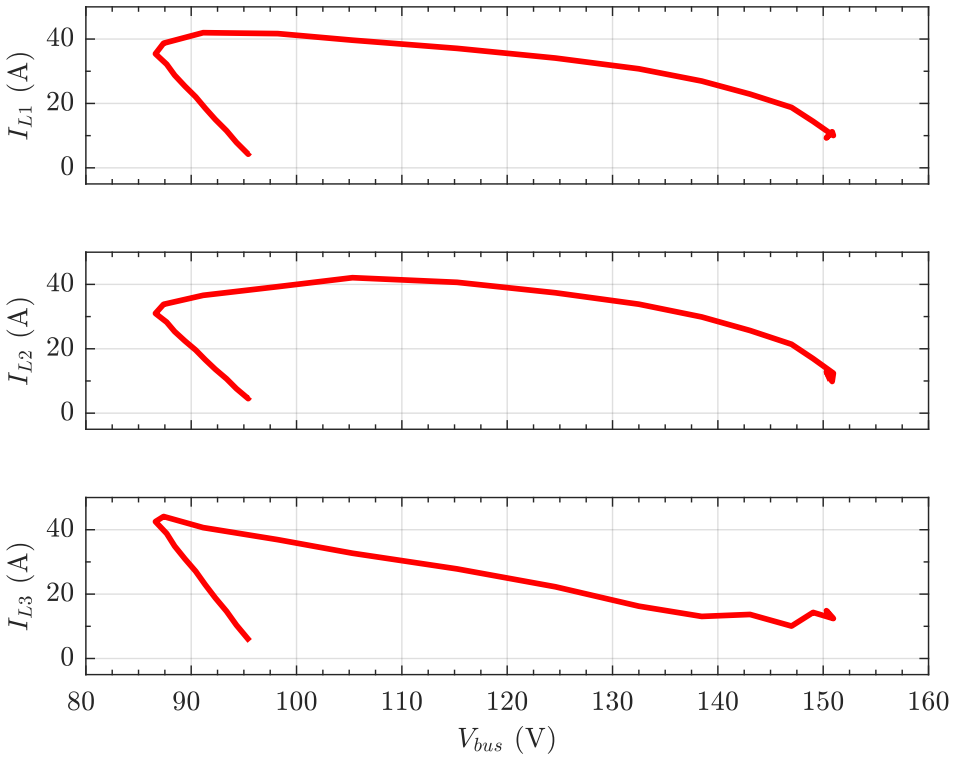


Figure 4.6: System states during real-time execution of MTC projected as phase plane trajectories

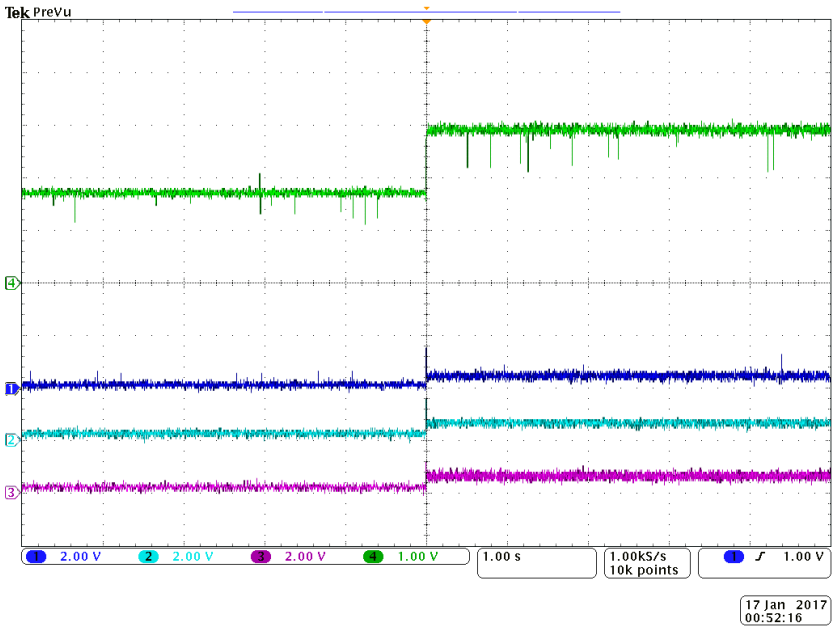


Figure 4.7: Oscilloscope capture of MTC operation on time scale of 1 s

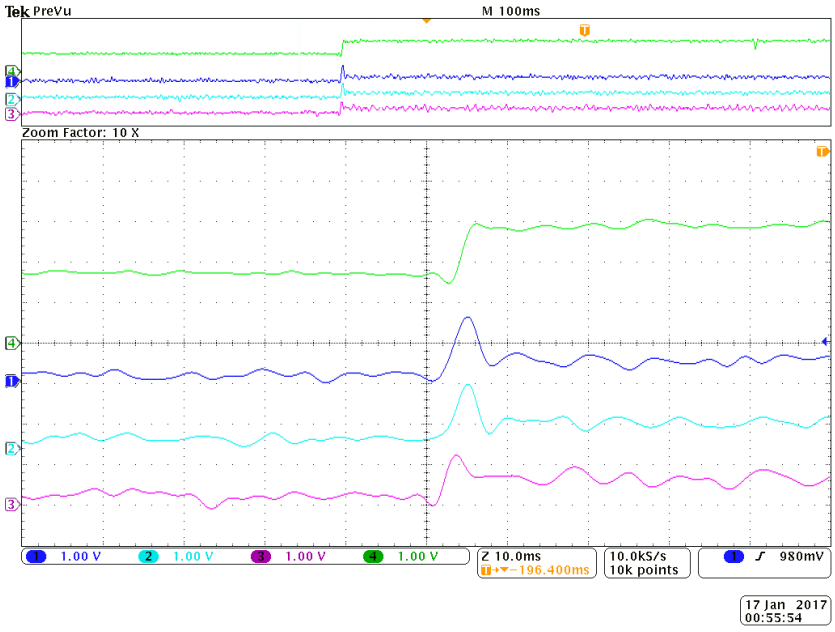


Figure 4.8: Horizontally stretched oscilloscope capture of MTC operation on time scale of 10 ms

4.3 Comparison of performance between SMC and MTC

In previous section, experimental results for an MTC execution accompanied by SMC as a primary control were illustrated. In order to compare the effectiveness of the MTC over any other robust control method, the same system was emulated solely on SMC based control method. Experimental results for both real-time emulation cases are illustrated in the following sections to highlight the effectiveness in response time/speed of response.

4.3.1 SMC vs MTC time domain comparison

Time domain representation for system states during only SMC emulation are shown in Figure 4.9, where a bus voltage makes the transition from initial value $95.12\,V$ to final value $149.6\,V$ within $\sim 40\,ms$. A notable difference in only SMC emulation is that the bus voltage transition is very smooth while inductor currents switch from initial steady state to final steady state almost instantly, without any overshoot. The final inductor current levels will gradually transfer energy to the capacitor, ultimately increasing bus voltage.

In contrast, the MTC+SMC based emulation result shown in Figure 4.10 illustrate that the bus voltage make almost instantaneous transition from initial bus voltage $95.55\,V$ to final bus voltage $150.3\,V$ within merely $\sim 4\,ms$. The difference in the speed of response for both methods, only-SMC and SMC+MTC, is substantially high. MTC based emulation is nearly 10 times faster than Only-SMC based robust control. While the MTC based transition has only one switch cycle, very small

duration like 4 ms makes MTC one of the best feasible methods to implement minimum time control for multiple boost converters.

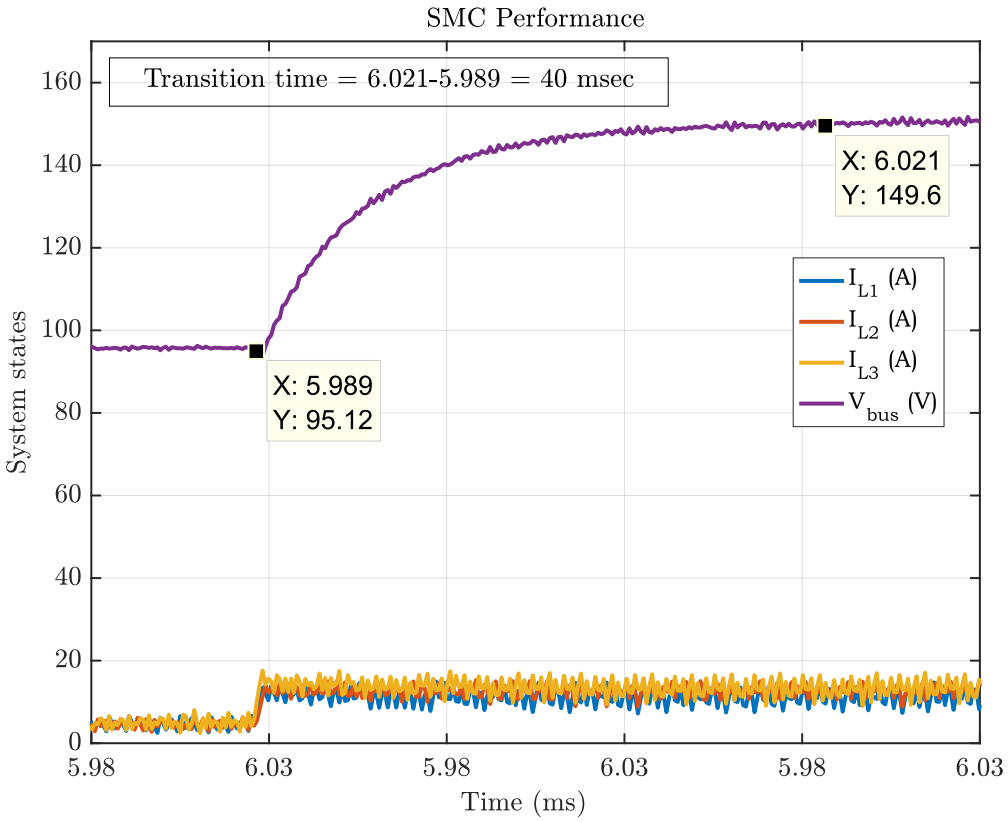


Figure 4.9: System states during real time execution of SMC

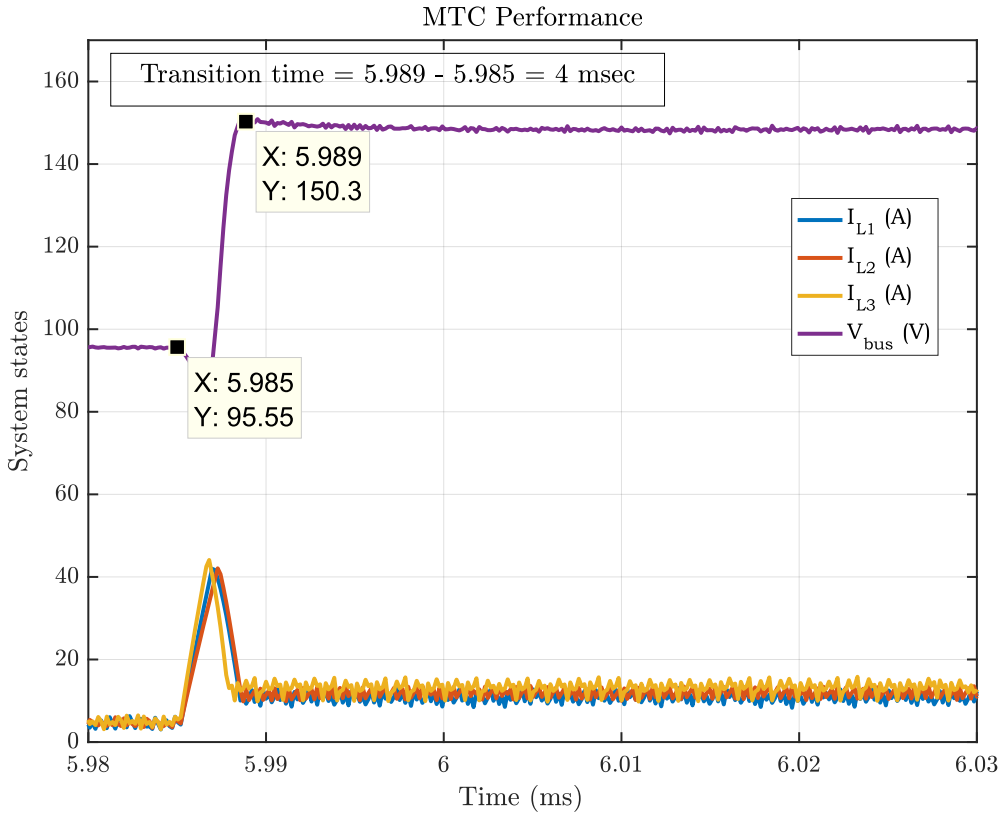


Figure 4.10: System states during real time execution of SMC

4.3.2 SMC vs MTC phase plane comparison

Phase plane trajectories provide detailed perspective for the analysis of both emulation cases. For only-SMC based emulation, inductor currents make instantaneous transition between initial and final steady states while gradually moving the bus voltage to its final steady state condition. The gradual transition of bus voltage forces inductor currents to be corrected at every control update and switches have to turn ON and OFF during this period to maintain the desired level.

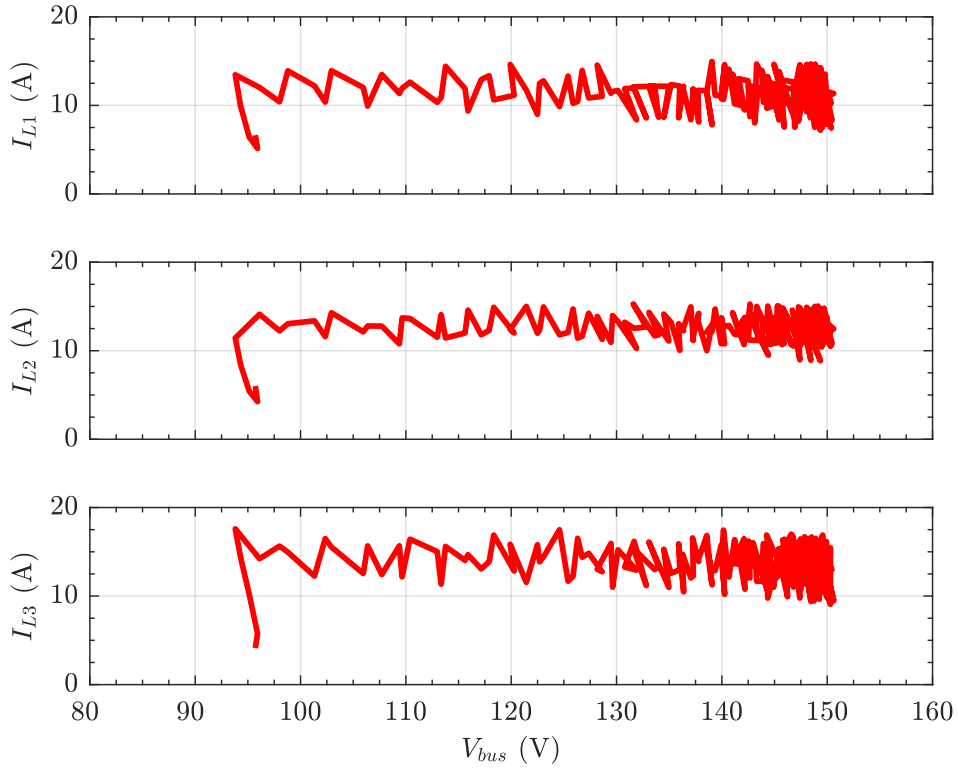


Figure 4.11: System states during real-time execution of SMC projected as phase plane trajectories

Phase plane trajectory for MTC based emulation is shown in Figure 4.12. The major difference here is the amplitude of the inductor currents and switching rate. Due to a single switching cycle MTC based approach forces inductor currents to rise to an extremely high level. It indicates the sudden transfer of energy from inductors to the capacitor bank. Note the variation in system states that originates from lower sampling rate for analog-to-digital conversion. Due to hardware limitations, there is a trade-off between choosing higher update rate for control loop and recording system states in real-time. For the experimental results shown hereafter are recorded with 0.1 ms. With faster processing capabilities, higher

sample can significantly increase bandwidth of recorded signals.

From the phase plane representation, it can be deduced that the MTC based approach facilitate high-speed transition of bus voltage compared to any other robust control methods. It is also noteworthy that quick speed of response is achieved by forcing system states, inductor currents, to operate to their extreme levels. In order to safeguard the hardware during MTC, it is recommended to have a high-priority real-time supervisory control over the MTC implementation.

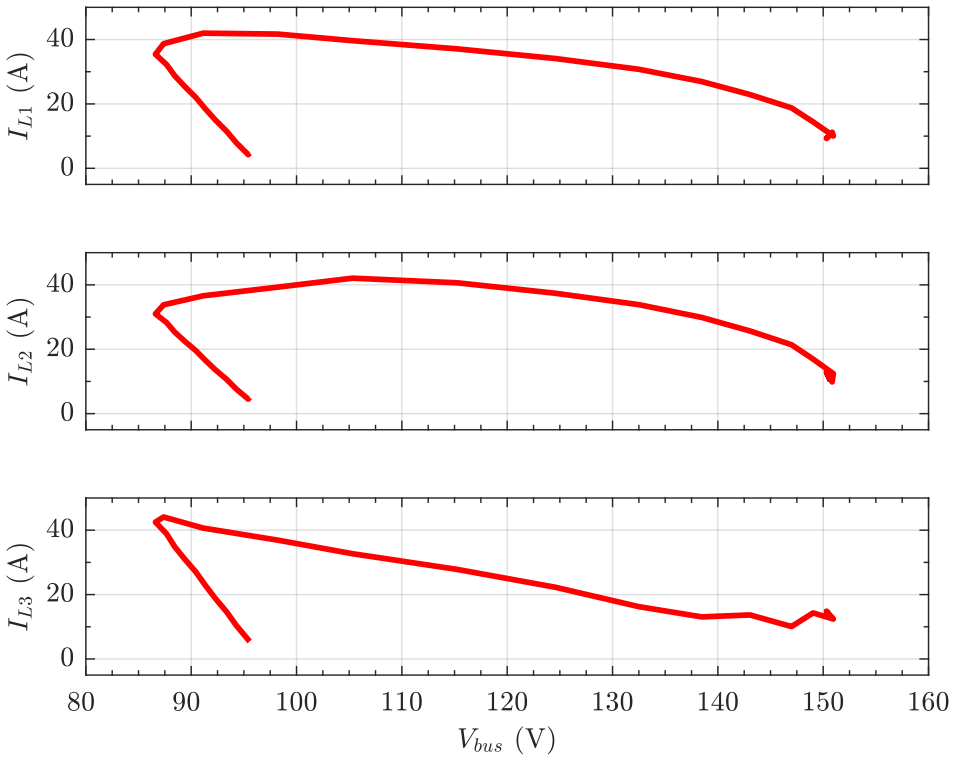


Figure 4.12: System states during real-time execution of MTC projected as phase plane trajectories

4.4 Parameter sensitivity analysis

One of the performance measures for a control system is the robustness. Ability to[26][27] withstand any unaccounted deviation in system parameters is a desired characteristic for a robust control implementation. In this section, system parameters like inductance and ESR of the capacitor bank are considered for measuring deviations in overall response during MTC. The measurement of parameter sensitivity is done by fining deviation in the final value of bus voltage. For comparison, emulation results presented before are considered as reference. The sensitivity analysis for MTC consider variation in: inductance of all boost converter stages within -30% to 30% of their nominal values, and ESR from 0Ω to $0.2\,\Omega$. Emulation results for all of these combinations are illustrated in Appendix A.4.

Figure 4.13 depicts the final bus voltage achieved through MTC with respect to the variation in inductance value. The expected final value of bus voltage is $150\,V$. It can be observed that the total deviation of bus voltage is around $\pm 8V$ for $\pm 30\%$ of deviation in inductance. In addition to that, a linear behavior can be observed in variation of bus voltage and it can be justified with the fact that higher the inductance than expected, longer the switch time.

Similarly, Figure 4.14 illustrate the deviation in bus voltage over the variation in ESR of capacitor bank. For range of $0 - 0.2\,\Omega$, the bus voltage varies in between $140V$ to $149V$. As noted previously for inductance, unaccounted higher ESR value will draw energy from capacitor bank during MTC operation and hence overall energy transfer is slightly lower than expected resulting in lower bus voltage.

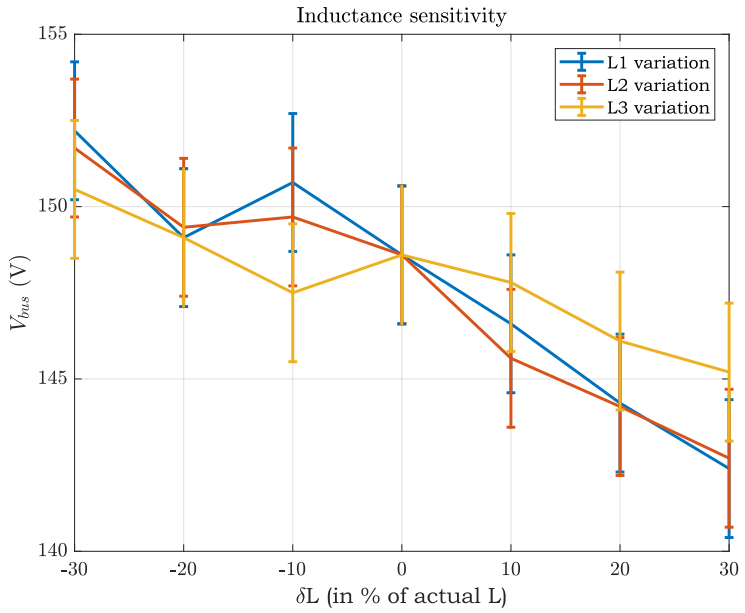


Figure 4.13: Sensitivity of V_{bus} with respect to inductance L_i of individual boost converter stage δ

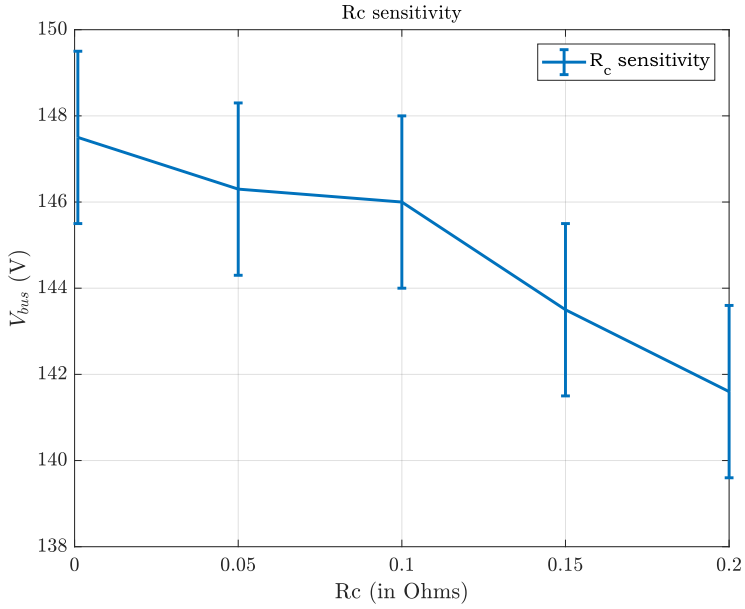


Figure 4.14: Sensitivity of V_{bus} with respect to ESR R_c of the Capacitor bank C_{bank}

5 Conclusion and Future Work

In this conclusive chapter, an overview of the work reported in this thesis is provided. In addition to that several key aspects that can significantly improve the performance of the the proposed work is also discussed briefly. Some notable challenges in implementation of those approaches are also highlighted.

5.1 Thesis summary

An optimal control method to to minimizing transition time for paralleled boost converter is introduced in this thesis. The proposed topology of the non-homogeneous paralleled boost converter includes a large subset of several known configurations, i.e. multiple boost converters, multi-phase boost converters or interleaved boost converters. The minimum time control method proposed and implemented in this thesis cover all of the previously mentioned configurations in a form of a special case of paralleled boost converter. One fundamental difference of the proposed method with previously studied methods is that it does not assume ideal or identical system components to simplify the overall system complexity from computation point of view. While the proposed method assumes all necessary parameters of the system, and, the complexity of system is not limited, it reflects more resembling implementation of practical system and results derived from it.

The effectiveness of the minimum time control is not just the reduction in real time transition of system states but also the time required to make necessary computation to implement the method successfully in real-time workflow. There

exist several techniques based on conventional optimization approach that can take extensive amount of computation power and hence required to be implemented offline for whole operational range, the real-time implementation of minimum time control provide light-weight alternative to time critical/high-speed operations.

One of the most important feature of the proposed MTC is that it can be scaled to n-level without adding substantial cost for computation. The concept of contribution developed for each stage makes this method easily expandable to create an aggregated configuration for very high power applications.

5.2 Future Work

A brief overview of future work proposed in this section entails improved execution of MTC for real time controller and detailed system modeling to accommodate non-linear behavior of system components.

5.2.1 Concurrent execution for Real time MTC

As described thoroughly in the chapter 4, the implemented real time control system for this thesis incorporated a single core processor with multi-threading capability that executed simulation of two β cases for finding optimal switching instances through the characteristic line. So, the controller was busy updating all I/O ports and their associated calculations were running along side with the MTC algorithm. The calculation overhead added to the routine process of primary control reduces the effective update rate for the controller and hence the

resolution/bandwidth of the overall control system. Figure 4.8 and 4.9 illustrate the impact on resolution of analog-to-digital conversion.

In order to overcome this problem and to utilize full potential of the main controller for the primary operation, it is recommended to transfer the load of MTC simulation to another dedicated processor/controller so that both- primary control system and MTC algorithm can execute at their maximum speed improving overall time to trigger Real-time execution. One added advantage of having a separate processor is that calculation for each β case can be done through multi-threading and even faster performance can be achieved. Hence, the separate computational hardware can significantly reduce the overall execution time of MTC.

5.2.2 Non-linear modeling of system parameters

It is assumed that the system components remain unchanged during the MTC operation and that is reflected through the static system parameters. Since the MTC operation tends to drive all of the system states to close to their maximum possible operating range, it is highly possible that the impedance of the passive elements might change significantly and deviate the final operating state.

Some of the known parametric deviations occur through variation in inductive impedance. Inductance of a coil is very sensitive parameter as it can be easily affected by the passive inductance of a nearby magnetic parts. The operating frequency of an inductor also plays crucial role in deciding overall inductance of a coil. As noted in several references, the effective inductance of coil reduces at high frequency. Another critical factor for inductance value is the current flowing

through the coil. The MTC operation forces inductor to charge at higher energy levels by pushing a large amount of current in very short duration. Due to this quick rise in current, magnetization in the core approaches towards saturation, non-linear portion of a B-H curve, and, resulting into different impedance value at different current levels.

Similar phenomenon of parametric deviation is also known for capacitive impedance, in which, the equivalent series resistance (ESR) changes according to the operating frequency and temperature. Variation in ESR represents the losses in load side capacitor bank that can affect the resolution of bus voltage. As shown in the sensitivity analysis results for ESR, it is clearly evident that a nominal parameter variation can sway the steady-state performance and settling time of overall system.

5.2.3 GPU based implementation of MTC as an alternative

The emulation results illustrated throughout this thesis were recorded from the Intel Atom process based SBC (single board computer) and all the computation that took place during any of implementation of MTC were sequential in nature. The primary motive of this thesis was to develop a method that can provide bus voltage transition in minimum possible time and it should be feasible to implement during real-time workflow. MTC can achieve this goal by minimizing the need for hybrid optimization down to only two simulation cases and then fining operating point from the linear characteristics as observed before. So, the advantage of having MTC implemented along side the primary control is that the computation

burden is reduced significantly to any of the known methods.

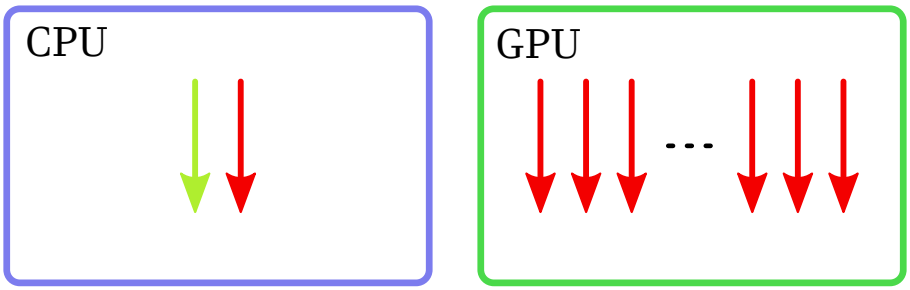


Figure 5.1: Execution of tasks in CPU(few high-speed heterogeneous threads) vs GPU(large number of homogeneous threads at moderate speed)

As an alternative to the CPU based implementation, another approach based on GPU (graphical processing unit) to find optimal switching instance was briefly explored during this research. Figure 5.1 shows inner working of two different computational architectures. CPU can handle very few non-homogeneous threads sequentially at much higher clock frequency, whereas GPU are capable of executing a large number of identical threads in parallel at a same time with lower clock frequency. GPUs are highly efficient at executing same task for multiple instances due to abundance of cores compared to CPUs. Of course, the frequency of CPU core is much higher than of a single GPU core, but the amount of processing a GPU can handle is significantly higher. A simple case of 2048 simulations, very similar to one that was used for one of the β case during MTC, were launched on a 256 core Nvidia Maxwell architecture GPU to measure the overall time of execution. While a CPU running took around $10msec$ to execute 2 simulation cases (refer Figure 4.2 and Figure **??**, GPU finished 2048 simulation cases in $\sim 60msec$(in more than 10 observations). CPU takes exact same time as of GPU to finish just 12 simulations. That is a very stark difference in efficiency of computation.

While GPU can expedite the computational process there are several trade-offs that also need to be considered. Some of the downsides can be listed as :

1. GPU is not a real-time (hard-timed) computer as CPU. Unlike CPU, execution time of same code GPU can take different amount of time on different run. Although there is not much difference in execution time but it cannot be strictly predicted or optimized like CPU.

2. GPU is a hardware dedicated to computation that has multiple instances of same task, so it benefits from pipeline instruction structure. Lower number of instances would add more delay to overall operation due to lower memory bandwidth.

3. GPU can not be used as standalone computer. It require an additional host computer to launch computation tasks (kernel) onto GPU. Launching kernel onto GPU and collecting results back to the host computer takes a significant amount of overall time duration. Memory bandwidth is one of the bottleneck for efficiency of GPU. This is the reason why GPU always work most efficiently for large number of threads, so that it can minimize memory latency and kernel overload.

4. GPU is expensive and power hungry device. For any cost effective solution both of these factor play big role for selection of GPU.

Although GPU based implementation is not proven to be faster than the MTC implementation running on CPU, for a large system with higher number of boost converter, overall execution time could be very close or even lesser than of CPU.

References

[1] C. Shi, B. Miller, K. Mayaram, and T. Fiez, "A multiple-input boost converter for low-power energy harvesting," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 58, pp. 827–831, Dec 2011.

[2] K. Sano and M. Takasaki, "A boost conversion system consisting of multiple dc-dc converter modules for interfacing wind farms and hvdc transmission," in *IEEE Energy Conversion Congress and Exposition*, pp. 2613–2618, Sept 2013.

[3] X. Yu and M. Zhihong, "Fast terminal sliding-mode control design for nonlinear dynamical systems," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 49, pp. 261–264, Feb 2002.

[4] R. de Castro, J. P. Trovao, P. Pacheco, P. Melo, P. G. Pereirinha, and R. E. Araujo, "Dc link control for multiple energy sources in electric vehicles," in *IEEE Vehicle Power and Propulsion Conference*, pp. 1–6, Sept 2011.

[5] O. Hegazy, J. V. Mierlo, and P. Lataire, "Modeling and control of interleaved multiple-input power converter for fuel cell hybrid electric vehicles," in *International Aegean Conference on Electrical Machines and Power Electronics and Electromotion,*, pp. 408–414, Sept 2011.

[6] S. H. Choung and A. Kwasinski, "Multiple-input dc-dc converter topologies comparison," in *Conference of IEEE Industrial Electronics*, pp. 2359–2364, Nov 2008.

[7] A. Kwasinski, "Identification of feasible topologies for multiple-input dc-dc converters," *IEEE Transactions on Power Electronics*, vol. 24, pp. 856–861, March 2009.

[8] G. E. Pitel and P. T. Krein, "Minimum-time digital control with raster surfaces," in *Workshop on Control and Modeling for Power Electronics*, pp. 1–8, Aug 2008.

[9] J. Cheng, J. Shi, and X. He, "A novel input-parallel output-parallel connected dc-dc converter modules with automatic sharing of currents," in *Proceedings of The 7th International Power Electronics and Motion Control Conference*, vol. 3, pp. 1871–1876, June 2012.

[10] V. J. Thottuvelil and G. C. Verghese, "Analysis and control design of paralleled dc/dc converters with current sharing," *IEEE Transactions on Power Electronics*, vol. 13, pp. 635–644, Jul 1998.

[11] R. F. Foley, R. C. Kavanagh, W. P. Marnane, and M. G. Egan, "Sensorless current-sharing in multiphase power converters," in *IEEE Workshops on Computers in Power Electronics*, pp. 117–122, July 2006.

[12] . Yazici and E. K. Yaylaci, "Fast and robust voltage control of dc-dc boost converter by using fast terminal sliding mode controller," *IET Power Electronics*, vol. 9, no. 1, pp. 120–125, 2016.

[13] G. Pitel, "Fast power converters and rapid digital design," 2008. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2016-05-28.

[14] D. J. Perreault, K. Sato, R. L. Selders, and J. G. Kassakian, "Switching-ripple-based current sharing for paralleled power converters," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 46, pp. 1264–1274, Oct 1999.

[15] G. Feng, E. Meyer, and Y. F. Liu, "A new digital control algorithm to achieve optimal dynamic performance in dc-to-dc converters," *IEEE Transactions on Power Electronics*, vol. 22, pp. 1489–1498, July 2007.

[16] R. D. Middlebrook and S. Cuk, "A general unified approach to modelling switching-converter power stages," in *IEEE Power Electronics Specialists Conference*, pp. 18–34, June 1976.

[17] W. Jiang, Y. f. Zhou, and J. n. Chen, "Modeling and simulation of boost converter in ccm and dcm," in *IEEE Power Electronics and Intelligent Transportation System*, vol. 3, pp. 288–291, Dec 2009.

[18] D. Maksimovic, A. M. Stankovic, V. J. Thottuvelil, and G. C. Verghese, "Modeling and simulation of power electronic converters," *Proceedings of the IEEE*, vol. 89, pp. 898–912, Jun 2001.

[19] J. Sun, D. M. Mitchell, M. F. Greuel, P. T. Krein, and R. M. Bass, "Averaged modeling of pwm converters operating in discontinuous conduction mode," *IEEE Transactions on Power Electronics*, vol. 16, pp. 482–492, Jul 2001.

[20] V. Utkin, "Variable structure systems with sliding modes," *IEEE Transactions on Automatic Control*, vol. 22, pp. 212–222, Apr 1977.

[21] S. R. Sanders and G. C. Verghese, "Lyapunov-based control for switched power converters," in *IEEE Power Electronics Specialists Conference*, pp. 51–58, 1990.

[22] M. Lopez, L. G. de Vicuna, M. Castilla, J. Matas, and O. Lopez, "Control loop design of parallel connected converters using sliding mode and linear control techniques," in *IEEE Power Electronics Specialists Conference. Conference Proceedings*, vol. 1, pp. 390–394 vol.1, 2000.

[23] S. C. Tan, Y. M. Lai, and C. K. Tse, "General design issues of sliding-mode controllers in dc-dc converters," *IEEE Transactions on Industrial Electronics*, vol. 55, pp. 1160–1174, March 2008.

[24] X. Xu and P. J. Antsaklis, "Optimal control of switched autonomous systems," in *Proceedings of the IEEE Conference on Decision and Control*, vol. 4, pp. 4401–4406 vol.4, Dec 2002.

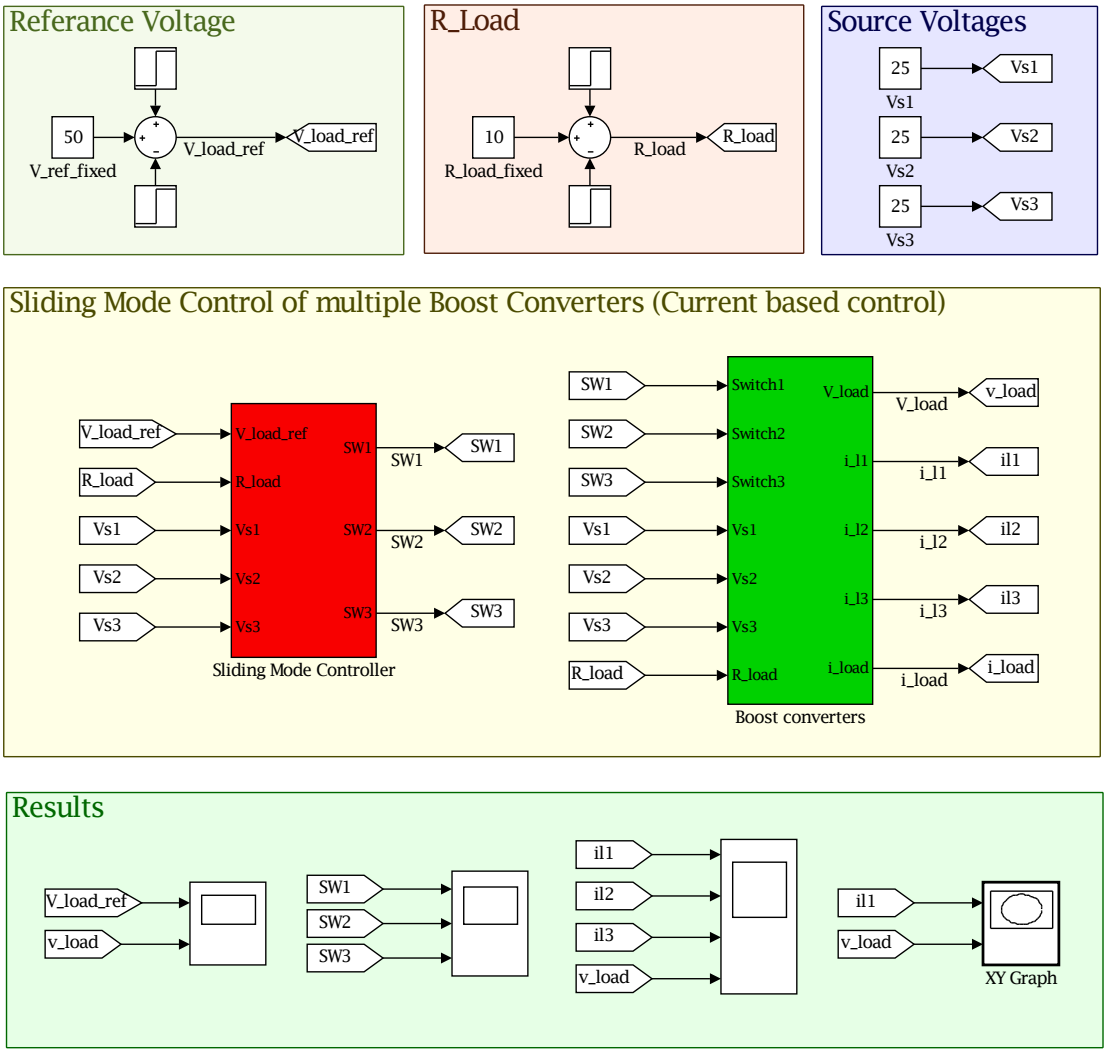
[25] J. Kinable, A. A. Cire, and W.-J. van Hoeve, "Hybrid optimization methods for time-dependent sequencing problems," *European Journal of Operational Research*, vol. 259, no. 3, pp. 887 – 897, 2017.

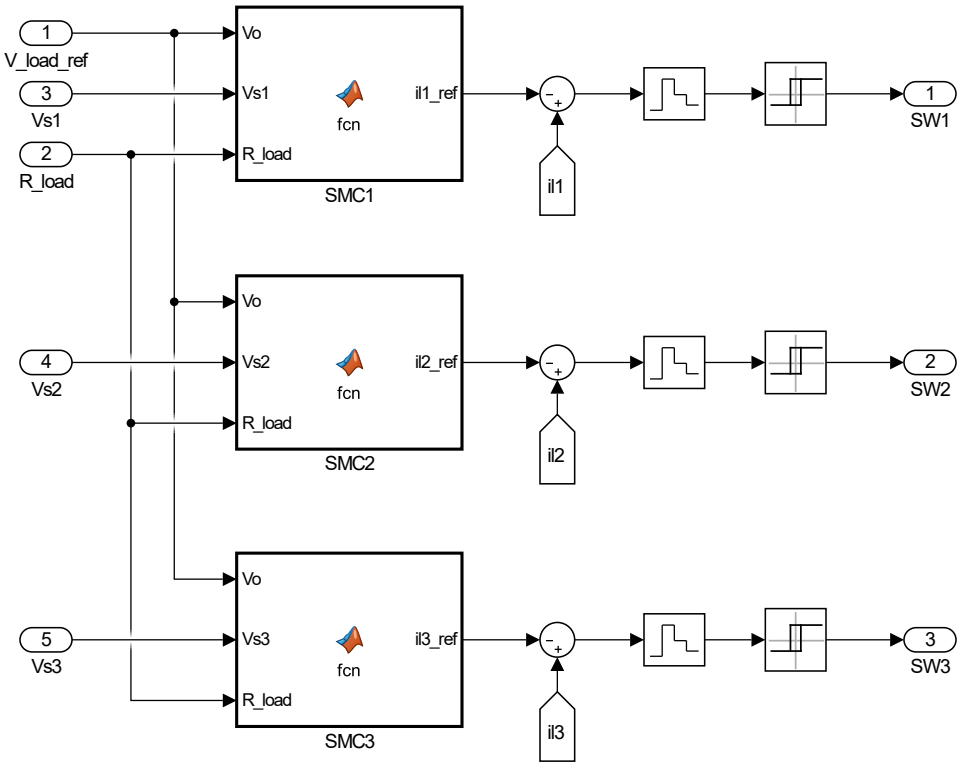
[26] L. A. Kamas and S. R. Sanders, "Parameter and state estimation in power electronic circuits," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 40, pp. 920–928, Dec 1993.

[27] G. E. Pitel and P. T. Krein, "Real-time system identification for load monitoring and transient handling of dc-dc supplies," in *IEEE Power Electronics Specialists Conference*, pp. 3807–3813, June 2008.

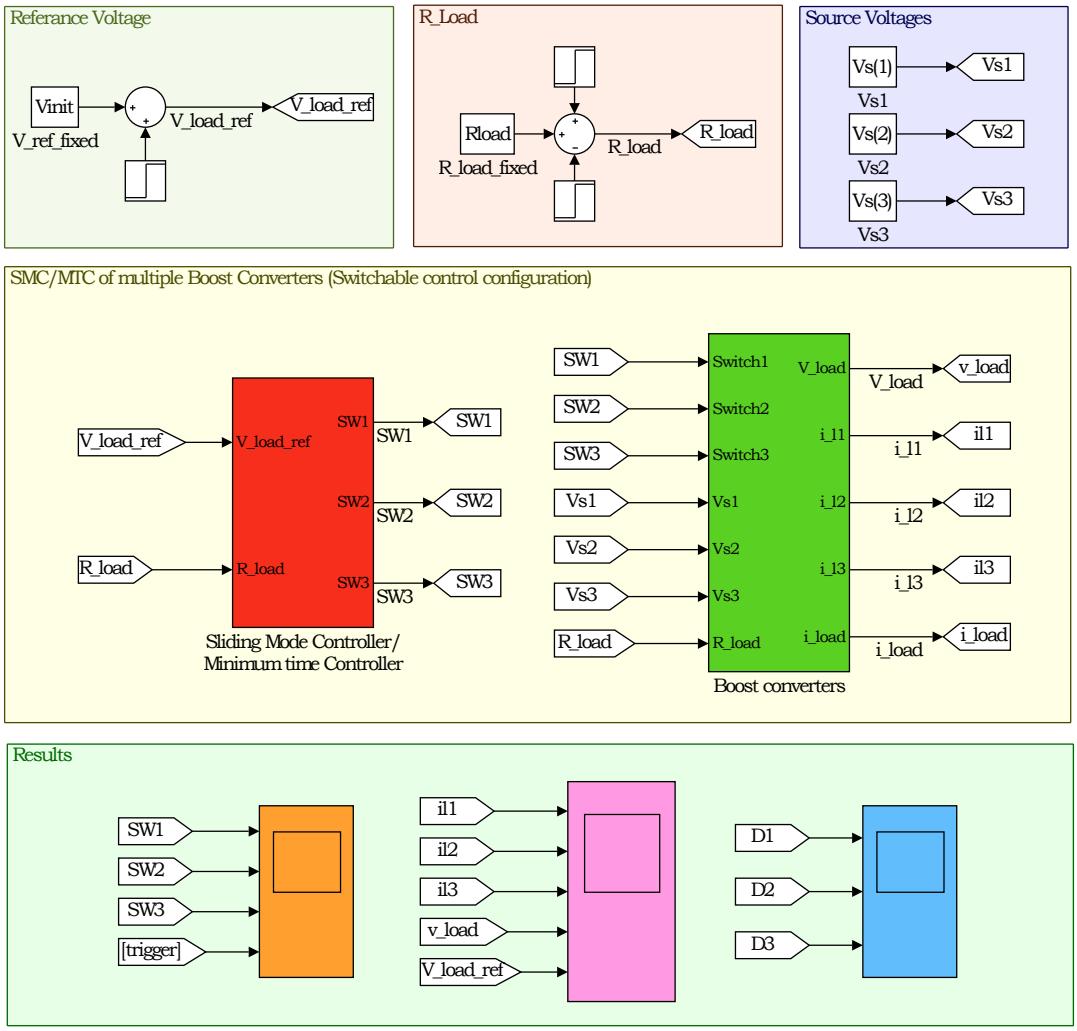
A Appendices

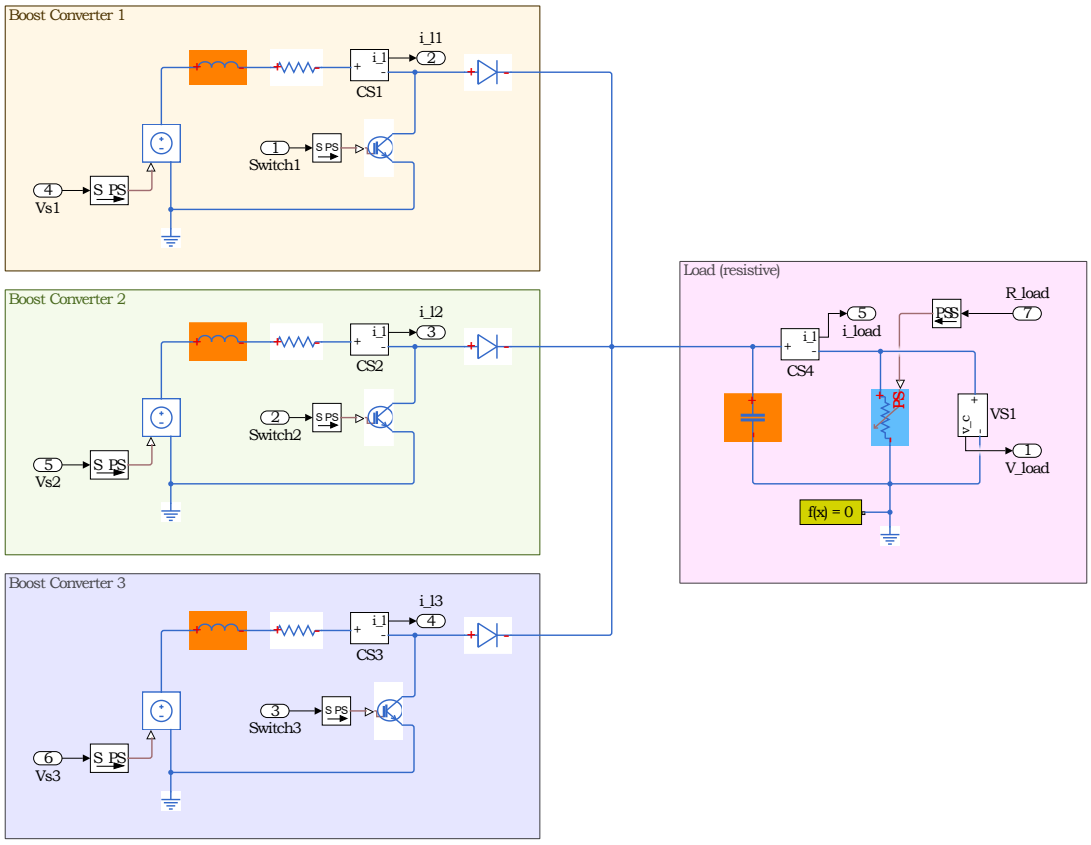
A.1 Simulink Model for SMC of Multiple boost converters

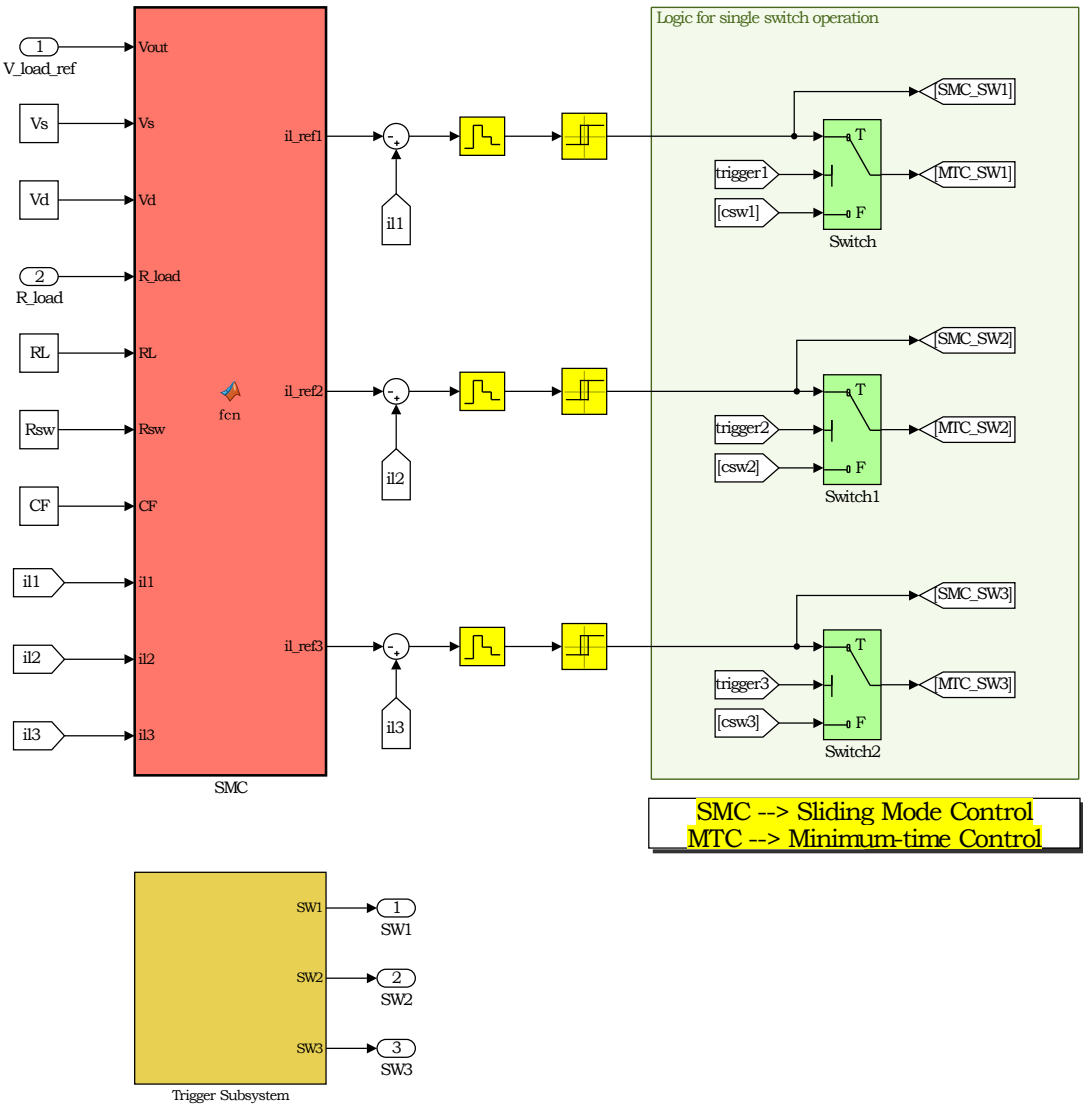



1
V_load_ref
3
Vs1
2
R_load
Vo
Vs1
R_load
fcn
il1_ref
SMC1
il1
1
SW1
4
Vs2
Vo
Vs2
R_load
fcn
il2_ref
SMC2
il2
2
SW2
5
Vs3
Vo
Vs3
R_load
fcn
il3_ref
SMC3
il3
3
SW3

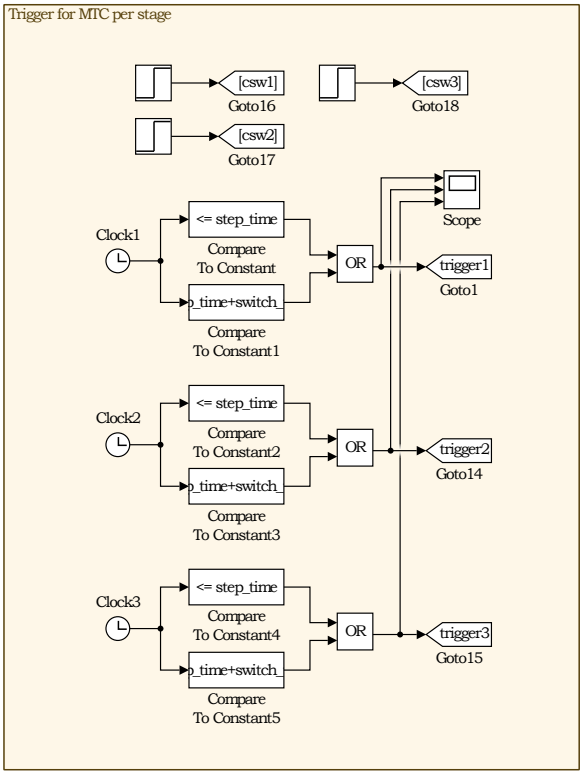
A.2 Simulink Model for MTC of Multiple boost converters

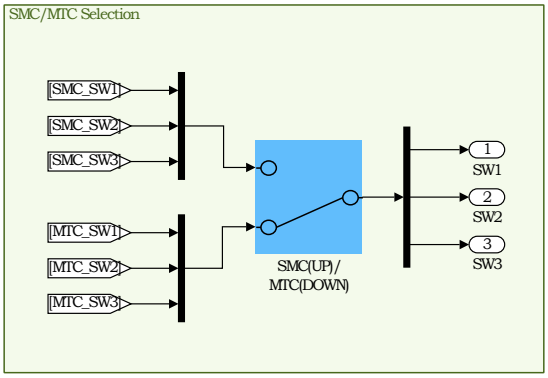


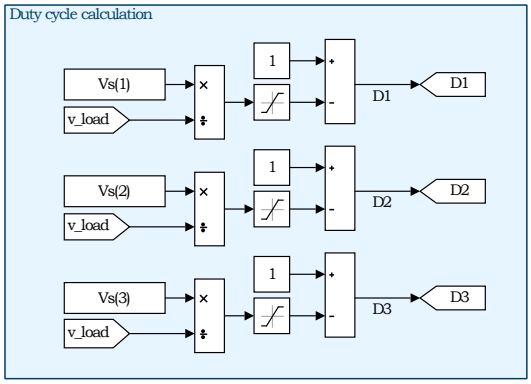

Boost Converter 1
i_l1
CS1
Switch1
Vs1
Boost Converter 2
i_l2
CS2
Switch2
Vs2
Boost Converter 3
i_l3
CS3
Switch3
Vs3
Load (resistive)
R_load
i_load
CS4
VS1
V_load
f[x] = 0


1
V_load_ref
Vs
Vd
2
R_load
RL
Rsw
CF
il1
il2
il3
Vout
Vs
Vd
R_load
RL
Rsw
CF
il1
il2
il3
fcn
SMC
il_ref1
il_ref2
il_ref3
il1
il2
il3
Logic for single switch operation
SMC_SW1
trigger1
[csw1]
T
F
MTC_SW1
Switch
SMC_SW2
trigger2
[csw2]
T
F
MTC_SW2
Switch1
SMC_SW3
trigger3
[csw3]
T
F
MTC_SW3
Switch2
SMC --> Sliding Mode Control
MTC --> Minimum-time Control

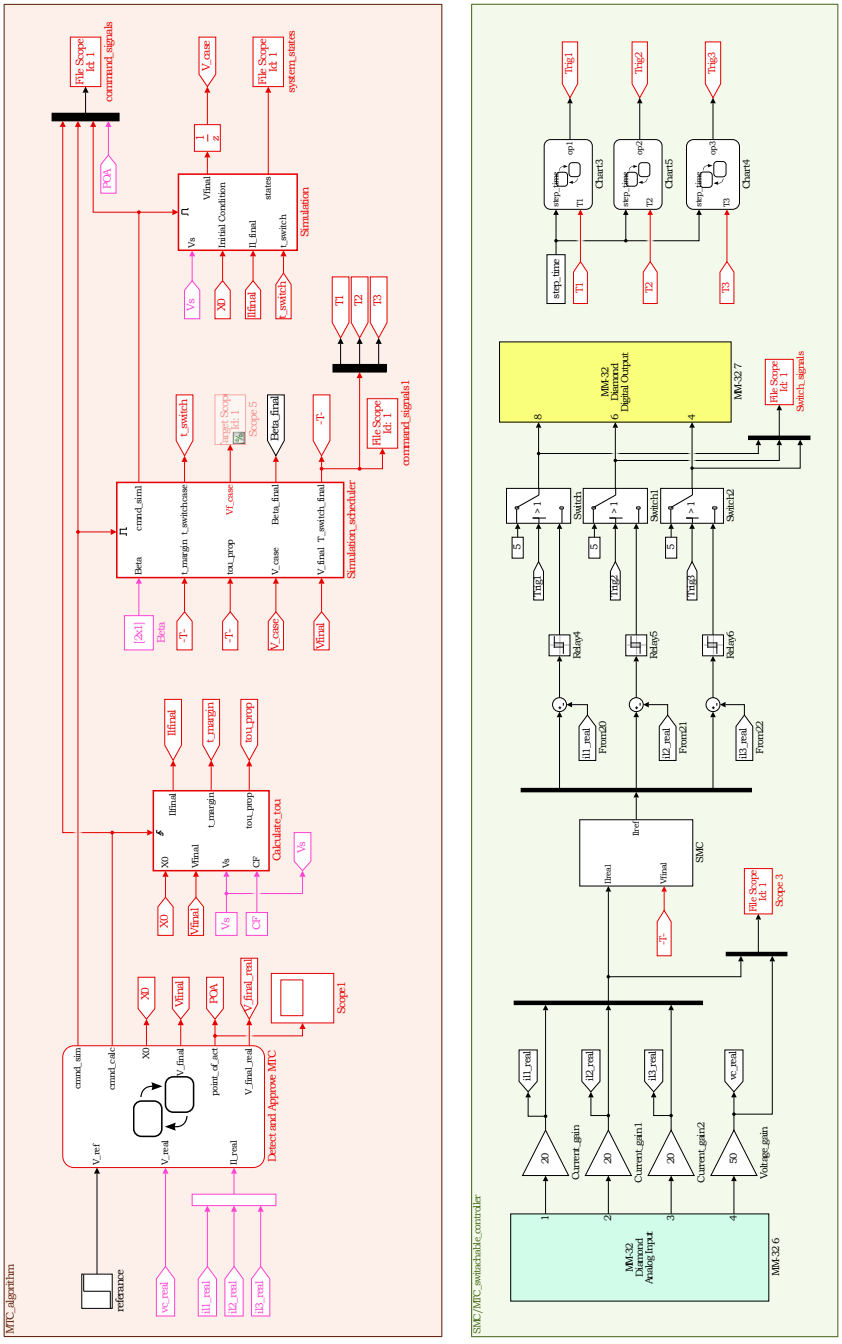
SW1
SW2
SW3
1
SW1
2
SW2
3
SW3
Trigger Subsystem

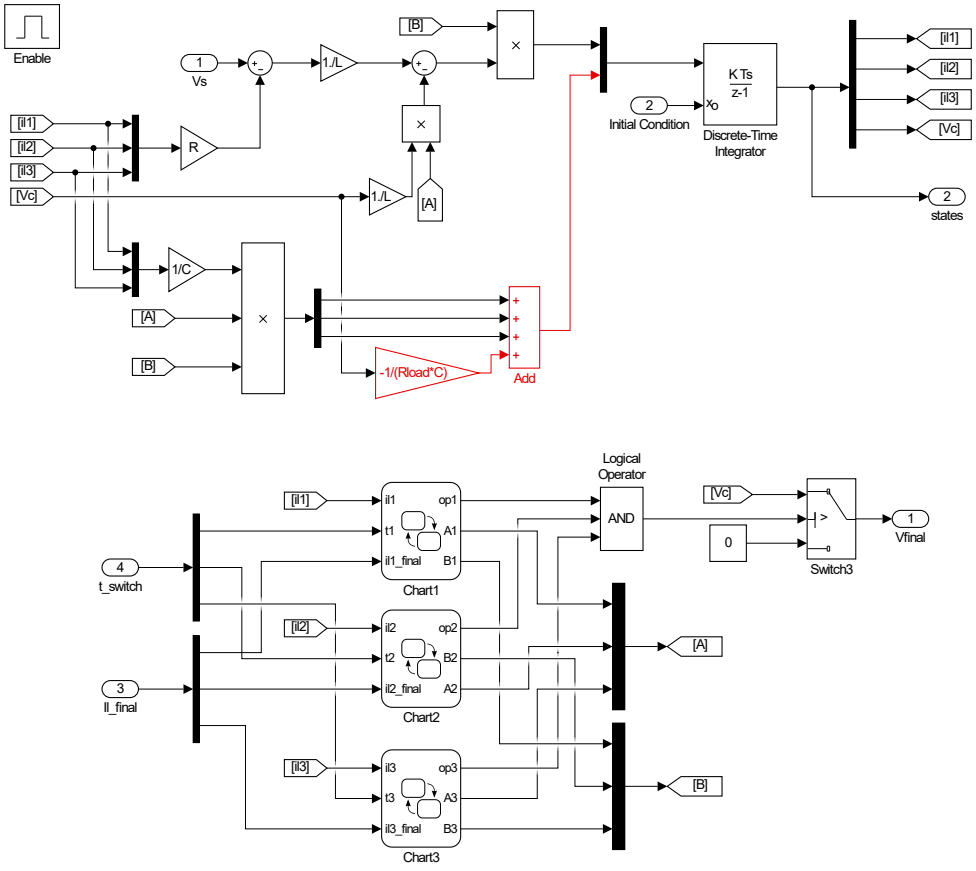

Trigger for MTC per stage
[csw1]
Goto16
[csw3]
Goto18
[csw2]
Goto17
Scope
Clock1
<= step_time
Compare
To Constant
OR
trigger1
Goto1
p_time+switch_
Compare
To Constant1
Clock2
<= step_time
Compare
To Constant2
OR
trigger2
Goto14
p_time+switch_
Compare
To Constant3
Clock3
<= step_time
Compare
To Constant4
OR
trigger3
Goto15
p_time+switch_
Compare
To Constant5

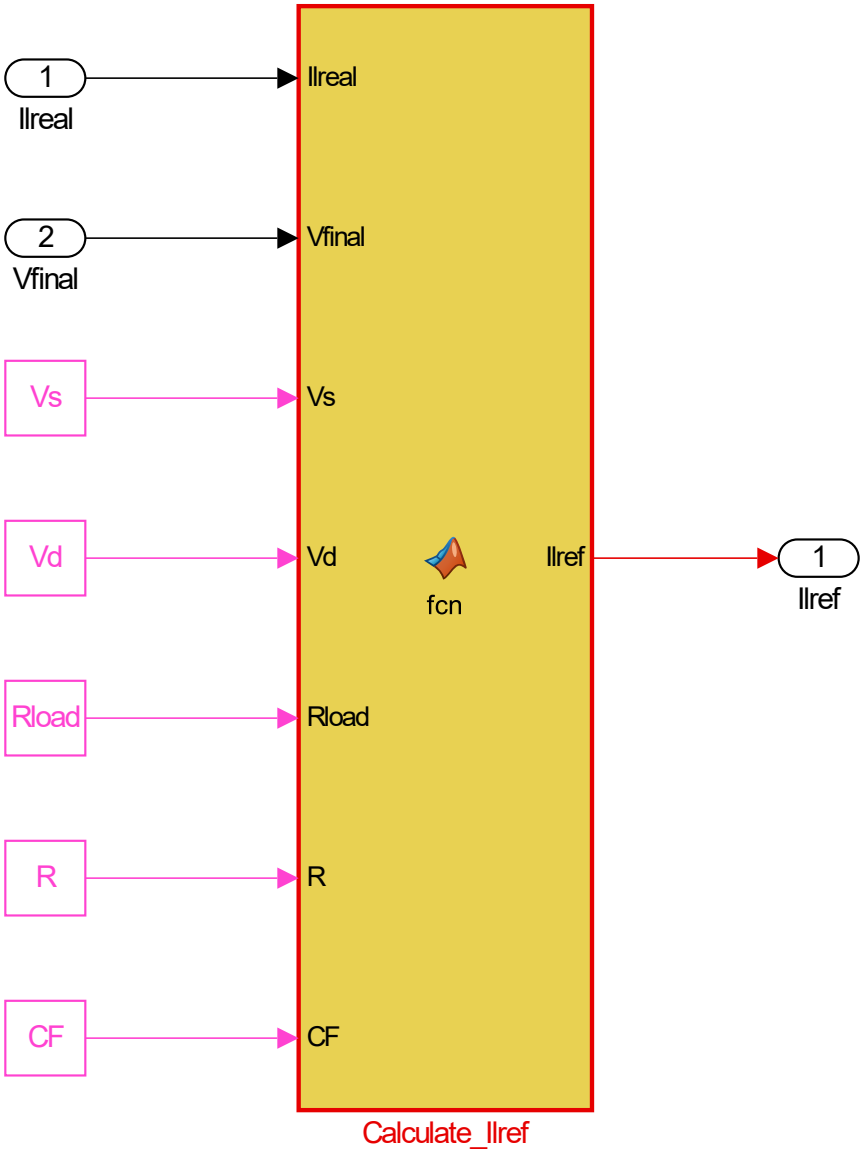

SMC/MTC Selection
[SMC_SW1]
[SMC_SW2]
[SMC_SW3]
[MTC_SW1]
[MTC_SW2]
[MTC_SW3]
SMC(UP)/
MTC(DOWN)
1
SW1
2
SW2
3
SW3

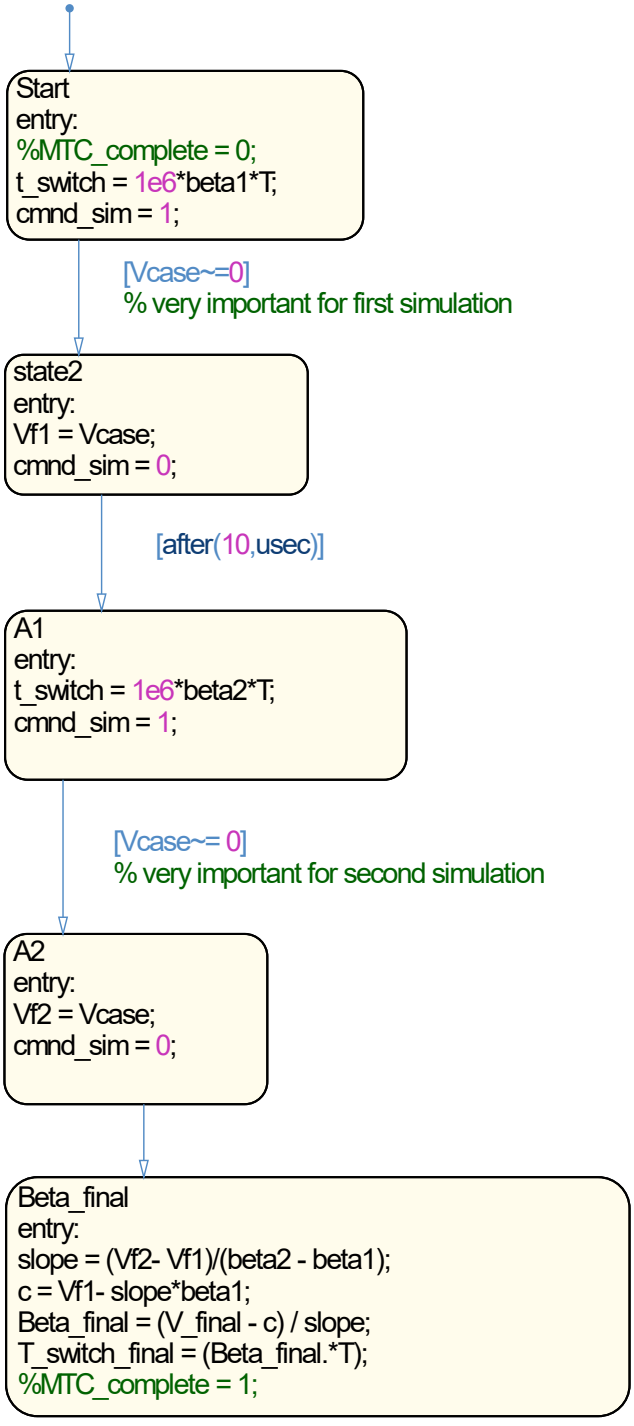

Duty cycle calculation
Vs(1)
v_load
1
D1
Vs(2)
v_load
1
D2
Vs(3)
v_load
1
D3

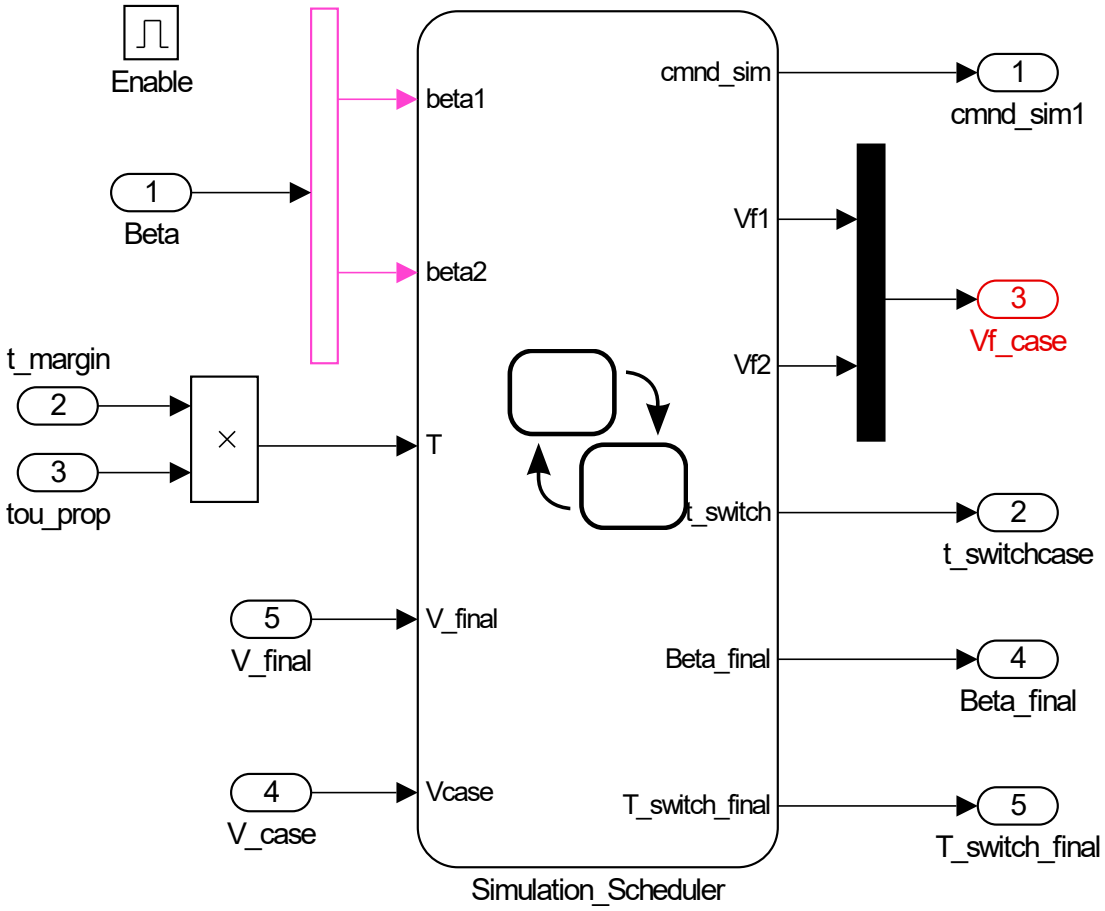
A.3 Simulink Real-time Model for MTC of Multiple Boost Converters

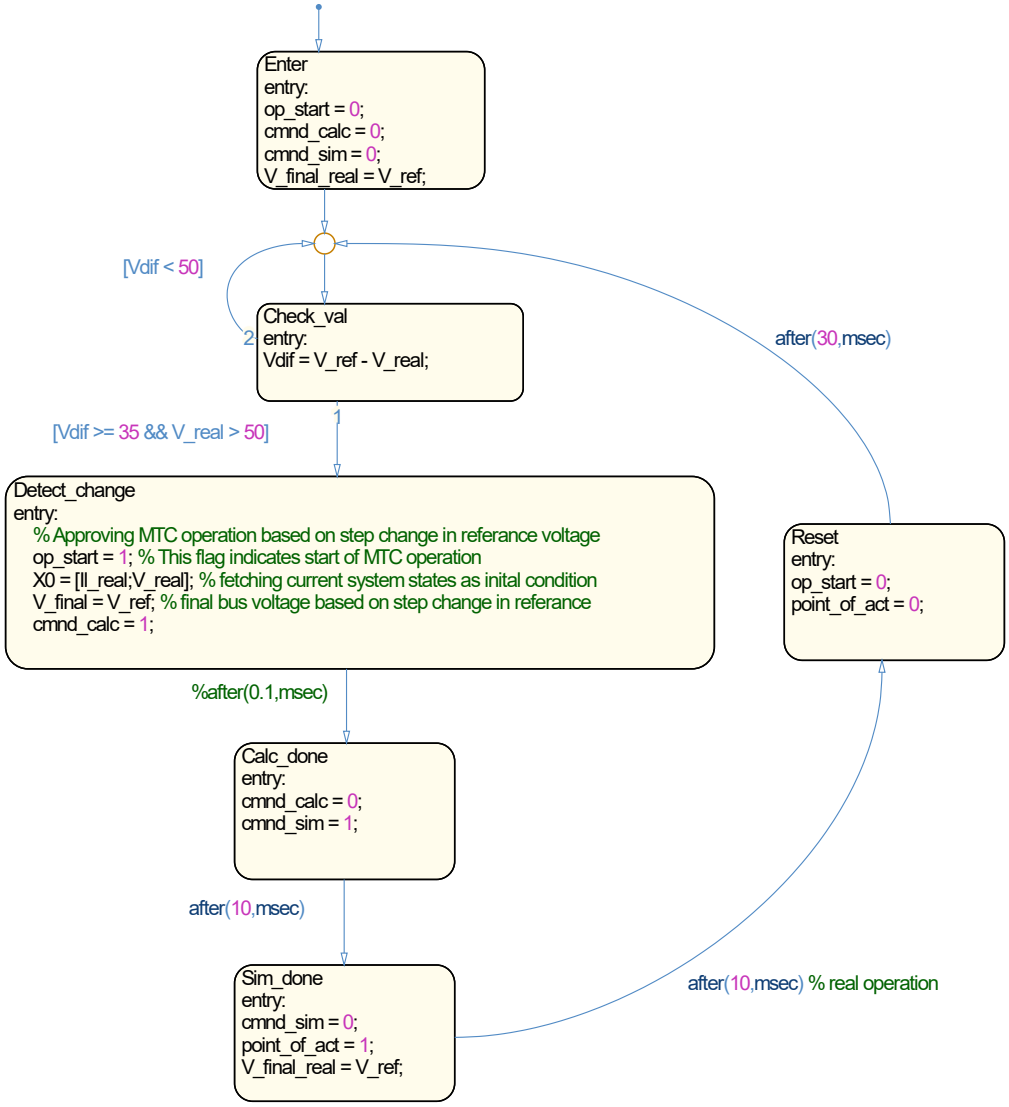



Enable
1
Vs
1./L
[B]
×
[il1]
[il2]
[il3]
[Vc]
R
×
1./L
[A]
2
Initial Condition
K Ts
z-1
Discrete-Time
Integrator
[il1]
[il2]
[il3]
[Vc]
2
states
1/C
[A]
[B]
×
+
+
+
+
-1/(Rload*C)
Add
Logical
Operator
AND
[Vc]
0
Switch3
1
Vfinal
[il1]
il1
t1
il1_final
op1
A1
B1
Chart1
4
t_switch
[il2]
il2
t2
il2_final
op2
B2
A2
Chart2
3
Il_final
[il3]
il3
t3
il3_final
op3
A3
B3
Chart3
[A]
[B]


1
Ilreal
Ilreal
2
Vfinal
Vfinal
Vs
Vs
Vd
Vd
Rload
Rload
R
R
CF
CF
fcn
Ilref
1
Ilref
Calculate_Ilref


Start
entry:
%MTC_complete = 0;
t_switch = 1e6*beta1*T;
cmnd_sim = 1;
[Vcase~=0]
% very important for first simulation
state2
entry:
Vf1 = Vcase;
cmnd_sim = 0;
[after(10,usec)]
A1
entry:
t_switch = 1e6*beta2*T;
cmnd_sim = 1;
[Vcase~= 0]
% very important for second simulation
A2
entry:
Vf2 = Vcase;
cmnd_sim = 0;
Beta_final
entry:
slope = (Vf2- Vf1)/(beta2 - beta1);
c = Vf1- slope*beta1;
Beta_final = (V_final - c) / slope;
T_switch_final = (Beta_final.*T);
%MTC_complete = 1;


Enable
1
Beta
t_margin
2
3
tou_prop
5
V_final
4
V_case
beta1
beta2
T
V_final
Vcase
cmnd_sim
Vf1
Vf2
t_switch
Beta_final
T_switch_final
Simulation_Scheduler
1
cmnd_sim1
3
Vf_case
2
t_switchcase
4
Beta_final
5
T_switch_final


Enter
entry:
op_start = 0;
cmnd_calc = 0;
cmnd_sim = 0;
V_final_real = V_ref;
[Vdif < 50]
2
Check_val
entry:
Vdif = V_ref - V_real;
after(30,msec)
1
[Vdif >= 35 && V_real > 50]
Detect_change
entry:
% Approving MTC operation based on step change in referance voltage
op_start = 1; % This flag indicates start of MTC operation
X0 = [Il_real;V_real]; % fetching current system states as inital condition
V_final = V_ref; % final bus voltage based on step change in referance
cmnd_calc = 1;
Reset
entry:
op_start = 0;
point_of_act = 0;
%after(0.1,msec)
Calc_done
entry:
cmnd_calc = 0;
cmnd_sim = 1;
after(10,msec)
Sim_done
entry:
cmnd_sim = 0;
point_of_act = 1;
V_final_real = V_ref;
after(10,msec) % real operation

A.4 HIL simulation results for experimental cases

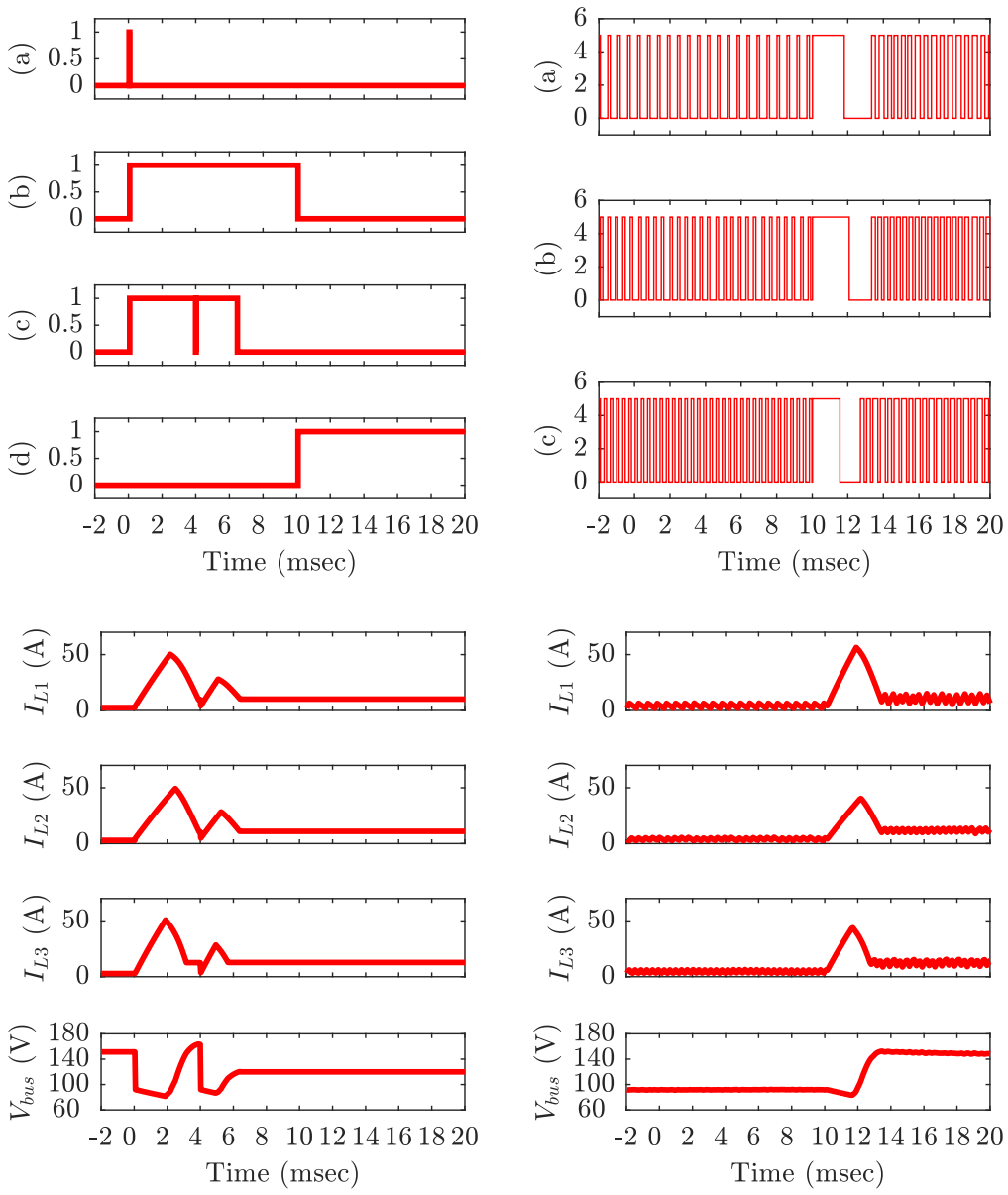


Figure A.1: HIL emulation results for $L_1(-30\%change)$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

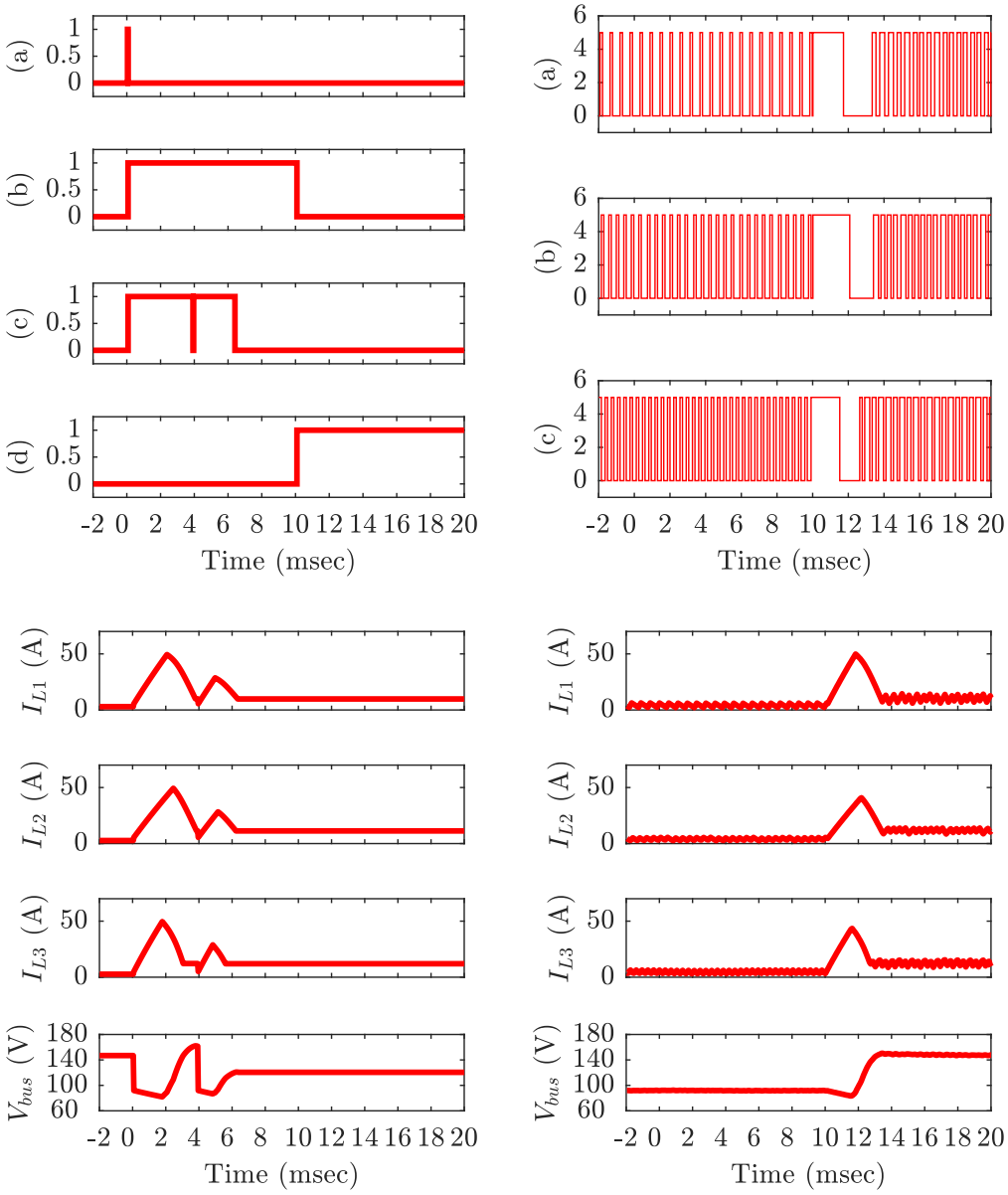


Figure A.2: HIL emulation results for $L_1(-20\%change)$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

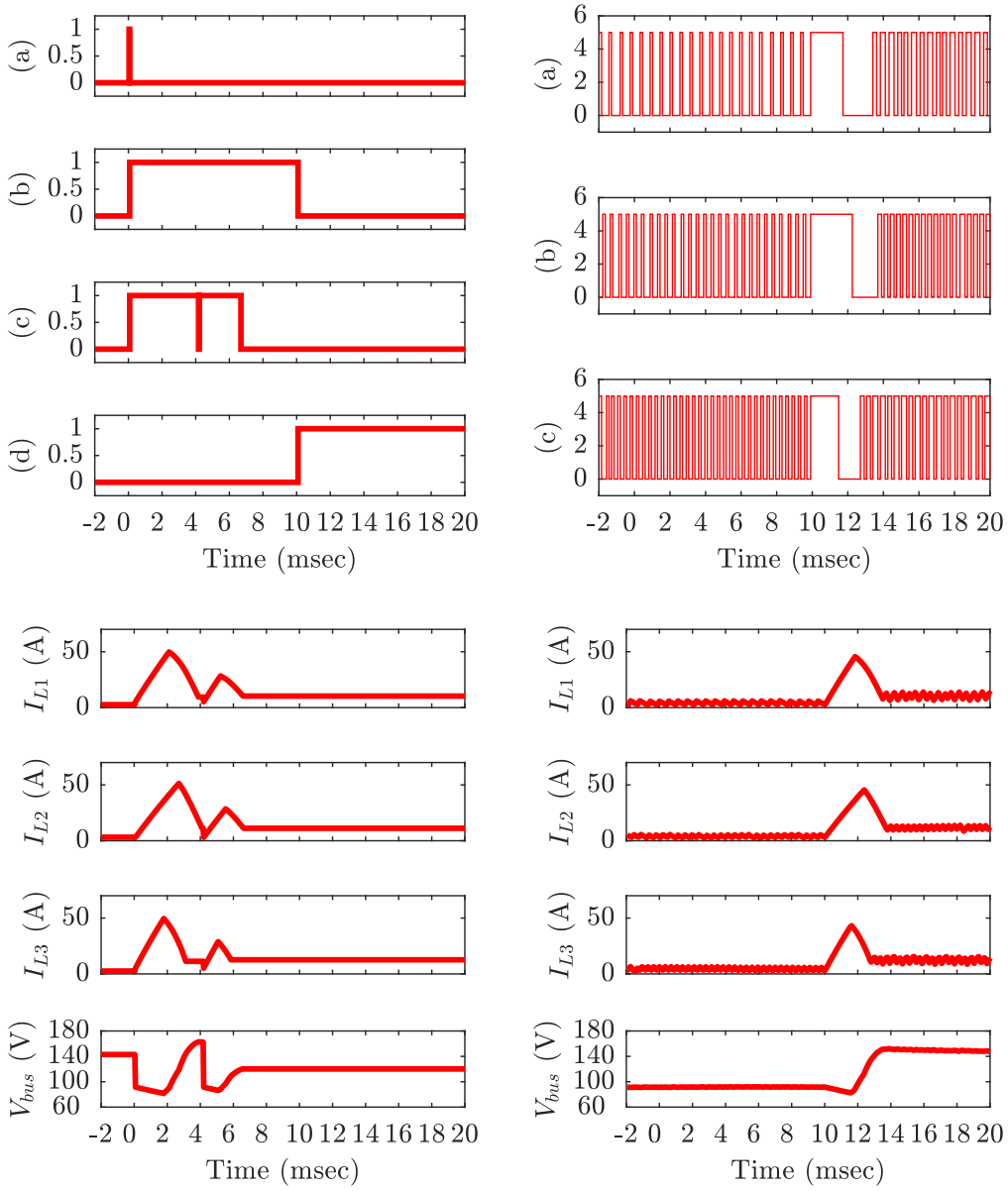


Figure A.3: HIL emulation results for $L_1(-10\%change)$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

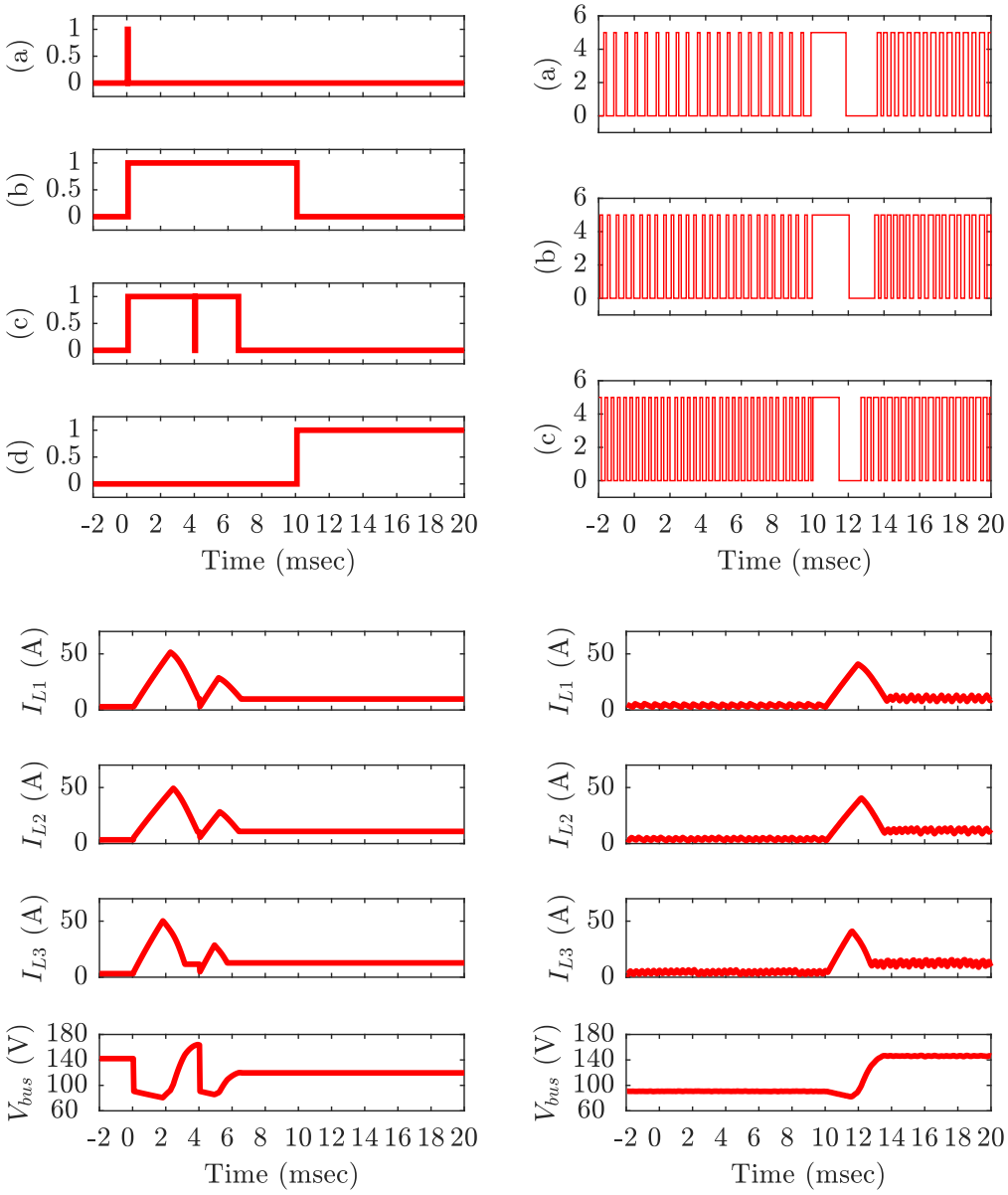


Figure A.4: HIL emulation results for $L_1(+10\%change)$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

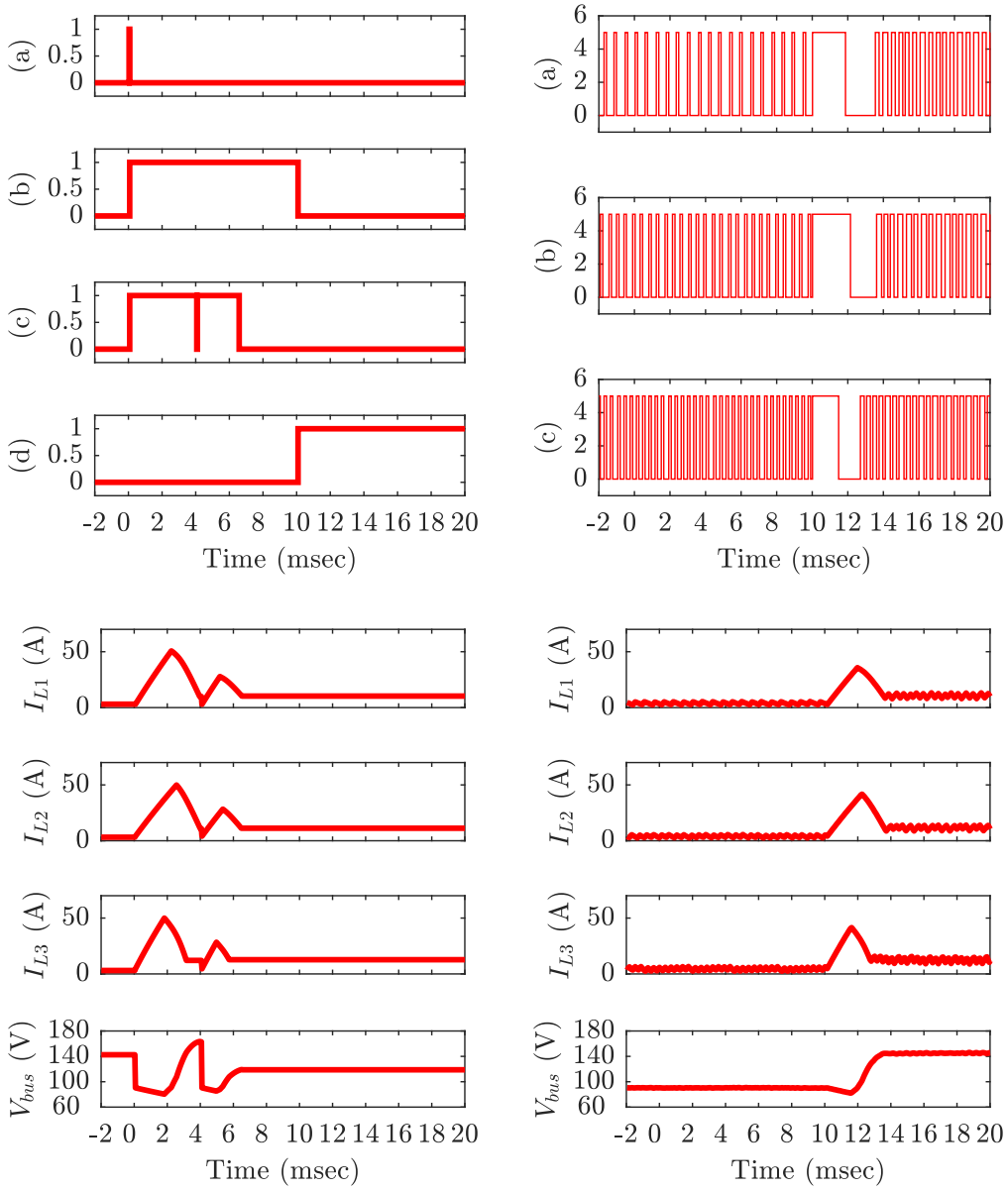


Figure A.5: HIL emulation results for $L_1(+20\%change)$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

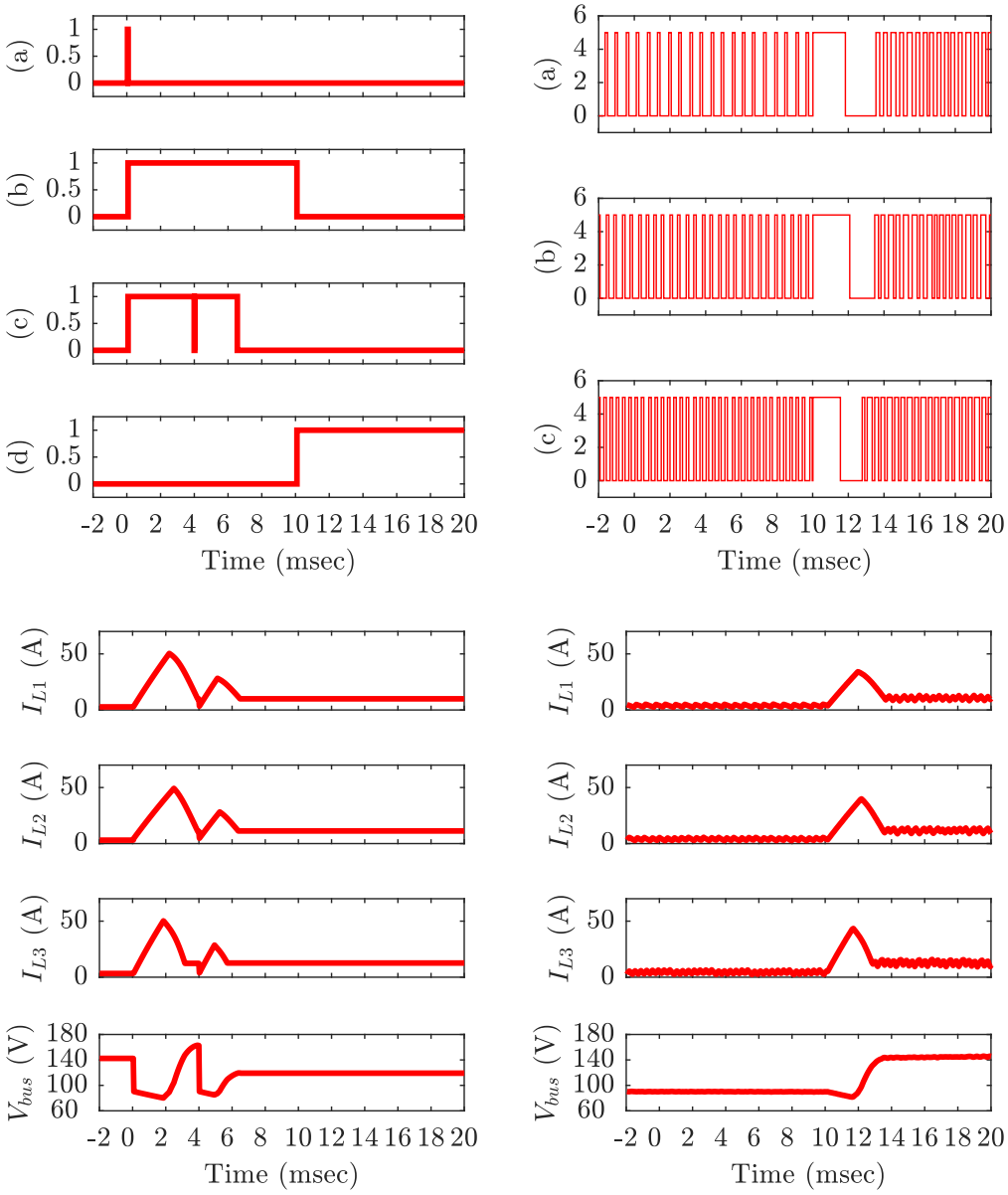


Figure A.6: HIL emulation results for $L_1(+30\%change)$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

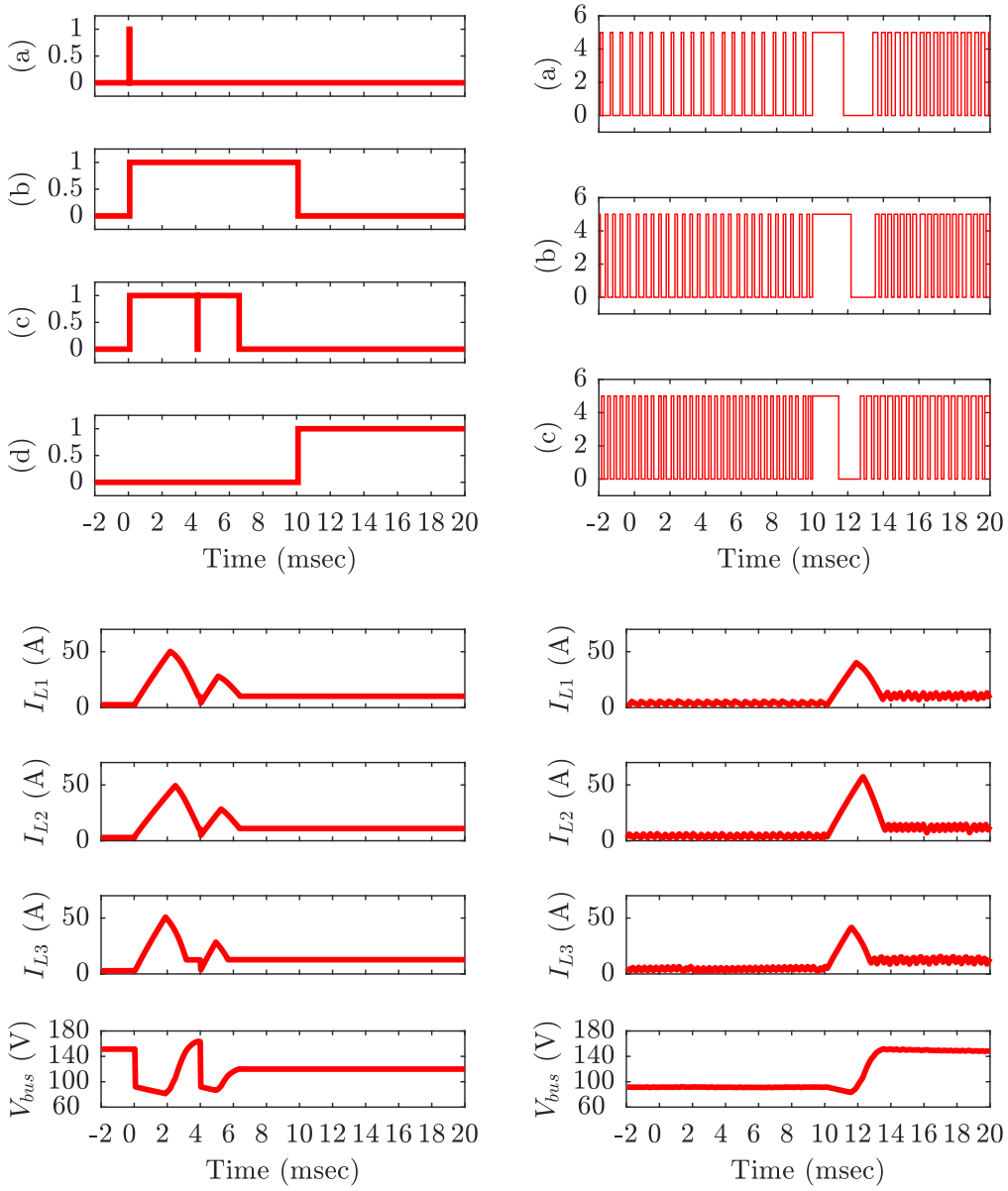


Figure A.7: HIL emulation results for $L_2(-30\%change)$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

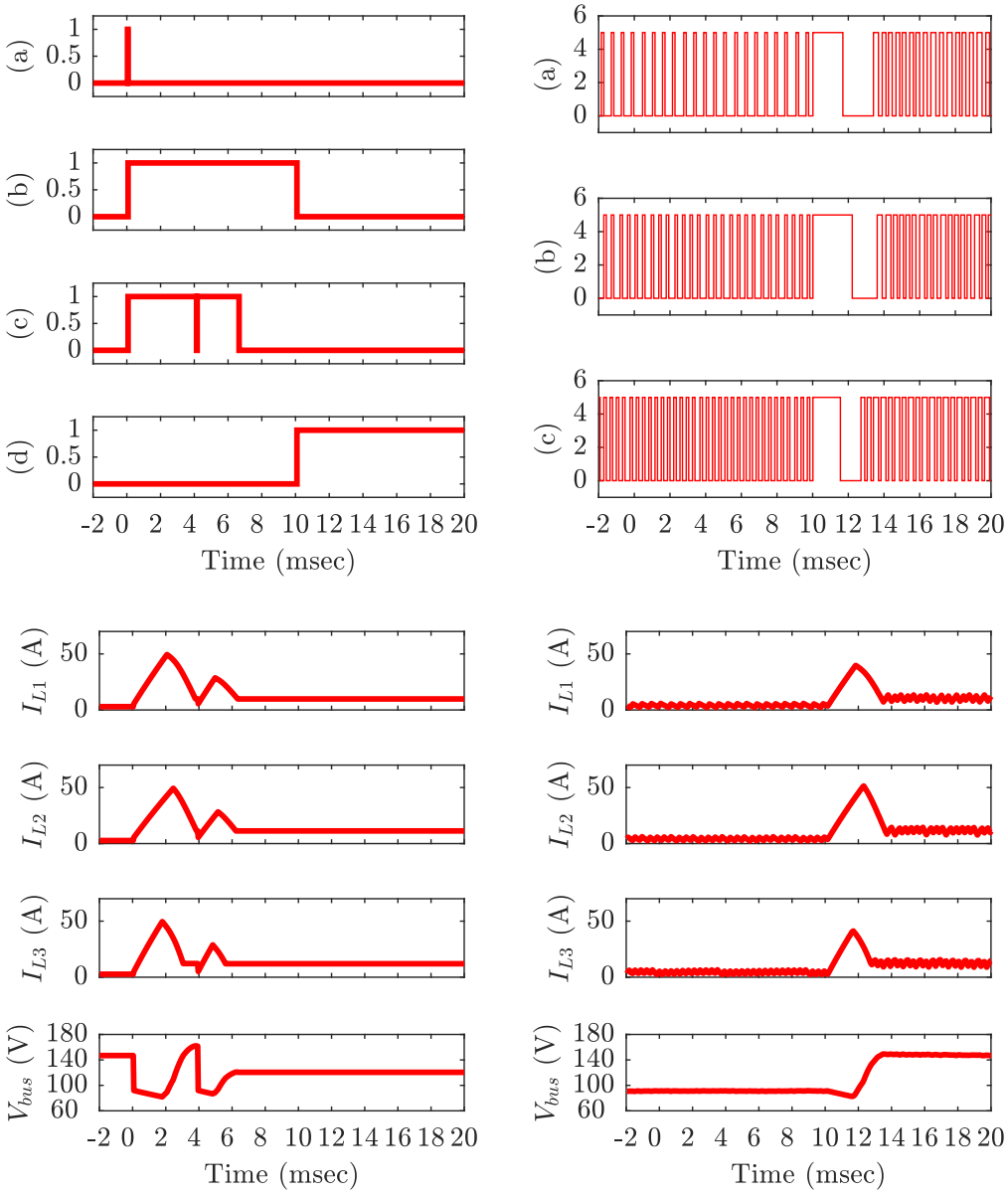


Figure A.8: HIL emulation results for $L_2(-20\%change)$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

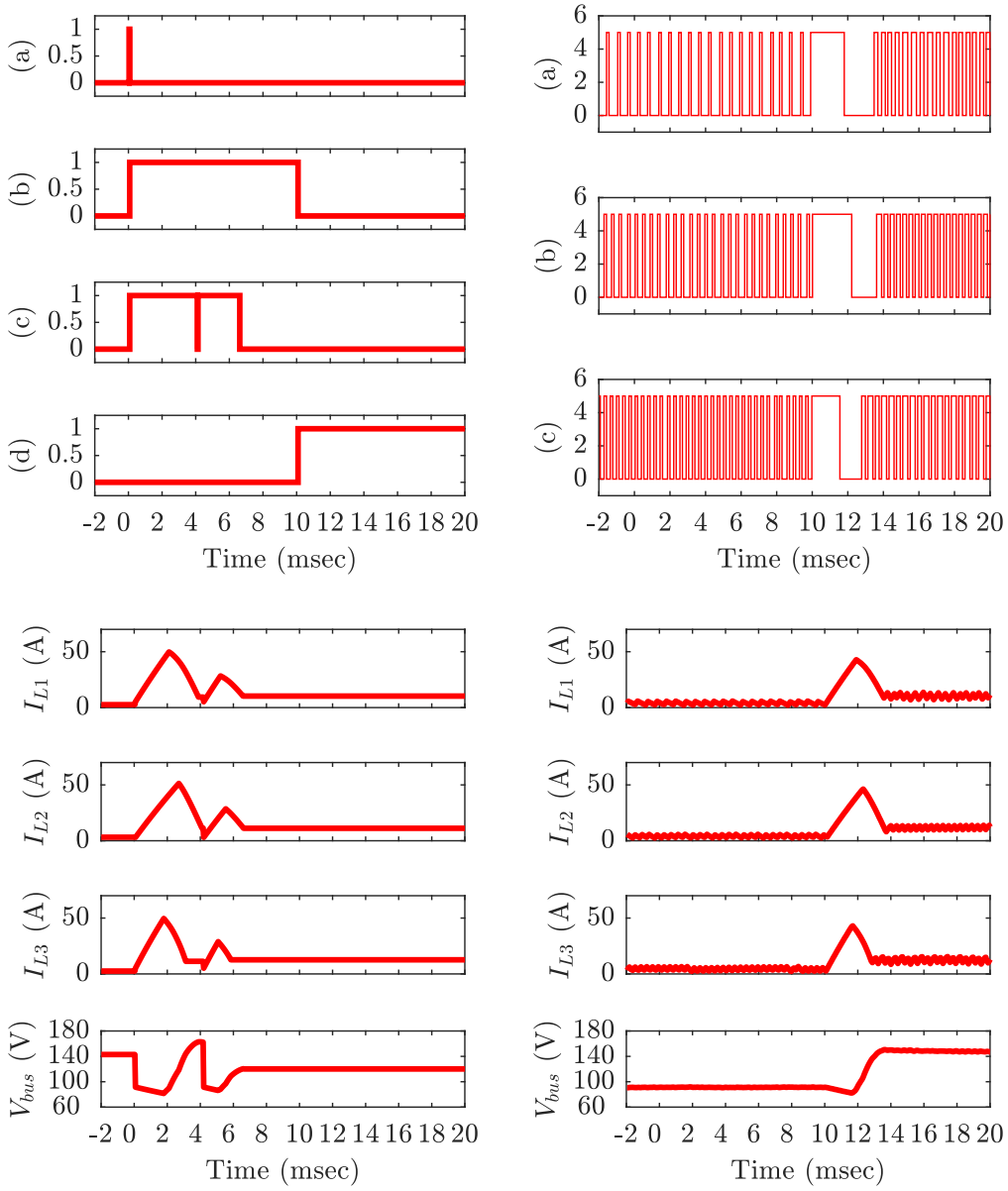


Figure A.9: HIL emulation results for $L_2(-10\%change)$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

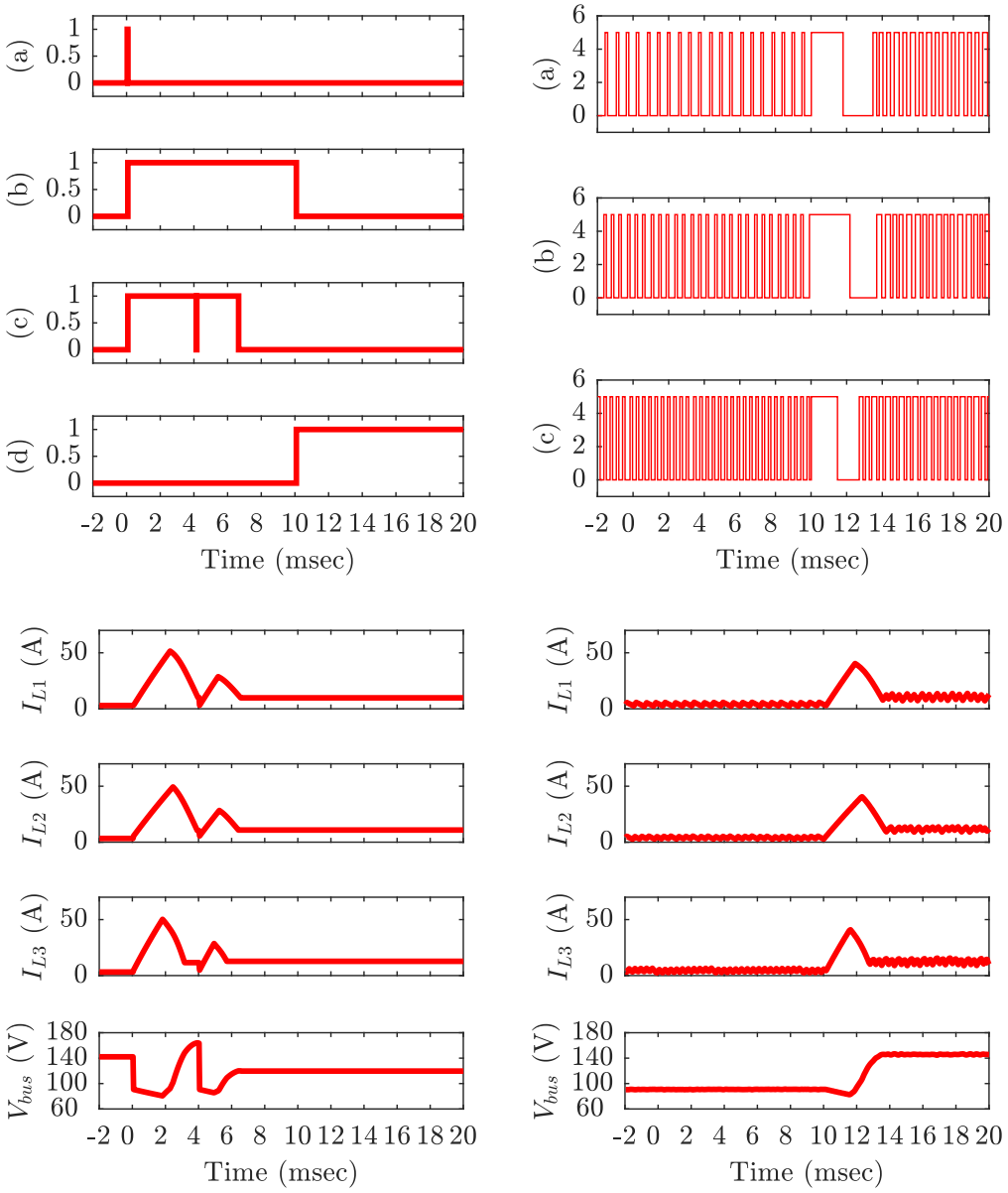


Figure A.10: HIL emulation results for $L_2(+10\%\ change)$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

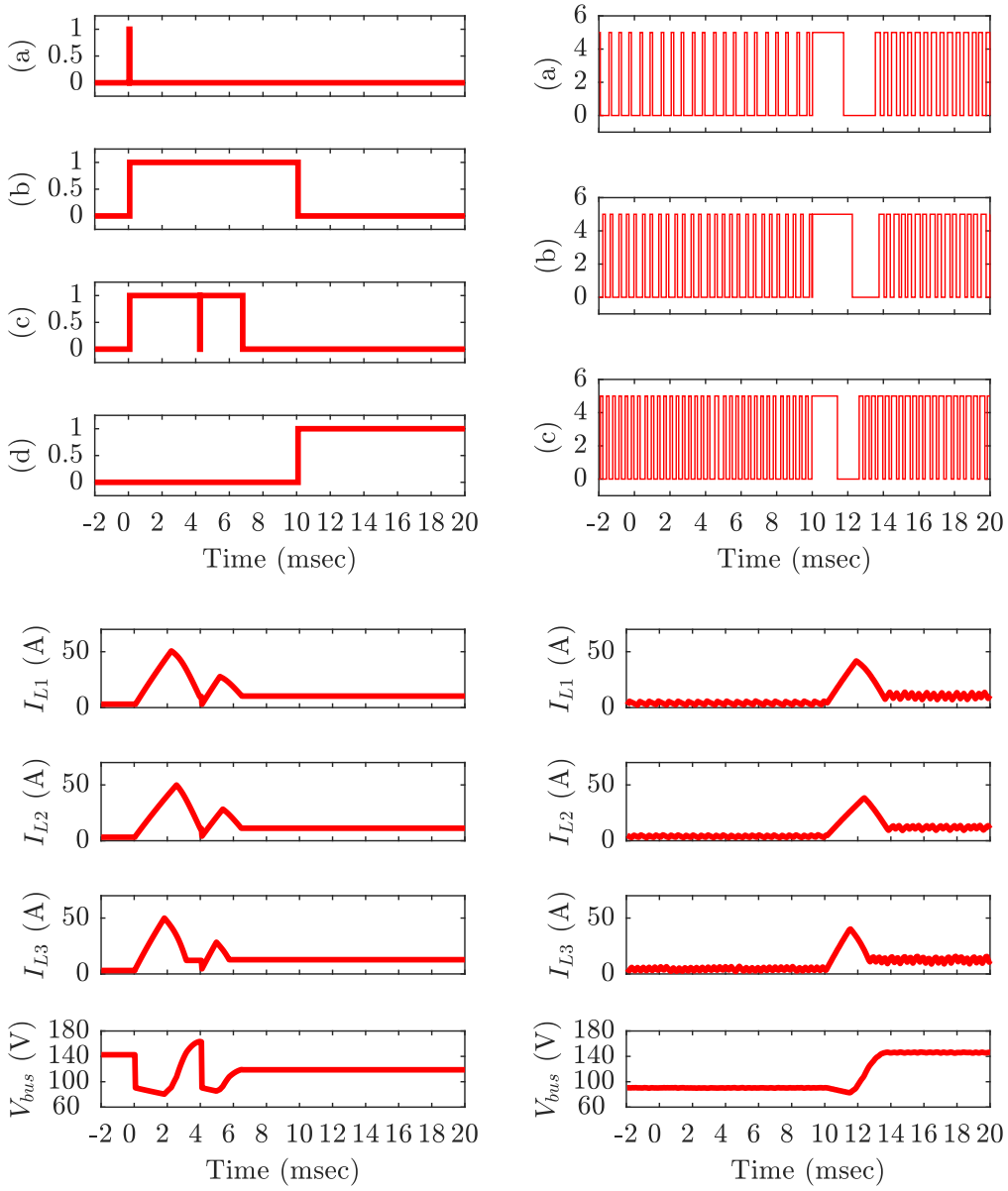


Figure A.11: HIL emulation results for $L_2(+20\%\ change)$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

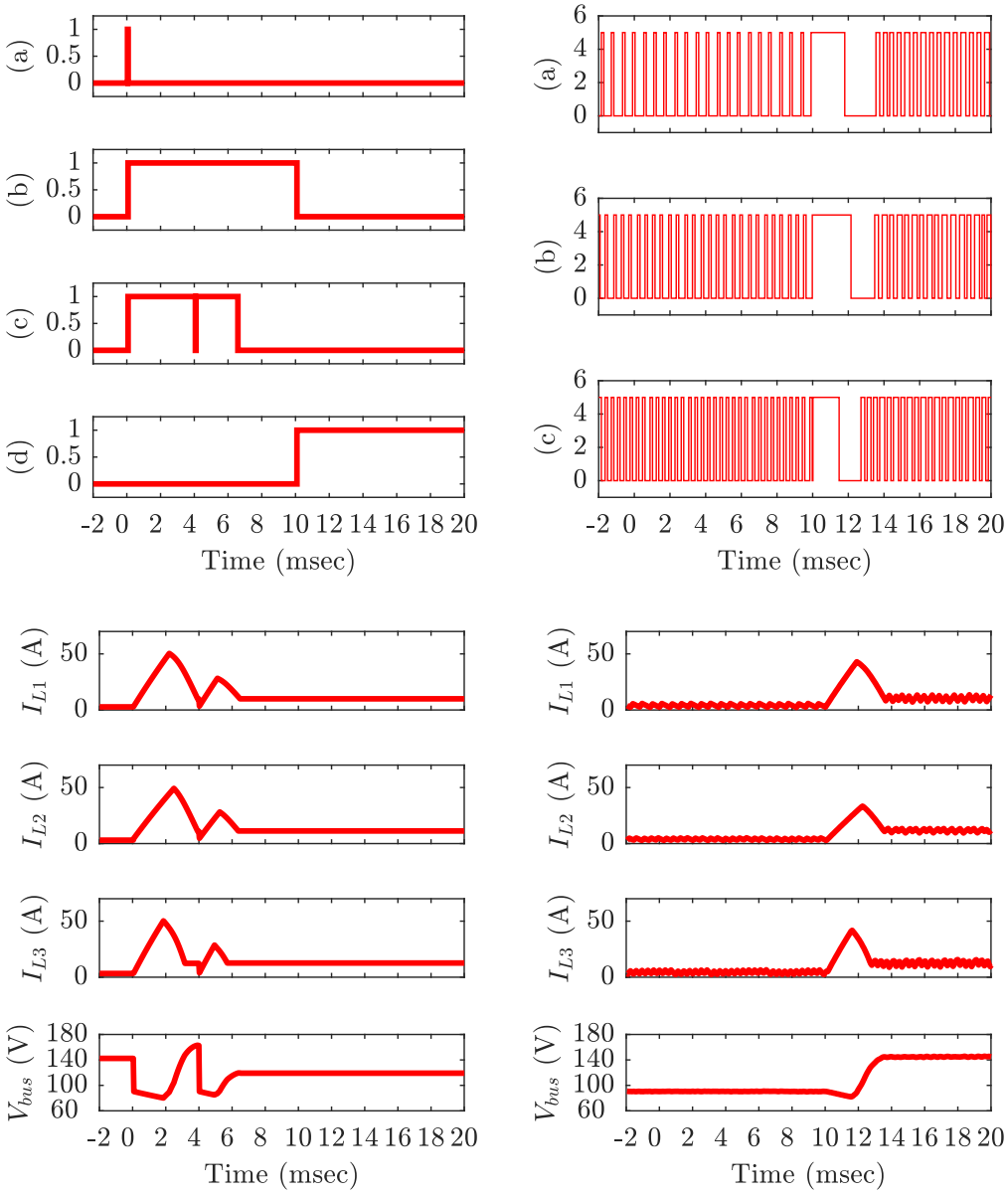


Figure A.12: HIL emulation results for $L_2(+30\%\ change)$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

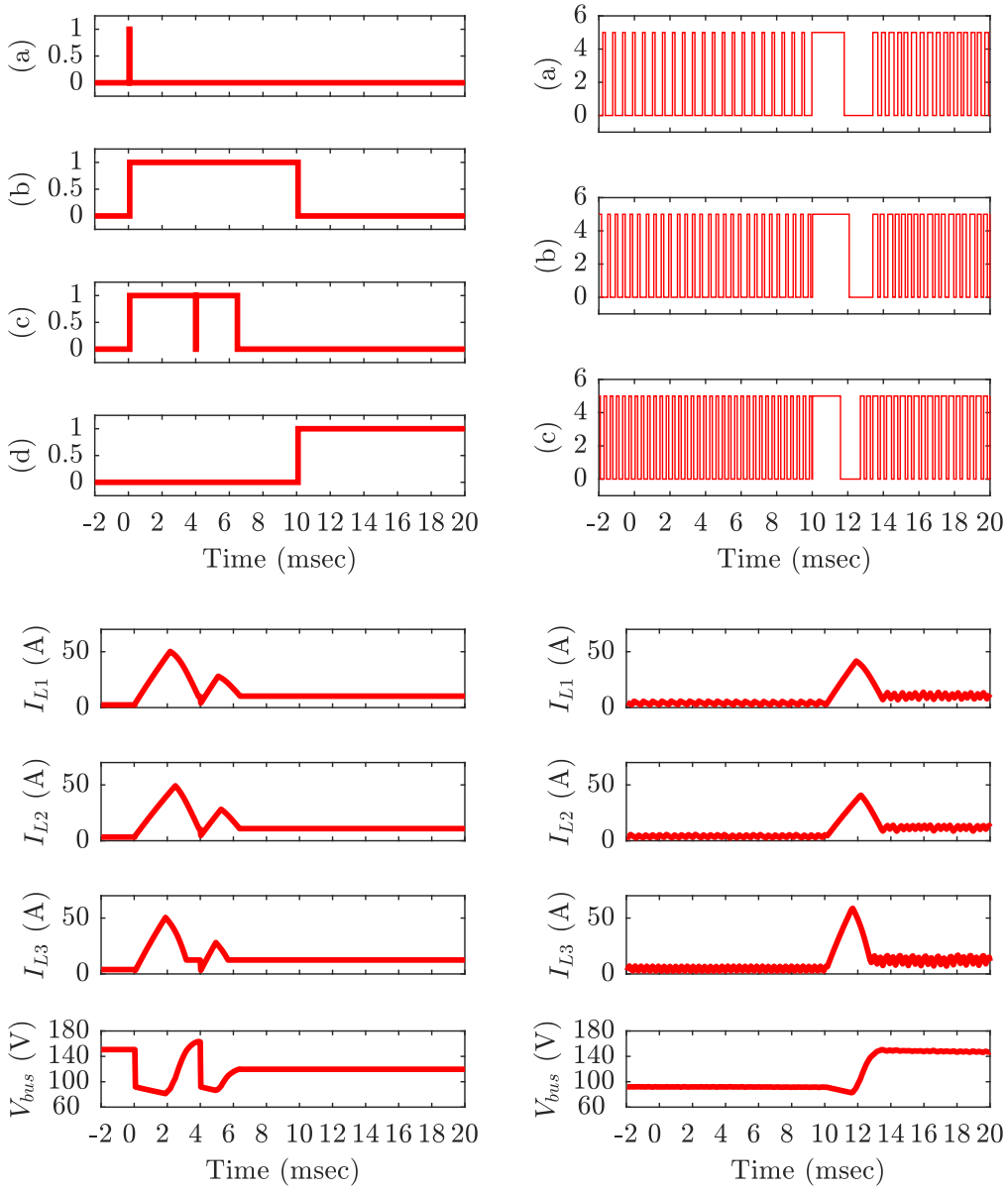


Figure A.13: HIL emulation results for $L_3(-30\%\ change)$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

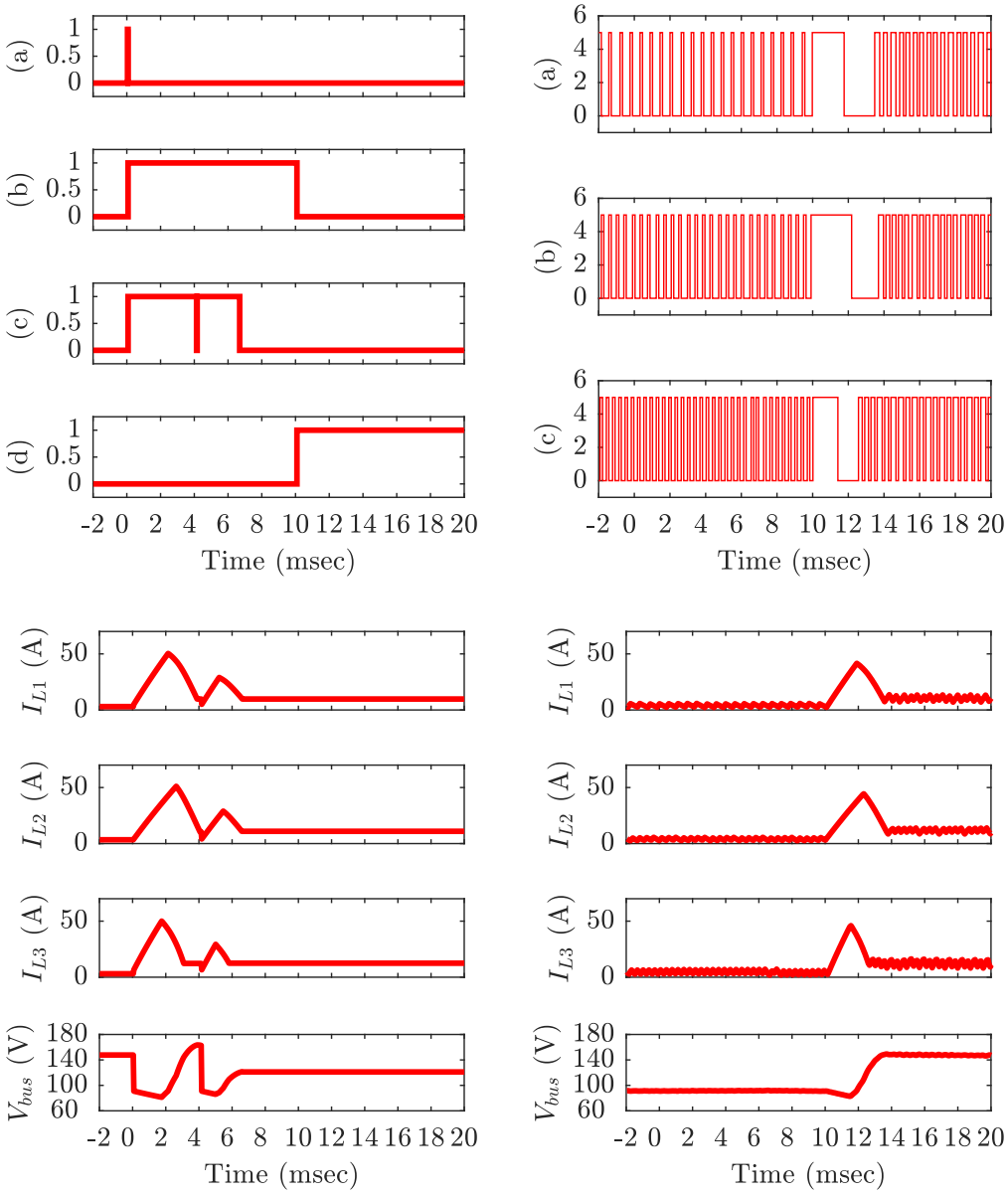


Figure A.14: HIL emulation results for $L_3(-20\%\ change)$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

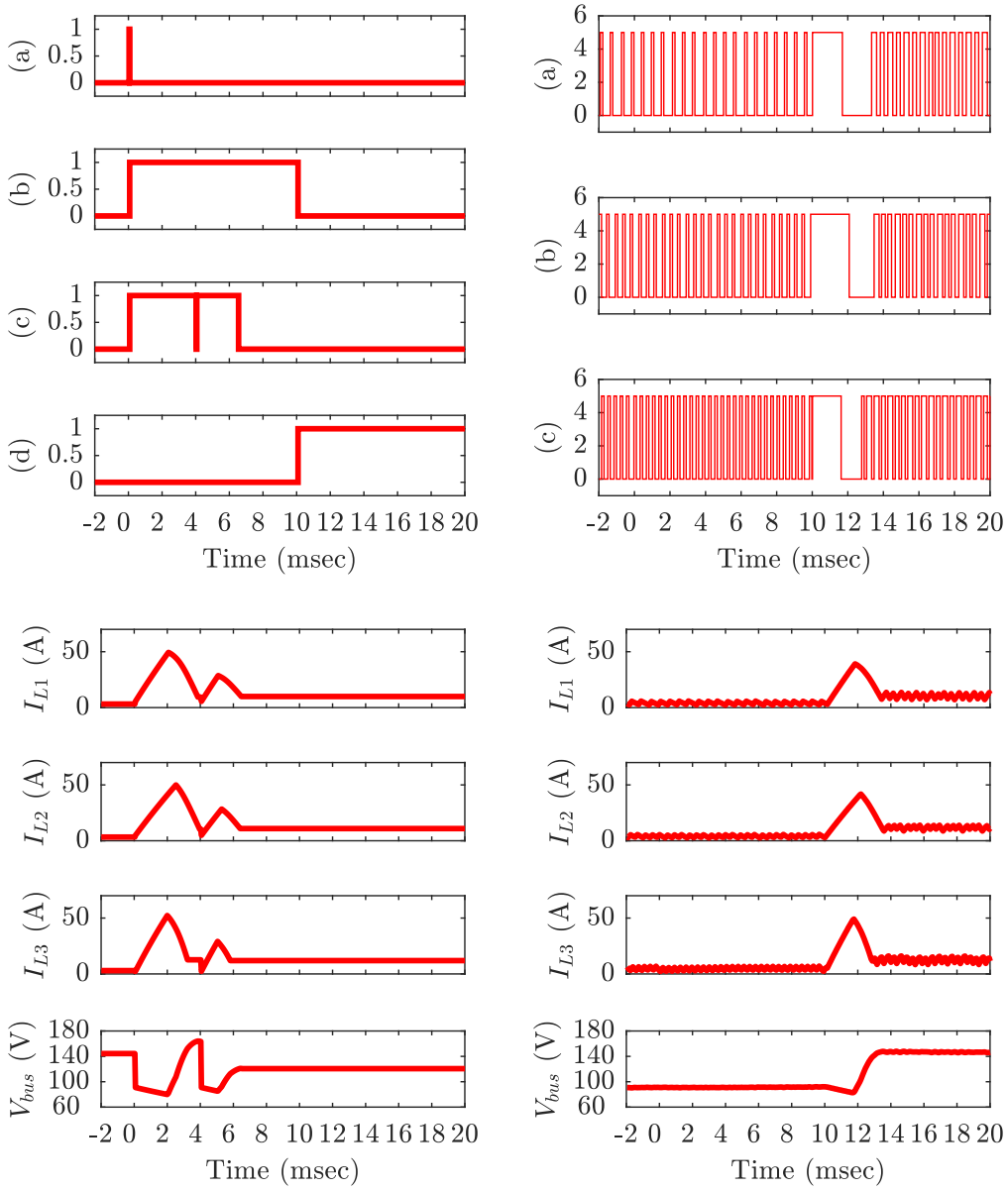


Figure A.15: HIL emulation results for $L_3(-10\%\ change)$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

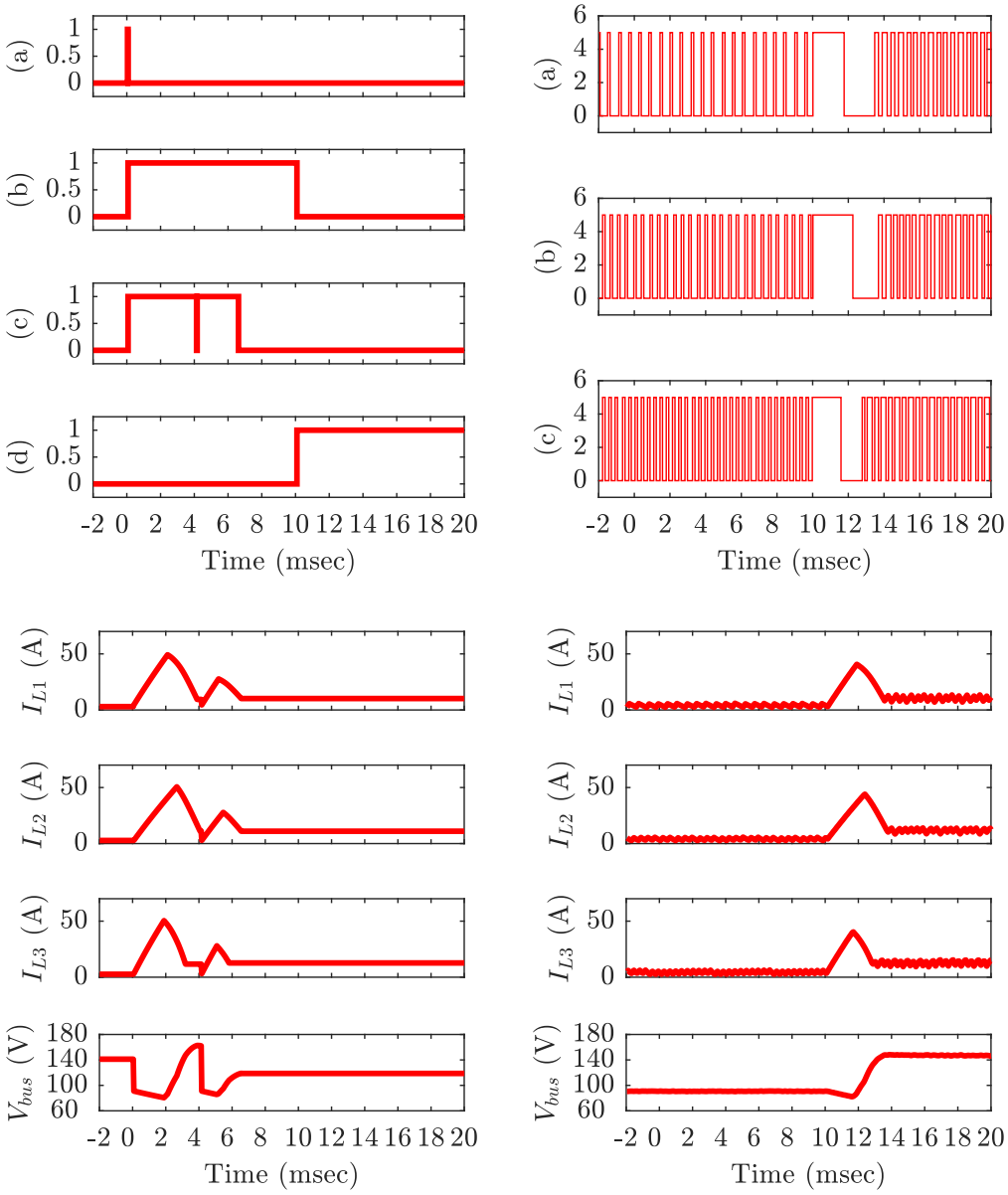


Figure A.16: HIL emulation results for $L_3(+10\%\ change)$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

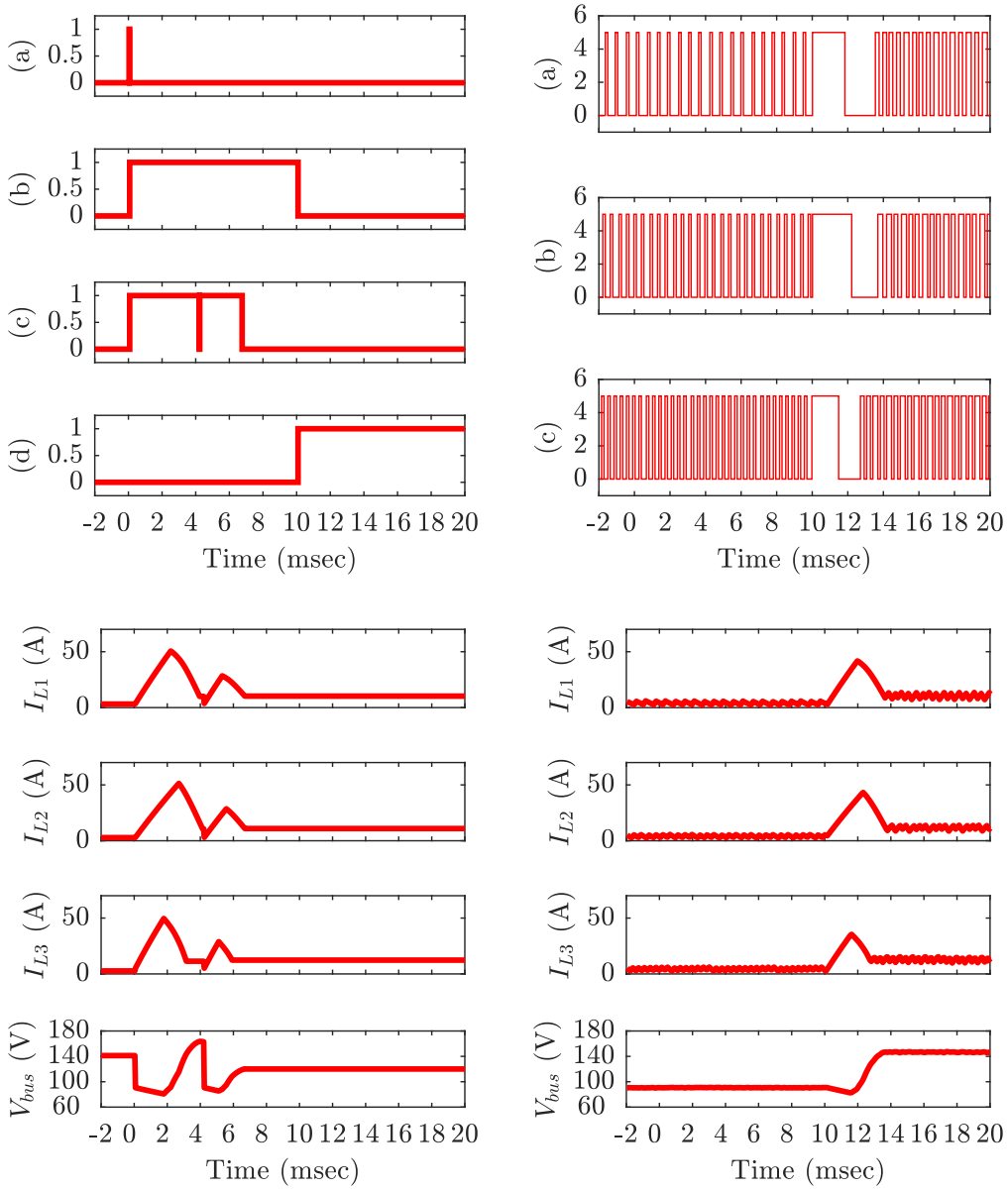


Figure A.17: HIL emulation results for $L_3(+20\%\ change)$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

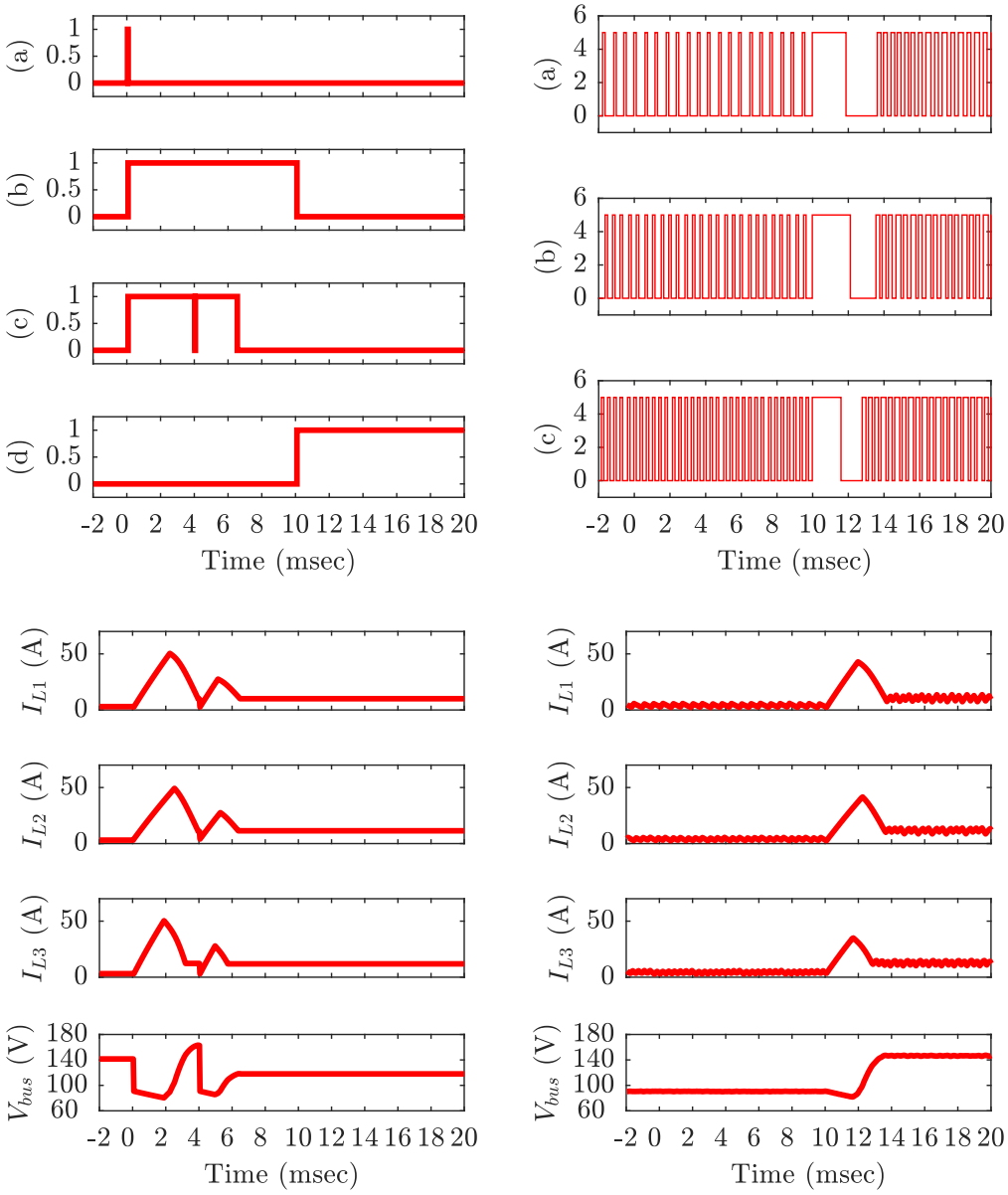


Figure A.18: HIL emulation results for $L_3(+30\%\ change)$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

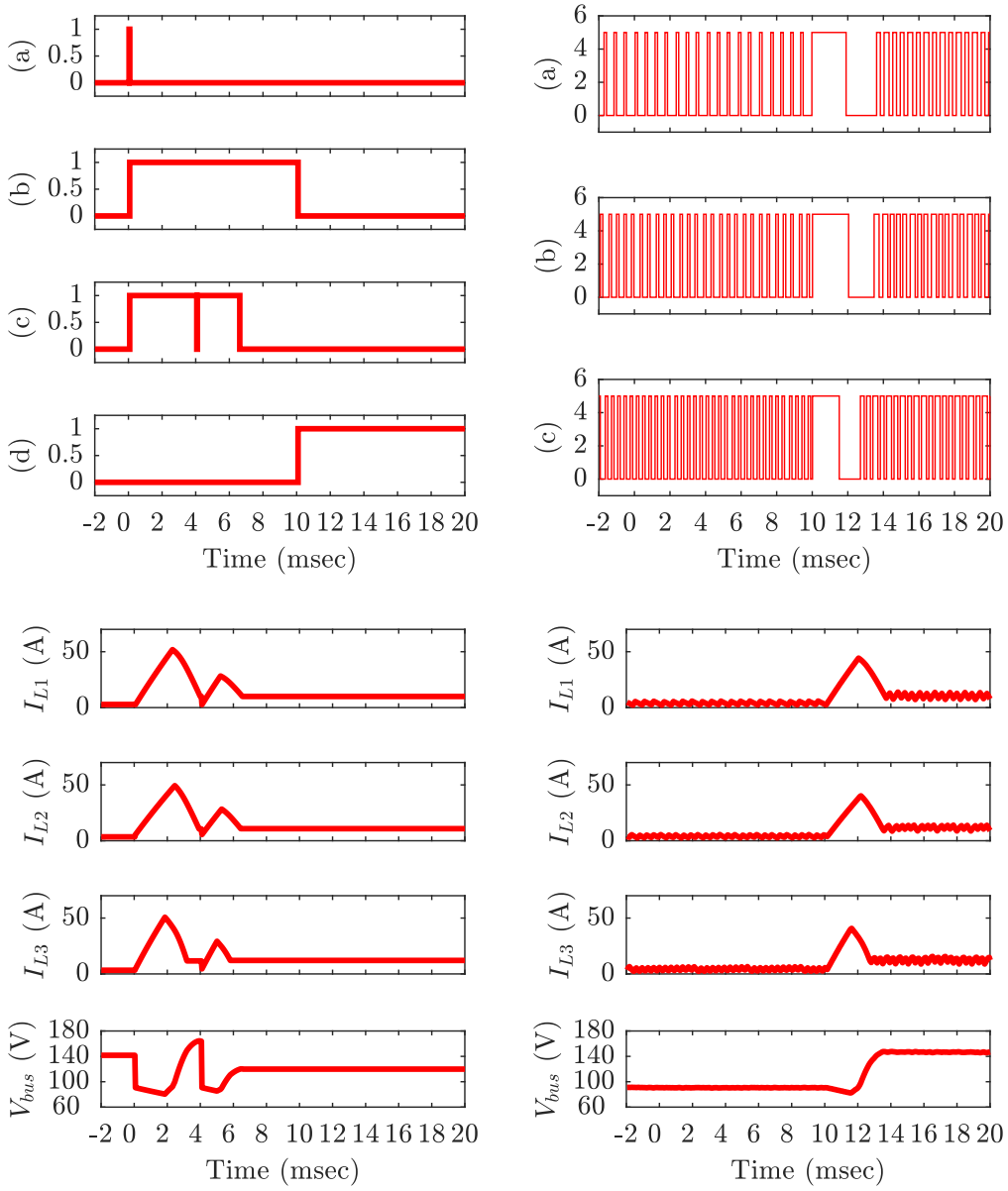


Figure A.19: HIL emulation results for $R_C = 0.001\Omega$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

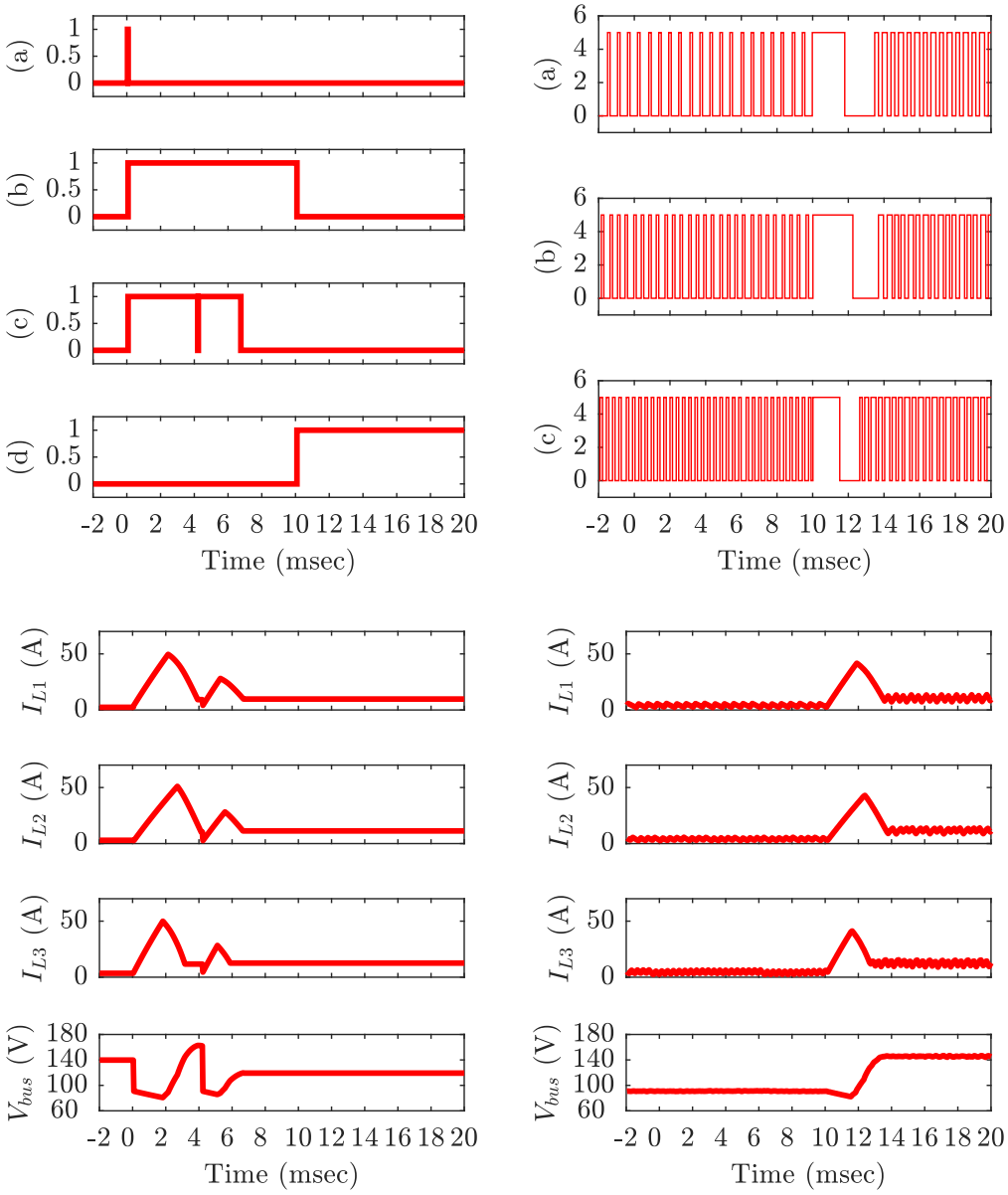


Figure A.20: HIL emulation results for $R_C = 0.05\Omega$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

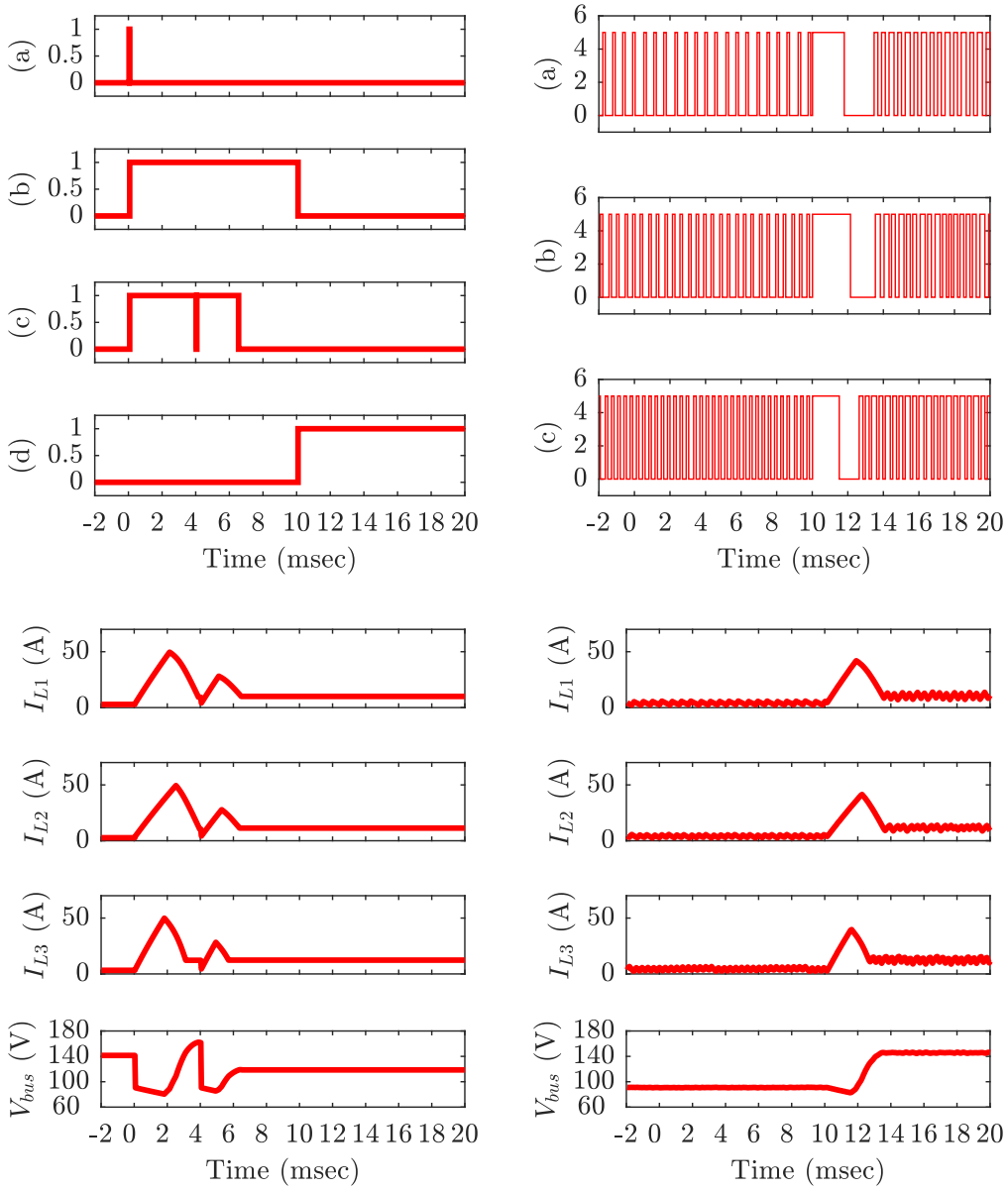


Figure A.21: HIL emulation results for $R_C = 0.1\Omega$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

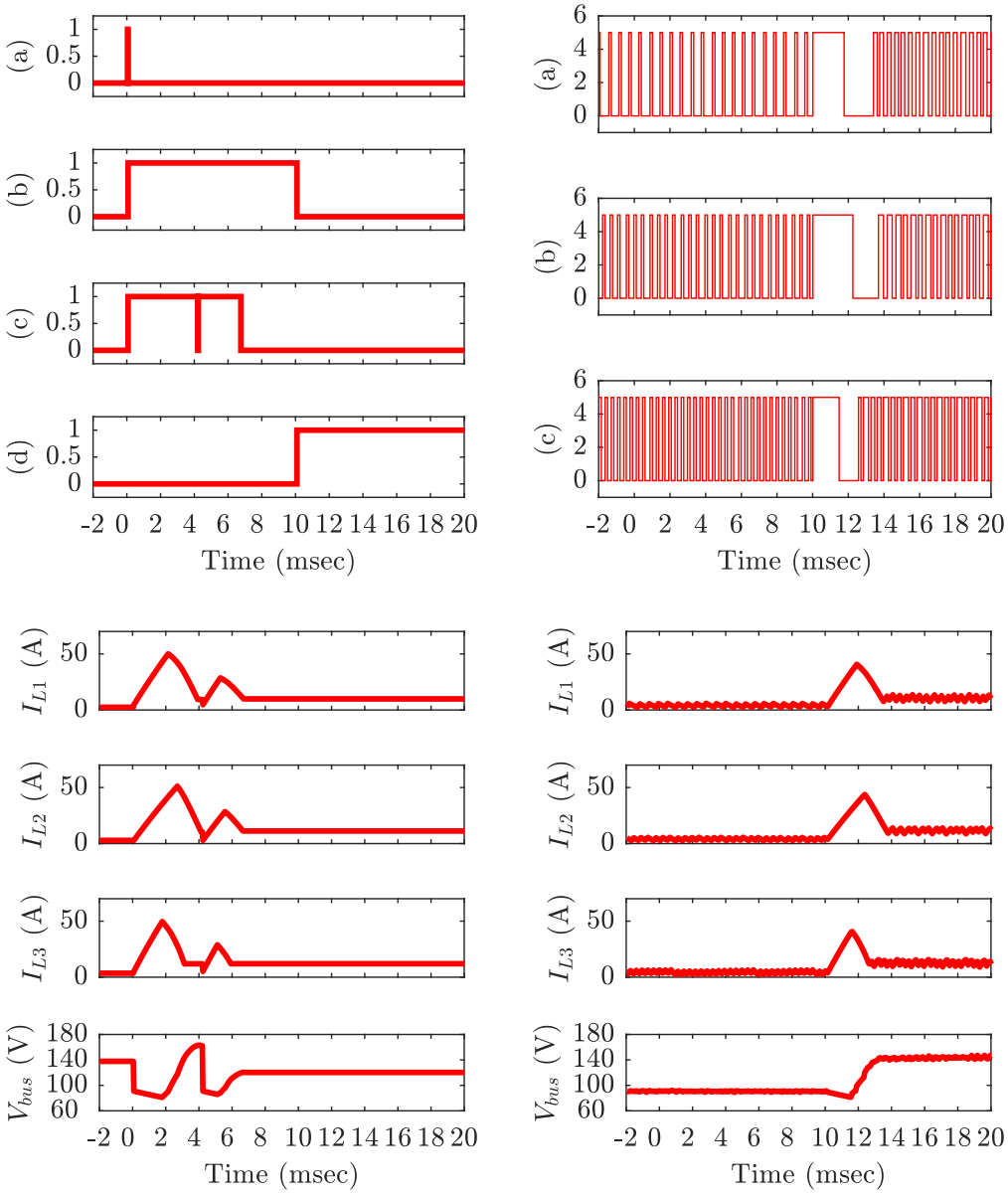


Figure A.22: HIL emulation results for $R_C = 0.15\Omega$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

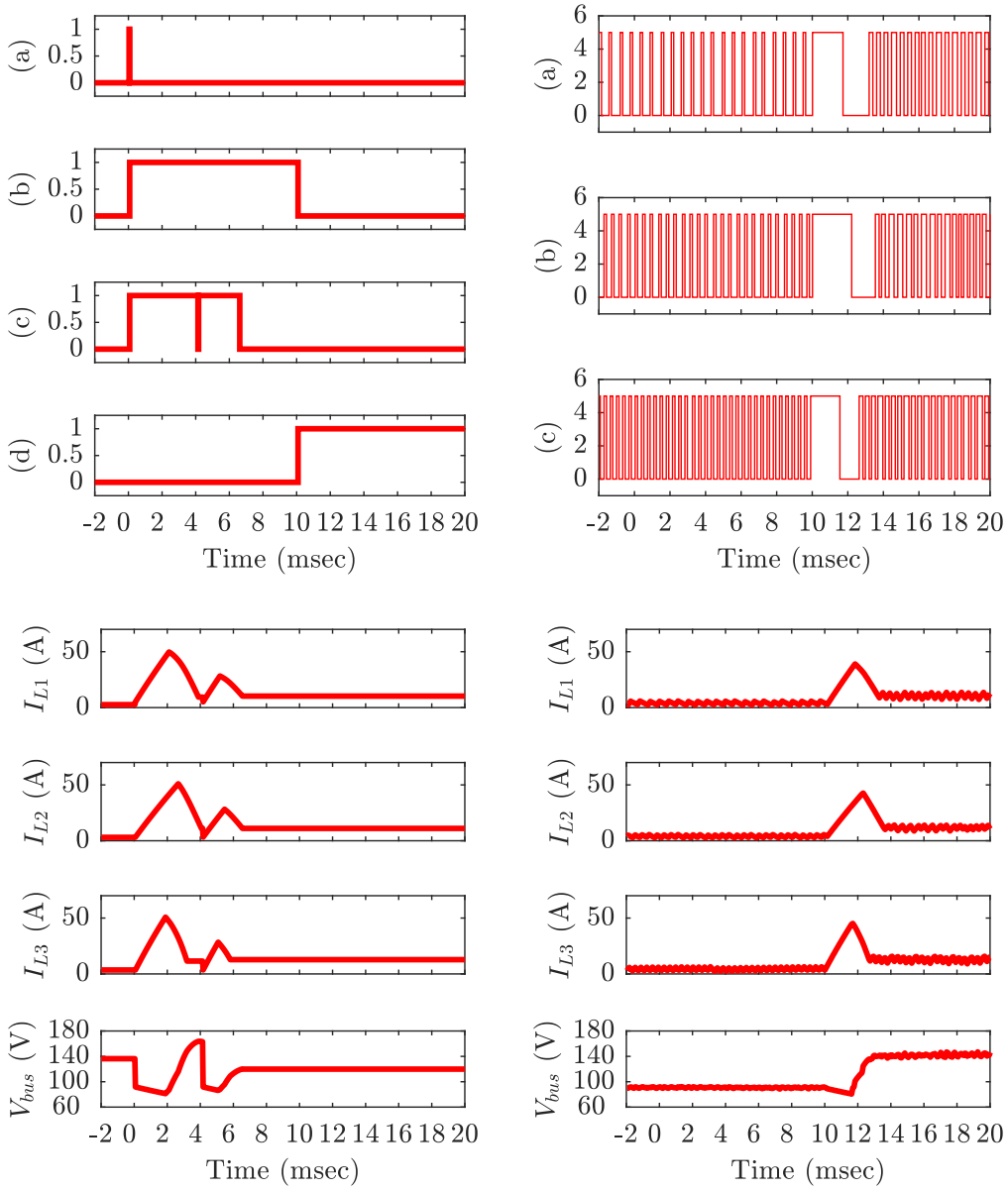


Figure A.23: HIL emulation results for $R_C = 0.20\Omega$; (left-half) (a)simulation trigger, (b)simulation strobe, (c)simulation strobe for $\beta = 0.8$ and $\beta = 0.4$ and , (d)emulation trigger; (right-half) (a) PWM signal for Q_1, (b) PWM signal for Q_2, (c) PWM signal for Q_3

A.5 MATLAB code for implementation of SMC and MTC for paralleled Boost converters

```
%% ==> main.m <==
% This file contains main function that invokes dependent processes

%% ------------------------> START OF SCRIPT <--------------------------
clc
clear
warning off
format shortg
sys = function1_load_system();
figure

%% Setting up intial and final values of voltage
tic
v0 = 90;
vfinal = 150;
for i = 1:sys.no_of_stage
il0(1,i) = 0.33*((v0-sys.stage(i).Vd_on)^2)/(sys.load.Rload*...
                                        sys.stage(i).Vs);
il_final(1,i) = 0.33*((vfinal-sys.stage(i).Vd_on)^2)/...
                                        (sys.load.Rload*sys.stage(i
   ).Vs);
end
% il_final = [11.5 9.8 9.3];
X0 = [il0 v0];
ton_safe = function8_ton_maximum(sys,il0)'

% finding tou and tou_prop
[tou,CF] = function7_calculate_tou(sys,il0)
CF_norm = CF/min(CF)

% best value of beta to get vfinal
beta = function9_find_beta(vfinal,sys,X0,il_final,ton_safe,CF_norm)

%% first phase of simulation
print_result = 1;

prime_scheduler = function3_prime_scheduler((beta.*CF_norm.*
   ton_safe),
```

```
                                        6*1e-3,sys);
primesim_result = function4_prime_simulation(sys,prime_scheduler,X0
    ,
                                        print_result);

% second phase of simulation to reach to the next operating point
% all stages are in dcm mode
last_state = primesim_result(end,:);
[vc_final,T_end,T_switchback] = function6_sub_simulation(sys,...
                                        last_state,il_final,
    print_result)
T_off = (beta.*CF_norm.*ton_safe)
toc;
```

```
%% simulation param
step_time = 0.06;
i_margin = 0.01;
vfinal = vfinal;
sample_time = 1e-5;
```

```
%% calculating energy shared by each stage
time = primesim_result(:,1);
A = 0.5.*primesim_result(:,2:end-1).^2;
L=[];
for i = 1:sys.no_of_stage
    L = horzcat(L,sys.stage(i).L);
end
temp = bsxfun(@times,A,L);

cap_energy = 0.5*sys.load.C.*(primesim_result(:,end).^2);
```

```
%% Plotting result
figure
plot(time,temp,'LineWidth',2);
hold on
plot(time,cap_energy,'LineWidth',2);
grid on

energy_result = [time, temp, cap_energy];
%% ------------------------> END OF SCRIPT <--------------------------
```

```
%% ==> main.m(revised version) <==
% This source file contains improved main function that invokes
% dependent processes
% for plotting MTC results

%% ------------------------> START OF SCRIPT
   <--------------------------
clc
clear

% fixed parameters
n       = 3;                            % number of boost converters
Vs      = [72 65 58];                   % source voltages (V)
Vd      = [0.10 0.15 0.20];             % voltage drop of IGBT/Diode
L       = [3.00e-3 3.20e-3 2.00e-3];    % inductanc values (H)
il_th   = [60 60 60];                   % Maximum allowed currents
RL      = [0.200 0.200 0.200];          % resistance of inductor (
   Ohm)
Rsw     = [0.020 0.020 0.020];          % resistnace of IGBT/Diode(
   Ohm)
C       = 1.5e-3;                       % capacitance of cap-bank(F)
Rload   = 10;                           % load resistance (Ohm)
CF      = [0.33 0.33 0.33];

step_time = 0.03;
sim_time = 3*step_time;

Vinit = 90;
Vfinal = 150;
[my_beta,switch_time] =  mtc(Vinit,Vfinal)

mdl = 'SMC_3_identical_boost_converter';
print_result = false;
i_margin = 0.01;
sample_time = 1e-5;

% for overshoot adjustment take 95% of actual switch time
switch_time = 0.95*switch_time;
sim(mdl);
clc
```

```
%% ------------------------------> END OF SCRIPT
   <------------------------------
```

```
%% ==> MTC implementation <==

function [mybeta,switchtime] =  mtc(Vinit,Vfinal)
%% Global configuration parameters used in this function
% Commonely used parameters and flags
tic
dt     = 0.5e-5;   % step-time/resolution for the differential
    system
% op_complete = false; % flag indicating "non-complete" status of
    the
                        % execution of this function

%% Define system parameters
% 1. Complete system parameters are defined here that are available
%    for later use in the script. Source voltage or any parameter
    that
%    is coming from the real-time measurement can also be supplied
%    here therough input to the function. In this script, all the
%    system parameters are assumed to be pre-loaded.

% 2. Vout is the most important input to this script and it will be
%    utilised later to find inductor currents at next opertaing
    point
%    based on CF_nuetral value. If any other value/pattern of
    inductor
%    currents are required at next operating point then they can
    also
%    be supplied as input to this script and the portaion of the
%    script calculating il_final will be skipped.

% 3. Source voltage and IGBT/diode drop-off voltage are merged and
%    given new name to represent effective Source voltage. It
    reduces
%    differetial system calculations. Same with the series
    inductance
%    resistance and IGBT/Diode forward resistance. For this, it is
%    assumed that forward voltage and resistance for IGBT/Diode in
%    both possible circuit combination(ON/OFF) are identical abd
%    hence can be reduced to one variable to make cdalculations
    easy.
```

```
%% ------------------------> START OF SCRIPT
   <-------------------------

% fixed parameters
n       = 3;                              % number of boost
   converters
Vs      = [72 65 58];                     % source voltages (V)
Vd      = [0.10 0.15 0.20];               % voltage drop of IGBT/
   Diode
L       = [3.00e-3 3.20e-3 2.00e-3];      % inductanc values (H)
il_th   = [60 60 60];                     % Maximum allowed currents
RL      = [0.200 0.200 0.200];     % resistance of inductor (Ohm)
Rsw     = [0.020 0.020 0.020];     % resistnace of IGBT/Diode (Ohm)
C       = 1.5e-3;                  % capacitance of capacitor bank
   (F)
Rload   = 10;                      % load resistance (Ohm)
CF      = [0.33 0.33 0.33];

% Modified parameters to simplify calculations
V       = Vs - Vd;                % effective source voltage
R       = RL + Rsw;               % effective resistance for both
   path

% finding Ilinit based on CF and SMC
Ilinit(1,1) = (CF(1))*((Vinit)^2)/(Rload*Vs(1));
Ilinit(1,2) = (CF(2))*((Vinit)^2)/(Rload*Vs(2));
Ilinit(1,3) = (CF(3))*((Vinit)^2)/(Rload*Vs(3));
Ilinit

% fiinding Ilfinal based on CF and SMC requirements
Ilfinal(1,1) = (CF(1))*((Vfinal)^2)/(Rload*Vs(1));
Ilfinal(1,2) = (CF(2))*((Vfinal)^2)/(Rload*Vs(2));
Ilfinal(1,3) = (CF(3))*((Vfinal)^2)/(Rload*Vs(3));
Ilfinal

% Run-time
%% Find "t_margin" for each stage
% 1. "t_margin" is the maximum time allowed for a boost converter
%    stage to stay ON. This duration of ON time is based on the
%    maximum current allowed through the inductor/IGBT switch.
```

```
   Based
%    on maximum allowable current limit, it is not feasible/safe to
%    push IGBT switchs to carry more currents than its thresholds.
   So,
%    t_margin is crucial factor for safe operation of system during
%    ON time.

% 2. Since, all the boost converter stages are electrically
   isolated
%    during ON time, it is very convenient to find time when
   current
%    reaches certain value, based on RL circuit analysis.

t_margin = zeros(1,n);
for i = 1:n
t_margin(i) = -(L(i)/R(i))*(log(1-(((il_th(i)-Ilinit(i))*R(i))/...
                                (V(i)))));
end

%% Find "nuetral CF" for SMC current control
% this section will be removed from this function and implemented
% saperately to work with SMC controller

%% Find next operating point(SMC) current from "Vout" and "nuetral
   CF"
% Since this function is directly accepting the final current
   values,
% calculation of these currents will be done externally to this
% function based on CF

%% Find "tou_proportional" and "CF" for the current system states
% 1. tou and tou_proportional are very important parameters. tou is
% used tocalculate tou_proportion which indicates comparative
% contribution of each boost converter stage in terms of switching
% time. With a value of 1, the respective stage is considered as
   base
% stage and rest of the stage will have switch time that is scaled
% based on tou_proportional.

% finding tou
```

```
tou = zeros(1,n);
alfa = sum(V(1:n)./L(1:n));
for i = 1:n
   tou(i) =  (il_th(i) - Ilinit(i)) / alfa;
end

% finding tou_proportional
a = min(tou);
tou_prop = tou./a;

%% Create two switching cases based on "beta"
% for characterization oflinear realtion between "beta" and "Vout"

% give beta input here
beta1 = 0.40:0.40:0.90;
a = length(beta1);
vresult = zeros(a,1);
iresult = zeros(a,n);
tswitchresult = zeros(a,n);

%% Switching case 1 for "beta" = 0.35
for m = 1:length(beta1)

T1 = sortrows([(1:n)' (beta1(m).*tou_prop.*t_margin)'],2);
t_start = 0;                      % start time for simulation
% maximum time for simulation 5*max(T1(:,2));
t_end = (1.5/beta1(m))*max(T1(:,2));
tstart1_temp = [t_start;T1(:,2)];     % start time matrix
tend1_temp = [T1(:,2);t_end];         % end time matrix
% logical operator for switch ON/OFF state
A1_temp = zeros(n+1,3);
for i = 2:n+1
   A1_temp(i,:) = A1_temp(i-1,:);
   A1_temp(i,T1(i-1,1)) = A1_temp(i-1,T1(i-1,1)) + 1;
end
% removing very small(<10e-5sec)/zero switching time states
ind1 = find(tend1_temp-tstart1_temp <= 5*dt);

% [tstart1_temp,tend1_temp,A1_temp]  % debug point, uncomment to
   see
```

```
%  output
invind1 = setdiff(1:n+1,ind1);
A1 = A1_temp(invind1,:);
tstart1 = tstart1_temp(invind1);
tend1 = tend1_temp(invind1);
% adjusting time discontinuities
tstart1(2:end) = tend1(1:end-1);
% [tstart1,tend1,A1]    % debug point, uncomment to see output


% Simulating system for switching time based on beta1
% setting inital conditions for simulation array

vend = 0;
iend = zeros(1,n);
B1 = [1 1 1];

for j = 1:length(tstart1)
   % length of this segment for loop
   l = round((tend1(j) - tstart1(j)) / dt);
   isim = zeros(l,n);
   vsim = zeros(l,1);

        if j == 1
            vsim(1) = Vinit;
            isim(1,:) = Ilinit;
        else
            vsim(1) = vend;
            isim(1,:) = iend;
        end

   for i = 2:l
    isim(i,1) = isim(i-1,1) + dt*(((V(1)-isim(i-1,1)*R(1))/(L(1)))-
   ...
                                                ((vsim(i-1)/L(1))*
   A1(j,1)))*B1(1);
    isim(i,2) = isim(i-1,2) + dt*(((V(2)-isim(i-1,2)*R(2))/(L(2)))-
   ...
                                                ((vsim(i-1)/L(2))*
   A1(j,2)))*B1(2);
```

```
    isim(i,3) = isim(i-1,3) + dt*(((V(3)-isim(i-1,3)*R(3))/(L(3)))-
...
                                                ((vsim(i-1)/L(3))*
A1(j,3)))*B1(3);
   vsim(i)  = vsim(i-1)  + dt*(-(vsim(i-1)/(Rload*C))...
                                    + A1(j,1)*B1(1)*(isim(i-1,1)/C)
...
                                    + A1(j,2)*B1(2)*(isim(i-1,2)/C)
...
                                    + A1(j,3)*B1(3)*(isim(i-1,3)/C)
);
end

    if all(A1(j,:)==0)

        vend = vsim(l);
        iend = isim(l,:);
%         vresult = vertcat(vresult,vsim);
%         iresult = vertcat(iresult,isim);

    else

        % check for SMC currents
        index = zeros(n,1);
%     for i = 1:n
%       if isempty(find(abs(isim(:,i)-Ilfinal(i))<= 0.1,1,'last'))
%        index(i,1) = l;
%       else
%        index(i,1) = find(abs(isim(:,i)-Ilfinal(i))<= 0.1,1,'last
');
%       end
%     end

      for i = 1:n
         if isempty(find(abs(isim(:,i)-Ilfinal(i))<= 0.1,1,'last
'))
              index(i,1) = l;
         else
                for z = l:-1:1
                    if abs(isim(z,i)-Ilfinal(i))<= 0.1
```

```
                    index(i,1) = z;
                end
            end
        end
    end


    % if SMC not detected,then set initial conditions and
store
            %  results
    if all(index == l) == 1
                % repeat = 0;
        vend = vsim(l);
        iend = isim(l,:);
                % vresult = vertcat(vresult,vsim);
                % iresult = vertcat(iresult,isim);

    else
        % if SMC detected single/multiple
        % find stages and change B flag
        temp_index = min(index);
        SMC_stage = ~(abs(temp_index - index) <= 5);
        B1 = B1.*SMC_stage';
        % set SMC flag for repeatation loop
        repeat = 1;
        % set end values as initial conditions
        vend = vsim(temp_index);
        iend = isim(temp_index,:);
        % store previous results
        % vresult = vertcat(vresult,vsim(1:temp_index,:));
        % iresult = vertcat(iresult,isim(1:temp_index,:));
        % update start and end time for simulation
        tstart = tstart1(j) + temp_index*dt;

        while repeat == 1
            % length of this segment for loop
            l = round((tend1(j) - tstart) / dt);
            isim = zeros(l,n);
            vsim = zeros(l,1);
            vsim(1) = vend;
```

```
                isim(1,:) = iend;

                for i = 2:l
                isim(i,1) = isim(i-1,1) + dt*(((V(1)-isim(i-1,1)
    ...
                                        *R(1))/(L(1)))-((vsim(i-1)/
    L(1))*A1(j,1)))*B1(1);
                isim(i,2) = isim(i-1,2) + dt*(((V(2)-isim(i-1,2)
    ...
                                        *R(2))/(L(2)))-((vsim(i-1)/
    L(2))*A1(j,2)))*B1(2);
                isim(i,3) = isim(i-1,3) + dt*(((V(3)-isim(i-1,3)
    ...
                                        *R(3))/(L(3)))-((vsim(i-1)/
    L(3))*A1(j,3)))*B1(3);
                  vsim(i)  = vsim(i-1)  + dt*(-(vsim(i-1)/...
                     (Rload*C))+ A1(j,1)*B1(1)*(isim(i-1,1)/C)
    ...
                     + A1(j,2)*B1(2)*(isim(i-1,2)/C)...
                     + A1(j,3)*B1(3)*(isim(i-1,3)/C));
                end

                % check for SMC currents
                index = zeros(n,1);
%                 for i = 1:n
%                     if isempty(find(abs(isim(:,i)-Ilfinal(i))
    <=
%                         0.1,1,'last'))index(i,1) = l;
%                     else
%                         index(i,1) = find(abs(isim(:,i)-...
%                             Ilfinal(i))<= 0.1,1,'last');
%                     end
%                 end

                for i = 1:n
                    if isempty(find(abs(isim(:,i)-Ilfinal(i))<=
    ...
                             0.1,1,'last'))
                        index(i,1) = l;
                    else
```

```
                for z = l:-1:1
                    if abs(isim(z,i)-Ilfinal(i))<= 0.1
                         index(i,1) = z;
                    end
                end
            end
        end


        if all(index == l) == 1
            repeat = 0;
            vend = vsim(l);
            iend = isim(l,:);
                     % vresult = vertcat(vresult,vsim);
                     % iresult = vertcat(iresult,isim);

        else
            % if SMC detected single/multiple
            % find stages and change B flag


            temp_index = min(index(B1==1));
            SMC_stage = ~(abs(temp_index - index) <= 3);
            B1 = B1.*SMC_stage';
            % set end values as initial conditions
            vend = vsim(temp_index);
            iend = isim(temp_index,:);
            % set SMC flag for repeatation loop
              if all(B1 == 0) == 1
                    repeat = 0;
                    % output values are below, required
for
                    % next phase of calculation
                    vresult(m) = vend;
                    iresult(m,:) = iend;
                    tswitchresult(m,:) = (beta1(m).*...
                     tou_prop.*t_margin);
                    % output to observe values on
command
                    % line finalresult = [beta1' vresult
```

```
                                iresult tswitchresult];
                            else
                                repeat = 1;
                            end
                        % store previous results
                        %  vresult = vertcat(vresult,vsim(1:temp_index
    ,:));
                        %  iresult = vertcat(iresult,isim(1:temp_index
    ,:));
                        % update start and end time for simulation
                           tstart = tstart + temp_index*dt;
                       end

                   end
               end

           end

end

% hold on
% plot(vresult,'Linewidth',2);
% plot(iresult,'Linewidth',2);
% grid on;
end
%% Find final "beta" value for desired Vout
slope = (vresult(2) - vresult(1))/(beta1(2) - beta1(1));
C = vresult(2) - slope*beta1(2);
mybeta = (Vfinal - C) / slope;
switchtime = (mybeta.*tou_prop.*t_margin);


%% Find final switching values "t_switch" as output of this
    function
% Important Note : The SMC will take control of operation after
% t_switch for each stage. So, the only required output is the
% switching time. Once the boost stage is in ON mode until t_switch
    ,
% the only possibility for SMC controller is to TURN OFF the switch
% and continue when current reaches to the next operating point.
```

```
   This
%  mechanism reduces the effort to design a scheduler for whole
   event.

% t_switch_final = 0;

%% End of Function
%op_complete = true;%flag indicating the completetion of the
   execution
                        % of this function
toc
end

%% ------------------------------> END OF SCRIPT
   <------------------------------
```

```
%% ==> Loading system parameters of multiple Boost converters <==
%
% This file consist the complete definition of the system (all
    Boost
% convereter stages & load) parameters. All the variables defined
% hereby are available to all other .m files. It creates the system
% structure named "sys" in the workspace. Inital values of the
    system
%(iL & Vc) are required inputs.
%
% Created by    : Shishir Patel
% Created on    : 11/27/2015 Friday
% Last Modified : 207/24/2016 Sunday

%% ------------------------> START OF SCRIPT
   <--------------------------

function [sys] = function1_load_system()

%% System parameters (time-independent)

% stage 1 data
sys.stage(1).Rl = 0.005; % paracitic resistance of the inductor
    coil
sys.stage(1).L  = 5e-3; % inductnace value
sys.stage(1).Vs = 65; % source voltage for this stage
sys.stage(1).il_th = 65; % threshold current for the indctor
sys.stage(1).Rsw = 0.002; % ON resitnace of the switch for this
    stage
sys.stage(1).Vd_on = 0.6; % Reverse recovery diode ON voltage
sys.stage(1).B_th = 0; % Maximum magnetic flux density in inductor
sys.stage(1).Rd = 0.002; % forward resitnace of the diode

% stage 2 data
sys.stage(2).Rl = 0.005; % paracitic resistance of the inductor
    coil
sys.stage(2).L  = 5.5e-3; % inductnace value
sys.stage(2).Vs = 72; % source voltage for this stage
sys.stage(2).il_th = 60; % threshold current for the indctor
sys.stage(2).Rsw = 0.002; % ON resitnace of the switch for this
```

```
   stage
sys.stage(2).Vd_on = 0.6; % Reverse recovery diode ON voltage
sys.stage(2).B_th = 0; % Maximum magnetic flux density in inductor
sys.stage(2).Rd = 0.002; % forward resitnace of the diode

% stage 3 data
sys.stage(3).Rl = 0.004; % paracitic resistance of the inductor
   coil
sys.stage(3).L  = 4.6e-3; % inductnace value
sys.stage(3).Vs = 75; % source voltage for this stage
sys.stage(3).il_th = 56; % threshold current for the indctor
sys.stage(3).Rsw = 0.002; % ON resitnace of the switch for this
   stage
sys.stage(3).Vd_on = 0.6; % Reverse recovery diode ON voltage
sys.stage(3).B_th = 0; % Maximum magnetic flux density in inductor
sys.stage(3).Rd = 0.002; % forward resitnace of the diode

% % stage 4 data
% sys.stage(4).Rl = 0.25; % paracitic resistance of the inductor
   coil
% sys.stage(4).L  = 1.89e-3; % inductnace value
% sys.stage(4).Vs = 22; % source voltage for this stage
% sys.stage(4).il_th = 38; % threshold current for the indctor
% sys.stage(4).Rsw = 0.02; % ON resitnace of the switch for this
   stage
% sys.stage(4).Vd_on = 0.6; % Reverse recovery diode ON voltage
% sys.stage(4).B_th = 0; % Maximum magnetic flux density in
   inductor
% sys.stage(4).Rd = 0.1; % forward resitnace of the diode
%
% % stage 5 data
% sys.stage(5).Rl = 0.2; % paracitic resistance of the inductor
   coil
% sys.stage(5).L  = 1e-3; % inductnace value
% sys.stage(5).Vs = 24; % source voltage for this stage
% sys.stage(5).il_th = 35; % threshold current for the indctor of
   this
% sys.stage(5).Rsw = 0.02; % ON resitnace of the switch for this
   stage
% sys.stage(5).Vd_on = 0.6; % Reverse recovery diode ON voltage
```

```
% sys.stage(5).B_th = 0; % Maximum magnetic flux density in
    inductor
% sys.stage(5).Rd = 0.1; % forward resitnace of the diode

% Output stage
sys.load.C = 1.5e-3; % output capacitance value
sys.load.Rc = 0.00; % series resistance of the load side capacitor
sys.load.Rload = 10; % Load resistance
sys.load.Vc_th = 250; % output capacitor threshold voltage

% number of boost converter stages defined as "whole system".
sys.no_of_stage = size(sys.stage,2);

sys.phase = ones(1,sys.no_of_stage);
% It represents an array of length of total number of stages.
    Primary
% use of this array is to denote that which stage is participating
    in
% particular operation; by indicating '1' in corresponding position
% and '0' for absense of the stage

%% System parameters (time-independent)
% these parameters of the system depends on the operating point. So
     it
% is run-time input variables. While running this script within
    model,
% these parameters are system states (iL,Vc) just a moment before
% switching the operation.

% these parameters will be removed from this file in later updates
    so
% that it do not require to load whole system for just 4 values
    !!!!!

% shifting this portion to function2 input
% sys.stage(1).il0 = il1_init;%initial inductor current for this
    stage
% sys.stage(2).il0 = il2_init;%initial inductor current for this
    stage
% sys.stage(3).il0 = il3_init;%initial inductor current for this
```

```
    stage
% sys.load.Vc0 = Vc_init; % initial voltage across the capacitor/
    LOAD

end
%% ------------------------> END OF SCRIPT
    <-------------------------
```

```
%% ==> Simulating system <==
%
% This file contains the simulation of system for given duration
    and
% initial conditions. Additional input for this function will be
    state
% of each boost converter stage (ON/OFF/DCM)/(1/0/-1). Depending
    upon
% the state of the system this function will create appropeiate
    state
% model for whole system and simulate it. At the end it will find
    the
% DCM point if any for any stage. This function returns all the
    system
% states and the last valid time instance from where next phase of
% simukation will continue.
%
% Created by    : Shishir Patel
% Created on    : 11/2/2016 Friday
% Last Modified : 20/2/2016 Sunday

%% ------------------------> START OF SCRIPT
   <--------------------------

function [A,B,U,Vout,T,X]=function2_simulate_system(sys,sys_phase,
    ...

    t_start,t_end,X0)

%% state matrix A and input matrix B
a = length(sys_phase); % number of independent boost converter
    stages
% or number of times we have to repeate the loop to configure
    currents

% Initializing system matrices with zeros
A = zeros(a+1,a+1);
B1 = zeros(a+1,a); % input matrix for source voltage Vs
B2 = zeros(a+1,a); % input matrix for diode forward voltage Vd_on
% B = [B1,B2];     % Cummmulative input matrix of size - (a+1)x(2a)
```

```
C = zeros(1,a+1);  % output voltage across load
D = zeros(1,2*a);

for i = 1:a
    % normal configuration for all cases
    % luckily it is also implementation for sys_phase(i) = -1 DCM
    mode
    A(end,end) = -1/(sys.load.C * (sys.load.Rload + sys.load.Rc));
    C(1,end) = sys.load.Rload / (sys.load.Rload + sys.load.Rc);

    % Piecewise configuration as per the mode of each boost
    converter
        % stage
    if sys_phase(i) == 1
        A(i,i) = -(sys.stage(i).Rl + sys.stage(i).Rsw) / ...
                                        (sys.stage(i).L);
        B1(i,i) = 1/sys.stage(i).L;
    elseif sys_phase(i) == 0
        % some commonly used constant terms in this section
        x = 1/(sys.load.Rload + sys.load.Rc);
        y = sys.load.Rload*x;
        z = 1/sys.stage(i).L;

        A(i,i) = -(1/sys.stage(i).L)*((sys.stage(i).Rl + ...
                                sys.stage(i).Rd + sys.load.Rc)-(x*
    sys.load.Rc^2));
        A(i,end) = -y*z;
        A(end,i) = y/sys.load.C;

        B1(i,i) = z;
        B2(i,i) = -z;

        C(1,i) = y*sys.load.Rc;
    end

end
B = [B1,B2]; % very important step to consider changes
```

%% Simulating linear system

```
system = ss(A,B,C,D); % creating state space model of the system
t = linspace(t_start,t_end,1e6*(t_end-t_start));%factor 1000 is for
    ms
%X0=[sys.stage(1).il0;sys.stage(2).il0;sys.stage(3).il0;sys.load.
   Vc0];
%initial condition and simulation times are coming from function
%input

F = repmat(ones(1,length(t)),2*a,1); % time vector for input
% preallocating size for input matrices
inp1 = zeros(a,1);
inp2 = zeros(a,1);
for j = 1:a
    inp1(j,1) = sys.stage(i).Vs; %matrix for voltage source as
   input
    inp2(j,1) = sys.stage(i).Vd_on; %matrix for diode voltage as
   input
end
% merging both inputs in one matrix
Input = [inp1;inp2];
% final input matrix for lsim
U = bsxfun(@times,F,Input);

% simulating system
[Vout,T,X] = lsim(system,U,t,X0);

end
%% ------------------------> END OF SCRIPT
   <----------------------------
```

```
%% ==> Simulating system <==
%
% This file contains the simulation of system for given duration
    and
% initial conditions. Additional input for this function will be
    state
% of each boost converter stage (ON/OFF/DCM)/(1/0/-1). Depending
    upon
% the state of the system this function will create appropeiate
    state
% model for whole system and simulate it. At the end it will find
    the
% DCM point if any for any stage. This function returns all the
    system
% states and the last valid time instance from where next phase of
% simukation will continue.
%
% Created by    : Shishir Patel
% Created on    : 11/2/2016 Friday
% Last Modified : 20/2/2016 Sunday

%% ------------------------> START OF SCRIPT
   <--------------------------

function [A,B,U,Vout,T,X] = function21_simulate_system(sys,...

    sys_phase,t_start,t_end,X0)

%% state matrix A and input matrix B
a = length(sys_phase); % number of independent boost converter
    stages
% or number of times we have to repeate the loop to configure
    currents

% Initializing system matrices with zeros
A = zeros(a+1,a+1);
B1 = zeros(a+1,a); % input matrix for source voltage Vs
B2 = zeros(a+1,a); % input matrix for diode forward voltage Vd_on
% B = [B1,B2];     % Cummmulative input matrix of size - (a+1)x(2a)
C = zeros(1,a+1);  % output voltage across load
```

```
D = zeros(1,2*a);

for i = 1:a
    % normal configuration for all cases
    % luckily it is also implementation for sys_phase(i) = -1 DCM
   mode
    A(end,end) = -1/(sys.load.C * (sys.load.Rload + sys.load.Rc));
    C(1,end) = sys.load.Rload / (sys.load.Rload + sys.load.Rc);

    % Piecewise configuration as per the mode of each boost
   converter
        % stage
    if sys_phase(i) == 1
        A(i,i) = -(sys.stage(i).Rl + sys.stage(i).Rsw) / ...
                                        (sys.stage(i).L);
        B1(i,i) = 1/sys.stage(i).L;
    elseif sys_phase(i) == 0
        % some commonly used constant terms in this section
        x = 1/(sys.load.Rload + sys.load.Rc);
        y = sys.load.Rload*x;
        z = 1/sys.stage(i).L;

        A(i,i) = -(1/sys.stage(i).L)*((sys.stage(i).Rl + ...
                                        sys.stage(i).Rd + sys.load.
   Rc)-(x*sys.load.Rc^2));
        A(i,end) = -y*z;
        A(end,i) = y/sys.load.C;

        B1(i,i) = z;
        B2(i,i) = -z;

        C(1,i) = y*sys.load.Rc;
    end

end
B = [B1,B2]; % very important step to consider changes
```

```
%% Simulating linear system

% preallocating size for input matrices
```

```
inp1 = zeros(a,1);
inp2 = zeros(a,1);
for j = 1:a
    inp1(j,1) = sys.stage(i).Vs; % matrix for voltage source as
   input
    inp2(j,1) = sys.stage(i).Vd_on;% matrix for diode voltage as
   input
end
% merging both inputs in one matrix
U = [inp1;inp2];

% f = @(t,x) A*[x(1);x(2);x(3);x(4)] + B*U;
% [T,X] = ode45(@odefun,timespan,X0,[],A,B,U);

step_size = 0.00001;
T = t_start:step_size:t_end;
t_sim = t_start;
X = zeros(length(T),a+1);
X(1,:) = X0;
i = 1;

while t_sim <= t_end
   X(i+1,:) = X(i,:) + (step_size*(A*(X(i,:)') + B*U)');
   t_sim = t_sim + step_size;
   i = i+1;
end

Vout = 0; %redundant variable,neet to remove
end
```

%% ————————————————————> **END OF SCRIPT**
<————————————————————

```
%% ==> Prime scheduler <==
%
% This function contains complete logic for the primary simulation
% under different switching configurations. This function
    implements
% core part of the thesis and will be linked to the auxilary
    scheduler
% that cokputes the simulation after all the stages in the DCM mode
    .
% The DCM mode will be implemented in saperate function and will be
% linked later with this script.
%
% Created by    : Shishir Patel
% Created on    : 7/29/2016 Friday

%% ------------------------> START OF SCRIPT
   <--------------------------

function prime_schedule = function3_prime_scheduler(t_switch,...
                                                    t_simtime,
    sys)

% time instances for major events during complete simulation
    including
% starting time. t_switch is the input that represents "tou".
    t_switch
% is column array
stage_num = (1:sys.no_of_stage)';

% Creating new array that contains stage number and switching
    instance
% for that stage.it will make indexinh much easier.
switchtime_array = [stage_num, t_switch];

% sorting new array in terms of ascending switching time. Also
    sorting
% no_of_stage array accordingly.
sorted_array = sortrows(switchtime_array,2);

% schedular 1 (without_DCM)
```

```
starting_time = [0;sorted_array(:,2)];
ending_time = [sorted_array(:,2);t_simtime];

sys_phase(1,:) = ones(1,sys.no_of_stage);
for i = 2:(sys.no_of_stage+1)
    temp = sys_phase(i-1,:);
    temp(1,sorted_array(i-1,1))=sys_phase(i-1,sorted_array(i-1,1))
   -1;
%   temp(sorted_array(i)) = temp(sorted_array(i)) - 1;
    sys_phase(i,:) = temp;
end

% shedule for first simulation
schedule = [starting_time,ending_time,sys_phase];
schedule = schedule;
% enhaced scheduler for synchronous switching enabled operation
time_difference = diff(schedule(:,1:2),1,2);
ind = time_difference <= 1e-6;
schedule(ind == 1,:) = []; % this is the final output of this
   script
state_change = [diff(schedule(:,3:end),1,1);zeros(1,sys.no_of_stage
   )];
% state_change is very important param. It indicates upcoming
   change.
% The value -1 is set for easy opeartion, so that the next phase of
% system can be changed by just adding value of state change to
% current system phase. It makes implementation easy.

% creating final table
t_start = schedule(:,1);
t_end = schedule(:,2);
switch_mode = schedule(:,3:end);
prime_schedule = table(t_start,t_end,switch_mode,state_change)

end
%% ------------------------> END OF SCRIPT
   <----------------------------
```

```
%% ==> Prime_simulation <==

%% ------------------------> START OF SCRIPT
   <--------------------------

function simresult = function4_prime_simulation(sys,prime_schedule,
    ...
                                                X0,print_result)

     sim.switch_state = prime_schedule.state_change;
     sim.phase = sys.phase;
     sim.t = [];
     sim.X = [];
     sim.dcm_active = 0;
     sim.dcm_index = 0;
     sim.X0 = X0;
     sim.n = size(sim.switch_state,1); % no of iteration for "for"
    loop
     sim.no_of_stage = sys.no_of_stage;

     % simresult = [];
     simresult = zeros(1e3*(prime_schedule.t_end(end)-...
                                prime_schedule.t_start(1)),sys.
    no_of_stage + 2); ...
                                % size of simresult preallocated
     % time || currents || capacitor voltage

     for i = 1:sim.n

         sim.tstart = prime_schedule.t_start(i);
         sim.tend = prime_schedule.t_end(i);

         % main timing simulation
         [~,~,~,~,sim.t,sim.X] = function2_simulate_system(sys,...
                                                                  sim
    .phase,sim.tstart,sim.tend,sim.X0);
         sim = function5_check_for_dcm(sim,print_result);
         simresult = vertcat(simresult,[sim.t,sim.X]);

         while sim.dcm_active == 1
```

```
            [~,~,~,~,sim.t,sim.X] = function2_simulate_system(sys,
    ...
                                                                    sim
    .phase,sim.tstart,sim.tend,sim.X0);
            sim = function5_check_for_dcm(sim,print_result);
            simresult = vertcat(simresult,[sim.t,sim.X]);
            if all(sim.phase <= -1) == 1
                break;
            end
        end

        % changing system state for next simulation cycle based on
        % switching
        sim.phase = sim.phase + sim.switch_state(i,:);
        % plot(simresult,'DisplayName','sim_result')
        % hold on
     end

end
%% ------------------------> END OF SCRIPT
   <----------------------------
```

```
%% ==> Checking for DCM mode <==

%% ------------------------> START OF SCRIPT
   <--------------------------

function sim = function5_check_for_dcm(sim,print_result)

    % last index of simulation for comparison
    sim.last_index = size(sim.t,1)*ones(1,sim.no_of_stage);

    % check whether DCM occured or not
    for i = 1:sim.no_of_stage
       if isempty(find(sim.X(:,i) <= 0.025,1))
           index(1,i) = length(sim.t);
       else
           index(1,i) = find(sim.X(:,i) <= 0.025,1);
                   % finding last non-negative index
       end
    end

%%  changing system flags according to current state of system

    % No DCM detected
    if all(index == sim.last_index) == 1
        % if all values of index matches last index
        % changing system flgs and results for next simulation
        sim.dcm_active = 0;
        sim.row_index = sim.t(end);
        sim.X0 = sim.X(end,:);

    % DCM detected
    else
        % changing system flgs and results for next simulation

        flag = (all(sim.phase) <= -1);
        if flag == 1
            sim.dcm_active = 0;

        else
            sim.dcm_active = 1;
```

```
            % only considering index of those stages that are not
    yet
                        % in dcm
            eff_index = index(sim.phase ~= -1);
            if ~isempty(eff_index)
                sim.dcm_index = min(unique(eff_index));
            else
                sim.dcm_index = 1;
            end
            sim.t = sim.t(1:sim.dcm_index);
            sim.X = sim.X(1:sim.dcm_index,:);
            sim.X0 = sim.X(sim.dcm_index,:);
            sim.tstart = sim.t(sim.dcm_index);

            % finding which stages will enter into DCM phase
            dcm_active_state = abs(index - sim.dcm_index) <= 5;
            % changing system phase for next simulation
            sim.phase = sim.phase - dcm_active_state;

        end

    end

    % plotting results
    if print_result
    plot(sim.t,sim.X(),'DisplayName','sim_result','LineWidth',2);
    hold on;xlabel('Time (second)');ylabel('System states');grid on
    ;
    title(['System phase : ', num2str(sim.phase)]);
    pause(0.1)
    end

end
%% ------------------------> END OF SCRIPT
   <--------------------------
```

```
%% ==> Sub simulation cases <==

%% ------------------------> START OF SCRIPT
   <--------------------------

function [vc_final,t_final,switchon_time] =function6_sub_simulation
    ...
                                          (sys,last_state,il_final,
    print_result)

    % Sub-scheduler is not implemented yet. Only final capcitor
    % voltage is calculated based on maximum time taken to reach to
    % next operating point. This function will return sub-scheduler
     as
        % output when finished!

    t_start = last_state(1,1);
    t_end = t_start + 2e-3; % assuming that by 2ms all stages will
        % reach to thrie next operating point
    sys_phase = ones(1,sys.no_of_stage);
    X0 = last_state(1,2:end);

    % simulating system with approximated final time to find
    currents
    [~,~,~,~,T,X] = function2_simulate_system(sys,sys_phase,...
                                                t_start,t_end,X0);

    % finding time instances when final currents will match
    for i = 1:sys.no_of_stage
       index(1,i) = find((X(:,i)-il_final(1,i)) <= 0.025,1,'last');
       switch_time(1,i) = T(index(1,i));
    end

    % finding maximum time period and corresponding voltage
    [t_final,last_stage_no] = max(switch_time);
    vc_final = X(index(1,last_stage_no),end);

    % switchon time
    switchon_time = (t_start + t_final - switch_time)';
```

```
    % plotting result, not exact and final result;just for checking
     vc
    if print_result
    plot(T(1:index(1,last_stage_no),:),X(1:index(1,last_stage_no),
    ...
                         :),'DisplayName','sim_result','LineWidth'
    ,2);
    hold on;xlabel('Time (second)');ylabel('System states');grid on
    ;
    title(['System phase : ', num2str(sys_phase)])
    end
end
%% ------------------------> END OF SCRIPT
   <--------------------------
```

```
%% ==> calculating tau (contribution factor) <==

%% ------------------------------> START OF SCRIPT
   <------------------------------

function [tou,CF] = function7_calculate_tou(sys,il0)

    % calculating alfa
    alfa = 0;
    for i = 1:sys.no_of_stage
        alfa = alfa + (sys.stage(i).Vs/sys.stage(i).L);
    end

    for i = 1:sys.no_of_stage
         tou(i,1) = (sys.stage(i).il_th-il0(i))/(alfa);
    end

    CF = tou./(sum(tou));

end
%% ------------------------------> END OF SCRIPT
   <------------------------------
```

```
%% ==> Calculating maximum ON time <==

%% ------------------------> START OF SCRIPT
   <----------------------------

function ton_safe = function8_ton_maximum(sys,il0)
    % calculating maximum safe ON time for each stage
    t_start = 0;
    t_end = t_start + 6e-3; % assuming that by 2ms all stages will
        % reach to thrie next operating point
    sys_phase = ones(1,sys.no_of_stage);
    X0 = [il0 0];

    % simulating system with approximated final time to find
   currents
    [~,~,~,~,T,X] = function2_simulate_system(sys,sys_phase,...
                                                t_start,t_end,X0);

    % finding time instances when final currents will match
    for i = 1:sys.no_of_stage
       ton_safe(1,i) = T(find((X(:,i)-sys.stage(i).il_th)...
                                                <= 0.025,1,'last'))
   ;
    end

end
%% ------------------------> END OF SCRIPT
   <----------------------------
```

```
%% ==> Calculating beta <==

%% ------------------------> START OF SCRIPT
   <--------------------------

function beta = function9_find_beta(vfinal,sys,X0,il_final,...
                                        ton_safe,CF_norm)
print_result = 1;

%% full sweep beta plot segment
% for beta1 = 0.1:0.05:0.75;
% prime_scheduler = function3_prime_scheduler((beta1.*(CF_norm).*
% ton_safe),6*1e-3,sys);
% primesim_result = function4_prime_simulation(sys,prime_scheduler,
   X0,
% print_result);
%
% % second phase of simulation to reach to the next operating point
% % all stages are in dcm mode
% last_state = primesim_result(end,:);
% [vc_final1,~,~] = function6_sub_simulation(sys,last_state,
   il_final,
% print_result);
% end

%% First case with beta = 0.35
beta1 = 0.35;
prime_scheduler = function3_prime_scheduler((beta1.*(CF_norm).*...
                                        ton_safe),6*1e-3,sys);
primesim_result =function4_prime_simulation(sys,prime_scheduler,X0,
   ...
                                        print_result);

% second phase of simulation to reach to the next operating point
% all stages are in dcm mode
last_state = primesim_result(end,:);
[vc_final1,~,~] = function6_sub_simulation(sys,last_state,il_final,
   ...
                                        print_result);
```

```
%% second case with beta = 0.65
beta2 = 0.65;
prime_scheduler = function3_prime_scheduler((beta2.*(CF_norm).*...
                                        ton_safe),6*1e-3,sys);
primesim_result = function4_prime_simulation(sys,prime_scheduler,
    ...
                                        X0,print_result);

% second phase of simulation to reach to the next operating point
% all stages are in dcm mode
last_state = primesim_result(end,:);
[vc_final2,~,~] = function6_sub_simulation(sys,last_state,...
                                        il_final,print_result);

%% finding slope and y-intercept of line to find beta for given
    vfinal
slope = (vc_final2 - vc_final1)/(beta2-beta1);
c = vc_final1 - slope*beta1;

%% finding beta for given vfinal
beta = (vfinal - c)/slope;
end
%% ------------------------> END OF SCRIPT
   <----------------------------
```

A.6 GPU based implementation for MTC

```
/*****************************************************************
% This CUDA program implements minimum time operation for full
   range of beta.Range of beta should be bound by [0.1,0.9].Purpose
    of the CUDA enabled program is to accerate simulation for full
   range and compare the result with the single core execution.
*****************************************************************/
/*****************************************************************
% Created by : Shishir Patel (sjpatel2@mtu.edu)
% Created on : 02/06/2017
*****************************************************************/

#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <cuda.h>


/*****************************************************************
----------->Constant definition BEGINS<-----------
*****************************************************************/

// Constant definitions used for kernel
#define numblk          1               //number of blocks
#define numtpblk        64      //number of threads per block

// Constant definitions used for simulation parameters
#define beta1           0.1             //min value of beta
#define beta2           0.9             //max value of beta
#define numbeta         2048    //number of beta cases(within
   specified range) to simulate
#define tstart          0.0             //start time for each
   simulation case
#define tend            0.008   //end time for each simulation case
#define Ts                      0.00001 //sample time/update time
   for the simulation
#define n                       800             //total number of
   time steps in a simulation
```

```
/*****************************************************************
---------------------->Constant definition ENDS<----------------------
*****************************************************************/


/*****************************************************************
---------------------------->GPU KERNEL1 BEGINS<----------------------
*****************************************************************/

__global__ void simulate(double *switchtime1, double *switchtime2,
    double *switchtime3, double *Vcfinal)
{
        int tid = blockIdx.x*blockDim.x + threadIdx.x;   //
    assigning thread index
        //printf("%d",tid);
        if (tid < numbeta)
        {
                //intial discrete states of system
                double B[3] = {1.0,1.0,1.0};
                double A[3] = {1.0,1.0,1.0};


                //intial and final states of the system stored in
    local registers
                double Vs[3]            =
    {72.0000,65.0000,58.0000};
                double il_temp[3]       = {3.7125,4.1123,4.6086};
                double vc_temp          = 90.0000;
                double ilf[3]           = {10.3130, 11.4230,
    12.8020};
                double c1[3]            = {73.3333, 68.7500,
    110.0000};
                double c2[3]            = {333.3333, 312.5000,
    500.0000};
                double c3[3]            = {73.3333, 68.75000,
    110.0000};
                double c4[3]            = {333.3333, 312.5000,
    500.0000};
```

```
                //starting the simulation loop
                double il[3], vc;
                for (int i = 0; i<n; i++)
                {
                        //calculating il based on A and B
                        for (int a = 0; a < 3; a++)
                        {
                                if ( B[a] == 1.0 )
                                {
                                        if ( A[a] == 1.0 )
                                        {
                                                il[a] = il_temp[a]
+ Ts*c1[a]*il_temp[a] + Ts*c4[a]*Vs[a];
                                        }
                                        if ( A[a] == 0.0 )
                                        {
                                                il[a] = il_temp[a]
- Ts*c2[a]*vc_temp - Ts*c3[a]*il_temp[a] + Ts*c4[a]*Vs[a];
                                        }
                                }
                                else
                                {
                                        il[a] = il_temp[a];
                                }
                        }

                        //calculating Vc based on A and B
                        double C[3];
                        for (int k = 0; k<3; k++)
                        {
                                if(A[k] == 1)
                                {
                                        C[k] = 0;
                                }
                                else
                                {
                                        C[k] = 1;
                                }
                        }
                        vc = vc_temp + Ts*666.666*(il_temp[0]*C[0]*
```

```
B[0] + il_temp[1]*C[1]*B[1] + il_temp[2]*C[2]*B[2] - vc_temp);


                    //updating discrete states for next time
instance
                    if (Ts*i > switchtime1[tid])
                    {
                            A[0] = 0.0;
                    }
                    if (Ts*i > switchtime2[tid])
                    {
                            A[1] = 0.0;
                    }
                    if (Ts*i > switchtime3[tid])
                    {
                            A[2] = 0.0;
                    }

                    if (il[0] > ilf[0])
                    {
                            B[0] = 0.0;
                    }
                    if (il[1] > ilf[1])
                    {
                            B[1] = 0.0;
                    }
                    if (il[2] > ilf[2])
                    {
                            B[2] = 0.0;
                    }


                    //updating states for next time instance
                    il_temp[0] = il[0];
                    il_temp[1] = il[1];
                    il_temp[2] = il[2];
                    vc_temp  = vc;

            }
```

```
                //copy last vc value into Vc_fina;
                Vcfinal[tid] = vc_temp;
    }

}

/*****************************************************************
------------------------->GPU KERNEL1 ENDS<-------------------------
*****************************************************************/



/*****************************************************************
------------------------->HOST MAIN BEGINS<-------------------------
*****************************************************************/

int main( int argc, char *argv [] )
{
    int count;

        printf ("This program, \"%s\", was called with following
   arguments.\n",argv[0]);

    if (argc > 1)
        {
          for (count = 1; count < argc; count++)
                {
                  printf("argv[%d] = %s\n", count, argv[count]);
                }
        }
        else
    {
      printf("The command had no other arguments.\n");
    }


        int print_results;
        if (strcmp(argv[1], "P") == 0) //print results if 1st
   argument P is given
        {
```

```
            print_results = 1;
    }
    else
    {
            print_results = 0;
    }
    printf("print_results = %d\n",print_results);

    //system intial conditions; need to transfer to constant
memory
    double ton_max[3]       = {0.0026192, 0.0030852,
0.0021525};                   //maximum switchtime per stage

    double beta[numbeta], switchtime1[numbeta], switchtime2[
numbeta], switchtime3[numbeta];
    double Vcfinal[numbeta]; //results from GPU will be
transfered to this variable

    //generate beta sqquence
    double betatemp = beta1;
    for (int i=0; i<numbeta; i++)
    {
            //generate beta sequence
            beta[i] = betatemp + (beta2-beta1)/numbeta;
            betatemp = beta[i];

            //generate switchtime from beta and ton_max
            switchtime1[i] = beta[i] * ton_max[0];
            switchtime2[i] = beta[i] * ton_max[1];
            switchtime3[i] = beta[i] * ton_max[2];

            //Uncomment if you want to check values of beta and
 switchtime
            /*
            if (print_results == 1)
            {
                    printf("\nbeta[%d]        : %f\n",numbeta,
beta[i]);
                    printf("Switchtime[%d]  : %f\t %f\t %f\t \n
", numbeta, switchtime1[i], switchtime2[i], switchtime3[i]);
```

```
            }
            */
    }

    //defining cuda variables
    double *dev_switchtime1, *dev_switchtime2, *dev_switchtime3
;
    double *dev_Vcfinal; //final voltage which is output of
this cuda program

    // generate timer event
 cudaEvent_t start,stop;
 cudaEventCreate(&start);
 cudaEventCreate(&stop);

    // allocate the memory on the GPU
 cudaMalloc( (void**)&dev_switchtime1, numbeta * sizeof(double)
) ;
 cudaMalloc( (void**)&dev_switchtime2, numbeta * sizeof(double)
) ;
 cudaMalloc( (void**)&dev_switchtime3, numbeta * sizeof(double)
) ;
 cudaMalloc( (void**)&dev_Vcfinal, numbeta * sizeof(double) ) ;


 // copy the beta and switchtime arrays to the GPU memory
 cudaMemcpy( dev_switchtime1, switchtime1, numbeta * sizeof(
double), cudaMemcpyHostToDevice );
    cudaMemcpy( dev_switchtime2, switchtime2, numbeta * sizeof(
double), cudaMemcpyHostToDevice );
    cudaMemcpy( dev_switchtime3, switchtime3, numbeta * sizeof(
double), cudaMemcpyHostToDevice );

 // Trigger event 'start'
 cudaEventRecord(start, 0);

 //launch the kernal
 simulate<<<numbeta/numtpblk,numtpblk>>>(dev_switchtime1,
dev_switchtime2, dev_switchtime3, dev_Vcfinal);
```

```
    // Trigger Stop event
    cudaEventRecord(stop, 0);

    // Sync events
    cudaEventSynchronize(stop);

    // Calculate runtime, write to elapsedTime
    float elapsedTime; // Initialize elapsedTime;
    cudaEventElapsedTime(&elapsedTime, start, stop);
    //-- cudaEventElapsedTime returns value in milliseconds.
   Resolution ~0.5ms

    // Destroy CUDA Event API Events
    cudaEventDestroy(start);
    cudaEventDestroy(stop);

    // copy the array 'Vcfinal' back from the GPU to the CPU
    cudaMemcpy( Vcfinal, dev_Vcfinal, numbeta * sizeof(double),
   cudaMemcpyDeviceToHost );
        for (int i=0; i<numbeta; i++)
        {
                printf("Vcfinal(@beta = %f) : %f\n",beta[i],
   Vcfinal[i]);
        }

        // Print Elapsed time
    printf("\n--> Execution Time of Kernel: %f (ms) \n",
   elapsedTime);

    // free the memory allocated on the GPU
    cudaFree( dev_switchtime1 );
    cudaFree( dev_switchtime2 );
    cudaFree( dev_switchtime3 );
    cudaFree( Vcfinal );

        return 0;
}
/*********************************************************************
------------------------------>HOST MAIN ENDS<-------------------------------
*********************************************************************/
```