Fall 2010

Modeling the practical performance of switched-capacitor converters and a method for automating state-space model generation

Jordan M. Henry

Department:

MODELING THE PRACTICAL PERFORMANCE OF SWITCHED-CAPACITOR CONVERTERS AND A METHOD FOR AUTOMATING STATE-SPACE MODEL GENERATION

by

JORDAN MICHAEL HENRY

A THESIS

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

2010

Approved by

Jonathan W. Kimball, Advisor
Norman R. Cox
Mehdi Ferdowsi

ABSTRACT

A new modeling technique and a method for automating the modeling process are introduced for analyzing complex switched-capacitor (SC) converters. The model uses conventional circuit analysis methods to derive state-space models of each switching state. Steady-state performance is derived and expressed as an equivalent resistance. Whereas previous techniques have provided either the detailed performance of a simple SC converter or the limiting performance of a complex SC converter, this new model is flexible enough to provide detailed performance for any practical converter. Nonuniform component choices, asymmetric duty cycles, and other deviations from an ideal converter can be readily included. Dynamics can also be analyzed. Iterative methods of design based on this model would require the formulation of many equations, which is time consuming if done manually. Therefore, an algorithm is introduced to automatically generate the equations required for this state-space based modeling. The state equations are generated algorithmically given a standard node incidence matrix generated from a user-defined netlist. The algorithm enables a designer to quickly iterate SC converter design solutions based on its predicted performance. The model and algorithm have been validated through simulation techniques and experimental data collected from laboratory testing.

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Jonathan Kimball for his direction and guidance that has allowed me to conduct this research. Dr. Kimball's vast knowledge in the field of electrical engineering has significantly improved my skills as a student in engineering and has given me new interest in the field of power electronics.

I would also like to acknowledge the members of my committee, Dr. Norman Cox and Dr. Mehdi Ferdowsi, for their interest in my research topic and their instruction in solar energy and advanced power electronics.

I would especially like to thank my family and friends, without whom I couldn't have hoped to reach this level of academic achievement. To my mother who, despite the challenges life has thrown her, has given me so much encouragement throughout my life. Her tenacity and strength has always inspired me and I am grateful for the many sacrifices she has made for me. Thank you mom.

Finally, thank you to the National Science Foundation and the University of Missouri research board for their financial support. This project was funded in part by NSF grant ECCS-0900940.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

NOMENCLATURE

Symbol	Description
SC	Switched-capacitor
R_{eq}	Converter output impedance
ESR	Capacitor equivalent series resistance
KVL	Kirchoff's voltage law
KCL	Kirchoff's current law
MNA	Modified nodal analysis
STF	Sparse tableau formulation
p	Number of capacitors
$\mathbf{v}$	Capacitor voltage vector
$\mathbf{i}$	Capacitor current vector
M	Static gain of converter
$\mathbf{C}$	Diagonal matrix of capacitor values
$\dot{\mathbf{v}}$	Time derivative of capacitor voltage vector
$\mathbf{u}$	Input and output source vector
$\mathbf{E}$	Capacitor current coefficient matrix
$\mathbf{F}$	Capacitor voltage coefficient matrix
$\mathbf{G}$	Input and output source coefficient matrix
$\mathbf{A}$	State matrix for state space representation
$\mathbf{B}$	Input coefficient for state space representation
T	Switching period
D	Duty ratio of switching signal
$\mathbf{x}$	Capacitor voltage matrix used in difference equations
$\mathbf{\Phi}$	State matrix used for difference equations
$\mathbf{\Gamma}$	Input coefficient matrix notation for difference equations
q	Charge delivered by capacitor
SSL	Slow switching limit
FSL	Fast switching limit

PCB	Printed circuit board
n	Number of nodes
b	Number of branches
$\mathbf{N}$	Netlist matrix
$\tilde{\mathbf{N}}$	Partitioned netlist matrix containing only source and capacitor branches
$\hat{\mathbf{N}}$	Partitioned netlist matrix containing only parasitic branches
$\mathbf{A_a}$	Node incidence matrix
$\mathbf{i_{br}}$	Branch current vector
$\tilde{\mathbf{A}}_\mathbf{a}$	Node incidence matrix written in its canonical form
$\hat{\mathbf{A}}$	Partitioned Aa matrix containing KCL relationships
$\tilde{\mathbf{A}}$	Matrix formed after deleting null row of Aa
$\mathbf{i_x}$	Independent branch current vector
$\mathbf{i_y}$	Dependent branch current vector
$\mathbf{v_{br}}$	Branch voltage vector
$\mathbf{B_b}$	Basic loop matrix
$\mathbf{N}^*$	Reordered netlist matrix
$\hat{\mathbf{G}}$	Matrix fromed after deleting null rows from G matrix
$\hat{\mathbf{F}}$	Matrix fromed after deleting null rows from F matrix
$\hat{\mathbf{B}}$	Parasitic loop matrix
$\hat{\mathbf{E}}$	Capacitor current coefficient matrix for parasitic branches
$\mathbf{K}$	Capacitor KCL matrix
$\mathbf{i_{cap}}$	Branch capacitor current matrix
$\mathbf{E}^*$	Incomplete capacitor current coefficient matrix

1. INTRODUCTION

1.1. MOTIVATION

Traditionally, switched-capacitor (SC) converters have been used to provide simple, unregulated power conversion at lower power levels [1]. Recent developments in capacitor and semiconductor technology have made SC converters more practical in higher power applications [2-4]. Furthermore, development of sophisticated control strategies has also added voltage regulation capabilities to SC converters [5]. These advancements have contributed to the increasing popularity of SC converters, both in integrated form [6-8] and in discrete circuits [9-10]. As the popularity of SC converters continues to rise, so does the need for practical analysis techniques to facilitate converter design.

As the primary performance metric of an SC converter, the output impedance is an important design parameter. The output impedance, R_{eq}, aggregates all losses in parasitic resistances and determines the voltage drop on the output terminal based on the load current. To ensure efficiency and output voltage regulation specifications are met, the design of practical SC converters frequently relies on accurate modeling of the output resistance. This work develops a new method of modeling the steady-state output impedance of an SC converter and provides an algorithm for automating the modeling process.

1.1.1. Modeling the Output Impedance. First, a new analysis technique is introduced for modeling the output impedance of switched-capacitor converters. As depicted in Fig. 1.1, a typical steady-state model of an SC converter is an ideal transformer, with a rational turns ratio governed by the topology, followed by an equivalent resistance. The equivalent resistance aggregates all the losses in parasitic resistances, such as MOSFET on-state resistance and capacitor equivalent series resistance (ESR). Other losses, such as gating power, are usually tallied separately.

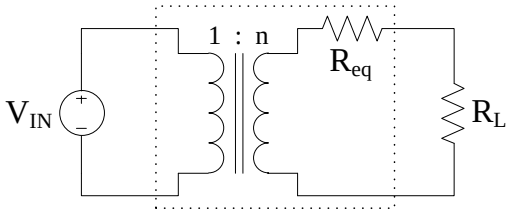

Figure 1.1. Typical steady-state model of an SC converter.

One existing approach to modeling the equivalent resistance analyzes an individual SC cell [11-12]. It requires development and solution of differential equations for each switching mode. It then imposes boundary conditions such that the converter operates in periodic steady-state. The resultant charge delivered to or from the capacitor divided by the switching cycle time equals the average current, which can be used to determine equivalent resistance. The work described in [12] includes asymmetric duty ratio and unequal resistance values in its analysis. That method remains useful for simple voltage doublers or other simple circuits.

Another approach directly analyzes charge flows in a complex SC converter [13-15]. For each switching mode, by inspection, it derives charge flow vectors for the capacitors and switches. These equations assume only a single input and a single output, where the charge flow in each capacitor and switch is expressed as the output charge flow multiplied by a constant vector, denoted a_c^j and a_r^j, respectively.

$$\begin{aligned} q_c^j &= a_c^j q_{out} = a_c^j \frac{I_{out}}{f_{sw}} \\ q_r^j &= a_r^j q_{out} = a_r^j \frac{I_{out}}{f_{sw}} \end{aligned} \tag{1}$$

The "a" vectors map the output charge flow q_{out}, which is equal to output current I_{out} divided by switching frequency f_{sw}, onto the charge flow of each capacitor. Using these vectors, the method derives fast switching and slow switching limits (FSL and SSL,

respectively) assuming 50% duty ratio switching. This method is useful for complex SC converters of regular structure operating at very low or very high frequencies. For converters with unusual structure, and particularly for converters that operate at practical intermediate frequencies, this method does not provide results directly.

An additional method expresses the converter losses as a function of the currents passing through each switching (flying) capacitor [16]. Energy loss is calculated for two switching modes separately (2), and the losses are summed to express the total as a function of the average current through each capacitor, where the average capacitor current is proportional to the output current (3).

$$E_j = \frac{\Delta V_j^2 \cdot C}{2}(1 - e^{-2\beta_j}) \tag{2}$$

$$E_R = E_1 + E_2 = \left(\frac{I_{C_{AV}}}{f_s C}\right)^2 \cdot \frac{C}{2} \cdot \left[\frac{\left(1+e^{-\beta_1}\right)}{\left(1-e^{-\beta_1}\right)} + \frac{\left(1+e^{-\beta_2}\right)}{\left(1-e^{-\beta_2}\right)}\right] \tag{3}$$

In the above equations, β_j is equal to $\frac{t_j}{RC}$, $I_{C_{AV}}$ represents the average current in each capacitor, f_s is the switching frequency, and j represents the switching mode. Total power loss P_T is derived from the total energy loss and expressed as an equivalent resistance:

$$P_T = \left(I_{C_{AV}}\right)^2 \cdot \left\{\frac{1}{2 f_s C}\left[\frac{\left(1+e^{-\beta_1}\right)}{\left(1-e^{-\beta_1}\right)} + \frac{\left(1+e^{-\beta_2}\right)}{\left(1-e^{-\beta_2}\right)}\right]\right\} \tag{4}$$

This method is useful for simple hard and soft switched two-mode converters.

The present work proposes a new method that resembles that in [11-12]. Rather than analyzing a single cell, however, it analyzes a complete converter using conventional circuit analysis methods. Kirchoff's voltage and current laws are applied

and as in [11-12], the differential equations for each switching mode are solved, and periodic steady-state assumptions are invoked. Although generic symbolic results are not possible, numerical results can be determined for a specific converter. Rather than just providing performance limits of an SC converter [17], this method also allows the dynamics of a converter to be analyzed. A typical design flow for an SC converter may require, first, the method in [13-15] to form the basic design, then the method proposed here to provide a detailed analysis, with iterations sufficient to develop a suitable design. This new modeling technique is validated by comparing simulation and experimental data with that of the projected model. It also describes the experimental procedures used and discusses the conclusions supported by this work.

1.1.2. State Model Generator. The proposed model relies on the development of state equations derived from KVL and KCL equations. Deriving these state equations manually can be very time consuming, especially for converters with large gains (i.e., many stages). An algorithm that can automatically generate the state equations would enable a designer to quickly iterate solutions for SC converter designs.

Previous work developed an automated state model generator to generate KVL and KCL equations needed for state-space analysis of switching converters [18]. That approach involved construction of a node incidence matrix used to establish the required independent KCL relationships. In matrix form, the KCL equations yielded the basic loop matrix of the circuit, which, along with proper representation of a branch's volt-ampere (VI) characteristics, was the basis for the generation of the state model. This method is useful for linearization and eigensystem analysis, but difficult to integrate with the model proposed here because the difference equations are not implemented at the individual branch level.

Another well known method of solving electrical networks algorithmically is modified nodal analysis (MNA) [19-20]. An extension of nodal analysis, MNA was developed to mitigate the difficulty of representing voltage-defined components (e.g., voltage sources) whose conductances are infinite and currents are unknown. MNA generates equations on a node-by-node basis by determining not only node voltages, but also voltage source currents. The equations are expressed in matrix form by augmenting the node voltage equations by the current equations for the voltage-defined elements.

They are solved using Gaussian elimination and LU factorization to find the solution of a linear system of simultaneous equations. In this method, the unknown variables are node voltages, voltage source currents, output currents, and controlling source currents. The MNA formulation is general and easy to implement on a computer. It yields relatively compact systems of equations, making its use popular in SPICE programs. MNA is effective for solving a circuit's VI characteristics numerically, but falls short for state equation generation.

Predating MNA, sparse tableau formulation (STF) is another approach to network analysis [21]. The unknowns for STF include node voltages, branch currents, and branch voltages. Unlike MNA, STF involves no special treatment of voltage sources or any other elements. The matrix is formed by augmenting three types of equations: KCL equations written in terms of branch currents for each node, KVL equations relating a branch voltage to its node voltages, and branch constitutive equations written for each branch in terms of its branch voltage and current. Generally, the STF matrix is larger than MNA matrices, but it is more sparse; making it easier to solve by Gaussian elimination. Although STF generates more equations per system than MNA, it includes fewer nonzero terms per equation and consequently, fewer mathematical operations are required to solve those equations. Efficient implementation of this method, however, requires sophisticated programming techniques and data structures.

Here, an algorithmic method to develop the state equations for complex SC converters is developed. The method resembles that described in [18]; however, state equations are generated at the individual branch level to ensure that capacitor voltages are correctly represented as state variables in the matrix form compatible with the proposed model. A node incidence matrix is generated for each switching state from user-defined netlists. Loop matrices are derived to find KVL relationships and used directly to construct the coefficient matrices for the capacitor voltages and sources used in the model. Branch currents are expressed as capacitor currents and compiled in matrix form to complete the model. This paper describes the computer implementation of this algorithm and presents algorithm simulations for multiple SC converter topologies to

illustrate its capabilities. The results of the algorithm simulations have been verified by Matlab[1] simulations and experimental data collected through laboratory testing.

1.1.3. Document Organization. The subject matter presented here is organized by first introducing the state modeling technique in Section 2 followed by the development of the automated state model generator in Section 3. Conclusions, including the summary of results and future project extensions, are explained in Section 4. Following the conclusions, Appendix A contains schematics and board layouts for the printed circuit boards used for collecting the experimental data and Appendix B includes the computer code written for the state model and the automated state model generator algorithm.

The content in Section 2 has been accepted for publication by IEEE Transactions on Power Electronics and is currently in press.

[1] Matlab is a registered trademark of The MathWorks, Inc.

2. PRACTICAL PERFORMANCE OF COMPLEX SWITCHED-CAPACITOR CONVERTERS

2.1. MODEL DERIVATION

The derivation of the model may be illustrated with a generic SC converter with two switching modes. The converter contains p capacitors, whose voltages are composed into a vector, **v**, and whose currents are composed into a vector, **i**. The capacitors and switches are arranged to provide a static gain M. The value of each capacitor is arranged in a diagonal matrix, **C**, where C_{jj} is the capacitance of capacitor j. Based on the definition of a capacitor,

$$\mathbf{i} = \mathbf{C}\dot{\mathbf{v}} \tag{5}$$

On the input and output ports are voltage sources V_{in} and V_{out}, respectively, which are composed into a vector $\mathbf{u} = \begin{bmatrix} V_{in} & V_{out} \end{bmatrix}^T$.

In the first switching mode, KVL and KCL can be applied to find p independent equations relating the capacitor voltages and currents, expressed in matrix form as

$$\mathbf{E}_1\mathbf{i} + \mathbf{F}_1\mathbf{v} + \mathbf{G}_1\mathbf{u} = \mathbf{0} \tag{6}$$

Each row of $\mathbf{E}_1$, $\mathbf{F}_1$, and $\mathbf{G}_1$ represent the application of either KVL or KCL. For KVL rows, entries in $\mathbf{E}_1$ are resistances and entries in $\mathbf{F}_1$ and $\mathbf{G}_1$ are ±1 or zero as voltage drops are summed around a loop. For KCL rows, entries in $\mathbf{E}_1$ are ±1 or zero, and entries in $\mathbf{F}_1$ and $\mathbf{G}_1$ are all zero as currents are summed at a node. If KVL and KCL have been applied correctly, then $\mathbf{E}_1$ is invertible. Solving for **i** yields

$$\mathbf{i} = -\mathbf{E}_1^{-1}\mathbf{F}_1\mathbf{v} - \mathbf{E}_1^{-1}\mathbf{G}_1\mathbf{u} \tag{7}$$

and substituting (5) gives

$$\dot{\mathbf{v}} = (-\mathbf{C}^{-1}\mathbf{E}_1^{-1}\mathbf{F}_1)\mathbf{v} + (-\mathbf{C}^{-1}\mathbf{E}_1^{-1}\mathbf{G}_1)\mathbf{u} \tag{8}$$

To simplify symbolic representation, matrices $\mathbf{A}_1$ and $\mathbf{B}_1$ are used to consolidate the coefficient vectors, resulting in

$$\begin{aligned} \dot{\mathbf{v}} &= \mathbf{A}_1\mathbf{v} + \mathbf{B}_1\mathbf{u} \\ \mathbf{A}_1 &= -\mathbf{C}^{-1}\mathbf{E}_1^{-1}\mathbf{F}_1 \\ \mathbf{B}_1 &= -\mathbf{C}^{-1}\mathbf{E}_1^{-1}\mathbf{G}_1 \end{aligned} \tag{9}$$

Similar analysis can find matrices $\mathbf{A}_2$ and $\mathbf{B}_2$ for the second switching mode. A variety of other circuit analysis techniques may also be used to find a model in the same form as (9).

In an SC converter, the switching modes alternate. The switches are in mode 1 for duration t_1 and mode 2 for duration t_2. Typically, t_1 is equal to D_1T and t_2 is equal to D_2T, where T is the switching period and D_1 and D_2 are duty ratios of the switching waveforms. Without loss of generality, the converter is assumed to switch to mode 1 at $t = 0$ and to mode 2 at $t = t_1$, and the cycle ends at $t = t_1 + t_2$. (Period T will be used later in the analysis.) If the capacitor voltages are identified as states, the vector notation can be modified so that **v** is equal to **x**. Thus, the state equations yield

$$\begin{aligned} \mathbf{x}(t_1) &= \mathbf{\Phi}_1\mathbf{x}(0) + \mathbf{\Gamma}_1\mathbf{u} \\ \mathbf{x}(t_1 + t_2) &= \mathbf{\Phi}_2\mathbf{x}(t_1) + \mathbf{\Gamma}_2\mathbf{u} \\ \mathbf{\Phi}_1 &= e^{\mathbf{A}_1 t_1} \\ \mathbf{\Phi}_2 &= e^{\mathbf{A}_2 t_2} \end{aligned} \tag{10}$$

To complete the model, the $\mathbf{\Gamma}$ matrices can be calculated as follows:

$$\begin{aligned} \mathbf{\Gamma}_1 &= \mathbf{A}_1^{-1}(e^{\mathbf{A}_1 t_1} - \mathbf{I})\mathbf{B}_1 \\ \mathbf{\Gamma}_2 &= \mathbf{A}_2^{-1}(e^{\mathbf{A}_2 t_2} - \mathbf{I})\mathbf{B}_2 \end{aligned} \tag{11}$$

Unfortunately, the symbolic formula involves matrix inversion of $\mathbf{A}_1$ and $\mathbf{A}_2$. In many cases, such as a ladder SC converter, one of the two switching modes yields a singular $\mathbf{A}$ matrix. Instead of the symbolic result, therefore, a numerical result is needed. The conventional algorithm, given in [22], is implemented in the Matlab function `c2d` (and similarly in other mathematical programs). For a given SC converter with known values and switching times, numerical values can be found for $\mathbf{\Phi}$ and $\mathbf{\Gamma}$. The complete sampled-data model, incorporating both switching modes and the sampling period T, is

$$\begin{aligned} \mathbf{x}\big((k+1)T\big) &= \mathbf{\Phi}\mathbf{x}\big(kT\big) + \mathbf{\Gamma}\mathbf{u}\big(kT\big) \\ \mathbf{\Phi} &= \mathbf{\Phi}_2\mathbf{\Phi}_1 \\ \mathbf{\Gamma} &= \mathbf{\Phi}_2\mathbf{\Gamma}_1 + \mathbf{\Gamma}_2 \end{aligned} \tag{12}$$

There are two uses for (12). First, this discrete-time model can be used to study the dynamic characteristics of the SC converter by placing voltage sources on the input and output terminals. For example, one might wish to determine how quickly voltages distribute among the capacitors. Second, steady-state conditions for (12) can be used to find the equivalent resistance of the converter. Recall that the key performance metric for a switched capacitor converter is this equivalent resistance.

At steady-state, $\mathbf{x}((k+1)T)$ is equal to $\mathbf{x}(kT)$. With this assumption, (12) can be solved for $\mathbf{x}_0$, the equilibrium value of $\mathbf{x}$ at the beginning of each cycle:

$$\mathbf{x}_0 = \big(\mathbf{I} - \mathbf{\Phi}\big)^{-1}\mathbf{\Gamma}\mathbf{u} \tag{13}$$

Here, $\mathbf{I}$ is the $p \times p$ identity matrix. Thus, one can determine the value of $\mathbf{x}$ at the midpoint of the cycle (i.e., when the switching mode changes from mode 1 to mode 2):

$$\mathbf{x}_1 = \mathbf{\Phi}_1\mathbf{x}_0 + \mathbf{\Gamma}_1\mathbf{u} \tag{14}$$

The designer identifies one capacitor, the i^{th} capacitor, that delivers all of the charge to the output. For example, in a converter with a ladder topology, the last switching (flying) capacitor would be chosen. The change in its voltage, multiplied by its capacitance, gives the charge it delivers, q. This charge divided by time is output current, i_{out}. As a result, the equivalent resistance of an SC converter with a static gain of *M* can then be easily derived as follows:

$$\begin{aligned} q &= C_{ii}\left(x_{1,i} - x_{0,i}\right) \\ i_{out} &= \frac{q}{T} \\ R_{eq} &= \frac{MV_{in} - V_{out}}{i_{out}} \end{aligned} \tag{15}$$

2.2. MODEL DEVELOPMENT AND SIMULATION FOR A FOUR-STAGE SC CONVERTER

To explore this new technique, an SC converter with a ladder topology was designed and tested. Figure 2.1 shows a simplified schematic. Each "switch" is actually two FDMS8460 MOSFETs in parallel, for an equivalent switch resistance of 3 mΩ (denoted as R_{sw} below). Each "capacitor" is actually eight TMK325BJ226MM-T ceramic capacitors from Nichicon (22 μF, 25 V), for a total equivalent series resistance of 10 mΩ (denoted as R_c below). For a four-stage converter, which has a gain of $M = 5$, KVL and KCL yield matrices (16) through (21) below. For other numbers of stages, the matrix structure is maintained; only the dimensions change.

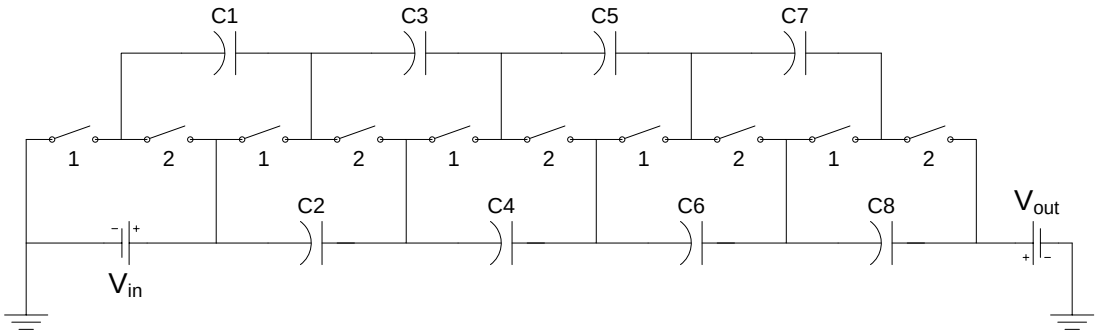

Figure 2.1. Four-stage ladder converter.

$$\mathbf{E}_1 = \begin{bmatrix} -(2R_{sw}+R_c) & 0 & R_{sw} & 0 & 0 & 0 & 0 & 0 \\ R_{sw} & R_c & -(2R_{sw}+R_c) & 0 & R_{sw} & 0 & 0 & 0 \\ 0 & 0 & R_{sw} & R_c & -(2R_{sw}+R_c) & 0 & R_{sw} & 0 \\ 0 & 0 & 0 & 0 & R_{sw} & R_c & -(2R_{sw}+R_c) & 0 \\ 0 & -R_c & 0 & -R_c & 0 & -R_c & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 \\ 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 \end{bmatrix} \quad (16)$$

$$\mathbf{F}_1 = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (17)$$

$$\mathbf{G}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -1 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{18}$$

$$\mathbf{E}_2 = \begin{bmatrix} -(2R_{sw}+R_c) & R_c & R_{sw} & 0 & 0 & 0 & 0 & 0 \\ R_{sw} & 0 & -(2R_{sw}+R_c) & R_c & R_{sw} & 0 & 0 & 0 \\ 0 & 0 & R_{sw} & 0 & -(2R_{sw}+R_c) & R_c & R_{sw} & 0 \\ 0 & 0 & 0 & 0 & R_{sw} & 0 & -(2R_{sw}+R_c) & R_c \\ 0 & -R_c & 0 & -R_c & 0 & -R_c & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{19}$$

$$\mathbf{F}_2 = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{20}$$

$$\mathbf{G}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -1 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{21}$$

The dimensions of the **E**, **F**, and **G** matrices are designated by the number of capacitors in the topology. This topology has eight capacitors that shuttle charge to the output; thus, the **E** and **F** matrix dimensions are 8×8, and the **G** matrices relating the input and output are 8×2. Symbolic computation of the corresponding $\mathbf{A_1}$, $\mathbf{B_1}$, $\mathbf{A_2}$, and $\mathbf{B_2}$ matrices results in several pages of output. However, numerical computation is straightforward. The $\mathbf{\Phi}$ and $\mathbf{\Gamma}$ matrices can be computed for a given switching frequency with the Matlab function `c2d`. Figure 2.2 shows the PLECS[2] schematic constructed for the four-stage ladder configuration. Simulation results for the equivalent resistance are compared to the analytical result in Fig. 2.3 for a switch duty cycle of 45%. The converter was simulated using PLECS, a blockset for Simulink, and it incorporated the same component parameters and switch duty cycle included in the model. As in the model, the voltage sources were placed on the input and output terminals of the converter. The input voltage was set at 5V and the output voltage at 24V.

[2] PLECS is a registered trademark of Plexim GmbH.

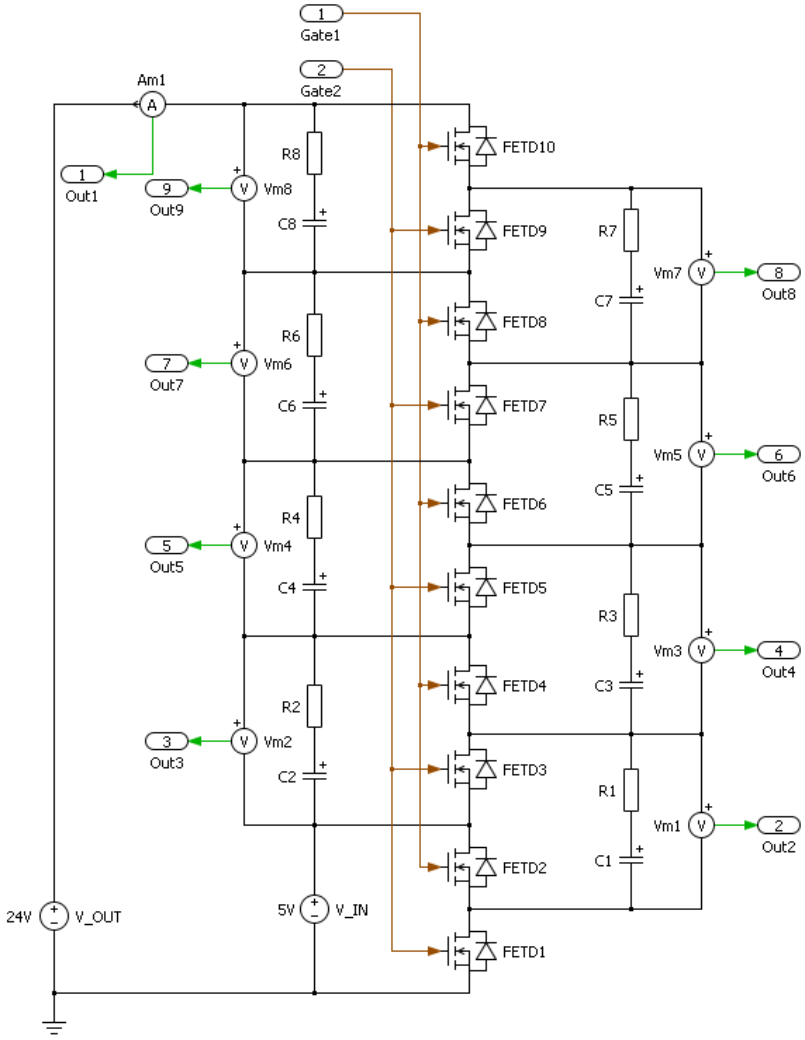

Figure 2.2. Simulation model for a four-stage ladder converter.

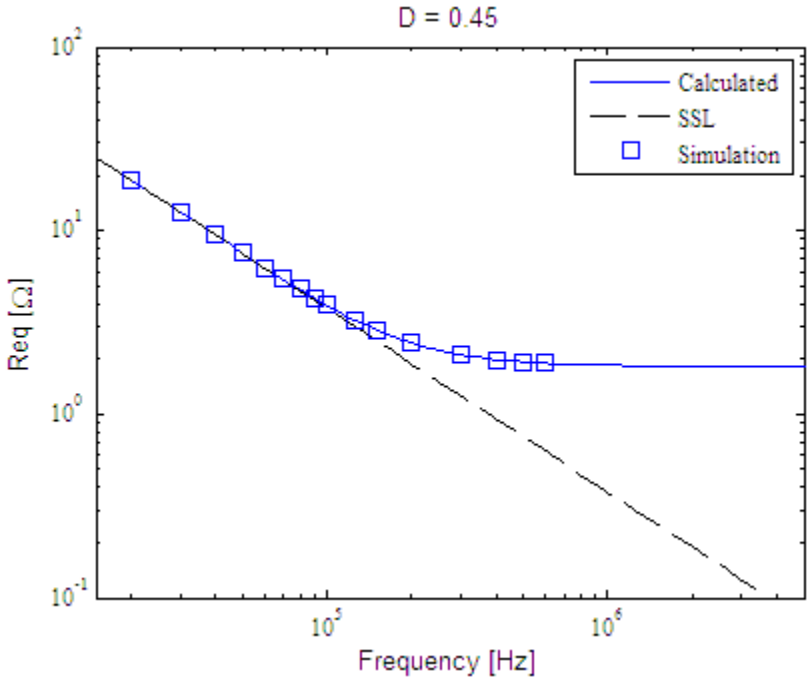

Figure 2.3. Model and simulation comparison of four-stage ladder converter shown in Figure 2.1.

As shown in Fig. 2.3 above, the equivalent resistance predicted by the model produced the expected results. At lower switching frequencies, the equivalent resistance is dominated by the impedance of the capacitors, following the slow switching limit (SSL). At higher switching frequencies, it is dominated by the resistance of the MOSFETs, following the fast switching limit (FSL). Identical simulation data confirms these results.

For the case of a single-stage ladder converter ($M = 2$), a voltage doubler with only one switching capacitor, this new method also agrees with the technique previously reported in [12].

2.3. EXPERIMENTAL PROCEDURE AND DATA

Experimental data was obtained through bench testing of the four-stage converter shown in Fig. 2.1. The experimental setup for bench testing is shown in Figures 2.4 and 2.5. The PCB used for testing was designed as a nine-stage ladder converter but was

assembled as a four-stage ladder converter. Once the ladder circuit was constructed, as shown in Fig. 2.6, two Fluke 8845A high-precision digital multimeters where placed on the input and output terminals to measure both voltage and current. A BK Precision 8502 electronic load placed on the output simulated a constant current load. At a given switching frequency, the load current was varied between 50 mA and 200 mA. The slope of output voltage versus current over the range tested revealed the equivalent resistance at that frequency. This procedure was repeated for switching frequencies between 20 kHz and 160 kHz due to switching restrictions explained below. The data was compared to the simulation and model data gathered previously. As expected, the results were consistent with both; they are in Fig. 2.7 below. The discrepancy at approximately 50 kHz is discussed below.

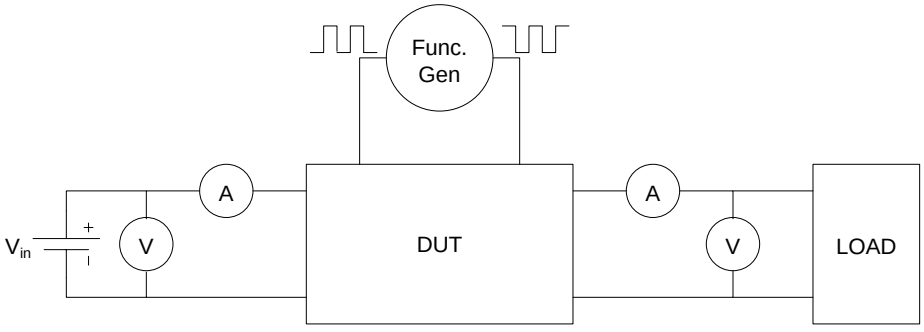

Figure 2.4. Schematic of experimental setup.

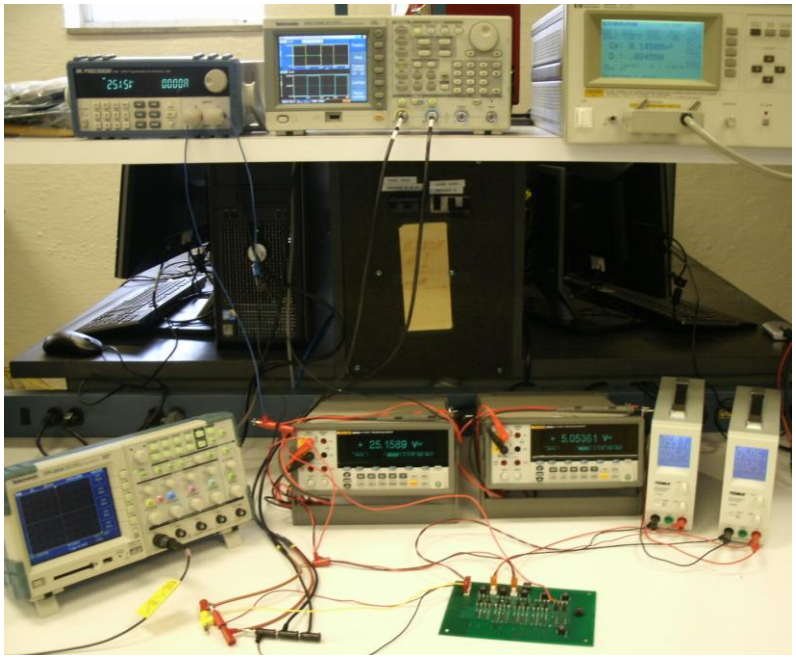

Figure 2.5. Physical bench setup for experimentation.

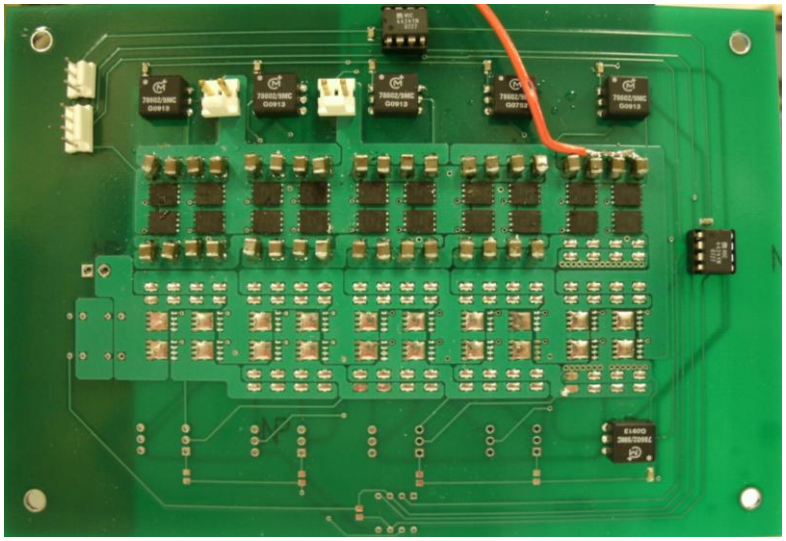

Figure 2.6. PCB assembled as four-stage ladder converter.

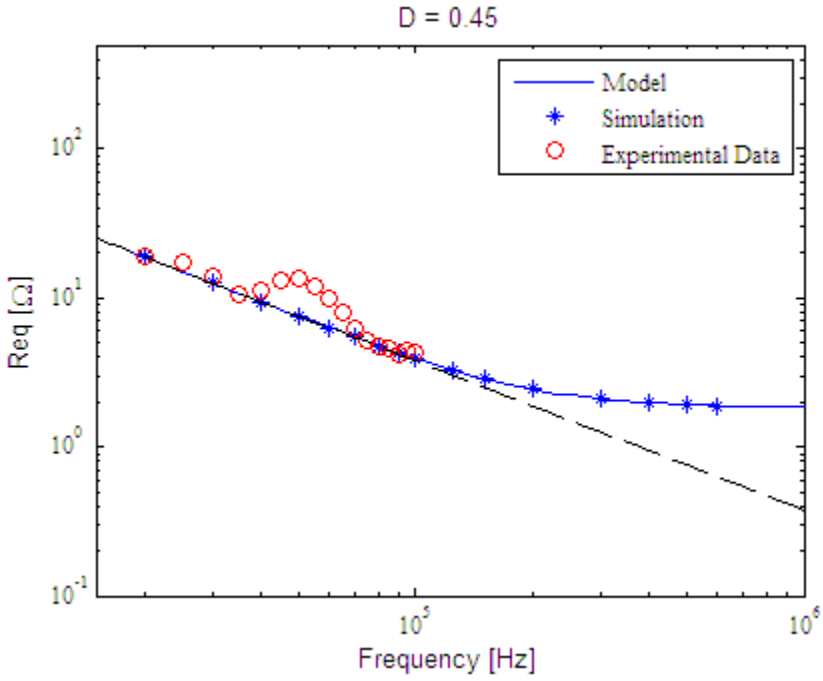

Figure 2.7. Model comparison of four-stage ladder converter.

Also included below are the expected and measured state variable waveforms of the last switching capacitor's voltage. This capacitor functions as the primary charge distributor to the load and, as seen in Figures 2.8 and 2.9, the magnitude of the capacitor voltage swing is similar to that observed in the simulation, although the shape is different due to parasitic inductances.

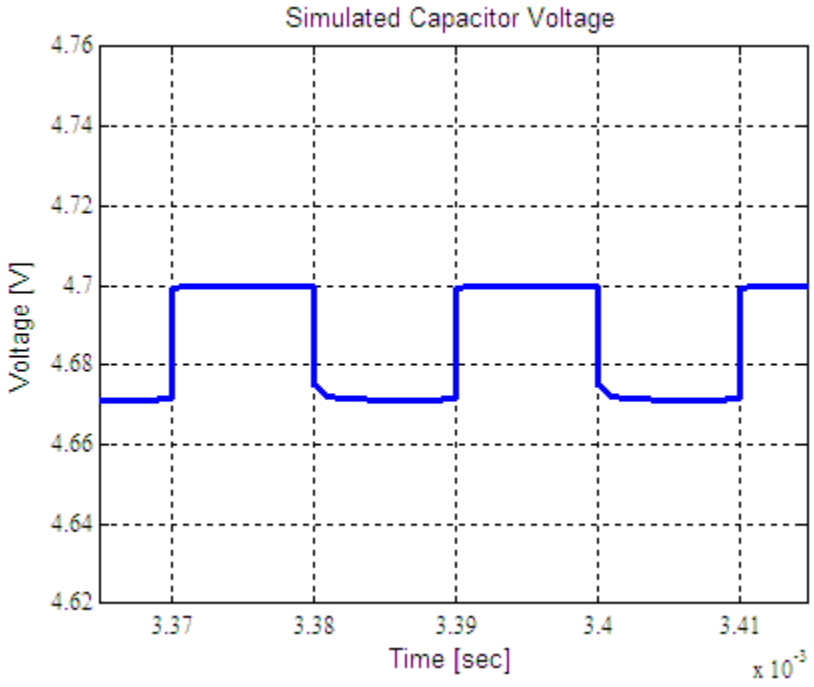

Figure 2.8. Expected voltage waveform of last switching capacitor.

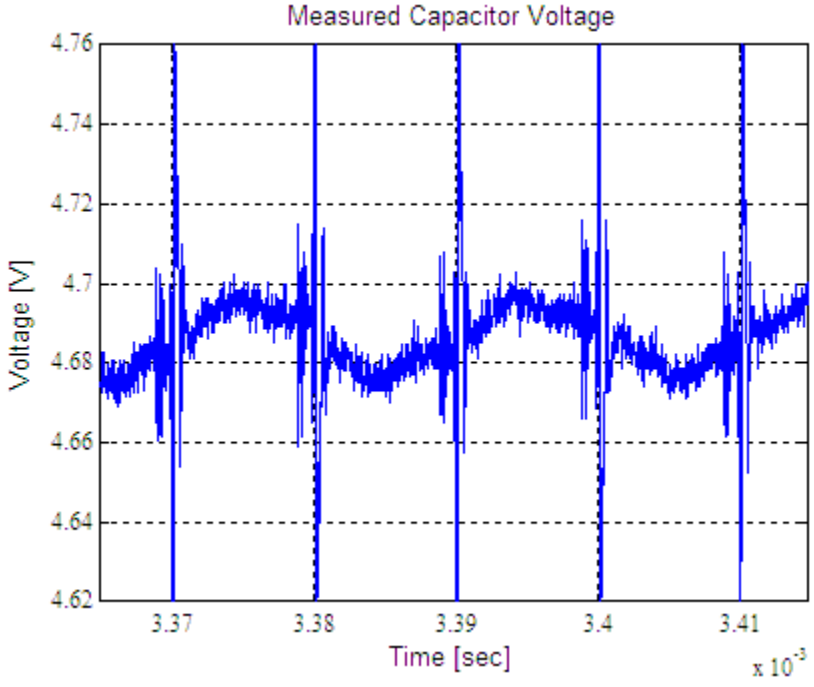

Figure 2.9. Measured voltage waveform of last switching capacitor.

Experimental data was also collected for a two-stage ladder converter, where $M = 3$, to ensure accuracy of the model. The same derivation procedure conducted for the four-stage converter yielded the following matrices for the two-stage converter:

$$\mathbf{E}_1 = \begin{bmatrix} -(2R_{sw} + R_c) & 0 & R_{sw} & 0 \\ R_{sw} & R_c & -(2R_{sw} + R_c) & 0 \\ 0 & -R_c & 0 & -R_c \\ 0 & 1 & 1 & -1 \end{bmatrix} \tag{22}$$

$$\mathbf{F}_1 = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{23}$$

$$\mathbf{G}_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -1 & 1 \\ 0 & 0 \end{bmatrix} \tag{24}$$

$$\mathbf{E}_2 = \begin{bmatrix} -(2R_{sw} + R_c) & R_c & R_{sw} & 0 \\ R_{sw} & 0 & -(2R_{sw} + R_c) & R_c \\ 0 & -R_c & 0 & -R_c \\ 1 & 1 & -1 & -1 \end{bmatrix} \tag{25}$$

$$\mathbf{F}_2 = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{26}$$

$$\mathbf{G}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -1 & 1 \\ 0 & 0 \end{bmatrix}. \tag{27}$$

Component values for this converter are identical to those of the four-stage converter, except that each "capacitor" was replaced with four rather than eight TMK325BJ226MM-T ceramic capacitors from Nichicon (22 μF, 25 V). The same nine-stage PCB was also used, but assembled as a two-stage converter. Figure 2.10 summarizes the resulting simulation and experimental data. Again, the data match the model prediction as expected. The discrepancy at approximately 70 kHz is discussed below.

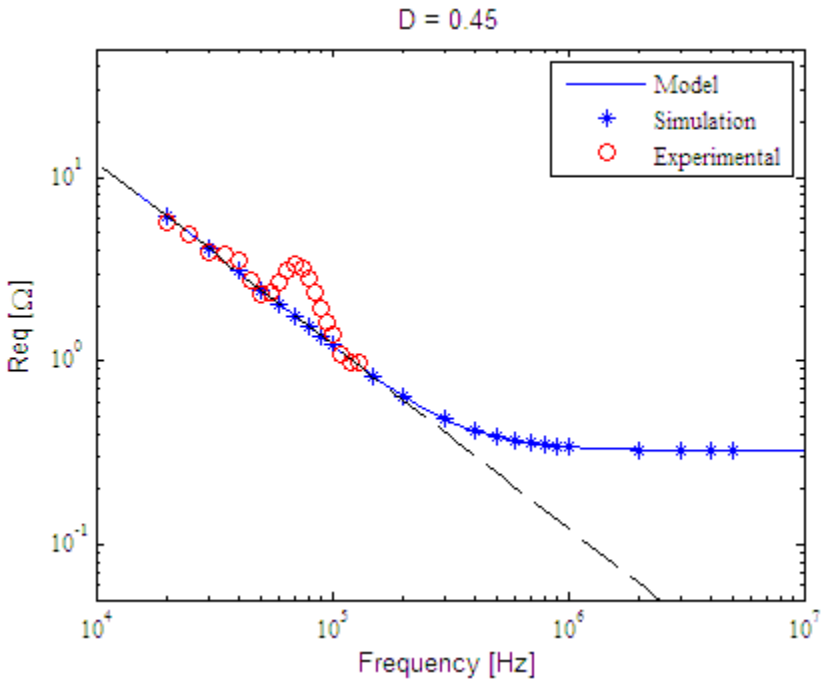

Figure 2.10. Model comparison of two-stage ladder converter.

2.4. ANALYSIS OF RESULTS

The results of both the experiment and the simulation verified the accuracy of the proposed model. Some practical effects must be considered when implementing the model. For instance, comparison of the experimental data with those of the two-stage converter model in Figure 2.10 reveals slight discrepancies at 35 kHz and 70 kHz, where the experimental data is slightly higher than those of the model. This discrepancy is explained by the inherent inductance of the PCB board due to its design. This inductance creates a resonant effect that interacts with the capacitors, increasing the resistance at harmonics of 35 kHz. This resonant effect becomes more apparent as the capacitance varies. Figure 2.11 shows the resonant frequency was shifted down by $1/\sqrt{2}$ when the capacitance is doubled, and up by $\sqrt{2}$ when the capacitance is halved. This same change is also apparent when comparing the experimental data for the two-stage and four-stage converters. The four-stage converter had double the capacitance of the two-stage converter; therefore, the resonant frequency of the four-stage converter shifted down by $\sqrt{2}$, from 75 kHz to 50 kHz. Also as expected, when the capacitance increased, equivalent resistance decreased, because the SSL curve was dominated by the capacitor impedance. The inherent board inductance causing the resonance was estimated as 30 nH.

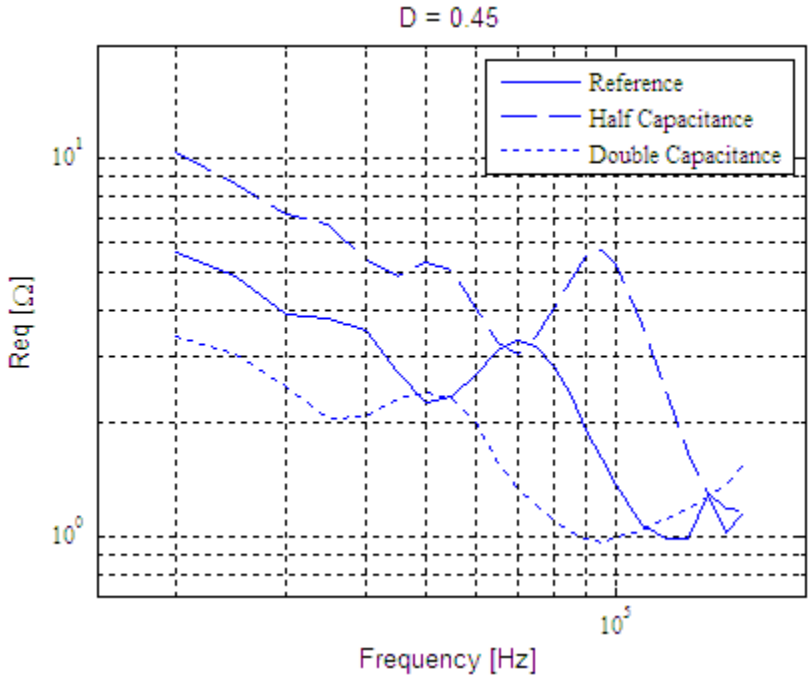

Figure 2.11. Capacitance effect on resonant frequency for two-stage ladder converter.

Experimental testing was also limited to switching frequencies of less than 160 kHz due to the limitations of the gate drivers selected to switch the MOSFETs. When switching at frequencies above 100 kHz, the switching waveforms were attenuated and had equivalent duty cycles of less than 45%, causing the resistance of the converter to increase sharply. This effect can be seen in the Figures 2.12 and 2.13 below, taken from a Tektronix TPS 2024 oscilloscope.

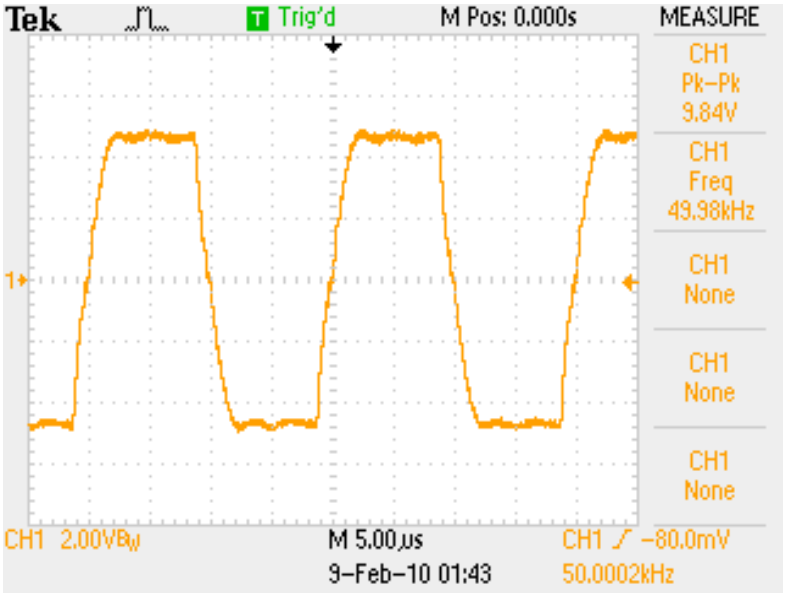

Figure 2.12. MOSFET gate switching waveform at 50 kHz.

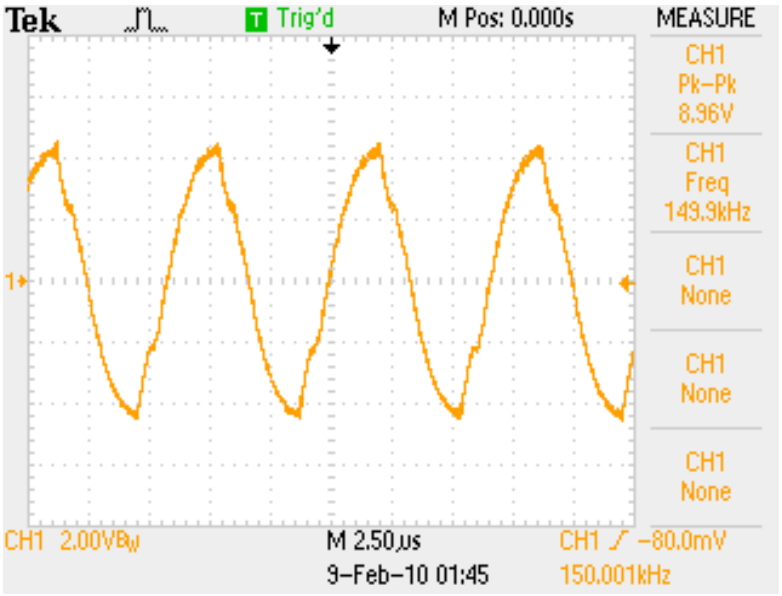

Figure 2.13. MOSFET gate switching waveform at 150 kHz.

The voltage coefficient of the capacitors was found to have a major impact on experimental converter performance. Initially, the experimental data corresponded to a higher resistance than expected based on the model and simulations. Often overlooked in power converter design is the variation of capacitance with the applied voltage [23]. This problem is specific to ceramic capacitors; it is worst for Z5U types and best for C0G (NPO) types. To achieve the desired capacitance, X7R types were needed. After closer inspection of the component data sheet [24], capacitance was found to decrease by up to 25% with an applied DC voltage of 5V, as shown in Figure 2.14. After also accounting for the 20% tolerance cited in the datasheet [15] (confirmed as 15% with an HP4284A precision LCR meter), the actual working capacitance during operation was found to be less than 60% of its nominal value. Once these corrections were included in the model, the experimental and simulation data matched closely. Figure 2.15 shows the impact of the capacitor's voltage coefficient on the equivalent resistance of the converter.

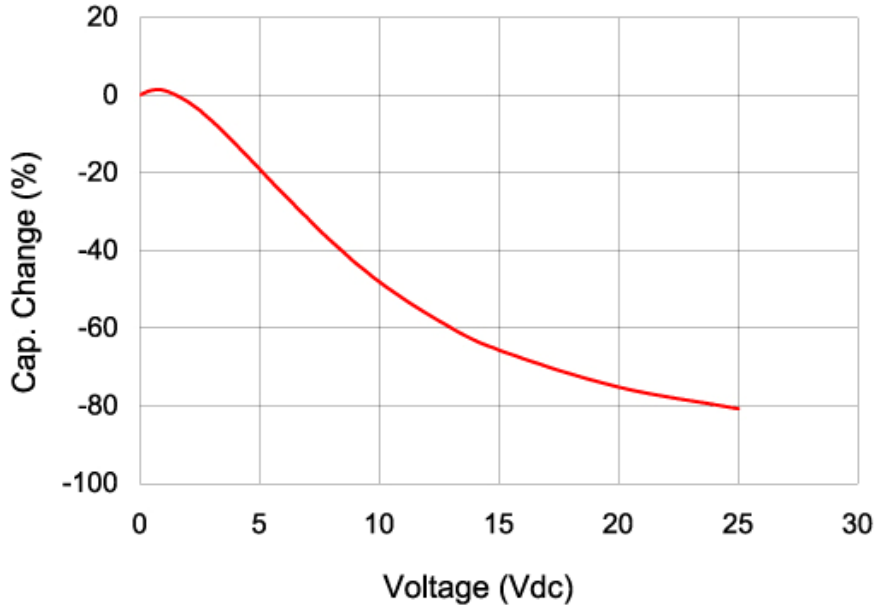

Figure 2.14. DC voltage characteristics of TMK 325BJ226MM-T ceramic capacitors [24].

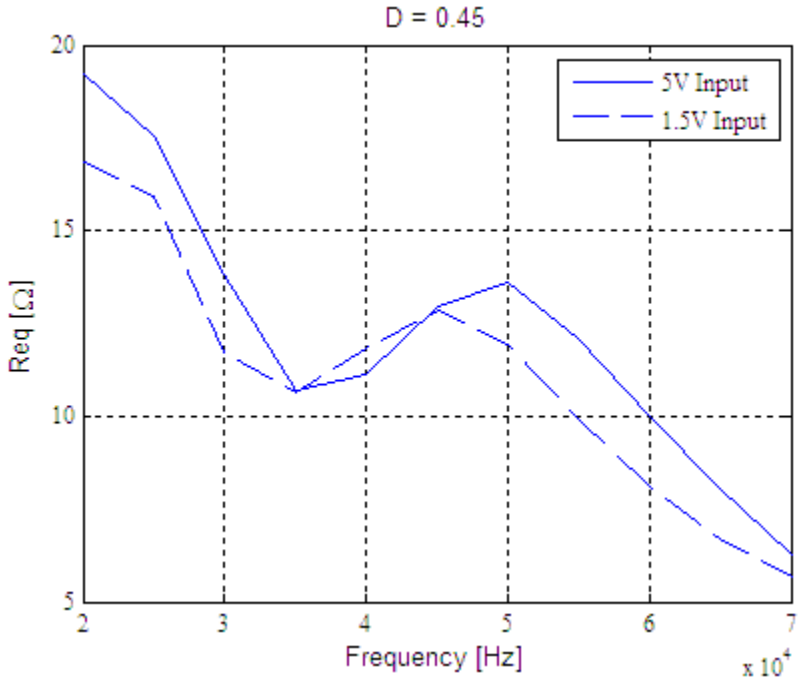

Figure 2.15. Equivalent resistance variation due to capacitor voltage coefficient.

2.5. EXTENSION OF MODEL TO OTHER SC CONVERTER TOPOLOGIES AND MODES

To demonstrate the model's flexibility, it was applied to another topology, the Fibonacci SC converter in Figure 2.16. The input voltage V_{in} was arbitrarily set at 5V and the topology gain, based on the Fibonacci sequence, was 5. The model was generated by again applying KVL and KCL for each switching mode, resulting in matrices (28) to (33).

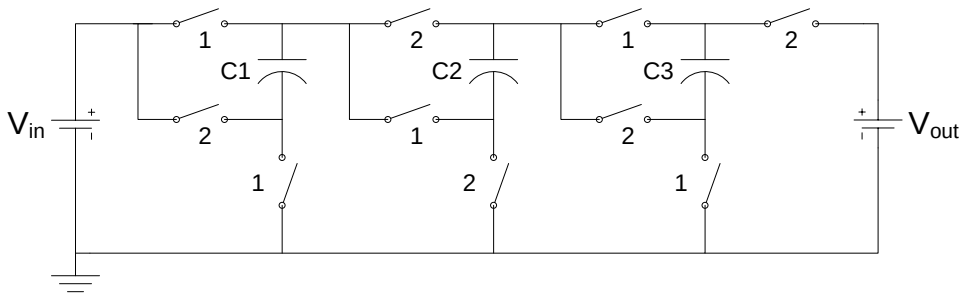

Figure 2.16. Fibonacci SC converter with $M = 5$.

$$\mathbf{E}_1 = \begin{bmatrix} -(2R_{sw} + R_c) & R_{sw} & 0 \\ (R_{sw} + R_c) & 0 & -(3R_{sw} + 2R_c) \\ 0 & 1 & 1 \end{bmatrix} \tag{28}$$

$$\mathbf{F}_1 = \begin{bmatrix} -1 & 0 & 0 \\ 1 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix} \tag{29}$$

$$\mathbf{G}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{30}$$

$$\mathbf{E}_2 = \begin{bmatrix} (2R_{sw} + R_c) & -(R_{sw} + R_c) & 0 \\ 0 & (R_{sw} + R_c) & (2R_{sw} + R_c) \\ 1 & 1 & -1 \end{bmatrix} \tag{31}$$

$$\mathbf{F}_2 = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{32}$$

$$\mathbf{G}_2 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \\ 0 & 0 \end{bmatrix} \tag{33}$$

The same component values in the two-stage ladder converter were also used in the model and simulation for this converter. For switching waveforms with a duty cycle of 45%, the model and simulation results are shown in Figure 2.17. The identical model and simulation data show the model's versatility in its application to various converter topologies.

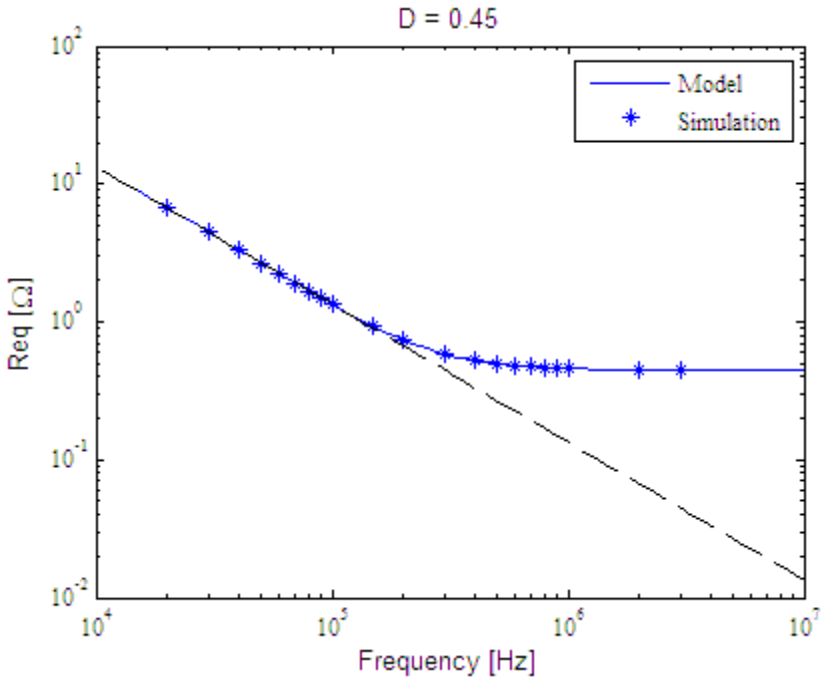

Figure 2.17. Model and simulation results of Fibonacci converter.

SC converters are becoming increasingly complex as researchers seek to improve output voltage regulation, giving rise to many different switching modes. This new model can also be extended to include SC converters with more than two switching

modes. As seen in [5], dithering can be used to switch from one conversion ratio, M_n, to another in SC converters based on extended binary or generic fractional number. The flexibility to change the conversion ratio as necessary allows for maximum converter efficiency over a range of outputs, especially when the resolution of the conversion ratio is high. In the method described in [16], codes are generated for multiple conversion ratios using a spawning technique. If n is equal to 3 and the conversion ratio is 3/8, five different codes can be spawned for that conversion ratio, each representing a different converter topology and switching mode. The dependence of the conversion ratio on the converter topology allows its control by switching between different switching modes, as shown in Figure 2.18. This technique can be incorporated into the proposed model. For example, if five switching modes were included by the use of dithering, then the $\mathbf{\Gamma}$ and $\mathbf{\Phi}$ matrices would simply be expanded to include five terms resulting in

$$\mathbf{\Gamma} = \mathbf{\Phi}_5\mathbf{\Phi}_4\mathbf{\Phi}_3\mathbf{\Phi}_2\mathbf{\Gamma}_1 + \mathbf{\Phi}_5\mathbf{\Phi}_4\mathbf{\Phi}_3\mathbf{\Gamma}_2 + \mathbf{\Phi}_5\mathbf{\Phi}_4\mathbf{\Gamma}_3 + \mathbf{\Phi}_5\mathbf{\Gamma}_4 + \mathbf{\Gamma}_5 \tag{34}$$

and

$$\mathbf{\Phi} = \mathbf{\Phi}_5\mathbf{\Phi}_4\mathbf{\Phi}_3\mathbf{\Phi}_2\mathbf{\Phi}_1 . \tag{35}$$

The **E**, **F**, and **G** matrices would all be constructed the same way by applying KVL and KCL for each of the five switching modes, and the model procedure would not change.

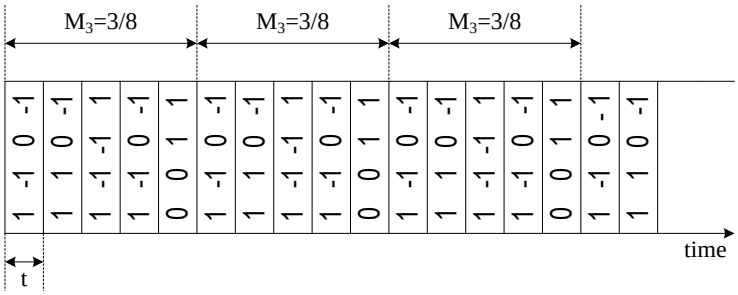

Figure 2.18. Visualization of spawning technique.

2.6. DESIGN PROCEDURE USING PROPOSED MODEL

The key performance metric of an SC converter is the equivalent resistance at the desired switching frequency. A typical design flow using the proposed model would begin with selection of the appropriate SC converter topology based on the particular application. The work presented in [13] details the performance characteristics of many SC converter topologies. After the appropriate topology and dc gain requirements are selected, a basic SC converter design is created incorporating p capacitors, MOSFET on-state resistances, and capacitor ESR. KVL and KCL is then applied to the circuit for each switching mode to find models in matrix form (6). These are converted to the dynamic form of (9) and used to determine the converter's equivalent resistance. Once the equivalent resistance is modeled generically, the designer can change component values to achieve the desired equivalent resistance at a particular frequency. The ideal operating point for an SC converter is near the inflection point of the resistance curve, shown in Figure 2.19. This operating point is selected to achieve the lowest possible equivalent resistance while minimizing switching losses. To achieve maximum efficiency, design iterations using the proposed model can be done to achieve a particular operating point. In general, an SC converter with large capacitance and low MOSFET resistance will achieve the highest efficiency.

The iterative method of determining component values using the proposed model starts with determination of any limiting factors, such as switching frequency,

capacitance, or MOSFET on-state resistance. The designer can then iteratively change component values to achieve a desired operating point. For example, if the maximum switching frequency turns out to be a limiting factor, then the designer can increase capacitance to move the ideal operating point to a lower switching frequency. If the equivalent resistance is still too high, then lowering MOSFET resistance will shift the operating point to a lower resistance. This can be accomplished either by paralleling existing MOSFETs or by finding a device with lower on-state resistance. These iterations continue until the desired operating point is met. Throughout this process, the structure of the model is unchanged, so the computational burden is low.

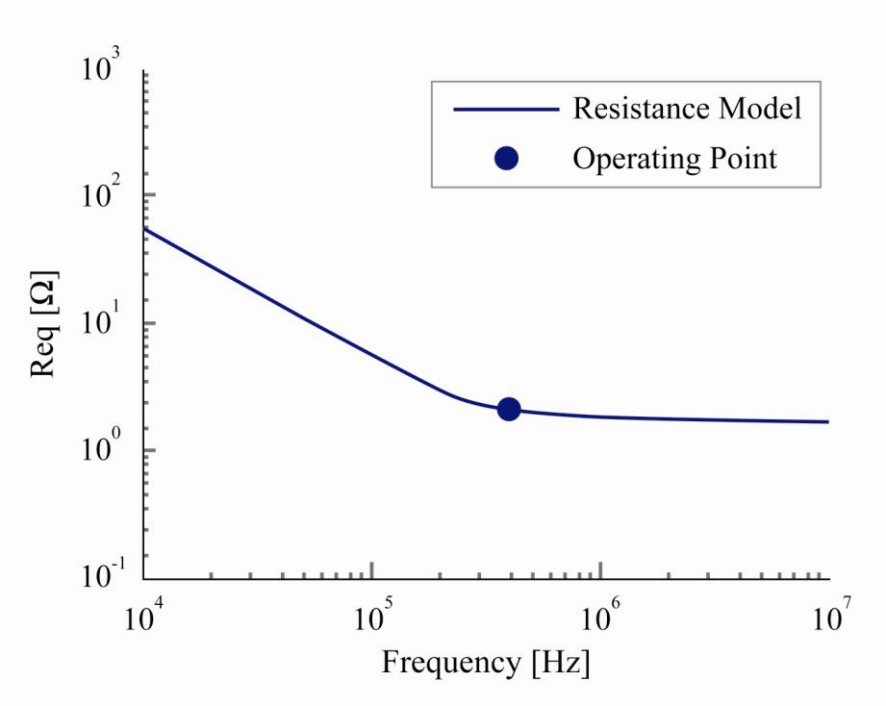

Figure 2.19. Preferred operating point of an SC converter.

3. AUTOMATED SWITCHED-CAPACITOR CONVERTER MODEL GENERATOR

3.1. GENERATING NETLISTS

Automating the model developed in Section 2 involves automating the construction of the **E**, **F**, and **G** matrices. Before matrix construction begins, branch data for the SC converter is compiled in a user-defined netlist. The procedure used to generate a netlist can be illustrated using the SC circuit shown in Fig. 3.1. This ladder topology with a static gain of $M = 2$ doubles the input voltage by alternating switching states and shuttling charge to the output. Each resistor indicates the resistance of a closed "switch" (typically a MOSFET).

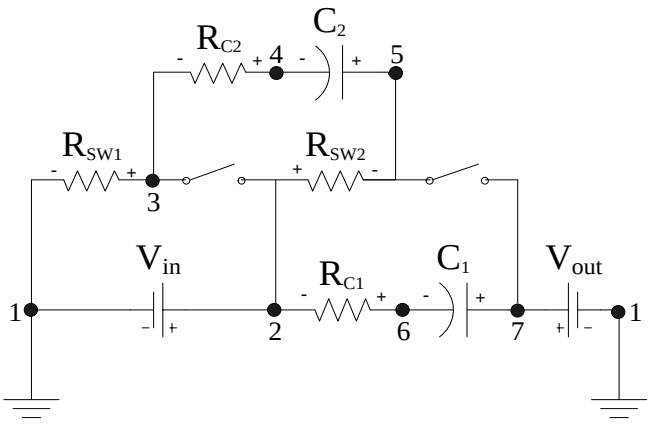

Figure 3.1. One-stage ladder converter during its first switching state.

The circuit, shown during its first switching state, has 7 nodes, 8 branches, and 2 capacitors. Let n denote the number of nodes, b denote the number of branches, and p denote the number of capacitors. The netlist matrix, $\mathbf{N}$, contains the value of each branch element and the nodes to which the branch element is incident. Nodes must be assigned at the individual branch level so that every circuit element has one node incident

to its positive terminal and one node incident to its negative terminal. The netlist matrix can be partitioned as follows:

$$\mathbf{N} = \begin{bmatrix} \tilde{\mathbf{N}}_{p+2,3} \\ \hat{\mathbf{N}}_{b-p-2,3} \end{bmatrix}, \tag{36}$$

where $\tilde{\mathbf{N}}$ contains source and capacitor branches, $\hat{\mathbf{N}}$ contains parasitic branches, and the subscripts denote the dimensions of the partitions. Each row of the matrix represents a branch element; the first entry of each row represents the node connected to the positive terminal of the branch element, and the second entry represents the node connected to the negative terminal of the branch element. The third column stores the value of each branch element. For each branch, the positive node is assumed to correspond to current entering the node. To facilitate computer implementation, the netlist matrix shown in (36) must be filled in the following order: input source branch, output source branch, capacitor branches, and parasitic branches. If neither an input nor an output voltage source is connected to the circuit during a particular switching state, zeros would be entered for all values in the row corresponding to that source. When filling the netlist with the capacitor branches, the designer must identify the capacitor that delivers all the charge to the output and enter that branch last. Simple subroutines can be written to comply with other netlist formats, such as those in SPICE programs, and thus to ensure compatibility with them. For a ladder topology, the last switching capacitor delivers charge to the output; thus, for the example SC circuit in Fig. 3.1, the first switching state netlist matrix is:

$$\mathbf{N} = \left[\begin{array}{cccc|cccc} 2 & 7 & 7 & 5 & 3 & 4 & 2 & 6 \\ 1 & 1 & 6 & 4 & 1 & 3 & 5 & 2 \\ V_{in} & V_{out} & C_1 & C_2 & R_{SW1} & R_{C1} & R_{SW2} & R_{C2} \end{array}\right]^{T}. \tag{37}$$

3.2. LOOP MATRIX DERIVATION

The node incident matrix $\mathbf{A}_a$ is constructed in conjunction with the netlist matrix. Each column corresponds to a branch and contains exactly two nonzero elements, one equal to $+1$ for its positive terminal, the other equal to -1 for its negative terminal. Each row corresponds to a node, where if the positive (negative) terminal of the j^{th} branch element is connected to node i, then $a_{ij} = 1$ ($a_{ij} = -1$). Following this convention, the node incident matrix for the example SC converter in Fig. 3.1 is

$$\mathbf{A_a} = \begin{bmatrix} -1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{38}$$

Next, if $\mathbf{i_{br}}$ is the vector of branch currents, then the i^{th} element of $\mathbf{A_a i_{br}}$ is the sum of currents leaving node i; this sum represents the KCL equation for node i. This equation can then be written as

$$\mathbf{A_a i_{br}} = \mathbf{0}. \tag{39}$$

Any matrix $\tilde{\mathbf{A}}_\mathbf{a}$ that is obtained by adding or subtracting one row from another in $\mathbf{A_a}$ also satisfies (39); thus, after performing row operations and possibly reordering the columns (or branches), $\tilde{\mathbf{A}}_\mathbf{a}$ can be written in row echelon form as

$$\tilde{\mathbf{A}}_\mathbf{a} = \begin{bmatrix} \mathbf{I}_{n-1,n-1} & \hat{\mathbf{A}}_{n-1,b-n+1} \\ \mathbf{0}_{1,n-1} & \mathbf{0}_{1,b-n+1} \end{bmatrix}. \tag{40}$$

The null row in $\tilde{\mathbf{A}}_\mathbf{a}$ is created from an inherent property of the $\mathbf{A}_\mathbf{a}$ matrix. Since each column of $\mathbf{A}_\mathbf{a}$ contains exactly one 1 and one -1, deleting a single row results in no loss of information because the row can be reconstructed from other rows in $\mathbf{A}_\mathbf{a}$. For the example SC converter in Fig. 3.1,

$$\tilde{\mathbf{A}}_\mathbf{a} = \left[\begin{array}{cccccc:cc} 1 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \tag{41}$$

and

$$\hat{\mathbf{A}} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \\ 0 & -1 \\ -1 & 0 \\ -1 & 0 \\ -1 & 0 \end{bmatrix}. \tag{42}$$

The $\tilde{\mathbf{A}}$ matrix is defined by deleting the null row in (40). Based on (39),

$$\tilde{\mathbf{A}}\mathbf{i}_\mathbf{br} = \mathbf{0}, \tag{43}$$

where the branch currents in $\mathbf{i}_\mathbf{br}$ may have been reordered due to row reducing operations performed on $\mathbf{A}_\mathbf{a}$. For the example SC converter, re-ordering was not necessary. The branch currents can then be partitioned into

$$\begin{bmatrix} \mathbf{I} & \hat{\mathbf{A}} \end{bmatrix} \begin{bmatrix} \mathbf{i}_y \\ \mathbf{i}_x \end{bmatrix} = \mathbf{0}\,, \tag{44}$$

where $\mathbf{i}_x$ is a vector of independent branch currents and the currents in $\mathbf{i}_y$ are dependent, meaning that they can be calculated from $\mathbf{i}_x$ using KCL. By expanding (44) and rearranging, $\mathbf{i}_y$ becomes

$$\mathbf{i}_y = -\hat{\mathbf{A}}\mathbf{i}_x\,, \tag{45}$$

and a new expression for the branch currents written as

$$\mathbf{i}_{br} = \begin{bmatrix} \mathbf{i}_y \\ \mathbf{i}_x \end{bmatrix} = \begin{bmatrix} -\hat{\mathbf{A}} \\ \mathbf{I} \end{bmatrix} \mathbf{i}_x = \mathbf{B}_b^T \mathbf{i}_x\,. \tag{46}$$

The $\mathbf{B}_b^T$ matrix relates the branch currents $\mathbf{i}_{br}$ to the independent currents in $\mathbf{i}_x$. By observing Fig. 3.1, it can then be verified that

$$\mathbf{B}_b \mathbf{v}_{br} = \mathbf{0}\,, \tag{47}$$

where $\mathbf{v}_{br}$ is the vector of branch voltages. Substituting (42) into (46), $\mathbf{B}_b$ for the example system is

$$\mathbf{B}_b = \begin{bmatrix} -1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & -1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{48}$$

Each row of $\mathbf{B}_b \mathbf{v}_{br} = \mathbf{0}$ represents the KVL equations applied to the two smaller loops of the example SC converter in Fig. 3.1. Thus, $\mathbf{B}_b$ represents the basic loop matrix [18, 25].

3.3. MATRIX GENERATION FOR SC CONVERTER MODEL

After the basic loop matrix is developed, the state matrices for the model can be constructed algorithmically. The two-stage SC converter shown in Fig. 3.2 better illustrates matrix generation.

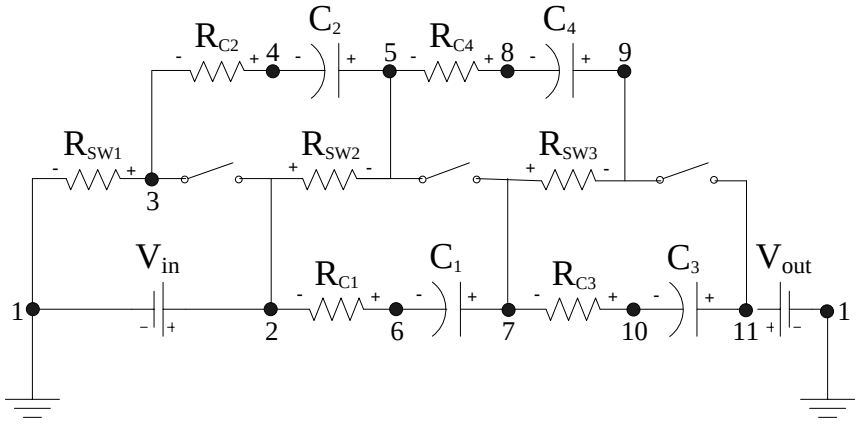

Figure 3.2. Two-stage ladder converter during its first switching state.

This converter has 11 nodes ($n=11$), 13 branches ($b=13$), 4 capacitors ($p=4$), and a static gain of 3 ($M=3$). The associated netlist matrix is

$$\mathbf{N}=\left[\begin{array}{cccccc|ccccccc} 2 & 11 & 7 & 5 & 11 & 9 & 3 & 6 & 2 & 4 & 7 & 10 & 8 \\ 1 & 1 & 6 & 4 & 10 & 8 & 1 & 2 & 5 & 3 & 9 & 7 & 5 \\ V_{in} & V_{out} & C_1 & C_2 & C_3 & C_4 & R_{SW1} & R_{C1} & R_{SW2} & R_{C2} & R_{SW3} & R_{C3} & R_{C4} \end{array}\right]^T. \quad (49)$$

During row reduction of $\mathbf{A_a}$ to $\tilde{\mathbf{A}}_\mathbf{a}$, branches corresponding to R_{C2} (column 10 of $\mathbf{A_a}$) and R_{SW3} (column 11 of $\mathbf{A_a}$) switch places so that column 10 of $\mathbf{B_b}$ corresponds to branch R_{SW3} and column 11 corresponds to branch R_{C2}. Any column swapping performed during this step must also be reflected in **N** to ensure proper branch

assignment. If column i and column j of $\mathbf{A_a}$ are swapped during row reduction, then a new netlist matrix, $\mathbf{N}^*$, must be created by swapping row i and row j of $\mathbf{N}$.

Recall from Section 2, KVL and KCL equations can be applied to find p independent equations relating voltages and currents, expressed in matrix form as shown in (6). The value of each capacitor is arranged in a diagonal matrix, **C**, where C_{jj} is the capacitance of capacitor j. Standard state equations can be written in the form shown in (9), where the state variables are the capacitor voltages. Therefore, to construct the state model quickly and accurately, the **E**, **F**, and **G** matrices must be developed algorithmically.

For each SC converter, there are $b-n+1$ KVL equations and $p+n-b-1$ KCL equations. The **F** and **G** matrices can be developed directly from the first $p+2$ columns of $\mathbf{B_b}$. In partitioned form,

$$\mathbf{B_b} = \left[\hat{\mathbf{G}}_{b-n+1,2} \quad \hat{\mathbf{F}}_{b-n+1,p} \quad \hat{\mathbf{B}} \right] \tag{50}$$

where $\hat{\mathbf{B}}$ is the parasitic loop matrix. For the example SC converter in Fig. 3.2,

$$\mathbf{B_b} = \left[\begin{array}{cc:cccc:ccccccc} -1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & -1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & -1 & -1 & 1 & 0 & 0 & 1 \end{array}\right]. \tag{51}$$

The **F** and **G** matrices are defined as

$$\mathbf{F} = \begin{bmatrix} \hat{\mathbf{F}} \\ \mathbf{0}_{p-(b-n+1),p} \end{bmatrix} \tag{52}$$

and

$$\mathbf{G} = \begin{bmatrix} \hat{\mathbf{G}} \\ \mathbf{0}_{p-(b-n+1),2} \end{bmatrix}. \tag{53}$$

Thus, based on equations (50)-(53),

$$\mathbf{F} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{54}$$

and

$$\mathbf{G} = \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{55}$$

The **E** matrix, however, cannot be developed directly due to the relationship of the capacitor currents. First, let the **E** matrix be defined as

$$\mathbf{E} = \begin{bmatrix} \hat{\mathbf{E}}_{b-n+1,p} \\ \mathbf{K}_{p-(b-n+1),p} \end{bmatrix}, \tag{56}$$

where $\hat{\mathbf{E}}$ represents the parasitic branches included in each KVL loop and $\mathbf{K}$ represents the KCL equations. For the model developed in Section 2, only capacitor currents are used for construction of the **E** matrix, but the $\mathbf{B}_\mathbf{b}^\mathbf{T}$ matrix relates all branch currents to the independent branch currents, which may not necessarily be capacitor currents. Thus, another matrix is created to relate each branch to the appropriate capacitor currents. The capacitor current matrix $\mathbf{i}_{\mathbf{cap}}$, size $p \times b$, is created by comparing the rows of $\mathbf{B}_\mathbf{b}^\mathbf{T}$ that relate the capacitor currents to the independent branch currents with the rows that relate

the rest of the branch currents to the independent branch currents. Each row of $\mathbf{i}_{\mathbf{cap}}$ corresponds to a capacitor current and contains either a $+1$ (if the branch and capacitor currents are the same), a -1 (if the branch and capacitor currents are in opposite directions), or 0 (if the branch currents cannot be related by just one capacitor current). For example, if the current in the j^{th} branch is the same as the current in the i^{th} capacitor (i.e., if row j of $\mathbf{B}_\mathbf{b}^\mathbf{T}$ is identical to the row in $\mathbf{B}_\mathbf{b}^\mathbf{T}$ corresponding to the i^{th} capacitor), then $\mathbf{i}_{\mathbf{cap}}(i, j)$ is equal to 1. Similarly, if the current in the j^{th} branch is the negative of the current in the i^{th} capacitor (i.e., if row j of $\mathbf{B}_\mathbf{b}^\mathbf{T}$ is equal to the row in $\mathbf{B}_\mathbf{b}^\mathbf{T}$ corresponding to the i^{th} capacitor multiplied by -1), then $\mathbf{i}_{\mathbf{cap}}(i, j)$ is equal to -1. The highlighting in equations (57) and (58) illustrate the development of $\mathbf{i}_{\mathbf{cap}}$ from $\mathbf{B}_\mathbf{b}^\mathbf{T}$ for the converter in Fig. 3.2.

$$\mathbf{B}_\mathbf{b}^\mathbf{T} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & -1 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{Capacitor Branches} \tag{57}$$

$$\mathbf{i}_{\mathbf{cap}} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{58}$$

The $\mathbf{i_{cap}}$ matrix can be verified by examining Fig. 3.2. As shown in (58), $i_{R_{C1}} = i_{C_1}$, $i_{R_{SW1}} = i_{R_{C2}} = i_{C_2}$, $i_{R_{C3}} = -i_{V_{out}} = i_{C_3}$, and $i_{R_{SW3}} = i_{R_{C4}} = i_{C_4}$. The ninth column in $\mathbf{i_{cap}}$ corresponding to branch R_{SW2} contains only zeros, indicating that the current through that branch cannot be expressed as a single capacitor current.

After determining which branch currents are equal to capacitor currents, it is possible to begin filling the $\hat{\mathbf{E}}$ matrix. Parasitic branches with branch currents that are equal to a capacitor current (or the negative of a capacitor current) can be stored in the $\hat{\mathbf{E}}$ matrix directly. Disregarding the source and capacitor branches, multiplying $\hat{\mathbf{B}}$ with the third column of $\hat{\mathbf{N}}$ element by element, and matching the resulting values to a capacitor current using $\mathbf{i_{cap}}$ places the parasitic resistance values in $\hat{\mathbf{E}}$. At this point, the $\hat{\mathbf{E}}^*$ matrix for the example circuit in Fig. 3.2 is

$$\hat{\mathbf{E}}^* = \begin{bmatrix} 0 & R_{SW} + R_C & 0 & 0 \\ R_C & 0 & R_C & 0 \\ -R_C & 0 & 0 & R_{SW} + R_C \end{bmatrix}, \tag{59}$$

where the asterisk denotes that the $\hat{\mathbf{E}}$ is incomplete. Parasitic branches whose branch currents are a function of multiple capacitor currents, in this case branch $R_{SW2,}$ are not yet included. These branches must be matched with KCL equations relating their branch currents to the capacitor currents.

Generating the required KCL equations requires finding the rows of $\mathbf{B_b^T}$ that relate the capacitor currents to independent branch currents that are also capacitor currents. For the converter in Fig. 3.2, equation (46) yields

$$\mathbf{B}_\mathbf{b}^\mathbf{T}\mathbf{i}_\mathbf{x} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 0 \\ \hline 0 & 1 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \hline 1 & 0 & 0 \\ 0 & 1 & -1 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i_{R_{C2}} \\ i_{R_{C3}} \\ i_{R_{C4}} \end{bmatrix}. \tag{60}$$

Rows 3 through 6 of (60) represent the currents through capacitors C_1 through C_4, respectively. In this case, all currents in $\mathbf{i_x}$ are capacitor currents since, as established from (58), $i_{R_{C1}} = i_{C_1}$, $i_{R_{C2}} = i_{C_2}$, and $i_{R_{C3}} = i_{C_3}$. Thus, the third row of (60) yields the KCL relationship $i_{C_1} = i_{C_3} - i_{C_4}$. For the converter in Fig. 3.2, there are three independent KVL equations; therefore, just one KCL equation is needed to satisfy the p equations required in the model. Adding the KCL relationship described above yields

$$\mathbf{K} = \begin{bmatrix} -1 & 0 & 1 & -1 \end{bmatrix}. \tag{61}$$

From (60), the current in branch R_{SW2}, corresponding to row nine of $\mathbf{B}_\mathbf{b}^\mathbf{T}$, is related to the capacitor currents by $i_{R_{SW2}} = i_{C_2} - i_{C_4}$; thus, from (51), R_{sw2} is added to the first loop and subtracted from the third. Adding these parasitic branches to $\hat{\mathbf{E}}$ yields

$$\hat{\mathbf{E}} = \begin{bmatrix} 0 & 2R_{SW} + R_C & 0 & -R_{SW} \\ R_C & 0 & R_C & 0 \\ -R_C & -R_{SW} & 0 & 2R_{SW} + R_C \end{bmatrix}. \tag{62}$$

Comparison of (61) and (62) with (56) yields,

$$\mathbf{E} = \begin{bmatrix} 0 & 2R_{SW} + R_C & 0 & -R_{SW} \\ R_C & 0 & R_C & 0 \\ -R_C & -R_{SW} & 0 & 2R_{SW} + R_C \\ -1 & 0 & 1 & -1 \end{bmatrix}. \tag{63}$$

3.4. COMPUTER IMPLEMENTATION

The algorithm proposed here was implemented using Matlab. Figure 3.3 shows the general structure of the algorithm. Bold boxes represent user-entered input quantites, and the dotted box represents the state model generator algorithm. Input quantities include the circuit netlist for each switching state, duty ratios for each switching state, and the converter's static gain value, M. Only the circuit netlists are used directly to generate the state matrix because they are needed to generate the node incidence matrix, $\mathbf{A}_a$. Row reduction of $\mathbf{A}_a$ was accomplished through the Matlab command `rref(`$\mathtt{A}_a$`)` so that no extra programming was required; however, if computer implementation relies on other software, algorithms for performing row reduction are described in [25]. Calculation of the remaining matrices in the state model generator requires only simple matrix operations.

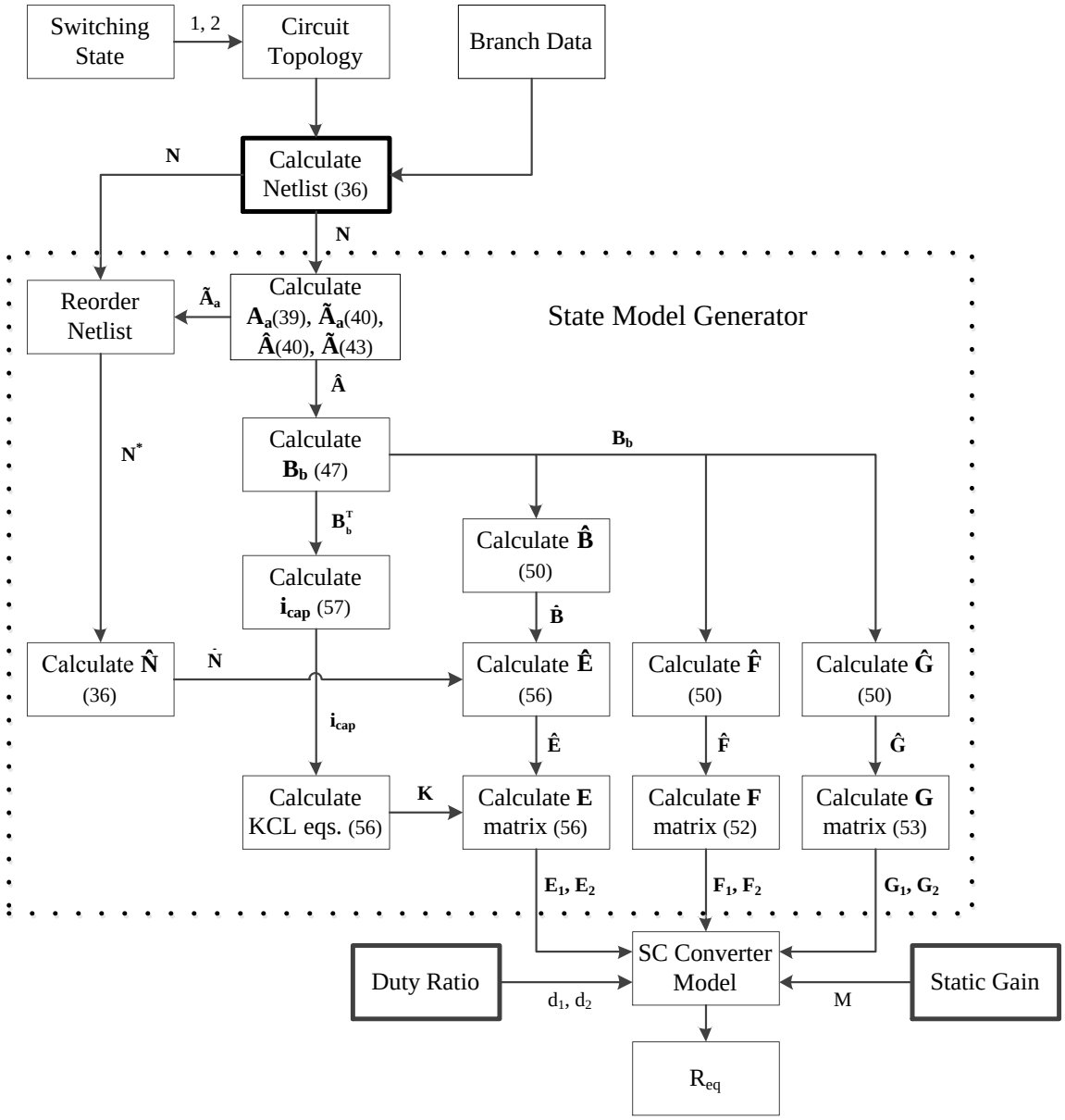

Figure 3.3. Block diagram of algorithm implementation.

Once the **E**, **F**, and **G** matrices for each switching state are calculated, they are used in the SC converter model developed in Section 2. The **Φ** and **Γ** matrices can be computed for a given switching frequency with the Matlab function `c2d`. Everything in the state model generator and the model is solved numerically.

3.5. ALGORITHM VERIFICATION

To demonstrate the validity of the proposed algorithm, simulation and experimental data from Sections 2.3 and 2.5 were compared to the algorithm results. The two-stage ladder converter analyzed above is redrawn more generically in Fig. 3.4 with ideal switches and their corresponding switching states. When a switch is on, it is represented by a switching resistor, R_{SW}.

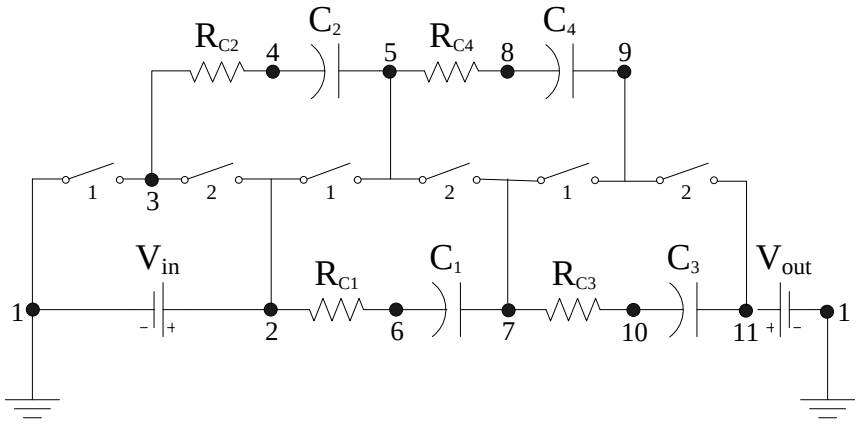

Figure 3.4. Two-stage ladder converter.

Each "switch" is actually two FDMS8460 MOSFETs in parallel, for an equivalent switch resistance of 3 mΩ (denoted as R_{SW} below). Each "capacitor" is actually four TMK325BJ226MM-T ceramic capacitors from Nichicon (22 µF, 25 V) in parallel, for a total equivalent series resistance (ESR) of 10 mΩ (denoted as R_C below) and capacitance of 88 µF. The duty ratios for each switching state were 45%, and the input voltage was set to 10 V. The resultant **E**, **F**, and **G** matrices generated manually and from the proposed algorithm (barred matrices) for each switching state are

$$\mathbf{E_1} = \begin{bmatrix} 0 & 2R_{SW} + R_C & 0 & -R_{SW} \\ -R_C & -R_{SW} & 0 & 2R_{SW} + R_C \\ R_C & 0 & R_C & 0 \\ 1 & 0 & -1 & 1 \end{bmatrix}, \tag{64}$$

$$\mathbf{F_1} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{65}$$

$$\mathbf{G_1} = \begin{bmatrix} -1 & 0 \\ 0 & 0 \\ 1 & -1 \\ 0 & 0 \end{bmatrix}, \tag{66}$$

$$\mathbf{E_2} = \begin{bmatrix} -(2R_{SW} + R_c) & R_c & R_{SW} & 0 \\ R_{SW} & 0 & -(2R_{SW} + R_c) & R_c \\ 0 & -R_c & 0 & -R_c \\ 1 & 1 & -1 & -1 \end{bmatrix}, \tag{67}$$

$$\mathbf{F_2} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{68}$$

and

$$\mathbf{G_2} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -1 & 1 \\ 0 & 0 \end{bmatrix}. \tag{69}$$

For the first switching state, algorithm values for $\bar{\mathbf{E}}_1$, $\bar{\mathbf{F}}_1$, and $\bar{\mathbf{G}}_1$ were derived above in (63), (54), and (55), respectively. For the second switching state,

$$\bar{\mathbf{E}}_2 = \begin{bmatrix} -R_C & (2R_{SW} + R_C) & 0 & -R_{SW} \\ R_C & 0 & R_C & 0 \\ R_C & -R_{SW} & 0 & (2R_{SW} + R_C) \\ -1 & -1 & 1 & 1 \end{bmatrix}, \tag{70}$$

$$\bar{\mathbf{F}}_2 = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{71}$$

and

$$\bar{\mathbf{G}}_2 = \begin{bmatrix} 0 & 0 \\ 1 & -1 \\ 1 & -1 \\ 0 & 0 \end{bmatrix}. \tag{72}$$

A comparison of the matrices demonstrates that the algorithm generates matrices that differ from those calculated manually. These are due to differences in the KVL loops selected for each switching stage. The manner in which the KVL loops are selected is irrelevant as long as all loops are independent of each other. Furthermore, the order in which the loops are calculated and the selection of KCL equations also changes the matrices, but does not change the final result. Figure 3.5 shows the loops selected during manual calculations (solid lines) and those selected by the algorithm (dashed lines) during the second switching state.

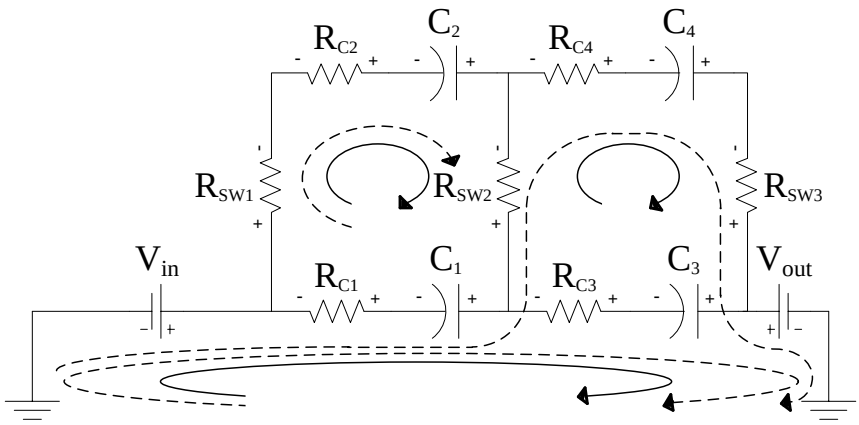

Figure 3.5. KVL loops for two-stage ladder converter during its second switching state.

Although the matrices may look different, they are fundamentally equivalent, as demonstrated in Fig. 3.6. Each set of matrices was used with the converter model, resulting in the same equivalent resistance curve.

The experimental data collected closely correlates with the algorithm and the manual calculations. Again, the slight discrepancies at 35 kHz and 70 kHz, where the experimental data is slightly higher than those of the model, is explained by the inherent inductance of the PCB board due to its design. This inductance creates a resonant effect that interacts with the capacitors, increasing the resistance at harmonics of 35 kHz. The inherent board inductance causing the resonance was estimated to be 30 nH.

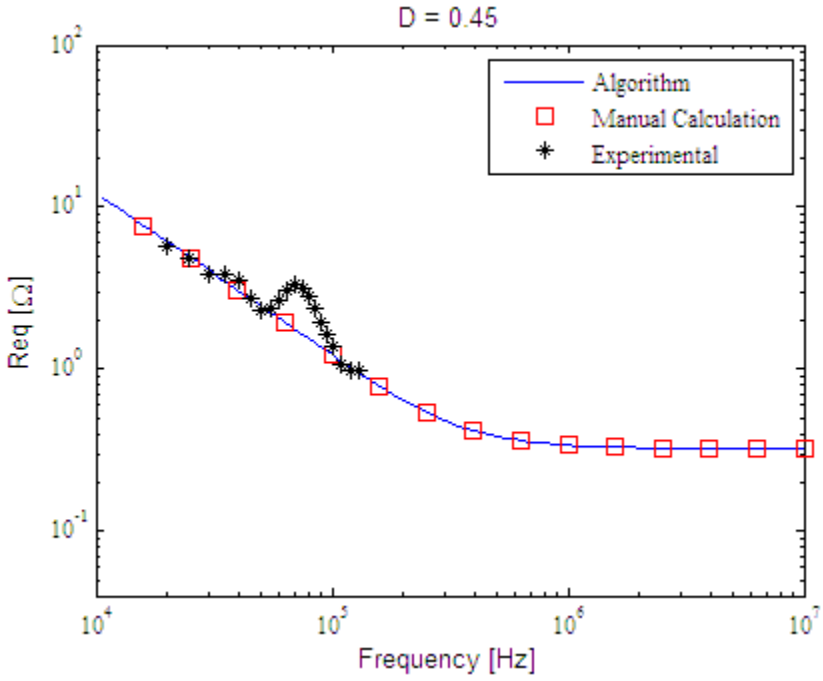

Figure 3.6. Comparison of algorithm with experimental and manually calculated data for the two-stage ladder converter.

The state model generator was designed to work with other SC converter topologies as well. The Fibonacci SC converter in Fig. 3.7 was analyzed to demonstrate the flexibility of the algorithm.

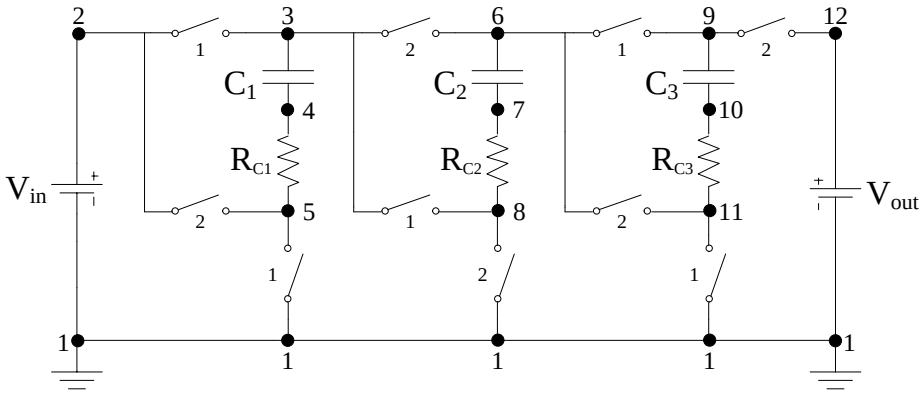

Figure 3.7. Fibonacci converter with gain of 5.

The input voltage V_{in} was arbitrarily set to 5V and the topology gain, based on the Fibonacci sequence, is 5. Component values are identical to those of the two-stage ladder converter, and the duty ratios for each switching state were 45%. The **E**, **F**, and **G** matrices resulting from manual calculations and the state model generator are summarized in Table 3.1 below.

Table 3.1. Comparison of coefficient matrices generated manually and by the algorithm

Manually Calculated Matrices	**Algorithm Calculated Matrices**
$\mathbf{E_1} = \begin{bmatrix} 2R_{SW}+R_C & -R_{SW} & 0 \\ -(R_{SW}+R_C) & 0 & 3R_{SW}+2R_C \\ 0 & 1 & 1 \end{bmatrix}$	$\bar{\mathbf{E}}_1 = \begin{bmatrix} 2R_{SW}+R_C & -R_{SW} & 0 \\ R_{SW} & -R_{SW} & 3R_{SW}+2R_C \\ 0 & 1 & 1 \end{bmatrix}$
$\mathbf{F_1} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$\bar{\mathbf{F}}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$
$\mathbf{G_1} = \begin{bmatrix} -1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\bar{\mathbf{G}}_1 = \begin{bmatrix} -1 & 0 \\ -1 & 0 \\ 0 & 0 \end{bmatrix}$
$\mathbf{E}_2 = \begin{bmatrix} -(2R_{SW}+R_C) & R_{SW}+R_C & 0 \\ 0 & -(R_{SW}+R_C) & -(2R_{SW}+R_C) \\ -1 & -1 & 1 \end{bmatrix}$	$\bar{\mathbf{E}}_2 = \begin{bmatrix} -(2R_{SW}+R_C) & R_{SW}+R_C & 0 \\ -(2R_{SW}+R_C) & 0 & -(2R_{SW}+R_C) \\ -1 & -1 & 1 \end{bmatrix}$
$\mathbf{F}_2 = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & -1 \\ 0 & 0 & 0 \end{bmatrix}$	$\bar{\mathbf{F}}_2 = \begin{bmatrix} -1 & 1 & 0 \\ -1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}$
$\mathbf{G}_2 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$	$\bar{\mathbf{G}}_2 = \begin{bmatrix} -1 & 0 \\ -1 & 1 \\ 0 & 0 \end{bmatrix}$

Again, the matrices generated by the algorithm differ slightly from those generated manually due to the different KVL loops selected by each. Although experimental data were not collected for the Fibonacci converter, simulation data were. The converter was again simulated using PLECS, incorporating the same component parameters and switch duty cycle included in the model and algorithm. As in the model, the voltage sources were placed on the input and output terminals of the converter. The input voltage was set at 5V and the output voltage at 24V. The PLECS simulation setup is shown in Fig. 3.8, and the data collected were compared to the results of manual calculations and the state model generator. Figure 3.9 reveals identical results for all three, validating the effectiveness of the SC model generator for the Fibonacci topology.

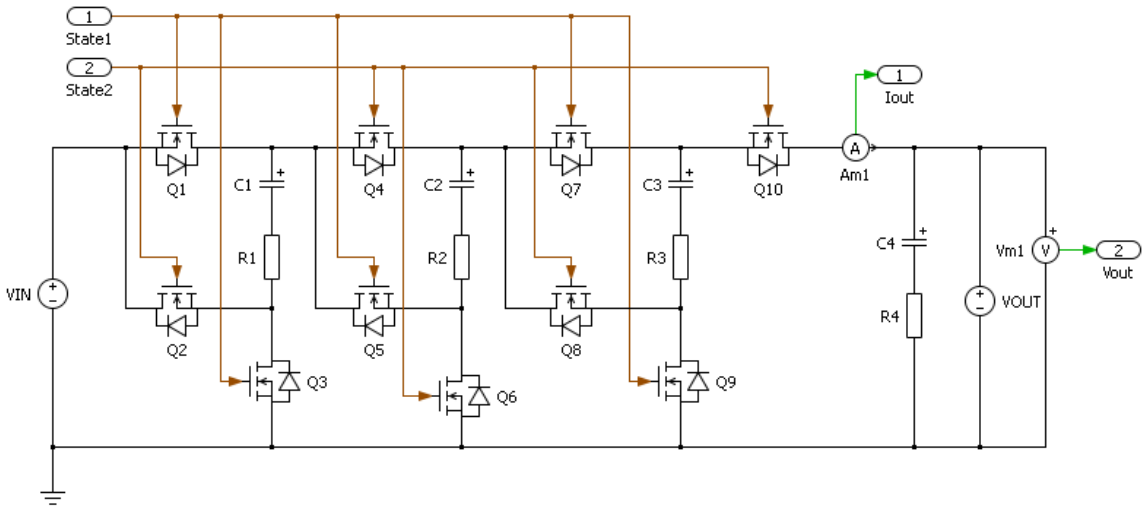

Figure 3.8. PLECS simulation model for Fibonacci converter.

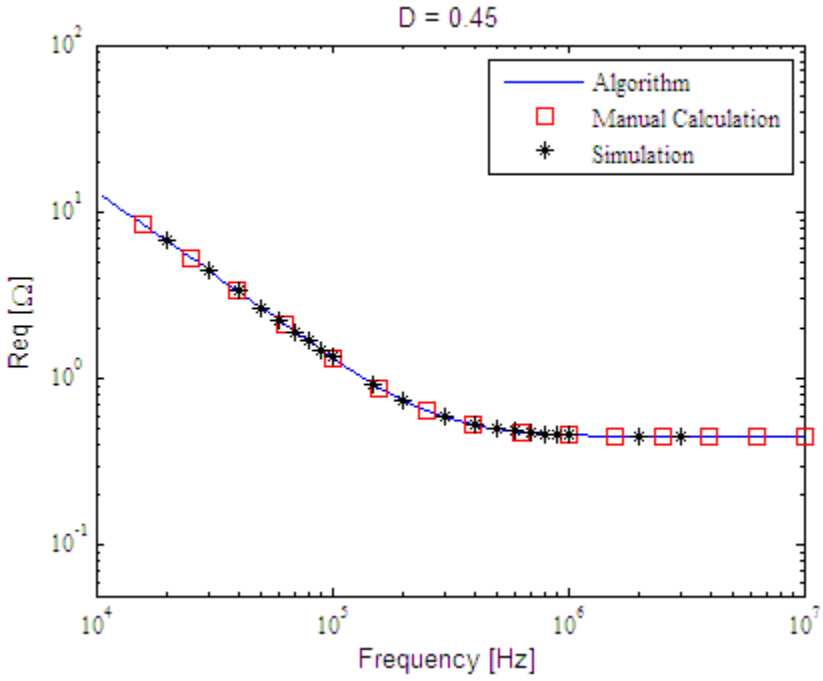

Figure 3.9. Comparison of algorithm to simulated and manually calculated data for the Fibonacci converter.

4. CONCLUSIONS

4.1. SUMMARY OF RESULTS

The work outlined in the previous sections derived a new analysis technique for modeling the equivalent resistance of complex SC converters and proposed a method of automating the model's development. It verified both methods with simulations and experimentation on a variety of SC converter topologies.

The proposed modeling technique is an analysis and design method superior to previous methods. The method in [11-12] was simple and direct, but it applied only to simple voltage doublers. The new model can analyze both simple and complex SC circuits with greater detail. The method outlined in [13, 15] had the advantage of quickly modeling regular converter structures operating at very low or very high frequencies; for converters with unusual structures that operate at intermediate frequencies, however, it does not provide results directly. The model proposed here can accurately model all complex converter designs at all frequencies where inductance effects are minimal. The method proposed in [16] relies on well known energy and power principles and is capable of analyzing simple, hard and soft switched SC converters, but it is limited to converters operating with just two switching modes. The method presented here can incorporate converters with any number of switching modes. Also, the design method in [10] focused on a particular application, and the analysis method in [26] is useful for results in steady state without consideration of parasitics. This new model is effective for modeling and designing SC converters of any topology and at any power level while also incorporating parasitics. For the degenerate case of a single capacitor, the new method agrees with a previously published method. This method is also useful when extended to complex switching techniques. Furthermore, the proposed model replicates the experimental, simulation, and previously published data while offering a superior means to analyze SC converters. With the ability to quickly calculate the equivalent resistance of many complex converters, this model is a powerful tool for SC converter design.

A new state generation algorithm was also introduced to automate the development of the SC converter model. Other methods of automating circuit analysis, such as MNA [19-20] and STF [21], are effective for solving a circuit's VI

characteristics, but are not suitable for state model generation. Furthermore, the state model generator outlined in [18] is difficult to integrate with the SC converter model because difference equations are not implemented at the individual branch level. The state model generator outlined here is better suited for integration with the model because circuit representation is done at the individual branch level. For each switching state, a user-defined netlist is used to generate the node incidence matrix. The node incidence matrix is row reduced and partitioned to create the basic loop matrix. The loop matrix is used to relate capacitor currents to independent branch currents so that state model matrices can be developed. This algorithm has been verified with simulations and experimentation for multiple switching topologies, demonstrating speed and accuracy in implementing the model.

Together, the new SC converter modeling technique and the automated state model generator algorithm improve SC converter design capabilities. The new modeling technique improves the accuracy of modeling the output impedance of an SC converter and the state model generator decreases the time spent on converter design. Combined, the result is a new valuable SC converter design tool.

4.2. EXTENSIONS

Although the automated state model generator algorithm enables automation of the SC converter model, further attention can improve its functionality. First, work could be done to reduce the amount of Matlab code required. The Matlab code for the state model generator, shown in Appendix B, is copied twice to calculate the state coefficient matrices for each switching state; however, if the code was modularized, just one set of code would be needed for all switching states. This feature would be of particular interest when considering SC converters with more than two switching states. If an SC converter with five switching states was being modeled, then the code would need to be copied five times, increasing the amount of code for automating the model substantially.

Integrating netlist compatibility with other circuit analysis software, such as PSPICE, is another addition worth mentioning. The capability to import text files into Matlab would be required and code would need to be written to integrate netlists

generated from other programs. This addition would facilitate netlist generation by eliminating manual construction.

Finally, verification testing should be done on cascaded SC converters. Theoretically, the state model generator would work as long as the correct netlist is created, but further testing is needed to ensure accuracy.

APPENDIX A.
PRINTED CIRCUIT BOARD DESIGN

This appendix includes the schematics and board layouts of the printed circuit boards described in Section 2.3. For clarity, the schematic diagram is broken into three parts and shown in Figures A.1 through A.3. The physical board layout, pictured in Fig. 2.6, is broken into two parts to show the top and bottom layers of the board separately. The board layout for the top layer is shown in Fig. A.4 and the board layout for the bottom layer is shown in Fig. A.5. The dashed polygons in the board layouts represent solder planes.

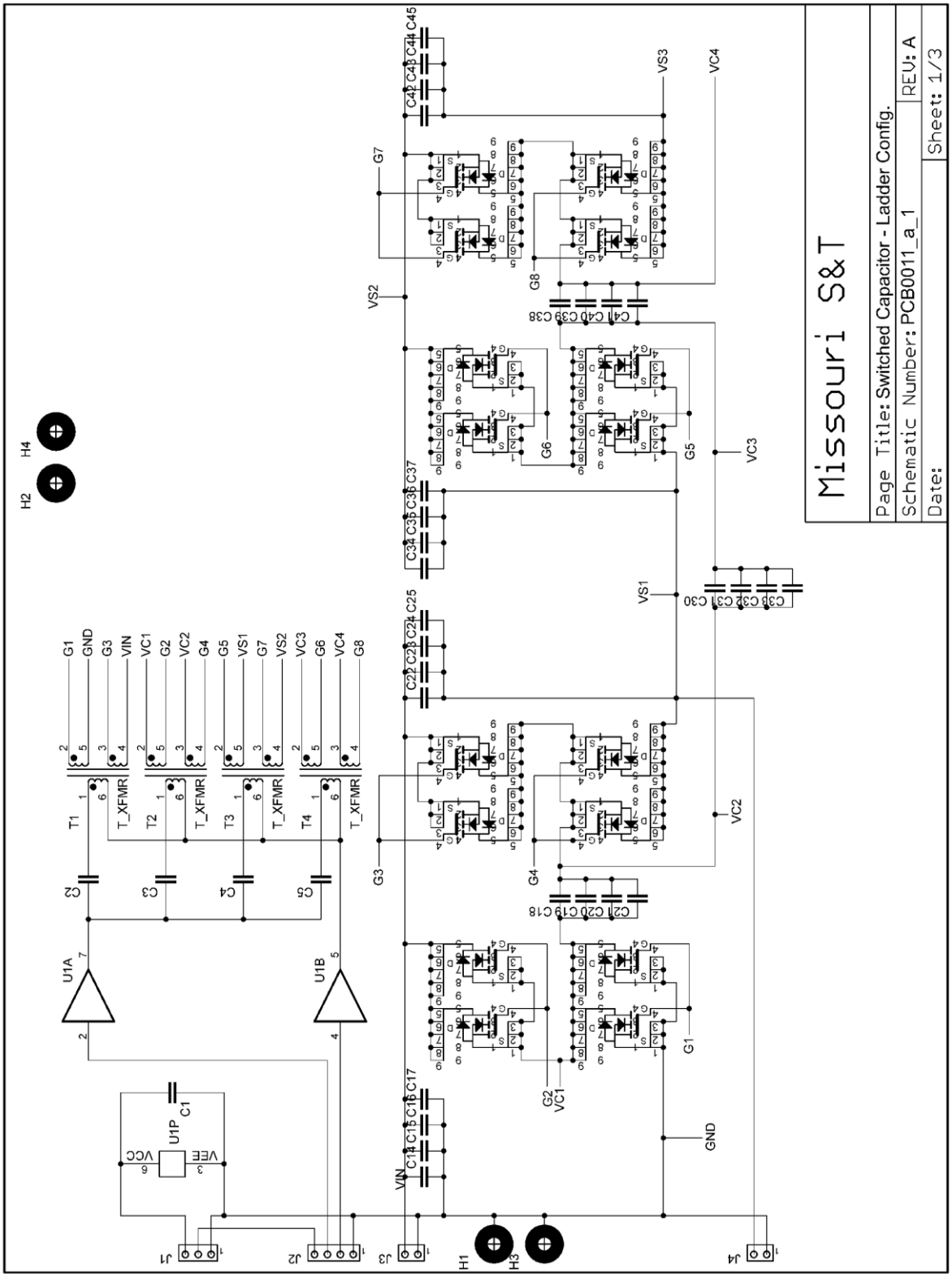

Figure A.1. Sheet one of printed circuit board schematic.

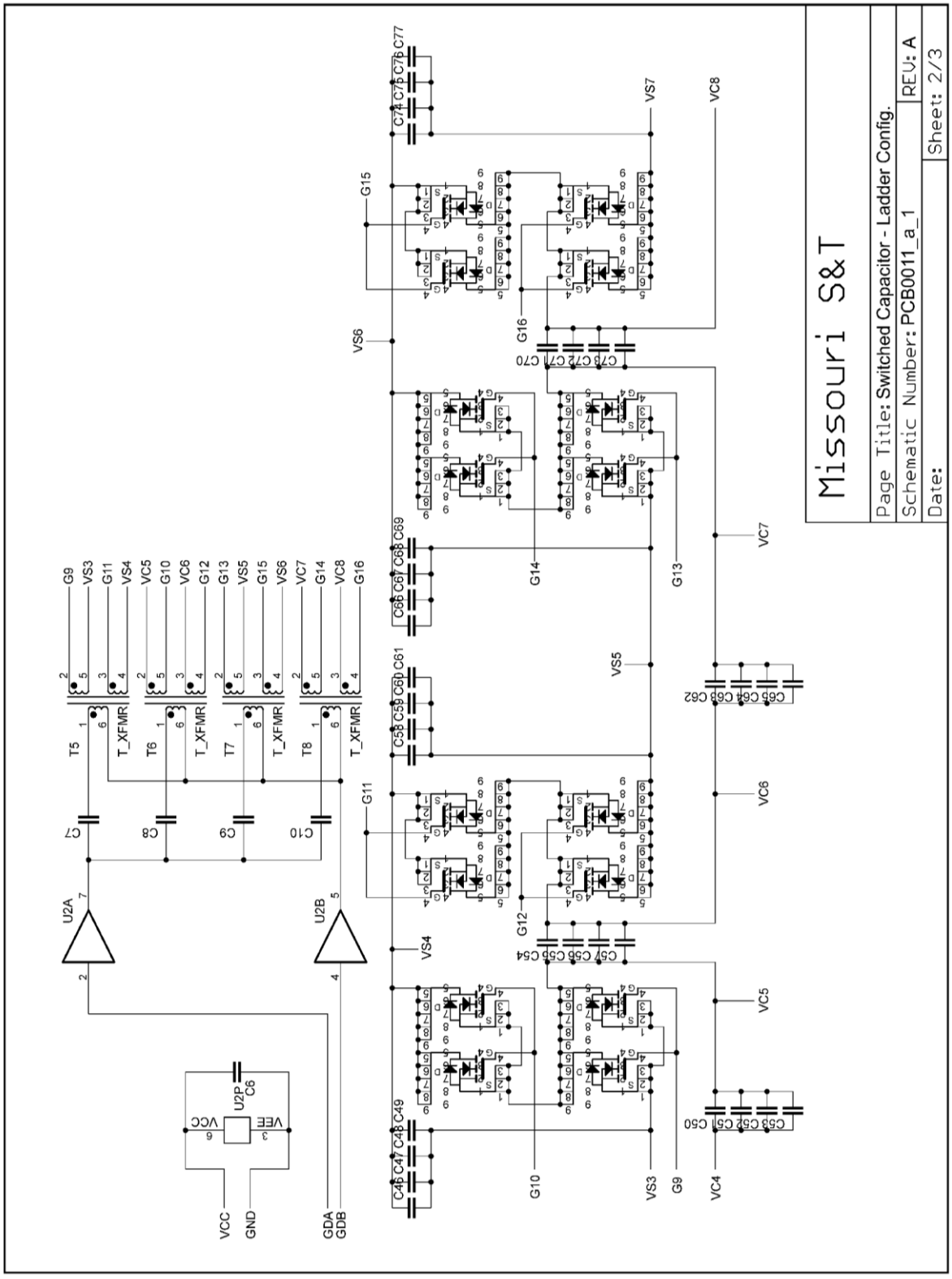

Figure A.2. Sheet two of printed circuit board schematic.

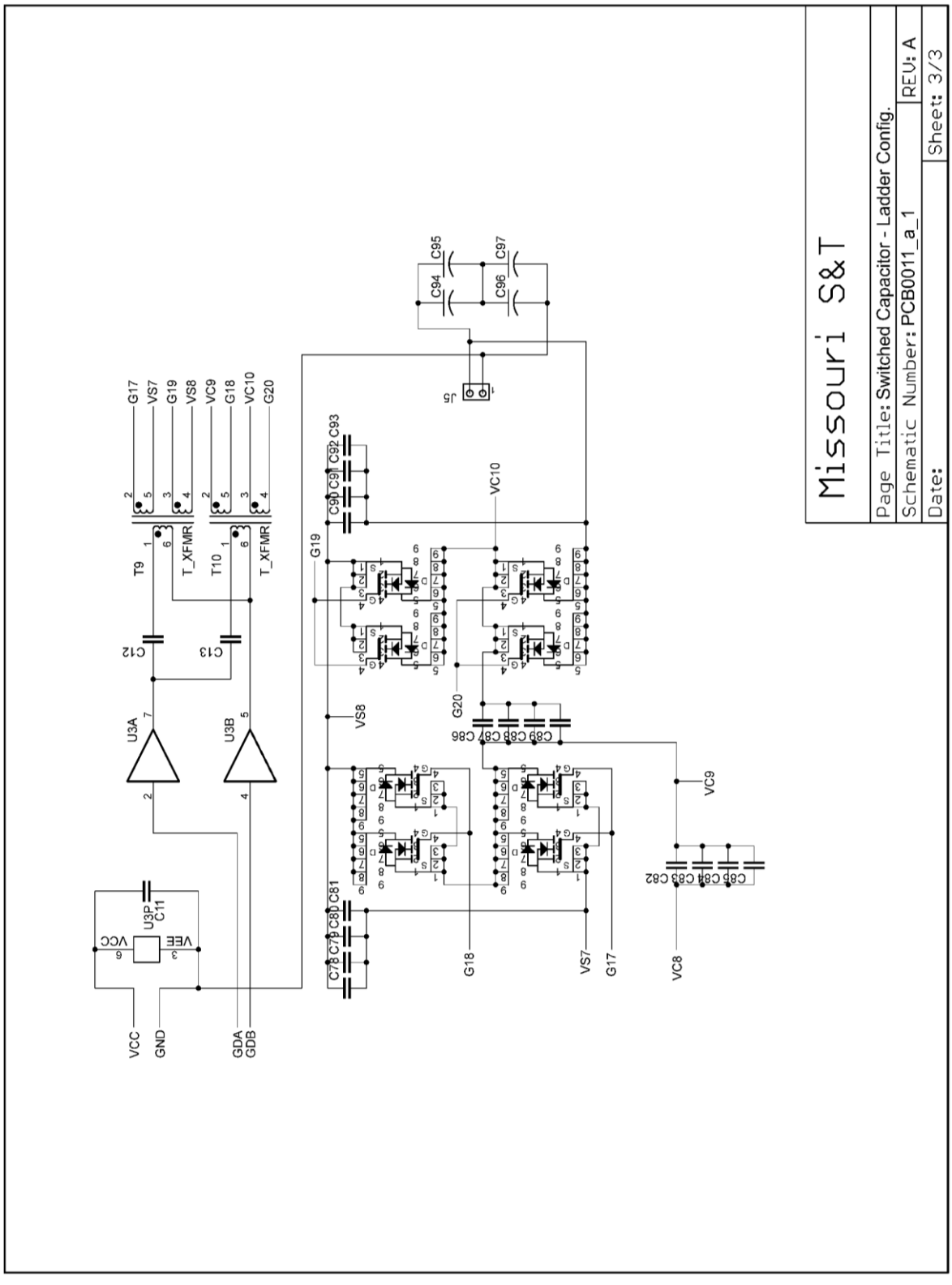

Figure A.3. Sheet three of printed circuit board schematic.

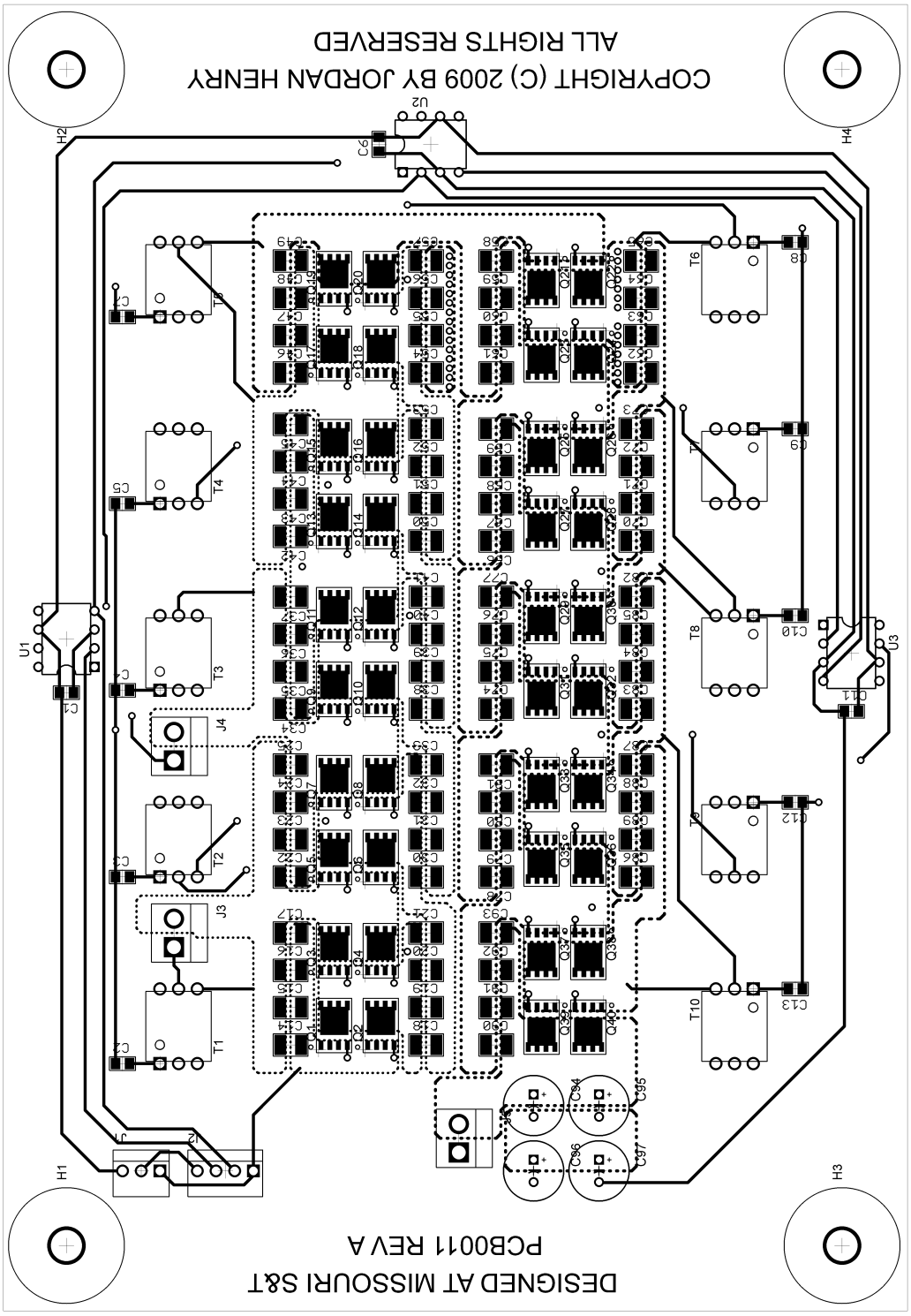

Figure A.4. Physical board layout of top layer.

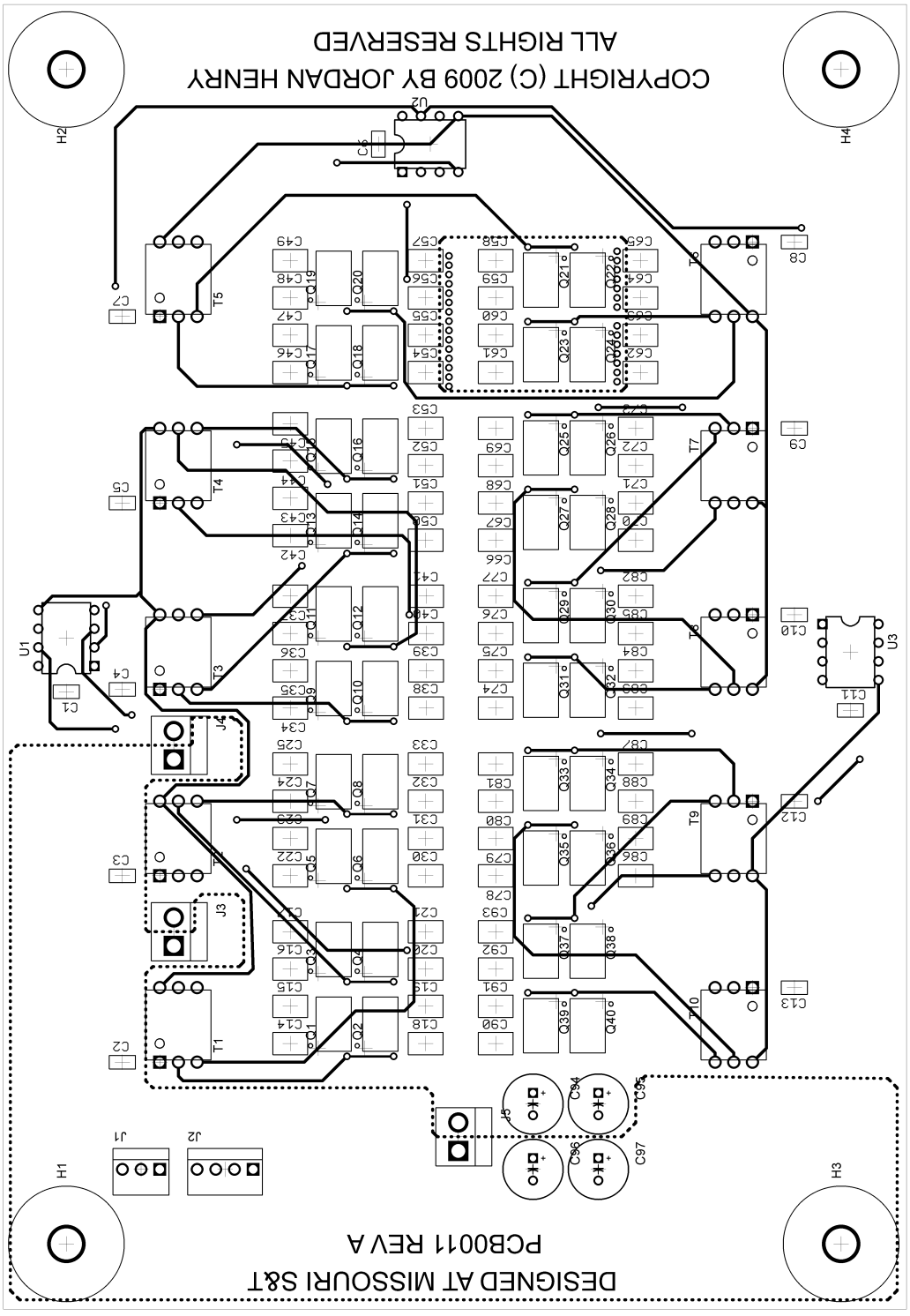

Figure A.5. Physical board layout of bottom layer.

APPENDIX B.
MATLAB CODE FOR THE AUTOMATED STATE MODEL GENERATOR

The automated state-model generator was implemented using Matlab R2009a. The code includes considerable commenting to explain the purpose of each block of code. Comments are denoted by text that is preceded by a percent symbol (%). Lines of code that are too long to fit on one line are continued on the next line and marked by three consecutive periods (…). The code below incorporates extra coding for compatibility with netlists that are more general in structure. The extra code, labeled "Format netlist", formats user-defined netlists into the format described in Section 3.1. The new, more general netlist format is described in the section of code labeled "FUNCTION DESCRIPTION".

```
%Copyright by Jordan Henry
%All Rights Reserved
%SCC Model and State Model Algorithm

function [E1,F1,G1,E2,F2,G2,Rmin] = sccmodel(list1,list2,gain,d1,d2)

%FUNCTION DESCRIPTION-------------------------------------------------------
%The sccmodel function is used to model the equivalent resistance of an
%SC converter. For the model to be generated properly, d1 and d2 must be be
%entered as a number between 0 and 1. The list1 and list2 inputs are
%matrices of size bx5, where b is the number of branches. Branch
%information is entered into the netlists in the following fashion:

%*The 1st column must contain the type of branch element ("V" for a voltage
%source, "C" for a capacitor, and "R" for a resistor)
%*The 2nd column must contain the number associated with the branch.
%Numbering starts from 1 for each branch type.
%*The 3rd column must contain the number of the positive node connected to
%the branch.
%*The 4th column must contain the number of the negative node connected to
%the branch.
%*The 5th column must contain the value of the branch element.

%The designer must identify the capacitor that delivers charge to the load
%and label that capacitor as the last capacitor, meaning that capacitor
%should have the highest capacitor number.
%The static gain of the converter must also be entered.

%---------------------------------------------------------------------------
%---------------------------------------------------------------------------
%Assign the letters a number, Matlab must store letters as a number
V = 1;
C = 2;
R = 3;

%Calculate the number of capacitors-----------------------------------------
capnum = 0;
for i = 1:length(list1)
    if list1(i,1) == C
        capnum = capnum+1;
    end
end
```

```
%Format netlists-------------------------------------------------------------
%LIST1

append = zeros(1,5);
m = 0;
n = 0;
p = 0;
q = 0;

for y = 1:length(list1)
    for j = 1:length(list1)-m
        if list1(j,1) == V
            for i = 1:length(list1)
                if list1(i,2) == 1+n & list1(i,1) == V
                    append = list1(i,:);
                    list1(i,:) = [];
                    list1 = vertcat(list1,append);
                    n = n+1;
                    m = m+1;
                end
            end
        end
    end
end
for y = 1:length(list1)
    for j = 1:length(list1)-m
        if list1(j,1) == C
            for i = 1:length(list1)
                if list1(i,2) == 1+p & list1(i,1) == C
                    append = list1(i,:);
                    list1(i,:) = [];
                    list1 = vertcat(list1,append);
                    p = p+1;
                    m = m+1;
                end
            end
        end
    end
end
for y = 1:length(list1)
    for j = 1:length(list1)-m
        if list1(j,1) == R
            for i = 1:length(list1)
                if list1(i,2) == 1+q & list1(i,1) == R
                    append = list1(i,:);
                    list1(i,:) = [];
                    list1 = vertcat(list1,append);
                    q = q+1;
                    m = m+1;
                end
            end
        end
    end
end

list1(:,1) = [];
list1(:,1) = [];

%LIST 2
append = zeros(1,5);
m = 0;
n = 0;
```

```
p = 0;
q = 0;

for y = 1:length(list2)
    for j = 1:length(list2)-m
        if list2(j,1) == V
            for i = 1:length(list2)
                if list2(i,2) == 1+n & list2(i,1) == V
                    append = list2(i,:);
                    list2(i,:) = [];
                    list2 = vertcat(list2,append);
                    n = n+1;
                    m = m+1;
                end
            end
        end
    end
end
for y = 1:length(list2)
    for j = 1:length(list2)-m
        if list2(j,1) == C
            for i = 1:length(list2)
                if list2(i,2) == 1+p & list2(i,1) == C
                    append = list2(i,:);
                    list2(i,:) = [];
                    list2 = vertcat(list2,append);
                    p = p+1;
                    m = m+1;
                end
            end
        end
    end
end
for y = 1:length(list2)
    for j = 1:length(list2)-m
        if list2(j,1) == R
            for i = 1:length(list2)
                if list2(i,2) == 1+q & list2(i,1) == R
                    append = list2(i,:);
                    list2(i,:) = [];
                    list2 = vertcat(list2,append);
                    q = q+1;
                    m = m+1;
                end
            end
        end
    end
end

list2(:,1) = [];
list2(:,1) = [];

%Determine which netlist includes both the input and output sources (used to
%decide which netlist is used in the SC model)-----------------------------
if list1(1,3) ~= 0 & list1(2,3) ~= 0
    list = list1;
elseif list2(1,3) ~= 0 & list2(2,3) ~= 0
    list = list2;
end

%Construct input voltage vector for SCC model-----------------------------
input = zeros(2,1);
```

```
input(1,1) = list(1,3);
input(2,1) = list(2,3);

%Construct capacitor matrix for SCC model
Cmat = zeros(capnum,capnum);

for i = 1:capnum
    Cmat(i,i) = list(i+2,3);
end


%************************************************************************
%1st phase calculation-------------------------------------------------
%************************************************************************

%Determine if any sources are not connected to circuit-----------------
sdis = 0;
for i = 1:2
    if sum(list1(i,:)) == 0
        sdis = sdis+1;
        list1(i,:) = []; %Remove the source from the list
    end
end

%Calculate number of nodes and branches--------------------------------
branch = length(list1);
node = max(list1(:,1));

%Construct Aa matrix---------------------------------------------------
Aa = zeros(node,branch);

for i = 1:branch
    Aa(list1(i,1),i) = 1;
    Aa(list1(i,2),i) = -1;
end

%Row reduce Aa matrix
Aa = rref(Aa);

%Find Ahat matrix------------------------------------------------------
Atilda = Aa;
Atilda(node,:) = [];

Ahat = Atilda;
j=1;
k=0;
for i = 1:branch
    if sum(abs(Ahat(:,j)))<=1
        Ahat(:,j) = [];
    else
        j = j+1;
        col(j-1) = i;
    end
end

%Re-order Netlist to reflect column swapping performed in Ahat calculation-
for i = 1:length(col)
    if i <= 1
        append = list1(col(i),:);
        list1(col(i),:) = [];
        list1 = vertcat(list1,append);
    else
        append = list1(col(i)-i+1,:);
```

```
        list1(col(i)-i+1,:) = [];
        list1 = vertcat(list1,append);
    end
end

%Calculate loop matrix-----------------------------------------------------
loopnum = (branch - node)+1;
Bbtranspose = vertcat(-Ahat,eye(loopnum));

Bb = Bbtranspose';

%Determine the number of KCL equations needed-----------------------------
kclnum = capnum-length(Bb(:,1));

%Determine which rows of Bbtranspose are the same as the rows of cap curr--
icap = zeros(capnum,branch);
k=1;
for i = 3-sdis:capnum+2-sdis
    for j = 1:length(list1)
        if Bbtranspose(j,:) == Bbtranspose(i,:)
            icap(k,j) = 1;
        elseif Bbtranspose(j,:) == -Bbtranspose(i,:)
            icap(k,j) = -1;
        end
    end
    k = k+1;
end

%Determine KCL equations---------------------------------------------------
%Determine which independent currents are not capacitor currents, but are
%dependent of capacitor currents
i_dependent = zeros(1,1);
j=1;
for i = 3-sdis:length(icap)-sdis
    if icap(:,i) == 0
        i_dependent(j,1) = i;
        j = j+1;
    end
end

%Figure out which column these currents correspond to
Bbcol = zeros(length(i_dependent),1);
k=1;
for i = 1:length(i_dependent)
    for j = 1:loopnum
        if i_dependent(i,1) ~= 0
            if Bbtranspose(i_dependent(i,1),j) == 1
                Bbcol(k,1) = j;
            end
        end
    end
    k = k+1;
end

%Check to see if all independent currents are indeed capacitor currents
kcl = zeros(kclnum,capnum);
u=0;
for m = branch-loopnum+1:branch
    if sum(abs(icap(:,m))) >= 1
        u = u+1; %store u value for E matrix construction & KCL equations
    end
end
```

```
m=1;
g=1;
if u == loopnum  %If all indep. curr. are cap curr., KCL is found directly
    for i = 3-sdis:capnum+2-sdis
        if sum(abs(Bbtranspose(i,:))) > 1
            kcl(m,i-2+sdis) = -1;
            for l = 1:loopnum
                if Bbtranspose(i,l) ~= 0
                    for j = 1:capnum
                        if icap(j,branch-loopnum+l) ~= 0
                            kcl(m,j) = Bbtranspose(i,l)*icap(j,branch-lo...
                                opnum+l);
                            g = g+1;
                            break
                        end
                    end
                end
            end
        end
        m = m+1;
    end
    if g ~= kclnum + 1 %Creates KCL eq. if 2 cap curr. are same or opposite
        for i = 3-sdis:capnum+2-sdis
            if g <= kclnum
                for j = 3-sdis:capnum+2-sdis
                    if Bbtranspose(i,:) == -Bbtranspose(j,:)
                        kcl(g,i-2+sdis) = 1;
                        kcl(g,j-2+sdis) = 1;
                        g = g+1;
                    end
                end
            end
        end
    end
else %If they are not all cap currents, KCL is not so direct
    for i = 3-sdis:capnum+2-sdis
        k = 0;
        for j = 1:length(Bbcol)
            if Bbtranspose(i,Bbcol(j,1)) == 0
            else
                k = k+1;
            end
        end
        if k == 0
            if sum(abs(Bbtranspose(i,:))) >= 2
                kcl(m,i-2+sdis) = -1;
                m = m+1;
                for j = 1:loopnum
                    g = 1;
                    if abs(Bbtranspose(i,j)) == 1
                        for r = 1:capnum
                            if icap(r,branch-loopnum+j) ~= 0
                                kcl(m-1,g) = Bbtranspose(i,j);
                            else
                                g = g+1;
                            end
                        end
                    end
                end
            end
        end
    end
end
```

```
%Check to see if KCL matrix is full
y=0;
for i = 1:kclnum
    if sum(abs(kcl(i,:))) == 0
        y = y+1;
    end
end

%If y does not equal to zero, then find other kcl equations
m=1;
g=1;
if y ~= 0
    while g <= y
        for i = 3+m-1-sdis:capnum+2-sdis
            k = 0;
            for j = 1:loopnum
                if Bbtranspose(m+2-sdis,j) == -Bbtranspose(i,j)
                    k = k+1;
                else
                    if j ~= Bbcol(:,1)
                        k = k+1;
                    end
                end
            end
            if k == loopnum
                for l = 1:loopnum
                    if Bbtranspose(m+2-sdis,l) ~= -Bbtranspose(i,l)
                        for r = 1:capnum
                            if icap(r,branch-loopnum+l) ~= 0
                                kcl(kclnum-y+g,m) = -1;
                                kcl(kclnum-y+g,i-2+sdis) = -1;
                                if Bbtranspose(m+2-sdis,l) ~= 0
                                    kcl(kclnum-y+g,r) = Bbtranspose(m+2-...
                                        sdis,l);
                                else
                                    kcl(kclnum-y+g,r) = Bbtranspose(i,l);
                                end
                            end
                        end
                    end
                end
                g = g+1;
            end
        end
        m = m+1;
    end
end

%Generate E1 matrix-------------------------------------------------------
E1 = zeros(loopnum,capnum);

%Figure out which rows of Bbtranspose are cap currents
Esetup = zeros(branch-2-capnum,1);
for i = 2+capnum+1-sdis:branch
    for j = 1:capnum
        if icap(j,i) == 1
            Esetup(i-2-capnum+sdis,1) = j;
        elseif icap(j,i) == -1
            Esetup(i-2-capnum+sdis,1) = -j;
        end
    end
end
```

```
%Shorten list and Bb matrices b/c we don't need to worry about cap currents
%or the currents through the sources
Bbtrunk = Bb;
listtrunk = list1;

for i = 1:2+capnum-sdis
    listtrunk(1,:) = [];
end
for i = 1:2
    listtrunk(:,1) = [];
end

for i = 1:2+capnum-sdis
    Bbtrunk(:,1) = [];
end
Bbtrunk = Bbtrunk';

%Make the list matrix the same number of columns as the Bbtrunk matrix
listtrunkm = listtrunk;
for i = 1:loopnum-1
    listtrunkm = horzcat(listtrunkm,listtrunk);
end

%Multiply the matrices to store the actual KVL loop values
loopval = Bbtrunk.*listtrunkm;

%Begin construction of E matrix: This step does not construct the full E
%matrix if there are currents that are combinations of capacitor currents.
%It only places the resistor values that correspond to actual
%capacitor currents. If there are resistors whose currents are a
%combination of cap currents, such as some MOSFET currents in the ladder
%topology, they will be entered next
for i = 1:branch-2-capnum+sdis
    for j = 1:loopnum
        if Esetup(i,1) > 0
            E1(j,abs(Esetup(i,1))) = E1(j,abs(Esetup(i,1)))+loopval(i,j);
        elseif Esetup(i,1) < 0
            E1(j,abs(Esetup(i,1))) = E1(j,abs(Esetup(i,1)))-loopval(i,j);
        end
    end
end

%So far, E matrix is incomplete if there are currents that can only be
%expressed as a function of other cap currents. For example, for a ladder
%topology, now inlude the MOSFET resistor values whose currents cannot be
%expressed as a single capacitor current
m=1;
e=1;
if i_dependent(:,1) ~= 0 %if there are no dependent cap curr., skip script
    if u == loopnum %use this script if all indepen. curr. are cap currents
        for i = 1:loopnum
            for j = 1:length(i_dependent)
                if Bb(i,i_dependent(j,1)) ~= 0
                    for g = 1:loopnum
                        if Bbtranspose(i_dependent(j,1),g) ~= 0
                            for r = 1:capnum
                                if icap(r,branch-loopnum+g) ~= 0
                                    E1(m,r) = E1(m,r)+Bbtranspose(i_depe...
                                        ndent(j,1),g)*Bb(i,i_dependent(j...
                                        ,1))*(list1(i_dependent(j,1),3))...
                                        *icap(r,branch-loopnum+g);
                                    break
```

```
                            end
                        end
                    end
                end
            end
        end
        m = m+1;
    end
else %use this if all independent currents are not cap currents
    for i = 1:length(Bbcol)
        m = 0;
        for j = 3:2+capnum
            if m < 1
                if abs(Bbtranspose(j,Bbcol(i,1))) == 1
                    for r = 3:2+capnum
                        k = 0;
                        if m < 1
                            if Bbtranspose(r,i) == 0
                                for g = 1:length(Bbcol)
                                    if Bbtranspose(j,Bbcol(g,1)) ~= ...
                                            -Bbtranspose(r,Bbcol(g,1))
                                        k = k+1;
                                    end
                                end
                            end
                        end
                        if k == 1
                            for f = 1:loopnum
                                if f ~= Bbcol(:,1)
                                    if Bbtranspose(j,f) ~= 0
                                        for p = 1:capnum
                                            if icap(p,branch-loopnum...
                                                    +f) ~= 0
                                                E1(e,p) = E1(e,p) - ...
                                                    Bbtranspose(j,Bb...
                                                    col(i))*Bbtransp...
                                                    ose(j,f)*list1(b...
                                                    ranch-f+1,3);
                                                break
                                            end
                                        end
                                    end
                                    if Bbtranspose(r,f) ~= 0
                                        for p = 1:capnum
                                            if icap(p,branch-loopnum...
                                                    +f) ~= 0
                                                E1(e,p) = E1(e,p) - ...
                                                    Bbtranspose(j,Bb...
                                                    col(i))*Bbtransp...
                                                    ose(r,f)*list1(b...
                                                    ranch-f+1,3);
                                                break
                                            end
                                        end
                                    end
                                end
                            end
                            E1(e,j-2) = E1(e,j-2) + Bbtranspose(j,Bb...
                                col(i))*list1(branch-f+1,3);
                            E1(e,r-2) = E1(e,r-2) + Bbtranspose(j,Bb...
                                col(i))*list1(branch-f+1,3);
                            m = m+1;
                        end
```

```
                    end
                end
            end
        end
        e = e+1;
    end
end
end

E1 = vertcat(E1,kcl);

%Generate F1 matrix-------------------------------------------------------
F1 = zeros(capnum,capnum);

for j = 1:capnum-kclnum
    k = 1;
    for i = 3-sdis:capnum+2-sdis
        F1(j,k) = Bb(j,i);
        k = k+1;
    end
end

%Generate G1 matrix-------------------------------------------------------
G1 = zeros (capnum,2);

for j = 1:capnum-kclnum
    for i = 1:2-sdis
        G1(j,i) = Bb(j,i);
    end
end

%**************************************************************************
%2nd phase calculation----------------------------------------------------
%**************************************************************************

%Determine if any sources are not connected to circuit--------------------
sdis = 0;
for i = 1:2
    if sum(list2(i,:)) == 0
        sdis = sdis+1;
        list2(i,:) = []; %Remove the source from the list
    end
end

%Calculate number of nodes and branches-----------------------------------
branch = length(list2);
node = max(list2(:,1));

%Construct Aa matrix------------------------------------------------------
Aa = zeros(node,branch);

for i = 1:branch
    Aa(list2(i,1),i) = 1;
    Aa(list2(i,2),i) = -1;
end

%Row reduce Aa matrix
Aa = rref(Aa);

%Find Ahat matrix---------------------------------------------------------
Atilda = Aa;
Atilda(node,:) = [];
```

```
Ahat = Atilda;
j=1;
k=0;
for i = 1:branch
    if sum(abs(Ahat(:,j)))<=1
        Ahat(:,j) = [];
    else
        j = j+1;
        col(j-1) = i;
    end
end

%Re-order Netlist to reflect column swapping performed in Ahat calculation-
for i = 1:length(col)
    if i <= 1
        append = list2(col(i),:);
        list2(col(i),:) = [];
        list2 = vertcat(list2,append);
    else
        append = list2(col(i)-i+1,:);
        list2(col(i)-i+1,:) = [];
        list2 = vertcat(list2,append);
    end
end

%Calculate loop matrix-----------------------------------------------------
loopnum = (branch - node)+1;
Bbtranspose = vertcat(-Ahat,eye(loopnum));

Bb = Bbtranspose';

%Determine the number of KCL equations needed------------------------------
kclnum = capnum-length(Bb(:,1));

%Determine which rows of Bbtranspose are the same as the rows of cap curr--
icap = zeros(capnum,branch);
k=1;
for i = 3-sdis:capnum+2-sdis
    for j = 1:length(list2)
        if Bbtranspose(j,:) == Bbtranspose(i,:)
            icap(k,j) = 1;
        elseif Bbtranspose(j,:) == -Bbtranspose(i,:)
            icap(k,j) = -1;
        end
    end
    k = k+1;
end

%Determine KCL equations---------------------------------------------------
%Determine which independent currents are not capacitor currents, but are
%dependent of capacitor currents
i_dependent = zeros(1,1);
j=1;
for i = 3-sdis:length(icap)-sdis
    if icap(:,i) == 0
        i_dependent(j,1) = i;
        j = j+1;
    end
end

%Figure out which column these currents correspond to
Bbcol = zeros(length(i_dependent),1);
k=1;
```

```
for i = 1:length(i_dependent)
    for j = 1:loopnum
        if i_dependent(i,1) ~= 0
            if Bbtranspose(i_dependent(i,1),j) == 1
                Bbcol(k,1) = j;
            end
        end
    end
    k = k+1;
end

%Check to see if all independent currents are indeed capacitor currents
kcl = zeros(kclnum,capnum);
u=0;
for m = branch-loopnum+1:branch
    if sum(abs(icap(:,m))) >= 1
        u = u+1; %store u value for E matrix construction & KCL equations
    end
end

m=1;
g=1;
if u == loopnum  %If all indep. curr. are cap curr., KCL is found directly
    for i = 3-sdis:capnum+2-sdis
        if sum(abs(Bbtranspose(i,:))) > 1
            kcl(m,i-2+sdis) = -1;
            for l = 1:loopnum
                if Bbtranspose(i,l) ~= 0
                    for j = 1:capnum
                        if icap(j,branch-loopnum+l) ~= 0
                            kcl(m,j) = Bbtranspose(i,l)*icap(j,branch-lo...
                                opnum+l);
                            g = g+1;
                            break
                        end
                    end
                end
            end
        end
        m = m+1;
    end
    if g ~= kclnum + 1 %Creates KCL eq. if 2 cap curr. are same or opposite
        for i = 3-sdis:capnum+2-sdis
            if g <= kclnum
                for j = 3-sdis:capnum+2-sdis
                    if Bbtranspose(i,:) == -Bbtranspose(j,:)
                        kcl(g,i-2+sdis) = 1;
                        kcl(g,j-2+sdis) = 1;
                        g = g+1;
                    end
                end
            end
        end
    end
else %If they are not all cap currents, KCL is not so direct
    for i = 3-sdis:capnum+2-sdis
        k = 0;
        for j = 1:length(Bbcol)
            if Bbtranspose(i,Bbcol(j,1)) == 0
            else
                k = k+1;
            end
        end
```

```
        if k == 0
            if sum(abs(Bbtranspose(i,:))) >= 2
                kcl(m,i-2+sdis) = -1;
                m = m+1;
                for j = 1:loopnum
                    g = 1;
                    if abs(Bbtranspose(i,j)) == 1
                        for r = 1:capnum
                            if icap(r,branch-loopnum+j) ~= 0
                                kcl(m-1,g) = Bbtranspose(i,j);
                            else
                                g = g+1;
                            end
                        end
                    end
                end
            end
        end
    end
end

%Check to see if KCL matrix is full
y=0;
for i = 1:kclnum
    if sum(abs(kcl(i,:))) == 0
        y = y+1;
    end
end

%If y does not equal to zero, then find other kcl equations
m=1;
g=1;
if y ~= 0
    while g <= y
        for i = 3+m-1-sdis:capnum+2-sdis
            k = 0;
            for j = 1:loopnum
                if Bbtranspose(m+2-sdis,j) == -Bbtranspose(i,j)
                    k = k+1;
                else
                    if j ~= Bbcol(:,1)
                        k = k+1;
                    end
                end
            end
            if k == loopnum
                for l = 1:loopnum
                    if Bbtranspose(m+2-sdis,l) ~= -Bbtranspose(i,l)
                        for r = 1:capnum
                            if icap(r,branch-loopnum+l) ~= 0
                                kcl(kclnum-y+g,m) = -1;
                                kcl(kclnum-y+g,i-2+sdis) = -1;
                                if Bbtranspose(m+2-sdis,l) ~= 0
                                    kcl(kclnum-y+g,r) = Bbtranspose(m+2-...
                                        sdis,l);
                                else
                                    kcl(kclnum-y+g,r) = Bbtranspose(i,l);
                                end
                            end
                        end
                    end
                end
                g = g+1;
```

```
            end
        end
        m = m+1;
    end
end

%Generate E2 matrix-------------------------------------------------------
E2 = zeros(loopnum,capnum);

%Figure out which rows of Bbtranspose are cap currents
Esetup = zeros(branch-2-capnum,1);
for i = 2+capnum+1-sdis:branch
    for j = 1:capnum
        if icap(j,i) == 1
            Esetup(i-2-capnum+sdis,1) = j;
        elseif icap(j,i) == -1
            Esetup(i-2-capnum+sdis,1) = -j;
        end
    end
end

%Shorten list and Bb matrices b/c we don't need to worry about cap currents
%or the currents through the sources
Bbtrunk = Bb;
listtrunk = list2;

for i = 1:2+capnum-sdis
    listtrunk(1,:) = [];
end
for i = 1:2
    listtrunk(:,1) = [];
end

for i = 1:2+capnum-sdis
    Bbtrunk(:,1) = [];
end
Bbtrunk = Bbtrunk';

%Make the list matrix the same number of columns as the Bbtrunk matrix
listtrunkm = listtrunk;
for i = 1:loopnum-1
    listtrunkm = horzcat(listtrunkm,listtrunk);
end

%Multiply the matrices to store the actual KVL loop values
loopval = Bbtrunk.*listtrunkm;

%Begin construction of E matrix: This step does not construct the full E
%matrix if there are currents that are combinations of capacitor currents.
%It only places the resistor values that correspond to actual
%capacitor currents. If there are resistors whose currents are a
%combination of cap currents, such as some MOSFET currents in the ladder
%topology, they will be entered next
for i = 1:branch-2-capnum+sdis
    for j = 1:loopnum
        if Esetup(i,1) > 0
            E2(j,abs(Esetup(i,1))) = E2(j,abs(Esetup(i,1)))+loopval(i,j);
        elseif Esetup(i,1) < 0
            E2(j,abs(Esetup(i,1))) = E2(j,abs(Esetup(i,1)))-loopval(i,j);
        end
    end
end
```

```
%So far, E matrix is incomplete if there are currents that can only be
%expressed as a function of other cap currents. For example, for a ladder
%topology, now inlude the MOSFET resistor values whose currents cannot be
%expressed as a single capacitor current
m=1;
e=1;
if i_dependent(:,1) ~= 0 %if there are no dependent cap curr., skip script
    if u == loopnum %use this script if all indepen. curr. are cap currents
        for i = 1:loopnum
            for j = 1:length(i_dependent)
                if Bb(i,i_dependent(j,1)) ~= 0
                    for g = 1:loopnum
                        if Bbtranspose(i_dependent(j,1),g) ~= 0
                            for r = 1:capnum
                                if icap(r,branch-loopnum+g) ~= 0
                                    E2(m,r) = E2(m,r)+Bbtranspose(i_depe...
                                        ndent(j,1),g)*Bb(i,i_dependent(j...
                                        ,1))*(list2(i_dependent(j,1),3))...
                                        *icap(r,branch-loopnum+g);
                                    break
                                end
                            end
                        end
                    end
                end
            end
            m = m+1;
        end
    else %use this if all independent currents are not cap currents
        for i = 1:length(Bbcol)
            m = 0;
            for j = 3:2+capnum
                if m < 1
                    if abs(Bbtranspose(j,Bbcol(i,1))) == 1
                        for r = 3:2+capnum
                            k = 0;
                            if m < 1
                                if Bbtranspose(r,i) == 0
                                    for g = 1:length(Bbcol)
                                        if Bbtranspose(j,Bbcol(g,1)) ~= ...
                                                -Bbtranspose(r,Bbcol(g,1))
                                            k = k+1;
                                        end
                                    end
                                end
                            end
                            if k == 1
                                for f = 1:loopnum
                                    if f ~= Bbcol(:,1)
                                        if Bbtranspose(j,f) ~= 0
                                            for p = 1:capnum
                                                if icap(p,branch-loopnum...
                                                        +f) ~= 0
                                                    E2(e,p) = E2(e,p) - ...
                                                        Bbtranspose(j,Bb...
                                                        col(i))*Bbtransp...
                                                        ose(j,f)*list2(b...
                                                        ranch-f+1,3);
                                                    break
                                                end
                                            end
                                        end
                                        if Bbtranspose(r,f) ~= 0
```

```
                                        for p = 1:capnum
                                            if icap(p,branch-loopnum...
                                                    +f) ~= 0
                                                E2(e,p) = E2(e,p) - ...
                                                    Bbtranspose(j,Bb...
                                                    col(i))*Bbtransp...
                                                    ose(r,f)*list2(b...
                                                    ranch-f+1,3);
                                                break
                                            end
                                        end
                                    end
                                end
                            end
                            E2(e,j-2) = E2(e,j-2) + Bbtranspose(j,Bb...
                                col(i))*list2(branch-f+1,3);
                            E2(e,r-2) = E2(e,r-2) + Bbtranspose(j,Bb...
                                col(i))*list2(branch-f+1,3);
                            m = m+1;
                        end
                    end
                end
            end
            e = e+1;
        end
    end
end

E2 = vertcat(E2,kcl);

%Generate F2 matrix-------------------------------------------------------
F2 = zeros(capnum,capnum);

for j = 1:capnum-kclnum
    k = 1;
    for i = 3-sdis:capnum+2-sdis
        F2(j,k) = Bb(j,i);
        k = k+1;
    end
end

%Generate G2 matrix-------------------------------------------------------
G2 = zeros (capnum,2);

for j = 1:capnum-kclnum
    for i = 1:2-sdis
        G2(j,i) = Bb(j,i);
    end
end

%*************************************************************************
%SCC Model----------------------------------------------------------------
%*************************************************************************
C = list1(3,3);

A1 = -(Cmat^-1)*(E1^-1)*F1;
B1 = -(Cmat^-1)*(E1^-1)*G1;
C1 = zeros(1,capnum);
C1(1,capnum) = 1;

A2 = -(Cmat^-1)*(E2^-1)*F2;
B2 = -(Cmat^-1)*(E2^-1)*G2;
```

```
C2 = zeros(1,capnum);
C2(1,capnum) = 1;

sys1 = ss(A1,B1,C1,[0,0]);
sys2 = ss(A2,B2,C2,[0,0]);

D1 = d1;
D2 = d2;

Vdrop = input(1)*gain - input(2);

clear freq T t1 t2 current Req
for i = 1:90
    freq(i) = 10000*10^(i/30);
    T(i) = 1/freq(i);
    t1(i) = D1*T(i);
    t2(i) = D2*T(i);
    sys1d = c2d(sys1,t1(i));
    sys2d = c2d(sys2,t2(i));

    [phi1,gamma1,cj,dj,Tsj] = ssdata(sys1d);
    [phi2,gamma2,cj,dj,Tsj] = ssdata(sys2d);

    phi = phi2 * phi1;
    gamma = phi2*gamma1 + gamma2;

    x0 = (eye(capnum)-phi)^-1*gamma*input;
    x1 = phi1*x0+gamma1*input;
    dv = (x1(capnum)-x0(capnum));
    current(i) = list(capnum+2-sdis,3)*dv/T(i);

    Req(i) = Vdrop/current(i);
end

Rlowfreqonecell = 2*D1*T/C;
topologygain = Req(gain)/Rlowfreqonecell(gain);
Rlowfreq = Rlowfreqonecell * topologygain;

figure(1)
loglog(freq,Req,'b-')
xlabel('Frequency [Hz]')
ylabel('Req [\Omega]')
title('SCC Model (D = 0.45)')
hold on

Rmin = min(Req);
```

BIBLIOGRAPHY

[1] V. W. Ng, M. D. Seeman, and S. R. Sanders, "Minimum PCB Footprint Point-of-Load DC-DC Converter Realized with Switched-Capacitor Architecture," *IEEE Energy Conversion Congress and Exposition,* pp. 1575-1581, 2009.

[2] A. Ioinovici, "Switched-Capacitor Power Electronics Circuits," *IEEE Circuits and Systems Magazine,* vol. 1, pp. 37-42, 2001.

[3] F. Z. Peng, F. Zhang, and Z. Qian, "A Magnetic-Less DC-DC Converter for Dual-Voltage Automotive Systems," *IEEE Transactions on Industry Applications,* vol. 39, pp. 511-518, 2003.

[4] D. Cao and F. Z. Peng, "Zero-Current-Switching Multilevel Modular Switched-Capacitor DC-DC Converter," *IEEE Energy Conversion Congress and Exposition,* pp. 3516 - 3522 2009.

[5] S. Ben-Yaakov and A. Kushnerov, "Algebraic Foundation of Self Adjusting Switched Capacitor Converters," *Energy Conversion Congress and Exposition,* pp. 1582-1589, 2009.

[6] C.-H. Hu and L.-K. Chang, "Analysis and Modeling of On-Chip Charge Pump Designs Based on Pumping Gain Increase Circuits With a Resistive Load," *IEEE Transactions on Power Electronics,* vol. 23, pp. 2187-2194, 2008.

[7] M.-H. Huang, P.-C. Fan, and K.-H. Chen, "Low-Ripple and Dual-Phase Charge Pump Circuit Regulated by Switched-Capacitor-Based Bandgap Reference," *IEEE Transactions on Power Electronics,* vol. 24, pp. 1161-1172, 2009.

[8] G. Thiele and E. Bayer, "Voltage Doubler/Tripler Current-Mode Charge Pump Topology with Simple Gear Box", *Power Electronics Specialists Converence,* pp. 2348-2352, 2007.

[9] F. Zhang, L. Du, F. Z. Peng, and Z. Qian, "A New Design Method for High Efficiency DC-DC Converters with Flying Capacitor Technology," *Applied Power Electronics Conference and Exposition,* 2006.

[10] F. Zhang, L. Du, F. Z. Peng, and Z. Qian, "A New Design Method for High-Power High-Efficiency Switched-Capacitor DC-DC Converters," *IEEE Transactions on Power Electronics,* vol. 23, pp. 832-840, 2008.

[11] J. W. Kimball and P. T. Krein, "Analysis and Design of Switched Capacitor Converters," *Applied Power Electronics Conference and Exposition,* vol. 3, pp. 1473-1477, 2005.

[12] J. W. Kimball, P. T. Krein, and K. R. Cahill, "Modeling of Capacitor Impedance in Switching Converters," *IEEE Power Electronics Letters,* vol. 3, pp. 136-140, 2005.

[13] M. D. Seeman, "Analytical and Practical Analysis of Switched-Capacitor DC-DC Converters," Ph. D. dissertation, University of California, Berkeley, 2006.

[14] M. D. Seeman and S. R. Sanders, "Analysis and Optimization of Switched-Capacitor DC-DC Converters," *IEEE Workshops on Computers in Power Electronics,* 2006.

[15] M. D. Seeman and S. R. Sanders, "Analysis and Optimizaiton of Switched-Capacitor DC-DC Converters," *IEEE Transactions on Power Electronics,* vol. 23, pp. 841-851, 2008.

[16] S. Ben-Yaakov and M. Evzelman, "Generic and Unified Model of Switched Capacitor Converters," *Energy Conversion Congress and Exposition,* pp. 3501-3508, 2009.

[17] M. S. Makowski and D. Maksimovic, "Performance Limits of Switched-Capacitor DC-DC Converters," *Power Electronics Specialists Converence,* vol. 2, pp. 1215-1221, 1995.

[18] O. Wasynczuk and S. D. Sudhoff, "Automated State Model Generation Algorithm for Power Circuits and Systems," *IEEE Transactions on Power Systems,* vol. 11, pp. 1951-1956, 1996.

[19] A. Vladimirescu, *The Spice Book*: John Wiley & Sons, Inc., 1994.

[20] L. M. Wedepohl and L. Jackson, "Modified Nodal Analysis: An Essential Addition to Eelectrical Circuit Theory and Analysis," *Engineering Science and Education Journal,* vol. 11, pp. 84-92, 2002.

[21] G. D. Hachtel, R. K. Brayton, and F. G. Gustavson, "The Sparse Tableau Approach to Network Analysis and Design," *IEEE Transactions on Circuit Theory,* vol. 18, pp. 101-113, 1971.

[22] G. F. Franklin, J. D. Powell, and M. L. Workman, *Digital Control of Dynamic Systems*: Addison-Wesley Publishing Co., 1990.

[23] R. K. Hester, K.-S. Tan, M. De Wit, J. W. Fattaruso, S. Kiriaki, and J. R. Hellums, "Fully Differential ADC with Rail-to-Rail Common-Mode Range and Nonlinear Capacitor Compensation," *IEEE Journal of Solid-State Circuits,* vol. 25, pp. 173-183, 1990.

[24] L. Murata Manufacturing Co. (2006, January 15). *Electrical Characteristics Data 1210 X5R 22uF 25V*.

[25] L. O. Chua and P.-M. Lin, *Computer-Aided Analysis of Electronic Circuits*. Englewood Cliffs: Prentice-Hall, Inc., 1975.

[26] J. Han, A. von Jouanne, and G. C. Temes, "A New Approach to Reducing Output Ripple in Switched-Capacitor-Based Step-Down DC-DC Converters," *IEEE Transactions on Power Electronics*, vol. 21, pp. 1548-1555, 2006.

VITA

Jordan Michael Henry was born on August 13, 1986 in Saint Charles, Missouri. He received the Bachelor's of Science degree in Electrical Engineering from Missouri S&T in 2009 and the Master's of Science degree in Electrical Engineering from Missouri S&T in 2010.

Jordan has been a student member of IEEE since 2006 and a member of the engineering honor societies Eta Kappa Nu and Tau Beta Pi since 2008. Jordan was awarded the Chancellor's Fellowship in 2009 and was also a Grainger Award recipient in 2009 for academic excellence displayed in the area of power engineering.