FERRITE CHARACTERIZATION TECHNIQUES
&
PARTICLE SIMULATIONS FOR SEMICONDUCTOR DEVICES

by

NICHOLAS GARRETT ERICKSON

A DISSERTATION

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER ENGINEERING

2019

Approved by

PUBLICATION DISSERTATION OPTION

This dissertation consists of the following three articles which have been submitted for publication, or will be submitted for publication, as listed below.

Paper I: Pages 4-13, “Permeability Extraction of Layered Magnetic Material Using Keysight 16454A Fixture," has been submitted to IEEE Magnetics Letters.

Paper II: Pages 14-37, “Ferrite Sheet Characterization Using a Planar PCB Coil," has been submitted to IEEE Transactions on Power Electronics.

Paper III: Pages 38-62, “Monte Carlo Particle Simulations for Studying Through-Silicon-Via (TSV) Capacitance," will be submitted to IEEE Transactions on Electron Devices.

ABSTRACT

This dissertation is divided into three papers, covering two major topics. The first topic, techniques for ferrite characterization, is discussed over the course of two papers. The second topic, particle simulations for semiconductor devices, is discussed in the last paper. In the first paper, the method for extracting permeability from ferrite materials is discussed for the Keysight 16454A permeability extraction fixture, where the ferrite material to be characterized is assumed to be homogeneous. Then the method is updated to account for layered materials. The updated method is verified through full-wave simulations. In the second paper, a planar printed circuit board (PCB) coil is proposed as an alternative to the Keysight 16454A fixture for extracting permeability from ferrite materials. The method of extraction is verified through full-wave simulations. The final paper (and second topic) develops a particle simulator, based on the Boltzmann transport equation (BTE) and Monte Carlo (MC) methods, for studying semiconductor devices with submicron feature sizes. Particle simulations are advantageous because full-wave simulators based purely on Maxwell's equations are not able to capture certain semiconductor effects. This work specifically investigates metal-oxide-semiconductor (MOS) effects for a pair of through-silicon-vias (TSVs), and the corresponding accumulation and depletion regions formed for different bias voltages.

ACKNOWLEDGMENTS

I would like to express my sincere appreciation and gratitude to my advisor, Dr. Jun Fan, for guiding me throughout the course of both my M.S. degree and PhD degree. His advice will continue to help me for many years to come.

I would like to thank Dr. C.H. Wu for both serving as a member of my committee and for sparking my interest in electromagnetics in “Introduction to Electromagnetics" during my senior year of undergraduate study. Thanks to Dr. Daryl Beetner for serving as my undergraduate advisor and PhD committee member, and especially for getting me started with the EMC group. Thanks to Dr. James Drewniak for serving as a committee member and for mentoring me during my first summer at the EMC lab. Thanks to Dr. Brice Achkir for serving as a committee member and for mentoring me during my first company sponsored project in the EMC lab. Thanks to all members of the EMC family, both new and old, for your friendship, advice, assistance, and teamwork for all lab related activities.

Special thanks to my "Rolla friends". I definitely would have never made it through this degree without your support, encouragement, and friendship. Also, special thanks to my parents, my sisters, and my brother, for their endless love and support in all my life’s pursuits. Finally, thanks to God for making all this possible.

This work was supported in part by the National Science Foundation (NSF) under Grant No. IIP-1440110. The support is much appreciated.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

SECTION

1. INTRODUCTION

1.1. FERRITE CHARACTERIZATION TECHNIQUES

Many consumer products now include near-field communication capabilities and/or wireless charging capabilities. These capabilities are enabled through the use of coils that can be magnetically coupled to one another. Common products include cell phones, gaming controllers and accessories, fitness trackers, and even tooth brushes. The magnetic fields inherently produced by the coils in these products can cause interference with other nearby electronic components and circuits. To reduce this unwanted interference, ferrite sheets are used to capture the magnetic flux produced by the coils and direct it away from nearby components and circuits. However, the ability to manage magnetic flux varies with each type of ferrite sheet, depending on that sheet's specific material characteristics.

In particular, it is useful for an engineer or system designer to be able to determine the permeability of a ferrite sheet, since permeability indicates a material's ability to support a magnetic field within itself. Many methods exist for determining the permeability of magnetic materials, including commercialized setups. One such setup is the Keysight 16454A magnetic material extraction fixture. It is particulary popular for characterizing ferrites used in the above described consumer products because it is highly accurate in the frequency range of 100 kHz to 100 MHz, covering the needs of radio-frequency identification (RFID: 125-148 kHz and 13.56 MHz), near-field communication (NFC: 13.56 MHz), and wireless power transfer (WPT: 110-205 kHz and 80-300 kHz) chargers based on the Qi standard.

In Paper I, the theory of the Keysight 16454A fixture is discussed. It is shown that the extraction equation for the fixture is only applicable to single-layered homogeneous materials. The work in Paper I proposes an updated extraction equation that is applicable for layered materials. The updated equation is particularly useful for new nanocrystalline ferrites that are constructed by alternating thin layers of ferrite and adhesive materials.

In Paper II, an alternative fixture to the Keysight 16454A fixture is proposed. Generally, fixtures used for low-frequency permeability extraction are based on geometries with known analytical relationships between inductance and permeability. This is the case for the Keysight fixture, since it is a single-turn toroid with a known analytical relationship between impedance and permeability. The fixture proposed in this work is a simple planar coil on a two layer printed circuit board (PCB). It does not have an analytical relationship between impedance and permeability. Instead, full-wave simulations are used to build the relationship. Permeability is extracted using a combination of measured impedance data, simulations, and interpolation functions. The proposed method is cheaper than purchasing the Keysight fixture and is readily available to all engineers. The proposed method also does not require the ferrite sample be shaped into a toroid; the shape can remain as a sheet.

1.2. PARTICLE SIMULATIONS FOR SEMICONDUCTOR DEVICES

Particle simulations based on the Boltzmann Transport Equation (BTE) and the Monte Carlo method are a powerful method for studying semiconductors in the nanometer to submicrometer regime. In particular, they allow semiconductor effects to be studied. Paper III discusses using particle simulations to study the voltage dependent capacitance of a through-silicon-via (TSV) pair. TSVs serve as vertical interconnects through integrated circuits (ICs), allowing chips to be stacked and thereby enabling 2.5D and 3D IC topologies. Regular PCB interconnects can be fully characterized using full-wave solvers based on Maxwell's equations since no semiconductor effects come into play. This approach, however, is not able to account for semiconductor effects that are present for IC interconnects, such as TSVs. Thus, particle simulations provide a rigorous way to analyze semiconductor effects for IC interconnects.

PAPER

I. PERMEABILITY EXTRACTION OF LAYERED MAGNETIC MATERIAL USING KEYSIGHT 16454A FIXTURE

Nicholas Erickson, *Student Member, IEEE*, Huilin Wu, and Jun Fan, *Fellow, IEEE*

ABSTRACT

The Keysight 16454A Magnetic Material Test Fixture is a popular choice for characterizing magnetic materials up to 1 GHz due to its simple two-step measurement procedure. The extraction is based on a simple equation, rigorously derived from electromagnetic laws. However, as given, the extraction equation is limited to a single layer of homogeneous material. In wireless power transfer (WPT) applications, nanocrystalline ferrite sheets are replacing traditional ferrite sheets. Many nanocrystalline ferrite sheets are not homogeneous, instead consisting of several alternating layers of ferrite powder and adhesive materials. In addition, the updated equation also leads to the concepts of effective permeability and effective thickness.

Keywords: magnetic instruments, soft magnetic materials

1. INTRODUCTION

Ferrite sheets are used to manage magnetic flux in a wide array of consumer products involving mangetically coupled communication technologies. Examples include products with near field communication (NFC), radio-frequency identification (RFID), or wireless power transfer (WPT) capabilities. In many of these applications, ferrite sheets are used to shield electronic components from the magnetic fields produced by the communication coils [1]. The 16454A magnetic material test fixture from Keysight is a popular choice for characterizing the permeability of ferrite sheets due to its high measurement accuracy in the 100 kHz to 100 MHz frequency range, a range that covers low and high frequency RFID (125-148 kHz, 13.56 MHz), NFC (13.56 MHz), and low and medium power WPT Qi chargers (110-205 kHz and 80-300 kHz). Traditional ferrite sheets are single layered and can use the current Keysight extraction equation directly. However, nanocrystalline ferrite sheets are growing in popularity, and are not single layered. Instead, they are made using thin, alternating layers of ferrite and adhesive materials. This work updates the extraction equation for the 16454A fixture to include this layered effect and describes how an effective permeability is extracted if the original extraction equation is used.

2. KEYSIGHT 16454A FIXTURE: ORIGINAL EXTRACTION EQUATION

The theory behind the Keysight 16454A fixture and its corresponding permeability extraction equation is fully described in [2]. Some of the main details are repeated here to aid in understanding the derivations given in the next section. The 16454A fixture creates a single-turn toroid with a rectangular cross-section, as shown in Fig. 1. The B-field for the toroid is related to the current according to Ampere's Law [3] as in (1).

$$\oint_C \vec{B} * d\vec{l} = \mu_0 \mu_r I \rightarrow B_\phi = \frac{\mu_0 \mu_r I}{2\pi r} \tag{1}$$

The inductance of the toroid can be further derived in 2. The current loop is shown in Fig. 1. The corresponding integration surface (with the ferrite material loaded into the fixture) is shown in Fig. 2. This integration surface can be broken into four pieces, where only one piece involves the unknown μ_r, as shown in Fig. 3. Breaking the integral in (2) into four pieces (one for each of the four areas), results in (3).

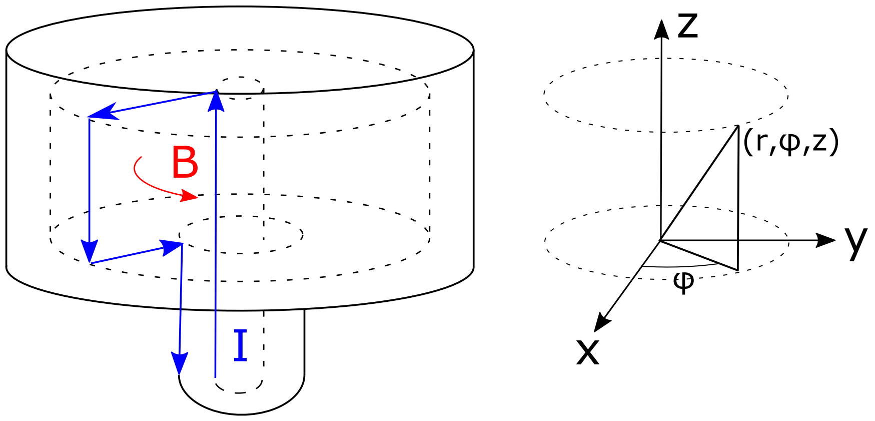

Figure 1. The Keysight 16454A fixture. The contour of integration for (1) can be taken anywhere in the cavity, along the direction indicated by the B-field. The outline of the example current path through the fixture also gives the surface necessary for the integration in (2). Cylindrical coordinates are used due to the symmetry of the structure.

$$L = \frac{\Psi}{I} = \frac{\int_S \vec{B} * d\vec{S}}{I} \rightarrow L = \int\int \frac{\mu_0\mu_r}{2\pi r} dr dz \tag{2}$$

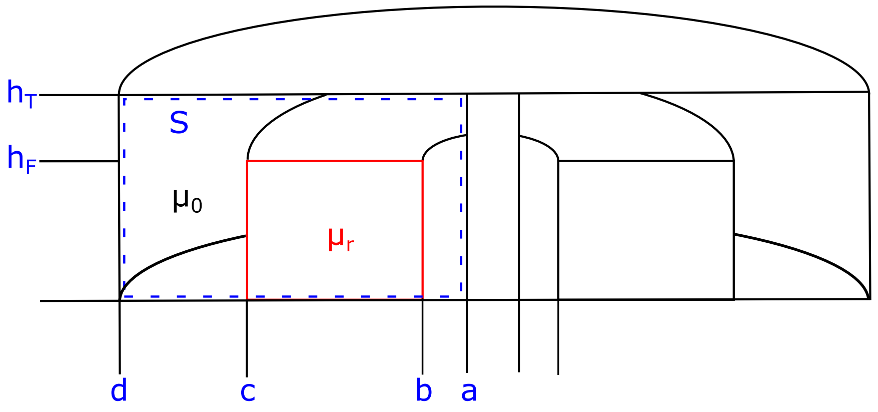

Figure 2. Integration surface for a single, homogeneous layer of ferrite material (3D View).

$$L_F = \int_0^{h_T}\int_a^b \frac{\mu_0}{2\pi r} dr dz + \int_0^{h_F}\int_b^c \frac{\mu_0\mu_R}{2\pi r} dr dz + \int_{h_F}^{h_T}\int_b^c \frac{\mu_0}{2\pi r} dr dz + \int_0^{h_T}\int_c^d \frac{\mu_0}{2\pi r} dr dz \tag{3}$$

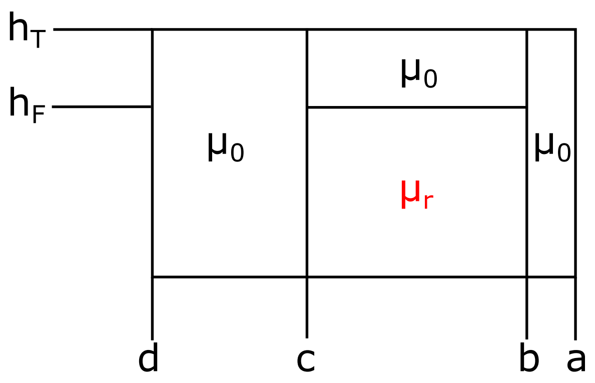

Figure 3. Integration surface for a single, homogeneous layer of ferrite material (2D View). Since the 16454A fixture is not completely filled with ferrite material, the integration surface is broken up into four different areas. Only one of the areas involves the unknown permeability, μ_r.

Performing the integrations and solving for μ_r gives (4). Note that the $\mu_0 h_T ln(\frac{d}{a})$ portion is the same as the inductance for the empty toroid, multiplied by 2π. Thus, (4) is equivalent to (5).

$$\mu_r = \frac{1}{h_F}\left[\frac{2\pi L_F - \mu_0 h_T ln(\frac{d}{a})}{\mu_0 ln(\frac{c}{b})}\right] + 1 \tag{4}$$

$$\mu_r = \frac{1}{h_F}\left[\frac{2\pi(L_F - L_E)}{\mu_0 ln(\frac{c}{b})}\right] + 1 \tag{5}$$

Let the measured impedance for the empty fixture be represented by Z_{ME}, and the measured impedance for the ferrite loaded fixture be represented by Z_{MF}. Assuming that both impedances involve a series residual impedance, Z_{res}, the residual impedance can be cancelled through subtraction, as shown in (6). Ideally, Z_F and Z_E are only composed of inductive elements, allowing for the substitution of (7) into (5), resulting in the final extraction expression of (8).

$$Z_{MF} - Z_{ME} = (Z_{res} + Z_F) - (Z_{res} + Z_E) \tag{6}$$

$$L_F - L_E = \frac{Z_{MF} - Z_{ME}}{j\omega} \tag{7}$$

$$\mu_r = \frac{1}{h_F}\left[\frac{2\pi(Z_{MF} - Z_{ME})}{j\omega\mu_0 ln(\frac{c}{b})}\right] + 1 \tag{8}$$

3. UPDATED EXTRACTION EQUATION FOR LAYERED FERRITES

As stated previously, some ferrite sheets, such as nanocrystalline ferrite sheets, are composed of alternating layers of ferrite and adhesive materials. This breaks the μ_r integration area shown in Fig. 3 into additional integration areas, as shown in Fig. 4. Note that the integrations can be customized for each specific situation. As an example, though, the μ_r derived here is for N layers of ferrite (start and stop layers), and N-1 layers of adhesive. Each respective layer is also assumed to be the same thickness (t_F for the ferrite layers and t_A for the adhesive layers). Adhesive layers are assumed to have a permeability equal to the vacuum permeability, μ_0. Now (3) is updated to (9), where the single integral in the second term has been replaced by additional integrals to account for each of the layers.

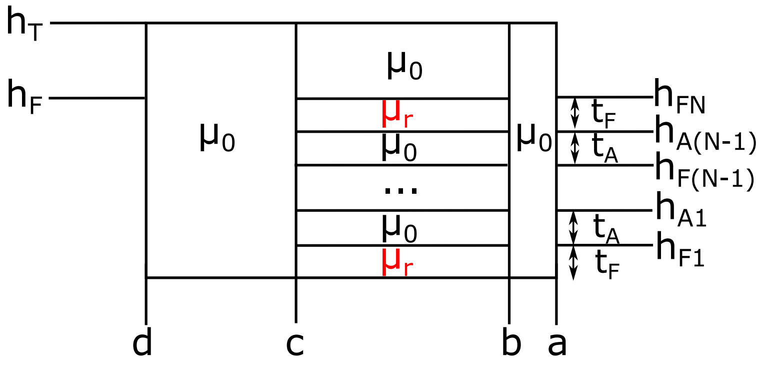

Figure 4. Integration surface for a composite material composed of alternating layers of ferrite and adhesive materials. Multiple areas involve the unknown permeability, μ_r.

$$
\begin{aligned}
L_F = & \int_0^{h_T}\int_a^b \frac{\mu_0}{2\pi r} drdz \\
& + \left[\int_0^{h_{F1}}\int_b^c \frac{\mu_0\mu_r}{2\pi r} drdz + \int_{h_{F1}}^{h_{A1}}\int_b^c \frac{\mu_0}{2\pi r} drdz + \ldots \right. \\
& \left. + \int_{h_{F(N-1)}}^{h_{A(N-1)}}\int_b^c \frac{\mu_0}{2\pi r} drdz + \int_{h_{A(N-1)}}^{h_{FN}}\int_b^c \frac{\mu_0\mu_r}{2\pi r} drdz \right] \\
& + \int_{h_F}^{h_T}\int_b^c \frac{\mu_0}{2\pi r} drdz + \int_0^{h_T}\int_c^d \frac{\mu_0}{2\pi r} drdz
\end{aligned}
\tag{9}
$$

Performing the integrations and solving for μ_r results in (10). Using the same approach as given in (5)-(7), results in the final extraction expression for layered media, as shown in (11).

$$\mu_r = \frac{1}{Nt_F}\left[\frac{2\pi L_F - \mu_0 h_T ln(\frac{d}{a})}{\mu_0 ln(\frac{c}{b})}\right] + 1 \tag{10}$$

$$\mu_r = \frac{1}{Nt_F}\left[\frac{2\pi(Z_{MF} - Z_{ME})}{j\omega\mu_0 ln(\frac{c}{b})}\right] + 1 \tag{11}$$

4. COMPARISON OF EQUATIONS

Comparing (8) and (11), the only difference is in the fractional term outside of the brackets. Further analysis gives rise to two quantities, effective permeability and effective thickness.

4.1. EFFECTIVE PERMEABILITY

Assume a layered ferrite media is measured in the 16454A fixture, but the original equation is utilized for permeability extraction, where h_F is the total height of the layered material. In this case, the extracted permeability is not the permeability of the material that makes up each ferrite layer. Instead, it is an effective permeability for the whole layered material. In other words, it can be thought of as transforming the layered material into a single layer of homogeneous material, with the same total height, h_F, as shown in Fig. 5. Utilizing both the original equation and the updated equation (with knowledge of the layered structure), one can transfer between effective permeability (combined structure) and actual permeability (of ferrite material only) using (12) and (13).

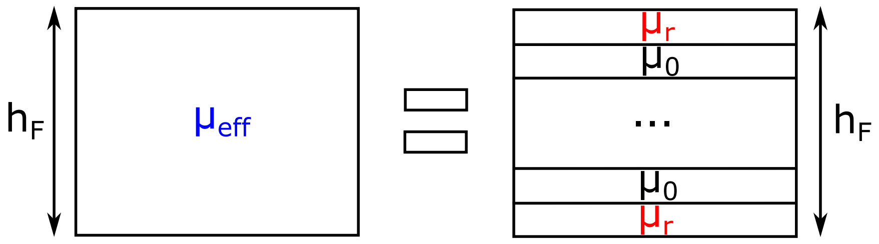

Figure 5. For the same total thickness, a single layer of homogeneous material with μ_{eff} is equivalent to a layered structure, where each ferrite layer is described by μ_r. μ_{eff} is less than μ_r as described by (12).

$$\mu_{eff} = \frac{Nt_F(\mu_r - 1) + h_F}{h_F} \tag{12}$$

$$\mu_r = \frac{h_F(\mu_{eff} - 1) + Nt_f}{Nt_F} \tag{13}$$

4.2. EFFECTIVE THICKNESS

In a similar fashion, assume again that a layered media is measured in the 16454A fixture. This time the updated equation is utilized for permeability extraction. Replacing h_F with h_{eff} in the original equation (8), and setting this equal to the updated equation (11), results in (14).

$$h_{eff} = Nt_F \tag{14}$$

This shows that the original extraction equation can also give the permeability of the ferrite material, if the material inside is assumed to be a single layer of homogeneous material with a thickness equal to only the thickness of the combined ferrite layers (adhesive layers omitted).

4.3. DISCUSSION

Most engineers and system designers are probably less concerned with the permeability of the ferrite material and more concerned with the effective permeability of the composite material (the whole ferrite sheet), since they will want to know how the whole sheet affects their design. So they can continue using the original equation, even though the sheet may be layered, to extract out effective permeability. On the other hand, ferrite manufacturers and material scientists may be more interested in using the udpated equation to get the permeability of the actual ferrite material that they are trying to characterize. For those engineers that are interested in detailed numerical simulations, the effective permeability and effective thickness concepts can be useful for simplifying simulations by transferring layered media to a single layer of homogeneous material.

5. SIMULATION VALIDATION

To validate the results presented above, the Keysight 16454A fixture was simulated in Ansys' HFSS. Depictions of the simulated structure, with and without a ferrite sample loaded, are shown in Fig. 6. The coaxial feed line was sized to be the same as an APC 7 connector, with a center pin radius of 1.5 mm and an outer shield radius of 3.5 mm. The inner cavity of the toroid section had a radius of 12 mm and a height of 9 mm. The layered ferrite sample had an inner radius of 3.2 mm, an outer radius of 7 mm, and a total thickness of 0.12 mm. In total, there were five ferrite layers (each with a thickness of 0.02 mm) and four adhesive layers (each with a thickness of 0.005 mm).

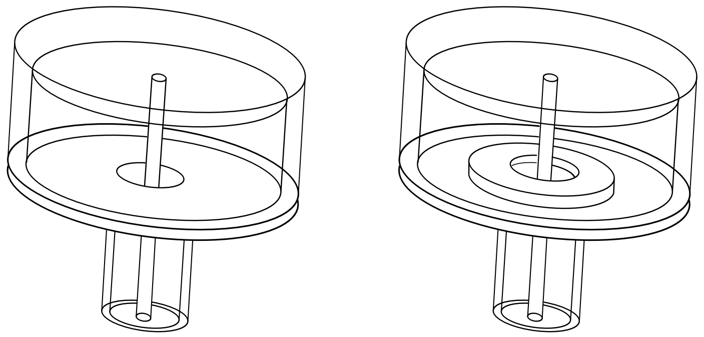

Figure 6. Depictions of HFSS simulation models for the 16454A fixture. The depiction on the left represents the empty fixture and the depiction on the right represents the fixture loaded with a ferrite sample.

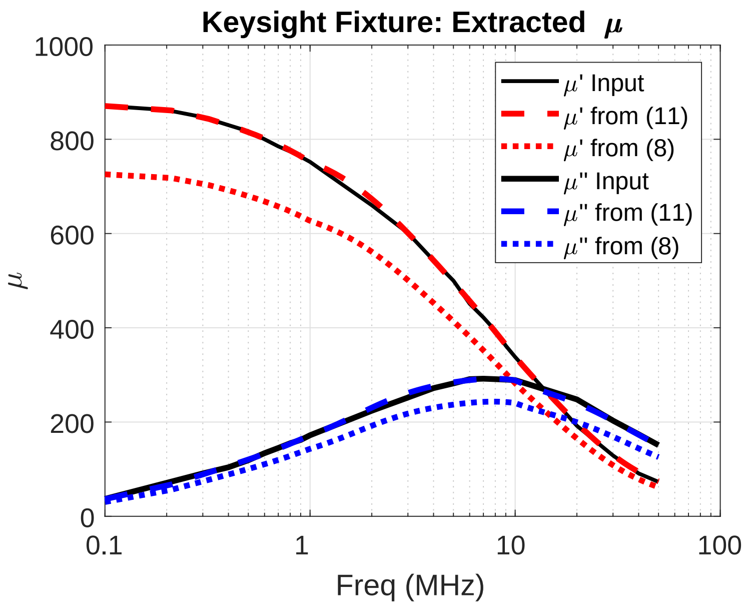

Figure 7. The extraction results from the layered ferrite simulation. Solid lines represent the input to the simulation for the material characteristics of the ferrite layers. The dashed lines represent the extraction results using the updated extraction equation (11). The dotted lines represent the extraction results using the original extraction equation (8).

In Fig. 7, the solid black lines represent the μ' and μ'' assigned to the ferrite material layers in HFSS. The dashed lines represent the results using the updated extraction equation (11). They are overlaid on the solid black lines showing that μ_r for the ferrite material is appropriately extracted. The dotted lines represent the results using the original extraciton equation (8). They show a lower value than the input permeability because they actually represent the effective permeability for the entire composite structure. As long as one result is known, the other result can be obtained using (12) or (13).

6. CONCLUSION

An updated extraction equation is provided for the Keysight 16454A magnetic material extraction fixture for layered materials and is validated through full-wave simulations. It is shown that if the original extraction equation is used for a layered material, the extracted permeability is not the permeability of the ferrite material, but an effective permeability for the composite structure. This is still likely to be the quantity sought by engineers and system designers, but may not be the quantity sought by ferrite manufacturers and material scientists. Depending of the need of the user, the appropriate equation should be utilized.

REFERENCES

[1] J. Ahuir, “Going Wireless with Magnetic Shielding," *Wurth Elektronik App. Note*, pp. 1-10, Oct. 2013.

[2] “Keysight 16454A Magnetic Material Test Fixture," *Keysight Operation and Service Manual*, Sept. 2017, Available: http://literature.cdn.keysight.com/litweb/pdf/16454-90020.pdf.

[3] U. Inan and A. Inan. *Engineering Electromagnetics*, Menlow Park, CA: Addison-Wesley, pg. 473, 1999.

II. FERRITE SHEET CHARACTERIZATION USING A PLANAR PCB COIL

Nicholas Erickson, *Student Member, IEEE*, Jingdong Sun, *Student Member, IEEE*, Gyucheol Han, *Student Member, IEEE*, Tun Li, *Member, IEEE*, Siming Pan, *Member, IEEE*, Huilin Wu, Jun Fan, *Fellow, IEEE*

ABSTRACT

Ferrite sheets are important components in wireless power transfer (WPT) systems because they help shield other electronic components in the system from the magnetic flux produced by the WPT coils. Engineers and system designers need a simple and accurate method to characterize ferrite sheets so that they can be incorporated into WPT systems without adversely affecting the system efficiency or the system performance. Several methods exist for extracting the permeability of ferrite materials, depending on the frequency range of interest. For WPT applications, a popular commercialized setup for extracting permeability is the Keysight 16454A magnetic material extraction fixture, used in conjuction with a radio-frequency (RF) impedance analyzer. This work presents a simple planar printed circuit board (PCB) coil and simulation method as an alternative to the Keysight 16454A extraction method. Full-wave simulations are used to compare the 16454A extraction method to the method proposed in this work. Further validation for the method is provided using a second coil.

Keywords: ferrite sheet, planar coil, planar inductor, permeability, wireless power transfer

1. INTRODUCTION

Currently, one of the biggest trends in consumer electronics is the incorporation of wireless charging capabilities. Ferrite sheets are an integral component of such WPT applications. Mainly, they help to shield other system components from the fields produced by the WPT coils [1]. However, in order to appropriately incorporate ferrite sheets into WPT systems, the permeability of the sheets must be accurately characterized. The unique needs and issues for characterizing ferrite sheets for WPT systems are discussed in [2].

Many methods exist for extracting the permeability characteristics of ferrites, ranging from quite simple to quite complicated. High-frequency extraction methods are usually based on s-parameter measurements of waveguide structures [3]-[6], or resonant cavities [7]. Low-frequency extraction methods are usually based on inductor geometries with known analytical equations that relate inductance to permeability. The classic examples for a long solenoid, a toroid (circular cross-section), and a coaxial cable, are given in (1)-(3), respectively.

$$L = \frac{\mu_o \mu_r N^2 A}{l} \tag{1}$$

$$L = \frac{\mu_o \mu_r N^2 A}{2\pi r_m} \tag{2}$$

$$L = \frac{\mu_o \mu_r l}{2\pi} ln \frac{b}{a} \tag{3}$$

In the equations, N is the number of turns, A is the cross-sectional area, l is the length, r_m is the mean radius of the toroid, a is the radius of the inner conductor of the coaxial cable, and b is the radius of the outer conductor of the coaxial cable. The derivations for (1)-(3) are readily available in any introductory electromagnetics textbook [8]. In fact, the Keysight 16454A extraction method [9] is based on relating permeability to the measured impedance of a toroid of rectangular cross-section. Similarly, other methods are also based on toroid based analytical equations [10]-[12]. This work focuses on low-frequency methods since the WPT Qi standard is only concerned with frequencies in the range of 87 kHz to 205 kHz [13].

The method presented in this work is similarly based on relating impedance to permeability; placing a ferrite sheet near to a planar coil will change the coil's impedance. However, in this case, there is not an analytical equation available. Instead, a set of full-wave simulations are used to determine the relationship between inductance and permeability. Compared to [14] and [15], this work relies on interpolation instead of optimization algorithms or iterative runs of simulation models. Compared to the Keysight 16454A extraction method, the proposed method is advantageous to WPT designers for several reasons. For one, the Keysight 16454A fixture is fairly expensive and may not be readily available to the engineer, whereas the simple PCB coil presented here is cheap to fabricate. Most engineers also have access to full-wave simulation softwares and vector network analyzers (VNAs), as is required for the proposed method. Secondly, the 16454A fixture requires the ferrite under test to have a toroid shape, with precise dimensions. For thin ferrites, several layers may also need to be stacked together to reach an appropriate thickness for use with the fixture. This requires extra steps on the engineer's part to prepare the ferrite sample for testing. The PCB coil, on the other hand, does not require any modifications to the shape of the ferrite; sheets are fully suitable. Finally, the simulations required by the proposed method can easily be reused for simulations throughout the WPT design stage, since the setup closely mimics the real WPT setup.

Section 2 discusses the characteristics of the ferrite that will be used to validate both the Keysight 16454A extraction method and the extraction method proposed in this work. Section 3 discusses the theory behind the Keysight 16454A fixture and extraction method and validates the method using full-wave simulations. Section 4 discusses the coil design and extraction method proposed in this work, and provides validations using full-wave simulations. Section 5 highlights issues associated with the proposed method and discusses potential solutions to those issues. Section 6 shows the results of the proposed extraction method, using measurement data obtained using the coil discussed in Section 4.2. Section 6 further provides validation of the proposed method using measurement and simulation data for a second coil. Finally, the conclusion is given in Section 5.

2. FERRITE CHARACTERISTICS

Both the Keysight 16454A extraction method and the extraction method proposed in this work are individually validated using full-wave simulation models. A ferrite with known characteristics was used as the material under test (MUT) in these models. The ferrite was 0.12 mm thick, with μ' and μ'' characteristics as shown in Fig. 1.

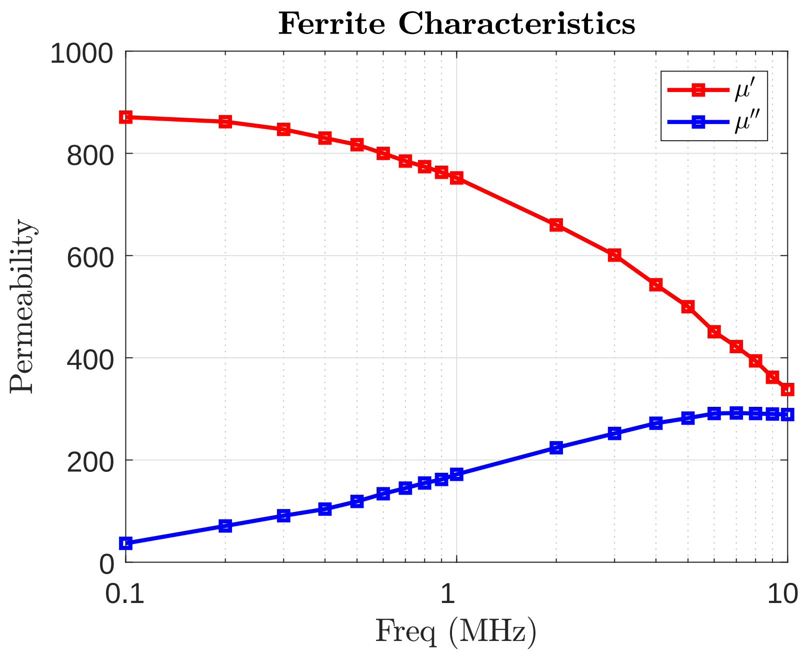

Figure 1. Permeability characteristics for the ferrite MUT.

Since HFSS [16] does not accept μ'' as an input, μ'' was transferred to tan δ, according to $tan\ \delta = \frac{\mu''}{\mu'}$. Note that ϵ_r for the ferrite was set to be 1. The effects for other values of ϵ_r are discussed in Section 5.

3. KEYSIGHT 16454A EXTRACTION METHOD

3.1. 16454A EXTRACTION THEORY

The theory for the Keysight 16454A fixture is fully described in its operation manual [9]. The main details are highlighted here for completeness. The 16454A fixture is a single-turn toroid of rectangular cross-section. The relationship between B-field and current for the toroid has a known form, given by Ampere's Law [8], as is shown in (4). Using the definition of inductance, in conjunction with the known B-field, results in (5).

$$\oint_C \vec{B} * d\vec{l} = \mu_0 \mu_r I \rightarrow B_\phi = \frac{\mu_0 \mu_r I}{2\pi r} \tag{4}$$

$$L = \frac{\Psi}{I} = \frac{\int_S \vec{B} * d\vec{S}}{I} \rightarrow L = \int \int \frac{\mu_0 \mu_r}{2\pi r} dr dz \tag{5}$$

When the fixture is loaded with a ferrite, the integration in (5) can be broken up into four pieces, as given in (6) and illustrated in Fig. 2. Computing the integrals and recognizing that $2\pi L_E = \mu_0 h_T ln(\frac{d}{a})$, gives (7), where L_E is the inductance of the empty fixture. Subtracting the measured impedance of the empty structure from the measured impedance of the ferrite loaded structure cancels any residual series impedance associated with the measurements. Assuming the remaining impedances are only composed of inductive elements, the final permeability extraction equation for the Keysight 16454A fixture is as shown in (8), where Z_{MF} and Z_{ME} are the measured impedances of the ferrite loaded fixture and the empty fixture, respectively.

$$L_F = \int_0^{h_T} \int_a^b \frac{\mu_0}{2\pi r} dr dz + \int_0^{h_F} \int_b^c \frac{\mu_0 \mu_R}{2\pi r} dr dz + \int_{h_F}^{h_T} \int_b^c \frac{\mu_0}{2\pi r} dr dz + \int_0^{h_T} \int_c^d \frac{\mu_0}{2\pi r} dr dz \tag{6}$$

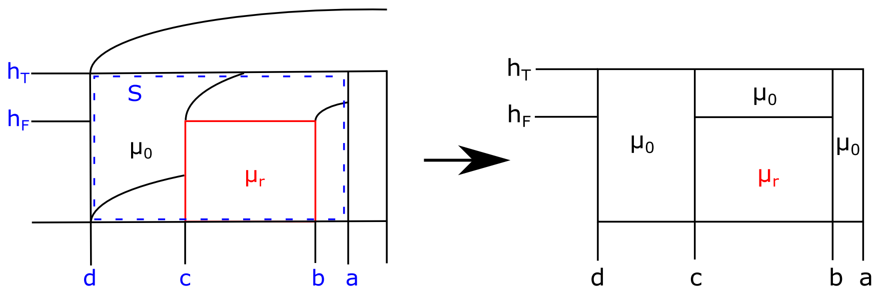

Figure 2. Integration surface for 16454A fixture, including the loaded ferrite material.

$$\mu_r = \frac{1}{h_F}\left[\frac{2\pi(L_F - L_E)}{\mu_0 ln(\frac{c}{b})}\right] + 1 \tag{7}$$

$$\mu_r = \frac{1}{h_F}\left[\frac{2\pi(Z_{MF} - Z_{ME})}{j\omega\mu_0 ln(\frac{c}{b})}\right] + 1 \tag{8}$$

3.2. 16454A SIMULATION VALIDATION

The 16454A fixture was not physically available to the authors. However, a simple simulation model was constructed in Ansys' HFSS to validate the method for comparison with the proposed method. The model consisted of a coaxial cable feeding an enclosed toroidal shaped cavity, where the signal pin of the coaxial cable extended to the top of the cavity. The feed line dimensions were the same as those of an APC 7 connector, with a center pin radius of 1.5 mm and an outer shield radius of 3.5 mm. The inner cavity had a height of 9 mm and a radius of 12 mm. The ferrite sample had an inner radius of 3.2 mm, an outer radius of 7 mm, and a thickness of 0.12 mm. Depictions of the empty and ferrite loaded model are given in Fig. 3.

The 1 port s-parameters for both the empty fixture and the ferrite loaded fixture were obtained from the HFSS models. The s-parameters were converted to z-parameters and then (8) was used to perform the permeability extraction. The results are shown in Fig. 4. In the figure, the solid black lines represent the data input into HFSS for the ferrite MUT,

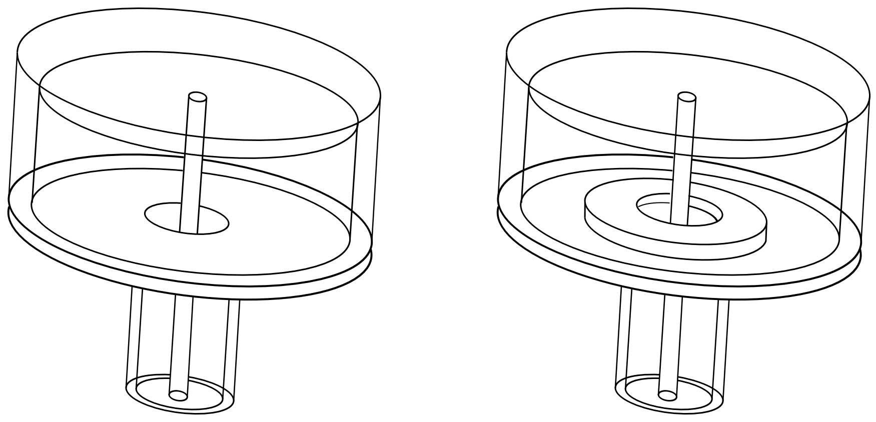

Figure 3. Depictions of the HFSS simulation models for the 16454A fixture. The depiction on the left represents the empty fixture and the depiction on the right represents the fixture loaded with a ferrite sample.

and are the same as those given in Fig. 1. The dashed lines represent the extracted results. The Keysight 16454A method is able to exactly extract the permeability of the simulated ferrite material.

4. PCB EXTRACTION METHOD

4.1. BACKGROUND

As stated previously, the basic idea for extracting permeability using a planar PCB coil is the same as that used for the Keysight 16454A fixture; the presence of the ferrite changes the impedance of the setup. However, no analytical equation is available for the planar PCB coil. So simulations must be used to relate the impedance of the ferrite and coil setup to the permeability of the ferrite. Before describing the simulation setup and method, some background information needs to be discussed first. This background information leads to an optimized setup for extracting permeability by using the planar PCB coil.

The background information is provided through a series of papers from Roshen and extended by Hurley and Duffy [17]-[20]. The main points that are applicable to this work are repeated here. The first main point is that, for a coil with an infinitely thick ferrite

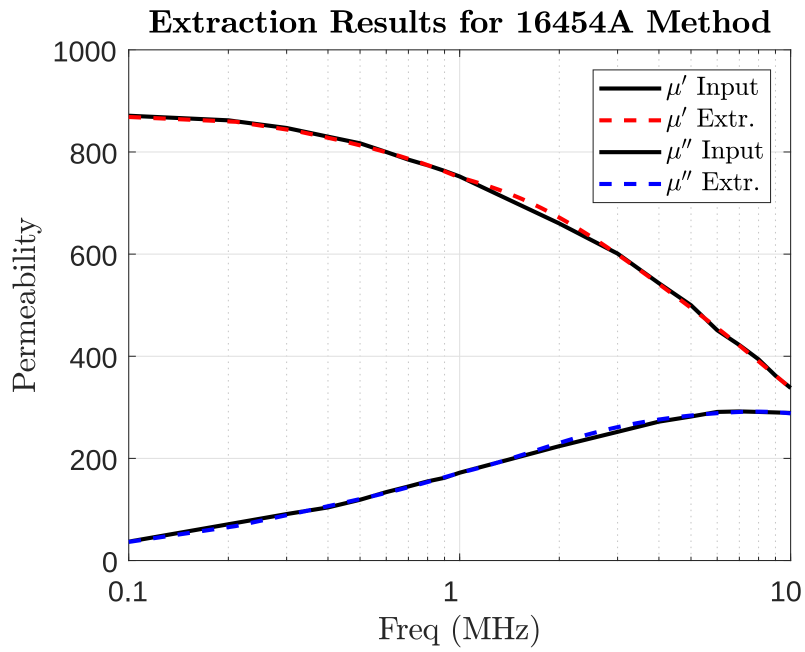

Figure 4. Simulation results for the Keysight 16454A fixture.

on one side, the maximum inductance (L_f) that can be achieved is a doubling of the air inductance (L_o), as given by (9). Doubling is achieved when μ' is equal to infinity. Next, for a finite thickness ferrite on one side of the coil [18], the inductance enhancement is reduced from the doubling limit and is dependent on the thickness of the ferrite. Roshen's results are confirmed using the coil from this work, as shown in Fig. 5. The details of the coil used in this work are given in the next section.

$$L_f = \left(\frac{2\mu'}{\mu' + 1} \right) L_o \tag{9}$$

The issue with using a single, finite thickness ferrite is that for high values of permeability, there is little change in inductance. Such a small change in inductance may be masked by measurement variability in a practical setup.

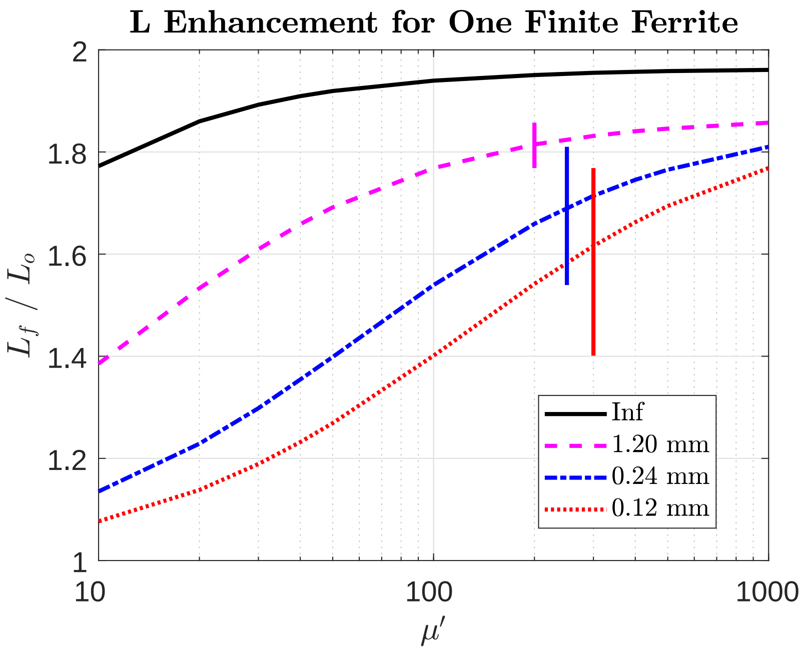

Figure 5. Inductance enhancement for one, finite thickness ferrite. The different lines represent different thicknesses of the ferrite. The vertical lines represent the dynamic range in inductance for μ' values between 100 and 1000, for each respective thickness.

Extending his work in [17] and [18], Roshen showed that for a coil sandwiched between two infinitey thick ferrites, the inductance enhancement can be more than doubled [19]. He showed that for no gap (coil within an infinite ferrite), the inductance enhancement goes as in (10).

$$L_f = \mu' L_o \tag{10}$$

For a non-zero gap (coil in the space between two infinite ferrites), the inductance enhancement is reduced from that shown in (10). Referring to Roshen's conclusion for a finite thickness ferrite, the inductance enhancement is expected to be reduced even further from (10) for the case of a coil sandwiched between two finite thickness ferrites. Nevertheless, the inductance enhancement can still be more than doubled, giving more sensitivity than the case of the single, finite thickness ferrite. Fig. 6 shows the stackup for the case of a single, finite thickness ferrite (Setup I) and the case of two, finite thickness ferrites (Setup II). For this work, Setup II is utilized since it provides a larger change in inductance for higher values of permeability, as is shown in Fig. 7.

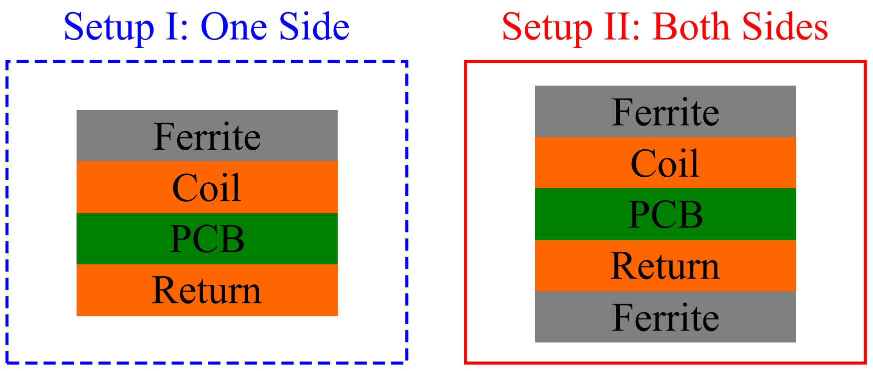

Figure 6. Two coil setups for permeability extraction. The corresponding data in Fig. 7 corresponds to sheets of 0.12 mm thickness.

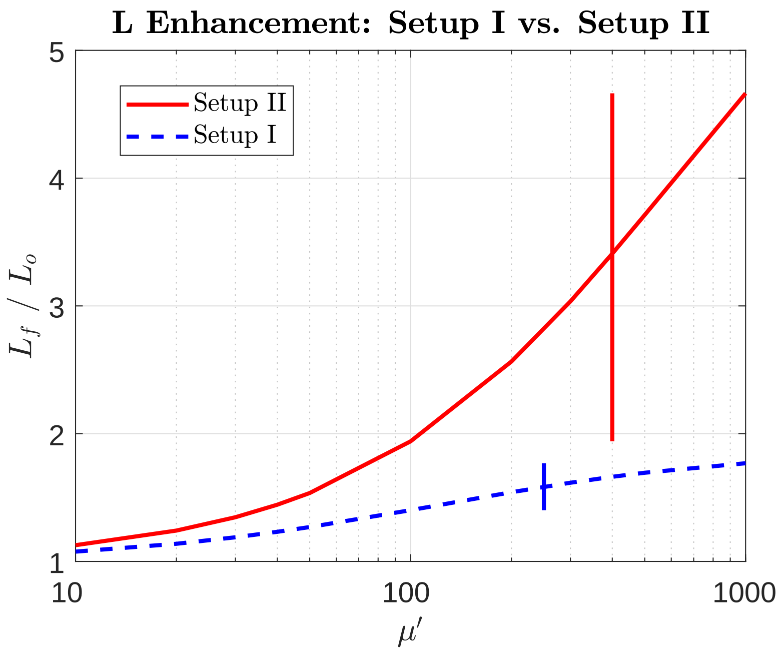

Figure 7. Inductance enhancement comparison for Setup I (single ferrite sheet) and Setup II (double ferrite sheet). Vertical lines represent the dynamic range in inductance enhancement for μ' values between 100 and 1000.

4.2. COIL DESIGN

The coil used in this work is octagonal, with an inner radius of 22.3 mm, as shown in Fig. 8. The coil consists of five turns, with each successive turn separated by 1 mm. Traces are also 1 mm wide. The overall PCB has two layers, with the coil structure on the top layer and a simple return trace on the bottom layer. The board thickness is 1.524 mm (60 mils). The total PCB is square, with each edge measuring 60 mm. The coil design is arbitrary, but the board is sized to accommodate a square ferrite sheet with an edge length of 50 mm, as is the size of the primary shielding specified by the Qi specification [13].

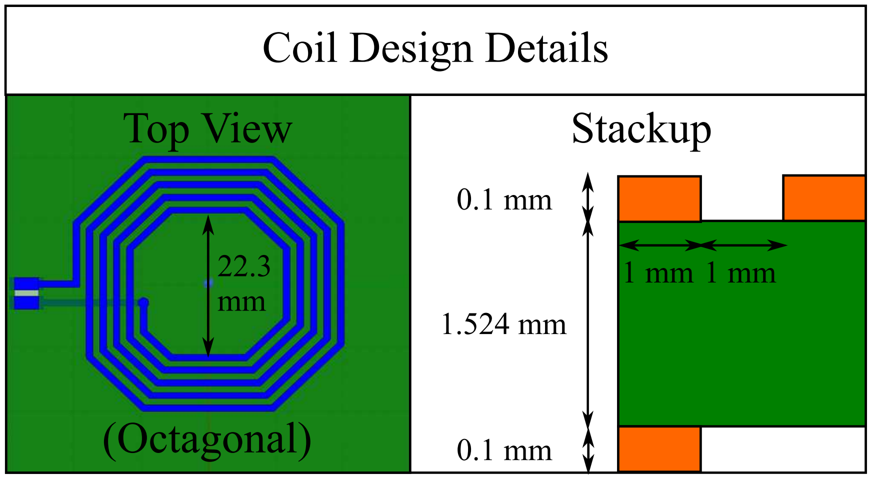

Figure 8. Design details for Coil 1.

4.3. EXTRACTION PROCEDURE

As stated previously, there is not an analytical expression relating inductance to permeability for the planar PCB coil discussed in Section 4.2, so simulations must be relied upon to determine the relationship. In this work, 30 simlutions were used to build up a scattered solution space, where the MATLAB functions interp3 [21] and scatteredInterpolant [22] were utilized to linearly interpolate between the scattered data points. The 30 simulations consisted of two parameter sweeps, for frequencies between 100 kHz and 10 MHz. The first parameter, μ', was swept for six points: 100, 250, 500, 750, 1000, and 1250.

Then, for each μ' point, the second parameter, tan δ, was swept for five points: 0, 0.1, 0.2, 0.4, and 1.0. Following the discussion in Section 4.1, each simulation model included a ferrite sheet on the top and bottom sides of the coil, corresponding to Setup II. The ferrite sheets had a thickness of 0.12 mm. For other values of ferrite thickness, the simulations would have to be repeated.

In Fig. 9, it is clear that varying tan δ affects the resulting inductance. The effect is small for values of tan δ less than 0.4, but is large for greater values of tan δ. Thus, the extraction procedure must take this effect into account. The detailed steps for the extraction procedure are outlined below:

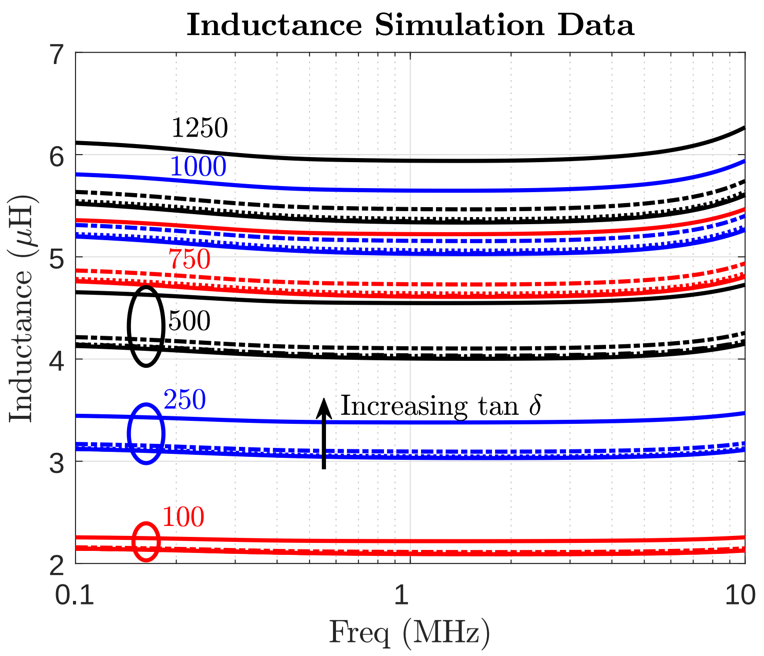

Figure 9. Inductance data from simulations. Each colored group represents a μ' value set, where each line within the set is for a different value of tan δ. In each set, going from the bottom line to the top line corresponds to increasing tan δ values.

1. Sweep μ' and tan δ in simulation.

2. Convert the s-parameters from the simulations to z-parameters.

   - $Z = 50(1 + S)/(1 - S)$

3. Extract inductance (L) and resistance (R) from the z-parameters.
    - $L = imag(Z)/\omega$
    - $R = real(Z)$
4. Build a 3D matrix for both L and R using the simulation data.
    - Dimensions: tan δ x μ' x frequency ; L and R
5. Create meshgrid for use with interp3 function.
6. Create MATLAB interpolation objects for both L and R using interp3 function (for 3D interpolation along frequency axis).
7. For each frequency point, take the corresponding “slice" out of the L and R interpolation objects (2D solution space at that frequency point).
8. Create a scatteredInterpolant object for both “slices" (for 2D interpolation along tan δ and μ' dimensions at that frequency point).
9. Feed the measured inductance into the scatteredInterpolant object for inductance and extract all pairs of μ' and tan δ that give the same inductance from the interpolated simulation data set.
10. For those pairs, feed the measured resistance into the scatteredInterpolant object for resistance and extract the subset of μ' and tan δ pairs that give the same resistance from the interpolated simulation data set.
11. Search the two subsets (one for inductance and one for resistance) for the common pair of μ' and tan δ.

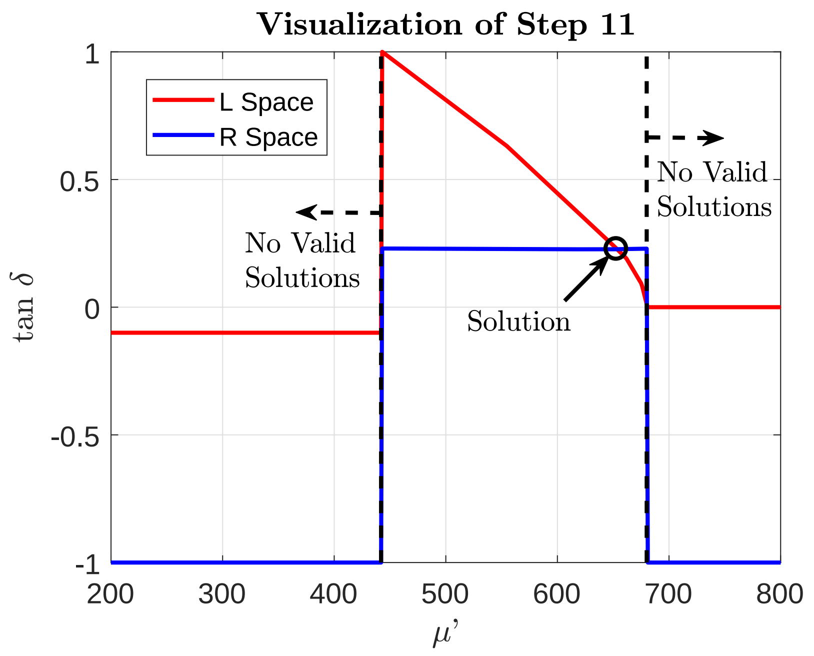

Figure 10. Visualization of Step 11 in the extraction procedure. The frequency was at 0.98 MHz.

Since a given inductance and resistance can correspond to multiple points in each solution space, both solution spaces must be searched. Then, the common (μ', tan δ) point must be chosen as the correct solution. A visual representation of this process is shown in Fig. 10. The figure represents steps 9-11 in the extraction process.

4.4. SIMULATION VALIDATION

For validation, an additional simulation was run with the same ferrite characteristics as discussed in Section 2. The results from this model were then used as “measurement" data in the extraction procedure. The results are shown in Fig. 11. Although not perfect, the proposed method is able to accurately extract the permeability characteristics of the ferrite material, up to 10 MHz. The slight discrepancies are likely due to the reliance on linear interpolation functions.

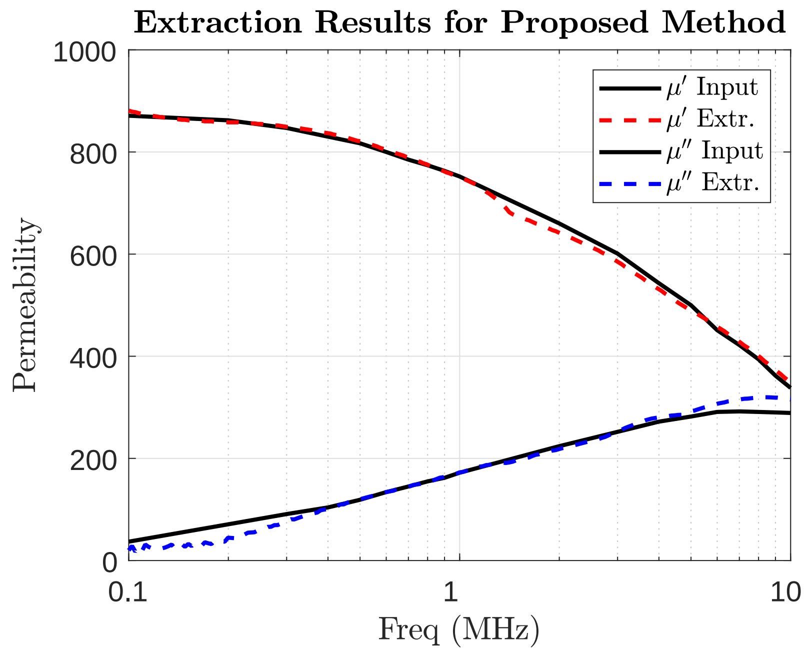

Figure 11. Extraction results using the proposed method.

5. POTENTIAL ISSUES

5.1. FERRITE PERMITTIVITY

The proposed method appears to work up to 10 MHz. However, that is assuming that ϵ_r is equal to 1 for the ferrite material. For values of ϵ_r greater than 1, capacitive effects begin to come into play, causing the coil to resonate. In this case, it cannot be assumed that the imaginary portion of the coil's impedance is purely inductive; instead, the imaginary portion of the coil's impedance will be a function of resistance, capacitance, and inductance. Fig. 12 and Fig. 13 show the extraction results when the permittivity for the ferrite is set to 100 and 1000, respectively. Still, the method is fairly accurate up to 1 MHz, even if the ferrite has high values of permittivity, since the impedance of the coil is still mostly inductive in this frequency range. Since the method is accurate to 1 MHz, the method is still suitable for characterizing ferrites for Qi WPT applications, since the frequency range of interest is from 87 kHz to 205 kHz [13].

Although not investigated, it is suspected that the method proposed in this work could reliably be extended to 10 MHz, even for high values of permittivity, for a different coil design. In particular, an updated coil should be sized such that it does not resonate until well beyond 10 MHz, even in the presence of ferrites with high values of permittivity. The main concern with this approach, though, is that there may be a loss in sensitivity in measured inductance. There is some discussion about resonance and useful frequency range for coil inductors presented in [23].

As discussed, high permittivity ferrites are detrimental to the proposed extraction method since they cause the coil's resonance to shift to lower frequency, reducing the valid frequency range for extraction. However, this information is still useful for WPT designers, since it gives some idea of the ferrite's permittivity. In this way, the proposed setup can provide additional qualitative information about real WPT setups, whereas such qualitative information cannot be provided by the Keysight setup. As discussed in [2], the permittivity of ferrite shields is also an important contributor to the overall performance of WPT systems.

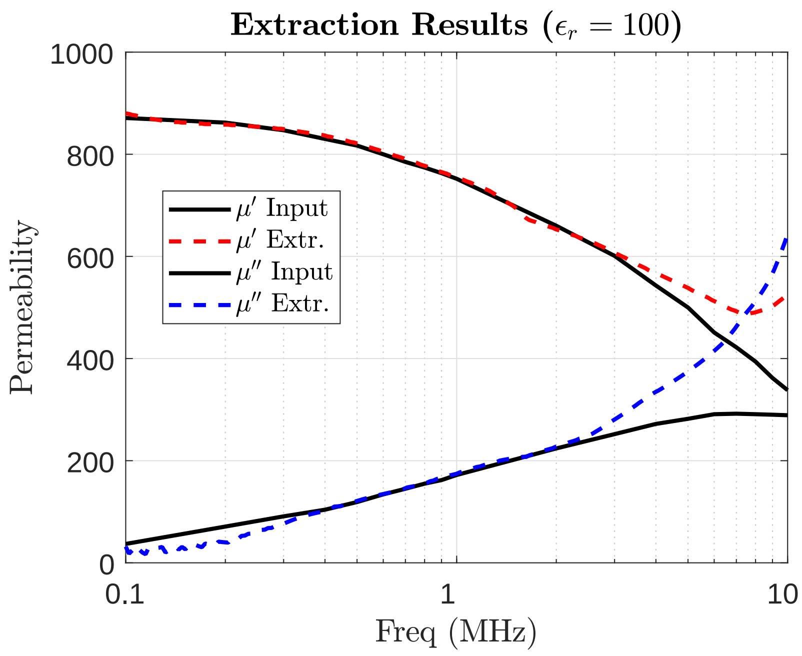

Figure 12. Extraction results for when the ferrite has $\epsilon_r = 100$.

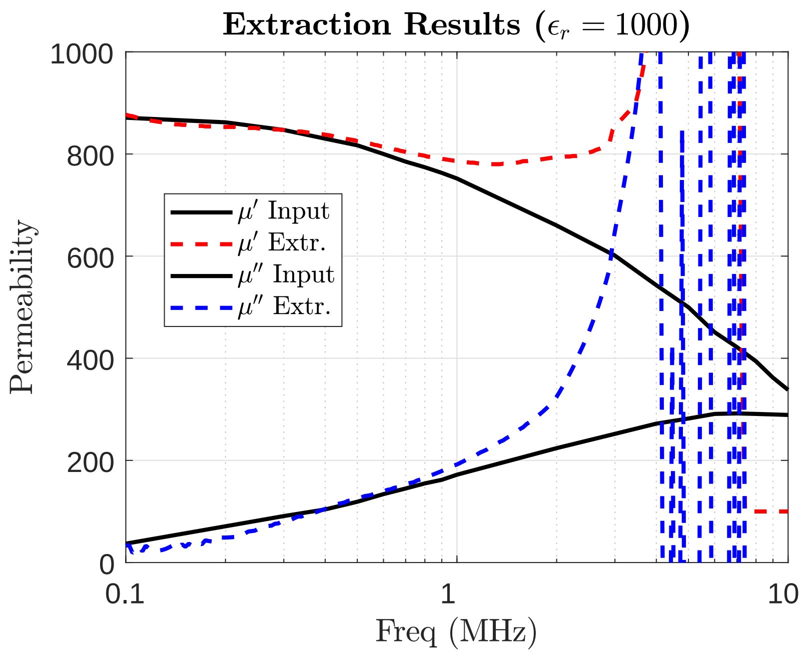

Figure 13. Extraction results for when the ferrite has $\epsilon_r = 1000$.

5.2. REAL MEASUREMENTS

In the case of real measurements, two issues may present themselves. First, the resistance and inductance from the coil model (no ferrites) need to match the values obtained in real measurements (also coil only, no ferrites), since the method relies on simulation data to perform the extraction. In other words, the extraction method will not work well if the relationship between measurements and simulation is not 1 to 1. Secondly, measured inductance will be strongly dependent on ferrite placement. So a measurement setup would need to be carefully designed to minimize variability. For example, this work uses a clamp to minimize variability in measurements. Additional details are provided in Section 6.1.

6. MEASUREMENT VALIDATION WITH SECOND COIL

Since the Keysight 16454A extraction fixture was not available to the authors for making comparisons with measurement data, the proposed method was validated using measurements from two different coils. First, a ferrite was measured using the coil and method described in Section 4. Then, the same ferrite was measured using a second coil (details given in Section 6.3). The second coil was then simulated, using the permeability values extracted from the measurements of the first coil. Finally, the simulated and measured inductance and resistance values for the second coil were compared (see Section 6.4).

6.1. MEASUREMENT SETUP

Measurements were completed using an Agilent E5071C ENA Series Network Analyzer. This network analyzer has a frequency range of 100 kHz to 8.5 GHz, but measurements for this study were only taken from 100 kHz to 1 MHz. Calibration was completed using a 3.5 mm short-open-load (SOL) mechanical calibration kit from Maury Microwave. An Irwin Quick-Grip clamp and two pieces of acrylic were used to minimize variability in the measurement setup, in accordance with the points discussed in Section 5.2. The pieces of acrylic were 2 mm thick and covered the same area as the ferrite sheets (50 mm by 50 mm). Since most of the fields should be redirected by the ferrite sheets before they can interact with the acrylic sheets or clamp, the acrylic sheets and clamp are not expected to affect the measurements. Fig. 14 gives a depiction of the clamp setup, and Fig. 15 shows a photo of the real measurement setup, including the clamp.

6.2. PERMEABILITY EXTRACTION USING FIRST COIL

The coil and method proposed in Section 4 were used to extract the permeability of a real ferrite. The results are shown in Fig. 16.

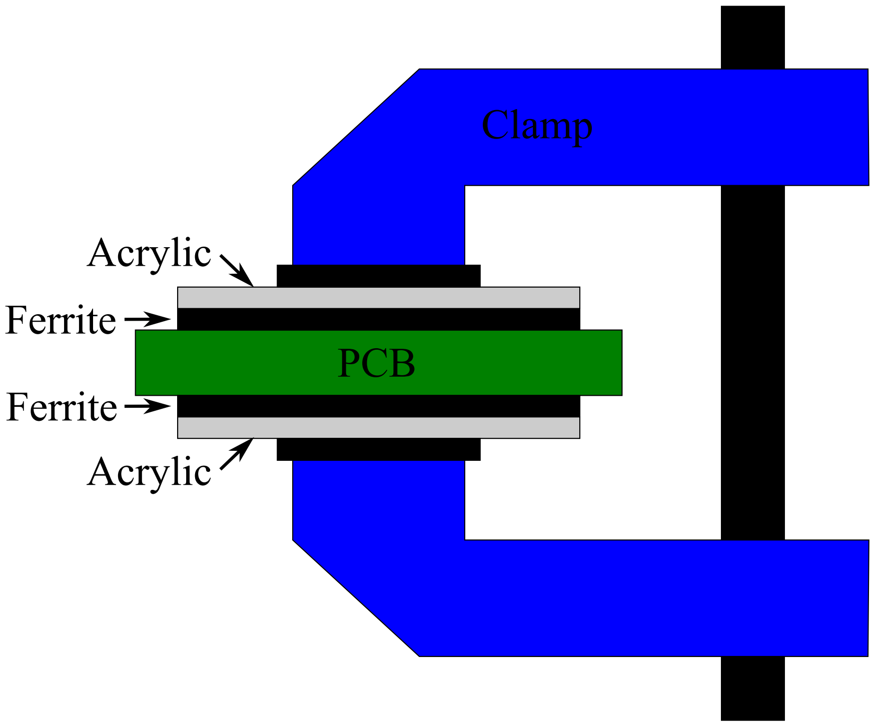

Figure 14. Clamp + acrylic setup for measurements.

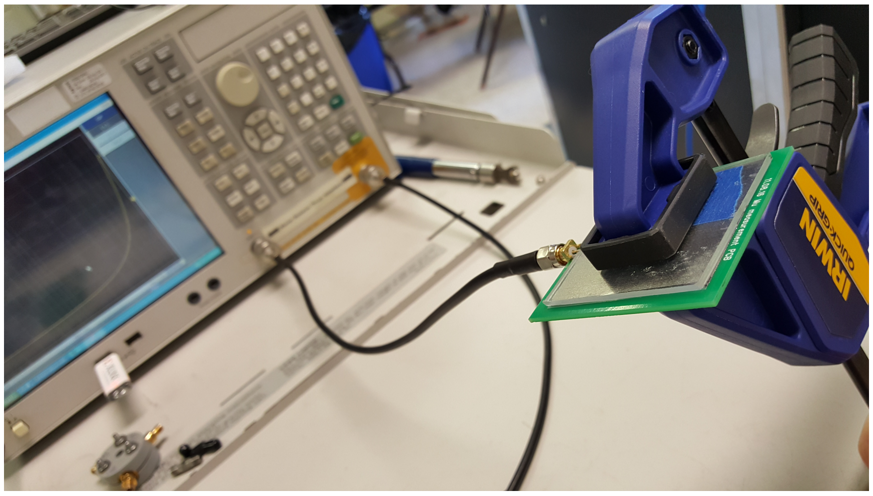

Figure 15. Photo of real measurement setup.

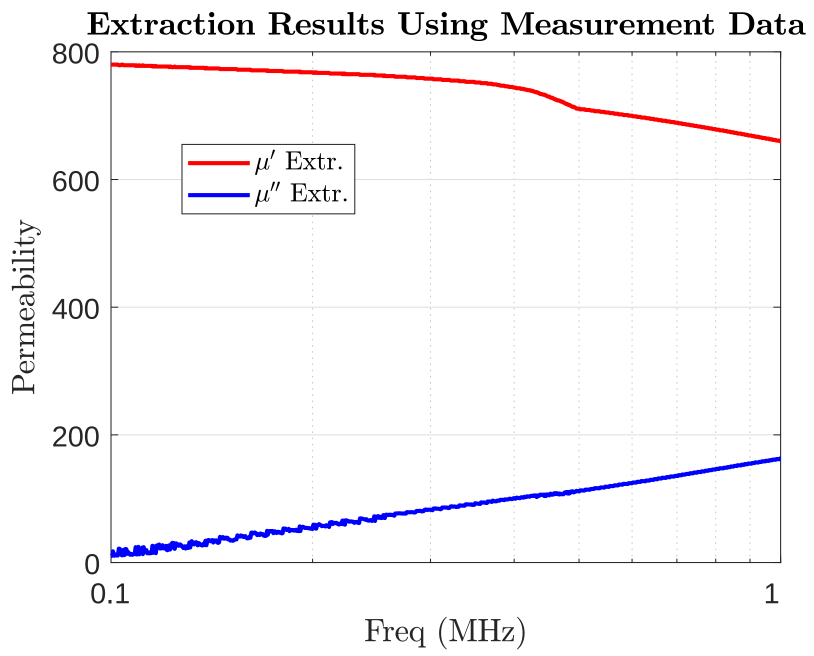

Figure 16. Extraction results using measurement data from Coil 1.

6.3. DETAILS FOR SECOND COIL

The second coil used in the validation process was a simple, one-turn square coil. It was fabricated with a piece of acrylic and some copper tape. Details for the second coil are given in Fig. 17. The piece of acrylic was from the Optix product line of acrylic sheets made by Plaskolite, LLC.

6.4. VALIDATION USING SECOND COIL

The second coil was simulated in HFSS using the extracted permeability from the first coil. The resulting inductance and resistance for the second coil were then compared to the measured values. The inductance comparison is given in Fig. 18, and the resistance comparison is given in Fig. 19. There is less than a 1% difference in inductance and less than a 5% difference in resistance.

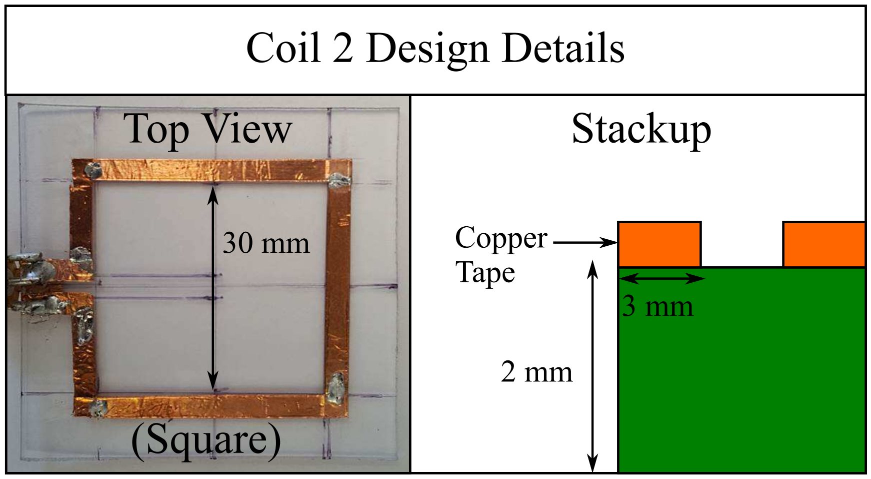

Figure 17. Design details for Coil 2.

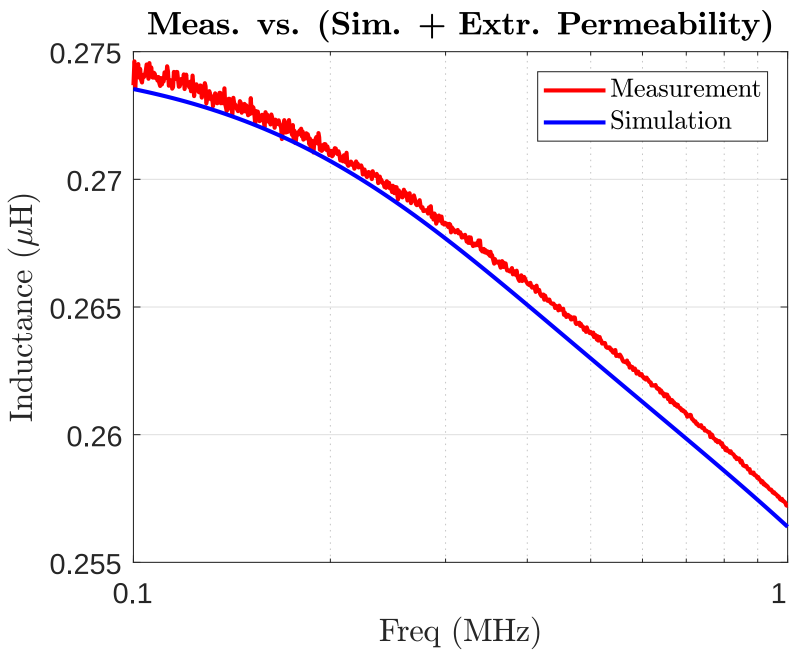

Figure 18. Comparison of simulated and measured inductance for Coil 2. The simulation utilized the extracted permeability from the measurements of the first coil.

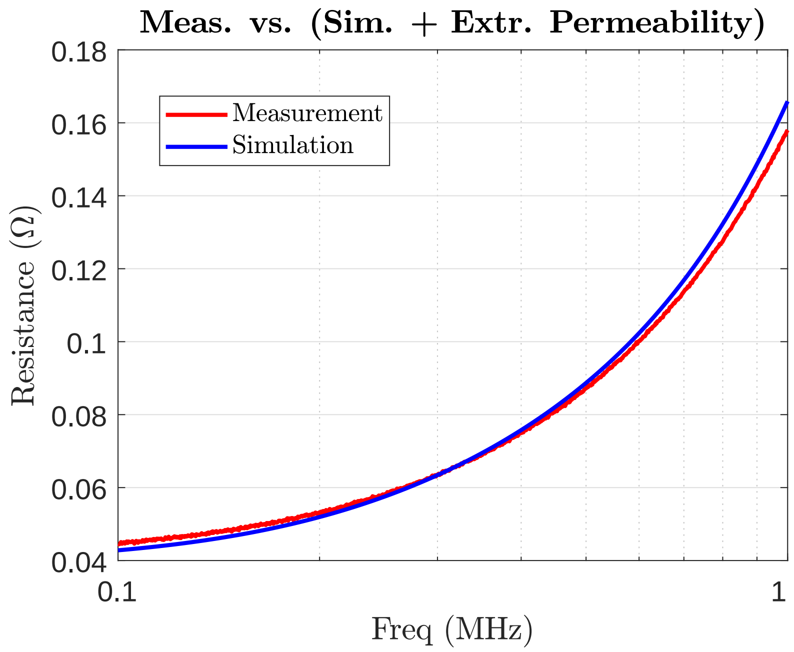

Figure 19. Comparison of simulated and measured resistance for Coil 2. The simulation utilized the extracted permeability from the measurements of the first coil.

7. CONCLUSIONS

The work in this paper demonstrated that a simple, planar PCB coil can be used to extract the frequency dependent permeability characteristics of ferrite sheets using a co-simulation/measurement methodology. The method was compared against the extraction method for the commercialized 16454A fixture from Keysight. The method was also validated using simulations and measurements for a second coil. Compared to the Keysight 16454A extraction method, the proposed method is less expensive and does not require any modifications to the shape of the ferrite. In addition, the simulations required for the proposed method can be easily reused throughout the various design stages for a WPT system. The proposed method was validated up to 1 MHz, and a discussion was provided on how to extend the method to 10 MHz.

REFERENCES

[1] J. Ahuir, “Going Wireless with Magnetic Shielding," *Wurth Elektronik App. Note*, pp. 1-10, Oct. 2013.

[2] C.P. Dick, C. Polak, and E. Waffenschmidt, “Proposal of Figure of Merit for the Characterization of Soft-Magnetic Shielding Material Used in Inductive Wireless Power Transmission Systems", *IEEE Journal Emerging and Selected Topics in Power Electronics*, vol. 3, no. 1, pp. 272-279, March 2015.

[3] J. Hu, A. Sligar, C.H. Chang, S.L. Lu, and R.K. Settaluri, “A Grounded Coplanar Waveguide Technique for Microwave Measurement of Complex Permittivity and Permeability," *IEEE Trans. Magn.*, vol. 42, no. 7, pp. 1929-1931, July 2006.

[4] S. Wang, M. Niu, and D. Xu, “A Frequency-Varying Method for Simultaneous Measurement of Complex Permittivity and Permeability with an Open-Ended Coaxial Probe," *IEEE Trans. Microw. Theory Tech.*, vol. 46, no. 12, pp. 2145-2147, Dec. 1998.

[5] M.W. Hyde IV, and M.J. Havrilla, “Simple, Broadband Material Characterization Using Dual-Ridged Waveguide to Rectangular Waveguide Transitions", *IEEE Trans. Electromagn. Compat.*, vol. 56, no. 1, pp. 239-242, Feb. 2014.

[6] M.H. Hosseini, H. Heidar, and M.H. Shams, “Wideband Nondestructive Measurement of Complex Permittivity and Permeability Using Coupled Coaxial Probes", *IEEE Trans. Instrum. Meas.*, vol. 66, no. 1, pp. 148-157, Jan. 2017.

[7] K. Han, M. Swaminathan, R. Pulugurtha, H. Sharma, R. Tunnala, B. Rawlings, and V. Nair, “RF Characterization of Magnetodielectric Material Using Cavity Perturbation Technique", *IEEE Trans. Compon. Manuf. Tech.*, vol. 5, no. 12, pp. 1850-1859, Dec. 2015.

[8] U. Inan and A. Inan. *Engineering Electromagnetics*, Menlow Park, CA: Addison-Wesley, pg. 508-510, 1999.

[9] “Keysight 16454A Magnetic Material Test Fixture," *Keysight Operation and Service Manual*, Sept. 2017, Available: http://literature.cdn.keysight.com/litweb/pdf/16454-90020.pdf.

[10] J. Füzerová, J. Füzer, P. Kollár, L. Hegedüs, R. Bures, and M. Fáberová, “Analysis of the Complex Permeability Versus Frequency of Soft Magnetic Composites Consisting of Iron and $Fe_{73}Cu_1Nb_3Si_{16}B_7$," *IEEE Trans. Magn.*, vol. 48, no. 4, pp. 1545-1548, April 2012.

[11] M. Szewczyk, K. Kutorasinski, and W. Piasecki, “Quantitative Analysis of High-Frequency Material Properties in Thin-Ribbon Magnetic Cores," *IEEE Trans. Magn.*, vol. 51, no. 8, Aug. 2015.

[12] Q. Yu, T. Holmes, and K. Naishadham, “RF Equivalent Circuit Modeling of Ferrite-Core Inductors and Characterization of Core Materials," *IEEE Trans. EMC*, vol. 44, no. 1, pp. 258-262, Feb. 2012.

[13] “The Qi Wireless Power Transfer System Power Class 0 Specification," *Wireless Power Consortium: Version 1.2.3*, Feb. 2017. Available: https://www.wirelesspowerconsortium.com/knowledge-base/specifications/download-the-qi-specifications.html

[14] A. Hosseinbeig, S. Marathe, and D. Pommerenke, “Characterization of Relative Complex Permittivity and Permeability for Magneto-Dielectric Sheets," *IEEE Trans. Electromagn. Compat.*, vol. 60, no. 6, pp. 1786-1794, Dec. 2018.

[15] K. Han, M. Swaminathan, P. Raj, and H. Sharma, “Extraction of Electrical Properties of Nanomagnetic Materials through Meander-Shaped Inductor and Inverted-F Antenna Structures," *Proc. IEEE 62nd Electron. Compon. Technol. Conf.*, pp. 1808-1813, May/June 2012.

[16] ANSYS Electronics Desktop 2017.2, ANSYS, Inc., Canonsburg, PA.

[17] W. Roshen and D. Turcotte, “Planar Inductors on Magnetic Substrates," *IEEE Trans. Magn.*, vol. 24, no. 6, pp. 3213-3216, Nov. 1988.

[18] W. Roshen, “Effect of Finite Thickness of Magnetic Substrate on Planar Inductors," *IEEE Trans. Magn.*, vol. 26, no. 1, pp. 270-275, Jan. 1990.

[19] W. Roshen, “Analysis of Planar Sandwich Inductors by Current Images," *IEEE Trans. Magn.*, vol. 26, no. 5, pp. 2880-2887, Sept. 1990.

[20] W.G. Hurley and M.C. Duffy, “Calculation of Self and Mutual Impedances in Planar Magnetic Structures," *IEEE Trans. Magn.*, vol. 31, no. 4, pp. 2416-2422, July 1995.

[21] “interp3 Documentation," MATLAB Release 2017A, The MathWorks, Inc., Natick, MA. Available: https://www.mathworks.com/help/matlab/ref/interp3.html [Accessed: Feb. 11, 2019].

[22] “scatteredInterpolant Documentation," MATLAB Release 2017A, The MathWorks, Inc., Natick, MA. Available: https://www.mathworks.com/help/matlab/ref/scatteredinterpolant.html [Accessed: Feb. 11, 2019].

[23] K. Naishadham, “Extraction of RF Permeability of Ferrite Materials Using Direct Measurement of Inductors on Ferrite Cores," *IEEE International Symposium on Antennas and Propagation*, pp. 1863-1866, July 2011.

III. MONTE CARLO PARTICLE SIMULATIONS FOR STUDYING THROUGH-SILICON-VIA (TSV) CAPACITANCE

Nicholas Erickson, *Student Member, IEEE*, Jingdong Sun, *Student Member, IEEE*, Ze Sun, *Student Member, IEEE*, Ryan From, and Jun Fan, *Fellow, IEEE*

ABSTRACT

Serving as vertical interconnects for both signals and power distribution, through-silicon-vias (TSVs) are integral components for enabling 2.5D and 3D IC technology. However, analyzing TSVs is a non-trivial task. At the board level, interconnects such as PCB traces can be fully characterized using only electromagnetic theory, and are easily analyzed using full-wave solvers. At the chip level, though, proper analysis of an interconnect such as a TSV requires accounting for both electromagnetic and semiconductor effects. Specifically, TSVs inherently form a metal-oxide-semiconductor (MOS) structure. It is well known that such structures have a voltage dependent capacitance, an effect that can only be described using semiconductor theory. Full-wave solvers are not able to desribe such semiconductor effects, at least, not in a rigorous manner. In this work, semi-classical Monte Carlo particle simulations (implemented in MATLAB) are used to analyze the voltage dependent capacitance of a TSV pair. In addition, the particle simulation is implemented using a finite-element mesh, in order to easily account for the circular shape of the TSVs.

Keywords: tsv, semiconductor, mos, particle, monte carlo

1. INTRODUCTION

With semiconductor foundries producing chips at the 7 nm process node, transistor scaling is quickly approaching its physical limit. In an effort to keep pace with Moore's Law, without relying on transistor scaling, many chip makers now depend on 2.5D and

3D integrated circuit (IC) topologies [1]. Vertical interconnects enable 2.5D and 3D IC topologies by allowing chips to be stacked on top of each other. The main type of vertical interconnect currently in use is the through-silicon-via (TSV). Real products already using 2.5D and 3D topoogies that utilize TSV technology include the Micron hybrid memory cube [2], high bandwidth memory DRAM from SK Hynix [3] and Samsung [4], Xilinx SoCs [5], and AMD GPUs [6].

The structure of a TSV consists of a conductor going through some or all of the silicon substrate, separated from the substrate by a thin insulating layer (usually SiO_2). Diameters in the range of 0.3 μm - 1.5 μm, and pitches in the range of 1.6 μm - 3 μm, are expected for intermediate TSVs in the 2017-2025 time frame, as outlined by the International Roadmap for Devices and Systems (IRDS) [7]. Inherently, the geometry of a TSV forms a metal-oxide-semiconductor (MOS) structure. Due to the phenomena of accumulation, depletion, and inversion, the capacitance of a TSV can change, depending on the applied voltage bias.

There have been several publications on calculating the capacitance of TSVs and TSV arrays. Some authors completely neglect the MOS effect in their models [8]-[10]. Others attempt to incorporate some type of MOS effect by assuming some predetermined depletion radius [11] or by using the full depletion approximation [12]. Some take the MOS effect into account by solving Poisson's equation in cylindrical coordinates, as in [13] and [14]. This work calculates the capacitance of a TSV using particle simulations. In this way, no approximations or assumptions are needed for the capacitance calculation, outside of knowing the flat-band voltage, V_{FB}, for the TSV MOS structure. The flat-band voltage must be calculated ahead of time because it must be applied as part of the boundary condition for the Poisson solver. Note that only holes are simulated in this work, so effects of inversion are ignored. However, this has little consequence for TSVs carrying high-frequency signals, since the generation of minority carriers (electrons) is too slow for an inversion layer to be formed [14]-[17].

Section 2 describes the simulated TSV structure. Section 3 discusses the details of the particle simulator implemented in this work. Since the finite-element-method (FEM) is used in this work, differences between it and the more commonly implemented finite-difference method (FDM) are highlighted. The resulting capacitance values for the TSV structure are given in Section 4. The conclusion is given in Section 5.

2. TSV GEOMETRY

The structure simulated in this work consists of a signal-ground TSV pair, in a P-Type silicon substrate, as shown in Fig. 1. The TSVs each have a 0.5 μm radius, are separated from the substrate by a 0.1 μm thick SiO_2 liner, and have a pitch of 2 μm. These sizes fall in the ranges specified by the IRDS [7]. The substrate is doped to 1e21$/m^3$ (1e15$/cm^3$), and is 5 μm by 5 μm in size. The TSV material is aluminum. No ground contacts are assumed for the substrate. Dirichlet boundary conditions (including V_{FB}) are only enforced on the nodes of the signal and ground TSVs. Neumann conditions, such that $\frac{dV}{dn} = 0$, are enforced on the nodes of the bounding box.

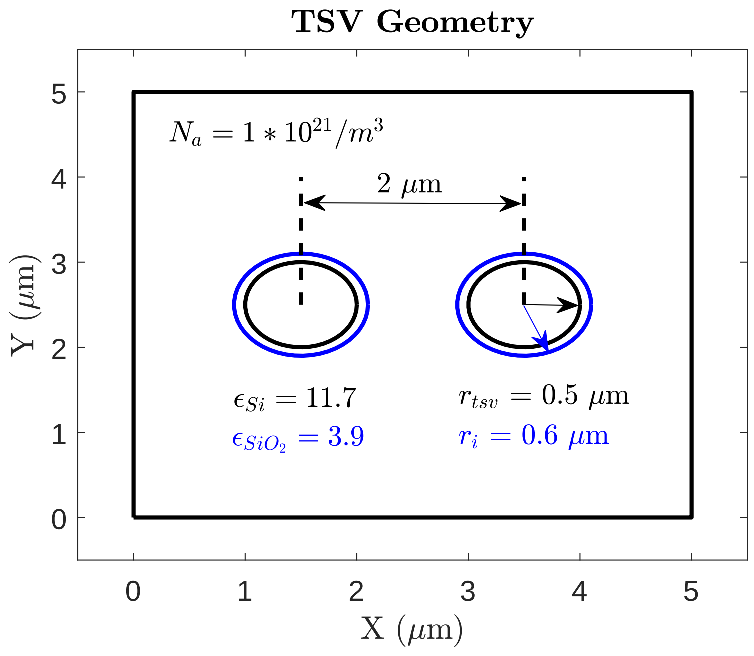

Figure 1. Geometry for the TSV pair.

3. DESCRIPTION OF PARTICLE SIMULATOR

3.1. BOLTZMANN TRANSPORT EQUATION (BTE) [18]-[20]

The foundation of Monte Carlo (MC) particle simulators is the Boltzmann Transport Equation (BTE), as is shown in (1). It describes the evolution of a distribution of particles (f), based on the particles' motion, forces acting on the particles, and how the particles scatter. The second term represents spatial variations due to temperature or concentration gradients, and the last term represents changes due to scattering events. The third term represents the effects of electric or magnetic fields, as is apparent when $\frac{\partial \vec{k}}{\partial t}$ is expanded, as shown in (2).

$$\frac{\partial f}{\partial t} + \vec{v}_g * \nabla_r f + \frac{\partial \vec{k}}{\partial t} * \nabla_k f = \left.\frac{\partial f}{\partial t}\right|_{coll} \tag{1}$$

$$\frac{\partial \vec{k}}{\partial t} = \frac{1}{\hbar}\frac{\partial p}{\partial t} = \frac{q}{\hbar}(\vec{F} + \vec{v}\ x\ \vec{B}) \tag{2}$$

Numerically, several core functions are needed to actually solve the BTE by following the evolution of the particle distribution function. The functional blocks and general code flow are given in Fig. 2. Each block is described in detail in the proceeding sections.

3.2. HOLE BAND MODEL

Since the TSV pair studied in this work is in a P-Type silicon substrate, the majority carriers are holes. To model the behavior of the holes, a three band model (heavy, light, and split-off) was adopted. Holes in the heavy and light bands follow a warped E-k relationship, as described by [21] and [22], and shown in (3). The function g is defined in (4). Holes in the split-off band follow a spherical E-k relationship, as shown in (5).

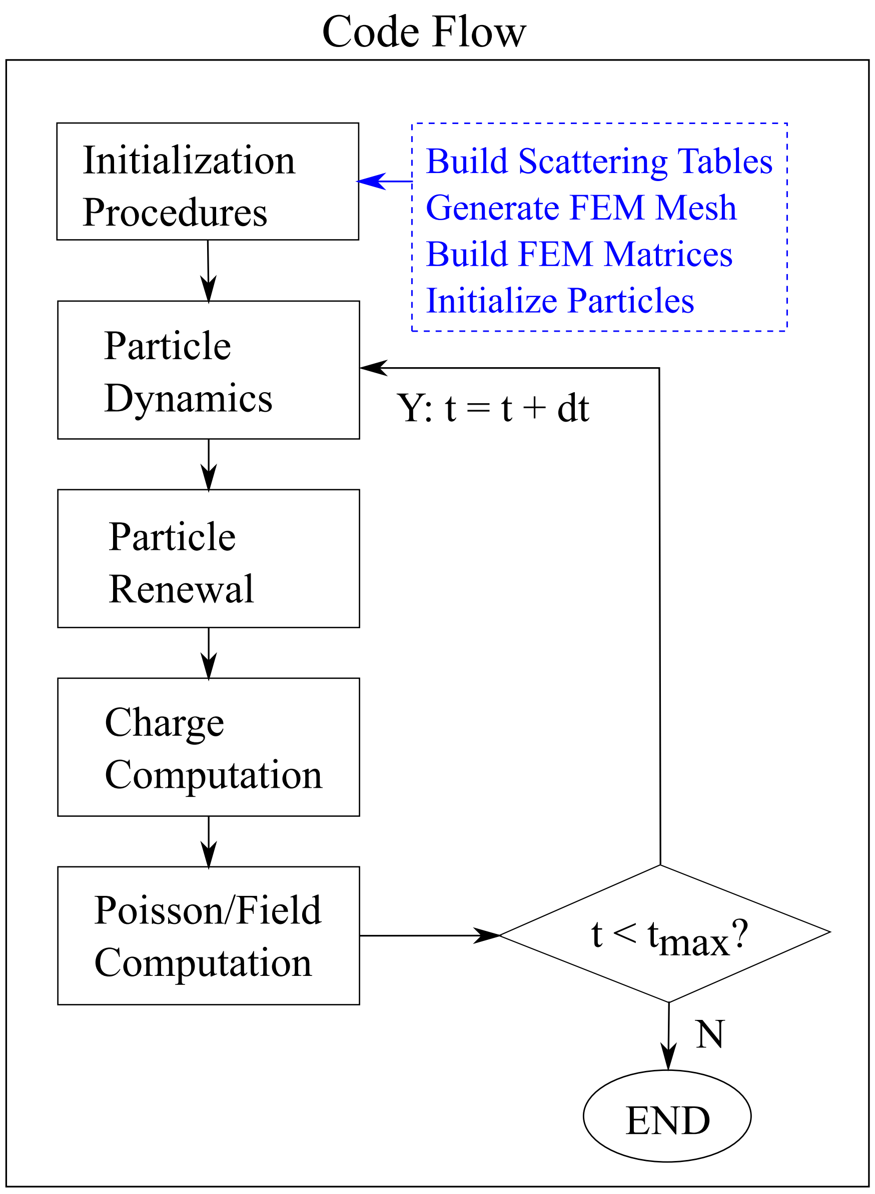

Figure 2. Code flow for solving the BTE.

$$E(k) = \frac{\hbar^2|A|k^2}{2m_o}[1 \mp g(\theta,\phi)] \tag{3}$$

where

$$g = \left[\frac{B}{A}^2 + \frac{C}{A}^2 * \left(sin^2\theta * cos^2\theta + sin^4\theta * cos^2\phi * sin^2\phi\right)\right]^{\frac{1}{2}} \tag{4}$$

$$E(k) = \frac{\hbar^2 k^2}{2m_{so}} \tag{5}$$

In (5), m_{so} stands for the effective mass of the hole in the split-off band ($m_{so} \approx 0.29m_o$). Comparing (3) to (5), the effective mass for holes in the heavy or light valley can be defined as in (6). From (6), it is clear that heavy and light holes have an effective mass that changes with g.

$$m^*_{HL} = \frac{m_o}{|A|(1 \mp g)} \tag{6}$$

In the above equations, $\hbar$ is the reduced Planck constant ($1.0546 * 10^{-34} J * s$), k is the hole wavevector, m_o is the resting mass of an electron in free space ($9.1095 * 10^{-31} kg$), and A, B, and C are inverse band mass parameters (-4.22, -0.78, 4.8, respectively). The inverse band mass parameters are the same as those used in [21]. The $\mp$ denotes the heavy (-) and the light (+) bands, and θ and ϕ represent the polar and azimuthal angles of $\vec{k}$.

3.3. HOLE SCATTERING RATES

Only two types of scattering were considered in this work: acoustic scattering and non-polar optical (NPOP) scattering. NPOP scattering involves both absorption and emission processes. Intraband and interband scatterings are calculated for both acoustic and NPOP scattering types. The equations used for calculating the scattering rates follow from [20] and [22].

Acoustic scattering was tabulated using (7). In the equation, Ξ represents the acoustic deformation potential (9.5 eV), k_b the Boltzmann constant ($1.3807 * 10^{-23}\ \frac{m^2 kg}{s^2 K}$), T the temperature ($300K$), c_l the elastic constant ($c_l = \rho v_s^2$), and m^* the effective mass of the hole. For c_l, ρ is the crystal density ($2329\ kg/m^3$) and v_s is the sound velocity ($9040\ m/s$).

$$P = \left[\frac{\Xi^2 k_b T}{4\pi\hbar c_l}\right]\left[\frac{\sqrt{E}(2m^*)^{\frac{3}{2}}}{\hbar^3}\right] \tag{7}$$

NPOP scattering was tabulated using (8), where N_o is the phonon occupation number, calculated from the Bose-Einstein distribution given in (9). In (8) and (9), DK is the optical deformation potential ($5 * 10^{10}\ eV/m$) and $\hbar\omega_o$ is the phonon energy ($0.063\ eV$).

$$P = \frac{DK^2 (m^*)^{\frac{3}{2}} \sqrt{E \pm \hbar\omega_o}}{2\sqrt{2}\pi\rho\hbar^3\omega_o} \begin{Bmatrix} N_o \\ N_o + 1 \end{Bmatrix} \tag{8}$$

$$N_o = \frac{1}{e^{\frac{\hbar\omega_o}{k_b T}} - 1} \tag{9}$$

Since m^* is dependent on $\vec{k}$ for the heavy and light bands, m^* was taken to maximize the scattering rates, as was done in [22]. For the heavy band, m^* is maximal when (4) is maximized; for the light band, m^* is maximal when (4) is minimized. The portion of (4) involving the sine and cosine functions is plotted in Fig. 3.

The sine/cosine function has a minimum value of 0 and a maximum value of 1/3. Substituting these values into (4), and those values subsequently into (6), results in effective masses of $m_{heavy} \approx 0.75m_o$ and $m_{light} \approx 0.2m_o$. Substituting these values into (7) and (8) results in maximized scattering rates for the heavy and light valleys. Rates do not need maximized for the split-off band; rates are simply calculated using $m^* = m_{so}$. A plot of the scattering rates for holes in the heavy valley is given in Fig. 4. The actual treatment of scattering events using the maximized rates is given in Section 3.8.

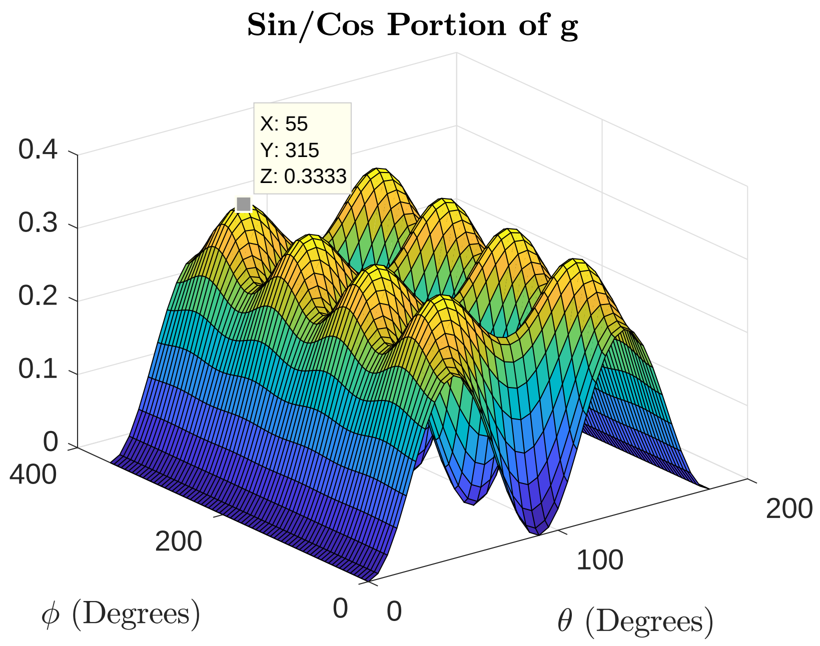

Figure 3. Min and max of sine/cosine portion of g. The max value for the function is 1/3.

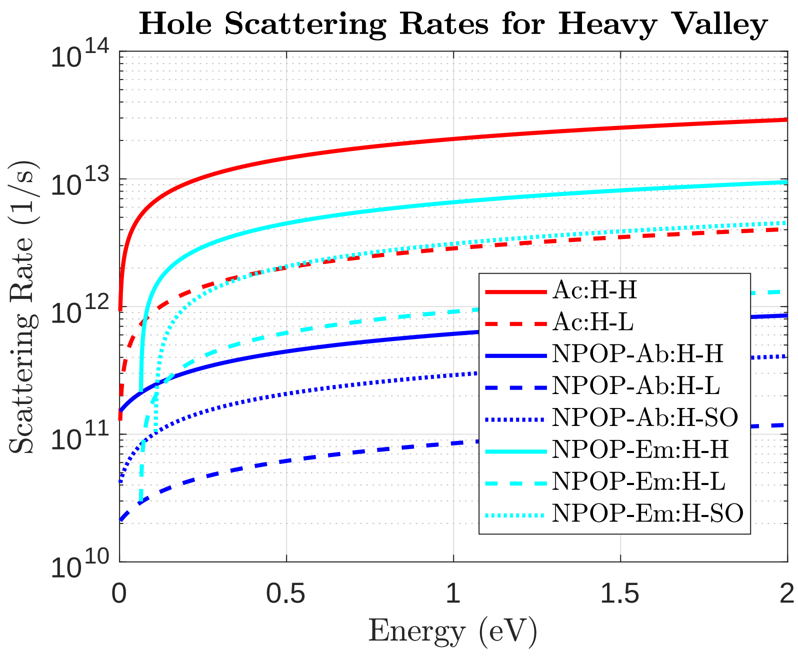

Figure 4. Scattering rates for heavy holes.

3.4. FEM MESH GENERATION

To generate the FEM mesh for the simulations, the freely available “triangle" program was utilized from J. Shewchuk [23]. Within MATLAB, the required .poly file is generated based on user inputs. Then MATLAB performs a system call to the triangle program. The triangle program then outputs a .ele file and a .node file that are then read back into MATLAB to create the connectivity and node matrices required for the FEM portion of the program. The connectivity matrix is an T x 3 matrix, where T is the total number of triangles in the mesh. The columns of the connectivity matrix correspond to the three nodes that form each triangle. The node matrix is an N x 3 matrix, where N is the total number of nodes in the mesh. The columns of the node matrix correspond to the x and y coordinates of each node, and each node’s boundary condition specifier.

To ensure stability of the simulation, constraints on the time step and mesh were adapted from those given in [20] for a finite-difference mesh with uniform grid spacing. The main adaptation is that instead of constraining dx (edge of a square cell), the area of each FEM triangle is constrained. The area constraint is calculated using the steps below. The mesh for the TSV structure studied in this work is shown in Fig. 5.

1. Requirement: $dt << 1/\omega_p$
   - Calculate: $\omega_p = \sqrt{\frac{q^2 dope_{substrate}}{\epsilon_{stat} m_{light}}}$
   - Set: $dt = 0.1 * 1/\omega_p$
2. Requirement: $l_{max} < dx, dy < \lambda_D$
   - Calculate: $l_{max} = v_{hole}^{max} * dt$
   - Calculate: $\lambda_D = \sqrt{\frac{\epsilon_{stat} k_b T}{q^2 dope_{substrate}}}$
   - Set: $tri_{edge} = 0.5 * (l_{max} + \lambda_D)$
   - Set: $tri_{area} = 0.5 * tri_{edge}^2$

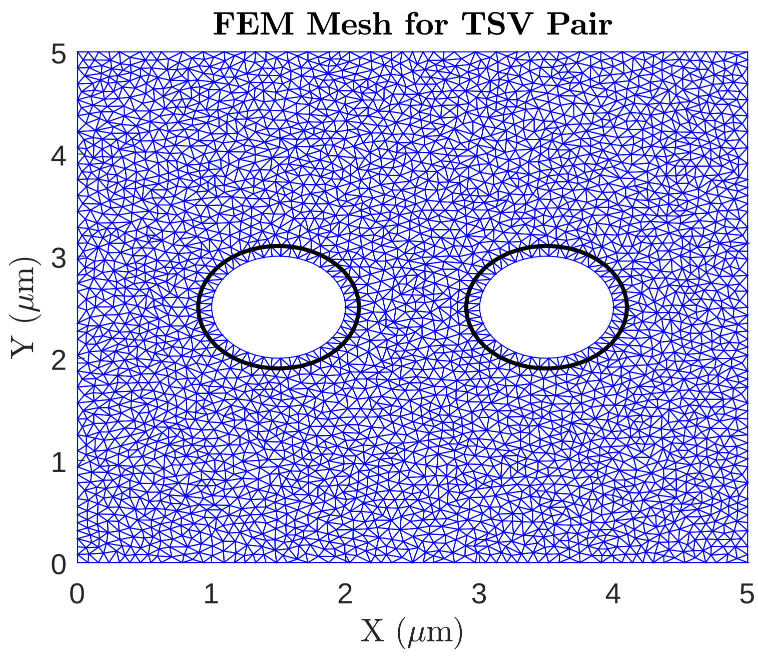

Figure 5. FEM mesh for the TSV pair. The solid black lines represent the oxide liner around the TSV.

3.5. BRIEF REVIEW OF FEM [24]

From FEM theory, the weak-form representation of the Poisson equation is shown in (10). Equivalently, (10) can be written in matrix form as shown in (11), where K is a square matrix involving the interpolation functions, b is the charge vector, and g is the Neumann vector. The FEM formulation results in the K matrix being sparse, so MATLAB's sparse function was utilized in the code to save on the memory required for creating the K matrix [25].

$$\int_{\Omega^e} \epsilon \nabla \Psi_i^e * \nabla V d\Omega - \int_{\Omega^e} \Psi_i^e \rho d\Omega - \int_{\Gamma^e} \Psi_i^e \epsilon \frac{dV}{dn} dl = 0 \tag{10}$$

$$[K][V] - [b] - [g] = 0 \tag{11}$$

V can be approximated using linear interpolation functions, as shown in (12). The coefficients a, b, and c can be solved according to (13), where the α's, β's, and γ's are known functions involving the coordinates of the triangle's three nodes. The formulas for the α's, β's, and γ's are shown in (14)-(16), where i, j, and k represent the three nodes for the element.

$$V^e = a + bx + cy \tag{12}$$

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \frac{1}{2A^e} \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \\ \beta_1 & \beta_2 & \beta_3 \\ \gamma_1 & \gamma_2 & \gamma_3 \end{bmatrix} \begin{bmatrix} V_1^e \\ V_2^e \\ V_3^e \end{bmatrix} \tag{13}$$

$$\alpha_i = x_j y_k - x_k y_j \tag{14}$$

$$\beta_i = y_j - y_k \tag{15}$$

$$\gamma_i = x_k - x_j \tag{16}$$

Every triangle contributes to nine entries in the K matrix. When more than one triangle contributes to an entry, the contributions are added together. The nine entries from each triangle ($K_{11}, K_{12}, K_{13}, K_{21}, K_{22}, K_{23}, K_{31}, K_{32}, K_{33}$) are computed using (17), where i and j stand for the triangle's nodes.

$$K_{ij}^e = \frac{\epsilon^e}{4A^e}(\beta_i^e \beta_j^e + \gamma_i^e \gamma_j^e) \tag{17}$$

3.6. INITIALIZING PARTICLES

All holes are initialized in the heavy band, with initial energy and initial k vector computed using (18) and (19), where g is computed according to (4), using random angles computed from (20) and (21).

$$E_i = \frac{k_b T}{q} * ln(rand()) * 1.5 \tag{18}$$

$$k_i = \sqrt{\frac{2m_o E_i q}{|A|\hbar^2(1-g)}} \tag{19}$$

$$cos\theta = 1 - 2 * rand() \tag{20}$$

$$\phi = 2\pi * rand() \tag{21}$$

To randomly distribute the particles in each triangle, the equations from [26] were utilized. The equations involve two random numbers (r_1 and r_2) and the x and y coordinates of each triangle's three nodes, as shown in (22) and (23).

$$x = (1 - \sqrt{r_1})x_1 + \sqrt{r_1}(1 - r_2)x_2 + (r_2\sqrt{r_1})x_3 \tag{22}$$

$$y = (1 - \sqrt{r_1})y_1 + \sqrt{r_1}(1 - r_2)y_2 + (r_2\sqrt{r_1})y_3 \tag{23}$$

3.7. PARTICLE MOVEMENT

Between scattering events, a particle's motion is driven by changes in its $\vec{k}$. Changes in a particle's $\vec{k}$ are driven by the fields within the device. Assuming no magnetic field effects, the x and y components of a hole's $\vec{k}$ are updated according to (24) and (25), which follow from (2). In the equations, F_x and F_y stand for the x and y components of the electric field at the hole's position, respectively. The free-flight time (the time between scattering events) is represented by τ. Note that the relationship is positive since holes are the particles being simulated; for electrons, a "-" sign would need to be added to (24) and (25).

$$dk_x = \frac{q}{\hbar} F_x \tau \tag{24}$$

$$dk_y = \frac{q}{\hbar} F_y \tau \tag{25}$$

The distance a particle travels is computed by multiplying its free-flight time by its mean velocity. A particle's mean velocity is calculated by (26). Depending on the hole's valley, (3) or (5) can be substituted into (26) for E. The result for the x component of velocity is given in (27) and (28), for heavy/light and split-off holes, respectively. The average x component of $\vec{k}$ during the free-flight is k_x^{avg} and is given in (29). Putting everything together, a hole's x position is updated according to either (30) or (31), depending on the valley.

$$v = \frac{1}{\hbar} \frac{dE}{dk} \tag{26}$$

$$v_x = \frac{\hbar |A| k_x^{avg}}{m_o} [1 \mp g] \tag{27}$$

$$v_x = \frac{\hbar k_x^{avg}}{m_{so}} \tag{28}$$

$$k_x^{avg} = \frac{k_x + (k_x + dk_x)}{2} = k_x + 0.5 * dk_x \tag{29}$$

$$x_f = x_i + \left[\frac{\hbar |A| (k_x + 0.5 * dk_x)(1 \mp g)}{m_o} \right] \tau \tag{30}$$

$$x_f = x_i + \frac{\hbar (k_x + 0.5 * dk_x)}{m_{so}} \tau \tag{31}$$

3.8. PARTICLE SCATTERING

For particles following a spherical E-k relationship, such as holes in the split-off band, adjusting $\vec{k}$ after an isotropic scattering event is straightforward. This is because the spherical E-k relationship allows for any angle to be valid for $\vec{k}$. So $\vec{k}$ is simply updated using (32). Then the x, y, and z components of $\vec{k}$ are calculated using (33)-(35), where the angles are computed using (20) and (21).

$$k_f = \frac{\sqrt{2 m_{so} E_f q}}{\hbar} \tag{32}$$

$$k_x^f = k_f sin\theta cos\phi \tag{33}$$

$$k_y^f = k_f sin\theta sin\phi \tag{34}$$

$$k_z^f = k_f cos\theta \tag{35}$$

Since heavy and light holes follow a warped E-k relationship, however, only certain angles are permitted for $\vec{k}$. In this case, the rejection method from [22] is used to find valid angles for $\vec{k}$. This is also the reason for maximizing the scattering rates, as was discussed in Section 3.3. The rejection procedure includes four steps. First, random angles are drawn from a sphere according to (20) and (21). Then the function f is calculated using those angles, where f is defined in (36), and g is still defined as in (4). Then a random number is drawn and tested against the condition in (37), using the f computed from (36). In (37), f_{max} is also computed from (36), using the maximum or minimum value of g, depending on the hole's valley (see Section 3.3). If the condition in (37) is satisfied, the set of angles is valid for scattering. If not, new random angles are chosen and the process is repeated, until a valid set of angles is found. For valid angles, $\vec{k}$ is computed using (38). The x, y, and z components of $\vec{k}$ then follow from (33)-(35).

$$f = \frac{1}{\left[\frac{\hbar^2|A|}{2m_o}(1 \mp g)\right]^{\frac{3}{2}}} \tag{36}$$

$$r \leq f/f_{max} \tag{37}$$

$$k_f = \sqrt{\frac{2m_o E_f q}{|A|\hbar^2(1 \mp g)}} \tag{38}$$

3.9. PARTICLE HANDLING AT BOUNDARIES

Usually, transistors are the focus of particle simulations. For particle simulations involving transistors, a particle renewal function is typically required to properly handle contacts that allow for current flow; that is, a function that handles particle flow by deleting or adding particles at the contacts. However, in the case of the TSV pair, both the signal TSV and the GND TSV are surrounded by an oxide liner. In this case, no particle flow is allowed through either TSV. So the total number of particles in the simulation is always the same. No particles are ever added or deleted, so a renew function is not required. However, a routine is still needed to handle particle reflections from the bounding box and from the TSV insulators.

At the bounding box, reflections are simple. When a particle moves past the bounding box, the distance the particle moved in the x and y direction (beyond the box) is simply mirrored across the edge of the bounding box. Then the particle's x or y component of $\vec{k}$ is negated so that the particle is directed into the simulation domain.

Reflections from the oxide liner are more complicated since the oxide surface is circular. The steps to reflect a particle from the oxide liner are shown below. An illustration of the steps is given in Fig. 6.

1. Shift coordinates so that the center of the intersected TSV is the origin.
2. Find the equation of the line formed between the intial particle location and the final particle location.
3. Find the intersection of that line with the oxide liner (solve equation of line and circle).
4. Find the equation of the tangent line at that intersection point.
5. Find the equation of the line that goes from the final particle location to the tangent line, such that the line is normal to the tangent line.
6. Find the intersection point between the tangent line and the normal line.
7. Find Δx and Δy from the final particle location to the intersection point of the tangent line and the normal line.
8. Add Δx and Δy to the intersection point. This results in the final particle location being reflected over the tangent line to a point outside of the oxide liner.

Consistent with a particle reflection from the bounding box, a particle reflection from the oxide liner also requires adjustments to $\vec{k}$ so that the particle is directed away from the oxide liner. During the reflection routine, the x and y distance from the particle's final location to the intersection point of the normal and tangent line is calculated. The x and

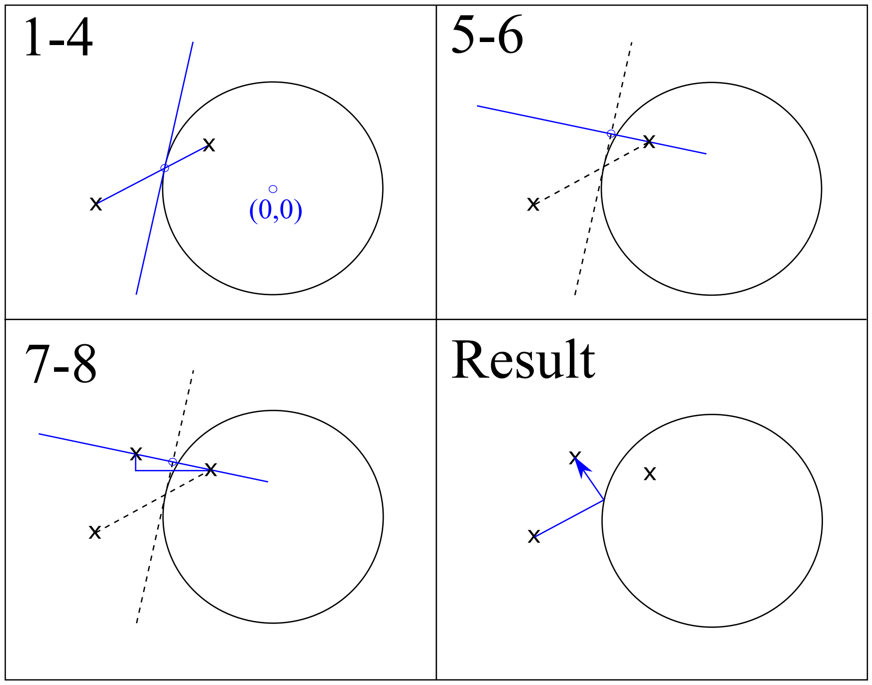

Figure 6. Illustration of steps for reflecting particle from circular structure.

y component distances are then normalized by dividing by the straight-line distance to the intersection point. These normalized components then determine how to adjust the final x and y components of $\vec{k}$. The final signs of k_x and k_y are determined by checking if the particle's final location is to the left or right of the intersection point, and if the particle's final location is above or below the intersection point.

3.10. CHARGE COMPUTATION

In (11), $[b]$ must be updated at each time step. To update $[b]$, all particles are looped through, with all of a particle's charge being assigned to the nearest triangle. Here, the definition of nearest triangle means nearest triangle center. However, this step gives charge per triangle, not charge per node, as is required by $[b]$.

$$\rho^e_{node} = \frac{1}{3}\rho^e_{net}A^e \tag{39}$$

To assign charge per node, (39) was used to split a triangle's charge equally between its three nodes. This requires looping through all triangles, where multiple contributions to a node are summed together.

Assigning charge on an unstructured triangular mesh causes a non-zero “self-force" on the particles, as discussed in [27]. However, it is claimed in [28], that the influece of “self-force" is almost negligible, if the number of super particles used in the simulation is large. In this work, $\approx 33,000$ super particles were used in the simulations.

3.11. POISSON/FIELD COMPUTATION

Calculating the voltage at every node is straightforward using FEM theory. Reorganizing (11) based on Dirichlet boundary nodes and inner nodes results in (40):

$$\begin{bmatrix} K_{II} & K_{ID} \\ K_{DI} & K_{DD} \end{bmatrix} \begin{bmatrix} V_I \\ V_D \end{bmatrix} = \begin{bmatrix} b_i \\ b_D \end{bmatrix} + \begin{bmatrix} g_i \\ g_D \end{bmatrix} \tag{40}$$

In (40), b_i is known from the charge computation and $g_i = 0$. For the signal TSV nodes, V_D is the requested bias condition minus the flatband voltage; for the ground TSV, V_D is taken as zero minus the flatband voltage. The flatband voltage, V_{FB}, is calculated according to (41). For aluminum, ϕ_m = 4.1 eV. For silicon, X_{Si} = 4.05 eV, E_g = 1.12 eV, and $n_i = 1.5 * 10^{16}/m^3$. Using these values, $V_{FB} \approx -0.8V$. As an example, consider a requested bias voltage of -2V. In this case, the boundary condition on the signal TSV is -2V minus -0.8V, resulting in -1.2V. The boundary condition on the ground TSV is then 0V minus -0.8V, resulting in 0.8V. The voltage difference between the two TSVs is still 2V (1.2V plus 0.8V).

$$V_{FB} = \phi_m - \phi_{Si} = \phi_m - X_{Si} - \frac{E_g}{2} - \frac{k_b T}{q} ln\left(\frac{N_a}{N_i}\right) \tag{41}$$

Thus, the only unknowns in (40) are V_I and g_D. The solution for V_I is found using (42). Once V_I is known, g_D can be solved according to (43).

$$V_I = K_{II}^{-1}[b_i - K_{ID}V_D] \tag{42}$$

$$g_D = K_{DI}V_I + K_{DD}V_D \tag{43}$$

Calculating the electric field is also straightforward using FEM theory. Taking the negative derivative of (12) with respect to x and y results in (44) and (45). Utilizing (13) in conjunction with (44) and (45) results in (46) and (47).

$$E_x^e = -\frac{dV^e}{dx} = -b \tag{44}$$

$$E_y^e = -\frac{dV^e}{dy} = -c \tag{45}$$

$$E_x^e = -\frac{1}{2A^e}(\beta_1 V_1^e + \beta_2 V_2^e + \beta_3 V_3^e) \tag{46}$$

$$E_y^e = -\frac{1}{2A^e}(\gamma_1 V_1^e + \gamma_2 V_2^e + \gamma_3 V_3^e) \tag{47}$$

4. TSV CAPACITANCE EXTRACTION

Three capacitances can be extracted from the particle simulation. The first capacitance is the capacitance between the signal and ground TSV. The solution of (43) gives the charge at each node for both the signal TSV and the ground TSV. Summing the charge for all the nodes belonging to a TSV is the same as solving (48), where the enclosing surface, S, is right at the surface of the TSV. This is evident from the FEM definition of g, as given in (10) and repeated in (49). The charge for each TSV is equal, but with opposing signs. Capacitance is then simply Q/V, where V is equal to V_{Bias}. A plot of the charge on the signal and ground TSVs for different bias conditions is shown in Fig. 7. When one TSV is in accumulation, the other TSV is in depletion. The relationship is symmetric about $V_{Bias} = 0V$. So, if one TSV is in accumulation for negative bias voltages, it will be in depletion for positive bias voltages.

$$Q = \oint_S \vec{D} * d\vec{s} \tag{48}$$

$$g_i^e = \int_{\Gamma^e} \Psi_i^e \epsilon \frac{\partial V}{\partial n} dl \tag{49}$$

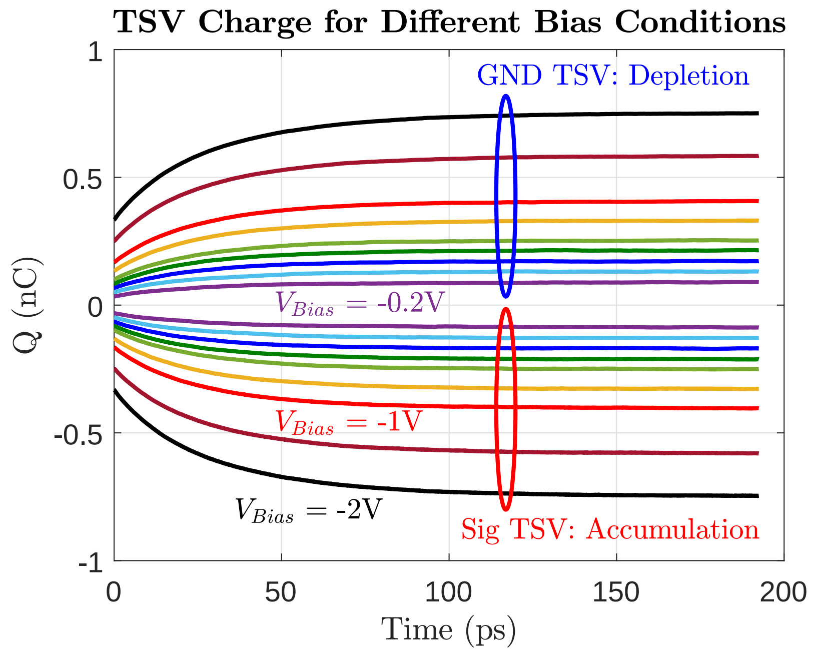

Figure 7. Computed charge on each TSV for different bias conditions. When one TSV is in accumulation, the other TSV is in depletion.

The other two capacitances that can be calculated are the capacitances of each TSV to the zero field point within the substrate. Here, the g_D solution is still used to determine the charge on each TSV. However, V is now the potential difference between the TSV and the zero field point of the substrate. The series combination of these two capacitances gives the capacitance between the signal TSV and the ground TSV.

An example is given in Fig. 8 and Fig. 9, for $V_{Bias} = -1.5V$. In Fig. 8, the lack of particles around the GND TSV clearly shows that the GND TSV is in depletion. Fig. 9 correspondingly shows a large voltage drop in the region around the GND TSV that is devoid of particles. Fig. 9 also clearly shows how the voltage quickly settles to a constant value in areas of the substrate that are away from the depleted and accumulated regions regions around the TSV. Since the voltage is constant in these regions, the electric field goes to zero. This is the zero-field voltage, mentioned above, that is used to calculate the individual TSV capacitances. All three TSV capacitances are given in Fig. 10, for different bias conditions. Qualitatively, the curves agree with the curves given in [16] and [17]. Note that the capacitance for $V_{Bias} = 0V$ was interpolated from surrounding points. This is because, for $V_{Bias} = 0V$, both the signal and GND TSV are at the same potential. This means the electric field throughout the device is weak and random, so particle motion is random, making the Q computation noisy. Plus, Q/V, where V=0, results in an undefined capacitance.

5. CONCLUSION

This paper demonstrated a new approach for calculating the voltage-dependent capacitance of a TSV pair by using Monte Carlo particle simulations. This method is advantageous since it doesn't require any assumptions, outside of knowing and applying the flatband voltage as a boundary condition, to model semiconductor effects. Although results were only shown for a temperature of 300K, this approach can also provide results

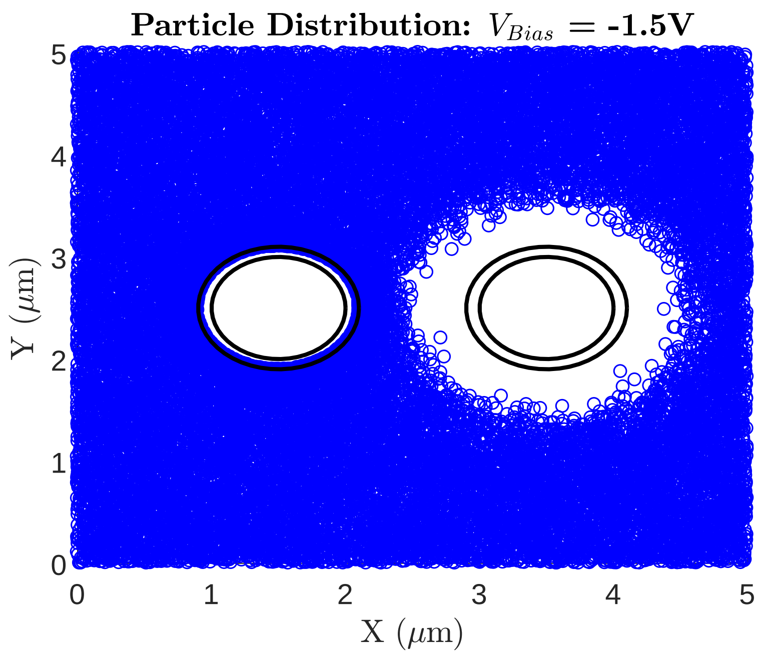

Figure 8. Final particle distribution for $V_{Bias} = -1.5V$. The GND TSV is in depletion and the signal TSV is in accumulation.

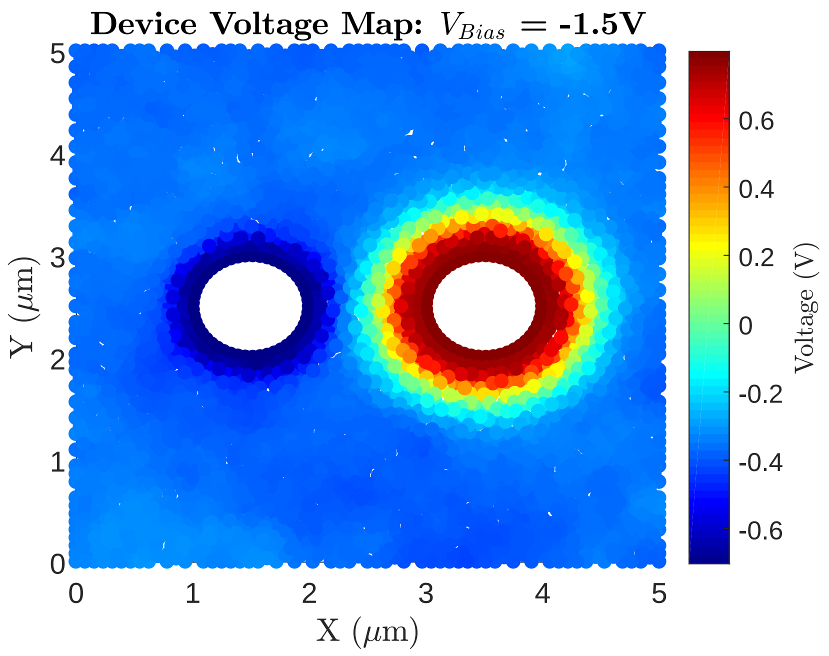

Figure 9. Device voltage map for $V_{Bias} = -1.5V$. The GND TSV is in depletion and the signal TSV is in accumulation.

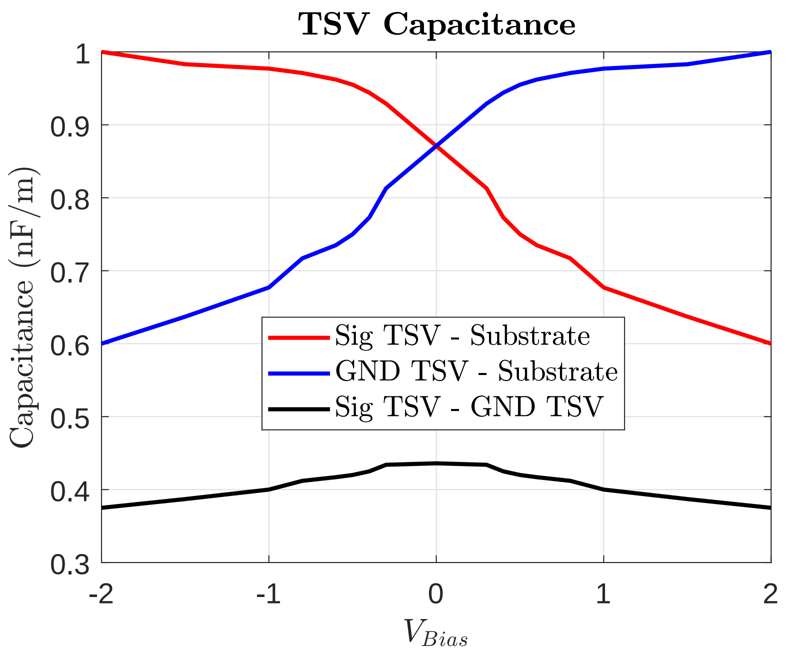

Figure 10. TSV capacitances: Sig TSV - Substrate, GND TSV - Substrate, Sig TSV - GND TSV.

for any other static temperature, as temperature effects are already inherently built into the simulation. Qualitatively, the capacitance curves extracted in this work are similar to those presented by other authors, where completely different techniques were utilized.

REFERENCES

[1] P.A. Thadesar, X. Gu, R. Alapati, and M.S. Bakir, “Through-Silicon-Vias: Drivers, Performance, and Innovations," *IEEE Trans. Compon. Packag. Manuf. Technol.*, vol. 6, no. 7, pp. 1007-1016, July 2016.

[2] Micron, “HybridMemory Cube HMC Gen2," HMC Gen2 datasheet, 2018. Available: https://www.micron.com/~/media/documents/products/data-sheet/hmc/gen2/hmc_gen2.pdf [Accessed: Feb. 14, 2019].

[3] H. Jun, J. Cho, K. Lee, H.Y. Son, K. Kim, H. Jin, and K. Kim, “HBM (High Bandwidth Memory) DRAM Technology and Architecture," *IEEE International Memory Workshop*, May 2017.

[4] "Samsung Starts Producing 8-Gigabyte High Bandwidth Memory-2 with Highest Data Transmission Speed," Jan. 11, 2018. Available: https://www.samsung.com/semiconductor/insights/news-events/samsung-starts-producing-8-gigabyte-high-bandwidth-memory-2-with-highest-data-transmission-speed/ [Accessed: Feb. 14, 2019].

[5] X. Wu, "3D-IC Technologies and 3D FPGA," *IEEE International 3D Systems Integration Conference*, 2015. Available: https://www.xilinx.com/products/silicon-devices/3dic.html [Accessed: Feb. 14, 2019].

[6] C.C. Lee, C.P. Hung, C. Cheung, P.F. Yang, C.L. Kao, D.L. Chen, M.K. Shih, C.L.C. Chien, Y.H. Hsiao, L.C. Chen, M. Su, M. Alfano, J. Siegel, J. Din, and B. Black, "An Overview of the Development of a GPU with integrated HBM on Silicon Interposer," *IEEE 66th Electronic Components and Technology Conference*, 2016.

[7] "International Roadmap for Devices and Systems: Metrology," 2017. Available: https://irds.ieee.org/images/files/pdf/2017/2017IRDS_MET.pdf [Accessed: Feb. 26, 2019].

[8] D. Liu, S. Pan, B. Achkir, and J. Fan, "Fast Admittance Computation for TSV Arrays," *IEEE International Symposium on Electromagnetic Compatibility*, pp. 28-33, 2012.

[9] I. Ndip, B. Curran, K. Lobbicke, S. Guttowski, H. Reichl, K.D. Lang, and H. Henke, "High-Frequency Modeling of TSVs for 3-D Chip Integration and Silicon Interposers Considering Skin-Effect, Dielectric Quasi-TEM and Slow-Wave Modes," *IEEE Trans. Compon. Packag. Manuf. Technol.*, vol. 1, no. 10, pp. 1627-1641, Oct. 2011.

[10] I. Ndip, K. Zoschke, K. Lobbicke, M.J. Wolf, S. Guttowski, H. Reichl, K.D. Lang, and H. Henke, "Analytical, Numerical-, and Measurement-Based Methods for Extracting the Electrical Parameters of Through Silicon Vias (TSVs)," *IEEE. Trans. Compon. Packag. Manuf. Technol.*, vol. 4, no. 3, pp. 504-514, Mar. 2014.

[11] K.J. Han, M. Swaminathan, and J. Jeong, "Modeling of Through-Silicon-Via (TSV) Interposer Considering Depletion Capacitance and Substrate Layer Thickness Effects," *IEEE Trans. Compon. Packag. Manuf. Technol.*, vol. 5, no. 1, pp. 108-118, Jan. 2015.

[12] E.X. Liu, E.P. Li, W.B. Ewe, H.M. Lee, T.G. Lim, and S. Gao, "Compact Wideband Equivalent-Circuit Model for Electrical Modeling of Through-Silicon-Via," *IEEE Trans. Microw. Theory Tech.*, vol. 59, no. 6, pp. 1454-1460, June 2011.

[13] G. Katti, M. Stucchi, K. DeMeyer, and W. Dehaene, "Electrical Modeling and Characterization of Through Silicon Via for Three-Dimensional ICs," *IEEE Trans. Electron Devices*, vol. 57, no. 1, pp. 256-262, Jan. 2010.

[14] T. Bandyopadhyay, K.J. Han, D. Chung, R. Chatterjee, M. Swaminathan, and R. Tummala, "Rigorous Electrical Modeling of Through Silicon Vias (TSVs) with MOS Capacitance Effects," *IEEE Trans. Compon. Packag. Manuf. Technol.*, vol. 1, no. 6, pp. 893-903, June 2011.

[15] C. Hwang, B. Achkir, and J. Fan, “Capacitance-Enhanced Through-Silicon Via for Power Distribution Networks in 3D ICs," *IEEE Electron Device Lett.*, vol. 37, no. 4, pp. 478-481, April 2016.

[16] R. Fang, X. Sun, M. Miao, and Y. Jin, “Characteristics of Coupling Capacitance Between Signal-Ground TSVs Considering MOS Effect in Silicon Interposers," *IEEE Trans. Electron Devices*, vol. 62, no. 12, pp. 4161-4168, Dec. 2015.

[17] W.S. Zhao, J. Zheng, S. Chen, X. Wang, and G. Wang, “Transient Analysis of Through-Silicon Vias in Floating Silicon Substrate," *IEEE Trans. Electromagn. Compat.*, vol. 59, no. 1, pp. 207-215, Feb. 2017.

[18] C. Moglestue, *Monte Carlo Simulation of Semiconductor Devices*. Springer, 1993.

[19] D. Vasileska, S.M. Goodnick, and G. Klimeck, *Computational Electronics: Semiclassical and Quantum Device Modeling and Simulation*. Boca Raton, FL: CRC Press, 2010.

[20] K. Tomizawa, *Numerical Simulation of Submicron Semiconductor Devices*. Boston, MA: Artech House, 1993.

[21] J. Dewey and M.A. Osman, “Monte Carlo Study of Hole Transport in Silicon," *J. Appl. Phys.*, vol. 74, no.5, pg. 3219-3223, 1993.

[22] N. Nintunze and M.A. Osman, “Hole Drift Velocity in the Warped Band Model of GaAs," *Smicond. Sci. Technol.*, vol. 10, pp. 11-17, 1995.

[23] J. Shewchuck, “Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator," *Applied Computation Geometry*, vol. 1148, pp.203-222, 1996.

[24] J.M. Jin, *Theory and Computation of Electromagnetic Fields*. Hoboken, NJ: John Wiley and Sons, Inc., 2010.

[25] “Sparse Documentation," MATLAB Release R2018b, The MathWorks, Inc., Natick, MA. Available: https://www.mathworks.com/help/matlab/ref/sparse.html [Accessed: Feb. 26, 2019].

[26] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, “Shape Distributions," *ACM Trans. on Graphics*, vol. 21, no. 4, pg.807-832, Oct. 2002.

[27] S. Laux, “On Particle-Mesh Coupling in Monte Carlo Semiconductor Device Simulation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 15, no. 10, pp. 1266-1277, Oct. 1996.

[28] M. Aldegunde and K. Kalna, “Energy conserving, self-force free Monte Carlo simulations of semiconductor devices on unstructured meshes," *Computer Physics Communications*, vol. 189, pp. 31-36, April 2015.

SECTION

2. CONCLUSION

2.1. FERRITE CHARACTERIZATION TECHNIQUES

In the first paper of this dissertation, the commercialized Keysight 16454A magnetic material extraction fixture was analyzed. It was found that the given extraction equation for the Keysight fixture is only valid for a single layer of homogeneous material. For ferrite materials made of alternating layers of magnetic and adhesive materials, an updated extraction equation is required. Otherwise, the extracted permeability is an effective permeability for the composite material. This work provides an updated equation that allows the permeability of only the magnetic material to be extracted.

In the second paper, an alternative fixture to the Keysight 16454A fixture was proposed. The fixture, a simple planar coil on a two layer PCB, was validated through simulations and measurements to be able to extract permeability nearly as accurately as the Keysight fixture, for frequencies below 1 MHz. Compared to the Keysight fixture, the PCB coil is inexpensive to manufacture and is thus, easily accessible to engineers. Additionally, the PCB coil does not require the ferrite sample to be reshaped into a toroid.

2.2. PARTICLE SIMULATIONS FOR SEMICONDUCTOR DEVIES

The third paper in this dissertation discussed using particle simulations for analyzing semiconductor devices. In particular, it investigated a pair of TSVs and was able to extract out the voltage dependent capacitance for the pair, where the voltage dependence is due to semiconductor effects. Full-wave solvers based on Maxwell's equations neglect such semiconductor effects. For PCB interconnects, this is acceptable since there are no semiconductor effects. For on-chip interconnects, such as TSVs, these effects must be included. Particle simulations provide one way to include these semiconductor effects. As integration continues to move from the PCB level to the chip level, the ability to rigorously include semiconductor effects for interconnect analysis will become more and more important.

APPENDIX

MATLAB CODE FOR PARTICLE SIMULATIONS

1. MAIN FUNCTION:

SI_TSV_PAIR.M

```
clearvars;
clc;

%------------------User Inputs for Simulation--------------------
%Allowed Geometry: Bounding box of p-type silicon material with
%one signal conductor and one GND conductor. Each conductor
%includes an insulating layer of sio2 around it.

%Assume lower left corner of bounding box starts at (0,0)
xmax=5e-6;             %right side of bounding box
ymax=5e-6;             %top side of bounding box

%Signal TSV Location and Size
xSig=1.5e-6;           %x-coord for center of signal TSV
ySig=2.5e-6;           %y-coord for center of signal TSV
rSig=0.5e-6;           %radius for signal TSV
rI=0.6e-6;             %radius for signal TSV insulator

%GND TSV Location and Size
xGND=3.5e-6;           %x-coord for center of GND TSV
```

```
yGND=2.5e-6;             %y-coord for center of signal TSV
rGND=0.5e-6;             %radius for GND TSV

%Bias voltage between signal and GND TSV
Vbias=-1.0;

num_angles=50;           %# segments for circles
sub_doping=1e21;         %Na for P-Type Substrate
PhiM=4.1;                %Workfunction for Aluminum

T=300;                   %temperature for simulation
tsteps=2500;             %total time steps for simulation

%Controls # of particles assigned to each triangle
particle_multiplier=5;

de=0.002;                %Energy step for scattering table
Vmax=2;                  %Maximum energy for scattering table

%Maximum number of particles to allow in simulation
max_particles=50000;

%Flags--------
pScat=0;                 %Plots scattering rates if =1
pPoly=0;                 %Plots poly segments/nodes and fem mesh
pInit=1;                 %Plots initial particle distribution
pMovie=1;                %Creates a movie of particle motion.
pAVI=1;                  %Creates .avi instead of a .gif
```

```
%--------------------------------------------------------------
%---------------Don't Change Anything Below Here-----------------
%---------------Unless you know what you are doing!--------------
%--------------------------------------------------------------
fprintf('------TSV Simulator - Si P Type Material-------\n\n');

%----------------------Get Constants---------------------------
fprintf('Loading global and material constants...');

sC=getConstants();
%Unpack general constants struct
bk=sC.bk;                       %Boltzmann Constant
q=sC.q;                         %Charge of Electron
h=sC.h;                         %Planck's Constant (/2pi)
eps_o=sC.eps_o;                 %Vacuum Permittivity
emR=sC.emR;                     %Resting Mass of Electron

sM=get_Si_constants(sC);
%Unpack material constants struct
eps_stat=sM.eps_stat;           %Static permittivity of si
A=sM.A;                         %Coefficient for hole warping
B=sM.B;                         %Coefficient for hole warping
C=sM.C;                         %Coefficient for hole warping
emSO=sM.emSO;                   %Mass for split-off hole
Eg=sM.Eg;                       %Band gap for Si
Chi=sM.Chi;                     %Electron affinity for Si
intrin_dop=sM.intrin_dop;       %Intrinsic doping for Si

%Calculate Flat Band Voltage
```

```
PhiS=Chi+Eg/2+(bk*T/q)*log(sub_doping/intrin_dop);
Vbi=PhiM-PhiS;

%Build geometry struct, based on user inputs,
%for passing to functions
sGeom.sub_doping=sub_doping;
sGeom.particle_multiplier=particle_multiplier;
sGeom.xmax=xmax;
sGeom.ymax=ymax;
sGeom.xSig=xSig;
sGeom.ySig=ySig;
sGeom.rSig=rSig;
sGeom.rI=rI;
sGeom.xGND=xGND;
sGeom.yGND=yGND;
sGeom.rGND=rGND;
sGeom.max_particles=max_particles;
sGeom.vSig=Vbias-Vbi;
sGeom.vGnd=-Vbi;

%For Naming Output Files
vSig_str=num2str(Vbias);
spaces_i=find(vSig_str==' ');
vSig_str(spaces_i)='_';
decimal_i=find(vSig_str=='.');
vSig_str(decimal_i)='p';
minus_i=find(vSig_str=='-');
vSig_str(minus_i)='m';
vSig_str=strcat(vSig_str,'V');
```

```
fprintf('done!\n\n');

%------------------Create Scattering Table---------------------
fprintf('Generating scattering tables...');
[scatSi_hole,Gmh,Gml,Gmso]=...
    make_Si_hole_scatTable_3valley(T,de,Vmax,sC,sM,pScat);
ie_max=length(scatSi_hole(1,:,1));
fprintf('done!\n\n');

%---------------Generate FEM Mesh------------------------------
fprintf('Generating FEM mesh...');
[CM,NM,dt]=...
    generate_fem_mesh_tsv_pair(sGeom,num_angles,sC,sM,T,pPoly);
if isempty(CM)
    fprintf('\n\n!!!!Simulation Aborted!!!!\n\n')
    return;
end
fprintf('done!\n\n');

%----------------Get FEM Matrices-----------------------------
fprintf('Building FEM matrices...');
[CM,NM,sFEM]=...
    get_tsv_pair_fem_matrices_sparse(CM,NM,sGeom,sM,sC);
fprintf('done!\n\n');

%Unpack sFEM struct
K_II=sFEM.K_II;
K_ID=sFEM.K_ID;
```

```
K_DI=sFEM.K_DI;
K_DD=sFEM.K_DD;
V_D=sFEM.V_D;
beta1=sFEM.beta1;
beta2=sFEM.beta2;
beta3=sFEM.beta3;
gamma1=sFEM.gamma1;
gamma2=sFEM.gamma2;
gamma3=sFEM.gamma3;
new_order=sFEM.new_order;
tot_nodes=sFEM.tot_nodes;
tot_tri=sFEM.tot_tri;
tri_center=sFEM.tri_center;
tri_area=sFEM.tri_area;
sig_node_i=sFEM.sig_node_i;
gnd_node_i=sFEM.gnd_node_i;

%----------------Initialize Particles-------------------------
fprintf('Initializing particles...');
[particles,valley,part_in_tri,bg_charge,sInit]=...
    init_particles_tsv_pair_fem_neighbors(CM,NM,sFEM,sGeom,...
    sC,sM,Gmh,T,max_particles);
fprintf('done!\n\n');

tri_neighbors=sInit.tri_neighbors;

if pInit==1
    active_particles_i=find(valley~=9);
    drawmesh tsv_pair.1
    hold on
    scatter(particles(active_particles_i,6),...
        particles(active_particles_i,7),'r')
    title('Initial Particle Distribution')
    xlabel('X (m)')
    ylabel('Y (m)')
    set(gca,'FontSize',14)
end
fprintf('done!\n\n');

%-----------------Initial Charge Computation--------------------
fprintf('Computing initial charge distribution...');
bi_vect=...
    charge_tsv_pair_fem_neighbors(valley,part_in_tri,sFEM,...
    CM,sInit.cpsp,bg_charge,q,max_particles);
fprintf('done!\n\n');

%----------------Initial Poisson/Field Computation--------------
fprintf('Computing initial voltages and fields...');
%Solve for V_I
V_I=K_II\(bi_vect-K_ID*V_D);
%Solve for g_D
g_D=K_DI*V_I+K_DD*V_D;
%Assemble V_tot
V_tot=[V_I.' V_D.'].';
%Assemble g_tot
g_I(tot_nodes-length(g_D))=0;
g_tot=[g_I g_D.'].';
```

```
%Put V and g back in original order
put_back_order(new_order)=1:tot_nodes;
Voo=V_tot(put_back_order);
goo=g_tot(put_back_order);

Qsig=sum(goo(sig_node_i));
Qgnd=sum(goo(gnd_node_i));

%Compute E-Field
fx=-1*(1./(2*tri_area)).*((Voo(CM(:,1)))...
    .*beta1+(Voo(CM(:,2))).*beta2+(Voo(CM(:,3))).*beta3);
fy=-1*(1./(2*tri_area)).*((Voo(CM(:,1)))...
    .*gamma1+(Voo(CM(:,2))).*gamma2+(Voo(CM(:,3))).*gamma3);

if pInit == 1
    figure
    scatter3(NM(:,1),NM(:,2),Voo)
    set(gca,'FontSize',14)
    title('Initial Voltage Solution')
    xlabel('X (m)');
    ylabel('Y (m)');
    zlabel('Voltage (V)')

    figure
    quiver(tri_center(:,1),tri_center(:,2),fx,fy,2)
    set(gca,'FontSize',14)
    title('Initial Electric Field')
    xlabel('X (m)')
    ylabel('Y (m)')
    xlim([0-0.1*xmax xmax+0.1*xmax])
    ylim([0-0.1*ymax ymax+0.1*ymax])
end
fprintf('done!\n\n');

%For Nameing Output Files
dt_str=num2str(dt);
dash_mark_i=find(dt_str=='-');
dt_str(dash_mark_i)=[];
decimal_i=find(dt_str=='.');
dt_str(decimal_i)='p';

angle_vect=linspace(0,2*pi,num_angles);
if pMovie==1
    if pAVI==1
        filename=...
            sprintf('tsv_pair_fem_%s_dt_%s.avi',...
            vSig_str,dt_str);
        v=VideoWriter(filename);
        v.FrameRate=5;
        v.Quality=100;
        open(v)
    else
        filename=...
            sprintf('tsv_pair_fem_%s_dt_%s.gif',...
            vSig_str,dt_str);
    end
    figure
    hH=scatter(particles(active_particles_i,6),...
        particles(active_particles_i,7),'b');
    hold on
    plot(xSig+rSig.*cos(angle_vect),...
        ySig+rSig.*sin(angle_vect),'b')
    plot(xSig+rI.*cos(angle_vect),...
        ySig+rI.*sin(angle_vect),'k')
    plot(xGND+rGND.*cos(angle_vect),...
        yGND+rGND.*sin(angle_vect),'b')
    plot(xGND+rI.*cos(angle_vect),...
        yGND+rI.*sin(angle_vect),'k')
    xlim([0-0.1*xmax xmax+0.1*xmax])
    ylim([0-0.1*ymax ymax+0.1*ymax])
    xlabel('X (m)');
    ylabel('Y (m)')
    title('Particle Distribution Within Device');

    frame=getframe(gcf);
    if pAVI==1
        writeVideo(v,frame);
    else
        im=frame2im(frame);
        [imind,cm]=rgb2ind(im,256);
        imwrite(imind,cm,filename,'gif','Loopcount',inf);
    end
end

%Save Initial State
```

```
file_str=...
    sprintf('tsv_pair_fem_%s_dt_%s_initial.mat',...
    vSig_str,dt_str);
save(file_str);

%------------------Main Loop------------------------------------
Qvect(tsteps,2)=0;
for ti=1:tsteps
    fprintf('Time Step: %i\n',ti);
    t=(ti-1)*dt;
    tdt=t+dt;

    for n=1:max_particles
        if valley(n,1)~=9
            ts=particles(n,4);  %Get scatter time
            t1=t;

            while ts < tdt
                tau=ts-t1;
                %Drift-----------------------------------------
                iv=valley(n,1);
                if iv~=9
                    kx=particles(n,1);
                    ky=particles(n,2);
                    kz=particles(n,3);
                    ei=particles(n,5);
                    xi=particles(n,6);
                    yi=particles(n,7);
                    iv=valley(n,1);
```

```
%Find distance to nearest triangle center
%no need to perform sqrt though...just
%wasted computation
tri_i=part_in_tri(n);
dist_to_tri=...
    ((xi-tri_center(tri_neighbors(tri_i,...
    :),1)).^2)+((yi-tri_center(...
    tri_neighbors(tri_i,:),2)).^2);
[~, min_i]=min(dist_to_tri);
min_i=tri_neighbors(tri_i,min_i);
part_in_tri(n)=min_i;

%Momentum Change from Applied Field
dkx=(q/h)*fx(min_i)*tau;
dky=(q/h)*fy(min_i)*tau;

kxf=kx+dkx;
kyf=ky+dky;

kf=sqrt(kxf*kxf+kyf*kyf+kz*kz);
cos_t=kz/kf;
sin_t=sqrt(1-cos_t*cos_t);
sin_p=kyf/kf/sin_t;
cos_p=kxf/kf/sin_t;

sin2t=sin_t*sin_t;
cos2t=cos_t*cos_t;
sin4t=sin2t*sin2t;
```

```
cos2p=cos_p*cos_p;
sin2p=sin_p*sin_p;
g=sqrt(((B/A)^2)+((C/A)^2)*...
    (sin2t*cos2t+sin4t*cos2p*sin2p));

%Heavy Hole
if iv == 1
    ef=(abs(A)*h*h*kf*kf/(2*emR))*(1-g);
    ef=ef/q;
    xf=xi+(abs(A)*h*(1-g)*...
        (kx+0.5*dkx)/emR)*tau;
    yf=yi+(abs(A)*h*(1-g)*...
        (ky+0.5*dky)/emR)*tau;
%Light Hole
elseif iv == 2
    ef=(abs(A)*h*h*kf*kf/(2*emR))*(1+g);
    ef=ef/q;
    xf=xi+(abs(A)*h*(1+g)*...
        (kx+0.5*dkx)/emR)*tau;
    yf=yi+(abs(A)*h*(1+g)*...
        (ky+0.5*dky)/emR)*tau;
%Split-Off Hole
else
    ef=(h*h*kf*kf)/(2*emSO);
    ef=ef/q;
    xf=xi+(h/emSO)*(kx+0.5*dkx)*tau;
    yf=yi+(h/emSO)*(ky+0.5*dky)*tau;
end
```

```
%-----Boundary Conditions----
%Left Side of Bounding Box
if xf < 0
    xf=-xf;
    kxf=-kxf;
%Right Side of Bounding Box
elseif xf > xmax
    xf=xmax-(xf-xmax);
    kxf=-kxf;
end

%Bottom of Bounding Box
if yf < 0
    yf=-yf;
    kyf=-kyf;
%Top of Bounding Box
elseif yf > ymax
    yf=ymax-(yf-ymax);
    kyf=-kyf;
end

%Insulators
dist_to_sig_insul=...
    sqrt(((xf-xSig).^2)+((yf-ySig).^2));
dist_to_gnd_insul=...
    sqrt(((xf-xGND).^2)+((yf-yGND).^2));

%Find which insulator particle is inside,
%if any
```

```
if dist_to_sig_insul < rI
    [xff,yff,kx_part,ky_part]=...
        particle_reflect_from_insulator(...
        xSig,ySig,rI,xi,yi,xf,yf);
    xf=xff;
    yf=yff;
    kxf=sqrt(kf*kf-kz*kz)*kx_part;
    kyf=sqrt(kf*kf-kz*kz)*ky_part;
elseif dist_to_gnd_insul < rI
    [xff,yff,kx_part,ky_part]=...
        particle_reflect_from_insulator(...
        xGND,yGND,rI,xi,yi,xf,yf);
    xf=xff;
    yf=yff;
    kxf=sqrt(kf*kf-kz*kz)*kx_part;
    kyf=sqrt(kf*kf-kz*kz)*ky_part;
end

%Update Particle Attributes
particles(n,1)=kxf;
particles(n,2)=kyf;
particles(n,5)=ef;
particles(n,6)=xf;
particles(n,7)=yf;

%Scatter-------------------------------------
if valley(n,1)~=9
    [particles(n,:),valley(n,1),scatType...
        ]=si_scat_emc_hole_3valley(...
                    particles(n,:),valley(n,1),...
                    scatSi_hole,de,sC,sM,ie_max);

                t1=ts;

                %New scattering time based on valley
                iv=valley(n,1);

                if iv==1
                    ts=t1-log(rand())/Gmh;
                elseif iv==2
                    ts=t1-log(rand())/Gml;
                else
                    ts=t1-log(rand())/Gmso;
                end
            end

        else
            ts=tdt;
        end
    end

    tau=tdt-t1;
    %Drift---------------------------------------------
    iv=valley(n,1);

    if iv~=9
        kx=particles(n,1);
        ky=particles(n,2);
```

```
kz=particles(n,3);

xi=particles(n,6);
yi=particles(n,7);



%Find distance to nearest triangle center
%No need to perform sqrt though...just
%wasted computation
tri_i=part_in_tri(n);
dist_to_tri=...
    ((xi-tri_center(tri_neighbors(tri_i,:),...
    1)).^2)+((yi-tri_center(...
    tri_neighbors(tri_i,:),2)).^2);
[~, min_i]=min(dist_to_tri);
min_i=tri_neighbors(tri_i,min_i);
part_in_tri(n)=min_i;

%Momentum Change from Applied Field
dkx=(q/h)*fx(min_i)*tau;
dky=(q/h)*fy(min_i)*tau;

kxf=kx+dkx;
kyf=ky+dky;

kf=sqrt(kxf*kxf+kyf*kyf+kz*kz);
cos_t=kz/kf;
sin_t=sqrt(1-cos_t*cos_t);
sin_p=kyf/kf/sin_t;
```

```
cos_p=kxf/kf/sin_t;


sin2t=sin_t*sin_t;
cos2t=cos_t*cos_t;
sin4t=sin2t*sin2t;
cos2p=cos_p*cos_p;
sin2p=sin_p*sin_p;
g=sqrt(((B/A)^2)+((C/A)^2)...
    *(sin2t*cos2t+sin4t*cos2p*sin2p));


%Heavy Hole
if iv == 1
    ef=(abs(A)*h*h*kf*kf/(2*emR))*(1-g);
    ef=ef/q;
    xf=xi+(abs(A)*h*(1-g)*...
        (kx+0.5*dkx)/emR)*tau;
    yf=yi+(abs(A)*h*(1-g)*...
        (ky+0.5*dky)/emR)*tau;
%Light Hole
elseif iv == 2
    ef=(abs(A)*h*h*kf*kf/(2*emR))*(1+g);
    ef=ef/q;
    xf=xi+(abs(A)*h*(1+g)*...
        (kx+0.5*dkx)/emR)*tau;
    yf=yi+(abs(A)*h*(1+g)*...
        (ky+0.5*dky)/emR)*tau;
%Split-Off Hole
else
    ef=(h*h*kf*kf)/(2*emSO);
    ef=ef/q;
    xf=xi+(h/emSO)*(kx+0.5*dkx)*tau;
    yf=yi+(h/emSO)*(ky+0.5*dky)*tau;
end

%-----Boundary Conditions----
%Left Side of Bounding Box
if xf < 0
    xf=-xf;
    kxf=-kxf;
%Right Side of Bounding Box
elseif xf > xmax
    xf=xmax-(xf-xmax);
    kxf=-kxf;
end

%Bottom of Bounding Box
if yf < 0
    yf=-yf;
    kyf=-kyf;
%Top of Bounding Box
elseif yf > ymax
    yf=ymax-(yf-ymax);
    kyf=-kyf;
end

%Insulators
dist_to_sig_insul=sqrt(((xf-xSig).^2)+...
    ((yf-ySig).^2));
```

```
dist_to_gnd_insul=sqrt(((xf-xGND).^2)+...
    ((yf-yGND).^2));

%Find which insulator particle is inside,
%if any
if dist_to_sig_insul < rI
    [xff,yff,kx_part,ky_part]=...
        particle_reflect_from_insulator(...
        xSig,ySig,rI,xi,yi,xf,yf);
    xf=xff;
    yf=yff;
    kxf=sqrt(kf*kf-kz*kz)*kx_part;
    kyf=sqrt(kf*kf-kz*kz)*ky_part;
elseif dist_to_gnd_insul < rI
    [xff,yff,kx_part,ky_part]=...
        particle_reflect_from_insulator(...
        xGND,yGND,rI,xi,yi,xf,yf);
    xf=xff;
    yf=yff;
    kxf=sqrt(kf*kf-kz*kz)*kx_part;
    kyf=sqrt(kf*kf-kz*kz)*ky_part;
end

%Update Particle Attributes
particles(n,1)=kxf;
particles(n,2)=kyf;
particles(n,4)=ts;
particles(n,5)=ef;
particles(n,6)=xf;
```

```
                particles(n,7)=yf;
            end
        end
    end

    %-------------------Charge Computation--------------------
    bi_vect=...
        charge_tsv_pair_fem_neighbors(valley,part_in_tri,...
        sFEM,CM,sInit.cpsp,bg_charge,q,max_particles);

    %----------------Poisson Computation---------------------
    %Solve for V_I
    V_I=K_II\(bi_vect-K_ID*V_D);
    %Assemble V_tot
    V_tot=[V_I.' V_D.'].';
    %Solve for g_D
    g_D=K_DI*V_I+K_DD*V_D;
    %Assemble g_tot
    g_I(tot_nodes-length(g_D))=0;
    g_tot=[g_I g_D.'].';

    %Put vectors back in original order
    Voo=V_tot(put_back_order);
    goo=g_tot(put_back_order);

    Qvect(ti,1)=sum(goo(sig_node_i));
    Qvect(ti,2)=sum(goo(gnd_node_i));

    %---------------E-Field Computation----------------------
    fx=-1*(1./(2*tri_area)).*((Voo(CM(:,1)))...
        .*beta1+(Voo(CM(:,2))).*beta2+(Voo(CM(:,3))).*beta3);
    fy=-1*(1./(2*tri_area)).*((Voo(CM(:,1)))...
        .*gamma1+(Voo(CM(:,2))).*gamma2+(Voo(CM(:,3))).*gamma3);

    if pMovie==1
        active_particles_i=find(valley~=9);
        set(hH,'XData',particles(active_particles_i,6),...
            'YData',particles(active_particles_i,7));
        drawnow;

        frame=getframe(gcf);
        if pAVI==1
            writeVideo(v,frame);
        else
            im=frame2im(frame);
            [imind,cm]=rgb2ind(im,256);
            imwrite(imind,cm,filename,'gif',...
                'WriteMode','append');
        end
    end

    if mod(ti,200) == 0
        file_str=sprintf('tsv_pair_fem_%s_dt_%s_ti_%i.mat',...
            vSig_str,dt_str,ti);
        save(file_str);
    end
end
```

```
%Save Last State
file_str=sprintf('tsv_pair_fem_%s_dt_%s_ti_%i.mat',...
    vSig_str,dt_str,ti);
save(file_str);

if pAVI==1
    close(v);
end
```

2. CONSTANTS FILE:

GETCONSTANTS.M

```
function sC = getConstants()
%Returns structure (sC) containing non-material
%specific constants

%Boltzmann's Constant
sC.bk=1.38066e-23;

%Charge of Electron
sC.q=1.60219e-19;

%Planck's Constant (/2pi) (J*s)
sC.h=1.05459e-34;

%Mass of Resting Electron
sC.emR=9.10953e-31;

%Vacuum Permittivity
```

```
sC.eps_o=8.85*10^-12;


end
```

3. SI CONSTANTS FILE:
GET_SI_CONSTANTS.M

```
function sSi = get_Si_constants(sC)
%Requires sC (struct holding general constants) as input
%Returns structure (sSi) containing Si specific constants


%-------------------References--------------------------------
%1. Numerical Simulation of Submicron Semiconductor Devices
%   by Tomizawa (pgs. 256-260)
%2. Monte Carlo Study of Electron Transport in Silicon
%   Inversion Layers by M.V. Fischetti and S.E. Laux (pg 48)
%3. Carrier Transport in Nanoscale MOS Transistors by
%   H. Tsuchiya and Y. Kamakura (pg. 46)
%-------------------------------------------------------------


sSi.Eg=1.12;                          %Band Gap for Si
sSi.Eso=0.044;                        %Gap Between H/L and SO


%Mass of Electron in X Band: Longitudinal Direction
sSi.emL=0.92*sC.emR;
%Mass of Electron in X Band: Transverse Direction
sSi.emT=0.19*sC.emR;
sSi.emC=(3*sSi.emL*sSi.emT)/(sSi.emT+2*sSi.emL);
%Density of States Mass for Herring Vogt Transform
```

```
sSi.emD=(sSi.emL*sSi.emT*sSi.emT)^(1/3);
sSi.eM=[sSi.emL,sSi.emT];

sSi.alpha=(1/sSi.Eg)*(1-(sSi.emC/sC.emR))^2;

sSi.eps_si=11.7;                    %Permittivity for Si
sSi.eps_sio2=3.9;                   %Permittivity for SiO2
sSi.eps_stat=sSi.eps_si*sC.eps_o;   %Static Permittivity



sSi.Chi=4.05;                       %Electron Affinity for Si

%Intrinsic Carrier Concentration (1/m^3, not 1/cm^3!)
sSi.intrin_dop=1.5*10^16;

%For Acoustic Phonon Scattering
sSi.rho=2329;                       %Crystal Density (kg/m^3)
sSi.sv=9040;                        %Sound Velocity (m/s)
sSi.cl=sSi.rho*sSi.sv*sSi.sv;       %Elastic Constant

%Acoustic Deformation Potential
sSi.adp=9.5*sC.q; %Fischetti:9,Tomizawa:6.55

%Intervalley Phonon Scattering
%f-scattering, TA deformation potential (eV/m)
sSi.FTA=3e9*sC.q;
%f-scattering, LA deformation potential (eV/m)
sSi.FLA=2e10*sC.q;
%f-scattering, TO deformation potential (eV/m)
```

```
sSi.FTO=2e10*sC.q;
%g-scattering, TA deformation potential (eV/m)
sSi.GTA=5e9*sC.q;
%g-scattering, LA deformation potential (eV/m)
sSi.GLA=8e9*sC.q;
%g-scattering, LO deformation potential (eV/m)
sSi.GLO=1.1e11*sC.q;
sSi.FG=[sSi.FTA,sSi.FLA,sSi.FTO,sSi.GTA,sSi.GLA,sSi.GLO];
%-----
%f-scattering, TA phonon energy (eV)
sSi.hw_FTA=0.019;
%f-scattering, LA phonon energy (eV)
sSi.hw_FLA=0.0474;
%f-scattering, TO phonon energy (eV)
sSi.hw_FTO=0.059;
%g-scattering, TA phonon energy (eV)
sSi.hw_GTA=0.012;
%g-scattering, LA phonon energy (eV)
sSi.hw_GLA=0.0185;
%g-scattering, LO phonon energy (eV)
sSi.hw_GLO=0.0612;
sSi.hwFG=[sSi.hw_FTA,sSi.hw_FLA,sSi.hw_FTO,sSi.hw_GTA,...
    sSi.hw_GLA,sSi.hw_GLO];
%----
%Number of Equivalent Valleys for g-scattering
sSi.ZG=1;
%Number of Equivalent Valleys for f-scattering
sSi.ZF=4;
```

```
%Inverse Band Mass Parameters for Si Holes

sSi.A=-4.22;

sSi.B=-0.78;

sSi.C=4.80;


%Max and min for g function for H/L holes

sSi.g111=sqrt(((sSi.B/sSi.A)^2)+(1/3)*((sSi.C/sSi.A)^2));

sSi.g100=sSi.B/sSi.A;


%Optical Phonon Energy

sSi.hwo=0.063;

sSi.hwoq=sSi.hwo*sC.q;

sSi.wo=sSi.hwoq/sC.h;


%Optical Deformation Potential for Holes (eV/m)

sSi.DK=5*10^10*sC.q;


%Effective Mass for Split-Off Band

sSi.emSO=0.29*sC.emR;


%Average Effective Mass for Heavy Holes

sSi.emh=0.495*sC.emR;


%Average Effective Mass for Light Holes

sSi.eml=0.1614*sC.emR;
```

4. SCATTERING TABLE FUNCTION: MAKE_SI_HOLE_SCAT_TABLE_3VALLEY.M

```
function [scatSi_hole,Gmh,Gml,Gmso]=...
    make_Si_hole_scatTable_3valley(T,de,Vmax,sC,sM,pScat)


%------------------Hole Energy Steps for Graphs------------------
%Hole Energy Step
delt_Ek=de;
%Number of sample points for hole energy
Ek_pts=Vmax/de;
eV_axis=(1:Ek_pts)*delt_Ek;
%Matrix holding scattering rates for heavy holes
scat_h(10,Ek_pts)=0;
%Matrix holding scattering rates for light holes
scat_l(10,Ek_pts)=0;
%Matrix holding scattering rates for split-off holes
scat_so(10,Ek_pts)=0;


%----------------General Constants------------------------------
%Unpack from struct of constants passed from main
q=sC.q;                         %Charge of electron
bk=sC.bk;                       %Boltzmann's Constant
h=sC.h;                         %Planck's Constant (/2pi)
emR=sC.emR;                     %Resting Mass of Electron


%---------------Si Specific Constants----------------------------
%Unpack from struct of material constants passed from main
%Inverse band mass parameters for H/L holes
```

```
A=sM.A;

B=sM.B;

C=sM.C;

%Effective Mass for Split-Off Valence Band

emSO=sM.emSO;

%Energy Difference Between H/L and Split-Off Bands

Eso=sM.Eso;



%------------Calculate Max Masses for H/L Holes----------------

theta=0:0.1:2*pi;

phi=0:0.1:pi;

g(length(theta),length(phi))=0;

emh(length(theta),length(phi))=0;

eml(length(theta),length(phi))=0;

for i=1:length(theta)

    for j=1:length(phi)

        sin2t=sin(theta(i))^2;

        cos2t=cos(theta(i))^2;

        sin4t=sin2t*sin2t;

        cos2p=cos(phi(j))^2;

        sin2p=sin(phi(j))^2;

        g(i,j)=sqrt((B/A)*(B/A)+(C/A)*(C/A)...

            *(sin2t*cos2t+sin4t*cos2p*sin2p));

        emh(i,j)=1/(abs(A)*(1-g(i,j)));

        eml(i,j)=1/(abs(A)*(1+g(i,j)));

    end

end
```

```
emh_max=max(max(emh));
emh_max=emh_max*emR;
eml_max=max(max(eml));
eml_max=eml_max*emR;

%-----------Acoustic Phonon Scattering-------------------------
%Nintunze rates seem to have an extra 1/pi term
%So use Tomizawa equation for spherical/parabolic
%but include overlap factor of 1/2 from Nintunze
%and max hole mass

adp=sM.adp;                   %Acoustic Deformation Potential
cl=sM.cl;                     %Elastic Constant for Si
acoustic_const=(2*pi*adp*adp*bk*T)/(h*cl);
overlap_factor=0.5;
for i=1:Ek_pts
    ei=delt_Ek*i;

    %Final is Heavy
    N_Ek=(((2*emh_max)^(3/2))/(4*pi*pi*h*h*h))*sqrt(ei*q);
    P=overlap_factor*acoustic_const*N_Ek;

    %Heavy to Heavy
    scat_h(1,i)=P;
    %Light to Heavy
    scat_l(2,i)=P;

    %Final is Light
    N_Ek=(((2*eml_max)^(3/2))/(4*pi*pi*h*h*h))*sqrt(ei*q);
    P=overlap_factor*acoustic_const*N_Ek;

    %Light to Light
    scat_l(1,i)=P;
    %Heavy to Light
    scat_h(2,i)=P;

    %Split-Off Band - Intraband
    N_Ek=(((2*emSO)^(3/2))/(4*pi*pi*h*h*h))*sqrt(ei*q);
    scat_so(1,i)=acoustic_const*N_Ek;

    %No Interband for Split-Off
    scat_so(2,i)=0;
end

%---------NPOP Scattering--------------------------------------
%Nintunze rates seem to have an extra 1/pi term
%So use Tomizawa equation for spherical/parabolic
%but include overlap factor of 1/2 from Nintunze and
%max hole mass

hwo=sM.hwo;
hwoq=sM.hwoq;
wo=sM.wo;
DK=sM.DK;
rho=sM.rho;
npop_const=(pi*DK*DK)/(rho*wo);
overlap_factor=0.5;
No=1/(exp((hwoq)/(bk*T))-1);
```

```
for i=1:Ek_pts
    ei=delt_Ek*i;

    ef=ei+hwo;
    %Absorption (End is Heavy)-----------------------------------
    N_Ek=(((2*emh_max)^(3/2))/(4*pi*pi*h*h*h))*sqrt(ef*q);
    P=overlap_factor*npop_const*N_Ek*No;

    %Heavy-Heavy Absorption
    scat_h(3,i)=P;
    %Empty slot for H-H Interband Combo
    scat_h(4,i)=0;
    %Light-Heavy Absorption
    scat_l(4,i)=P;

    %SplitOff-Heavy Absorption
    ef=ei+hwo+Eso;
    N_Ek=(((2*emh_max)^(3/2))/(4*pi*pi*h*h*h))*sqrt(ef*q);
    P=overlap_factor*npop_const*N_Ek*No;

    if ef > 0
        scat_so(4,i)=P;
    else
        scat_so(4,i)=0;
    end

    %Absorption (End is Light)-----------------------------------
    ef=ei+hwo;
    N_Ek=(((2*eml_max)^(3/2))/(4*pi*pi*h*h*h))*sqrt(ef*q);
```

```
P=overlap_factor*npop_const*N_Ek*No;


%Light-Light Absorption

scat_l(3,i)=P;

%Empty slot for L-L Interband Combo

scat_l(5,i)=0;

%Heavy-Light Absorption

scat_h(5,i)=P;


%SplitOff-Light Absorption

ef=ei+hwo+Eso;

N_Ek=(((2*eml_max)^(3/2))/(4*pi*pi*h*h*h))*sqrt(ef*q);

P=overlap_factor*npop_const*N_Ek*No;

if ef > 0

    scat_so(5,i)=P;

else

    scat_so(5,i)=0;

end


%Absorption (End is Split-Off)-----------------------------

%SplitOff-SplitOff Absorption

ef=ei+hwo;

N_Ek=(((2*emSO)^(3/2))/(4*pi*pi*h*h*h))*sqrt(ef*q);

P=npop_const*N_Ek*No;

scat_so(3,i)=P;

%Empty slot for SO-SO Interband Combo

scat_so(6,i)=0;


%Heavy-SplitOff Absorption
```

```
ef=ei+hwo-Eso;
N_Ek=(((2*emSO)^(3/2))/(4*pi*pi*h*h*h))*sqrt(ef*q);
P=npop_const*N_Ek*No;
scat_h(6,i)=P;
%Light-SplitOff Absorption
scat_l(6,i)=P;

%Emission (Not Between H/L and SO or SO and H/L)-----------
ef=ei-hwo;
if ef>0
    %Emission (End is Heavy)
    N_Ek=(((2*emh_max)^(3/2))/(4*pi*pi*h*h*h))*sqrt(ef*q);
    P=overlap_factor*npop_const*N_Ek*(No+1);

    %Heavy-Heavy Emission
    scat_h(7,i)=P;
    %Empty slot for H-H Interband Combo
    scat_h(8,i)=0;
    %Light-Heavy Emission
    scat_l(8,i)=P;

    %Emission (End is Light)
    N_Ek=(((2*eml_max)^(3/2))/(4*pi*pi*h*h*h))*sqrt(ef*q);
    P=overlap_factor*npop_const*N_Ek*(No+1);

    %Light-Light Emission
    scat_l(7,i)=P;
    %Empty slot for L-L Interband Combo
    scat_l(9,i)=0;
```

```
        %Heavy-Light Emission
        scat_h(9,i)=P;


        %Emission (End is Split-Off)
        N_Ek=(((2*emSO)^(3/2))/(4*pi*pi*h*h*h))*sqrt(ef*q);
        P=npop_const*N_Ek*(No+1);


        %SplitOff-SplitOff Emission
        scat_so(7,i)=P;
        %Empty slot for SO-SO Interband Combo
        scat_so(10,i)=0;
    else
        scat_h(7,i)=0;
        scat_h(8,i)=0;


        scat_l(7,i)=0;
        scat_l(9,i)=0;


        scat_so(7,i)=0;
        scat_so(10,i)=0;
    end


    %Emission (Between H/L and SO)-----------------------------
    ef=ei-hwo-Eso;
    if ef > 0
        %Heavy/Light to Split-Off Emission
        N_Ek=(((2*emSO)^(3/2))/(4*pi*pi*h*h*h))*sqrt(ef*q);
        P=npop_const*N_Ek*(No+1);
        scat_h(10,i)=P;
        scat_l(10,i)=P;
    else
        scat_h(10,i)=0;
        scat_l(10,i)=0;
    end

    %Emission (Between SO and H/L)-----------------------------
    ef=ei-hwo+Eso;
    if ef > 0
        %Split-Off to Heavy Emission
        N_Ek=(((2*emh_max)^(3/2))/(4*pi*pi*h*h*h))*sqrt(ef*q);
        P=overlap_factor*npop_const*N_Ek*(No+1);

        scat_so(8,i)=P;

        %Split-Off to Light Emission
        N_Ek=(((2*eml_max)^(3/2))/(4*pi*pi*h*h*h))*sqrt(ef*q);
        P=overlap_factor*npop_const*N_Ek*(No+1);

        scat_so(9,i)=P;
    else
        scat_so(8,i)=0;
        scat_so(9,i)=0;
    end
end

scatSi_hole(10,Ek_pts,3)=0;
scatSi_hole(1,:,1)=scat_h(1,:);
scatSi_hole(2,:,1)=scatSi_hole(1,:,1)+scat_h(2,:);
scatSi_hole(3,:,1)=scatSi_hole(2,:,1)+scat_h(3,:);
scatSi_hole(4,:,1)=scatSi_hole(3,:,1)+scat_h(4,:);
scatSi_hole(5,:,1)=scatSi_hole(4,:,1)+scat_h(5,:);
scatSi_hole(6,:,1)=scatSi_hole(5,:,1)+scat_h(6,:);
scatSi_hole(7,:,1)=scatSi_hole(6,:,1)+scat_h(7,:);
scatSi_hole(8,:,1)=scatSi_hole(7,:,1)+scat_h(8,:);
scatSi_hole(9,:,1)=scatSi_hole(8,:,1)+scat_h(9,:);
scatSi_hole(10,:,1)=scatSi_hole(9,:,1)+scat_h(10,:);

Gmh=max(scatSi_hole(10,:,1));
scatSi_hole(:,:,1)=scatSi_hole(:,:,1)./Gmh;

scatSi_hole(1,:,2)=scat_l(1,:);
scatSi_hole(2,:,2)=scatSi_hole(1,:,2)+scat_l(2,:);
scatSi_hole(3,:,2)=scatSi_hole(2,:,2)+scat_l(3,:);
scatSi_hole(4,:,2)=scatSi_hole(3,:,2)+scat_l(4,:);
scatSi_hole(5,:,2)=scatSi_hole(4,:,2)+scat_l(5,:);
scatSi_hole(6,:,2)=scatSi_hole(5,:,2)+scat_l(6,:);
scatSi_hole(7,:,2)=scatSi_hole(6,:,2)+scat_l(7,:);
scatSi_hole(8,:,2)=scatSi_hole(7,:,2)+scat_l(8,:);
scatSi_hole(9,:,2)=scatSi_hole(8,:,2)+scat_l(9,:);
scatSi_hole(10,:,2)=scatSi_hole(9,:,2)+scat_l(10,:);

Gml=max(scatSi_hole(10,:,2));
scatSi_hole(:,:,2)=scatSi_hole(:,:,2)./Gml;

scatSi_hole(1,:,3)=scat_so(1,:);
scatSi_hole(2,:,3)=scatSi_hole(1,:,3)+scat_so(2,:);
scatSi_hole(3,:,3)=scatSi_hole(2,:,3)+scat_so(3,:);
scatSi_hole(4,:,3)=scatSi_hole(3,:,3)+scat_so(4,:);
scatSi_hole(5,:,3)=scatSi_hole(4,:,3)+scat_so(5,:);
scatSi_hole(6,:,3)=scatSi_hole(5,:,3)+scat_so(6,:);
scatSi_hole(7,:,3)=scatSi_hole(6,:,3)+scat_so(7,:);
scatSi_hole(8,:,3)=scatSi_hole(7,:,3)+scat_so(8,:);
scatSi_hole(9,:,3)=scatSi_hole(8,:,3)+scat_so(9,:);
scatSi_hole(10,:,3)=scatSi_hole(9,:,3)+scat_so(10,:);

Gmso=max(scatSi_hole(10,:,3));
scatSi_hole(:,:,3)=scatSi_hole(:,:,3)./Gmso;

if pScat==1
    figure
    semilogy(eV_axis,scat_h(1,:),'r','LineWidth',2)
    hold on
    semilogy(eV_axis,scat_h(2,:),'r--','LineWidth',2)
    semilogy(eV_axis,scat_h(3,:),'b','LineWidth',2)
    semilogy(eV_axis,scat_h(5,:),'b--','LineWidth',2)
    semilogy(eV_axis,scat_h(6,:),'b:','LineWidth',2)
    semilogy(eV_axis,scat_h(7,:),'c','LineWidth',2);
    semilogy(eV_axis,scat_h(9,:),'c--','LineWidth',2);
    semilogy(eV_axis,scat_h(10,:),'c:','LineWidth',2);
    grid on
    xlabel('Energy (eV)','Interpreter','latex')
    ylabel('Scattering Rate (1/s)','Interpreter','latex')
    title('\textbf{Hole Scattering Rates for Heavy Valley}'...
        ,'Interpreter','latex')
    leg=legend('Ac:H-H','Ac:H-L','NPOP-Ab:H-H',...
        'NPOP-Ab:H-L','NPOP-Ab:H-SO','NPOP-Em:H-H',...
        'NPOP-Em:H-L','NPOP-Em:H-SO');
    leg.Interpreter='latex';
    set(gca,'FontSize',14)

    figure
    semilogy(eV_axis,scat_l(1,:),'r','LineWidth',2)
    hold on
    semilogy(eV_axis,scat_l(2,:),'r--','LineWidth',2)
    semilogy(eV_axis,scat_l(3,:),'b','LineWidth',2)
    semilogy(eV_axis,scat_l(4,:),'b--','LineWidth',2)
    semilogy(eV_axis,scat_l(6,:),'b:','LineWidth',2)
    semilogy(eV_axis,scat_l(7,:),'c','LineWidth',2)
    semilogy(eV_axis,scat_l(8,:),'c--','LineWidth',2);
    semilogy(eV_axis,scat_l(10,:),'c:','LineWidth',2);
    grid on
    xlabel('Energy (eV)')
    ylabel('Scattering Rate (1/s)')
    title('Hole Scattering Rates for Light Valley')
    legend('Ac:L-L','Ac:L-H','NPOP-Ab:L-L',...
        'NPOP-Ab:L-H','NPOP-Ab:L-SO','NPOP-Em:L-L',...
        'NPOP-Em:L-H','NPOP-Em:L-SO')
    set(gca,'FontSize',14)

    figure
    semilogy(eV_axis,scat_so(1,:),'r','LineWidth',2)
    hold on
    semilogy(eV_axis,scat_so(3,:),'b','LineWidth',2)
    semilogy(eV_axis,scat_so(4,:),'b--','LineWidth',2)
    semilogy(eV_axis,scat_so(5,:),'b:','LineWidth',2)
```

```
    semilogy(eV_axis,scat_so(7,:),'c','LineWidth',2)
    semilogy(eV_axis,scat_so(8,:),'c--','LineWidth',2);
    semilogy(eV_axis,scat_so(9,:),'c:','LineWidth',2);
    grid on
    xlabel('Energy (eV)')
    ylabel('Scattering Rate (1/s)')
    title('Hole Scattering Rates for Split-Off Valley')
    legend('Ac:SO-SO','NPOP-Ab:SO-SO',...
        'NPOP-Ab:SO-H','NPOP-Ab:SO-L','NPOP-Em:SO-SO',...
        'NPOP-Em:SO-H','NPOP-Em:SO-L')
    set(gca,'FontSize',14)

    figure
    semilogy(eV_axis,scatSi_hole(10,:,1).*Gmh,...
        'r','LineWidth',2)
    hold on
    semilogy(eV_axis,scatSi_hole(10,:,2).*Gml,...
        'b--','LineWidth',2)
    semilogy(eV_axis,scatSi_hole(10,:,3).*Gmso,...
        'k','LineWidth',2)
    grid on
    set(gca,'FontSize',14)
    xlabel('Energy (eV)')
    ylabel('Scattering Rate (1/s)')
    title('Total Hole Scattering Rates for Si')
end
end
```

5. MESH GENERATION FUNCTION: GENERATE_FEM_MESH_TSV_PAIR.M

```
function [CM,NM,dt]=...
    generate_fem_mesh_tsv_pair(sGeom,num_angles,sC,sM,T,pPoly)

%Unpack Geometry Struct
sub_doping=sGeom.sub_doping;
xmax=sGeom.xmax;
ymax=sGeom.ymax;
xSig=sGeom.xSig;
ySig=sGeom.ySig;
rSig=sGeom.rSig;
rI=sGeom.rI;
xGND=sGeom.xGND;
yGND=sGeom.yGND;
rGND=sGeom.rGND;

%Unpack Constants/Material Struct
bk=sC.bk;
q=sC.q;
eps_stat=sM.eps_stat;
eml=sM.eml;

%Divide Circular structures into num_angles segments
angle=linspace(0,2*pi,num_angles);

%Drop last point because it is same as start pt (0 = 2*pi)
xSig_pts(length(angle)-1)=0;
```

```
ySig_pts(length(angle)-1)=0;
xSigI_pts(length(angle)-1)=0;
ySigI_pts(length(angle)-1)=0;

for i=1:length(angle)-1
    %Create Points Along Signal Conductor
    xSig_pts(i)=xSig+rSig*cos(angle(i));
    ySig_pts(i)=ySig+rSig*sin(angle(i));

    %Create Points Along Insulator Around Signal Conductor
    xSigI_pts(i)=xSig+rI*cos(angle(i));
    ySigI_pts(i)=ySig+rI*sin(angle(i));
end

%Create Points Along GND Conductor
xGND_pts(length(angle)-1)=0;
yGND_pts(length(angle)-1)=0;
xGndI_pts(length(angle)-1)=0;
yGndI_pts(length(angle)-1)=0;
for i=1:length(angle)-1
    xGND_pts(i)=xGND+rGND*cos(angle(i));
    yGND_pts(i)=yGND+rGND*sin(angle(i));

    xGndI_pts(i)=xGND+rI*cos(angle(i));
    yGndI_pts(i)=yGND+rI*sin(angle(i));
end

%Points for Bounding Box
%Bottom Left -> Counter-Clock Wise
```

```
xBox=[0,xmax,xmax,0];

yBox=[0,0,ymax,ymax];


%Find total number of points(vertices) from above calculations

numVertices=length(xBox)+length(xSig_pts)+...

    length(xSigI_pts)+length(xGND_pts)+length(xGndI_pts);


%Open File for Writing

fileObj=fopen('tsv_pair.poly','w');


%Line 1 of Poly File:-----------------------------------------

%Col 1: Total Number of Fixed/Forced Nodes from Geometry

%Col 2: Dimension...must be 2

%Col 3: Flag for if Attributes Will be Used (0/1)

%Col 4: Flag for if Boundary Markers Will be Used (0/1)

fprintf(fileObj,'%i 2 0 1\n',numVertices);


%Lines in Node Section of Poly File:---------------------------

%Col 1: Unique Node # for Fixed Node (Starting at 1)

%Col 2: x-coord of node

%Col 3: y-coord of node

%Col 4: Boundary Marker


node_count=1;

%Write a line in poly file for each node in bounding box

%Bounding Box Nodes Get Boundary Marker = 1

for i=1:length(xBox)

    fprintf(fileObj,'%i %d %d %i\n',node_count,...

        xBox(i),yBox(i),1);
    node_count=node_count+1;
end

%Write a line in poly file for each node on the sig conductor
%Sig Nodes Get Boundary Marker = 2
for i=1:length(xSig_pts)
    fprintf(fileObj,'%i %d %d %i\n',node_count,xSig_pts(i),...
        ySig_pts(i),2);
    node_count=node_count+1;
end

%Write a line in poly file for each node on the GND conductor
%Gnd Nodes Get Boundary Marker = 3
for i=1:length(xGND_pts)
    fprintf(fileObj,'%i %d %d %i\n',node_count,...
        xGND_pts(i),yGND_pts(i),3);
    node_count=node_count+1;
end

%Write a line in poly file for each node on the sig insulator
%Each Insulator/Material Interface Gets Boundary Marker = 0
for i=1:length(xSigI_pts)
    fprintf(fileObj,'%i %d %d %i\n',node_count,...
        xSigI_pts(i),ySigI_pts(i),0);
    node_count=node_count+1;
end

%Write a line in poly file for each node on the gnd insulator
%Each Insulator/Material Interface Gets Boundary Marker = 0
```

```
for i=1:length(xGndI_pts)
    fprintf(fileObj,'%i %d %d %i\n',node_count,...
        xGndI_pts(i),yGndI_pts(i),0);
    node_count=node_count+1;
end

%First Line in Segment Section of Poly File:-------------------
%Col 1: Total Number of Segments
%Col 2: Flag for if Boundary Markers Will be Used (0/1)
fprintf(fileObj,'%i 1\n',numVertices);

%Lines in Segment Section of Poly File:------------------------
%Col 1: Unique # for Each Segment (Starting at 1)
%Col 2: Node # of Start Pt
%Col 3: Node # of End Pt
%Col 4: Boundary Marker

%For Bounding Box
%Boundary condition same as described above
seg_start=1;
seg_count=1;
for i=1:length(xBox)
    if i ~= length(xBox)
        fprintf(fileObj,'%i %i %i %i\n',seg_count,...
            seg_count,seg_count+1,1);
    %Last node of last segment is same as the start node of
    %first segment
    else
        fprintf(fileObj,'%i %i %i %i\n',seg_count,...
            seg_count,seg_start,1);
    end
    seg_count=seg_count+1;
end

%For Signal Conductors
%Boundary condition same as described above
seg_start=seg_count;
for i=1:length(xSig_pts)
    if i~=length(xSig_pts)
        fprintf(fileObj,'%i %i %i %i\n',seg_count,...
            seg_count,seg_count+1,2);
    %Last node of last segment is same as the start node of
    %first segment
    else
        fprintf(fileObj,'%i %i %i %i\n',seg_count,...
            seg_count,seg_start,2);
    end
    seg_count=seg_count+1;
end

%For GND Conductors
%Boundary condition same as described above
seg_start=seg_count;
for i=1:length(xGND_pts)
    if i~=length(xGND_pts)
        fprintf(fileObj,'%i %i %i %i\n',seg_count,...
            seg_count,seg_count+1,3);
    %Last node of last segment is same as the start node of
    %first segment
    else
        fprintf(fileObj,'%i %i %i %i\n',seg_count,...
            seg_count,seg_start,3);
    end
    seg_count=seg_count+1;
end

%For Sig Insulators
%Boundary condition same as described above
seg_start=seg_count;
for i=1:length(xSigI_pts)
    if i~=length(xSigI_pts(1,:))
        fprintf(fileObj,'%i %i %i %i\n',seg_count,...
            seg_count,seg_count+1,0);
    %Last node of last segment is same as the start node of
    %first segment
    else
        fprintf(fileObj,'%i %i %i %i\n',seg_count,...
            seg_count,seg_start,0);
    end
    seg_count=seg_count+1;
end

%For Gnd Insulators
%Boundary condition same as described above
seg_start=seg_count;
for i=1:length(xGndI_pts)
    if i~=length(xGndI_pts(1,:))
```

```
        fprintf(fileObj,'%i %i %i %i\n',seg_count,...
            seg_count,seg_count+1,0);
    %Last node of last segment is same as the start node of
    %first segment
    else
        fprintf(fileObj,'%i %i %i %i\n',seg_count,...
            seg_count,seg_start,0);
    end
    seg_count=seg_count+1;
end

%First Line in Hole Section of Poly File:----------------------
%Col 1: # of Holes
fprintf(fileObj,'%i\n',2);%Same number of holes as conductors

%Lines in Hole Section of Poly File:---------------------------
%Col 1: Hole #
%Col 2: x-coord of hole
%Col 3: y-coord of hole

%Use center point of conductors to specify holes
fprintf(fileObj,'%i %d %d\n',1,xSig,ySig);
fprintf(fileObj,'%i %d %d\n',2,xGND,yGND);

%Close file
fclose(fileObj);

%Mesh Constraints
wp=sqrt((q*q*sub_doping)/(eps_stat*eml));
```

```
dt=0.1*(1/wp);
lmax=1e6*dt; %vs_max=1e6;
lambda_d=sqrt((eps_stat*bk*T)/(q*q*sub_doping));
tri_edge=0.5*(lmax+lambda_d);

%Create string to use with system() to call triangle.exe
sys_str=sprintf('triangle -penq -a%0.20f tsv_pair &',...
    0.5*tri_edge*tri_edge);

%Call triangle.exe to create mesh from poly file
system(sys_str);

%Create plot of vertices and segements in
%poly file, if requested
if pPoly==1
    figure
    plot(xSig_pts,ySig_pts,'b+-')
    hold on
    plot(xGND_pts,yGND_pts,'b+-')
    plot(xSigI_pts,ySigI_pts,'r^-')
    plot(xGndI_pts,yGndI_pts,'r^-')
    plot([xBox 0],[yBox 0],'ks-')

    title('Vertices and Segements for .poly File')
    set(gca,'FontSize',14)
    xlim([-0.1*xmax xmax+0.1*xmax])
    ylim([-0.1*ymax ymax+0.1*ymax])

    drawmesh tsv_pair.1
    title('Generated FEM Mesh')
    set(gca,'FontSize',14)
end

%-----Read in .ele and .node file generated by triangle.exe----
%-------------Create CM and NM from these files----------------

%-------------------Read in .ele file-------------------------
fid=fopen('tsv_pair.1.ele','r');
%get first line (header telling total number of triangles)
L1=fgetl(fid);

%Characters up to first space specify total number of triangles
count=1;
c='c';
while c ~= ' '
    c=L1(count);
    count=count+1;
end
count=count-2;

%There is also a footer in the file, so only want to read
%the number of lines corresponding to the total number
%of triangles
lines=str2num(L1(1:count));
Mcell=textscan(fid,'%f %f %f %f',lines); %read into cell
fclose(fid);

%Convert cell contents into proper data type
```

```
ele=cell2mat(Mcell(1));
node_i=cell2mat(Mcell(2));
node_j=cell2mat(Mcell(3));
node_k=cell2mat(Mcell(4));

%Create connectivity matrix from converted cell structure
CM=[node_i node_j node_k];

%-------------------Read in .node file--------------------------
fid=fopen('tsv_pair.1.node','r');
%get first line (header telling total number of nodes)
L1=fgetl(fid);

%Characters up to first space specify total number of nodes
count=1;
c='c';
while c~=' '
    c=L1(count);
    count=count+1;
end
count=count-2;

%There is also a footer in the file, so only want to read
%the number of lines corresponding to the total number
%of nodes
lines=str2num(L1(1:count));
Mcell=textscan(fid,'%f %f %f %f',lines);
fclose(fid);
```

```
%Convert cell contents into proper data type
node=cell2mat(Mcell(1));
xcoord=cell2mat(Mcell(2));
ycoord=cell2mat(Mcell(3));
boundary=cell2mat(Mcell(4));

%Create node matrix from converted cell structure
NM=[xcoord ycoord boundary];
```

6. FEM MATRIX GENERATION FUNCTION: GET_TSV_PAIR_FEM_MATRICES_SPARSE.M

```
function [CM,NM,sFEM]=...
    get_tsv_pair_fem_matrices_sparse(CM,NM,sGeom,sM,sC)

%Unpack Geom Struct
xSig=sGeom.xSig;
ySig=sGeom.ySig;
rI=sGeom.rI;
xGND=sGeom.xGND;
yGND=sGeom.yGND;
vSig=sGeom.vSig;
vGnd=sGeom.vGnd;

%Unpack Material Struct
eps_si=sM.eps_si*sC.eps_o;
eps_sio2=sM.eps_sio2*sC.eps_o;

%Get total number of nodes and triangles in FEM Mesh
```

```
tot_nodes=length(NM(:,1));
tot_tri=length(CM(:,1));

%Get x,y coordinates of center of triangles
tri_center(tot_tri,2)=0;
for i=1:tot_tri
    node1=CM(i,1);
    node2=CM(i,2);
    node3=CM(i,3);

    xi=NM(node1,1);
    xj=NM(node2,1);
    xk=NM(node3,1);
    tri_center(i,1)=(xi+xj+xk)/3;

    yi=NM(node1,2);
    yj=NM(node2,2);
    yk=NM(node3,2);
    tri_center(i,2)=(yi+yj+yk)/3;
end

%Find which triangles are in insulator regions------------------
%Find distance to center of sig TSV
test_dist_sig=sqrt(((NM(:,1)-xSig).^2)+((NM(:,2)-ySig).^2));
%Find distance to center of gnd TSV
test_dist_gnd=sqrt(((NM(:,1)-xGND).^2)+((NM(:,2)-yGND).^2));

test_dist_log_sig=round(test_dist_sig,8)<=rI;
test_dist_log_gnd=round(test_dist_gnd,8)<=rI;
```

```
test_dist_logical=test_dist_log_sig | test_dist_log_gnd;

%Find which triangles are in insulator
in_insul_logical(tot_tri,1)=0;
eps_e(tot_tri,1)=0;
for i=1:tot_tri
    node1=CM(i,1);
    node2=CM(i,2);
    node3=CM(i,3);

    %triangles in insulator
    in_insul_logical(i)=test_dist_logical(node1) & ...
        test_dist_logical(node2) & test_dist_logical(node3);

    %Assign dielectric to each triangle
    if in_insul_logical(i)==1
        eps_e(i)=eps_sio2;
    else
        eps_e(i)=eps_si;
    end
end
in_insul=in_insul_logical.*(1:tot_tri).';
in_insul=in_insul(in_insul~=0);

%---------------------------------------------------------------
%Kij=(eps/4*A_e)*(beta_i*beta_j+gamma_i*gamma_j)
%A_e=0.5*(alpha1_e+alpha2_e+alpha3_e)
%alphai_e=xj*yk-xk*yj;
%betai_e=yj-yk;
```

```
%gammai_e=-(xj-xk);
%|a|= 1  |alpha1 alpha2 alpha3|
%|b| --- |beta1  beta2  beta3 |
%|c| 2A_e|gamma1 gamma2 gamma3|

%------------------For Alphas--------------------------------
%alpha1 = x2*y3-x3*y2

%x2 -> CM(:,2)=Node 2 indices
%NM(node index,1) = x-coord of node
x_a1_node2=NM(CM(:,2),1);

%y3 -> CM(:,3)=Node 3 indices
%NM(node index,2) = y-coord of node
y_a1_node3=NM(CM(:,3),2);

%x3 -> CM(:,3)=Node 3 indices
%NM(node index,1) = x-coord of node
x_a1_node3=NM(CM(:,3),1);

%y2 -> CM(:,2)=Node 2 indices
%NM(node index,2) = y-coord of node
y_a1_node2=NM(CM(:,2),2);

alpha1=x_a1_node2.*y_a1_node3-x_a1_node3.*y_a1_node2;

%alpha2 = x3*y1-x1*y3

%x3 -> CM(:,3)=Node 3 indices
```

```
%NM(node index,1) = x-coord of node
x_a2_node3=NM(CM(:,3),1);

%y1 -> CM(:,1)=Node 1 indices
%NM(node index,2) = y-coord of node
y_a2_node1=NM(CM(:,1),2);

%x1 -> CM(:,1)=Node 1 indices
%NM(node index,1) = x-coord of node
x_a2_node1=NM(CM(:,1),1);

%y3 -> CM(:,3)=Node 3 indices
%NM(node index,2) = y-coord of node
y_a2_node3=NM(CM(:,3),2);

alpha2=x_a2_node3.*y_a2_node1-x_a2_node1.*y_a2_node3;

%alpha3 = x1*y2-x2*y1

%x1 -> CM(:,1)=Node 1 indices
%NM(node index,1) = x-coord of node
x_a3_node1=NM(CM(:,1),1);

%y2 -> CM(:,2)=Node 2 indices
%NM(node index,2) = y-coord of node
y_a3_node2=NM(CM(:,2),2);

%x2 -> CM(:,2)=Node 2 indices
%NM(node index,1) = x-coord of node
```

```
x_a3_node2=NM(CM(:,2),1);


%y1 -> CM(:,1)=Node 1 indices

%NM(node index,2) = y-coord of node

y_a3_node1=NM(CM(:,1),2);


alpha3=x_a3_node1.*y_a3_node2-x_a3_node2.*y_a3_node1;


%----------------------For Betas--------------------------------

%beta1=y2-y3


%y2 -> CM(:,2)=Node 2 indices

%NM(node index,2) = y-coord of node

y_b1_node2=NM(CM(:,2),2);


%y3 -> CM(:,3)=Node 3 indices

%NM(node index,2) = y-coord of node

y_b1_node3=NM(CM(:,3),2);


beta1=y_b1_node2-y_b1_node3;


%beta2=y3-y1


%y3 -> CM(:,3)=Node 3 indices

%NM(node index,2) = y-coord of node

y_b2_node3=NM(CM(:,3),2);


%y1 -> CM(:,1)=Node 1 indices

%NM(node index,2) = y-coord of node
```

```
y_b2_node1=NM(CM(:,1),2);

beta2=y_b2_node3-y_b2_node1;

%beta3=y1-y2

%y1 -> CM(:,1)=Node 1 indices
%NM(node index,2) = y-coord of node
y_b3_node1=NM(CM(:,1),2);

%y2 -> CM(:,2)=Node 2 indices
%NM(node index,2) = y-coord of node
y_b3_node2=NM(CM(:,2),2);

beta3=y_b3_node1-y_b3_node2;

%-------------------For Gammas---------------------------------
%gamma1=-(x2-x3)

%x2 -> CM(:,2)=Node 2 indices
%NM(node index,1) = x-coord of node
x_g1_node2=NM(CM(:,2),1);

%x3 -> CM(:,3)=Node 3 indices
%NM(node index,1) = x-coord of node
x_g1_node3=NM(CM(:,3),1);

gamma1=-(x_g1_node2-x_g1_node3);
```

```
%gamma2=-(x3-x1)

%x3 -> CM(:,3)=Node 3 indices
%NM(node index,1) = x-coord of node
x_g2_node3=NM(CM(:,3),1);

%x1 -> CM(:,1)=Node 1 indices
%NM(node index,1) = x-coord of node
x_g2_node1=NM(CM(:,1),1);

gamma2=-(x_g2_node3-x_g2_node1);

%gamma3=-(x1-x2)

%x1 -> CM(:,1)=Node 1 indices
%NM(node index,1) = x-coord of node
x_g3_node1=NM(CM(:,1),1);

%x2 -> CM(:,2)=Node 2 indices
%NM(node index,1) = x-coord of node
x_g3_node2=NM(CM(:,2),1);

gamma3=-(x_g3_node1-x_g3_node2);

%----------------------For Area------------------------------
A=0.5*(alpha1+alpha2+alpha3);

%----------------------For Kij-------------------------------
%K is 3x3xnodes
```

```
%Kij=(eps/4*A_e)*(beta_i*beta_j+gamma_i*gamma_j)
%K11=(eps/4A)*(beta1*beta1+gamma1*gamma1)
%K22=(eps/4A)*(beta2*beta2+gamma2*gamma2)
%K33=(eps/4A)*(beta3*beta3+gamma3*gamma3)

%Need set of triplets for creating sparse matrix.
%Col1: Row Entry
%Col2: Col Entry
%Col3:Non-Zero Value @ that row,col
rcv_matrix(tot_tri*9,3)=0;

%Each Node gives 9 matrix entries in K, so create a
%submatrix to fill that will later be added to a portion
%of the triplet matrix
sub_matrix(9,3)=0;

%Counter for which portion of triplet matrix to add to
count=0;
for i=1:tot_tri
    %Get the three nodes of the triangle
    node1_i=CM(i,1);
    node2_i=CM(i,2);
    node3_i=CM(i,3);

    %Calculate the 9 entries to be entered into the K matrix
    K11=(eps_e(i)/(4*A(i)))*(beta1(i)*beta1(i)...
        +gamma1(i)*gamma1(i));
    K12=(eps_e(i)/(4*A(i)))*(beta1(i)*beta2(i)...
        +gamma1(i)*gamma2(i));
    K13=(eps_e(i)/(4*A(i)))*(beta1(i)*beta3(i)...
        +gamma1(i)*gamma3(i));

    K21=(eps_e(i)/(4*A(i)))*(beta2(i)*beta1(i)...
        +gamma2(i)*gamma1(i));
    K22=(eps_e(i)/(4*A(i)))*(beta2(i)*beta2(i)...
        +gamma2(i)*gamma2(i));
    K23=(eps_e(i)/(4*A(i)))*(beta2(i)*beta3(i)...
        +gamma2(i)*gamma3(i));

    K31=(eps_e(i)/(4*A(i)))*(beta3(i)*beta1(i)...
        +gamma3(i)*gamma1(i));
    K32=(eps_e(i)/(4*A(i)))*(beta3(i)*beta2(i)...
        +gamma3(i)*gamma2(i));
    K33=(eps_e(i)/(4*A(i)))*(beta3(i)*beta3(i)...
        +gamma3(i)*gamma3(i));

    %Create submatrix of triplets...
    %row entry, col entry, value entry
    sub_matrix(:,1)=[node1_i node1_i node1_i node2_i node2_i...
        node2_i node3_i node3_i node3_i];
    sub_matrix(:,2)=[node1_i node2_i node3_i node1_i node2_i...
        node3_i node1_i node2_i node3_i];
    sub_matrix(:,3)=[K11 K12 K13 K21 K22 K23 K31 K32 K33];

    %Fill portion of triplett matrix with submatrix
    rcv_matrix(count+1:count+9,:)=sub_matrix;
    count=count+9;
end
```

```
%Create Sparse K Matrix from Totally Filled Triplet Matrix
K=sparse(rcv_matrix(:,1),rcv_matrix(:,2),...
    rcv_matrix(:,3),tot_nodes,tot_nodes);

%----------Sort Nodes Based on Boundary Condition--------------
inner_node_i=find(NM(:,3)==0);      %Inner Nodes have BC = 0
neumann_node_i=find(NM(:,3)==1);    %Neumann Nodes have BC = 1
sig_node_i=find(NM(:,3)==2);
gnd_node_i=find(NM(:,3)==3);

V_D(length(sig_node_i)+length(gnd_node_i))=0;
V_D(1:length(sig_node_i))=vSig;
V_D(length(sig_node_i)+1:end)=vGnd;
V_D=V_D.';

%Create new order from nodes sorted based on boundary condition
new_order=[neumann_node_i' inner_node_i' ...
    sig_node_i' gnd_node_i'];
inner_length=length(inner_node_i);
neumann_length=length(neumann_node_i);
size_II=neumann_length+inner_length;

%Number of non-zero entries in Full K
num_not_zero=nnz(K);

%Accessing sparse matrices is kind of weird so extract
%out triplet values into three vectors
[rowT,colT,valT]=find(K);
```

```
%-------------For Re-Ordering Sparse K Matrix------------------
%Create Another Triplet Matrix
rcv_matrix2(num_not_zero,3)=0;

%Temp container for sorting a single row by columns
temp_row(tot_nodes)=0;

%Counter for which portion of triplett matrix to add to
start=0;
%Full K is nodes by nodes, so to re-organize K,
%need to go through all the nodes
for i=1:tot_nodes
    %Find row that is going to be moved around
    row = new_order(i);

    %Find which rows in sparse K corresond to the above row
    col_i=find(rowT==row);

    %Calculate how many entries that row has
    num_entry=length(col_i);

    %Temp row is as long as full row in full K
    %For the row above, fill in the appropriate values to the
    %corresponding column entries in the row
    temp_row(colT(col_i))=valT(col_i);

    %Shuffle columns based on new ordering
    temp_row=temp_row(new_order);
    %Find indices of non-zero entries in shuffled row
    new_col_i=find(temp_row);

    %Fill triplet matrix with reordered values
    %Reordered row entry
    rcv_matrix2(start+1:start+num_entry,1)=i;
    %Reordered column entries
    rcv_matrix2(start+1:start+num_entry,2)=new_col_i;
    %Corresponding values
    rcv_matrix2(start+1:start+num_entry,3)=temp_row(new_col_i);

    start=start+num_entry;
    temp_row(:)=0; %Clear out temp row
end
%Create Reordered Sparse K Matrix
K_Mod=sparse(rcv_matrix2(:,1),rcv_matrix2(:,2),...
    rcv_matrix2(:,3),tot_nodes,tot_nodes);

%Grab out sparse submatrices from reordered sparse K matrix
%Sparse matrix allows for usual matrix access here
K_II=K_Mod(1:size_II,1:size_II);
K_ID=K_Mod(1:size_II,size_II+1:tot_nodes);
K_DI=K_Mod(size_II+1:tot_nodes,1:size_II);
K_DD=K_Mod(size_II+1:tot_nodes,size_II+1:tot_nodes);

%Pack sFEM Struct
sFEM.tri_center=tri_center;
sFEM.in_insul_logical=in_insul_logical;
sFEM.in_insul=in_insul;
sFEM.eps_e=eps_e;
sFEM.V_D=V_D;
sFEM.K_II=K_II;
sFEM.K_ID=K_ID;
sFEM.K_DI=K_DI;
sFEM.K_DD=K_DD;
sFEM.neumann_node_i=neumann_node_i;
sFEM.inner_node_i=inner_node_i;
sFEM.sig_node_i=sig_node_i;
sFEM.gnd_node_i=gnd_node_i;
sFEM.beta1=beta1;
sFEM.beta2=beta2;
sFEM.beta3=beta3;
sFEM.gamma1=gamma1;
sFEM.gamma2=gamma2;
sFEM.gamma3=gamma3;
sFEM.new_order=new_order;
sFEM.tot_nodes=tot_nodes;
sFEM.tot_tri=tot_tri;
sFEM.tri_area=A;
```

7. PARTICLE INITIALIZATION FUNCTION: INIT_PARTICLES_TSV_PAIR_FEM_NEIGHBORS.M

```
function [particles,valley,part_in_tri,bg_charge,sInit]=...
    init_particles_tsv_pair_fem_neighbors(CM,NM,sFEM,sGeom,...
    sC,sM,Gmh,T,max_particles)
```

```
%Unpack constants struct
bk=sC.bk;
q=sC.q;
h=sC.h;
emR=sC.emR;

%Unpack material constants struct
A=sM.A;
B=sM.B;
C=sM.C;

%Unpack FEM struct
in_insul=sFEM.in_insul;
tri_area=sFEM.tri_area;
tot_tri=sFEM.tot_tri;
tri_center=sFEM.tri_center;

%Unpack geometry struct
sub_doping=sGeom.sub_doping;
particle_multiplier=sGeom.particle_multiplier;

%Assign Background Charge--------------------------------------
bg_charge(tot_tri,1)=0;
bg_charge(:,1)=-1*sub_doping;
bg_charge(in_insul,1)=0;

%Initialize Particles------------------------------------------
cpsp=sub_doping*mean(tri_area)/particle_multiplier;
```

```
npe(tot_tri,1)=0;   %number of particles per triangle
for i=1:tot_tri
    npe(i)=floor(abs(bg_charge(i))*tri_area(i)/cpsp+0.5);
end

if sum(npe) > max_particles
    fprintf('\nToo many particles needed: %d. Try decreasing
       particle/contact multiplier.\n',sum(npe));
    return;
end

%Initialize some matrices/vectors
particles(max_particles,7)=0;
valley(max_particles,1)=0;
valley(:,1)=9;
part_in_tri(max_particles,1)=0;

n=1;
for i=1:tot_tri
    x1=NM(CM(i,1),1);
    x2=NM(CM(i,2),1);
    x3=NM(CM(i,3),1);

    y1=NM(CM(i,1),2);
    y2=NM(CM(i,2),2);
    y3=NM(CM(i,3),2);

    for j=1:npe(i)
        ei=-(bk*T/q)*log(rand())*1.5;
```

```
cos_t=1-2*rand();
sin_t=sqrt(1-cos_t*cos_t);
phi_ang=2*pi*rand();

sin2t=sin_t*sin_t;
cos2t=cos_t*cos_t;
sin4t=sin2t*sin2t;
cos2p=cos(phi_ang)*cos(phi_ang);
sin2p=sin(phi_ang)*sin(phi_ang);
g=sqrt((B/A)^2+((C/A)^2)...
    *(sin2t*cos2t+sin4t*cos2p*sin2p));

%Initialize All Holes in Heavy Band
ki=sqrt((2*emR*ei*q)/(abs(sM.A)*h*h*(1-g)));

kx=ki*sin_t*cos(phi_ang);
ky=ki*sin_t*sin(phi_ang);
kz=ki*cos_t;

particles(n,1)=kx;
particles(n,2)=ky;
particles(n,3)=kz;
particles(n,4)=-log(rand())/Gmh;
particles(n,5)=ei;

rp1=rand();
rp2=rand();

%Use x,y coordinates of triangle vertices to randomly
```

```
        %place particle within triangle
        xp=(1-sqrt(rp1))*x1+(sqrt(rp1)*(1-rp2))*x2+...
            (rp2*sqrt(rp1))*x3;
        yp=(1-sqrt(rp1))*y1+(sqrt(rp1)*(1-rp2))*y2+...
            (rp2*sqrt(rp1))*y3;
        particles(n,6)=xp;
        particles(n,7)=yp;

        valley(n,1)=1;

        %Assign nearest triangle center for each particle
        dist_to_tc=((xp-tri_center(:,1)).^2)+...
            ((yp-tri_center(:,2)).^2);
        [~,min_i]=min(dist_to_tc);
        part_in_tri(n)=min_i;

        n=n+1;
    end
end

%Create a matrix where each row corresponds to a triangle and
%its 19 nearest neighbors
tri_neighbors(tot_tri,20)=0;
dist_to_neighbors(tot_tri,2)=0;
dist_to_neighbors(:,1)=1:tot_tri;
for i=1:tot_tri
    xc=tri_center(i,1);
    yc=tri_center(i,2);
    %Calculate triangle center to all other triangle center
    %Sort results based on ascending order
    %...then pick closest 20 triangles
    dist_to_neighbors(:,2)=((xc-tri_center(:,1)).^2)+...
        ((yc-tri_center(:,2)).^2);
    dist_to_neighbors_sorted=sortrows(dist_to_neighbors,2);
    tri_neighbors(i,:)=dist_to_neighbors_sorted(1:20,1);
end
sInit.cpsp=cpsp;
sInit.tri_neighbors=tri_neighbors;
```

8. CHARGE COMPUTATION FUNCTION: CHARGE_TSV_PAIR_FEM_NEIGHBORS.M

```
function bi_vect=...
    charge_tsv_pair_fem_neighbors(valley,part_in_tri,sFEM,...
    CM,cpsp,bg_charge,q,max_particles)

%Unpack FEM Struct
tot_tri=sFEM.tot_tri;
tri_area=sFEM.tri_area;
tot_nodes=sFEM.tot_nodes;
neumann_node_i=sFEM.neumann_node_i;
inner_node_i=sFEM.inner_node_i;

charge(tot_tri,1)=0;
for i=1:max_particles
    if valley(i) ~=9
        %Use NGP Method with Nearest Triangle Center
        %Nearest triangle is calculated in init/renew
        %functions, which immediately precede this charge
        %function, so nearest triangle should always be
        %valid/up-to-date
        tri_i=part_in_tri(i);
        charge(tri_i)=charge(tri_i)+1;
    end
end
%Compute actual charge
charge=(charge.*cpsp)./tri_area;

%Compute net charge from charge computed above and bg_charge
net_charge=charge+bg_charge;

%Split triangle charge evenly between 3 nodes
net_charge=(1/3)*net_charge.*tri_area;

%Sum each triangle's contribution to each node
node_charge(tot_nodes,1)=0;
for i=1:tot_tri
    node1_i=CM(i,1);
    node2_i=CM(i,2);
    node3_i=CM(i,3);
    rho_node=net_charge(i);
    node_charge(node1_i,1)=node_charge(node1_i,1)+rho_node;
    node_charge(node2_i,1)=node_charge(node2_i,1)+rho_node;
    node_charge(node3_i,1)=node_charge(node3_i,1)+rho_node;
end
```

```
%Shuffle vector to match modified K matrix
new_order=[neumann_node_i.' inner_node_i.'].';


bi_vect=node_charge(new_order);
bi_vect=bi_vect.*q;
```

9. PARTICLE REFLECT FUNCTION: PARTICLE_REFLECT_FROM_INSULATOR.M

```
function [xff,yff,kx_part,ky_part]=...
    particle_reflect_from_insulator(xc,yc,insul_rad,xi,yi,xf,yf)


%Inputs
%xc: x point for center of conductor (tsv)
%yc: y point for center of conductor (tsv)
%xi: x point for initial particle
%    location (outside of insulator radius)
%yi: y point for initial particle location
%   (outside of insulator radius)
%xf: x point for final particle location
%   (within insulator radius)
%yf: y point for final particle location
%   (within insulator radius)


%-----------------------Step 1----------------------------------
%Shift to local coordinates so that xc,yc is at origin
xi_s=xi-xc;
yi_s=yi-yc;
xf_s=xf-xc;
```

```
yf_s=yf-yc;



%-------------------------Step 2--------------------------------
%Find Point Where Path from P_initial to P_final Intersects
%Circular Boundary of Insulator (Tangent Point)

%Step 2a: Find Equation of Line Between P_initial and P_final
dx=xf_s-xi_s;
dy=yf_s-yi_s;

%Step 2b: Solve x2+y2=r2 and y=mx+b...two possible solutions
if dx == 0
    %Special Condition: If points on a vertical line,
    %then x = const instead of y=mx+b...use x point and
    %radius of insul to find y point
    x_tanf=xi_s;

    if yi_s < 0
        y_tanf=-sqrt(insul_rad*insul_rad-x_tanf*x_tanf);
    else
        y_tanf=sqrt(insul_rad*insul_rad-x_tanf*x_tanf);
    end
else
    %Regular Condition: Solve circle and line equation
    %to find intersection
    m=dy/dx;        %Slope of Line
    b=yf_s-m*xf_s;  %Y-Intercept of Line
    x_tan1=(-2*m*b+sqrt(4*m*m*b*b-4*(1+m*m)*...
        (b*b-insul_rad*insul_rad)))/(2*(1+m*m));
    x_tan2=(-2*m*b-sqrt(4*m*m*b*b-4*(1+m*m)*...
        (b*b-insul_rad*insul_rad)))/(2*(1+m*m));

    %Proper x solution should lie between xi_s and xf_s
    %...two conditions to check depending on if xi is
    %to the left of xf or to the right of xf.
    if xi_s < xf_s
        if x_tan1 > xi_s && x_tan1 < xf_s
            x_tanf=x_tan1;
        else
            x_tanf=x_tan2;
        end
    else
        if x_tan1 > xf_s && x_tan1 < xi_s
            x_tanf=x_tan1;
        else
            x_tanf=x_tan2;
        end
    end

    y_tanf=m*x_tanf+b;
end

%Step 3: Find equation for tangent line to circle (insulator)
%*Special Conditions for vertical line from origin
%to tangent point
```

```
%Step 4: Find equation of line from xf,yf to tangent line,
%that is normal(perpindicular) to tangent line (bisector line)

%Step 5: Find intersection point of bisector line and tangent
%line (solve two line equations for x)

%Step 6: Find distance from xf,yf to intersection point of
%bisector line and tangent line (for mirroring point across
%tangent line)

%Step 7: Mirror xf,yf over tangent line

if x_tanf == 0
    %Special Condition: Tangent Line is Horizontal
    %/ Bisector Line is Vertical
    x_intersect=x_tanf;
    y_intersect=y_tanf;

    delt_y=y_intersect-yf_s;

    xff=x_intersect;
    yff=y_intersect+delt_y;
else
    %Slope of line from origin to tangent point
    m_rad=y_tanf/x_tanf;
    if m_rad == 0
        %Special Condition: Tangent Line is Vertical
        %/ Bisector Line is Horizontal
        x_intersect=x_tanf;
        y_intersect=y_tanf;

        delt_x=x_intersect-xf_s;

        xff=x_intersect+delt_x;
        yff=y_intersect;
    else
        %Regular Condition

        %Slope of tangent line
        %(since perpindicular to line above)
        m_tan=-1/m_rad;
        %Y intercept for tangent line
        b_tan=y_tanf-m_tan*x_tanf;

        %Same slope as line from origin to tangent point
        m_bi=m_rad;
        %Y intercept for bisector line
        b_bi=yf_s-m_bi*xf_s;

        %Solve two lines for x
        x_intersect=(b_bi-b_tan)/(m_tan-m_bi);
        %Put x soln back into either of two lines
        y_intersect=m_bi*x_intersect+b_bi;

        delt_x=x_intersect-xf_s;
        delt_y=y_intersect-yf_s;

        xff=x_intersect+delt_x;
        yff=y_intersect+delt_y;

    end
end



%Step 8: Shift back to global coordinates
xff=xff+xc;
yff=yff+yc;
x_tanf=x_tanf+xc;
y_tanf=y_tanf+yc;

%Step 9: Figure some values for determining final kx and ky
x_dist=abs(xff-x_tanf);
y_dist=abs(yff-y_tanf);
hyp=sqrt(x_dist*x_dist+y_dist*y_dist);
kx_part=x_dist/hyp;
ky_part=y_dist/hyp;

if xff < x_tanf
    kx_part=-1*kx_part;
end

if yff < y_tanf
    ky_part=-1*ky_part;
end
```

10. PARTICLE SCATTER FUNCTION: SI_SCAT_EMC_HOLE_3VALLEY.M

```
function [particle,valley,scatType]=...
    si_scat_emc_hole_3valley(particle,valley,scatTable,...
    de,sC,sM,ie_max)

%Unpack non-material related constants
q=sC.q;
h=sC.h;
emR=sC.emR;

%Unpack material related constants
A=sM.A;
B=sM.B;
C=sM.C;
g111=sM.g111;
g100=sM.g100;
hwo=sM.hwo;
Eso=sM.Eso;
emSO=sM.emSO;

%Particle Attributes Before Scattering
kxi=particle(1,1);
kyi=particle(1,2);
kzi=particle(1,3);
ei=particle(1,5);
valleyi=valley;
```

```
%Get energy index for particle to be scattered,
%based on particle energy
ie=floor(ei/de)+1;

if ie ~= floor(ie)
    ie=1;
end

%If particle is beyond what is calculated in the
%scattering table, set particle energy to maximum
%energy in the table
if ie > ie_max
    ie = ie_max;
end

%Random Number to Choose Scattering Type
r1=rand();

%Find column in scat table corresponding to the particles
%energy. Then check which rows of the table are less than
%r1. The chosen scattering type will be one more than the
%number of rows less than r1.

%slr1=scat types less than r1
slr1=scatTable(:,ie,valleyi)<r1;
scatType=sum(slr1)+1;

%For checking allowable scattering angles
a=(h*h*abs(A))/(2*emR);
```

```
fH_max=1/((a*(1-g111))^(3/2));
fL_max=1/((a*(1+g100))^(3/2));

self_scatter=0;

if scatType > 10
    %Self-Scattering
    particle(1,1)=kxi;
    particle(1,2)=kyi;
    particle(1,3)=kzi;
    particle(1,5)=ei;
    valley=valleyi;
else
    %Find out final valley
    if valleyi == 1
        switch scatType
            %Acoustic (H-H)
            case 1
                ef=ei;
                valleyf=1;
                g_sign=-1;
                reject_flag=1;
                fmax=fH_max;
            %Acoustic (H-L)
            case 2
                ef=ei;
                valleyf=2;
                g_sign=1;
                reject_flag=1;
                fmax=fL_max;
            %NPOP-Ab (H-H)
            case 3
                ef=ei+hwo;
                valleyf=1;
                g_sign=-1;
                reject_flag=1;
                fmax=fH_max;
            %Shouldn't Happen - Copy Case 3
            case 4
                ef=ei+hwo;
                valleyf=1;
                g_sign=-1;
                reject_flag=1;
                fmax=fH_max;
            %NPOP-Ab (H-L)
            case 5
                ef=ei+hwo;
                valleyf=2;
                g_sign=1;
                reject_flag=1;
                fmax=fL_max;
            %NPOP-Ab (H-SO)
            case 6
                ef=ei+hwo-Eso;
                valleyf=3;
                reject_flag=0;
            %NPOP-Em (H-H)
            case 7
```

```
        ef=ei-hwo;
        if ef > 0
            valleyf=1;
            g_sign=-1;
            reject_flag=1;
            fmax=fH_max;
        else
            self_scatter=1;
        end
    %Shouldn't Happen - Copy Case 7
    case 8
        ef=ei-hwo;
        if ef > 0
            valleyf=1;
            g_sign=-1;
            reject_flag=1;
            fmax=fH_max;
        else
            self_scatter=1;
        end
    %NPOP-Em (H-L)
    case 9
        ef=ei-hwo;
        if ef > 0
            valleyf=2;
            g_sign=1;
            reject_flag=1;
            fmax=fL_max;
        else
```

```
                    self_scatter=1;
                end
            %NPOP−Em (H−SO)
            case 10
                ef=ei-hwo-Eso;
                if ef > 0
                    valleyf=3;
                    reject_flag=0;
                else
                    self_scatter=1;
                end
        end
    elseif valleyi == 2
        switch scatType
            %Acoustic (L−L)
            case 1
                ef=ei;
                valleyf=2;
                g_sign=1;
                reject_flag=1;
                fmax=fL_max;
            %Acoustic (L−H)
            case 2
                ef=ei;
                valleyf=1;
                g_sign=-1;
                reject_flag=1;
                fmax=fH_max;
            %NPOP−Ab (L−L)
            case 3
                ef=ei+hwo;
                valleyf=2;
                g_sign=1;
                reject_flag=1;
                fmax=fL_max;
            %NPOP-Ab (L-H)
            case 4
                ef=ei+hwo;
                valleyf=1;
                g_sign=-1;
                reject_flag=1;
                fmax=fH_max;
            %Shouldn't Happen - Copy Case 4
            case 5
                ef=ei+hwo;
                valleyf=1;
                g_sign=-1;
                reject_flag=1;
                fmax=fH_max;
            %NPOP-Ab (L-SO)
            case 6
                ef=ei+hwo-Eso;
                valleyf=3;
                reject_flag=0;
            %NPOP-Em (L-L)
            case 7
                ef=ei-hwo;
                if ef > 0
            valleyf=2;
            g_sign=1;
            reject_flag=1;
            fmax=fL_max;
        else
            self_scatter=1;
        end
    %NPOP-Em (L-H)
    case 8
        ef=ei-hwo;
        if ef > 0
            valleyf=1;
            g_sign=-1;
            reject_flag=1;
            fmax=fH_max;
        else
            self_scatter=1;
        end
    %Shouldn't Happen - Copy Case 8
    case 9
        ef=ei-hwo;
        if ef > 0
            valleyf=1;
            g_sign=-1;
            reject_flag=1;
            fmax=fH_max;
        else
            self_scatter=1;
        end
```

```
            %NPOP-Em (L-SO)
            case 10
                ef=ei-hwo-Eso;
                if ef > 0
                    valleyf=3;
                    reject_flag=0;
                else
                    self_scatter=1;
                end
        end
    else
        switch scatType
            %Acoustic (SO-SO)
            case 1
                ef=ei;
                valleyf=3;
                reject_flag=0;
            %Shouldn't Happen - Copy Case 1
            case 2
                ef=ei;
                valleyf=3;
                reject_flag=0;
            %NPOP-Ab (SO-SO)
            case 3
                ef=ei+hwo;
                valleyf=3;
                reject_flag=0;
            %NPOP-Ab (SO-H)
            case 4
                ef=ei+hwo+Eso;
                valleyf=1;
                g_sign=-1;
                reject_flag=1;
                fmax=fH_max;
            %NPOP-Ab (SO-L)
            case 5
                ef=ei+hwo+Eso;
                valleyf=2;
                g_sign=1;
                reject_flag=1;
                fmax=fL_max;
            %Shouldn't Happen - Copy Case 5
            case 6
                ef=ei+hwo+Eso;
                valleyf=2;
                g_sign=1;
                reject_flag=1;
                fmax=fL_max;
            %NPOP-Em (SO-SO)
            case 7
                ef=ei-hwo;
                if ef > 0
                    valleyf=3;
                    reject_flag=0;
                else
                    self_scatter=1;
                end
            %NPOP-Em (SO-H)
```

```
case 8
    ef=ei-hwo+Eso;
    if ef > 0
        valleyf=1;
        g_sign=-1;
        reject_flag=1;
        fmax=fH_max;
    else
        self_scatter=1;
    end
%NPOP-Em (SO-L)
case 9
    ef=ei-hwo+Eso;
    if ef > 0
        valleyf=2;
        g_sign=1;
        reject_flag=1;
        fmax=fL_max;
    else
        self_scatter=1;
    end
%Shouldn't Happen - Copy Case 9
case 10
    ef=ei-hwo+Eso;
    if ef > 0
        valleyf=2;
        g_sign=1;
        reject_flag=1;
        fmax=fL_max;
```

```
                else
                    self_scatter=1;
                end
        end
    end

    if self_scatter == 1
        particle(1,1)=kxi;
        particle(1,2)=kyi;
        particle(1,3)=kzi;
        particle(1,5)=ei;
        valley=valleyi;
    else
        if reject_flag==1
            %Initial try for valid angles after scattering
            cos_t=1-2*rand();
            sin_t=sqrt(1-cos_t*cos_t);
            phi_ang=2*pi*rand;

            sin2t=sin_t*sin_t;
            cos2t=cos_t*cos_t;
            sin4t=sin2t*sin2t;
            cos2p=cos(phi_ang)*cos(phi_ang);
            sin2p=sin(phi_ang)*sin(phi_ang);
            g=sqrt((B/A)^2+((C/A)^2)*...
                (sin2t*cos2t+sin4t*cos2p*sin2p));

            %Random number for checking if rF <= f/fmax
            rF=rand();
            f=1/((a*(1+g_sign*g))^(3/2));

            while rF > f/fmax
                cos_t=1-2*rand();
                sin_t=sqrt(1-cos_t*cos_t);
                phi_ang=2*pi*rand;

                sin2t=sin_t*sin_t;
                cos2t=cos_t*cos_t;
                sin4t=sin2t*sin2t;
                cos2p=cos(phi_ang)*cos(phi_ang);
                sin2p=sin(phi_ang)*sin(phi_ang);
                g=sqrt((B/A)^2+((C/A)^2)*...
                    (sin2t*cos2t+sin4t*cos2p*sin2p));

                f=1/((a*(1+g_sign*g))^(3/2));
            end

            kf=sqrt((2*emR*ef*q)/(abs(A)*h*h*(1+g_sign*g)));
            kxf=kf*sin_t*cos(phi_ang);
            kyf=kf*sin_t*sin(phi_ang);
            kzf=kf*cos_t;
        else
            cos_t=1-2*rand();
            sin_t=sqrt(1-cos_t*cos_t);
            phi_ang=2*pi*rand();

            kf=sqrt(2*emSO*ef*q)/h;
```

```
            kxf=kf*sin_t*cos(phi_ang);
            kyf=kf*sin_t*sin(phi_ang);
            kzf=kf*cos_t;
        end

        %Update Particle Attributes
        particle(1,1)=kxf;
        particle(1,2)=kyf;
        particle(1,3)=kzf;
        particle(1,5)=ef;
        valley=valleyf;
    end
end
```

VITA

Nicholas Garrett Erickson received his Bachelor of Science in Computer Engineering from the Missouri University of Science and Technology (Missouri S&T), with summa cum laude honors, in May 2012. He joined the Missouri S&T EMC Lab as a graduate research assistant in the summer of 2012. He completed the Master of Science degree in Computer Engineering from Missouri S&T in December 2013. From there, he continued with the EMC Lab as a PhD student. He received his Doctor of Philosophy in Computer Engineering from Missouri S&T in May 2019. His research interests included signal integrity in high-speed digital systems and particle simulations for semiconductor devices. His memberships included the electrical and computer engineering honor society of Eta Kappa Nu, the engineering honor society of Tau Beta Pi, the honor society of Phi Kappa Phi, and IEEE.