2014

Electromagnetic Modeling Of Coupled Transmission Lines For Millimeter-Wave/Terahertz Circuits In 65 Nm Cmos Technology

Monique Danyell Kirkman-Bey

North Carolina Agricultural and Technical State University

Electromagnetic Modeling of Coupled Transmission Lines for Millimeter-Wave/Terahertz Circuits in 65 nm CMOS Technology

Monique Danyell Kirkman-Bey

North Carolina A&T State University

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Department: Electrical and Computer Engineering

Major: Electrical Engineering

Major Professor: Dr. Numan S. Dogan

Greensboro, North Carolina

2014

The Graduate School
North Carolina Agricultural and Technical State University
This is to certify that the Master's Thesis of

Monique Danyell Kirkman-Bey

has met the thesis requirements of
North Carolina Agricultural and Technical State University

Greensboro, North Carolina
2014

Approved by:

Dr. Numan S. Dogan
Major Professor

Dr. Zhijian Xie
Committee Member

Dr. Daniel Limbrick
Committee Member

Dr. John Kelly
Department Chair

Dr. Sanjiv Sarin
Dean, The Graduate School

Biographical Sketch

Monique Danyell Kirkman-Bey is the middle of three children and an only girl. In her younger years, she traveled the world as an Army brat. Around her 10th birthday, she moved with her mom and brother to the Cabrini Green neighborhood of Chicago, Illinois. Her experiences in Cabrini Green motivated her to pursue higher education and a better quality of life. At an early age, Monique learned that she enjoyed disassembling and reassembling toys to understand their inner-workings, and building things with her hands. In middle school, she fell in love with computers and the possibilities they introduced. So, it wasn't a difficult decision to pursue higher education in the field of Electrical and Computer Engineering. She earned her Bachelor of Science in Computer Engineering at Howard University in Washington, DC. During this time, she focused her efforts on Digital and Embedded Systems Design, and Reconfigurable Computing. Her Senior Design project was supported by Northrop Grumman and earned her group second place in the annual Senior Design Competition. After completing her bachelor's degree, she moved to North Carolina to pursue a Master's in Electrical Engineering at North Carolina Agricultural and Technical State University. During her master's, Monique focused her efforts on analog and CMOS design in preparation for future work and to gain more in-depth knowledge of the topics pertinent to a successful Electrical and Computer Engineer. While completing her master's, she worked as a Research Assistant supported by the Army Research Lab and Dr. Numan S. Dogan. She has published two papers with the Institute of Electrical and Electronics Engineers (IEEE), one of which is directly related to her thesis. She presented these works at the IEEE 2014 Southeast Conference in Lexington, KY. Monique currently holds memberships in the Society for Women Engineers (SWE), National Society for Black Engineers (NSBE), and IEEE. She is also a lifetime member of Tau Beta Pi Engineering Honor Society.

Dedication

I would like to dedicate this thesis to my son and my significant other. They are my biggest fans, my support system, and my motivation to continue to work toward my dreams every day. I would also like to dedicate this to my mother and father who have supported my efforts in every endeavor and only asked that I give it my all every time.

Acknowledgements

I would like to first thank my advisory committee for their guidance over the duration of this research and the Army Research Lab for the funding of both my master's and my research. To my primary advisor, Dr. Dogan, thank you for the opportunity to do such groundbreaking research. I never expected to work in analog or CMOS design at the layout level. This experience has truly broadened my knowledge and helped me to become a better-rounded Electrical and Computer Engineer. Dr. Xie, thank you for your assistance throughout my research, especially during the writing phase of the thesis. Your feedback, suggestions, and willingness to read my paper over and over again were invaluable to me. You helped me more than I can express in words. I would also like to thank Dr. Marvin Aidoo, a member of the RF Micro Lab, and the research team in which I work, for all of his assistance. Without you, this work would not have been possible. You were always willing to help when I had trouble with simulations and results, and thoroughly explained concepts in terms that even I could understand.

Several professors in the department of Electrical and Computer Engineering at North Carolina A&T State University also deserve recognition. First, Dr. Cory Graves introduced me to application development and made it possible for me to quickly process the large amounts of data I generated. Without this knowledge, processing and analyzing the 1 million data points would have taken over a month to complete instead of a few hours. Dr. Christopher Doss, thank you for your guidance and mentorship and always having your door open when I needed to talk about the stresses of graduate school and life. I would also like to thank Dr. Clay Gloster, who I met at Howard University. Dr. Gloster introduced me to the Electrical and Computer Engineering concepts I love. He remains a mentor, leads by example every day, and is a large part of the reason I chose to pursue graduate studies at North Carolina A&T State University.

Table of Contents

List of Figures

List of Tables

Abstract

This work assesses the operation of differential transmission lines between 100 GHz and 200 GHz in a 65 nm CMOS process. The focus of this research is the impact of line dimensions and frequency on the odd-mode operation of coupled transmission lines. The research serves to identify a set of lines and line dimensions with high characteristic impedance, low loss, and minimal delay times. The following four coupled transmission line structures are assessed: coplanar waveguides, coplanar waveguides with a metal 1 ground layer, coplanar microstrips, and coplanar microstrips with a metal 1 ground plane. Simulation results from Sonnet and Cadence are presented. With these simulation results, overall trends of the losses, characteristic impedance, and delay of the lines are assessed. The results of this work yield a set of differential transmission lines with at least fifty ohm characteristic impedance, less than 1 decibel of loss, and propagation delay low enough to sustain a 100 - 200 GHz signal.

CHAPTER 1

Introduction

Transmission lines are an integral part of all analog and digital circuits as circuits cannot be completed without them. They serve the important role of carrying electrical signals from one component to another in larger systems. However, in most process design kits, transmission line models are not provided. So, in order to build a robust circuit, the limitations of the transmission lines carrying the signals need to be thoroughly explored and understood. It is imperative to accurately predict its behavior in the desired process and frequency range to ensure a working circuit.

This work serves to identify the limitations of differential transmission lines for use in a 65 nm circuit operating between 100 GHz and 200 GHz. In modeling these lines, attention is paid to the loss, electrical parameters (e.g. resistive, inductive, and capacitive values), characteristic impedance, and delay of the lines.

The thesis is organized as follows. First, an overview of pertinent background information is discussed followed by an explanation of the experiment and a presentation of the results. In Chapter 2, coupled transmission lines and the areas of concern for the transmission line model is explored. Following the overview of relevant background information, Chapter 3 presents a literature review of previous work on modeling differential lines. Chapter 4 presents the experimentation process. The SONNET and Cadence Virtuoso simulation results are discussed in Chapter 5 and the optimal structures and line dimensions for achieving the optimization goals are presented in Chapter 6. Lastly, conclusions and future work are found in Chapter 7.

CHAPTER 2

Differential Transmission Lines

2.1 Overview of Coupled Structures

Differential lines are a subset of coupled structures. They can be coupled in one of two ways, as demonstrated in Figure 1 below.

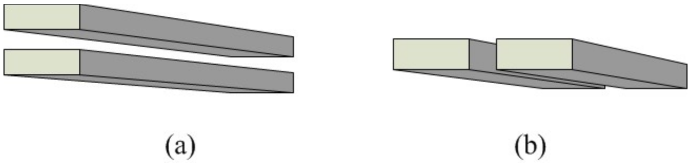


Figure 1. Line Coupling

In Figure 1a, broadside coupling is displayed. Edge coupling can be seen in Figure 1b. Broadside coupled structures are typically implemented on separate metal layers within a metal stack, while edge-coupled structures are implemented on the same metal layer. This work focuses on modeling edge-coupled structures.

The typical line dimensions of coupled lines are as shown in Figure 2. W represents the width of the signal line. H is the height from the substrate. S is the spacing between the lines. L (not shown) represents the length of the line. In this work, H and L are kept constant while W and S are varied.

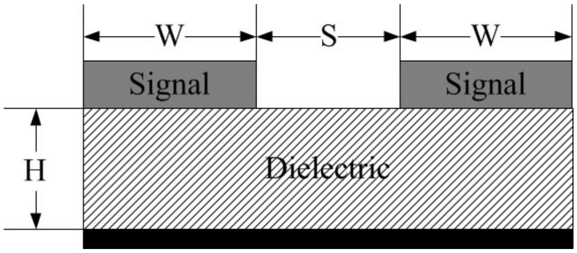


Figure 2. Transmission Line Dimensions

Differential lines may be symmetric or asymmetric. Symmetric lines maintain identical widths for the length of the line while asymmetric lines are two lines of different widths. The spacing between the lines can also be uniform or varied. These lines are referred to as uniformly or nonuniformly-coupled lines, respectively. The transmission lines studied in this work are symmetric uniformly-coupled lines.

2.1.1 Excitation Modes. Coupled lines have two modes of excitation - even and odd. In even mode excitation, both transmission lines are at the same potential and the signal in both lines travel in the same direction. In odd mode excitation, the two lines are at equal, but opposite potentials and the signals in each line travel in the opposite direction of each other. The lines in this work are evaluated for odd mode behavior.

2.2 Evaluated Structures

The following two structures are considered: coplanar microstrip lines, and coplanar waveguides. Each structure is also assessed with the addition of the metal 1 (M1) ground plane for a total of four evaluated structures.

2.2.1 Coplanar Microstrips. Figure 2 in section 2.1 depicts a cross-section of two coplanar microstrip lines. Coplanar microstrip lines are a set of two or more conducting microstrips implemented on the same metal layer running parallel and adjacent to each other [1], as shown in Figure 1b. Typically, the width of the lines, spacing between the lines, and height of the line remain constant for the length of the lines.

2.2.2 Coplanar Waveguides. Coplanar waveguides, first introduced in 1969 [2], consist of one or more conducting strips on a single plane running parallel and adjacent to each other with two ground lines running parallel and adjacent to the signal lines on a single plane. A cross-section of a coupled coplanar waveguide can be seen in Figure 3.

In a coplanar waveguide, the width of the ground line is typically equal to or slightly larger than the width of the signal lines. Previous research has shown that at lower frequencies, it is beneficial to have the ground lines slightly larger than the width of the signal lines. In this work, increasing the width of the ground line presented minimal improvements. So, the width of the ground line and the signal lines are modeled with one variable, W. Additionally, the spacing between the signal lines, S, is modeled with the same variable as the spacing between the signal and ground lines. This is shown in Figure 3 where W represents the width of all lines and S represents the spacing between all lines.

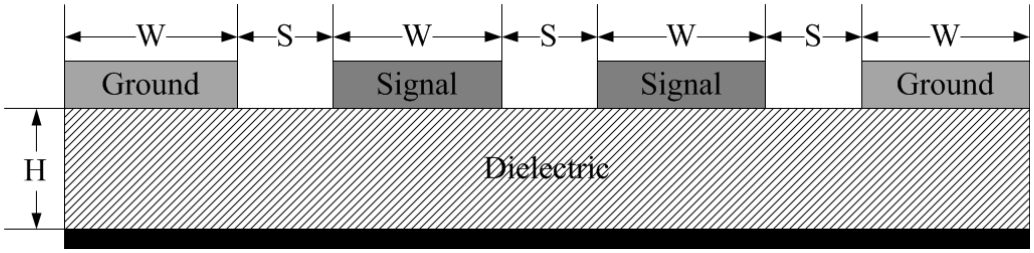


Figure 3. Coplanar Waveguide

2.2.3 M1 Ground Plane. For each of the two structures, the impact of an additional ground plane is also considered. This ground plane is implemented on the lowest metal layer, M1, for each of the structures as shown in Figures 4-5. Then, the impact of this structural modification on the optimization goals is assessed.

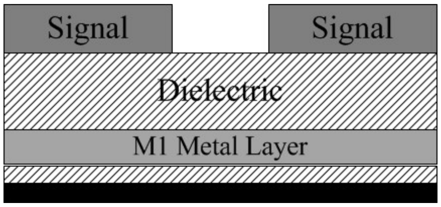


Figure 4. Coplanar Microstrips with M1 Ground Plane

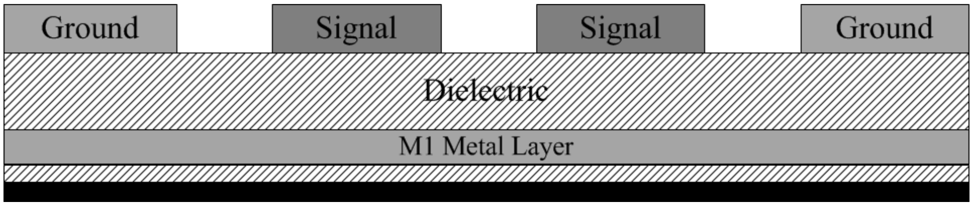


Figure 5. Coplanar Waveguide with M1 Ground Plane

CHAPTER 3

Impact of Structural Variances on Differential Transmission Lines

The objective in modeling the transmission lines is to optimize the loss, delay, and characteristic impedance of the line, and to understand the impact of line dimensions and frequency on these values. Past researchers have presented a multitude of methods for extracting the losses of the line [3]-[8], the characteristic impedance [9]-[10], and propagation through the line [11]-[12] in an effort to predict the behavior of the line prior to fabrication. It has been shown that line width, spacing between the lines, and distance to the ground level all have effects on the parameters this research strives to optimize. In the following sections, an overview of previous work on modeling the influence of the structural variances is reviewed.

3.1 Impact of Line Height

For high-frequency applications, it is best to implement the transmission line on the top metal layer. The additional layers of dielectric, and hence the increased distance between the transmission line and the lossy substrate, minimize the loss when transmitting signals through the lines. In addition to resulting in lower losses, the top metal implementation of the transmission line has direct impacts on the characteristic impedance of the lines. It has been shown that the top metal layer implementation allows the realization of higher transmission-line impedances than lower metal implementations [13]. For these reasons, the transmission lines modeled in this work are implemented on the uppermost metal layer of a 9-metal layer stack.

3.2 Impact of Line Dimensions

The user-defined dimensions of the transmission line, as shown in Figure 2 of section 2.1, also have varying impacts on the delay, characteristic impedance, and losses of the transmission line. First, the width of the line has a direct impact on the line resistance [9]. Wider lines yield lower resistance, but at the cost of lower characteristic impedances [4].

Spacing between the lines impacts the coupling between the lines, the characteristic impedance, and the capacitive loss of the lines. Wide spacing paired with narrow lines allow for high transmission-line impedance, as demonstrated in [14]. However, increased characteristic impedance leads to less coupling between the lines. The spacing also impacts the fringing capacitance, which is double its even-mode value in odd-mode operation. Decreased spacing leads to more capacitance, which negatively impacts the delay of the signal. This work serves to find optimal width and spacing combinations by exploring line widths and line spacing of 5 µm - 10 µm.

3.3 Impact of M1 Ground Plane

The addition of the ground plane on the lowest metal layer, M1, also has implications on the design. Without the M1 ground plane, the transmission line is not only susceptible to its own losses, but also suffers from substrate coupling and is susceptible to the losses of the lossy substrate. Previous researchers were able to combat this without the M1 ground plane by maintaining narrower line widths and narrower spacing [14]. It has been shown that in this configuration, inductance becomes largely dependent on the line separation and capacitance is directly affected by the height of the line. Higher inductance values are achieved with increase spacing. If desired, higher capacitances are realized by implementing the transmission line on lower metal layers to decrease the height, resulting in a slow wave propagation, but lower loss.

Although implementing the M1 ground plane shields the transmission line from the losses of the substrate, previous researchers found that with the M1 ground, series resistance is larger and inductance is lower [14] than they would be without the M1 ground. Researchers in

[14] found that in a coplanar waveguide, the addition of the M1 ground plane causes the return current to flow mainly through the ground plane, as opposed to through the ground lines. This structural difference enables the use of wider lines, allowing lower resistive loss to be realized. However, in area-constrained designs, the use of wider lines consumes more chip space and does not allow for a compact design. Additionally, wider lines lower characteristic impedance [4].

Patterned ground lines were explored in [15] and [16], where the ground plane consisted of parallel coplanar ground lines that ran perpendicular to the conductor lines. The structure effectively reduced eddy current losses by ensuring that ground current only runs perpendicular to the current in the conductor lines. However, it was shown that beyond 10 GHz, these benefits were minimal. So, patterned ground lines are not explored in this work.

CHAPTER 4

Experimental Procedure and Setup

This work explores differential transmission lines for use in a circuit operating in the frequency range of 100 GHz – 200 GHz using a TSMC 65 nm CMOS process. A bottom up approach is applied in modeling the lines. The flowchart of the overall process is presented below in Figure 6. These steps are further explained in the following sections.

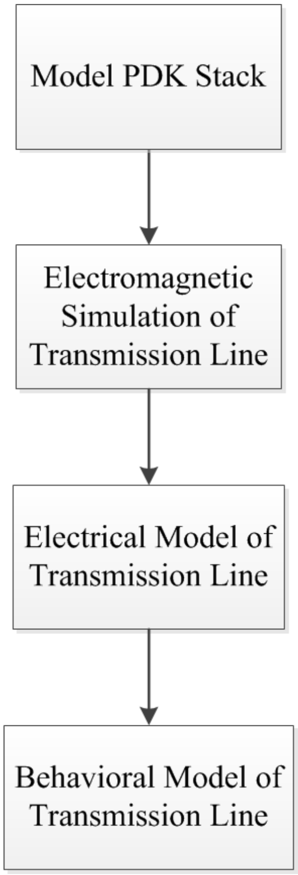

Model PDK Stack
Electromagnetic Simulation of Transmission Line
Electrical Model of Transmission Line
Behavioral Model of Transmission Line

Figure 6. Bottom-Up Experiment Process

4.1 Modeling the Metal Stack

Prior to modeling the transmission lines, an accurate representation of the substrate stack on which the transmission lines will be implemented is needed. In modeling the stack, both Sonnet and Agilent Advanced Design System (ADS) software are used. ADS allows for a substrate representation identical to that of the TSMC process design kit (PDK) due to the ability to change the location of the metal. For example, in ADS it is possible to place a metal in the substrate that intrudes up into the layer above or a metal that intrudes down into the layer below, such as in the PDK. However, the limitations of Sonnet software do not allow for such variations. In order to accurately represent the substrate in Sonnet, layers need to be lumped together, and effective dielectric constants and thicknesses need to be calculated. To ensure that the representation of the substrate in Sonnet is correct, S-parameter values are generated in both ADS and Sonnet and compared to ensure that the results are close. Additionally, the impact of modeling the lines with an expanded or lumped structure is also explored. It is important to note that ADS is 2.5D software, while Sonnet is 3D software. So, variations in the results are expected. However, they are not expected to be large.

4.2 Modeling the Transmission Lines

As previously stated, in a typical PDK, transmission line models are not provided. So, it is important to model the transmission lines to get an accurate representation of their behavior prior to fabrication to avoid costly reruns.

4.2.1 Electromagnetic Simulation. The first step of modeling the lines includes an electromagnetic simulation and the generation of S-parameter data. S-parameters allow for an understanding of the propagation of RF signals through a network. Using S-parameters, important characteristics of a network, e.g. input and output reflection, transmission gain/loss, and isolation between the ports, can be easily extracted.

In this work, S-parameters for a 4-port differential network are generated to understand the response of the lines to differential stimulus. The port numbering of the 4-port network in Sonnet is shown in Figure 7. Although the focus of the work is on differential operation, numbering the ports as follows in Sonnet allows for the generation of both common-mode and differential-mode data.

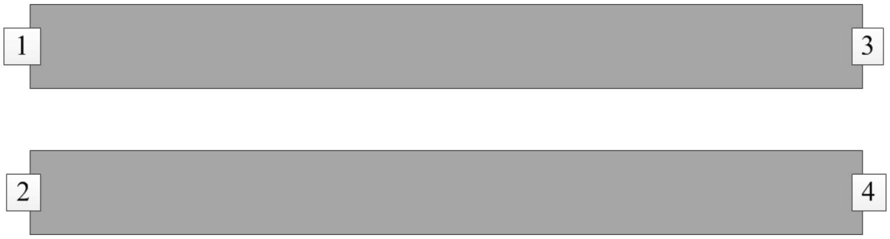


Figure 7. Transmission Line Port Numbers

In modeling the differential lines, the S-parameter values of concern are those that model the input and output reflection, input and output differential insertion loss, and isolation between the ports. These S-parameters are summarized below in equations (1)-(3) where V_{r1}, V_{r2}, and V_{r3} represent the reflected voltages at ports 1, 2, and 3, respectively. Z_1, Z_2, and Z_3 are the impedances at ports 1, 2, and 3, respectively, and V_{i1} is the incident wave at port 1. These quantities are then used to calculate S_{11}, the input refection, S_{21}, the isolation between the ports, and S_{31}, the insertion loss. Since the transmission lines represent a reciprocal network and the lines are symmetric, $S_{11} = S_{22}$. Hence, the input and output reflection are the same. Similarly, $S_{31} = S_{13}$, and $S_{21} = S_{12}$.

$$S_{11} = \frac{V_{r1}}{V_{i1}} \tag{1}$$

$$S_{21} = \sqrt{\frac{Z_1}{Z_2}}\left(\frac{V_{r2}}{V_{i1}}\right) \tag{2}$$

$$S_{31} = \sqrt{\frac{Z_1}{Z_3}}\left(\frac{V_{r3}}{V_{i1}}\right) \tag{3}$$

S-parameters are most often expressed in decibels (dB), as shown in equation (4), where S_{XY} represents a generic S-parameter quantity between port X and port Y. Insertion loss is typically expressed as a positive dB quantity as shown in equation (5). In this work, all S-parameter data is expressed in dB per unit length.

$$S_{XY}\ (\text{dB}) = 20log_{10}(S_{XY}) \tag{4}$$

$$\text{Insertion Loss (dB)} = -20log_{10}(S_{31}) \tag{5}$$

4.2.2 Electrical Model. The results of the electromagnetic simulation are translated into an equivalent electrical model. In this work, the electrical parameters of interest include the resistance (R), inductance (L), and capacitance (C) values, and the characteristic impedance these values yield. The lines are represented by the equivalent electrical model shown below in Figure 8.

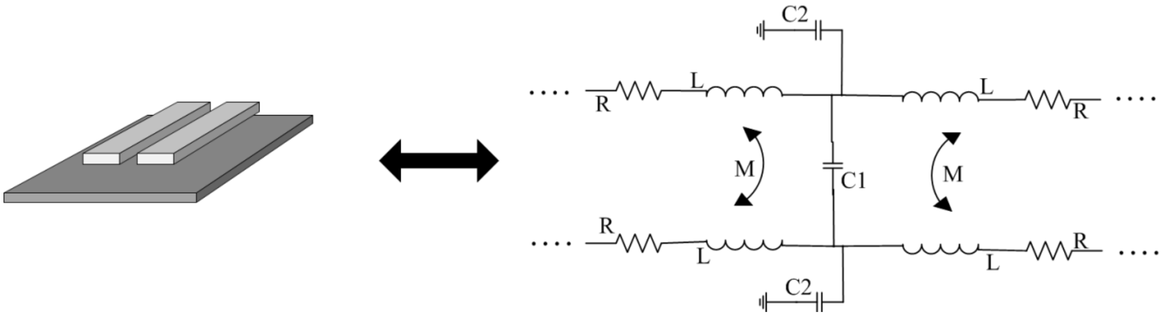


Figure 8. Electrical Model of Differential Transmission Lines

Figure 8 shows a single section of a coupled line. In reality, transmission lines are an infinite number of these sections serially connected. The resistance in the line is represented by R. The series inductance of the line is represented as L, with the mutual inductance between the lines represented by M. The capacitive components in the line are the coupling capacitance between the two signal lines and the coupling capacitance between the lines and the substrate, shown as C1 and C2, respectively.

To estimate the electrical models, the Sonnet N-coupled line output is used. First, the accuracy of the N-coupled line output data is verified in ADS using the following line-fitting method: ten sections of the equivalent circuit model, as shown in Figure 8, is simulated iteratively against the S-parameters generated in Sonnet using optimization and tuning. The iterations are continued until the RLC values of the equivalent circuit yield S-parameters that mimic the ones generated for the transmission line. These RLC values are then used as shown in equations (6) and (7) to calculate the characteristic impedance of the line, *Z₀*, and are compared to the RLC and Z_0 values generated in Sonnet.

$$Z_0 = \sqrt{\frac{2L-KL}{C_2+2C_1}} \tag{6}$$

$$\mathrm{K} = \frac{M}{\sqrt{L^2}} = \frac{M}{L} \tag{7}$$

After verifying the Sonnet N-coupled line data, a parametric sweep of the width and spacing for values of 5 µm – 10 µm is completed for each of the structures in 10 GHz frequency bands from 100 GHz to 200 GHz. Recall that in the coplanar waveguide, the signal and ground widths are kept the same. Also, the separation between all lines is modeled with a single parameter. The height and length of the lines remain constant. The height is dictated by the TSMC PDK. The length is modeled at 100 µm. In varying the width of the lines and separation between them, this work seeks to identify the trends in coupling capacitance, inductance, resistance, and characteristic impedance of the lines as impacted by the width and spacing of the lines and operating frequency.

The parametric sweep generates a total of 288 parameter files for each 10 GHz frequency band - 144 files for the electromagnetic simulation results, and 144 files for the electrical simulation results. Each parameter file includes 200 extraction points with either 3 or 6 data points of interest at each extraction point for the S-parameter and N-coupled line files, respectively. Due to this large amount of data, an application is developed to quickly extract the data from the output files. The application allows a user to import either a single parameter file, or an entire directory of files. The points are then extracted from the parameter file(s) and exported in .csv file format for further analysis in Matlab or excel. Screenshots of the graphical user interface (GUI) and the complete code for this application can be found in the Appendix.

4.2.3 Behavioral Model. Next, the results of the electrical modeling are used to model the behavior of the lines. To understand the behavior of the lines, Cadence Virtuoso is used to measure the propagation delay. In Cadence Virtuoso the mtline part is used and the Sonnet N-coupled Line output file is specified as the mtline RLGC input. Since the lines will be applied to a circuit in which the lines operate in odd mode, the mtline part is excited differentially. A transient simulation is run and the time it takes for the input to reach the output is measured using the Cadence calculator delay function.

CHAPTER 5

Results and Analysis

As previously stated, the goal of the work is to understand how variations in line dimensions and operating frequency impact the electromagnetic, electrical, and behavioral aspects of transmission lines implemented in a 65 nm CMOS process. Below, the results of the simulations are presented.

5.1 Electromagnetic Simulation Results

5.1.1 Input Reflection. S_{11} is used to model the reflection of the input signal. In equation (1) of Section 4.2.1 it is shown that S_{11} is the ratio of the reflected signal to the incident signal. So, smaller S_{11} values are ideal as they indicate less of the signal being reflected at the input port. In the following section, S_{11} is plotted in millidecibels (mdB) per unit length. On the decibel scale, larger negative magnitudes of S_{11} are indicative of less reflection of the signal at the input. First, the input reflection in the coplanar waveguide is assessed. Then, the reflection in the coplanar microstrip lines is assessed and compared to the reflection in the coplanar waveguide. For both structures, the impact of the additional M1 ground plane on input reflection is also discussed.

In the coplanar waveguide, input reflection is heavily impacted by the width of the lines, spacing between the lines, and frequency. For both the coplanar waveguide with the M1 ground plane (CPWM1) and the coplanar waveguide without the M1 ground plane (CPW), input reflection decreases with increases in line width, and increases with increases in frequency. Recall, in the coplanar waveguides the widths of the four lines are modeled with a single parameter. Likewise, the spacing between the four lines is modeled with a single parameter.

In the CPWM1, there is a noticeable change in the dependency on spacing and in the amount of reflection as compared to the CPW. However, the addition of the M1 ground plane is only apparent at line widths of 6 micron or greater. At 5 micron line widths, the value of S_{11} in the CPWM1 is identical to that seen in the 5 micron lines of the CPW. In the electromagnetic results that follow, all trends in the CPWM1 apply to line widths of 6-10 micron (µm).

As stated, the M1 ground plane causes a discernable change in the input reflection. Without the M1 ground, input reflection in the coplanar waveguide is directly related to spacing. This can be seen in Figures 9 and 10 where S_{11} for the CPW becomes less negative with increases in spacing, indicating more reflection of the signal. With the addition of the M1 ground plane, input reflection displays an inverse relationship with the separation between the lines. As separation between the lines increases, input reflection decreases in the CPWM1. It can also be seen that for smaller spacing values, e.g. below 8.5 micron, the CPW displays less reflection than the CPWM1. Also, the CPWM1 shows less of a dependency on spacing than the CPW. The impact of spacing and frequency on the CPW and CPWM1 can be seen in Figures 9 and 10, where S_{11} is plotted against spacing for 10 micron wide lines, at 100 GHz and 200 GHz respectively. The reflection in the CPW is plotted with a solid line while the reflection in the CPWM1 is plotted with a dashed line.

Overall, the CPW boasts S_{11} values between -97.5 and -369 mdB/µm. The CPW yields S_{11} values in a range of -133 to -366 mdB/µm.

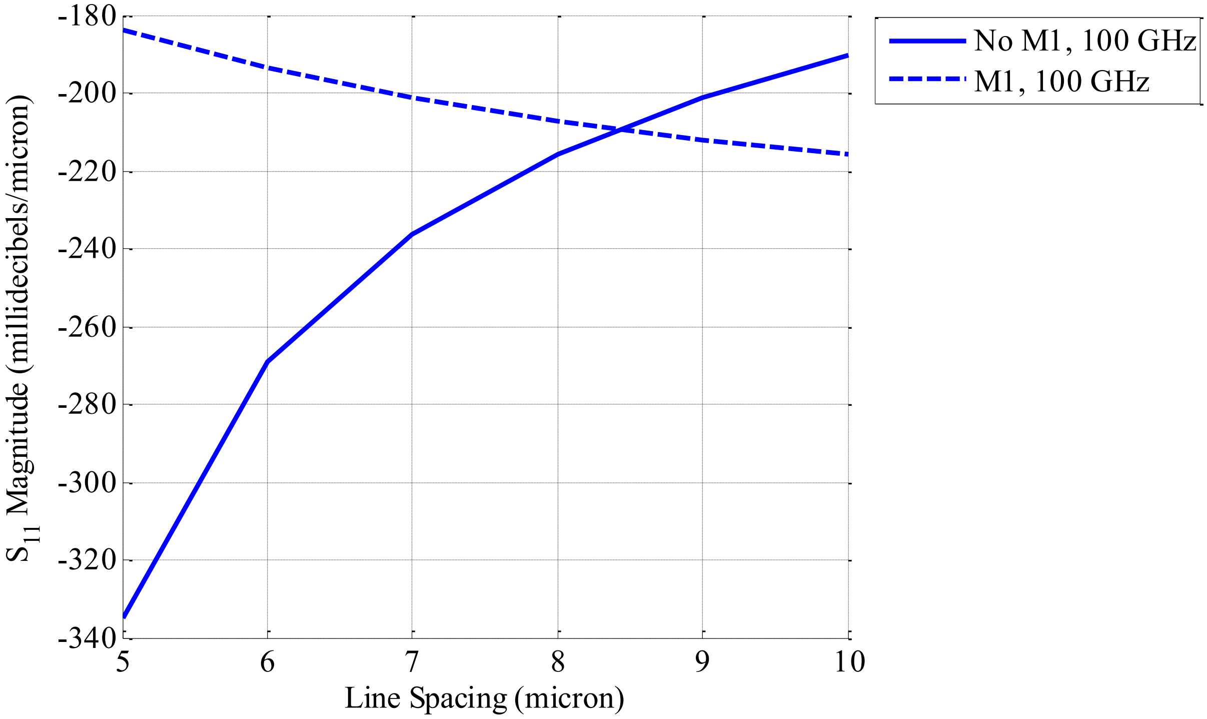


Figure 9. Waveguide: S_{11} Magnitude, 10 µm Lines, 100 GHz

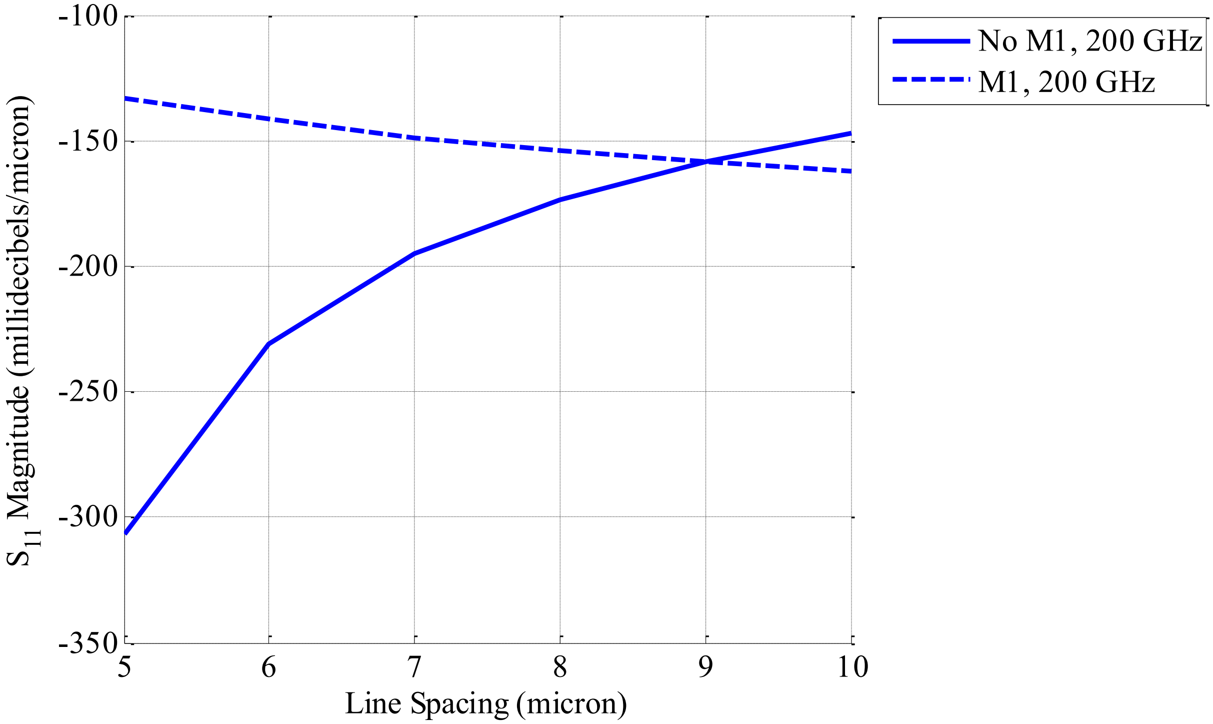


Figure 10. Waveguide: S_{11} Magnitude, 10 µm Lines, 200 GHz

The input reflection increases directly with frequency in both the coplanar microstrips with the M1 ground plane (CPMM1) and the coplanar microstrips without the M1 ground (CPM). However, the dependency on width and spacing varies for the two structures. In the CPM, reflection decreases as width of the signal lines increase, and increases as spacing between the lines increase. Also, wider lines are more heavily impacted by spacing than narrower lines.

In the CPMM1, the input reflection decreases overall when compared to the CPM. There is, however, an interesting impact of width and spacing on the input reflection. The input reflection in the CPMM1 lines of smaller widths does not display the same pattern of dependency on width and spacing as wider lines. In the 5-6 micron wide lines, input reflection decreases as width increases from 5 to 6 micron. However, in the 7-10 micron wide lines, reflection increases by as much as a third as the width of the lines increase. The dependency on spacing is also opposite at wider signal widths than the narrower widths. In Figures 11 and 12, the input reflection's dependency on spacing and frequency for the CPM and CPMM1 can be seen. The CPM is plotted with a solid line and the CPMM1 is plotted with a dashed line for a line width of 10 μm at 100 GHz and 200 GHz respectively.

Compared to the waveguide, the microstrips show more reflection when the M1 ground is removed. However, the addition if the M1 ground causes the microstrips to show a slight improvement in input reflection over the waveguide. In the CPM, the simulated S_{11} values are between -80.3 and -163 mdB/μm. The CPMM1 displays input reflection between -156 and -367 mdB/μm.

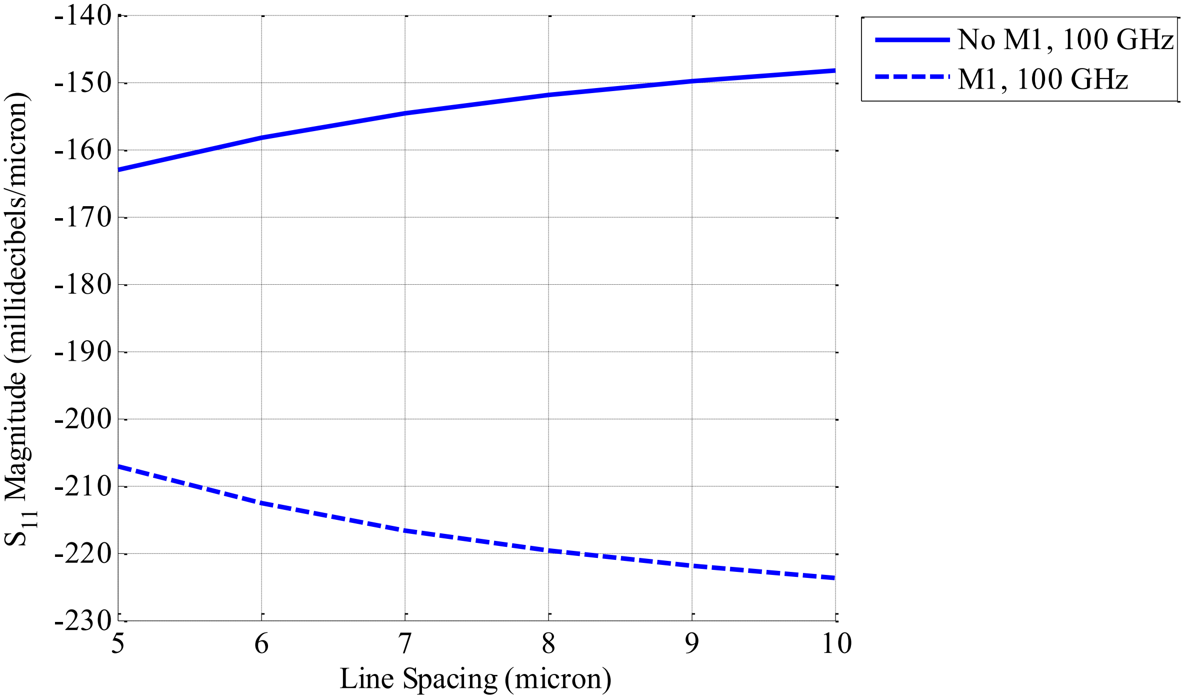


Figure 11. Microstrip: S_{11} Magnitude, 10 µm Lines, 100 GHz

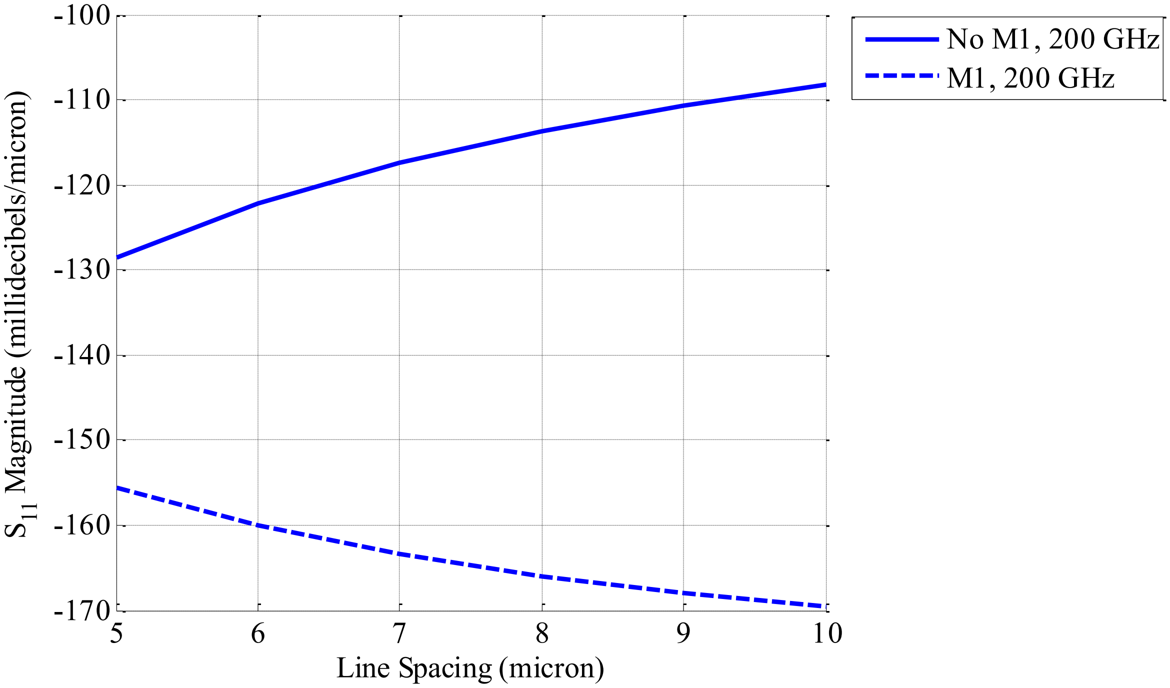


Figure 12. Microstrip: S_{11} Magnitude, 10 µm Lines, 200 GHz

5.1.2 Port Isolation. S_{21} is used to model the isolation between the ports. See Figure 7 of section 4.2.1 for the port numbering. In modeling isolation between the ports, S_{21} is plotted in mdB/µm, and hence, larger negative quantities are ideal. This displays more isolation between the ports and indicates that an input signal on one line has less of an impact on the signal in the second line. Port isolation in the waveguides is discussed first. Then, isolation in the microstrips are discussed and compared to isolation in the waveguides.

In both the coplanar waveguides, isolation between the ports decreases with increases in frequency. This can be seen in Figures 13 and 14 where S_{21} for the CPW and CPWM1 is plotted for 10-micron-wide lines at 100 GHz and 200 GHz respectively. In the CPW, isolation between the ports improves with both increases in spacing and width. However, isolation is more heavily impacted by spacing between the lines than it is by the width of the lines. As spacing increases from 5 to 10 micron for a constant width, S_{21} improves by approximately 15 mdB/µm. However, as width increases from 5 to 10 micron with constant spacing, port isolation only improves by 1.8 mdB/µm. Recall that spacing between the four lines is modeled with a single parameter in the waveguides. The widths of the four lines are also modeled with a single parameter.

In the CPWM1, the addition of the ground plane is again unapparent until line width reaches 6 micron. In the CPWM1 lines of 6-10 micron width, the addition of the M1 ground plane causes a noticeable improvement in isolation between the ports. Similar to the CPW, isolation between the ports improves as both width and spacing increases in the CPWM1. However, isolation in the CPWM1 displays a more heavy dependence on spacing than in the CPW. This can be seen in Figures 13 and 14, where the S_{21} plot against spacing for the CPWM1 has a larger negative slope than the line for the CPW. Overall, S_{21} in the CPWM1 ranges from -154 to -275 mdB/µm while S_{21} in the CPW ranges from -97.4 to -173 mdB/µm.

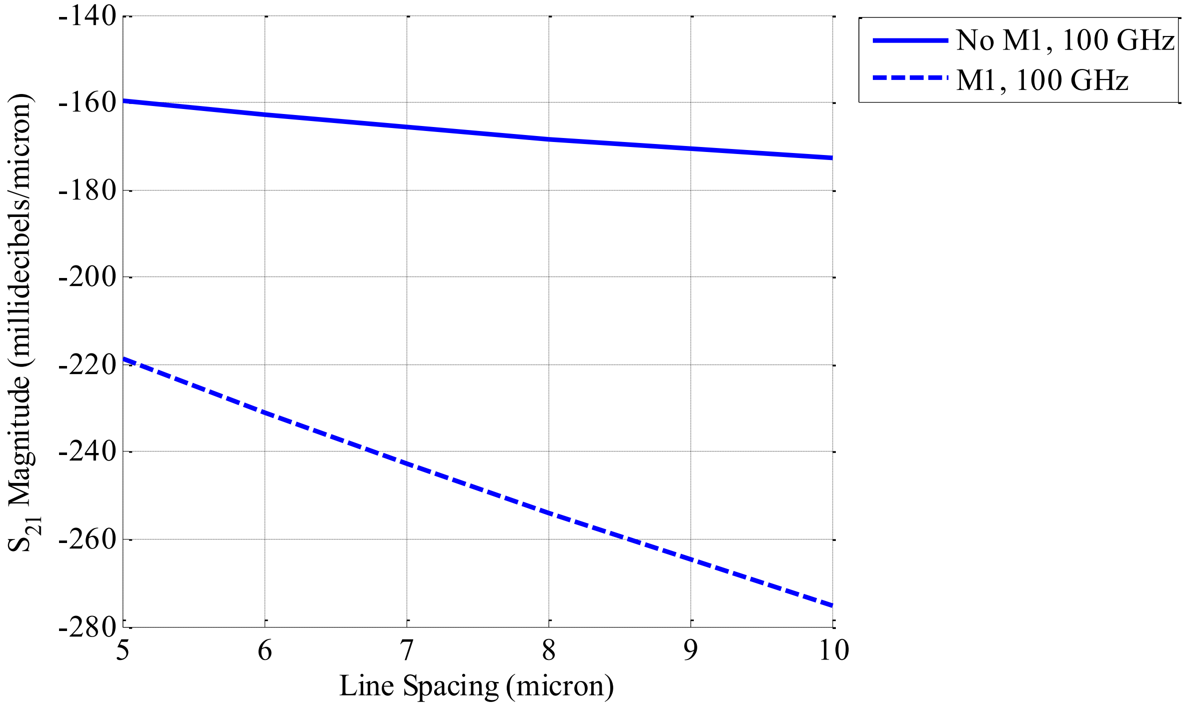


Figure 13. Waveguide: S_{21} Magnitude, 10 µm Lines, 100 GHz

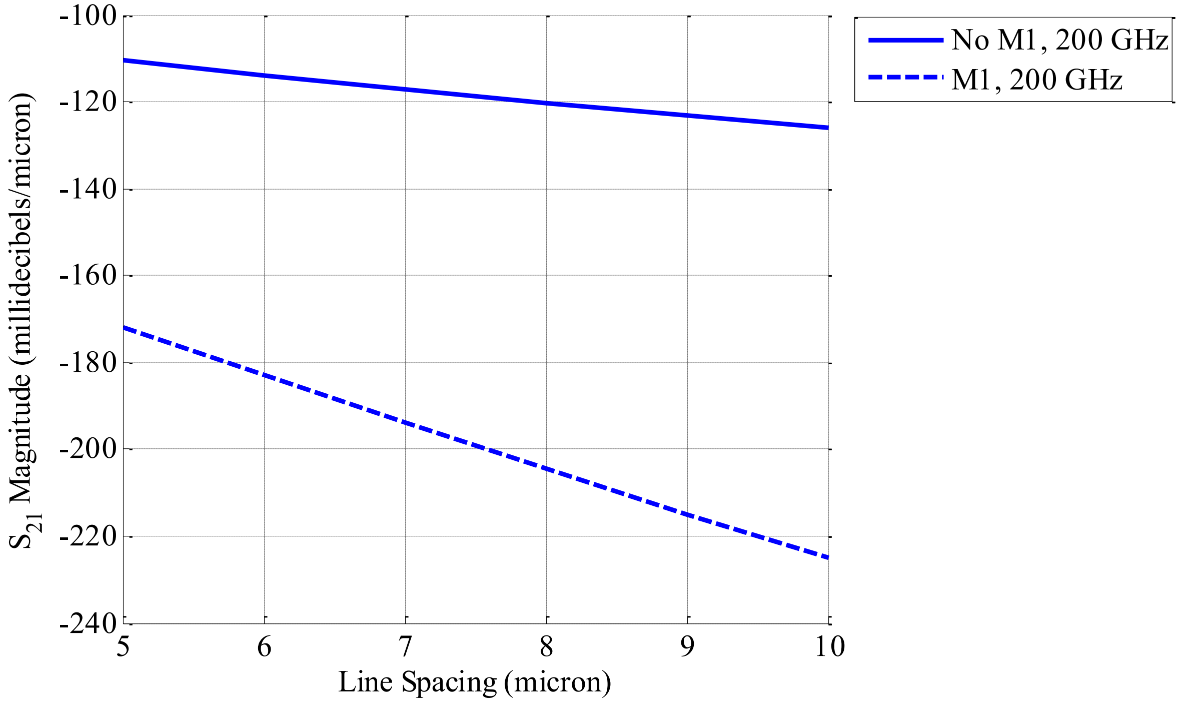


Figure 14. Waveguide: S_{21} Magnitude, 10 µm Lines, 200 GHz

In the CPM, isolation between the ports is minimally impacted by the width of the lines. Isolation improves as spacing increases and worsens as frequency increases. This can be seen in Figures 15 and 16 where 10-micron-wide microstrip lines are plotted against spacing at 100 GHz and 200 GHz respectively for the CPM and CPMM1.

In the CPMM1, the addition of the M1 ground yields an overall improvement in the isolation between the ports. The CPMM1 displays the same pattern of dependency on width, spacing and frequency as the CPM. However, the coplanar microstrips with the M1 ground plane are more heavily impacted by spacing than the coplanar microstrips without the M1 ground. Overall, the CPM and CPMM1 displays better isolation between the ports than the CPW and CPWM1, respectively. In the CPM, simulated values of S_{21} are between –81.3 and –149 mdB/µm. In the CPMM1, isolation between the ports has simulated values between -145 and - 273 mdB/µm.

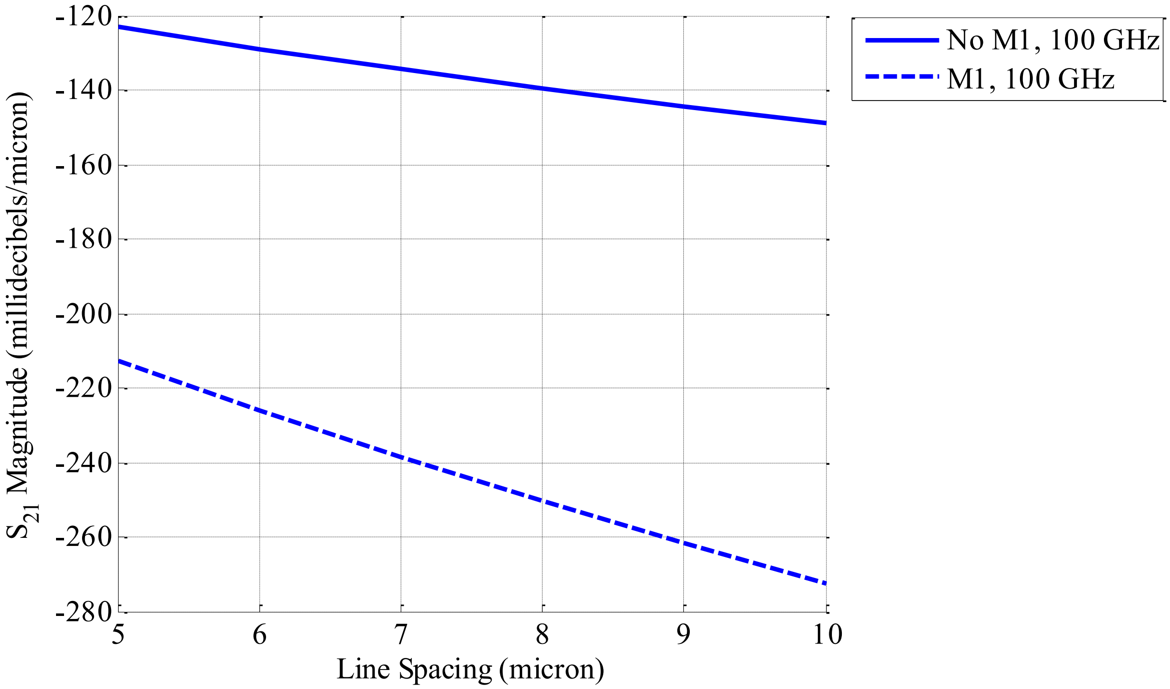


Figure 15. Microstrip: S_{21} Magnitude, 10 µm Lines, 100 GHz

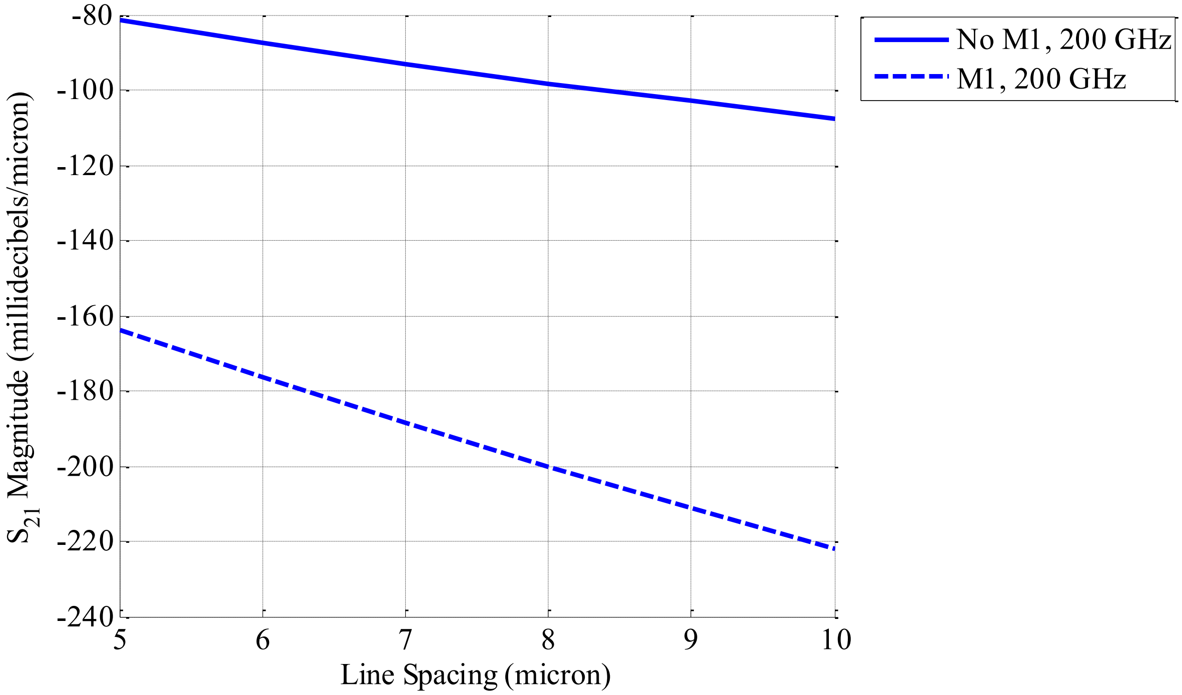


Figure 16. Microstrip: S_{21} Magnitude, 10 µm Lines, 200 GHz

5.1.3 Insertion Loss. S_{31} is used to model the transmission/insertion loss in the lines. Ideally, S_{31} will display values closer to 0, indicating a larger amount of the input signal reaching the output. In the discussion that follows, first the insertion loss in the CPW and CPWM1 are discussed. Then, the insertion loss in the CPM and CPMM1 are discussed and compared to the loss in the CPW and CPWM1.

In both the CPW and CPWM1, insertion loss increases as frequency increases. This can be seen in Figure 17, where the loss in the waveguides is plotted against frequency for 10-micron-wide lines with 5 micron spacing. The solid lines represent the loss in the CPW. The dashed lines represent the loss in the CPWM1.

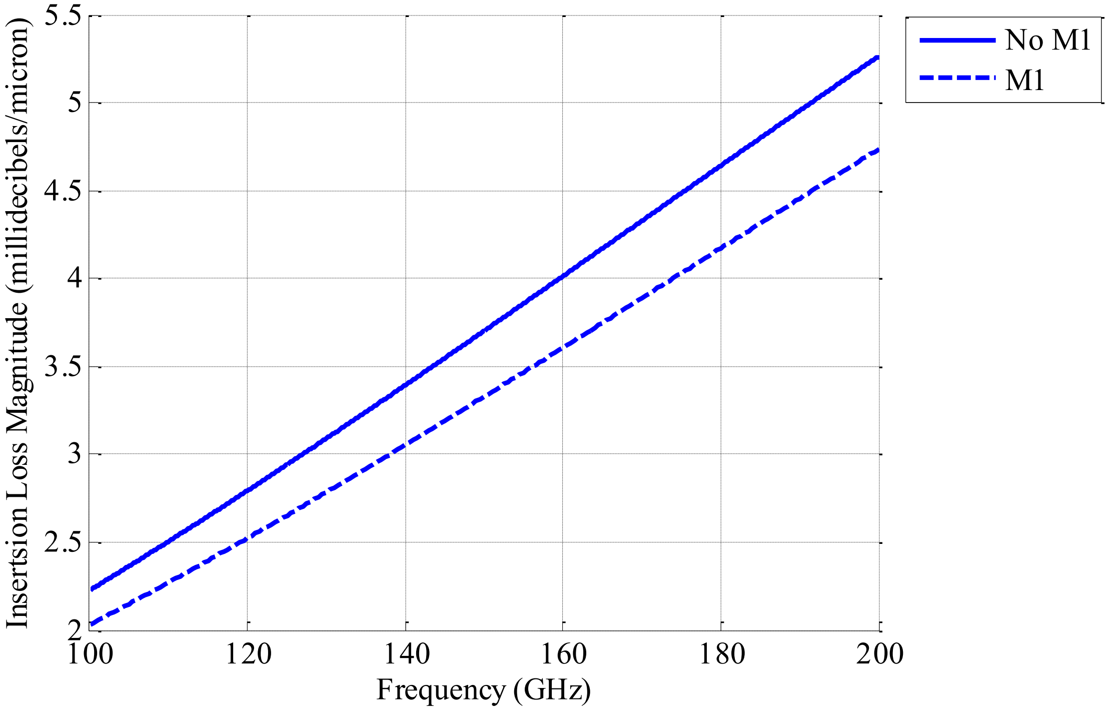


Figure 17. Waveguide: Loss by Frequency, 10 µm Lines, 5 µm Spacing

Width and spacing have an interesting impact on the transmission of the signal in the CPW. Overall, loss decreases as the width of the lines decrease. However, narrower spaced lines are more heavily impacted by width of the lines than wider spaced lines. Recall, the spacing between all lines is the same and the widths of all lines are the same. For line widths below 7.8 micron, insertion loss increases linearly with increases in spacing. However, when line width increases above 7.8 micron, this no longer applies. This is due to the fact that as spacing increases, the loss decreases at a slower rate with increases in line width. This can be seen in Figure 18 where the loss is plotted for line spacing values of 5, 6, and 7 micron at 200 GHz against signal line width. It can be seen that above 7.8 micron line width, the loss in the lines separated by 5 micron is larger than the loss of the lines separated by 6 micron. At 10 micron line width, the loss in the lines with 6 µm spacing approaches the loss in the lines separated by 7 µm.

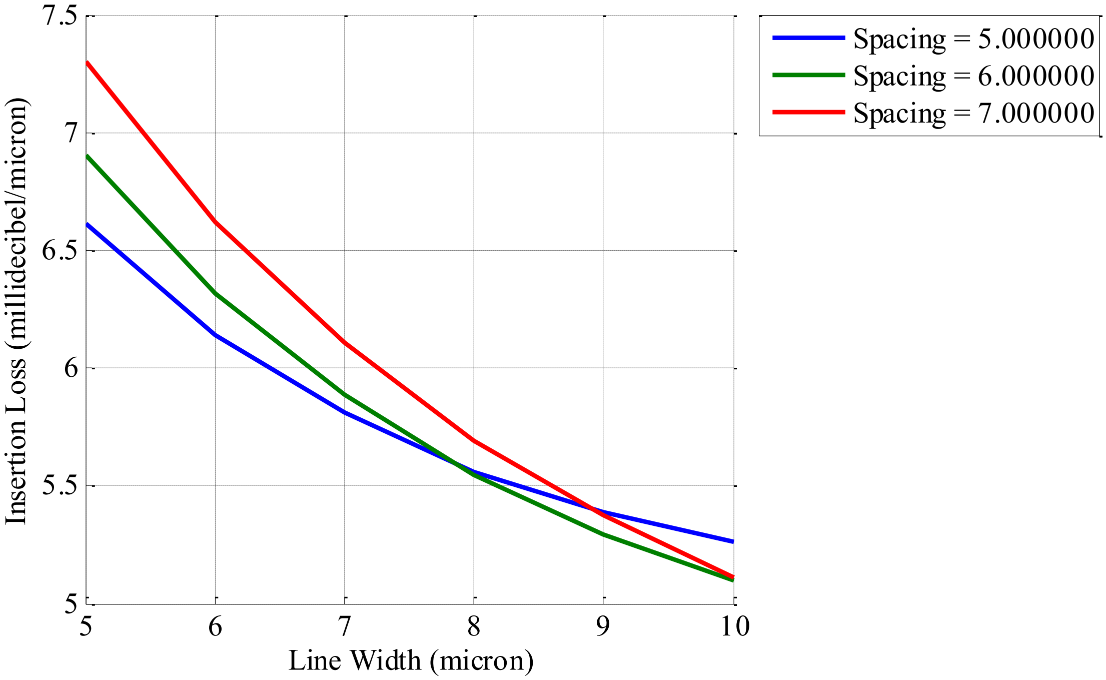


Figure 18. CPW Loss by Line Width for 6, 7, and 8 µm Spacing

In the CPWM1, there is an overall decrease in loss compared to the CPW. The addition of the M1 ground also causes a change in the dependency on spacing and width. In the CPWM1, insertion loss displays an inverse relationship with spacing. As spacing increases from 5 to 10 micron, loss in the lines decreases and more of the signal is transmitted. Also, for line widths of 6-10 micron, the loss decreases as width increases. This can be seen in Figures 19-20 where the loss of the CPW and CPWM1 are plotted together against width and spacing.

Overall, insertion loss for the CPW ranges from 2.2 to 14.4 mdB/µm. The CPWM1 has less insertion loss with simulated values between 1.3 and 4.7 mdB/µm.

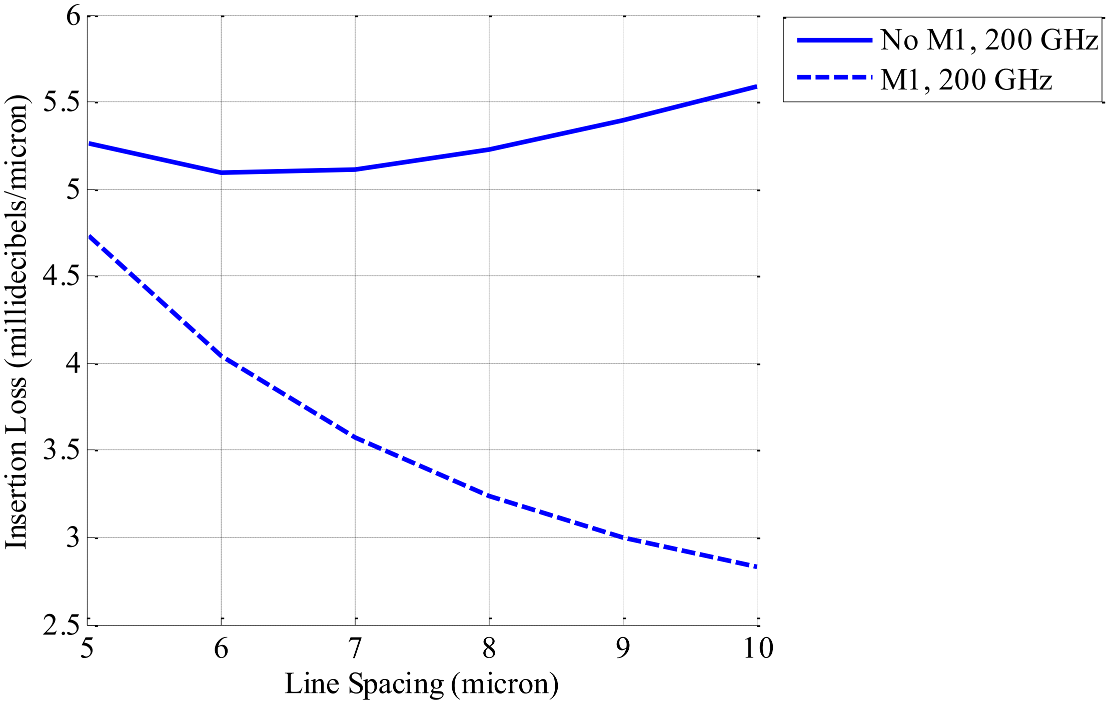


Figure 19. Waveguide: Loss by Spacing, 10 µm Lines, 200 GHz

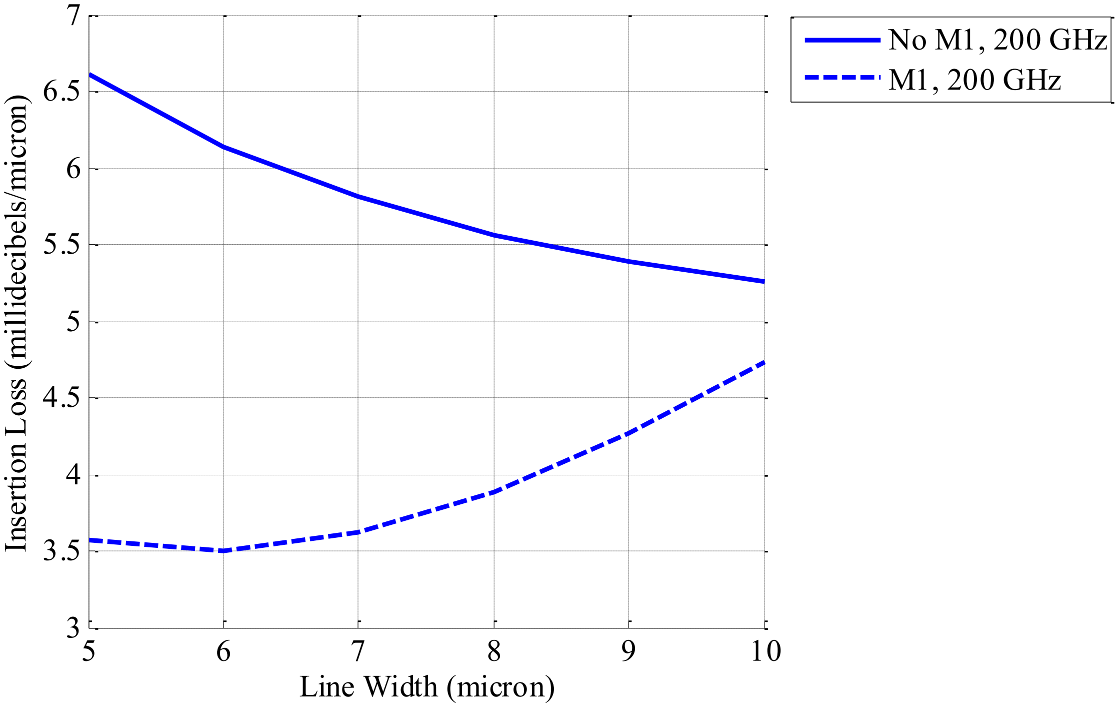


Figure 20. Waveguide: Loss by Line Width, 5 µm Spacing, 200 GHz

In the CPM, loss improves as both width and spacing increases, and worsens as frequency increases. These trends are shown in Figures 21-23 where the loss in the CPM and CPMMI are plotted together against frequency, spacing, and width, respectively.

The CPMM1 yields an improvement in loss over the CPM and is less impacted by changing frequencies than the CPMM1. This can be seen above in Figure 21 where 5-micron-wide lines with 10 micron spacing are plotted against frequency for the CPM and CPMM1. Overall, the insertion loss decreases with increases in spacing for the CPMM1, as shown in Figure 22.

However, the impact of line width on insertion loss varies in the CPMM1. Insertion loss improves slightly with increases in width until the width of the line reaches 7 micron. Further increases in line width cause insertion loss to increase and less of the signal to be transmitted. This pattern is visible in Figure 23.

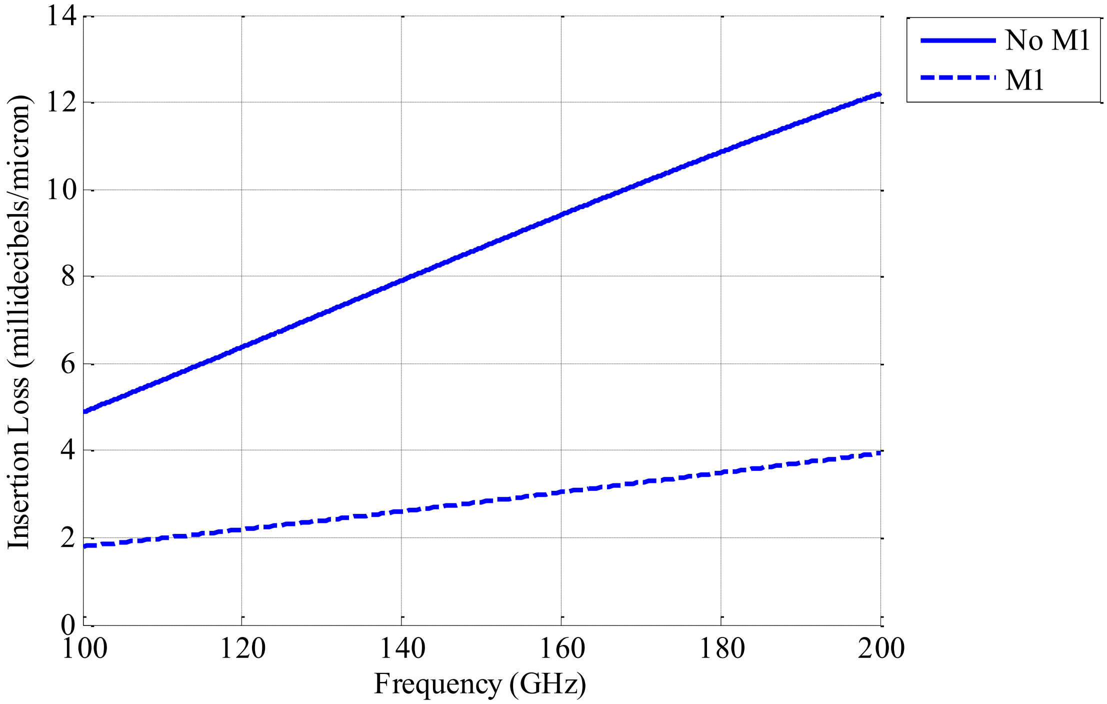


Figure 21. Microstrip: Loss by Frequency, 10 μm Lines, 5 μm Spacing

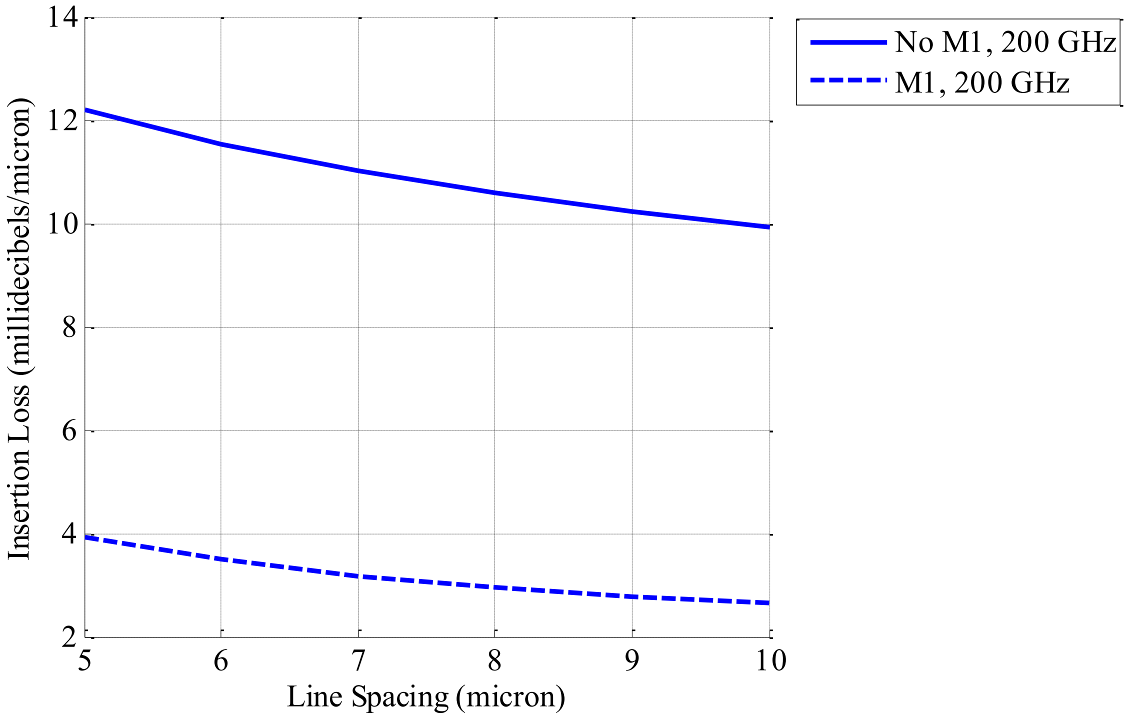


Figure 22. Microstrip: Loss by Spacing, 10 µm Lines, 200 GHz

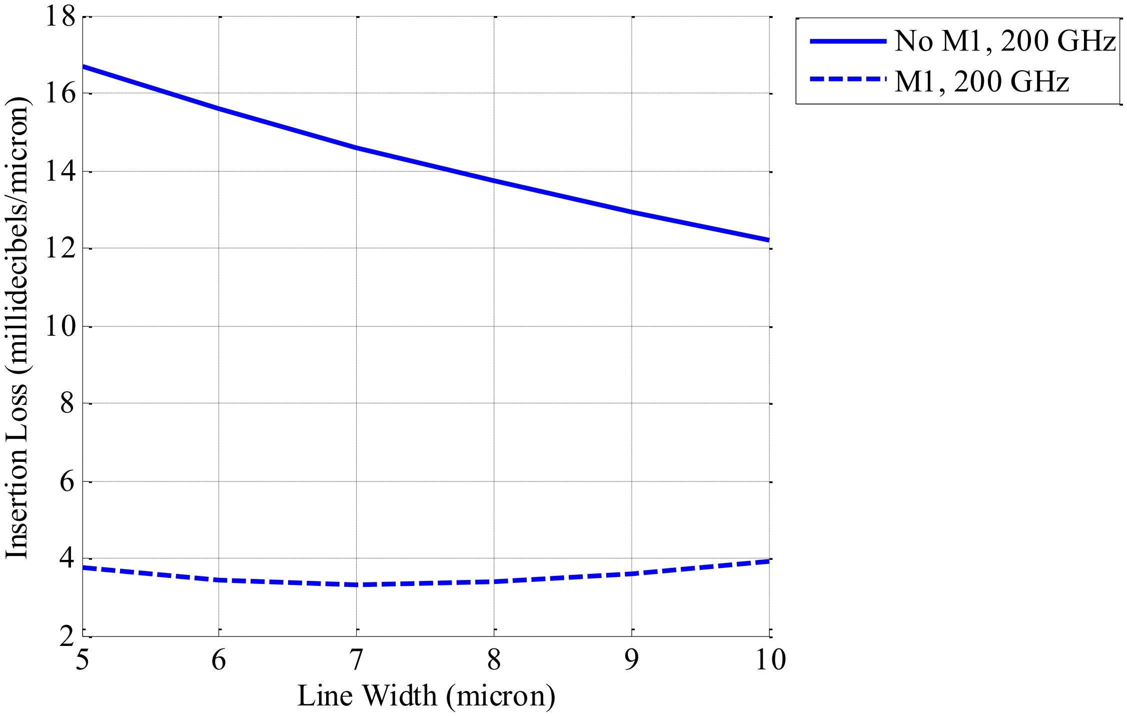


Figure 23. Microstrip: Loss by Line Width, 5 µm Spacing, 200 GHz

Compared to the CPW, the CPM has more insertion loss with a range of 3.98 to 16.7 mdB/µm. The CPMM1 yields the least amount of loss and a slight improvement in insertion loss over the CPWM1 with a range of 1.33 to 3.94 mdB/µm.

5.2 Electrical Model Results

5.2.1 Characteristic Impedance. Figures 24-25 show the simulated values of the characteristic impedance of the CPW and CPWM1 against spacing for a constant line width of 5 µm at 100 GHz and 200 GHz respectively. Recall that the spacing between all lines is equal, and the widths of all lines are equal in the coplanar waveguides.

In Figures 24 and 25, the solid lines represent the simulated characteristic impedance of the CPW while the dashed lines represent the simulated characteristic impedance of the CPWM1. It can be seen that the characteristic impedance remains fairly constant across the frequencies. It decreases by less than 1 ohm as frequency increases from 100 GHz to 200 GHz. The line dimensions, however, impact the characteristic impedance more strongly. In both the CPW and CPWM1, the characteristic impedance is inversely related to the width of the signal and ground lines and directly related to the spacing. The CPW yields characteristic impedances of 34.9-55.8 ohms across all width-spacing combinations. The addition of the M1 ground plane leads to an average drop in the characteristic impedance of 5 ohms and the CPWM1 yields values of 30.0-47.7 ohms across the width and spacing combinations.

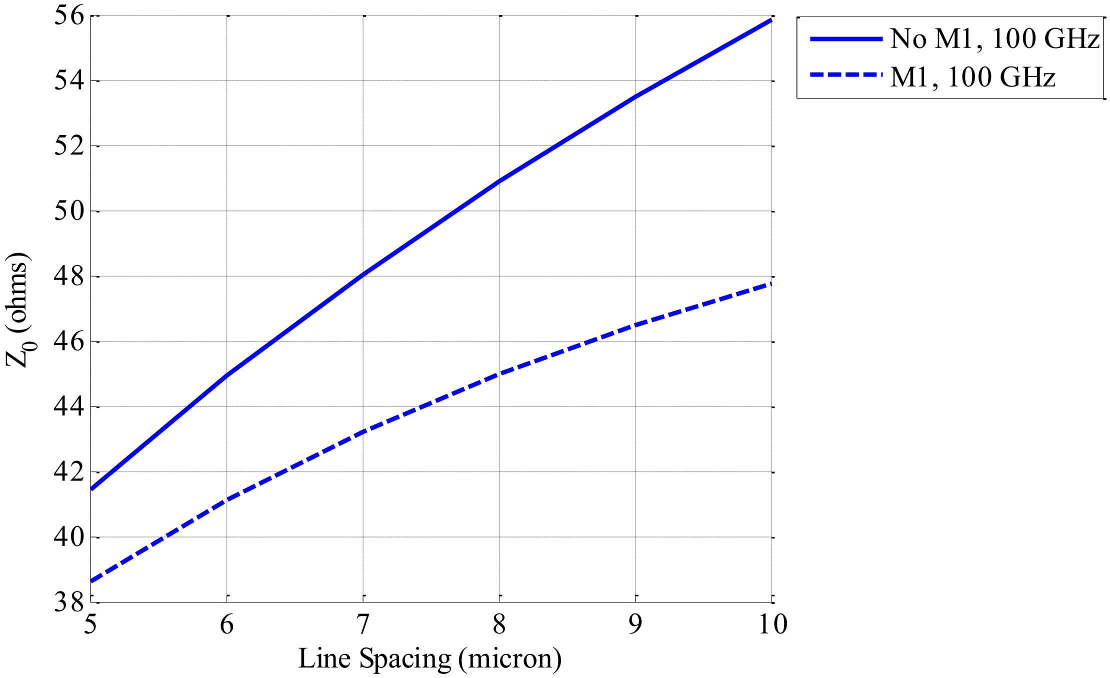


Figure 24. Waveguide: Characteristic Impedance, 5 µm Lines, 100 GHz

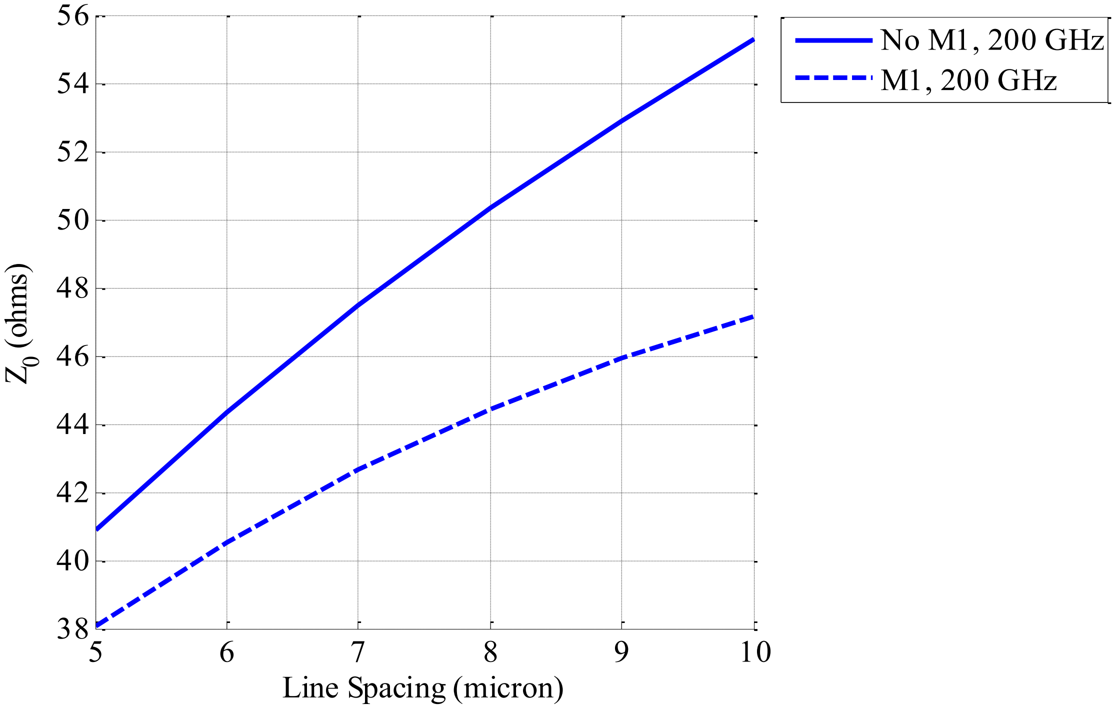


Figure 25. Waveguide: Characteristic Impedance, 5 µm Lines, 200 GHz

In Figures 26 and 27, the characteristic impedance of the CPM and CPMM1 are shown for a line width of 5 µm, at 100 GHz and 200 GHz respectively. The solid lines represent the CPM while the dashed lines represent the CPMM1.

The coplanar microstrips yield higher characteristic impedances than the coplanar waveguides, and again the characteristic impedance remains fairly constant across the frequencies. In both the CPM and CPMM1, the characteristic impedance is impacted by the line dimensions in the same fashion that it is impacted in the coplanar waveguide. It decreases as the lines become wider and increases when the spacing between the lines is increased. A total range of 39.4-59.3 ohms is observed in the microstrip lines without the M1 ground plane. The addition of the M1 ground plane again leads to characteristic impedances approximately 10 ohms lower than that seen on the microstrip lines without the M1 ground plane. In the microstrip lines with the M1 ground plane characteristic impedances between 31.6 and 48.5 ohms are achieved.

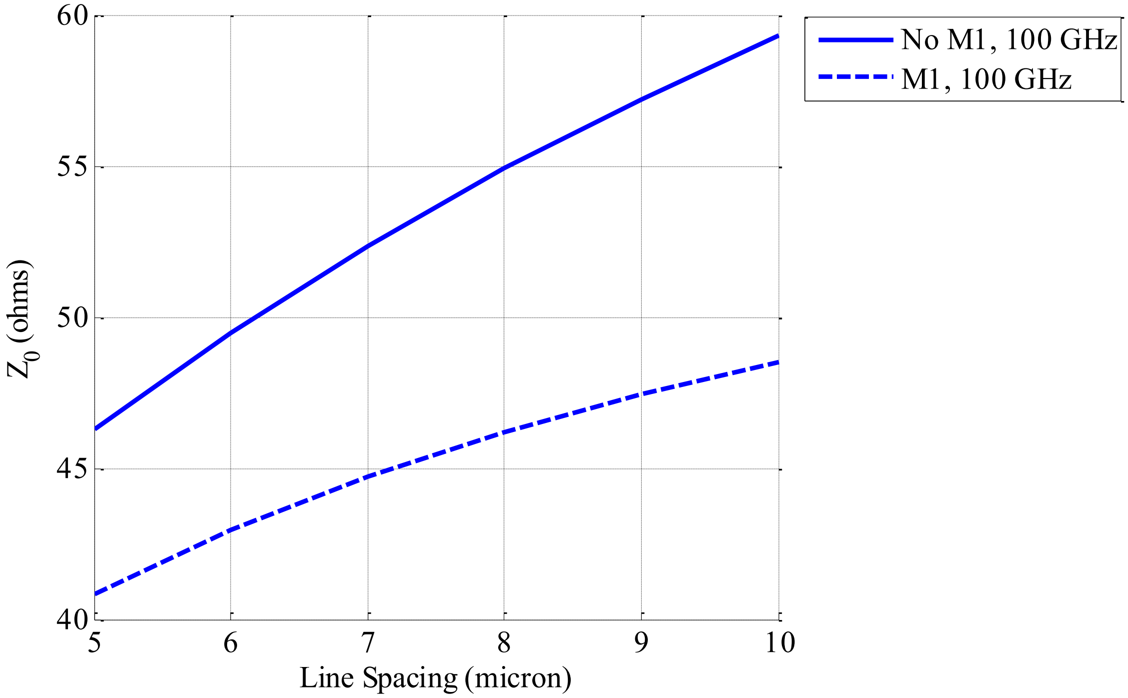


Figure 26. Microstrip: Characteristic Impedance, 5 µm Lines, 100 GHz

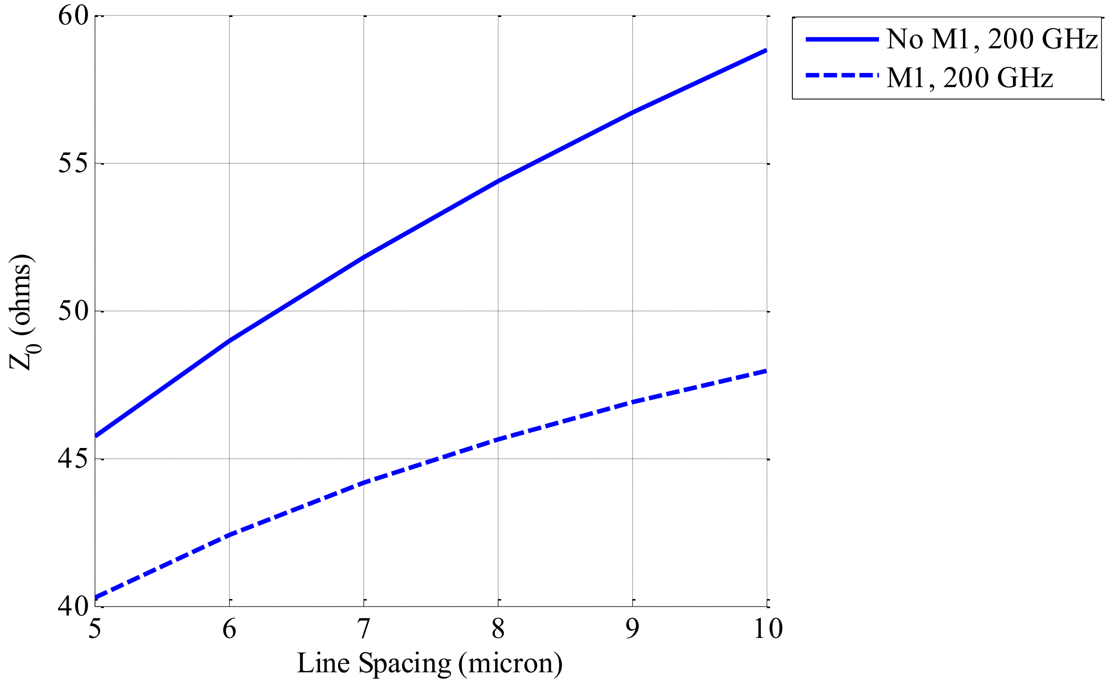


Figure 27. Microstrip: Characteristic Impedance, 5 μm Lines, 200 GHz

5.2.2 Capacitive Coupling between the Lines, C1. The coupling capacitance between the lines is represented by C1 in Figure 8 of section 4.2.2. Figures 28-29 show the simulated values of the coupling capacitance for the CPW and CPWM1 with line widths of 5 μm at 100 GHz and 200 GHz respectively.

By comparing Figures 28 and 29, it can be seen that the capacitance between the lines remains flat across the frequencies. In the coplanar waveguide without the M1 ground plane (CPW) the capacitance between the lines, C1, is impacted by both changes in line width and changes in spacing. C1 decreases as spacing increases and increases as the width of the line increases. Without the M1 ground plane, the coplanar waveguide yields C1 values of 26.6-49.7 attoFarad/micron.

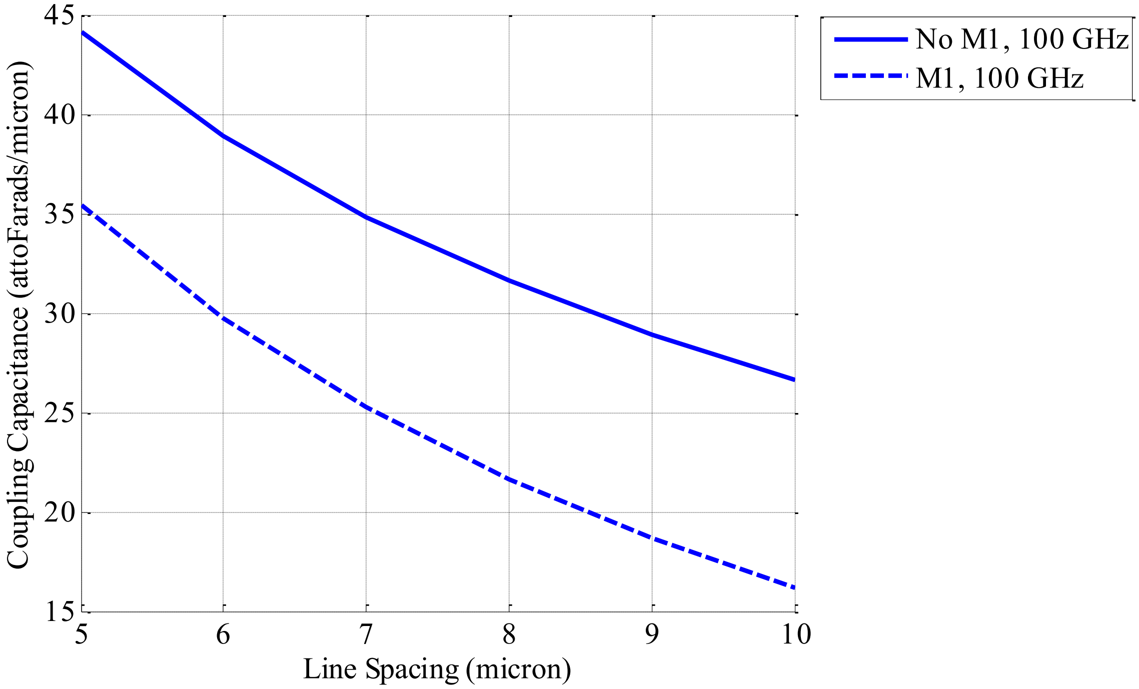


Figure 28. Waveguide: Capacitive Coupling, 5 µm Lines, 100 GHz

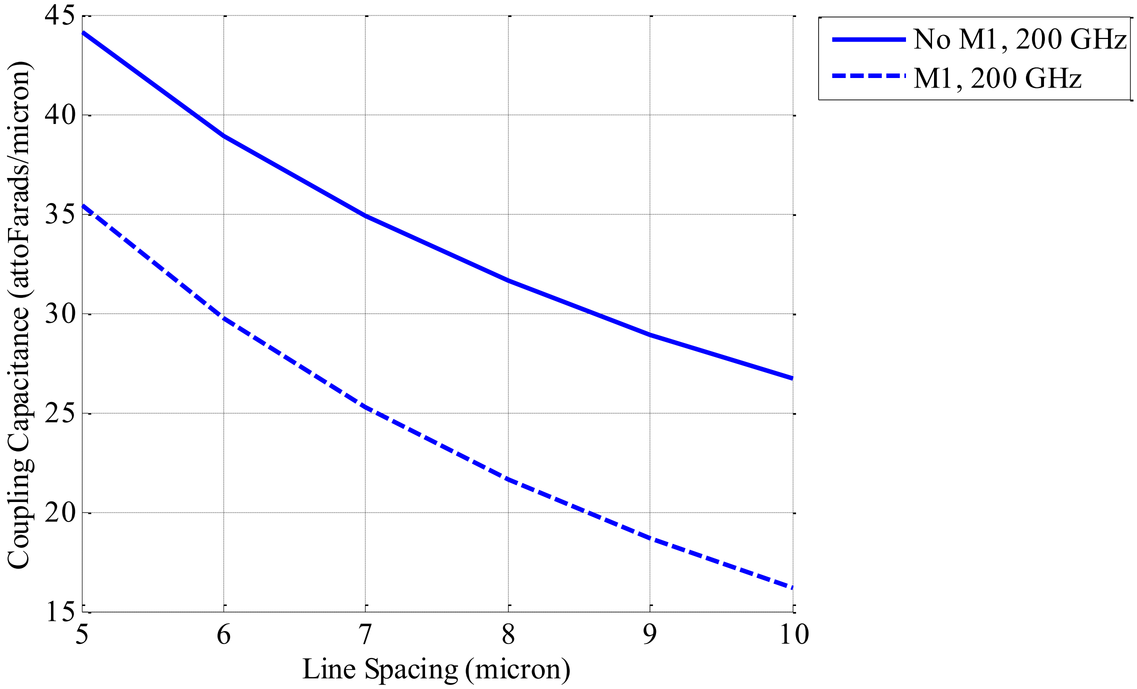


Figure 29. Waveguide: Capacitive Coupling, 5 µm Lines, 200 GHz

On the other hand, capacitance between the lines in the coplanar waveguide with the M1 ground (CPWM1) is only impacted by the spacing between the lines. A signal of 5-10 µm yields the same capacitive coupling given a constant spacing between the lines in the CPWM1. Overall, the addition of the M1 ground plane causes the capacitance to decrease by an amount of 10-13 attoFarad/micron. In the CPWM1, C1 ranges from 16.2-36.9 attoFarad/micron.

Figures 30 and 31 show the coupling capacitance for the coplanar microstrip lines with and without the M1 ground plane, CPMM1 and CPM, respectively, for line widths of 5 µm at 100 and 200 GHz respectively. The CPM is shown with a solid line. The dashed lines represent the simulated C1 values for the CPMM1.

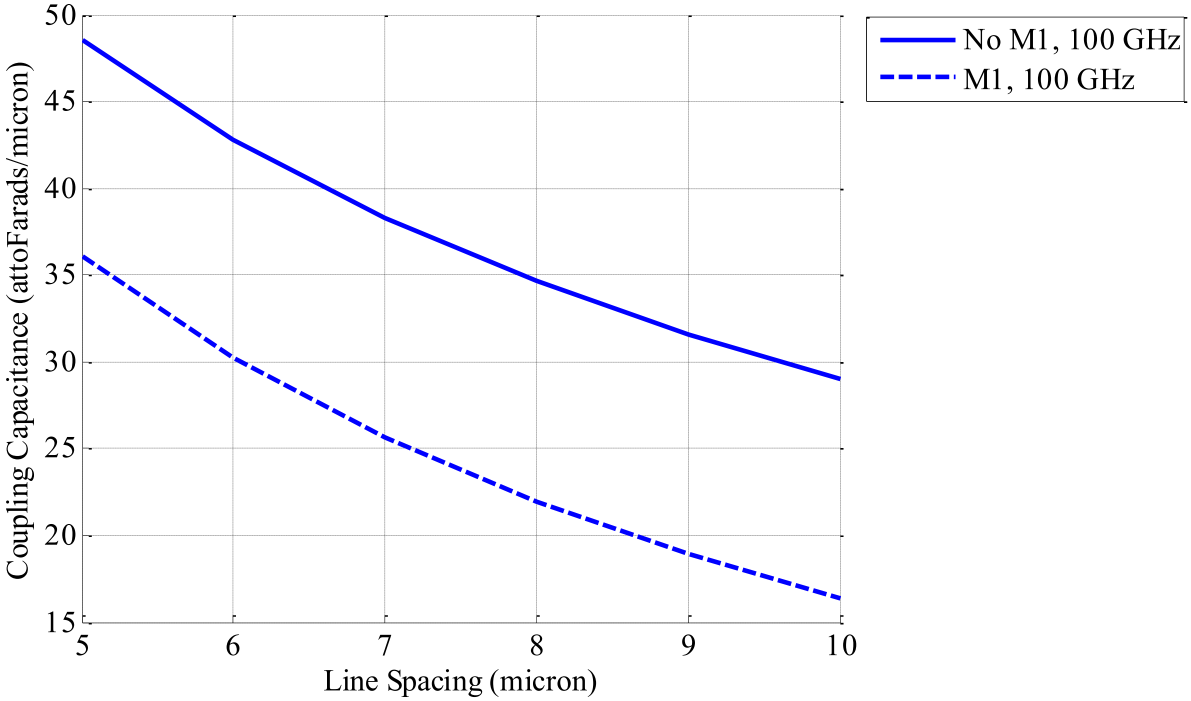


Figure 30. Microstrip: Capacitive Coupling, 5 µm Lines, 100 GHz

In the CPM, the capacitance between the lines is impacted only by the spacing between the lines for both the structure with the M1 ground plane and without. As spacing increases, the simulated values of the coupling capacitance decreases. For a given spacing, the value remains constant across all width and frequency values. Without the M1 ground plane, C1 is simulated between 28.9 and 53.5 attoFarad/micron. The addition of the M1 ground plane causes C1 to decrease to a range of 16.4-37.3 attoFarad/micron. Compared to the CPW, the CPM shows increased capacitance between the lines. However, the CPMM1 and CPWM1 yield nearly identical coupling capacitance values.

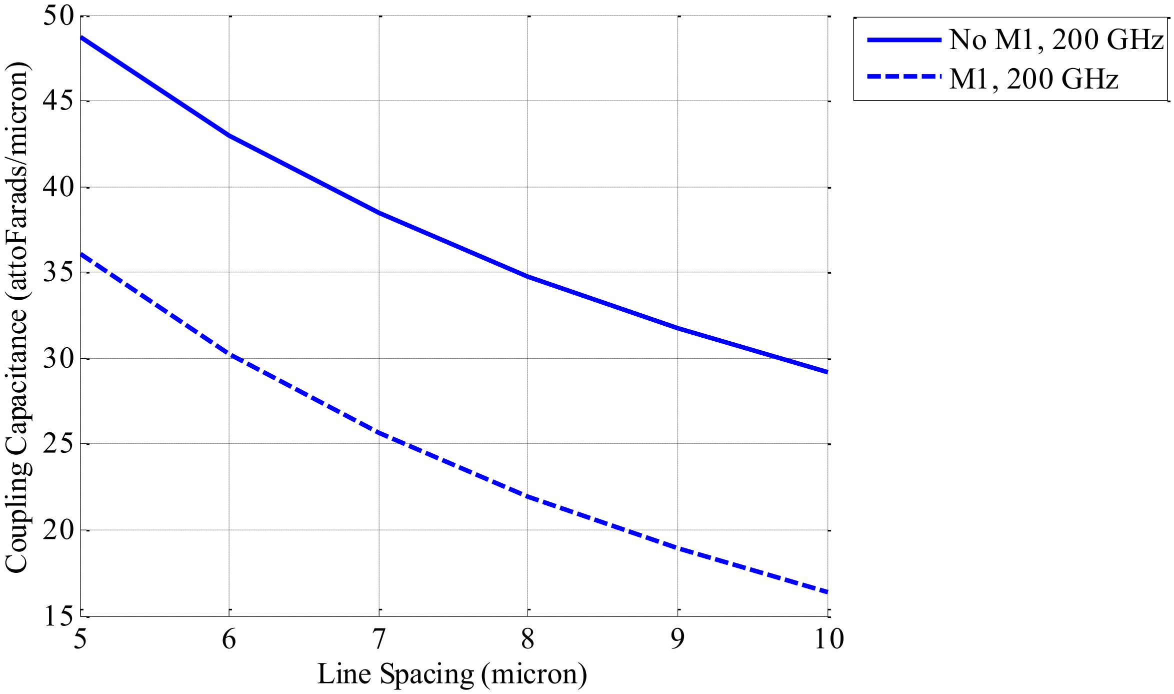


Figure 31. Microstrip: Capacitive Coupling, 5 μm Lines, 200 GHz

5.2.3 Capacitive Coupling to Ground, C2. The coupling capacitance between the lines and the substrate is represented by C2 in Figure 8 of section 4.2.2. Figures 32 and 33 show the simulated values of C2 for the CPW and CPWM1 with line widths of 5 μm at 100 GHz and 200 GHz respectively. The values shown are per C2 capacitor. So, the total capacitance to substrate in a single section of the line is twice the amount seen in the figures.

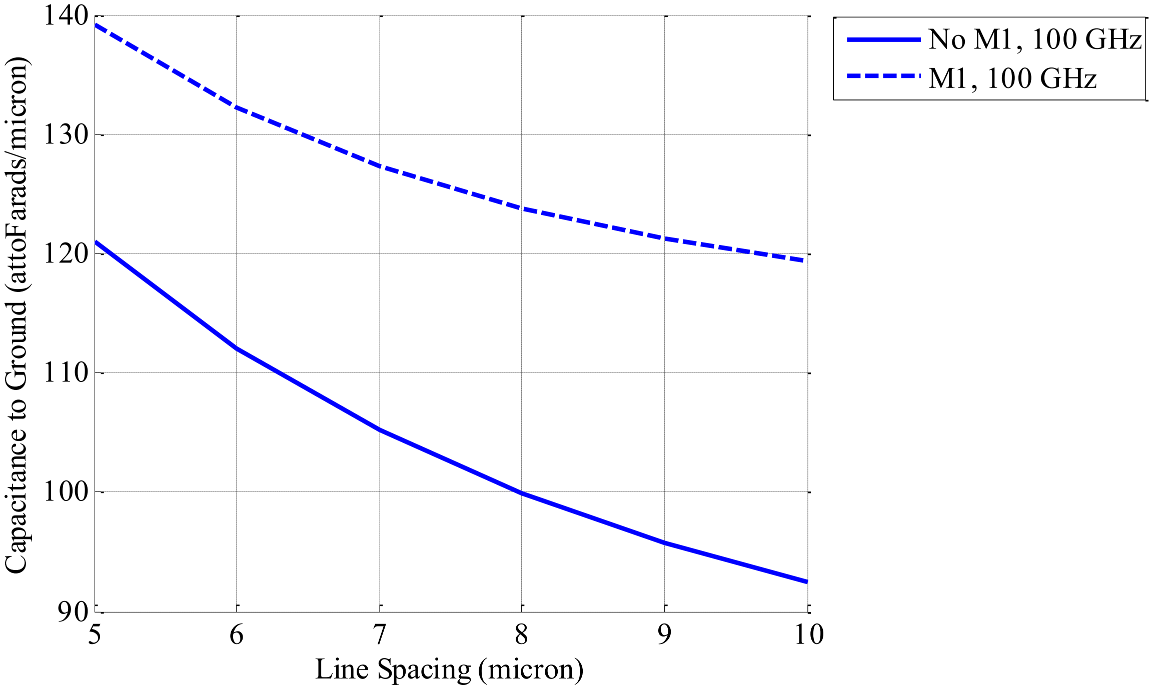


Figure 32. Waveguide: Capacitive Coupling to Ground, 5 µm Lines, 100 GHz

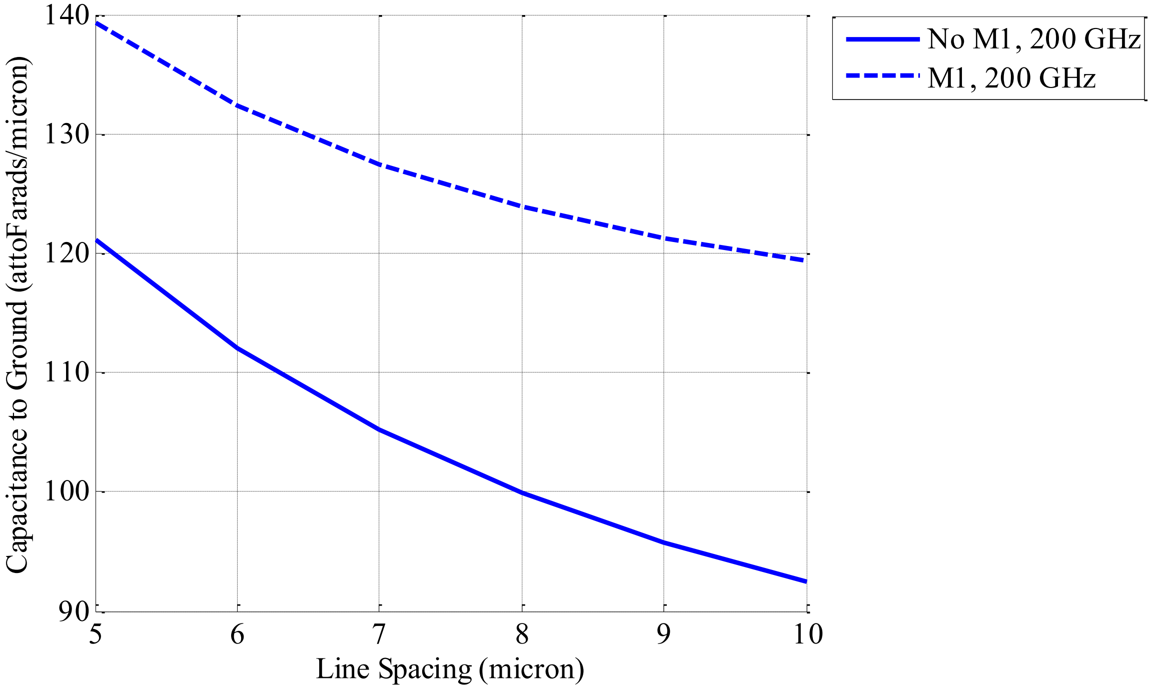


Figure 33. Waveguide: Capacitive Coupling to Ground, 5 µm Lines, 200 GHz

In the CPW, the capacitance to the ground plane remains flat across all frequencies. There is an increase in the simulated value of C2 as the width of the line increases and a decrease as spacing between the lines increases. Recall, the spacing between the signal lines and ground lines is modeled at identical values. The width of the signal and ground lines is also modeled with a single variable. Across all dimensions, the CPW has values ranging from 92.4 - 140.8 attoFarad/micron. The addition of the M1 ground plane causes this value to increase to 119.3 - 177.5 attoFarad/micron.

In Figures 34 and 35, the simulated results of C2 for the CPM and CPMM1 with line widths of 5 μm at 100 GHz and 200 GHz, respectively, are shown. The solid line represents the C2 values of the CPM and the dashed lines show the C2 values of the CPMM1.

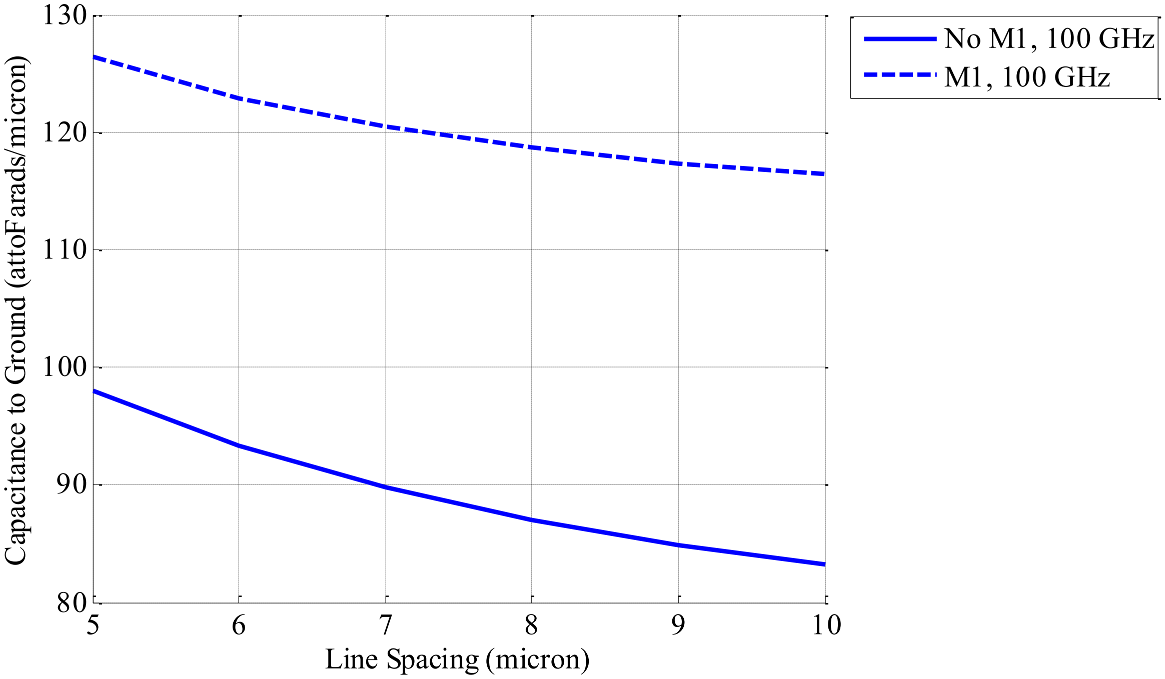


Figure 34. Microstrip: Capacitive Coupling to Ground, 5 μm Lines, 100 GHz

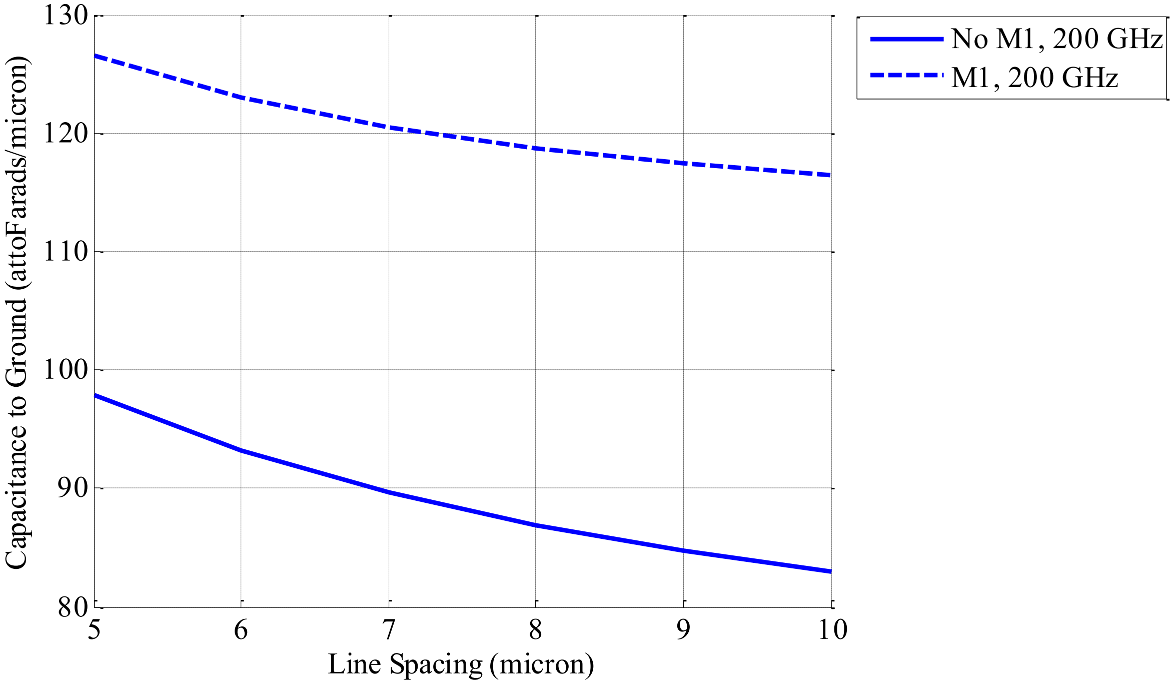


Figure 35. Microstrip: Capacitive Coupling to Ground, 5 µm Lines, 200 GHz

In the CPM, capacitance between the lines and the ground, C2, remains fairly constant across the frequencies. However, C2 increases with the width of the line and decreases as spacing between the lines increases. In the CPM, C2 ranges from 83.0 – 115.3 attoFarad/micron. The addition of the M1 ground plane leads to an increase in C2 by an average amount of 31.8 attoFarad/micron. In the CPMM1, C2 increases to a range of 116.4 – 164.2 attoFarad/micron. Overall, the microstrip lines display less capacitive coupling to ground than the waveguides.

5.2.4 Inductive Coupling Coefficient, K. The inductive coupling coefficient, K, is calculated from the extracted values as shown in equation (7) of Section 4.2. The inductive coupling coefficient can have any value between 0 and 1. A value of 0.5-1 represents lines that have strong inductive coupling, while values of 0-0.5 represent lines that are weakly coupled. In Figures 36-37, the simulated results of K for the CPW and CPWM1 are shown for a line width of 5 µm at 100 GHz and 200 GHz respectively.

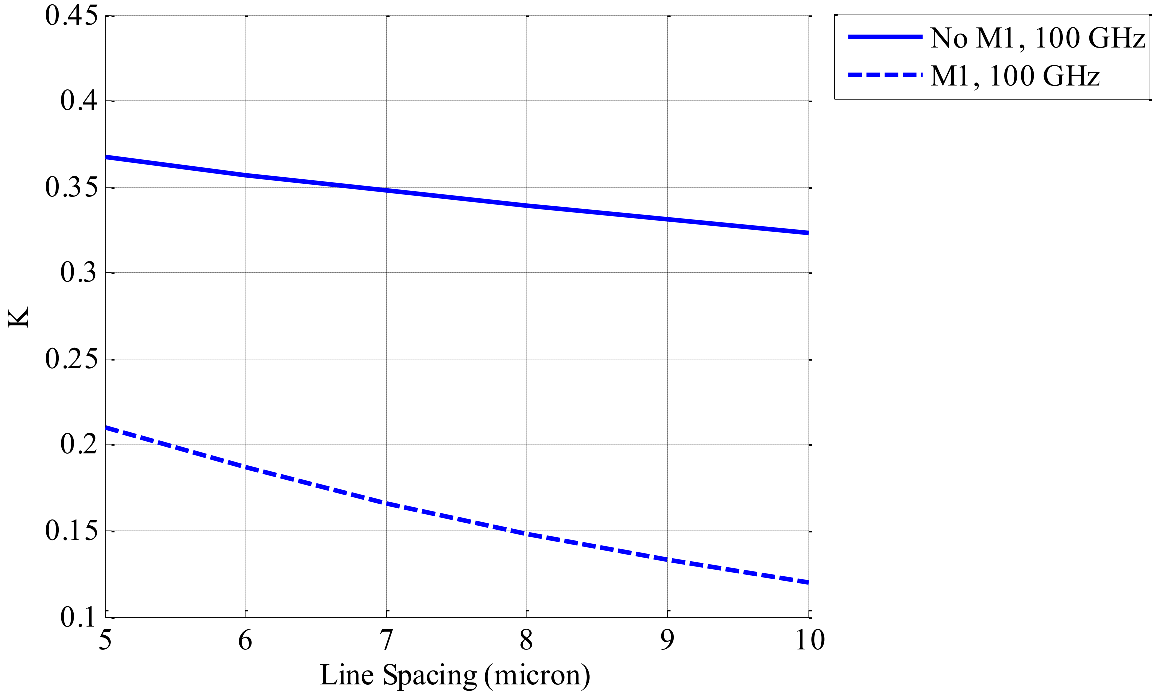


Figure 36. Waveguide: Inductive Coupling Coefficient, 5 μm Lines, 100 GHz

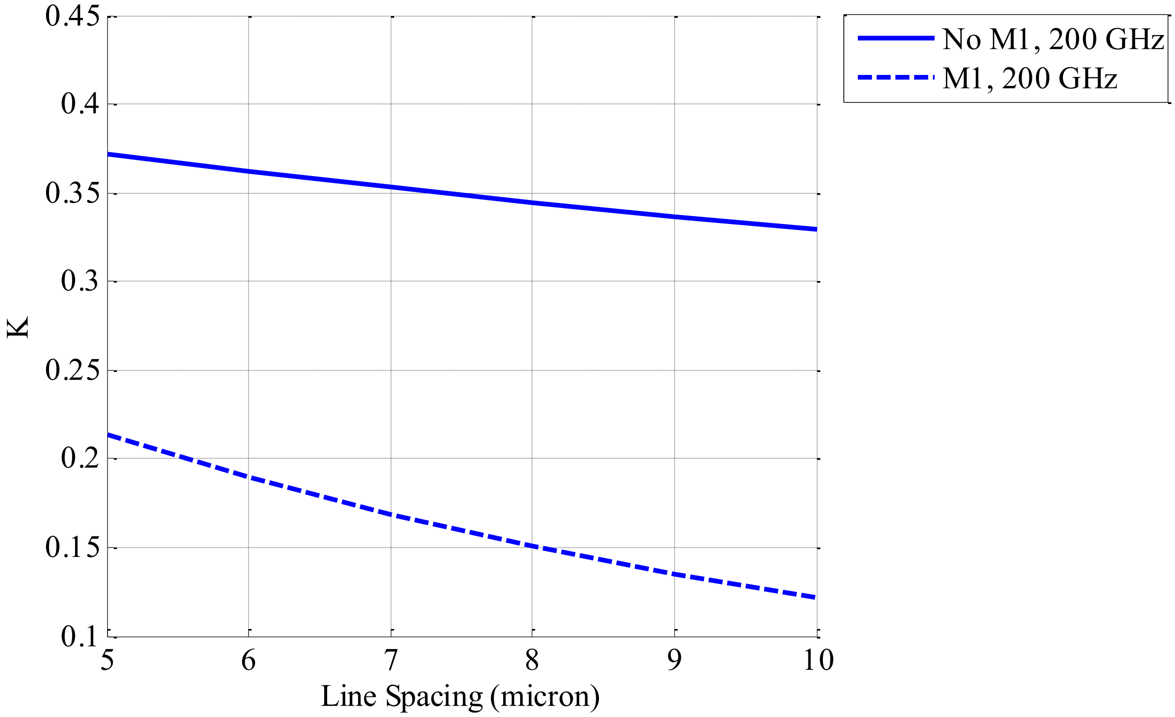


Figure 37. Waveguide: Inductive Coupling Coefficient, 5 μm Lines, 200 GHz

In both the CPW and CPWM1, the coupling coefficient remains fairly flat across the frequencies, decreases as spacing increases, and increases as the width of the lines increase. Recall that the spacing between all lines is the same and the width of all lines is the same in the coplanar waveguides. In the CPW, K ranges from 0.32 to 0.38. The addition of the M1 ground to the waveguides causes an overall drop in the inductive coupling coefficient to a range of 0.11 - 0.21.

Figures 38 and 39 show the simulated values of the inductive coupling coefficient against spacing for the CPM and CPMM1 with line widths of 5 micron at 100 and 200 GHz, respectively. It can be seen that the CPM displays more inductive coupling than the CPW. The addition of the M1 ground plane to the coplanar microstrips causes the CPMM1 to have nearly identical inductive coupling coefficients as the CPWM1.

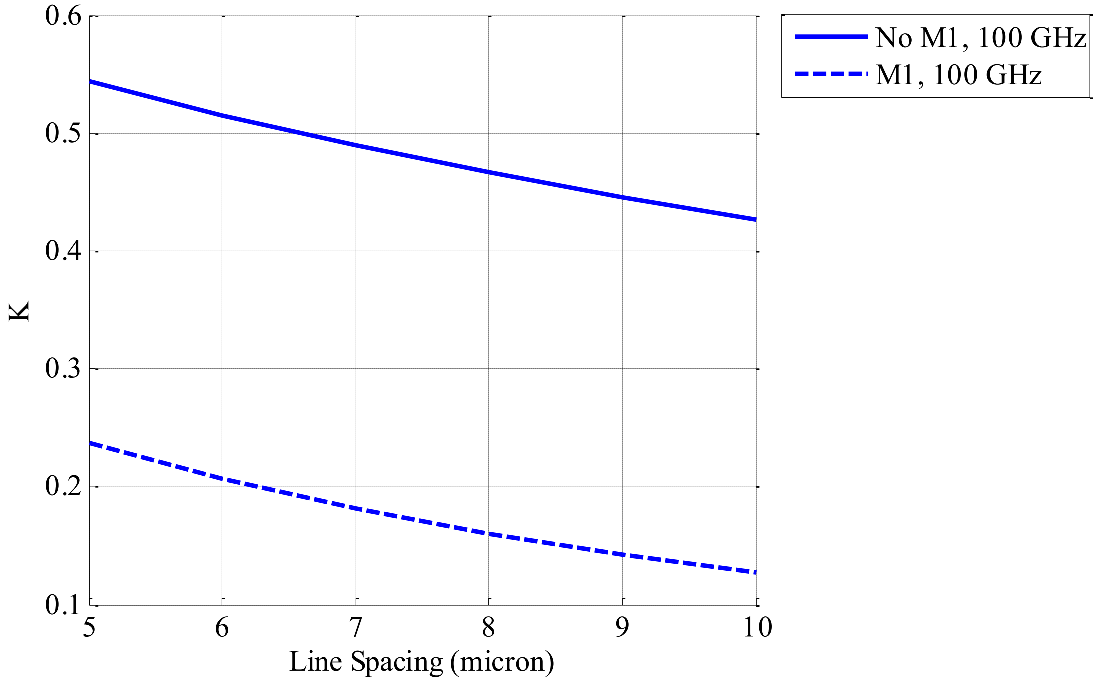


Figure 38. Microstrip: Inductive Coupling Coefficient, 5 μm Lines, 100 GHz

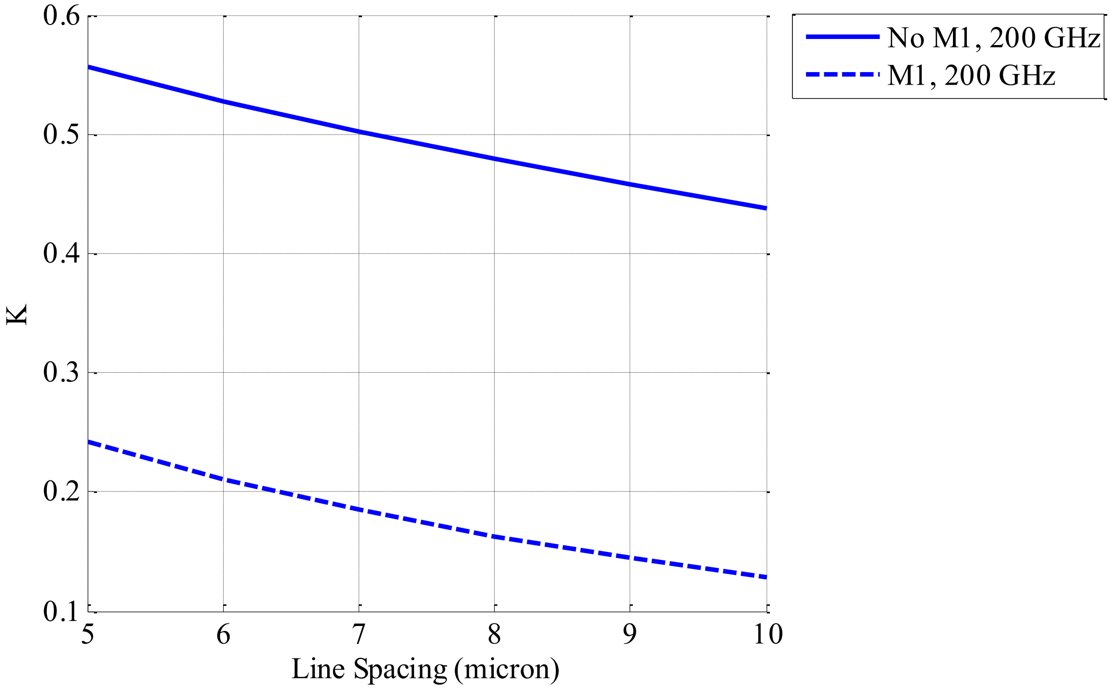


Figure 39. Microstrip: Inductive Coupling Coefficient, 5 μm Lines, 200 GHz

In the microstrip lines, inductive coupling between the lines also displays a flat response across the frequencies. In both the CPM and CPMM1, K decreases with increases in line width and decreases with increases in spacing. The CPM lines remain strongly coupled and maintain an inductive coupling coefficient with a range of 0.42 - 0.55 across all spacing and width dimensions. The addition of the ground plane causes the lines to display less coupling. In the CPMM1, the inductive coupling coefficient drops to a range of 0.11-0.24 across all spacing and width dimensions.

5.2.5 Series Inductance, L. The series inductance is represented by L in Figure 8 of section 4.2.2. Figures 40 and 41 show the simulated value of L for the CPW and CPWM1 with line widths of 5 μm at 100 and 200 GHz respectively. The values shown are per inductor, L. So, the total series inductance in a single line is twice the amount shown in the figures.

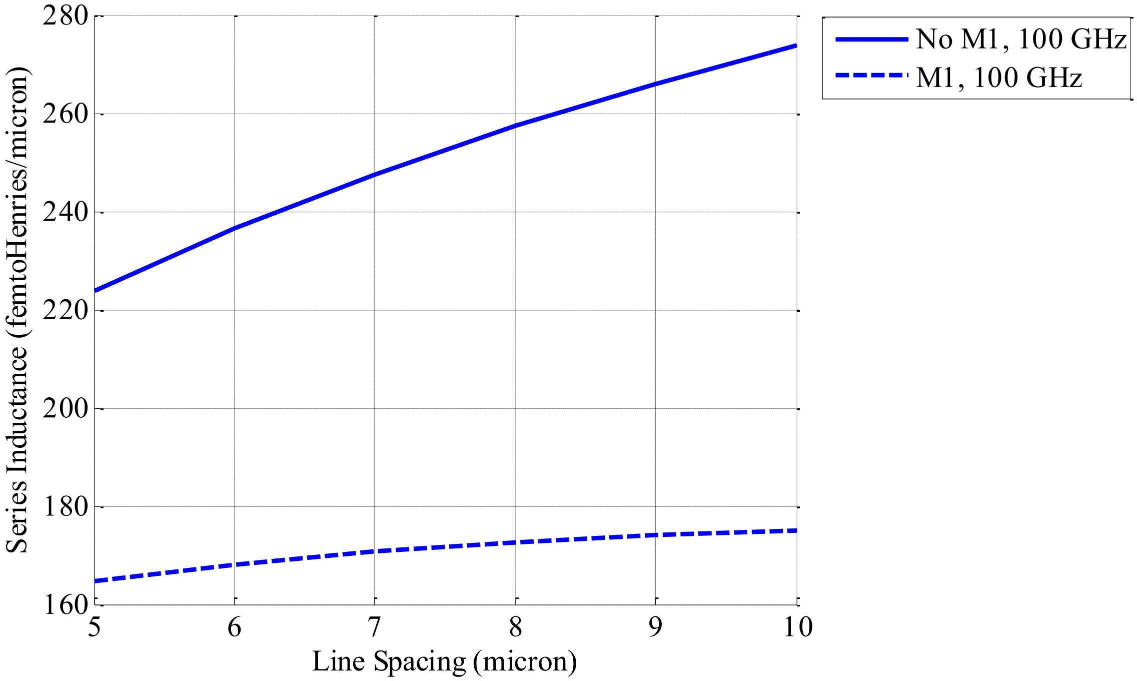


Figure 40. Waveguide: Series Inductance, 5 µm Lines, 100 GHz

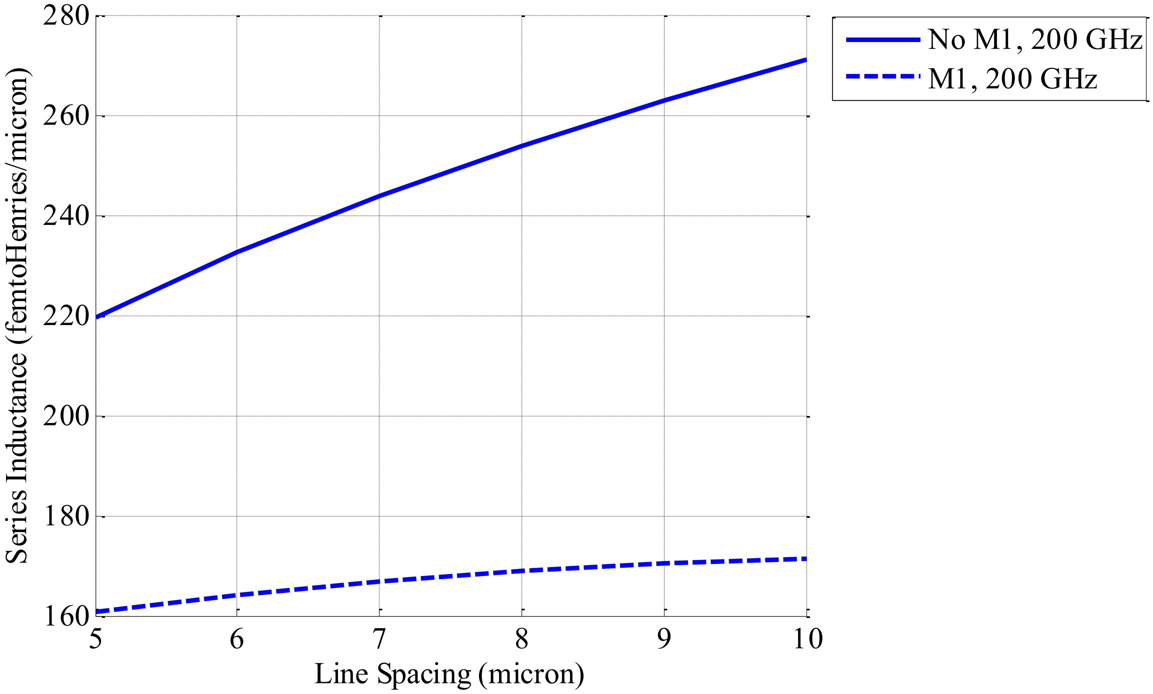


Figure 41. Waveguide: Series Inductance, 5 µm Lines, 200 GHz

The series inductance of the coplanar waveguides is minimally impacted by increases in frequency, as shown in Figures 40 and 41. In both the CPW and CPWM1, the series inductance decreases as width of the signal line increases from 5 to 10 micron. However, the CPW is more heavily impacted by spacing than the CPWM1. In the CPW, L exhibits an increase of 45 – 54 femtoHenry/micron as width remains constant and spacing increases from 5 to 10 µm. However, when the M1 ground is added, L only increases by 9 – 15 femtoHenry/micron given a constant width and an increase in spacing from 5 µm to 10 µm. Overall, the CPW displays a higher inductance value than the CPWM1. In the CPW, L is between 187.9 femtoHenry/micron and 273.8 femtoHenry/micron. When the M1 ground is added, these values decrease to a range of 125 to 170 femtoHenry/micron. Recall that these values are per inductor, L, in Figure 8 of section 4.2.2. In a single line, the total inductance will be twice the amounts discussed above.

In Figures 42 and 43, the series inductance of the coplanar microstrip lines is shown.

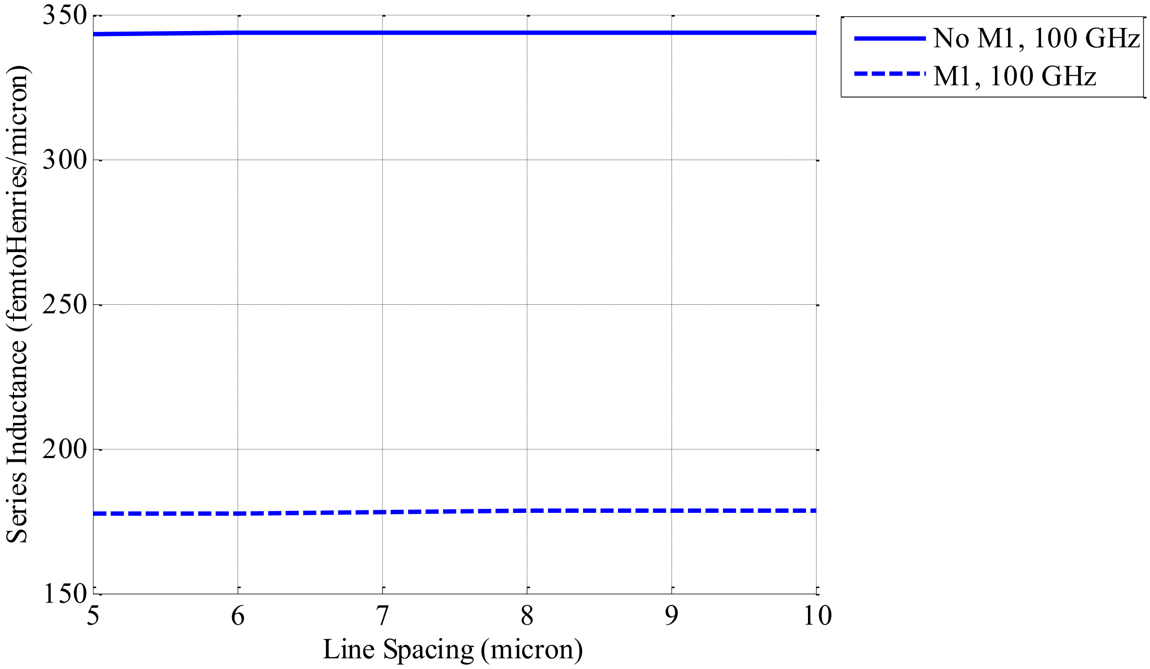


Figure 42. Microstrip: Series Inductance, 5 µm Lines, 100 GHz

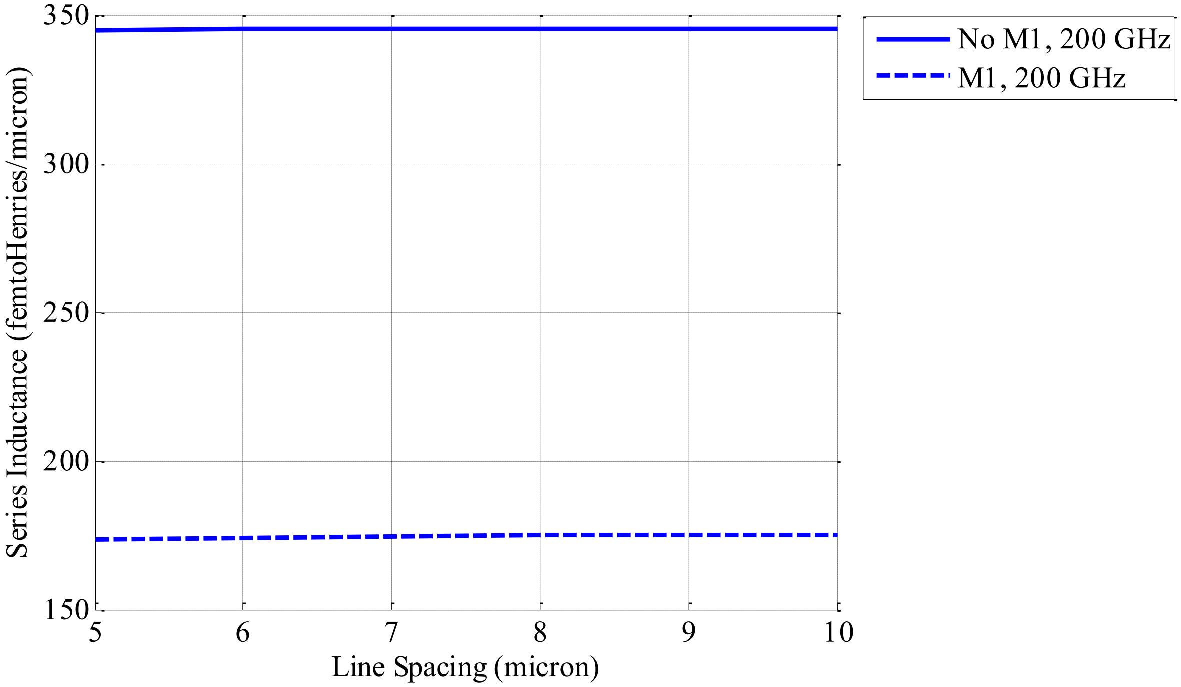


Figure 43. Microstrip: Series Inductance, 5 µm Line Lines, 200GHz

In the coplanar microstrip lines, the series inductance, L, displays the same trends for both the lines with the M1 ground plane and without. L is relatively unaffected by changes in spacing or the operating frequency, and as line width increases, L decreases. The impact of frequency and spacing between the lines can be seen in Figures 42 and 43 where the simulated values remain mostly flat across spacing and are nearly identical at 100 and 200 GHz, respectively. It can also be seen in Figures 42 and 43 that L is, however, strongly impacted by the change in the location of the ground plane. With the introduction of the M1 ground plane, series inductance along the line is cut in half. In the CPM, L ranges from 284.0 – 345.5 femtoHenry/micron. The CPMM1 sees a drop in L to a range of 125.9 – 178.1 femtoHenry/micron. Overall, the microstrips have higher series inductance than the waveguides.

5.2.6 Series Resistance, R. The series resistance is represented by R in Figure 8 of section 4.2.2. Figures 44 and 45 show the simulated value of R for the CPW and CPWM1 with line widths of 5 µm at 100 GHz and 200 GHz, respectively. The values shown are per resistor. So, the total series resistance in a single line is twice the amount shown in the figures.

In the coplanar waveguide, R is impacted by frequency, width of the signal line, and spacing. In both the CPW and CPWM1, the value of R increases as frequency increases and as spacing decreases. As expected, R decreases as the width of the signal lines increase. The addition of the M1 ground plane has an interesting impact on the resistance of the line. Notice in figures 44 and 45 that there is a change in the slope of the resistance when the M1 ground plane is added. With the addition of the M1 ground plane, the value of the resistance becomes more constant across the spacing values for a given width. Also, as frequency increases from 100 GHz to 200 GHz, the resistance increases at a slower rate in the CPWM1 than it does in the CPW. The total range of values for the CPW is 5.12 - 14.0 milliohms/micron and 5.81 - 12.5 milliohms/micron for the CPWM1.

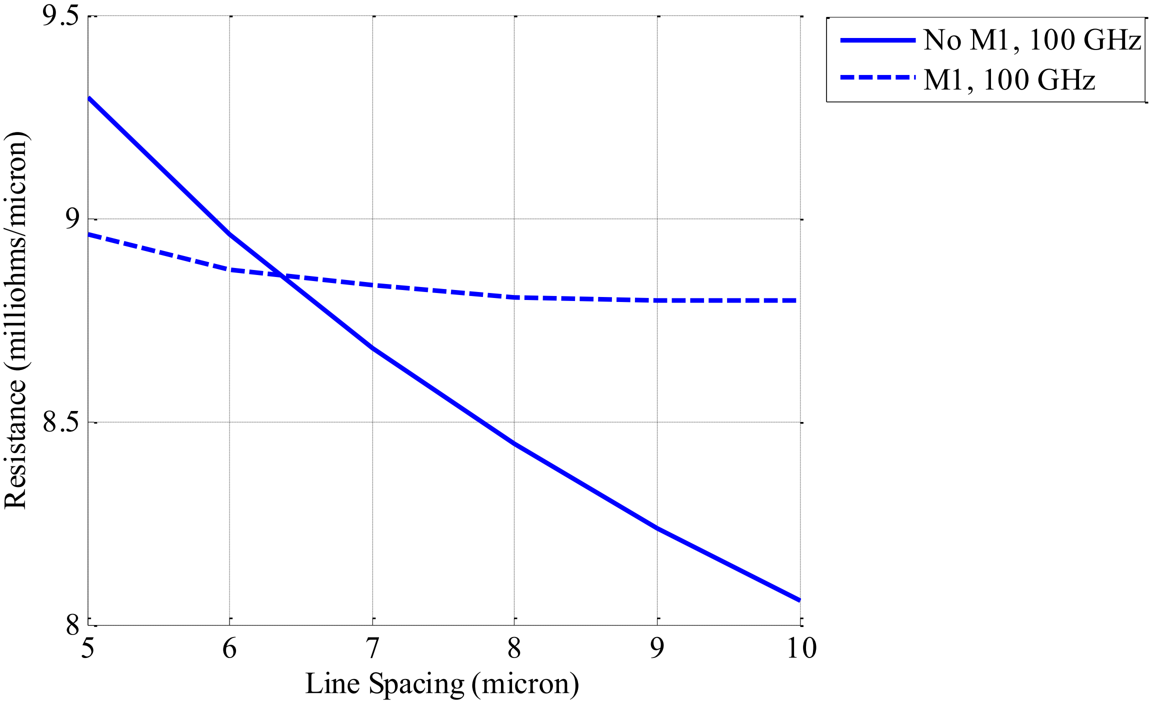


Figure 44. Waveguide: Series Resistance, 5 µm Lines, 100 GHz

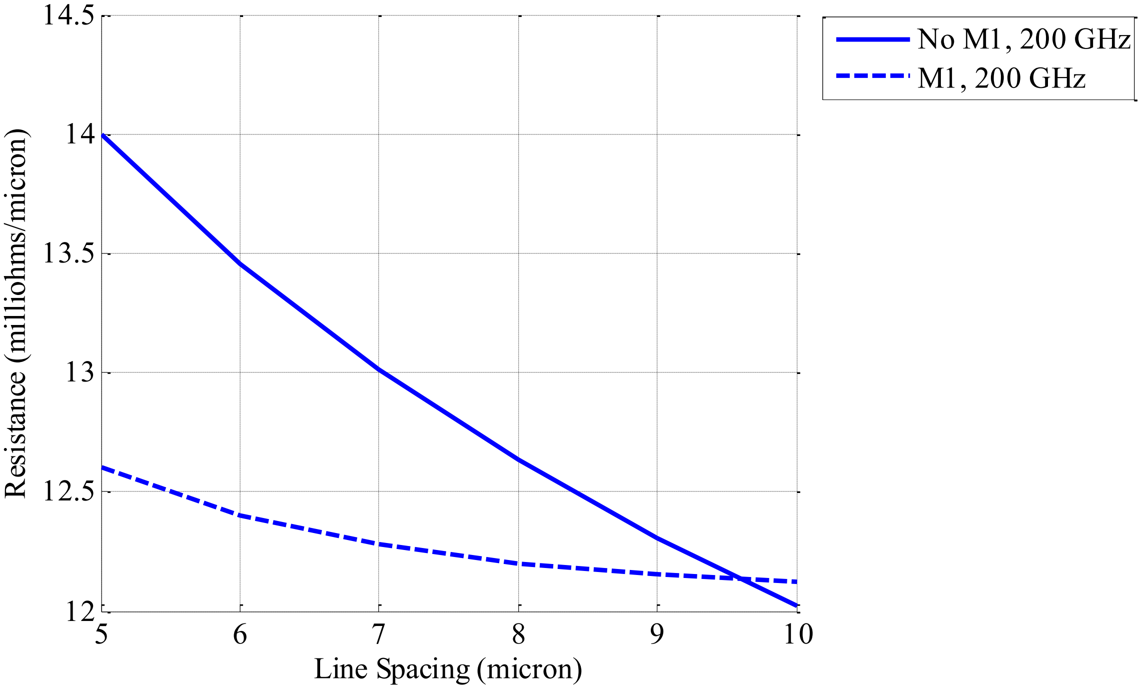


Figure 45. Waveguide: Series Resistance, 5 µm Lines, 200 GHz

The series resistance of the coplanar microstrip lines is shown below in Figures 46 and 47 for line widths of 5 µm at 100 GHz and 200 GHz respectively.

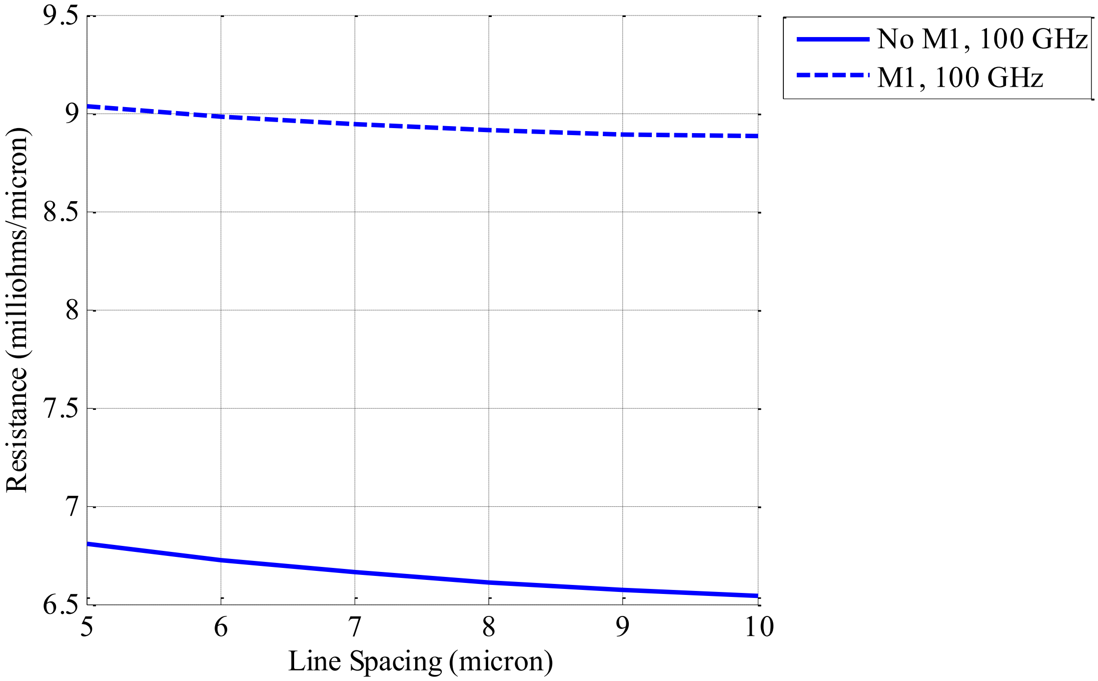


Figure 46. Microstrip: Series Resistance, 5 µm Lines, 100 GHz

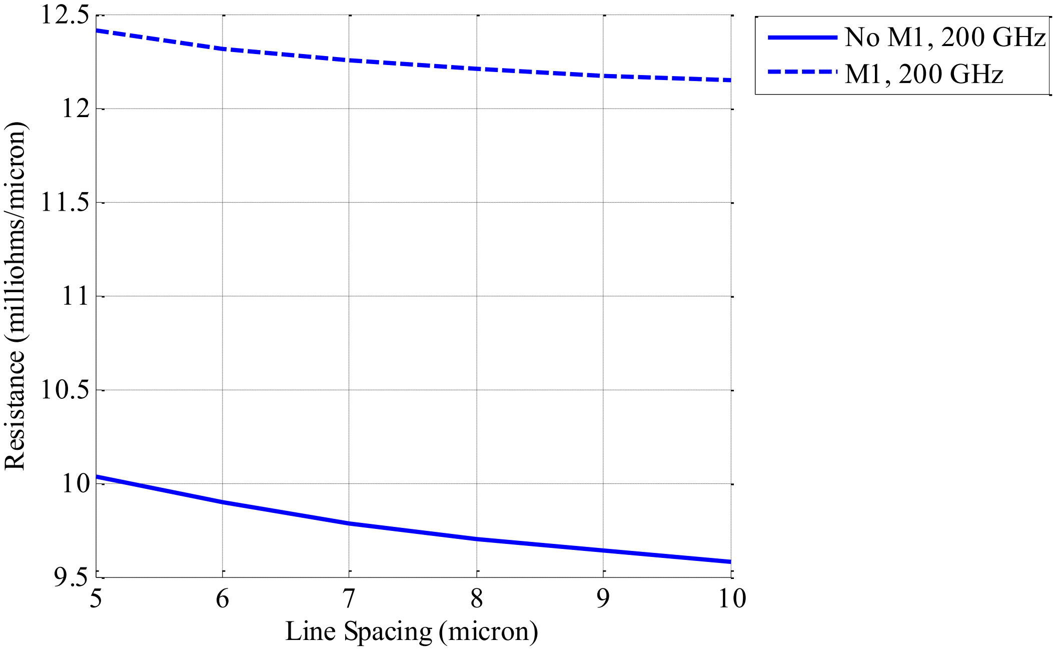


Figure 47. Microstrip: Series Resistance, 5 µm Lines, 200 GHz

In the coplanar microstrip lines, R is impacted directly by frequency and inversely by spacing and width. The CPM has resistance values of 3.9 – 10 milliohms/micron across the varying spacing-width combinations. The addition of the M1 ground plane leads to an increase of approximately 2 milliohms/micron, and the CPMM1 has a resistance range of 5.9 – 12.4 milliohms/micron. The series resistance in the lines of the CPM is slightly less than the series resistance in the lines of the CPW. However, the CPMM1 and CPWM1 show similar R values.

5.3 Behavioral Model Results

The propagation delay of the lines per unit length for the coplanar waveguides is shown in Figures 48 and 49, for 5 µm lines at 100 GHz and 200 GHz, respectively. The solid line represents the CPW while the dashed line represents the CPWM1. Recall that in the coplanar waveguide the spacing between the signal lines is the same as the spacing between the signal and ground lines, and the width of the signal lines is the same as the width of the ground lines.

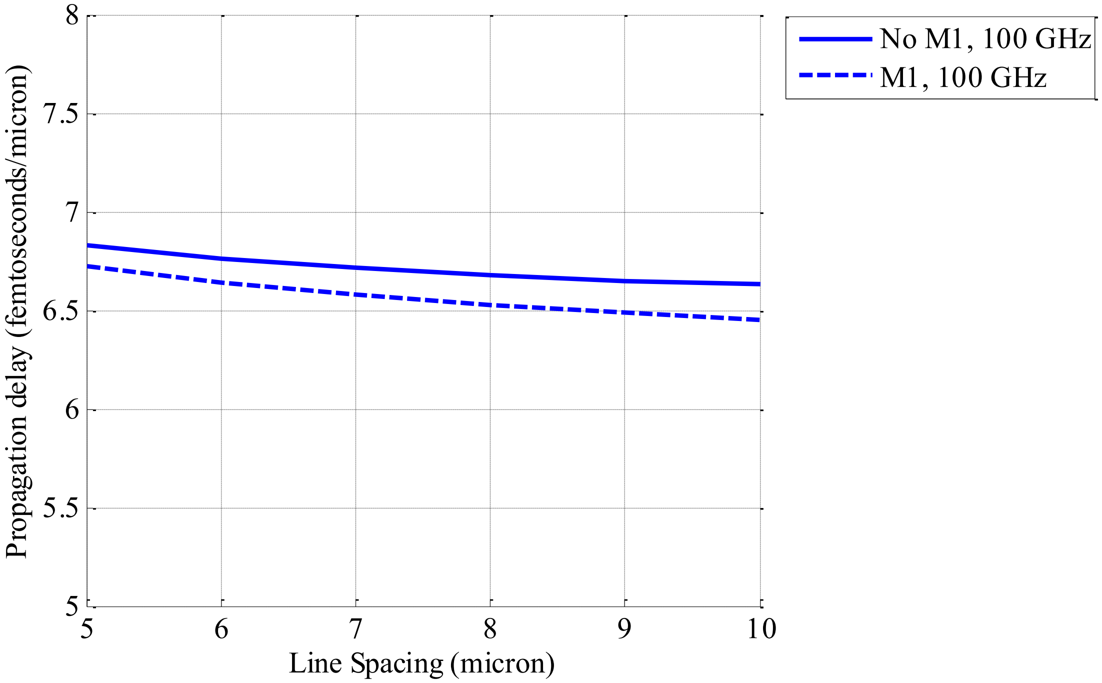


Figure 48. Waveguide: Propagation Delay, 5 µm Lines, 100 GHz

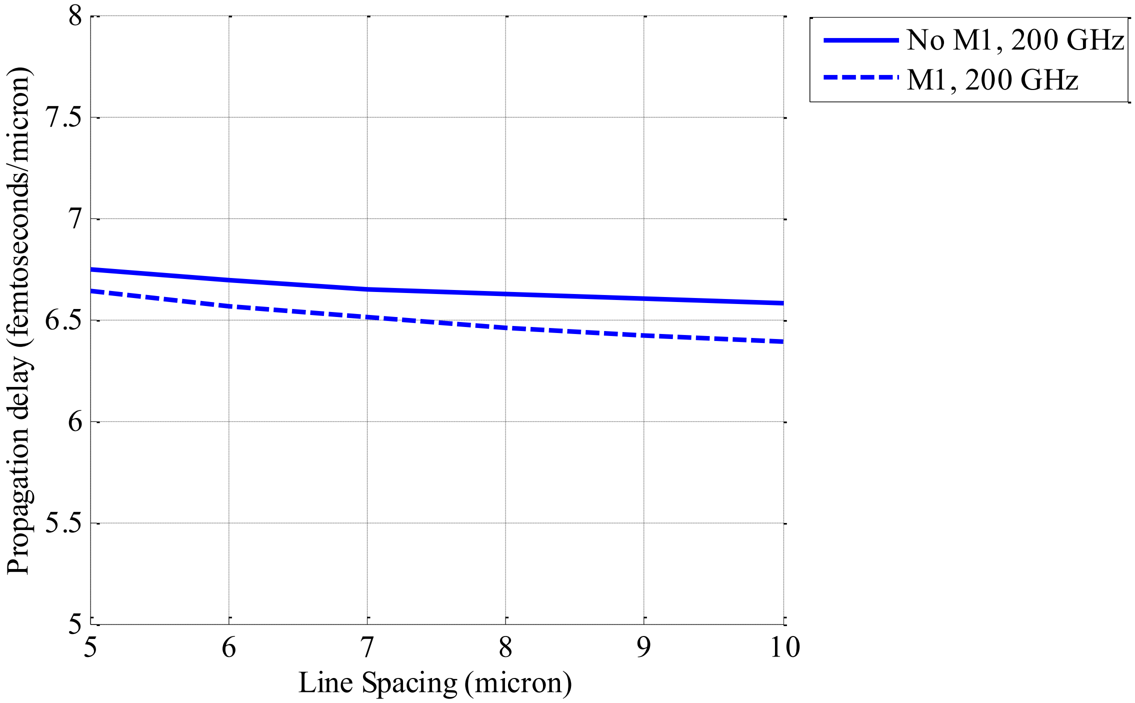


Figure 49. Waveguide: Propagation Delay, 5 µm Lines, 200 GHz

In Figures 48 and 49, it can be seen that changes in spacing and maximum frequency have little impact on the propagation delay. This is also true of the width of the lines. As spacing increases from 5 to 10 micron for a constant width, the delay decreases by a mere 0.019 femtoseconds/micron. As width increases from 5 to 10 micron for a constant spacing, the propagation delay decreases by 0.05 femtoseconds/micron. The addition of the M1 ground plane has the largest impact on the propagation delay, but it is still small enough to be considered negligible. The addition of the M1 ground causes an overall decrease in the propagation time, by an average amount of 0.1-0.3 femtoseconds/micron. In the CPW, propagation times range from 6.54 to 6.827 femtoseconds/micron. The signals in the CPWM1 travel slightly faster with a propagation delay of 6.23 - 6.72 femtoseconds/micron. However, for signals traveling at 100 – 200 GHz, the propagation delay can be considered flat across the frequencies, width, and spacing for both the CPW and CPWM1.

Figures 50 and 51 show the propagation delay per unit length for coplanar microstrips with lines of 5 micron width at 100 GHz and 200 GHz respectively. In the CPM and CPMM1, changes in width, spacing, and frequency again have negligible impacts on the propagation delay. As seen with the CPW and CPWM1, the largest impact on the delay is seen with the addition of the M1 ground plane, which causes an overall decrease in the propagation delay. However, the changes are still minimal and the delay of the coplanar microstrips can also be considered flat across all spacing, width, and frequency combinations. The propagation delay per unit length of the CPM is simulated as 6.59 – 6.77 femtoseconds/micron. With the addition of the M1 ground plane to the coplanar microstrip structure, the value of the propagation delay drops to 6.28 – 6.49 femtoseconds/micron.

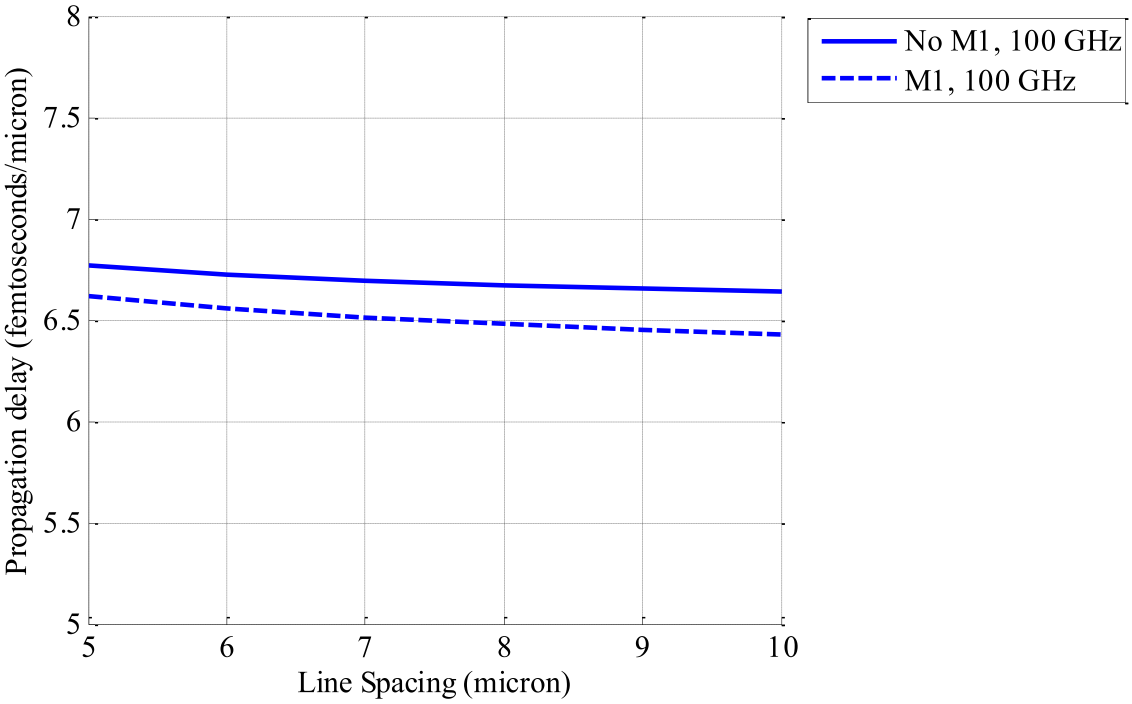


Figure 50. Microstrip: Propagation Delay, 5 µm Lines, 100 GHz

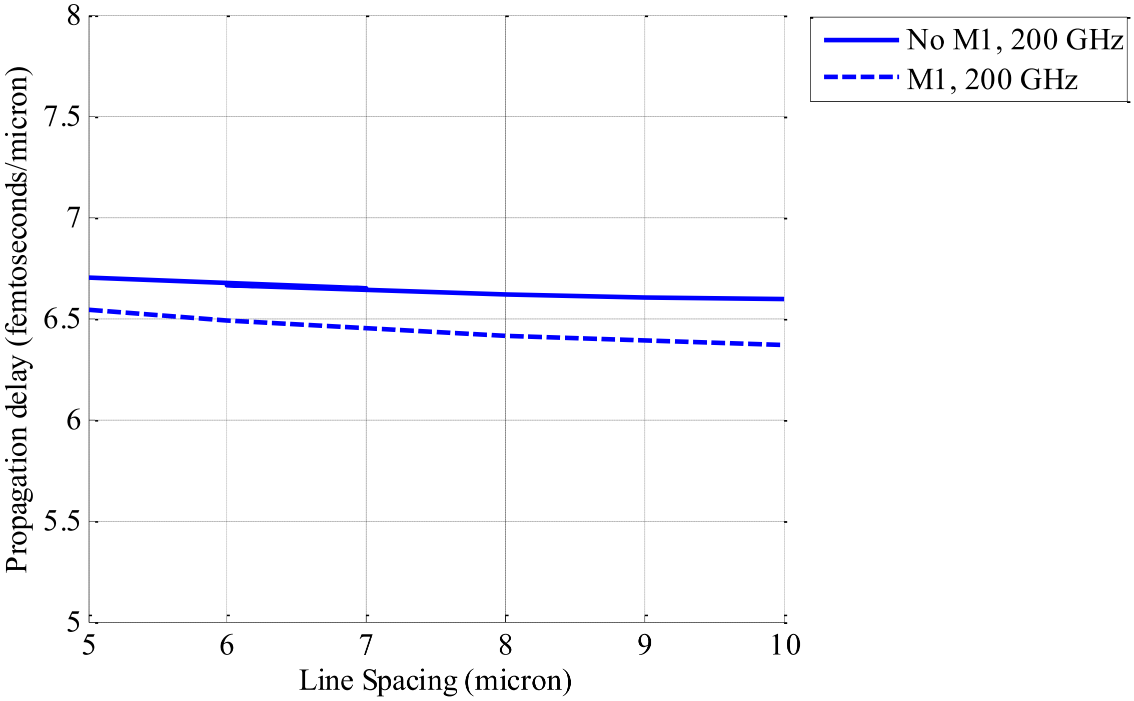


Figure 51. Microstrip: Propagation Delay, 5 µm Lines, 200 GHz

CHAPTER 6

Optimum Structures

In modeling the transmission lines, this work seeks to identify a set of structures that minimize delay and loss, and maximize characteristic impedance. Since the propagation delay of the lines are all small enough to support 100 – 200 GHz signals, the focus in identifying optimum structures is narrowed down to two optimization goals - the loss in the lines and the characteristic impedance of the lines. Ideally, the optimum structures will have characteristic impedance values of 50 ohms or higher, and loss values of 1 dB or less. For the 100 μm lines used in this work, that equates to a loss of 10 mdB/μm or less. Below, a set of line dimension that allow all three optimization goals to be met is presented.

6.1 Optimum Structures

To achieve characteristic impedances of 50 ohms or more, using line widths of 5 – 10 micron and spacing of 5 – 10 micron in a TSMC 65 nm process, it is best to eliminate the M1 ground. In the coplanar waveguide, the structures with the M1 ground achieve a maximum characteristic impedance of 47.7 ohms. The coplanar microstrip with the M1 ground yield a maximum characteristic impedance of 48.5 ohms. In order to achieve higher characteristic impedances with the M1 ground plane, lines separated by more than 10 μm are needed. However, in this work, separation between the lines is restricted to 10 micron.

To achieve loss below 10 mdB/μm while achieving characteristic impedance above 50 ohms, the coplanar waveguide structure is recommended. In the coplanar microstrip, there is only one width-spacing combination that allows all three optimization goals to be met across the 100 GHz to 200 GHz frequency band.

The width and spacing dimensions in Tables 1 and 2 are for the coplanar waveguide and coplanar microstrips, respectively, without the M1 ground plane. The lines in Table 1 yield 50.3 - 55.8 ohms of characteristic impedance across the frequencies of 100-200 GHz for a coplanar waveguide. The line dimensions in Table 2 yield a characteristic impedance of 59.2 ohms. It can be seen that the coplanar waveguide allows for more options than the coplanar microstrip in achieving characteristic impedances of 50 ohms or more and loss of 10 mdB/µm or less.

Table 1

Optimum Coplanar Waveguide Line Dimensions

Width (µm)	Spacing (µm)
5	8, 9, 10
6	9, 10
7	10

Table 2

Optimum Coplanar Microstrip Line Dimensions

Width (µm)	Spacing (µm)
10	10

The coplanar waveguide lines shown in Table 1 are loosely coupled with a simulated value of the inductive coupling coefficient between 0.3236 and 0.3445. Total series inductance in these lines is between 502 and 546 femtoHenry/micron. The total parasitic resistance ranges from 12.8-24 milliohms/micron for these structures. Capacitance values are small. The total capacitance seen between the ground and the signal lines is between 185 and 200 attoFarad/micron, and the capacitance between the lines is simulated between 53.2 and 63.2 attoFarad/micron.

The coplanar microstrip lines referenced in Table 2 have an inductive coupling coefficient of 0.4204-0.4322 as frequency increases from 100 GHz to 200 GHz. The total observed series inductance in this structure ranges from 571 to 575.8 femtoHenry/micron. The total resistance of the coplanar microstrip structure above is 7.95-11.82 milliohms/micron. Capacitance to the substrate falls between 199 and 199.4 attoFarad/micron, and the coupling capacitance between the lines ranges from 32.9 and 33.1 attoFarad/micron.

The propagation delay for the structures in Table 1 and Table 2 is 6.6-6.7 femtoseconds/micron.

CHAPTER 7

Conclusions and Future Work

7.1 Conclusions

This work sought to identify a set of lines with characteristic impedance above 50 ohms, loss below 1 dB, and propagation velocity sufficient to handle a 100-200 GHz signal in a 65 nm CMOS process. In identifying the lines, the following four differential transmission line structures were analyzed: the coplanar waveguide, with and without the lowest metal ground layer, and the coplanar microstrips, with and without the lowest metal ground layer. In identifying the impact of line width, spacing between the lines, and frequency, the width of the lines and spacing between the lines is swept from 5 to 10 micron, and the frequency was swept from 100 to 200 GHz. In the coplanar waveguide, where the ground lines run parallel to the signal lines, the width of the ground and signal lines remain constant and are swept together. Additionally, the spacing between the signal lines and the spacing between the signal and ground lines are also modeled with a single variable and swept simultaneously.

Simulations suggest that in order to achieve the three goals listed above, it is best to use the coplanar waveguide without the M1 ground plane. Using line widths of 5-7 micron and spacing between the lines of 8-10 micron in the coplanar waveguide, it is possible to meet all three optimization goals. The coplanar microstrip lines were also able to meet all three goals when line width and line spacing are both 10 micron. The addition of the M1 ground plane lowers loss in both the coplanar waveguide and the coplanar microstrip lines. However, with the addition of the M1 ground plane, neither the coplanar waveguide nor the coplanar microstrip lines were able to achieve 50 ohm characteristic impedances. To achieve characteristic impedances that high in structures with the M1 ground plane, it would be necessary to increase the spacing between the lines above 10 micron. Although larger separation between the lines is allowable, this work sought to identify lines that meet these goals within the 5-10 micron limits. So, the M1 ground plane needs to be eliminated within these dimensions.

7.2 Future Work

Recall that in this work several dimensions of the coplanar waveguide are modeled with a single variable. More specifically, the spacing between the signal lines and the spacing between the signal and ground lines are modeled with one variable. Also, the width of both the signal lines and the ground lines are modeled with a single variable.

Future work should identify the impact of spacing between signal and ground lines and spacing between the signal lines separately in the coplanar waveguides. For example, model the spacing between the signal and ground lines as SG, and model the spacing between the two signal lines as SS. Again, sweep both SG and SS from 5 micron to 10 micron. Identify the impact by studying the changes in the electromagnetic, electrical, and behavioral parameters of the lines as SG remains constant for a swept SS, and as SS remains constant for a swept SG.

Future work should also model the width of the ground line and the width of the signal line separately in the coplanar waveguides. Although increases in the width of the ground line were found to be minimal, an exhaustive exploration of the impact of the ground lines having a different width than the signal lines is beneficial. In doing so, the width of the ground line can be modeled using a single variable, WG, while the width of the signal line is modeled with a separate variable, WS. Then, the impacts of these variable values on the electromagnetic, electrical, and behavioral models of the transmission lines should be explored.

References

[1] R. K. Mongia, I. J. Bahl, P. Shartia, and J. Hong, *RF and Microwave Coupled-Line Circuits*, Second ed. United States: Artech House, Inc, 2007.

[2] C. P. Wen, "Coplanar Waveguide: A Surface Strip Transmission Line Suitable for Nonreciprocal Gyromagnetic Device Applications," *Microwave Theory and Techniques, IEEE Transactions on,* vol. 17, pp. 1087-1090, 1969.

[3] L. F. Tiemeijer, R. M. T. Pijper, J. A. van Steenwijk, and W. Van Noort, "Systematic Lumped-Element Modeling of Differential IC Transmission Lines," *Microwave Theory and Techniques, IEEE Transactions on,* vol. 57, pp. 1572-1580, 2009.

[4] B. Razavi, "The role of monolithic transmission lines in high-speed integrated circuits," in *Custom Integrated Circuits Conference, 2002. Proceedings of the IEEE 2002*, 2002, pp. 367-374.

[5] H. M. Greenhouse, "Design of Planar Rectangular Microelectronic Inductors," *Parts, Hybrids, and Packaging, IEEE Transactions on,* vol. 10, pp. 101-109, 1974.

[6] T. Sakurai and K. Tamaru, "Simple formulas for two- and three-dimensional capacitances," *Electron Devices, IEEE Transactions on,* vol. 30, pp. 183-185, 1983.

[7] N. Delorme, M. Belleville, and J. Chilo, "Inductance and capacitance analytic formulas for VLSI interconnects," *Electronics Letters,* vol. 32, pp. 996-997, 1996.

[8] U. Choudhury and A. Sangiovanni-Vincentelli, "Automatic generation of analytical models for interconnect capacitances," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on,* vol. 14, pp. 470-480, 1995.

[9] W. R. Eisenstadt and Y. Eo, "S-parameter-based IC interconnect transmission line characterization," *Components, Hybrids, and Manufacturing Technology, IEEE Transactions on,* vol. 15, pp. 483-490, 1992.

[10] L. F. Tiemeijer, R. M. T. Pijper, and W. Van Noort, "On the Accuracy of the Parameters Extracted From S-Parameter Measurements Taken on Differential IC Transmission Lines," *Microwave Theory and Techniques, IEEE Transactions on,* vol. 57, pp. 1581-1588, 2009.

[11] A. Hamidipour, M. Jahn, H. Jaeger, and A. Stelzer, "Characterization of Differential Transmission Lines for Integrated Millimeter-Wave Applications," *Microwave and Wireless Components Letters, IEEE,* vol. 22, pp. 188-190, 2012.

[12] B. Gustavsen, "Time delay identification for transmission line modeling," in *Signal Propagation on Interconnects, 2004. Proceedings. 8th IEEE Workshop on*, 2004, pp. 103-106.

[13] C. Warns, W. Menzel, and H. Schumacher, "Transmission lines and passive elements for multilayer coplanar circuits on silicon," *Microwave Theory and Techniques, IEEE Transactions on,* vol. 46, pp. 616-622, 1998.

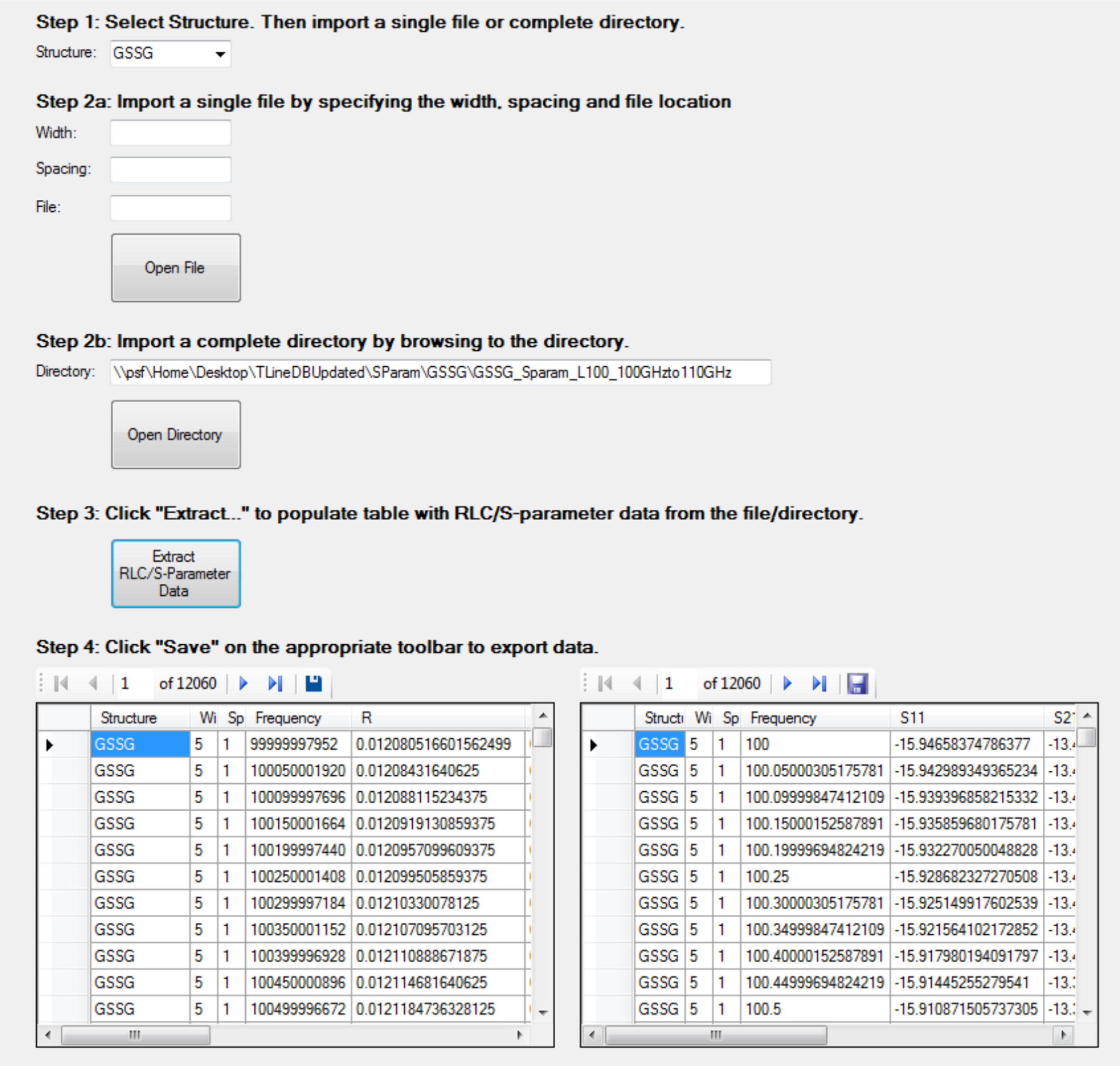
[14] B. Kleveland, C. H. Diaz, D. Vook, L. Madden, T. H. Lee, and S. S. Wong, "Exploiting CMOS reverse interconnect scaling in multigigahertz amplifier and oscillator design," *Solid-State Circuits, IEEE Journal of,* vol. 36, pp. 1480-1488, 2001.

[15] L. F. Tiemeijer, R. M. T. Pijper, R. J. Havens, and O. Hubert, "Low-Loss Patterned Ground Shield Interconnect Transmission Lines in Advanced IC Processes," *Microwave Theory and Techniques, IEEE Transactions on,* vol. 55, pp. 561-570, 2007.

[16] L. F. Tiemeijer, R. M. T. Pijper, R. J. Havens, and O. Hubert, "Corrections on "Low-Loss Patterned Ground Shield Interconnect Transmission Lines in Advanced IC Processes"," *Microwave Theory and Techniques, IEEE Transactions on,* vol. 55, pp. 1811-1811, 2007.

Appendix

A.1 Graphical User Interface



Step 1: Select Structure. Then import a single file or complete directory.

Structure: GSSG

Step 2a: Import a single file by specifying the width, spacing and file location

Width:

Spacing:

File:

Step 2b: Import a complete directory by browsing to the directory.

Directory: \\psf\Home\Desktop\TLineDBUpdated\SParam\GSSG\GSSG_Sparam_L100_100GHzto110GHz

Step 3: Click "Extract..." to populate table with RLC/S-parameter data from the file/directory.

Step 4: Click "Save" on the appropriate toolbar to export data.

1 of 12060

Structure	Wi	Sp	Frequency	R
GSSG	5	1	99999997952	0.012080516601562499
GSSG	5	1	100050001920	0.01208431640625
GSSG	5	1	100099997696	0.012088115234375
GSSG	5	1	100150001664	0.0120919130859375
GSSG	5	1	100199997440	0.0120957099609375
GSSG	5	1	100250001408	0.012099505859375
GSSG	5	1	100299997184	0.01210330078125
GSSG	5	1	100350001152	0.012107095703125
GSSG	5	1	100399996928	0.012110888671875
GSSG	5	1	100450000896	0.012114681640625
GSSG	5	1	100499996672	0.0121184736328125

1 of 12060

Structi	Wi	Sp	Frequency	S11	S2
GSSG	5	1	100	-15.94658374786377	-13.4
GSSG	5	1	100.05000305175781	-15.942989349365234	-13.4
GSSG	5	1	100.09999847412109	-15.939396858215332	-13.4
GSSG	5	1	100.15000152587891	-15.935859680175781	-13.4
GSSG	5	1	100.19999694824219	-15.932270050048828	-13.4
GSSG	5	1	100.25	-15.928682327270508	-13.4
GSSG	5	1	100.30000305175781	-15.925149917602539	-13.4
GSSG	5	1	100.34999847412109	-15.921564102172852	-13.4
GSSG	5	1	100.40000152587891	-15.917980194091797	-13.4
GSSG	5	1	100.44999694824219	-15.91445255279541	-13.3
GSSG	5	1	100.5	-15.910871505737305	-13.3

A.2 C# Code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Data.Entity;
using System.Diagnostics;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;

namespace TLineDBFormApplication
{
    public partial class DBForm : Form
    {
        //
        //global variables declaration
        //
        private string fileName, folderName;
        string[] directoryContents;

        //holds data from the input .dat file
        float Z0;
        float freq;
        float L1_1, L2_1, L2_2;
        float R1_1, R2_1, R2_2;
        float C1_1, C2_1, C2_2;
        float G1_1, G2_1, G2_2;

        //holds data from the input .s4p file
        double S11, S21, S31, S41;

        string CSVfile; //location of output file

        //structure and dimensions of the line
        int lineWidth;
        int lineSpacing;
        string structure;

        int RLCsaveCounter = 0;    //to identify data being extracted: 1=L, 2=R, 3=C, 4=G
        int SParamsaveCounter = 0; //to identify data being extracted: 1=S11, 2=S21, 3=S31, 4=S41

        tLineDataSet.StructuresRow newRLCline;
        SparamDataSet.SParamTableRow newSParamline;

        //file or directory flags, mutually exclusive
        bool fileSelected = false;
        bool directorySelected = false;

        int paramFile = 0;
        int formatFile = 0;
        int[,] WidthSpacingArray = new int[61,2]; // stores width-spacing assignments
        int arraylineWidth, arraylineSpacing;
```

```
public DBForm()
{
   InitializeComponent();
}

private void DBForm_Load(object sender, EventArgs e)
{
   // TODO: This line of code loads data into the 'sparamDataSet.SParamTable' table.
   this.sParamTableTableAdapter.Fill(this.sparamDataSet.SParamTable);
   // TODO: This line of code loads data into the 'tLineDataSet1.Structures' table.
   this.structuresTableAdapter.Fill(this.tLineDataSet1.Structures);
}

//saves selected structure
private void strucExtractComboBox_SelectedIndexChanged(object sender, EventArgs e)
{
   if (strucExtractComboBox.SelectedIndex == 0)
      structure = "GSSG";
   else if (strucExtractComboBox.SelectedIndex == 1)
      structure = "GSSG M1";
   else if (strucExtractComboBox.SelectedIndex == 2)
      structure = "SS";
   else if (strucExtractComboBox.SelectedIndex == 3)
      structure = "SS M1";
}//end method strucExtractComboBox_SelectedIndexChanged

private void openButton_Click(object sender, EventArgs e)
{
   DialogResult result;   //create dialog box enabling user to open file

   using (OpenFileDialog fileChooser = new OpenFileDialog())
   {
      result = fileChooser.ShowDialog();  // shows the OpenFile Dialog box
      fileName = fileChooser.FileName;    //saves selected file name and location
   } // end using fileChooser

   // exit event handler if user clicked Cancel
   if (result == DialogResult.OK)  //if user clicks ok
   {
      // show error if user specified invalid file
      if (fileName == string.Empty)   //check that file name is not empty
         MessageBox.Show("Invalid File Name", "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
      else
      {
         fileTextBox.Width = 500;    //increase width of text box
         fileTextBox.Text = fileName;    //display selected file name in text box
         directoryTextBox.Text = "";     //clear directory text box
         fileSelected = true;    //tell the program that a single file was selected
         directorySelected = false;  //set directory flag to false
      } // end else
   } // end if
}// end method openButton_Click

private void openDirectoryButton_Click(object sender, EventArgs e)
{
   // Show the FolderBrowserDialog.
   DialogResult result;

   using (FolderBrowserDialog folderChooser = new FolderBrowserDialog())
   {
      result = folderChooser.ShowDialog();
      folderName = folderChooser.SelectedPath;
   }

   if (result == DialogResult.OK)    //exit if user clicks cancel
   {
      if (folderName == string.Empty)
         MessageBox.Show("Invalid Directory Name", "Error",
           MessageBoxButtons.OK, MessageBoxIcon.Error);
      else
      {
         directoryTextBox.Width = 500;
         directoryTextBox.Text = folderName;
         fileTextBox.Text = "";
         directorySelected = true;
         fileSelected = false;

         directoryContents = Directory.GetFiles(@folderName);
      }
   }
}   //end method openDirectory_Click

//saves file/directory name, and line dimensions, and calls appropriate extraction method
private void extractButton_Click(object sender, EventArgs e)
{
   Stopwatch stopWatch = new Stopwatch();

   stopWatch.Start();

  try
   {
      if (structure == null)
      {
         MessageBox.Show("For which Structure does this data apply?");
      }
      else
      {
         if (fileSelected || (fileTextBox.Text != ""))   //if a file was selected
         {
            //if width and/or spacing is missing, alert user and exit method
            if (widthTextBox.Text == "" || spacingTextBox.Text == "")
   {
      MessageBox.Show("Oops! You forgot the line dimensions.");
      return;
   }
   else
   {
      fileName = fileTextBox.Text;  //save file name

      bool DATFile = fileName.EndsWith(".dat",
               System.StringComparison.CurrentCultureIgnoreCase);
      bool s4pFile = fileName.EndsWith(".s4p",
               System.StringComparison.CurrentCultureIgnoreCase);

      if (DATFile || s4pFile)
      {
         //save string/character as an integer to the lineWidth and lineSpacing variables
         lineWidth = Int32.Parse(widthTextBox.Text); //save line width
         lineSpacing = Int32.Parse(spacingTextBox.Text); //save line spacing

         //call appropriate extraction method
         if (DATFile)
            extractRLCValues();
         else if (s4pFile)
            extractSParamValues();
      }
      else //if not DATFile or s4pFile
      {
         MessageBox.Show("This application only accepts .dat and .s4p file formats.");
         return;
      }
   }
}  //end if fileSelected
else if (directorySelected)
{
   foreach (string file in directoryContents)
   {
      fileName = file;   //save file from array to fileName

      bool DATFile = fileName.EndsWith(".dat",
               System.StringComparison.CurrentCultureIgnoreCase);
      bool s4pFile = fileName.EndsWith(".s4p",
               System.StringComparison.CurrentCultureIgnoreCase);

      if (fileName.Contains("output_files"))
      {
         //capture width-spacing-paramfile combinations
         CreateWidthSpacingArray();
      }
      else if (fileName.Contains("_param"))
      {
         //check file name to assign width and spacing
```

```
            int index1;
            index1 = fileName.IndexOf("_param");

            string parameter = fileName.Substring((index1 + 6), 2);

            paramFile = Int32.Parse(parameter);

            lineWidth = WidthSpacingArray[paramFile, 0];
            lineSpacing = WidthSpacingArray[paramFile, 1];

            //call appropriate extraction method
            if (DATFile)
            {
              extractRLCValues();
            }
            else if (s4pFile)
            {
              extractSParamValues();
            }
            else
            {
              MessageBox.Show("This program only accepts .dat and .s4p file formats for
                      extractions.");
            }
          }
        }   //end for each file in directory

        paramFile = 0;  //reset paramFile value after iterating through all the parameter files
      } //end if directory selected

      //clear array after extracting all data from the directory
      Array.Clear(WidthSpacingArray, 0, 61);

      stopWatch.Stop();

      // Get the elapsed time as a TimeSpan value.
      TimeSpan ts = stopWatch.Elapsed;
      string elapsedTime = String.Format("{0:00}:{1:00}:{2:00}.{3:00}", ts.Hours, ts.Minutes,
                ts.Seconds, ts.Milliseconds / 10);

      MessageBox.Show("Extraction complete. Please review the data before exporting. Elapsed
                Time: " + elapsedTime);
    }
  }   //end try to extract file name and line dimensions
  catch (Exception error)
  {
    string error1 = error.ToString();
    MessageBox.Show(error1);
  }
}   // end method extractRLCButton_Click
```

```
//extracts width and spacing key from the key file
private void CreateWidthSpacingArray()
{
  FileStream fileFormat = null;

  fileFormat = new FileStream(fileName, FileMode.Open, FileAccess.Read);

  try
  {
    // specify file from which data is read (fileReader of StreamReader type)
    //fileReader will read the data using the stream that was created
    using (StreamReader fileReader = new StreamReader(fileFormat))
    {
      try
      {
        // go back to the beginning of the file
        fileFormat.Seek(0, SeekOrigin.Begin);

        // traverse file until end of file
        while (fileReader.Peek() > -1)
        {
          //zero out row 0
          WidthSpacingArray[0,0] = 0;
          WidthSpacingArray[0,1] = 0;

          //read in line
          string formatLine = fileReader.ReadLine();

          if (formatLine.Contains("_param"))
          {
            //extract the parameter file number
            int index2;
            index2 = formatLine.IndexOf("_param");

            string parameter = formatLine.Substring((index2 + 6), 2);

            formatFile = Int32.Parse(parameter);

            for (int i = 0; i < 4; i++)
            {
              formatLine = fileReader.ReadLine(); //read in a new line

              if (formatLine.StartsWith("W")) //capture line width for parameter file
              {
                int equal = formatLine.LastIndexOf("=");
                string w = formatLine.Substring((equal + 2));

                int period = w.LastIndexOf(".0");
                w = w.Substring(0, period);

                arraylineWidth = Int32.Parse(w);
```

```
                            }
                            else if (formatLine.StartsWith("S"))    //capture spacing for parameter file
                            {
                                int equal = formatLine.LastIndexOf("=");
                                string s = formatLine.Substring((equal + 2));

                                int period = s.LastIndexOf(".0");
                                s = s.Substring(0, period);

                                arraylineSpacing = Int32.Parse(s);
                            }
                        }

                        //assign spacing and width of the parameter file
                        //row = file number, column 0 = width, column 1 = spacing
                        WidthSpacingArray[formatFile, 0] = arraylineWidth;
                        WidthSpacingArray[formatFile, 1] = arraylineSpacing;

                        //reset array variables before exiting the while loop to avoid errors
                        arraylineWidth = 0;
                        arraylineSpacing = 0;
                    }
                }
            }
            catch (IOException ioExcept) // handle exception when errors occur within the try statement
            {
                MessageBox.Show(ioExcept.ToString());
            } // end catch
        }

        fileFormat = null;   //dispose of fileFormat streamreader
    }
    catch (Exception except)
    {
        MessageBox.Show(except.ToString());
    }//end catch
    finally //expose of reader if it has not been done already
    {
        if (fileFormat != null)
            fileFormat.Dispose();
    }
}   //end method CreateWidthSpacingArray

//Loops through file to extract data
private void extractRLCValues()
{
    FileStream input = null;

    input = new FileStream(fileName, FileMode.Open, FileAccess.Read);

    try
```

```
{
   //specify file from which data is read (fileReader of StreamReader type)
   //fileReader will read the data using the stream that was created
   using (StreamReader fileReader = new StreamReader(input))
   {
      bool DATFile = fileName.EndsWith(".dat",
                     System.StringComparison.CurrentCultureIgnoreCase);

      if (DATFile)
      {
         try
         {
            //go back to the beginning of the file
            input.Seek(0, SeekOrigin.Begin);

            //traverse file until end of file
            while (fileReader.Peek() > -1)
            {
               string[] inputFields; //stores individual pieces of data

               // get next line in file
               string inputLine = fileReader.ReadLine();

               //flag to check for header or comment lines
               bool commentHeader = (inputLine.StartsWith("; ") ||
                              inputLine.StartsWith("FORMAT"));

               // at the end of the file, exit method
               if (inputLine == null)
                  return;

               //statements below test the string for certain conditions to discard the
               //comment and formatting lines, and/or prepare the valid data for storing
               else if (commentHeader) //skip comment lines
               {
                  //check that Z0 is not in the comments
                  if (!inputLine.Contains("(ohms)"))
                  {
                     inputLine = string.Empty;
                  }
                  //if this is the header for the Z0, save Z0
                  else if (inputLine.Contains("(ohms)"))
                  {
                     //read in next line where Z0 is contained
                     inputLine = fileReader.ReadLine();

                     //trim comment marker
                     inputLine = inputLine.Substring(1);

                     while (inputLine.StartsWith(" "))
                     {
```

```
            inputLine = inputLine.Substring(1);
        }

        //Separate data by space between them. Save values into a separate array index
        inputFields = inputLine.Split(' ');

        Z0 = float.Parse(inputFields[2]);

        inputLine = string.Empty;
    }
}
//string manipulation to delete leading spaces that are read in
//during the readline operation
else if (inputLine.StartsWith(" ")) //if string begins with spaces
{
    //delete all leading spaces
    while (inputLine.StartsWith(" "))   //while a leading space exists
    {
        inputLine = inputLine.Substring(1); //truncate first character
    }

    //skip formatting lines hidden by leading spaces
    if (inputLine.StartsWith("R1:1"))
    {
        inputLine = string.Empty;
    }
    else if (inputLine.StartsWith("C1:1"))
    {
        inputLine = string.Empty;
    }
    else if (inputLine.StartsWith("G1:1"))
    {
        inputLine = string.Empty;
    }
}

if (inputLine != string.Empty)
{
    //if colon is present eliminate colon and space after the colon
    if (inputLine.Contains(':'))
    {
        int index;
        index = inputLine.IndexOf(':');

        inputLine = inputLine.Remove(index, 2);
    }

    //separate data by space between them and save values into a separate array index
    inputFields = inputLine.Split(' ');

    if (inputFields.Length > 4)    //if data has been extracted and saved to the test array
```

```
{
   //check to see if frequency value is present
   if (inputFields.Length > 6) //frequency value will require two additional cells
   {
      //extract frequency and L data
      freq = float.Parse(inputFields[0]);
      L1_1 = float.Parse(inputFields[2]);
      L2_1 = float.Parse(inputFields[4]);
      L2_2 = float.Parse(inputFields[6]);

      RLCsaveCounter++;  //increment save counter
   } //end save frequency and L data
   else  //otherwise, read in R, C, G data
   {
      if (RLCsaveCounter == 1)    //reading in R data
      {
         R1_1 = float.Parse(inputFields[0]);
         R2_1 = float.Parse(inputFields[2]);
         R2_2 = float.Parse(inputFields[4]);

         RLCsaveCounter++;  //increment save counter
      } //end save R data
      else if (RLCsaveCounter == 2)  //reading in C data
      {
         C1_1 = float.Parse(inputFields[0]);
         C2_1 = float.Parse(inputFields[2]);
         C2_2 = float.Parse(inputFields[4]);

         RLCsaveCounter++;  //increment save counter
      }   //end save C data
      else if (RLCsaveCounter == 3)  //reading in G data
      {
         G1_1 = float.Parse(inputFields[0]);
         G2_1 = float.Parse(inputFields[2]);
         G2_2 = float.Parse(inputFields[4]);

         //create new row
         newRLCline = tLineDataSet1.Structures.NewStructuresRow();

         //save extracted values to the columns in the new row
         newRLCline.Structure = structure;
         newRLCline.Width = lineWidth;
         newRLCline.Spacing = lineSpacing;
         newRLCline.Frequency = freq;
         newRLCline.R = (R1_1 * 1E-6) / 2;
         newRLCline.Ls = (L1_1 * 1E-6) / (2 * 1E-9);
         newRLCline.Km = L2_1 / (Math.Sqrt(L1_1 * L2_2));
         newRLCline.Csubs = (C1_1 * 1E-6) / 1E-12;
         newRLCline.Cline = C2_1 / 1E-6;
         newRLCline.Z0 = Z0;
```

```
                              //check that row does not already exist
                              DataRow foundrow =
                                 tLineDataSet1.Structures.FindByStructureWidthSpacingFrequency(
                                 structure, lineWidth, lineSpacing, freq);

                              if (foundrow == null)         //if row does not exist already
                              {
                                 //add new row to the dataset
                                 tLineDataSet1.Structures.AddStructuresRow(newRLCline);
                              }
                              else if (foundrow != null)
                              { }//do nothing

                              RLCsaveCounter = 0;  //reset save counter
                              foundrow = null;   //reset row check variable
                           } //end save G data
                        } //end copy R, C, G data
                     }//end copy data to variables and database
                  } //end "if input not empty"
               } // end while not end of file
            } // end try statement
            catch (IOException ioExcept) // handle exception when errors occur within try statement
            {
               MessageBox.Show(ioExcept.ToString());
            } // end catch
         }   //end if DATFile
         else //if not DATFile
         {
            MessageBox.Show("This application only accepts .dat file formats for RLGC data.");
         }

         input = null;   //dispose of input
      }   //end using streamreader
   } //end try
   catch (Exception except)
   {
      MessageBox.Show(except.ToString());
   }//end catch
   finally //expose of reader if it has not been done already
   {
      if (input != null)
         input.Dispose();
   }
} // end method extractRLCValues

//Loops through File to extract S-Parameter Data
private void extractSParamValues()
{
   FileStream input = null;

   input = new FileStream(fileName, FileMode.Open, FileAccess.Read);
```

```
try
{
   // set file from where data is read (fileReader of StreamReader type)
   //fileReader will read the data using the stream that was created
   using (StreamReader fileReader = new StreamReader(input))
   {
      bool s4pFile = fileName.EndsWith(".s4p",
           System.StringComparison.CurrentCultureIgnoreCase);

      if (s4pFile)
      {
         try
         {
            // go back to the beginning of the file
            input.Seek(0, SeekOrigin.Begin);

            // traverse file until end of file
            while (fileReader.Peek() > -1)
            {
               string[] inputFields; // stores individual pieces of data

               // get next line in file
               string inputLine = fileReader.ReadLine();

               //flag to check for header or comment lines
               bool commentHeader = (inputLine.StartsWith("!") || inputLine.StartsWith("#"));

               // at the end of the file, exit method
               if (inputLine == null)
                  return;

               //if statements below test the string for certain conditions to discard the
               //comment and formatting lines, and/or prepare the valid data for storing
               else if (commentHeader) //skip comment lines
               {
                  //clear the input line and discard data that is read in
                  inputLine = string.Empty;
               }
               //string manipulation to delete leading spaces that are read in
               //during the readline operation
               else if (inputLine.StartsWith(" ")) //begin with space
               {
                  //delete all leading spaces
                  while (inputLine.StartsWith(" "))   //while a leading space exists
                  {
                     inputLine = inputLine.Substring(1); //truncate first whitespace
                  }
               }
```

```
if (inputLine != string.Empty)
{
   //separate data by space between them and save values into a separate array index
   inputFields = inputLine.Split(' ');

   if (inputFields.Length > 7)    //if data has been extracted and saved to the test array
   {
      //check to see if frequency value is present
      if (inputFields.Length > 8) //frequency value will require one additional cell
      {
         //extract frequency and S11 magnitude data
         freq = float.Parse(inputFields[0]);
         double hold = double.Parse(inputFields[1]);
         S11 = 20 * Math.Log10(hold);

         SParamsaveCounter++;  //increment save counter
      } //end save frequency and S11 magnitude data
      else  //otherwise, only S-parameter values are present
      {
         if (SParamsaveCounter == 1)    //reading in S2 values
         {
            double hold = double.Parse(inputFields[0]);
            S21 = 20 * Math.Log10(hold);

            SParamsaveCounter++;  //increment save counter
         } //end save S2 data
         else if (SParamsaveCounter == 2)  //reading in S3 data
         {
            double hold = double.Parse(inputFields[0]);
            S31 = 20 * Math.Log10(hold);

            SParamsaveCounter++;  //increment save counter
         }   //end save S3 data
         else if (SParamsaveCounter == 3)  //reading in S4 data
         {
            double hold = double.Parse(inputFields[0]);
            S41 = 20 * Math.Log10(hold);

            //create new row
            newSParamline = sparamDataSet.SParamTable.NewSParamTableRow();

            //save extracted values to the columns in the new row
            newSParamline.Structure = structure;
            newSParamline.Width = lineWidth;
            newSParamline.Spacing = lineSpacing;
            newSParamline.Frequency = freq;
            newSParamline.S11 = (float)S11;
            newSParamline.S21 = (float)S21;
            newSParamline.S31 = (float)S31;
            newSParamline.S41 = (float)S41;
```

```
                              //if row does not exist already
                              DataRow foundrow =
                                 sparamDataSet.SParamTable.FindByStructureWidthSpacingFrequency(
                                 structure, lineWidth, lineSpacing, freq);

                              if (foundrow == null)
                              {
                                 //add new row to the dataset
                                 sparamDataSet.SParamTable.AddSParamTableRow(newSParamline);
                              }
                              else if (foundrow != null)
                              { }//do nothing

                              SParamsaveCounter = 0;  //reset save counter
                              foundrow = null;
                           } //end save S4 data
                        } //end copy S-parameter data
                     }//end copy data to variables and database
                  }   //end "if input not empty"
               } // end while not end of file
            } // end try statement
            catch (IOException ioExcept) // handle exception when errors occur within try statement
            {
               MessageBox.Show(ioExcept.ToString());
            } // end catch
         }   //end if s4pFile
         else //if not s4pFile
         {
            MessageBox.Show("This application only accepts .s4p file formats for S-parameter
                data.");
         }

         input = null;   //dispose of input
      }   //end using streamreader
   } //end try
   catch (Exception except)
   {
      MessageBox.Show(except.ToString());
   }//end catch
   finally //expose of reader if it has not been done already
   {
      if (input != null)
         input.Dispose();
   }
}  // end method extractSParamValues

//assign output file name and location. Then, call WriteCSVfile function
private void structuresBindingNavigatorSaveItem_Click(object sender, EventArgs e)
{
   //check structure value and save to appropriate file
   if (structure == "GSSG")
```

```
    CSVfile = @"\\psf\Home\Documents\Visual Studio 2013\Projects\DataExtraction\GSSG
          RLC.csv";
  else if (structure == "GSSG M1")
    CSVfile = @"\\psf\Home\Documents\Visual Studio 2013\Projects\DataExtraction\GSSG_M1
          RLC.csv";
  else if (structure == "SS")
    CSVfile = @"\\psf\Home\Documents\Visual Studio 2013\Projects\DataExtraction\SS
          RLC.csv";
  else
    CSVfile = @"\\psf\Home\Documents\Visual Studio 2013\Projects\DataExtraction\SS_M1
          RLC.csv";

  WriteCSVfile();
}   //end method structuresBindingNavigatorSaveItem_Click

//assign output file name and location. Then, call WriteCSVfile function
private void SParamBindingNavigatorSaveItem_Click(object sender, EventArgs e)
{
  //check structure value and save to appropriate file
  if (structure == "GSSG")
    CSVfile = @"\\psf\Home\Documents\Visual Studio 2013\Projects\DataExtraction\GSSG
          SParam.csv";
  else if (structure == "GSSG M1")
    CSVfile = @"\\psf\Home\Documents\Visual Studio 2013\Projects\DataExtraction\GSSG_M1
          SParam.csv";
  else if (structure == "SS")
    CSVfile = @"\\psf\Home\Documents\Visual Studio 2013\Projects\DataExtraction\SS
          SParam.csv";
  else
    CSVfile = @"\\psf\Home\Documents\Visual Studio 2013\Projects\DataExtraction\SS_M1
          SParam.csv";

  WriteCSVfile();
}   //end method SParamBindingNavigatorSaveItem_Click

//write values to .csv file
private void WriteCSVfile()
{
  int countColumn;

  try
  {
    //open stream writer object
    System.IO.StreamWriter csvFileWriter = new StreamWriter(CSVfile, false);

    //hold column header
    string columnHeaderText = "";

    //get column count of correct data table
    if (CSVfile.Contains("RLC"))
      countColumn = structuresDataGridView.ColumnCount - 1;
```

```
else
   countColumn = SParamdataGridView.ColumnCount - 1;

//if first column, save as header
if (countColumn >= 0)
{
   if (CSVfile.Contains("RLC"))
      columnHeaderText = structuresDataGridView.Columns[0].HeaderText;
   else
      columnHeaderText = SParamdataGridView.Columns[0].HeaderText;
}

//append remaining column values of this row to the column header variable
for (int i = 1; i <= countColumn; i++)
{
   if (CSVfile.Contains("RLC"))
      columnHeaderText = columnHeaderText + ',' +
                  structuresDataGridView.Columns[i].HeaderText;
   else
      columnHeaderText = columnHeaderText + ',' +
                  SParamdataGridView.Columns[i].HeaderText; ;
}

//write column header to file
csvFileWriter.WriteLine(columnHeaderText);

if (CSVfile.Contains("RLC"))   //if writing RLC data, copy data from RLC data grid view
{
   //for each row in the data grid
   foreach (DataGridViewRow dataRowObject in structuresDataGridView.Rows)
   {
      //if New Row
      if (!dataRowObject.IsNewRow)
      {
         string dataFromGrid = "";

         dataFromGrid = dataRowObject.Cells[0].Value.ToString();

         for (int i = 1; i <= countColumn; i++)
         {
            dataFromGrid = dataFromGrid + ',' + dataRowObject.Cells[i].Value.ToString();
         }

         csvFileWriter.WriteLine(dataFromGrid);
      }
   }
}
else    //if writing S-Param data, copy data from SParam data grid view
{
   foreach (DataGridViewRow dataRowObject in SParamdataGridView.Rows)
   {
```

```
                  //if New Row
                  if (!dataRowObject.IsNewRow)
                  {
                     string dataFromGrid = "";

                     dataFromGrid = dataRowObject.Cells[0].Value.ToString();

                     for (int i = 1; i <= countColumn; i++)
                     {
                        dataFromGrid = dataFromGrid + ',' + dataRowObject.Cells[i].Value.ToString();
                     }

                     csvFileWriter.WriteLine(dataFromGrid);
                  }
               }
            }

            csvFileWriter.Flush();
            csvFileWriter.Close();
         }
         catch (Exception exceptionObject)
         {
            MessageBox.Show(exceptionObject.ToString());
         }
      }//end method WriteCSVfile
   }
}
```