ON-CHIP IMPEDANCE TRANSFORMATIONS FOR A STANDARD CMOS PROCESS

by

Charles Edward Greene

Bachelor of Science in Electrical Engineering, University of Pittsburgh, 2001

Submitted to the Graduate Faculty of

the School of Engineering in partial fulfillment

of the requirements for the degree of

Master of Science in Electrical Engineering

University of Pittsburgh

2002

UNIVERSITY OF PITTSBURGH

SCHOOL OF ENGINEERING

This thesis was presented

by

Charles Edward Greene

It was defended on

December 10, 2002

and approved by

Ronald G. Hoelzeman, Associate Professor, Electrical Engineering

James T. Cain, Professor, Electrical Engineering

Thesis Advisor: Marlin H. Mickle, Nickolas A. DeCecco Professor, Electrical Engineering

ABSTRACT

ON-CHIP IMPEDANCE TRANSFORMATIONS FOR A STANDARD CMOS PROCESS

Charles Edward Greene, MS

University of Pittsburgh, 2002

On-chip impedance matching has become a major focus as companies and institutions move closer to a complete System on a Chip (SoC). With limited design area, it is important to obtain maximum power transfer to the required load. This research presents commonly used impedance matching techniques and extends them to include on-chip networks. These networks have inherent problems caused by the common substrate. It will be shown that the resulting parasitics can be calculated to allow analysis and manipulation of the overall design. It will also be demonstrated that the use of on-chip inductors will cause severe mismatch and loss due to their low quality factors. Finally, test networks will be fabricated in a 1.5-micron process to show the validity of the concepts presented.

ACKNOWLEDGEMENTS

This research was supported in part by the Pittsburgh Digital Greenhouse through the project “A Stand Alone RF Powered SOC” and in part by the Department of Education through the GAANN fellowship.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

1.0 INTRODUCTION

1.1 Background Problem

Current Radio-Frequency Identification (RFID) tags require the use of both receiving circuitry and an external antenna. The recent objective has been to create a System on a Chip (SoC) by integrating the antenna along with the required circuitry. This poses a problem when the antenna is connected to the driving amplifier. In standard systems, the output of the amplifier and the impedance of the antenna were designed to be 50 ohms. The connection of the two independent parts could then be done using standard 50-ohm coaxial cable or a microstrip transmission line. An on-chip antenna does not necessarily have an impedance of 50 ohms, and there is no need to use coaxial cable. For these reasons, the impedance does not need to be 50 ohms. The amplifier and antenna can be designed to optimize their on-chip performance without having to worry about fixed impedance values.

To obtain the maximum power transfer to the antenna for transmitting, an impedance matching network must be inserted between the two components. These networks, however, have inherent parasitic effects caused by the grounded substrate, which means they cannot simply be designed using standard methods.

1.2 Initial Overview

To better understand the problems with on-chip impedance matching, the available narrowband networks will be examined in detail. These topologies will be manipulated to obtain multiple networks of inductors and capacitors that can be used to match the source and load impedances. Once a network is obtained, the architecture of the components must be taken into consideration to help minimize the effects of the substrate. This must be done to insure the network performs in a desirable way.

In the following sections, the matching elements are analyzed in order to fabricate test networks with robust performance. An impedance matching program will also be presented to help characterize the matching networks. The program will be designed to include the parasitic effects of the substrate and inherent conductor losses to determine whether the network

consumes more power than it can deliver to the load. Finally, test networks will be fabricated on an analog CMOS process to see if the characterization process has been successful.

2.0 IMPEDANCE MATCHING BACKGROUND

2.1 Theory of Maximum Power Transfer

As devices become smaller, generally batteries and other energy supplying sources are also scaled in size. One way to deal with the decrease in available power is to conserve AC energy by minimizing reflections. Reflections are caused by unequal output and input impedances like the ones seen in Figure 2.1.

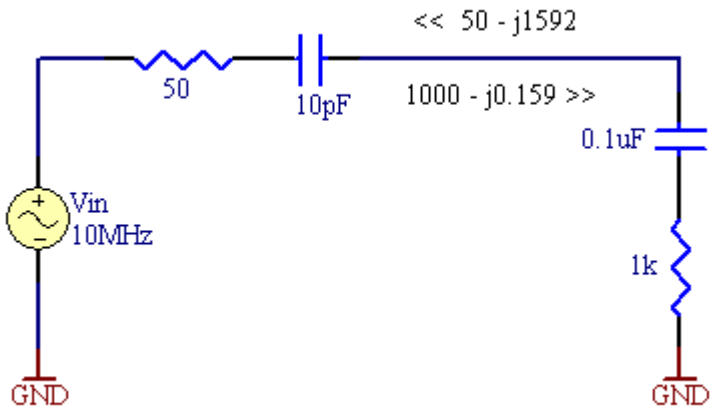


Figure 2.1 – Unequal Output and Input Impedances at 10MHz

The Maximum Power Transfer Theorem states that the maximum power is received at the load when the output impedance of the driving stage is the complex conjugate of the load impedance [1]. This is best illustrated through an example. The output power for the circuit seen in Figure 2.2 was calculated for different load impedances. The results can be seen in Table 2.1. Figure 2.3 shows how changes in the load resistance and reactance affect the output power.

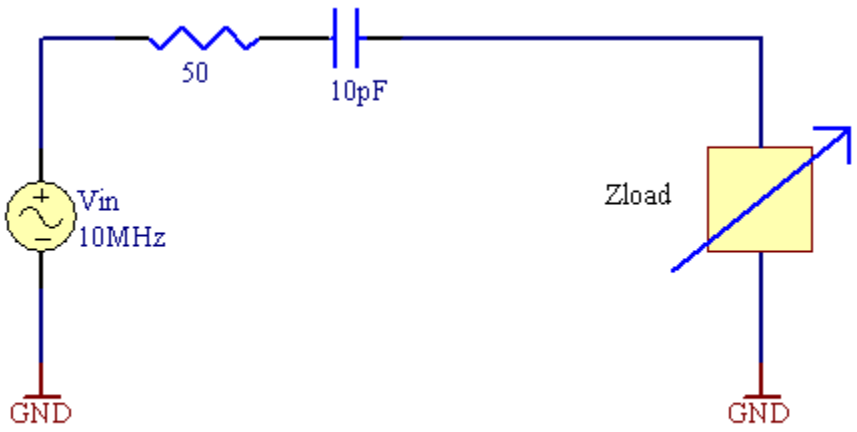


Figure 2.2 – Circuit Diagram with Variable Load

Table 2.1 – RMS Power Delivered to the Load Resistor for Different Loads with Vin=1V Peak

Load Impedance (Ω)	Power in Load Resistor (mW), Vin = 1V
1000-j0.1592	0.1375
1000+j0.1592	0.1375
500-j15.92	0.0866
500+j15.92	0.0904
200-j159.2	0.0320
200+j159.2	0.0473
100-j1592	0.0049
100+j1592	2.2222
50-j1592	0.0025
50+j1592 **(complex conjugate)**	**2.5**

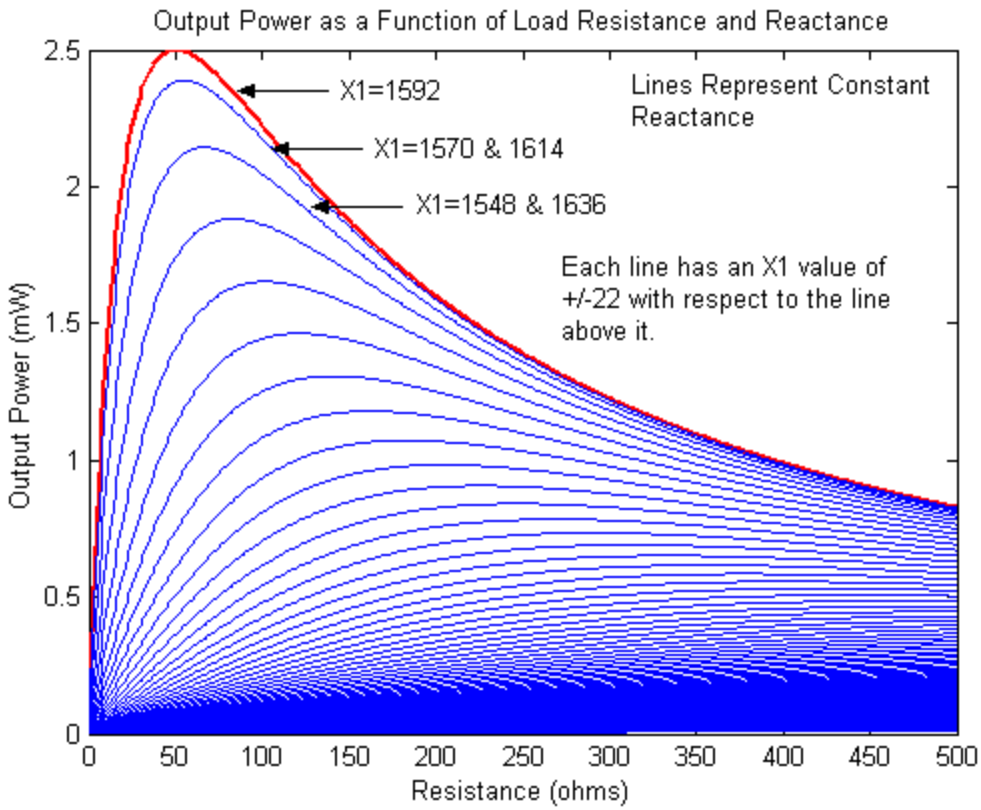


Figure 2.3 – Output Power (RMS) with Reactive Conjugate in Red

The Matlab code used to produce the preceding figure along with Figures 2.7 & 2.8 can be seen in Appendix A.

As Table 2.1 and Figure 2.3 show, the maximum load power is achieved with the complex conjugate pair. This can be seen by examining the equivalent impedance of the circuit.

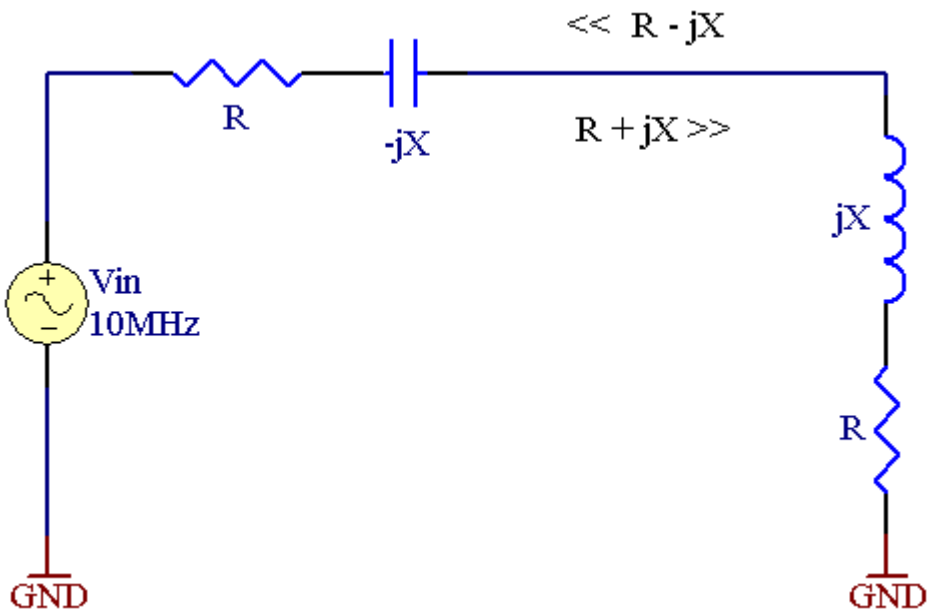


Figure 2.4 – Circuit Diagram Zout=Zload*

$$Z_{EQ} = R - jX + R + jX = 2R \qquad (2.1)$$

$$I = \frac{V_{IN}}{Z_{EQ}} = \frac{V_{IN}\angle 0^{o}}{2R} = \frac{V_{IN}}{2R}\angle 0^{o} \qquad (2.2)$$

As seen in Equation 2.1, the equivalent impedance is totally resistive because the reactive components have canceled. The absence of a reactive element guarantees that the voltage and current will be in phase, as Equation 2.2 shows. Figure 2.5 shows the difference in instantaneous power for in-phase and out-of-phase voltages and currents.

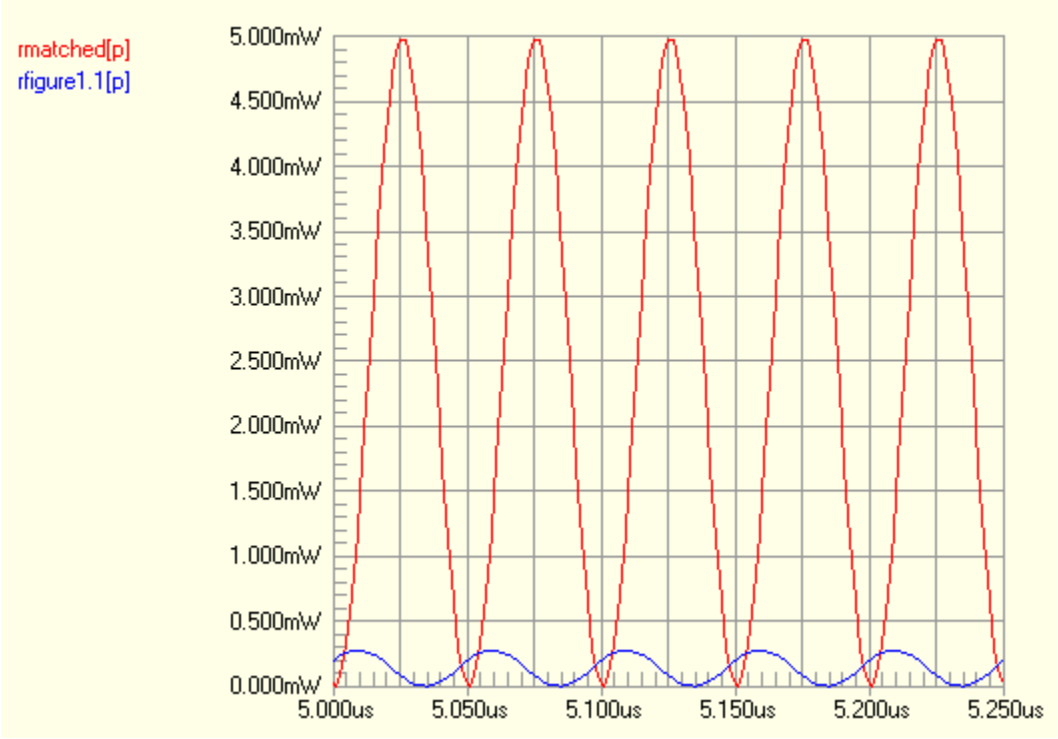


Figure 2.5 – Instantaneous Power for the Complex Conjugate Load and the Load Given in Figure 2.1

This graph again shows that the maximum power is achieved at the load when the load impedance is the complex conjugate of the output impedance of the driving stage.

2.2 Theory of Maximum Voltage Transfer

To obtain the maximum transferred voltage, the load resistance must be much greater than the output resistance of the driving stage [2]. The output voltage was calculated for the circuit seen in Figure 2.6. The results are displayed in Table 2.2.

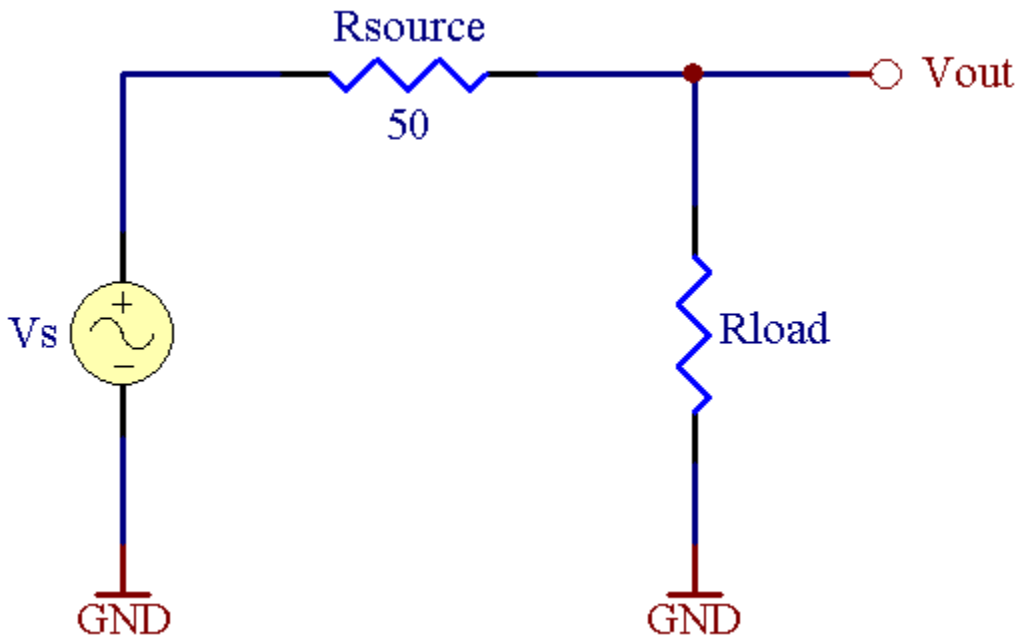


Figure 2.6 – Simple Resistive Circuit

Table 2.2 – Output Voltage for Different Resistor Values

Load Resistance (Ω)	Average Load Power (mW)	Peak Load Voltage (V) Vs=1V
5	0.826	0.091
25	2.222	0.333
50	2.5	0.500
100	2.222	0.667
1K	0.454	0.952
5K	0.098	0.990
10K	0.050	0.995
100K	0.005	0.9995
1M	0.0005	0.99995

As Table 2.2 shows, the output voltage asymptotically approaches the input voltage as the value of the output resistance is increased. This simple concept can be extended to complex output and input impedances. The following graph shows how variations in the load resistance and reactance affect the output voltage. The data were calculated using the circuit shown in Figure 2.2 with an input voltage of one-volt peak.

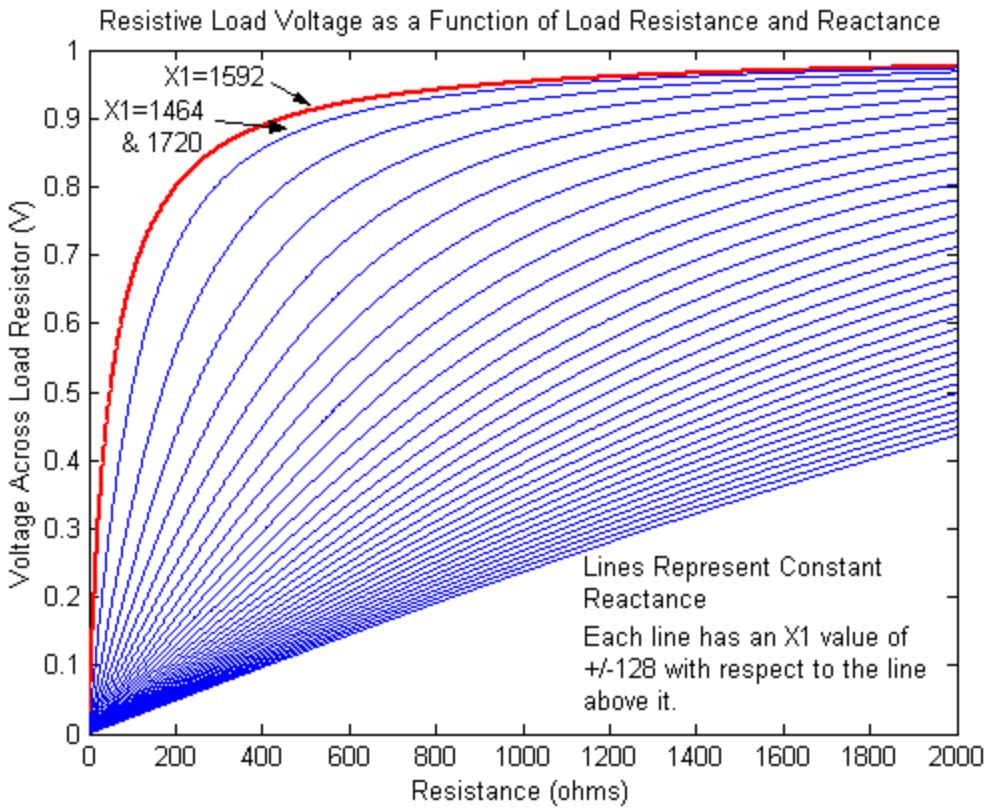


Figure 2.7 – Voltage Across Load Resistor for Circuit in Figure 2.2 with Reactive Conjugate in Red

It can be seen from Figure 2.7 that the voltage across the load resistor increases as its value is increased. It can also be seen that the maximum voltage occurs when the reactive part of the load is the conjugate of the reactive component of the source impedance. From this, it can be concluded that maximum voltage transfer is achieved when the load impedance meets the following criteria.

$$R_{LOAD} \gg R_{SOURCE}$$
$$X_{LOAD} = -X_{SOURCE}$$

It should be noted that this analysis was done using the voltage across the load resistor. In cases where the voltage across the entire load is desired, the maximum occurs when the resistive part is zero and the reactive element is the conjugate of the source reactance. This is illustrated by Figure 2.8, which was calculated using Figure 2.2 with an input voltage of one-volt peak. The

maximum occurs at this point since it is a resonant circuit. Resonant circuits generally have a circuit Q-value, Q_C, that is significantly larger than one [2].

$$Q_C = \frac{X}{R_T} \qquad (2.3)$$

As Equation 2.3 shows, the total resistance in the circuit is inversely proportional to the circuit Q-value. By making the load resistance as small as possible, the circuit Q-value is maximized. A high valued Q_C guarantees a large and narrow peak around the resonant frequency.

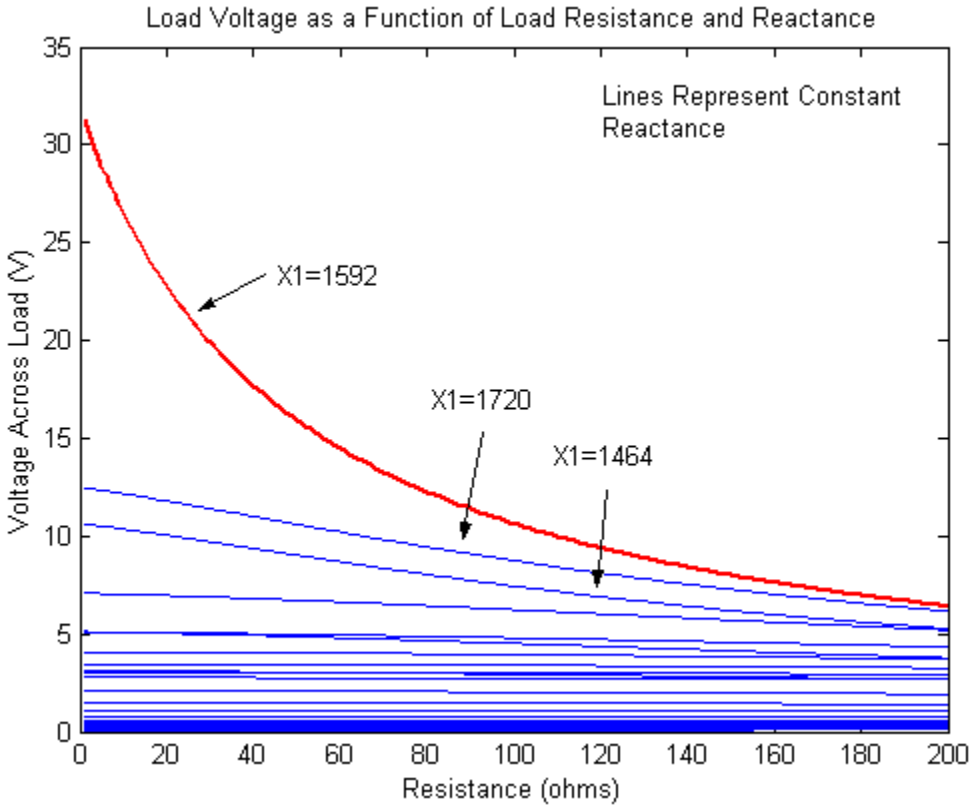


Figure 2.8 – Output Voltage with Reactive Conjugate in Red

To obtain the maximum voltage transfer to the entire load, the criteria become the following.

$$R_{LOAD} \ll R_{SOURCE}$$
$$X_{LOAD} = -X_{SOURCE}$$

2.3 Impedance Conversions

The input or output impedance is not always given or desired in the series format. The figures and equations below show a simple way of converting between the series and parallel forms [2].

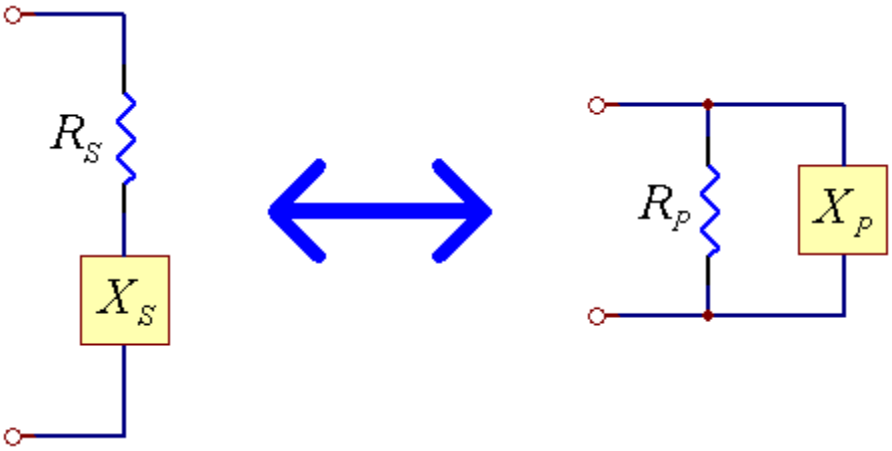


Figure 2.9 – Series and Parallel Conversions

$$R_P = \frac{R_S^2 + X_S^2}{R_S} \tag{2.4}$$

$$X_P = \frac{R_S^2 + X_S^2}{X_S} \tag{2.5}$$

$$R_S = \frac{R_P \cdot X_P^2}{R_P^2 + X_P^2} \tag{2.6}$$

$$X_S = \frac{R_P^2 \cdot X_P}{R_P^2 + X_P^2} \tag{2.7}$$

Another useful impedance transformation is between the wye and delta formats. The equations for the Y – Δ and the Δ – Y conversions are given below along with there circuit diagrams [2].

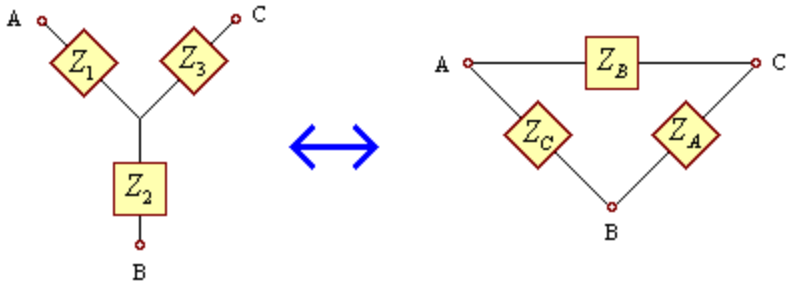


Figure 2.10 – Y and Δ Circuit Diagrams

$$Z_A = \frac{Z_1 \cdot Z_2 + Z_1 \cdot Z_3 + Z_2 \cdot Z_3}{Z_1} \qquad (2.8)$$

$$Z_B = \frac{Z_1 \cdot Z_2 + Z_1 \cdot Z_3 + Z_2 \cdot Z_3}{Z_2} \qquad (2.9)$$

$$Z_C = \frac{Z_1 \cdot Z_2 + Z_1 \cdot Z_3 + Z_2 \cdot Z_3}{Z_3} \qquad (2.10)$$

$$Z_1 = \frac{Z_B \cdot Z_C}{Z_A + Z_B + Z_C} \qquad (2.11)$$

$$Z_2 = \frac{Z_A \cdot Z_C}{Z_A + Z_B + Z_C} \qquad (2.12)$$

$$Z_3 = \frac{Z_A \cdot Z_B}{Z_A + Z_B + Z_C} \qquad (2.13)$$

2.4 Statement of the Problem

The problem to be addressed is the optimum matching of impedance between a circuit output and the input of a second circuit. Both of the circuits and the matching elements will be fabricated on an analog or analog/digital CMOS die.

Initially, a review of the suitable networks for impedance matching will be presented to determine the best topology with the minimal complexity. Using these candidate topologies and additional computed data, the effects of the parasitics and component tolerances will be examined. In order to simplify this design procedure, a program will be developed to determine the behavior of a specified network taking into account the parasitic effects of the common substrate along with inherent conductor losses for a given process.

The inductors and capacitors used in these networks may be laid out in a variety of fashions to obtain the desired values. These layouts, however, may not be suitable for the precise function of impedance matching. Thus, candidate layouts for both inductors and capacitors will be presented to simplify the layout process.

Specific example matching problems will be analyzed and designed to demonstrate the techniques and challenges involved in practical matching situations. These examples will be laid out and fabricated on a CMOS die to show the results. The CMOS die will then be tested, and the results will be included as a part of this research. Finally, the results will be summarized along with recommendations for future research.

3.0 IMPEDANCE MATCHING NETWORKS

3.1 L – Network

The L – Network is one of the simplest impedance matching networks. It contains two elements with the possibility of two different configurations. The figure below shows the two L – Network matching circuits.

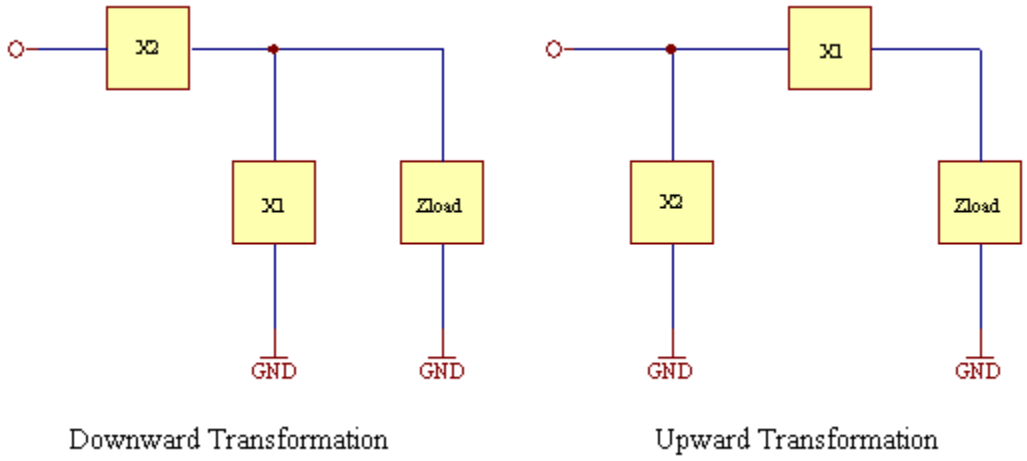


Figure 3.1 – Two L – Network Transformations

These two networks can be used to transform any load impedance into the complex conjugate of the driving output impedance. The position of the element closest to the load determines whether the resistive transformation is upward or downward. In general, the terms upward and downward are used to describe the equivalent resistance of the network compared to the original load resistance. If the first element is in parallel with the load, the transformation is downward. If it is in series, an upward transformation is performed [3]. Given this information, it is easy to see that this element is responsible only for the resistance transformation. The element closest to the source or driver is used to cancel out or add reactance to that added by the resistive transforming element.

Now that the elements of the typical matching networks are understood, the equations used to derive their values must be presented. There will be two sets of equations presented. These sets will correspond to the downward and the upward transformations.

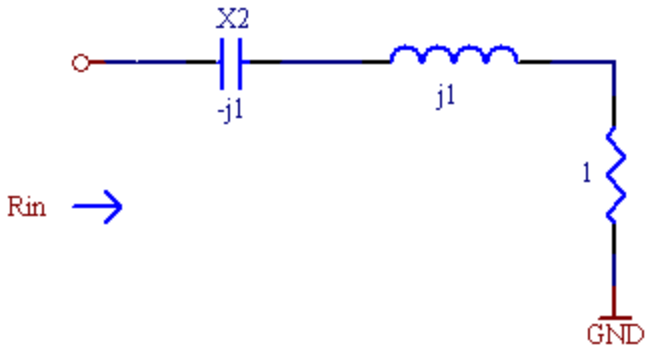
3.1.1 Downward Transformation

A downward transformation is required if the load resistance is greater than the resistive component of the source impedance. This comparison can be done with both the source and load impedances in the series or parallel format. It can also be done with one impedance in the series format while the other is in its parallel format. For small resistive transformations, both downward and upward transformations will give the desired impedance. This special case occurs when the steps outlined in both the downward and upward transformation sections yield a noncomplex value for Q. For further explanation of this case, refer to the example in Appendix B. As explained above, the downward transformation utilizes the left hand circuit seen in Figure 3.1.

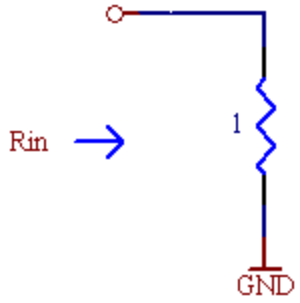
The first step in any impedance transformation is to manipulate the circuit to obtain the most desirable configuration. A desirable configuration can be defined as the circuit form that allows simple visual inspection along with uncomplicated equation derivation. For the downward transformation, it is more straightforward to have the load in its parallel format and the desired impedance (complex conjugate of the driving stage) in a series representation. This is true because the first element in the matching network is in parallel with the load while the second element is in series with the source impedance. These impedances can be quickly transformed to their parallel or series equivalent using the methods outlined in Section 2.3.

Now that the circuit is in the proper format, it can be transformed to the necessary impedance. The following figure shows the circuit representations from the beginning to the end of the transforming process.

Transform a two-ohm resistor to a resistance of one ohm:



Complete network that will be simplified below:



Resistor and inductor have been converted to their series form using the method outlined in Section 2.3:

Adding the inductor and capacitor cancels the reactive component leaving the transformed resistance:

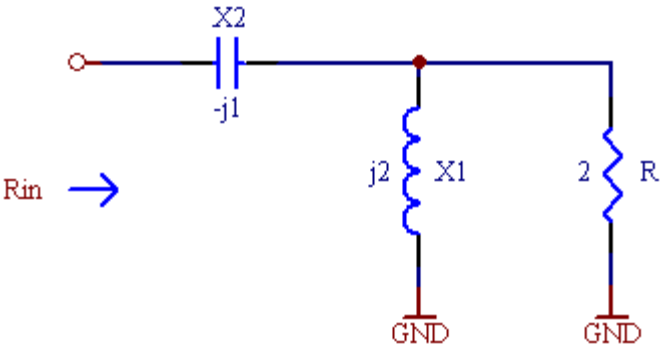


Figure 3.2 – Example of a Complete Downward Transformation

The transformation in Figure 3.2 can be realized using the following equations [3]. The variables R and X represent the starting impedance or load while R_{IN} and X_{IN} correspond to the transformed impedance or complex conjugate of the driving stage. As stated before, the load is desired in the parallel format. If the load is in the series format, it must first be converted to its parallel equivalent. If R and X are taken from the series circuit, these equations will yield incorrect results.

$$Q = \pm\sqrt{\frac{R}{R_{IN}} - 1} \tag{3.1}$$

$$X_1 = \frac{X \cdot R}{Q \cdot X - R} \tag{3.2}$$

$$X_2 = X_{IN} - Q \cdot R_{IN} \tag{3.3}$$

When X or X_{IN} are not present in the problem, as with the previous example, the equations defined above must be modified. Since X and R are in the parallel format for the downward transformation, the absence of X corresponds to an infinite reactance. The input elements, X_{IN} and R_{IN}, are in the series format making the absence of X_{IN} appear as a short or zero reactance. For these situations, the following equations should be substituted where needed for those described previously in this subsection.

For $X \rightarrow \infty$:

$$X_1 = \frac{R}{Q} \tag{3.4}$$

For $X_{IN} = 0$:

$$X_2 = -Q \cdot R_{IN} \tag{3.5}$$

As seen in Equation 3.1, the value of Q can be positive or negative. Its magnitude describes the bandwidth of the matching network. Larger values of Q produce smaller bandwidths while small Q-values enable wider bands. As a good approximation, the Q of the overall circuit is one half the magnitude of the Q-value for the L – Network [3]. The value of the overall Q as a function

of bandwidth is described by the following equation where f_o is the center frequency and BW is the bandwidth. This equation becomes a better approximation as the value of Q increases [2].

$$Q = \frac{f_o}{BW} \tag{3.6}$$

For a given bandwidth, there are two possible matching networks. One will have a positive Q while the other will be negative. This allows the choice between two networks with different components. The network with the most desirable characteristics can be chosen for the specified application. These characteristics include component values and element parasitics, which will be explained in Section 4.3. This can be applied to the network in Figure 3.2, which was done with a positive Q-value. The network seen in Figure 3.3 will also perform the required transform by using a negative Q-value.

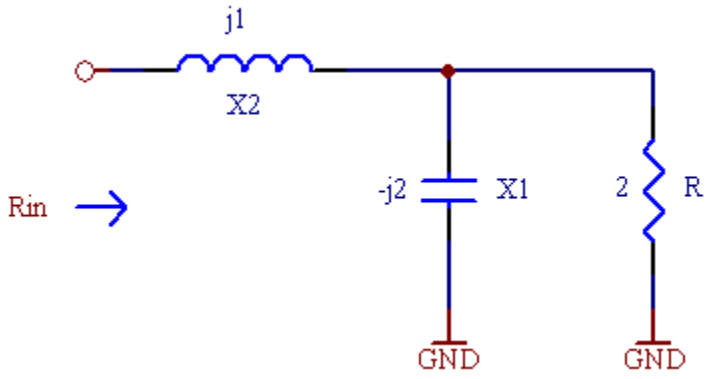


Figure 3.3 – Previous Example Using a Negative Q-value

An example of this complete process can be seen in the following numerical problem.

Example 1:

A source has an output impedance corresponding to a 50 ohm resistance in series with a 10pF capacitor. Match this output impedance to a load with a 250-ohm resistor in series with a 1nH inductor at a frequency of 1MHz. The circuit diagram can be seen below.

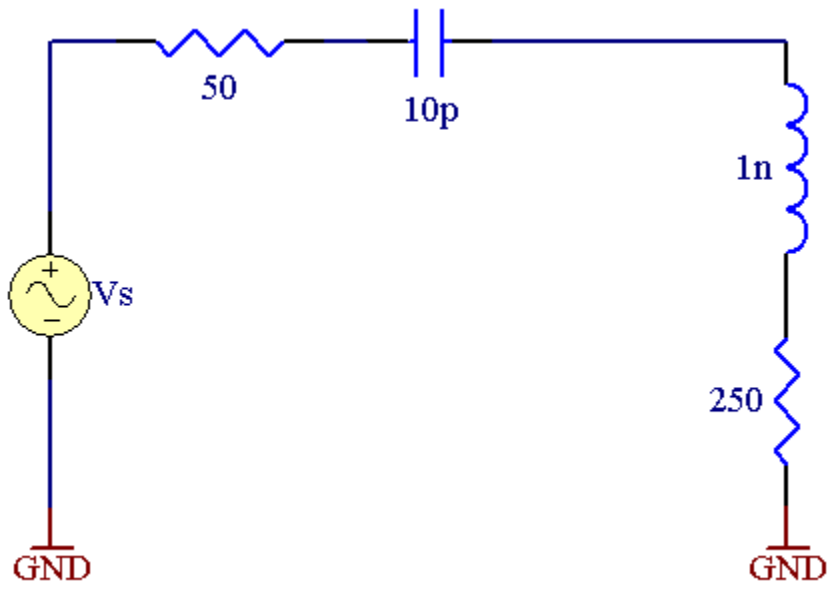


Figure 3.4 – Circuit Diagram

Solution:

For this example, the load resistor is significantly greater than the source impedance, which means a downward transformation is required. The first step is to calculate the reactive terms in ohms.

$$10pF \rightarrow \frac{-1}{\omega C} = \frac{-1}{2 \cdot \pi \cdot f \cdot C} = \frac{-1}{2 \cdot \pi \cdot 1M \cdot 10p} = -15.915K\Omega$$

$$1nH \rightarrow \omega L = 2 \cdot \pi \cdot f \cdot L = 2 \cdot \pi \cdot 1M \cdot 1n = 6.283m\Omega$$

Next, it is important to have the impedances in the proper format. For the downward transformation, the source impedance is required in the series format while the load should be in the parallel form. Since the source impedance is already in the series format, only the load must be transformed. Using the equations from Section 2.3:

$$R_S = 250\Omega$$
$$X_S = 6.283m\Omega$$
$$R_P = \frac{R_S^2 + X_S^2}{R_S} = 250\Omega$$
$$X_P = \frac{R_S^2 + X_S^2}{X_S} = 9.947M\Omega$$

Using the load in its parallel form, the variables for this example become the following values.

$$R = 250\Omega$$
$$X = 9.947M\Omega$$
$$R_{IN} = 50\Omega$$
$$X_{IN} = 15.915K\Omega \leftarrow Complex\ Conjugate$$

Using the equations in this subsection, the following matching elements were calculated.

$$Q = \pm\sqrt{\frac{R}{R_{IN}} - 1} = \pm\sqrt{\frac{250}{50} - 1} = \pm 2$$

Choosing a positive Q-value yields:

$$X_1 = \frac{X \cdot R}{Q \cdot X - R} = 125\Omega$$

$$X_2 = X_{IN} - Q \cdot R_{IN} = 15915 - 2 \cdot 50 = 15.815K\Omega$$

At 1MHz, these reactances are realized by the following components.

$$X_1 \rightarrow 19.894\mu H$$
$$X_2 \rightarrow 2.517mH$$

The completed circuit can be seen in the following figure.

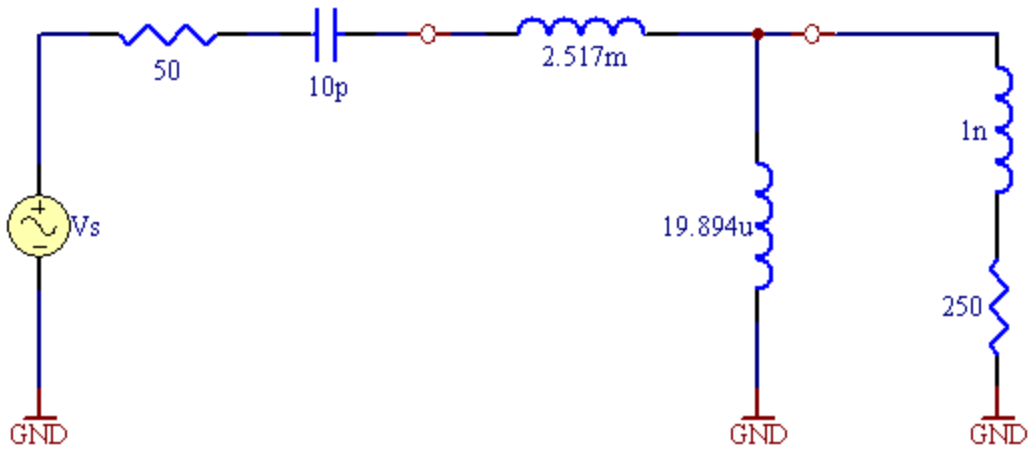


Figure 3.5 – Completed Circuit with Matching Network

For specific applications like on-chip design, a 2.517mH inductor is not realizable. Here, it is assumed that the analog circuit designer using the results of this research will be familiar with what values of inductance, capacitance, and resistance can be fabricated. Thus, any alternative designs will be evaluated in the same manner as illustrated here.

To correct the problem of the large valued inductor, the L – Network with a negative Q-value can be calculated for comparison. In this case, however, the network with a negative Q-value has a similar component value, which is depicted in the values below.

$$X_1 = -124.998 \rightarrow 1.273nF$$
$$X_2 = 16015 \rightarrow 2.549mH$$

This network is still unacceptable for on-chip design. Because these two networks are the only available L – Networks, there will be a need to use a three-element matching network to produce acceptable values. These alternative topologies are covered in Sections 3.2 and 3.3.

3.1.2 Upward Transformation

An upward transformation is required if the load resistance is less than the output resistance of the driving stage. To accomplish the upward transformation, the right hand circuit seen in Figure 3.1 is used. Like the downward transformation, the upward transformation also has a more manageable configuration. In this case, the load is desired in the series format while the

desired impedance should be placed in the parallel form. These forms can be obtained using the methods presented in Section 2.3. The circuit diagrams for an upward transformation process can be seen in the following figure [3].

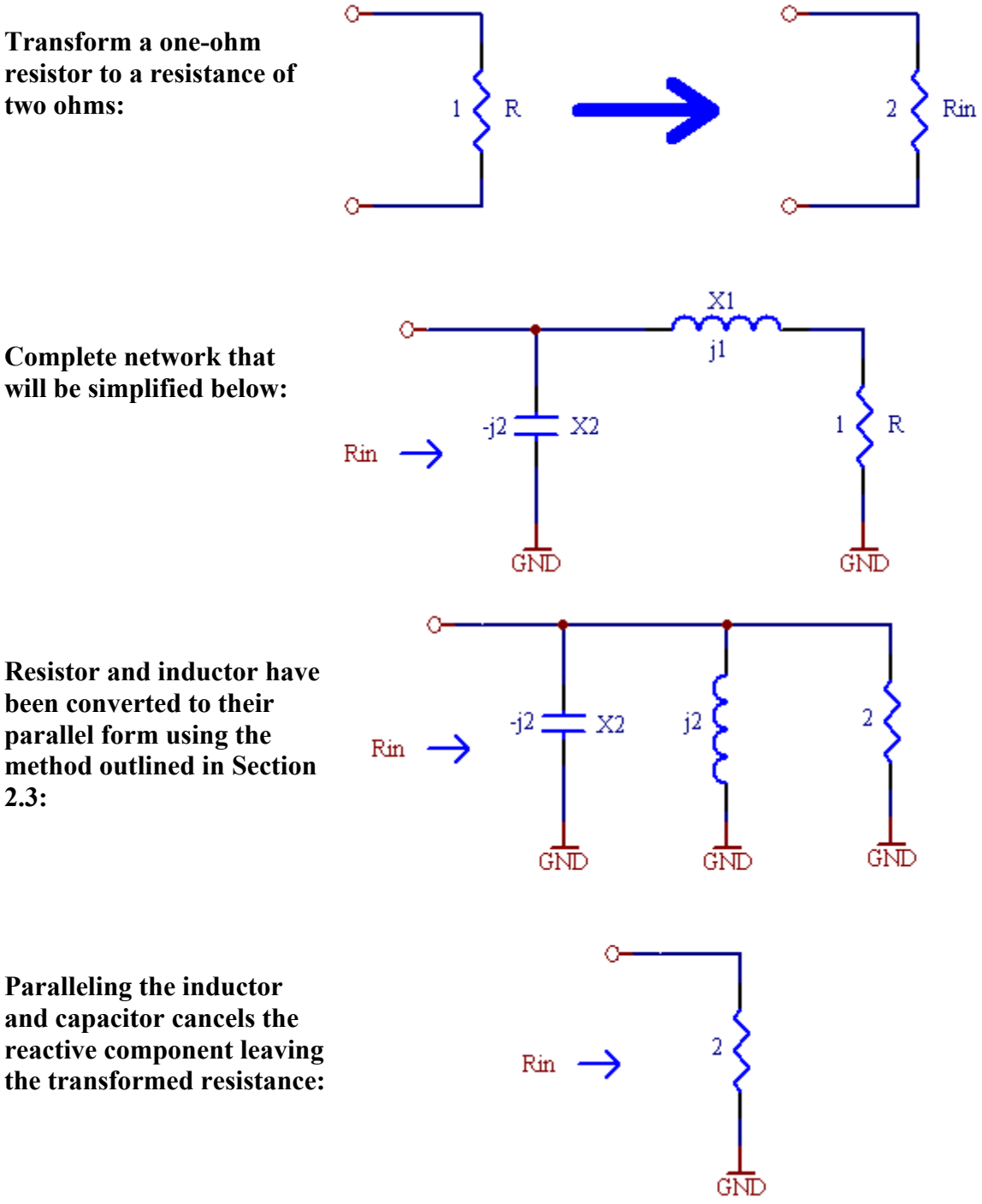


Figure 3.6 – Example of a Complete Upward Transformation

The circuit transformation in Figure 3.6 can be obtained using the following equations [3]. As previously explained, the variables R and X represent the starting impedance in its series format while R_{IN} and X_{IN} correspond to the transformed impedance in its parallel form.

$$Q = \pm\sqrt{\frac{R_{IN}}{R} - 1} \tag{3.7}$$

$$X_1 = Q \cdot R - X \tag{3.8}$$

$$X_2 = \frac{X_{IN} \cdot R_{IN}}{R_{IN} - Q \cdot X_{IN}} \tag{3.9}$$

As with the downward transformation, the absence of X or X_{IN} means the equations above need to be modified to reflect the missing element. If this is not done, Equation 3.9 will be completely invalid and yield some form of infinity over infinity. The upward transformation requires the load in the series format, which makes an absent X appear as a short or zero reactance. The desired impedance, on the other hand, is in the parallel format, which makes the lack of X_{IN} appear as an open circuit or infinite reactance. In either case, the following equations should be substituted where needed.

For X = 0:

$$X_1 = Q \cdot R \tag{3.10}$$

For $X_{IN} \rightarrow \infty$:

$$X_2 = -\frac{R_{IN}}{Q} \tag{3.11}$$

The complete L – Network upward transformation process can be seen in the following example.

Example 2:

Transform a load impedance of 40-j20 to match a source impedance of 200+j10. The circuit diagram can be seen below.

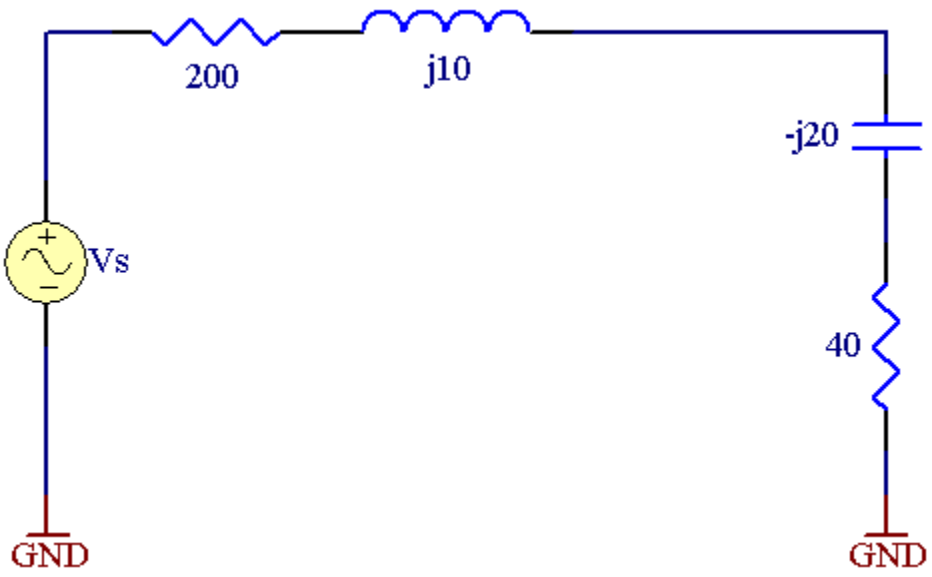


Figure 3.7 – Circuit Diagram

Solution:

The upward transformation requires the load impedance in the series format while the source impedance is in the parallel form. For the given example, the series source impedance becomes the following.

$$R_S = 200\Omega$$
$$X_S = 10\Omega$$
$$R_P = \frac{R_S^2 + X_S^2}{R_S} = 200.5\Omega$$
$$X_P = \frac{R_S^2 + X_S^2}{X_S} = 4.01K\Omega$$

Using the parallel source impedance, the values for this example become:

$$R = 40\Omega$$
$$X = -20\Omega$$
$$R_{IN} = 200.5\Omega$$
$$X_{IN} = -4.01K\Omega \leftarrow Complex\ Conjugate$$

Using the equations in this subsection, the following matching elements can be calculated.

$$Q = \pm\sqrt{\frac{R_{IN}}{R} - 1} = \pm\sqrt{\frac{200.5}{40} - 1} = \pm 2.003$$

Choosing a positive Q-value yields:

$$X_1 = Q \cdot R - X = 2.003 \cdot 40 + 20 = 100.12\Omega$$

$$X_2 = \frac{X_{IN} \cdot R_{IN}}{R_{IN} - Q \cdot X_{IN}} = \frac{-4.01K \cdot 200.5}{200.5 + 2.003 \cdot 4.01K} = -97.662\Omega$$

The matched circuit can be seen in Figure A.3.2.

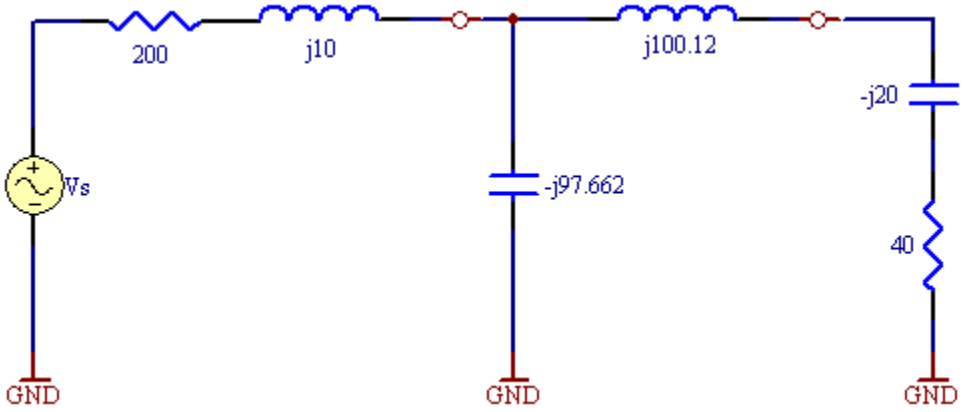


Figure 3.8 – Matched Circuit for Example 2

3.2 T – Network

The T – Network is a three-element impedance matching network. An example can be seen in Figure 3.9. It has one main advantage over the L – Network. The third element adds another degree of freedom in the form of a second Q variable. The additional Q-value means there will be an intermediate resistance, R', in the transformation process. Figure 3.10 illustrates the intermediate resistance with a simple graph [3]. The second resistive transformation allows a value to be chosen for one of the variables. This added flexibility allows one of the Q-values to be specified or more importantly for on-chip design, one of the network's components. This minimizes design time by allowing the use of an existing component layout.

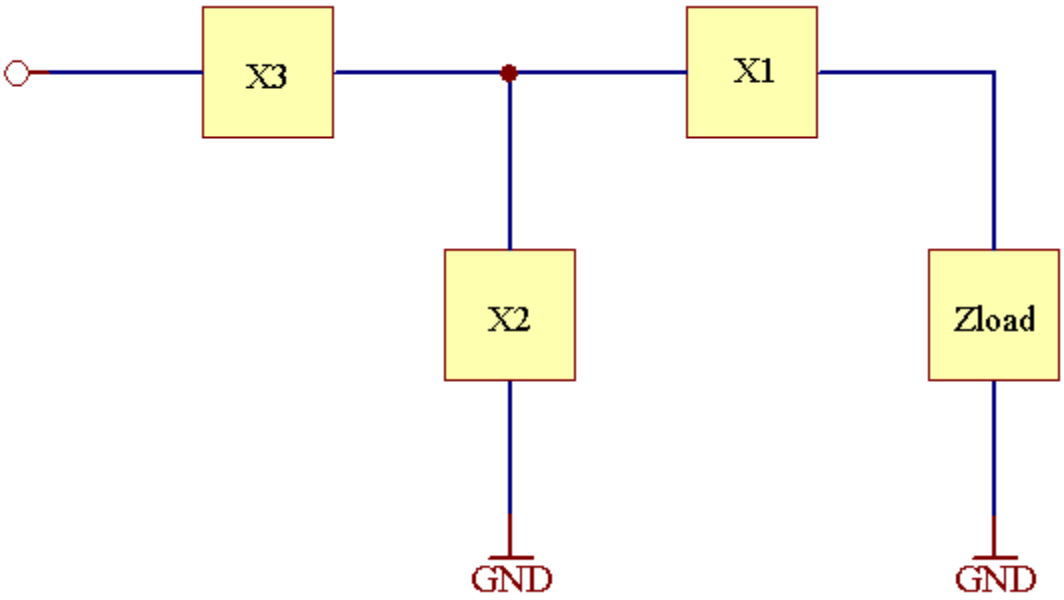


Figure 3.9 – Example of a T – Network

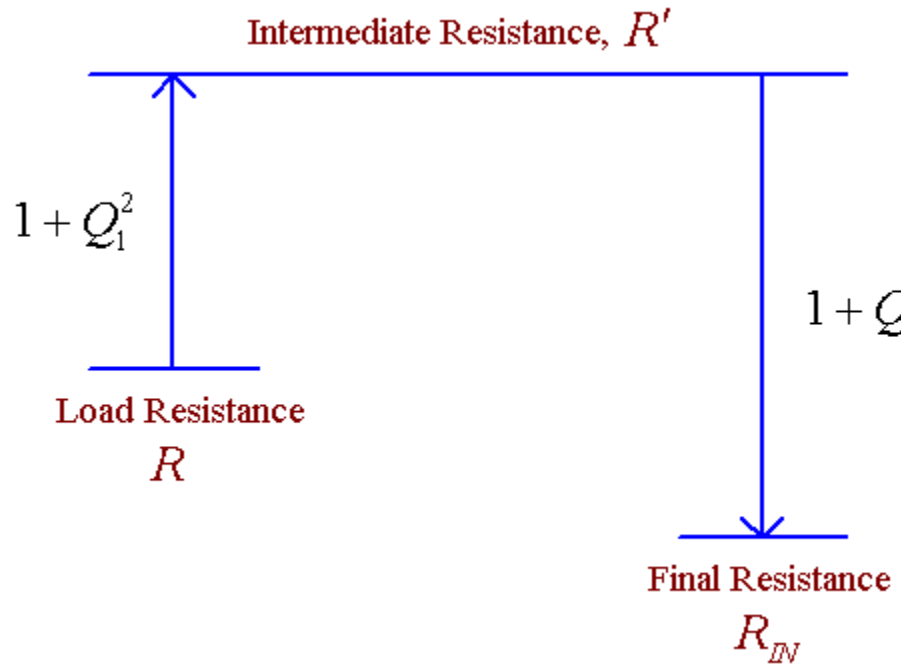


Figure 3.10 – Upward then Downward Resistive Transformation with a Net Downward Transformation

If the maximum Q-value is chosen, it must always be greater than or equal to the value of Q for the L – Network [4]. This can be explained by examining Figure 3.10. The load resistance is first transformed upward to a value dictated by Q_1. It is then transformed downward to the

required resistance using Q_2. By setting Q_1 equal to zero, the intermediate resistance will be the same as the load resistance. This means that only the downward transformation will occur, and X_1 will be the negative value of the load reactance. Since Q_1 is generally greater than zero, Q_2 must be greater than the Q of the L – Network. The same applies when setting the value of Q_2 equal to zero.

As with the L – Network, a good approximation of the overall circuit Q for the T – network is one half of the maximum value of $|Q_1|$ and $|Q_2|$. This approximation becomes more accurate as Q_1 and Q_2 become significantly different [3].

As explained for the L – Network, the Q-values can have positive and negative values. Since the T – Network has two Q-values, there will be four possible matching networks for a given bandwidth. Both of the values can be positive or negative, or one can be negative while the other is positive as shown in Table 3.1. This added flexibility allows the designer to choose the best network for the given application.

Table 3.1 – Network Combinations for a Given Bandwidth

Combinations	**$|Q_1| = X$, $|Q_2| = Y$**
Network 1	$Q_1 = X$, $Q_2 = Y$
Network 2	$Q_1 = X$, $Q_2 = -Y$
Network 3	$Q_1 = -X$, $Q_2 = Y$
Network 4	$Q_1 = -X$, $Q_2 = -Y$

Although the T – Network does have its advantages, it also has a drawback. The addition of the third element introduces more parasitics and another component with potential variations that degrade the performance of the overall network. This will be discussed in more detail in Section 4.3.

3.2.1 Downward Transformation

The downward transformation is derived with the output and load impedances in the series format. A method for switching between parallel and series format was outlined in Section 2.3.

The equations for the downward transformation are listed below [3]. Because there is one degree of freedom, a value must be assigned to one of the variables.

$$Q_1 = \pm\sqrt{\frac{R'}{R} - 1} \tag{3.12}$$

$$Q_2 = \pm\sqrt{\frac{R'}{R_{IN}} - 1} \tag{3.13}$$

$$X_1 = Q_1 \cdot R - X \tag{3.14}$$

$$X_2 = \frac{-R'}{Q_1 + Q_2} \tag{3.15}$$

$$X_3 = Q_2 \cdot R_{IN} + X_{IN} \tag{3.16}$$

Where R, X, R_{IN}, and X_{IN} are in the series form and $R' > R \,\&\, R_{IN}$.

There is a special case when the input or output reactance is not present. Because the input and output impedances are in the series format, this corresponds to a reactance of zero or a short circuit. The previous equations are still valid since the reactance term is zeroed. For these situations, the preceding equations simplify to the following forms.

For X = 0:

$$X_1 = Q_1 \cdot R \tag{3.17}$$

For $X_{IN} = 0$:

$$X_3 = Q_2 \cdot R_{IN} \tag{3.18}$$

An example of the T – Network downward transformation can be seen below.

Example 3:

It is desired to match an on-chip 200-ohm resistor in parallel with a 1pF capacitor to a 50-ohm source impedance at 915MHz. Choose a Q_2 value of one. Find the matching network with positive Q-values.

Solution:

Using the equations presented in this subsection, the subsequent values can be calculated.

$$Q_2 = 1$$
$$Q_1 = 0.401$$
$$R' = 100\Omega$$
$$X_1 = 133.597\Omega \rightarrow 23.238nH$$
$$X_2 = -71.362\Omega \rightarrow 2.437\,pF$$
$$X_3 = 50\Omega \rightarrow 8.697nH$$

3.2.2 Upward Transformation

The upward transformation is also performed with the source and load impedances in the series format. The equations for the upward transformation are the same as the downward equations (3.12-3.18).

While the downward resistive transformation is explained by Figure 3.10, the resistive transformation for the upward case is presented in the following figure [3].

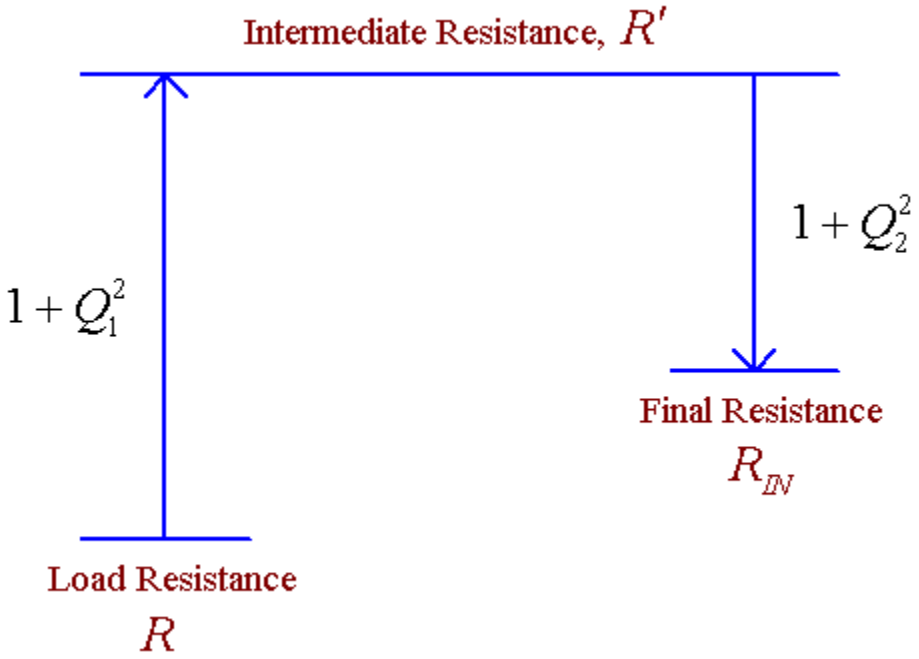


Figure 3.11 – Net Upward Resistive Transformation

The upward transformation process can be seen in the following numerical example.

Example 4:

Repeat Example 2 using a T – Network. Design the network so the magnitude of the maximum Q-value is equal to 10. Give all possible networks for this Q-value. Use the network with positive Q-values to show that the Π – Network (see Section 3.3) obtained from the Y – Δ conversion will have different values for Q_1 and Q_2.

Solution:

The T – Network requires the source and load impedances in the series format, which yields the following values.

$$R = 40\Omega$$
$$X = -20\Omega$$
$$R_{IN} = 200\Omega$$
$$X_{IN} = -10\Omega \leftarrow \textit{Complex Conjugate}$$

Figure 3.11 shows that Q_1 will have a larger value than Q_2 for the upward transformation. Using this graph,

$$Q_1 = 10$$

because the problem specifies the maximum Q-value.

Using equations 3.12 – 3.16,

$$R' = R \cdot (Q_1^2 + 1) = 40 \cdot (10^2 + 1) = 4.04K\Omega$$

$$Q_2 = \pm\sqrt{\frac{R'}{R_{IN}} - 1} = \pm\sqrt{\frac{4.04K}{200} - 1} = \pm 4.382$$

$$X_1 = Q_1 \cdot R - X = 10 \cdot 40 + 20 = 420\Omega$$

$$X_2 = \frac{-R'}{Q_1 + Q_2} = \frac{-4.04K}{10 + 4.382} = -280.907\Omega$$

$$X_3 = Q_2 \cdot R_{IN} + X_{IN} = 4.382 \cdot 200 - 10 = 866.4\Omega$$

This network and the other three possible networks for the given Q_1 value, can be seen in Figure 3.12.

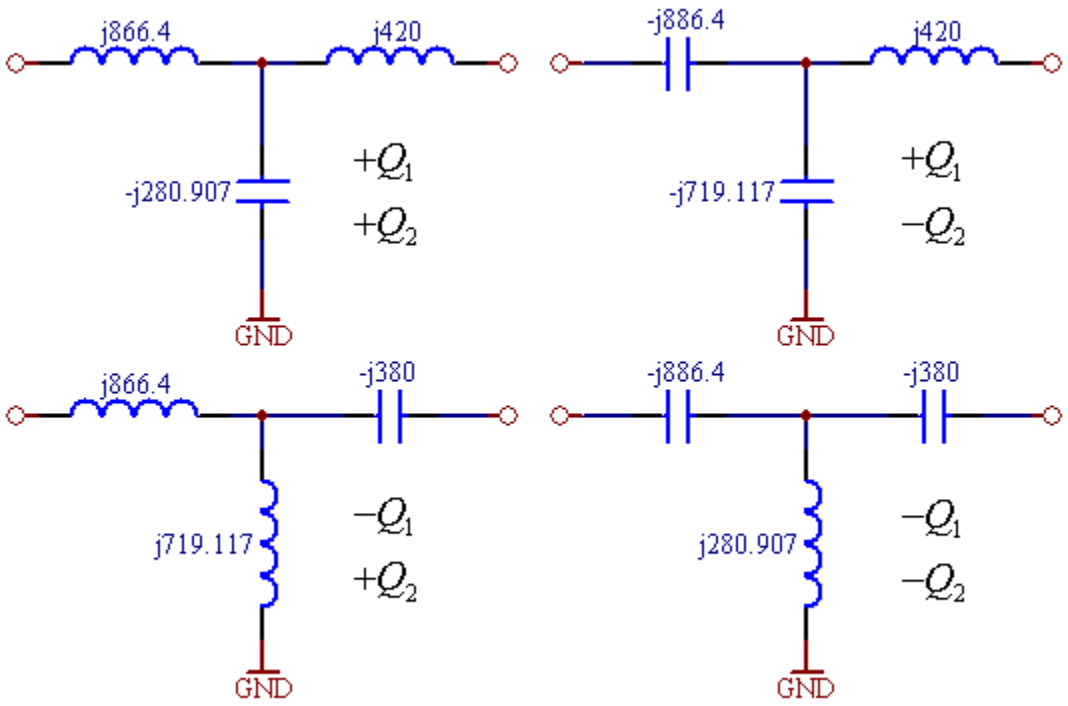


Figure 3.12 – Four Possible Networks for $|Q_1|=10$

The equivalent Π – Network for the one calculated previously was obtained using a Y – Δ transformation and can be seen in Figure 3.13.

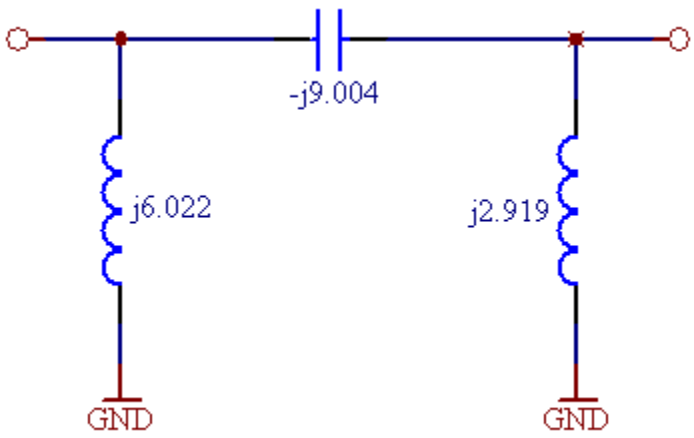


Figure 3.13 – Equivalent Π – Network

Using the equations outlined in Subsection 3.3.2, the values of Q_1 and Q_2 were calculated. The following table illustrates the results.

Table 3.2 – Comparison of T – and Π – Network Q-values

Network	Value of Q_1	Value of Q_2
Calculated T – Network	10	4.382
Converted Π – Network	-16.637	-33.360

The preceding table shows that the values of Q for a converted network do not match the original network. It should be advised that in applications where bandwidth must be specified specifically, the two networks, T and Π, need to be calculated independently to avoid potential problems.

3.3 Π – Network

The Π – Network is very similar to the T – Network. Figure 2.8 shows an example of a Π – Network. There are two ways to obtain this network. The T – Network can be calculated and then converted to the Π – Network using the Y – Δ transformation outlined in Section 2.3. The Δ – Y transformation works for calculating the T – Network from the Π – Network as well. In either case, the Q-values will not remain constant. For more details, refer to Example 4 in the preceding subsection. Also, the equations presented in the following subsections can be used to directly obtain the values for the Π – Network.

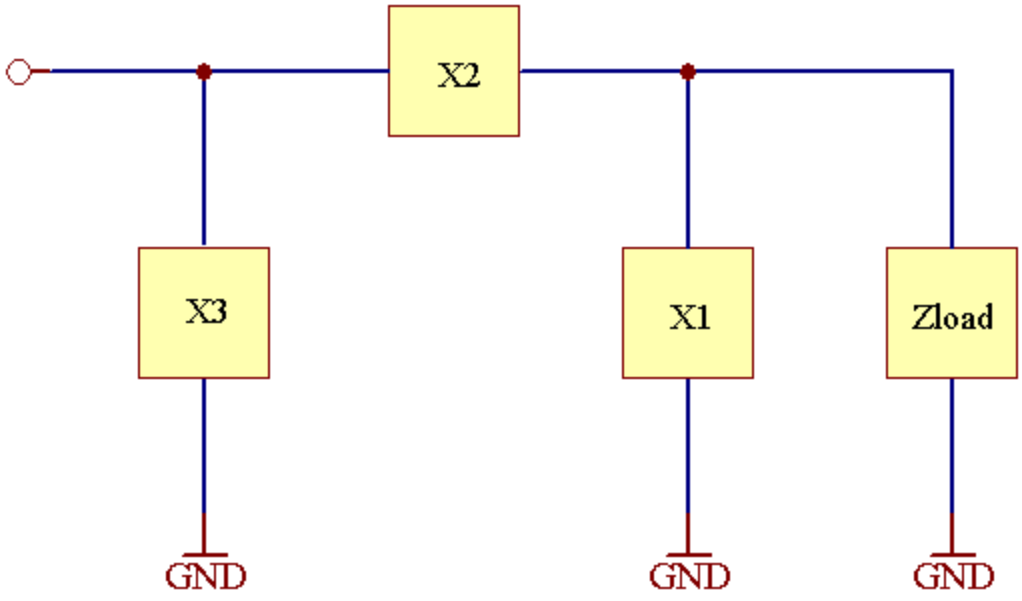


Figure 3.14 – Example of a Π – Network

As with the T – Network, the Π – Network must always have a maximum Q-value greater than or equal to that of the L – Network [3]. Also like the T – Network, the Π – Network has four possible network configurations for a given bandwidth. For more information, refer to Section 3.2.

The Π – Network has an advantage over the T – Network. It only contains one floating element while the T – Network has two. This means that the parasitic effects will be less for the Π – Network. This will be discussed in more detail in Section 4.3.

3.3.1 Downward Transformation

The downward transformation is performed with both the load and the output impedance of the driving stage in the parallel format. Again, the series to parallel conversion was explained in Section 2.3. As with the T – Network, the added degree of freedom requires one variable to be chosen. The downward transformation equations are given below [3].

$$Q_1 = \pm\sqrt{\frac{R}{R'} - 1} \tag{3.19}$$

$$Q_2 = \pm\sqrt{\frac{R_{IN}}{R'} - 1} \tag{3.20}$$

$$X_1 = \frac{-X \cdot R}{Q_1 \cdot X + R} \tag{3.21}$$

$$X_2 = R' \cdot (Q_1 + Q_2) \tag{3.22}$$

$$X_3 = \frac{X_{IN} \cdot R_{IN}}{R_{IN} - Q_2 \cdot X_{IN}} \tag{3.23}$$

Here, R, X, R_{IN}, and X_{IN} are in the parallel form and $R' < R \;\&\; R_{IN}$.

There is a special case when the input or output reactance is not present. Because the input and output impedances are in the parallel format, this corresponds to an infinite reactance. The

equations for X_1 and X_3 given previously are no longer valid since the equations yield some form of infinity in both the numerator and denominator. For these situations, the following equations should be substituted for those given above.

For $X \rightarrow \infty$:

$$X_1 = -\frac{R}{Q_1} \tag{3.24}$$

For $X_{IN} \rightarrow \infty$:

$$X_3 = -\frac{R_{IN}}{Q_2} \tag{3.25}$$

Unlike the resistive transformation for the T – Network, the Π – Network first transforms the resistance downward and then upward to the desired value. This can be seen in the following figure [3].

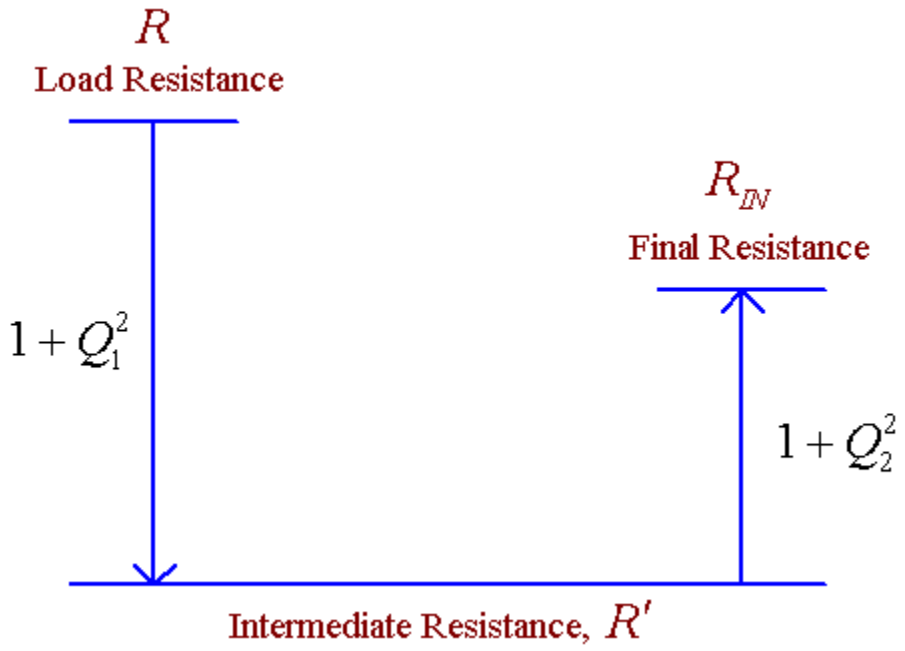


Figure 3.15 – Downward Resistive Transformation

This transformation is shown in detail by the problem presented in the following example.

Example 5:

It is desired to match an on-chip 200-ohm resistor in parallel with a 1pF capacitor to a 50-ohm source impedance at 915MHz. There is a 14.7nH inductor layout available. Let X_2 equal this value to minimize the layout time. Use a Monte Carlo sweep with component tolerances at ± 15% to show how the networks performance changes as its components change value. Assume that the load and source impedances are constant at the given values.

Solution:

Because the value of X_2 is given, the equations given in this subsection will need to be manipulated. Before this can be done, the impedances must be in the proper form. The Π – Network requires the source and load impedances in the parallel format. The load is specified in the parallel form, and the source impedance is in both the series and parallel form since there is no reactive term. Using Equations 3.19 – 3.25, the following values can be calculated.

$$X_2 = 14.7nH \rightarrow 84.5\Omega$$
$$R' = 20\Omega$$
$$Q_1 = \pm 3$$
$$Q_2 = \pm 1.225$$
$$X_1 = -108.098\Omega \rightarrow 1.61pF$$
$$X_3 = -40.816\Omega \rightarrow 4.26pF$$

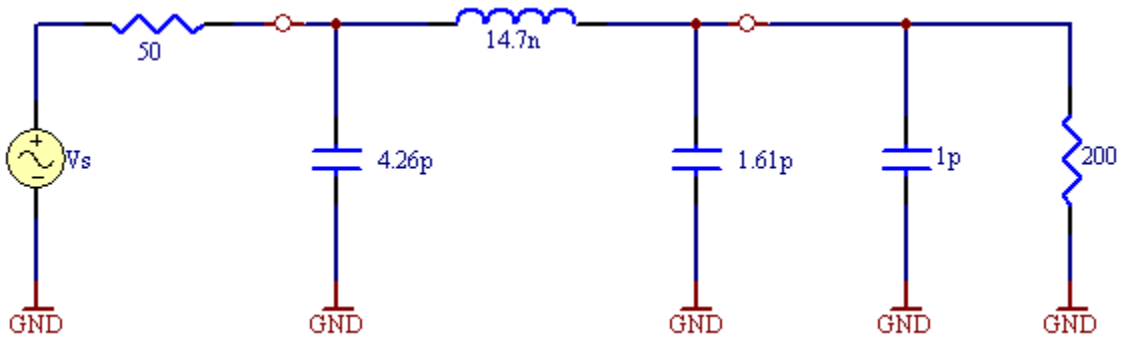


Figure 3.16 – Complete Circuit Diagram for Positive Q-values

The completed design using positive Q-values can be seen in Figure 3.16. The peak load power versus frequency was plotted in a Monte Carlo simulation to observe the effects of the component tolerances on the performance of the network. The results can be seen in the following graph.

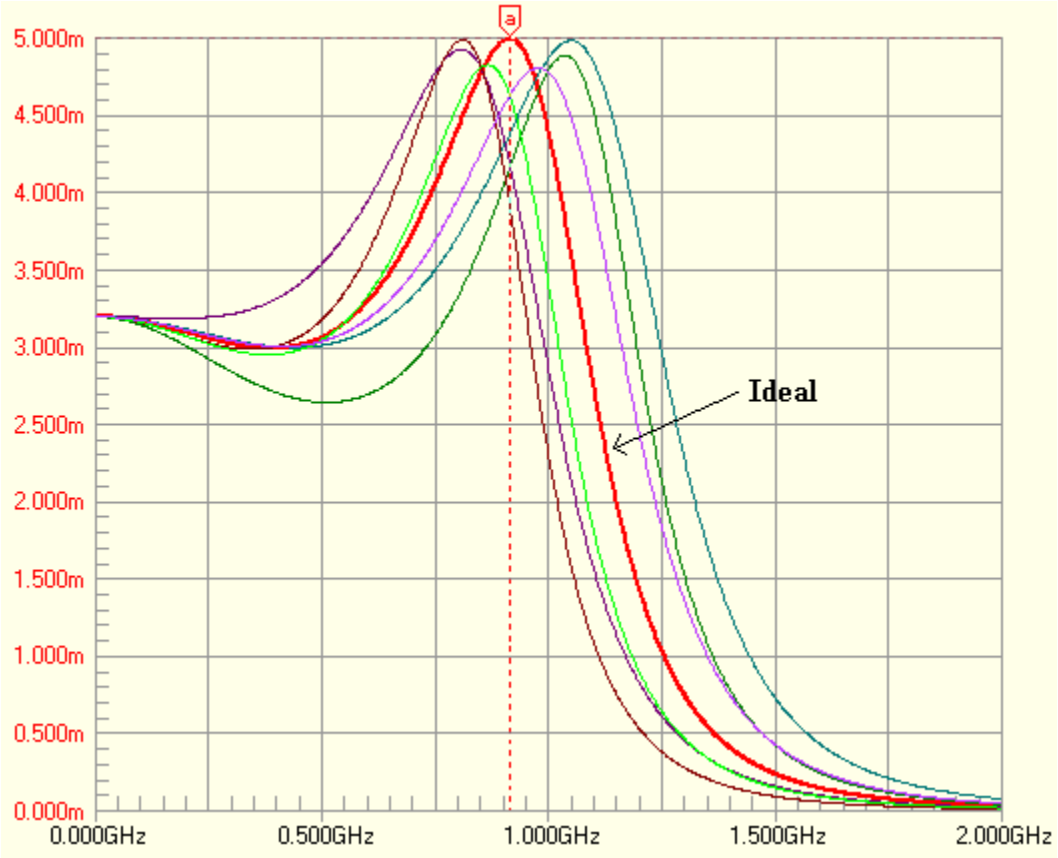


Figure 3.17 – Peak Power at the Load Versus Frequency Using a Worst Case Monte Carlo Sweep with the Ideal Case Shown in Red

As this graph shows, the ideal network provides exact matching allowing the maximum instantaneous power of 5mW to be received at the load. The graph also shows six worst-case examples of the AC power sweep. It can be seen that the maximum error is approximately 20% for this example.

3.3.2 Upward Transformation

The upward transformation is also performed with the output and load impedances in the parallel format. The equations for the upward transformation are the same as the downward transformation equations (3.19-3.25).

The resistive transformation graph for the upward transformation can be seen in the following figure [3]. In this case, Q_2 will be greater than the value of Q_1 because the second transformation performs a larger resistive jump.

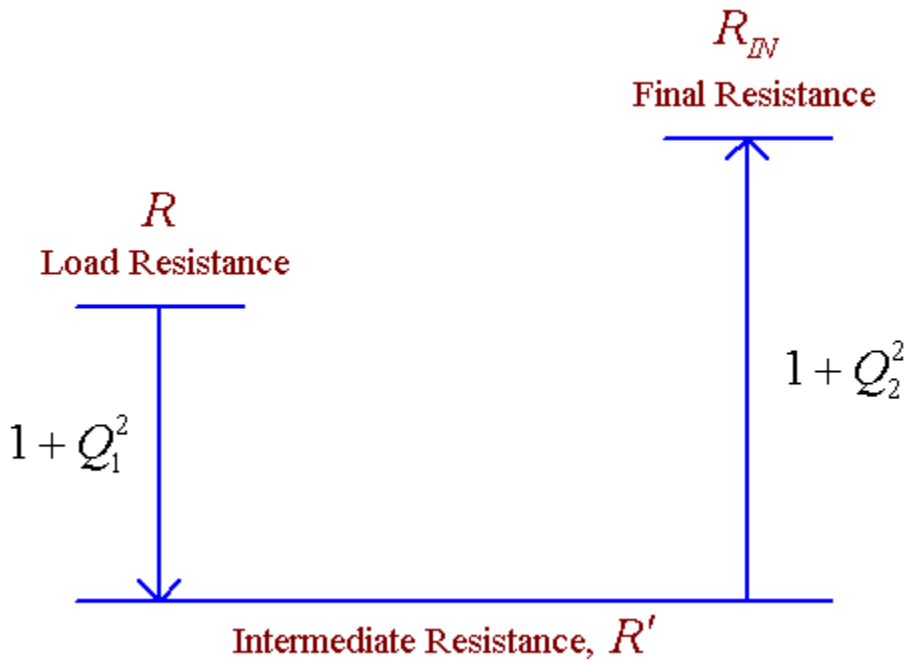


Figure 3.18 – Upward Resistive Transformation

An example of the Π – Network upward transformation can be seen below.

Example 6:

Repeat Example 2 to obtain a Π – Network with positive Q-values. Use the minimum value for Q_2. The given impedances are in the series format.

Solution:

Using the equations described in this subsection, the following values can be verified.

$$Q_2 = 1.735$$
$$Q_1 = 0$$
$$R' = R_{IN}$$
$$X_1 = 20.016\Omega$$
$$X_2 = 86.849\Omega$$
$$X_3 = -53.611\Omega$$

3.4 Wideband Four-Element Networks

Wider bandwidths can be obtained using a four-element impedance matching network. The four-element network is similar to two cascaded L – Networks that were described in Section 3.1. The objective of this section is only to indicate that wider bandwidths can be obtained using different networks. The addition of the fourth element contributes another component tolerance along with more parasitics, which degrade the on-chip performance to an extent that the network would cause more error than it would correct. For off-chip matching networks, more information can be obtained in [3].

3.5 Summary

The preceding sections explained the multiple configurations that can be used for impedance matching. Each network has advantages and disadvantage for specific applications. It is up to the analog designer to pick the network that will have the best performance. Within each topology, there is also the choice of multiple networks because of positive and negative Q-values. Along with this, the three element networks have an infinite number of possible networks, which are obtainable by varying the network bandwidth. All of these possibilities must be evaluated to produce a high-quality matching network.

4.0 ON-CHIP LAYOUT AND CONSIDERATIONS

Designing impedance matching networks requires knowledge about each individual component and how they affect the network's performance. This section describes ways to model each component (L or C) with an equivalent circuit so that the principles from the following section can be applied to designing matching networks. The following components were laid out using the AMI Semiconductor ABN Process with a 1.5-micron minimum feature size. The software package was Cadence Virtuoso.

4.1 Inductors

4.1.1 ASITIC

Analysis and Simulation of Spiral Inductors and Transformers for Integrated Circuits (ASITIC) is a software package developed at the University of California at Berkley that aids in the design of on-chip spiral inductors. ASITIC allows a structure to be laid out and simulated. The output will give an equivalent two-port network that will be discussed in more detail in Subsection 4.3.1. An example input file can be seen below.

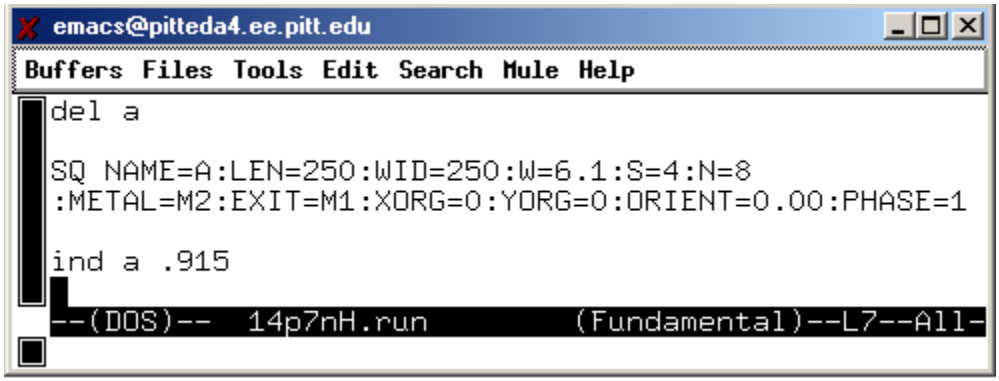


Figure 4.1 – Example ASITIC File for a 14.7nH Inductor

The "del" command deletes the spiral with the following name. In this case, the spiral name is "a". This line is not needed but is included to be sure the spiral named "a" has been cleared. The next line lists the program variables. An explanation of these variables can be seen in the following table. The final line gives a two-port equivalent circuit along with the inductance of spiral "a" at a frequency of 0.915GHz.

Table 4.1 – ASITIC Variable Explanations

Variable	Description
SQ NAME	-name of square spiral
LEN	-outer dimension in X direction
WID	-outer dimension in Y direction
W	-width of trace
S	-spacing between traces
N	-number of turns in 0.25 increments
METAL	-inductor is wound on this layer
EXIT	-center connection exits on this layer
XORG	-X origin on chip
YORG	-Y origin on chip
ORIENT	-angle of inductor layout
PHASE	-specifies positive terminal
	For more in-depth details, visit the ASITIC website. [5]

The output of this file can be seen in Figure 4.2.

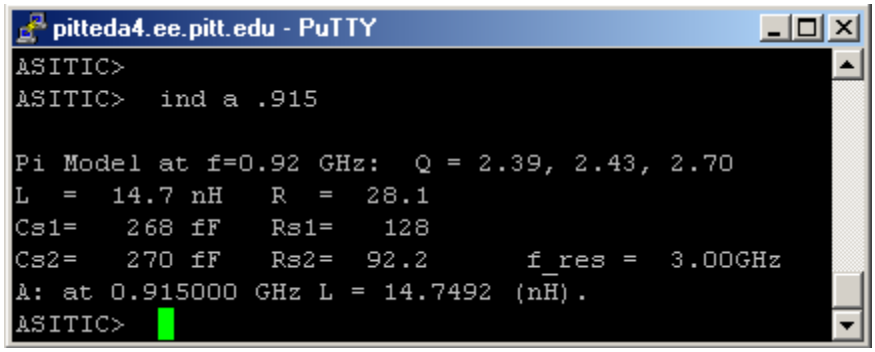


Figure 4.2 – ASITIC Output for 14.7nH Inductor

ASITIC returns the equivalent circuit and the inductance at the specified frequency, which is 0.915GHz in this case. It also plots the given inductor for visual inspection. The 14.7nH square spiral can be seen in the following figure.

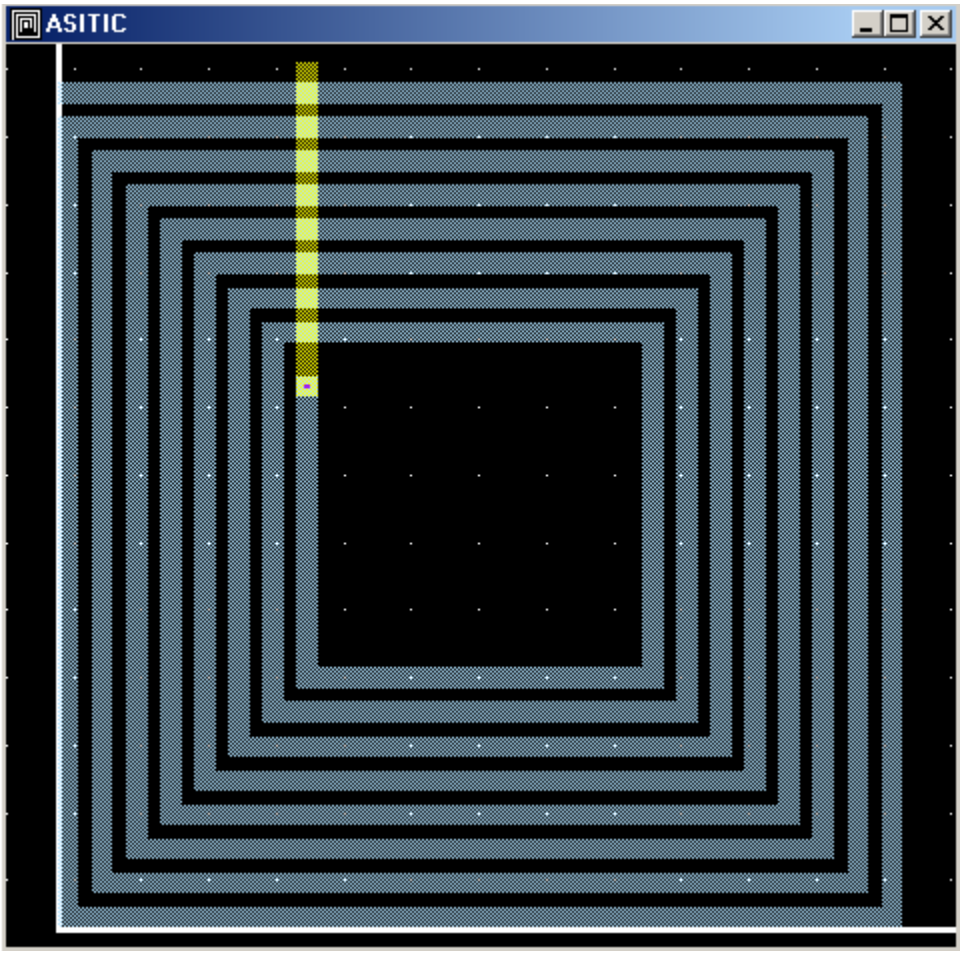


Figure 4.3 – ASITIC Graphic Output of 14.7nH Inductor

4.1.2 Current Sheet Approximation

The current sheet approximation is another way to analyze inductors. The square spiral is approximated as a solid plane like the one given in the following figure. The following figure and equations were presented by S. S. Mohan from Stanford University in his Ph.D. Oral Examination [6].

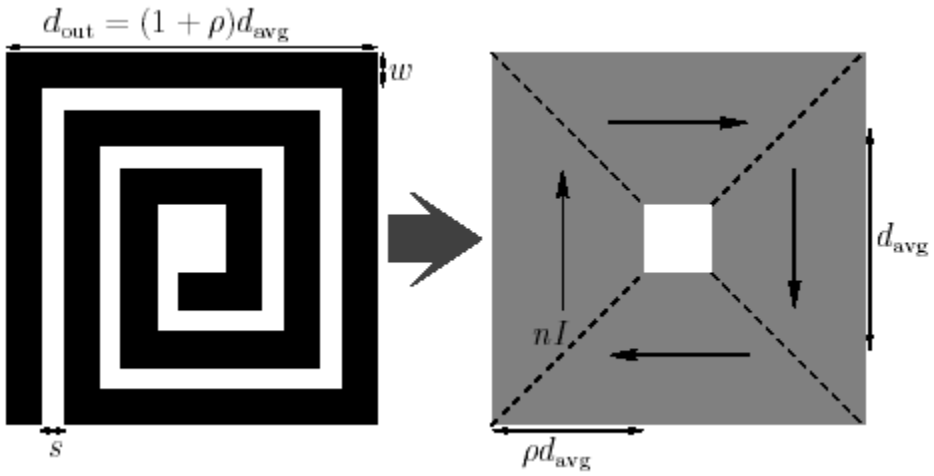


Figure 4.4 – Spiral Approximated as a Solid Sheet

The inductance of a spiral is given by the following expression.

$$L_{SQ} = \frac{2 \cdot \mu \cdot n^2 \cdot d_{avg}}{\pi}\left[\ln\left(\frac{2.067}{\rho}\right) + 0.178 \cdot \rho + 0.125 \cdot \rho^2\right] \qquad (4.1)$$

where

$$\rho = \frac{d_{out} - d_{in}}{d_{out} + d_{in}} \qquad (4.2)$$

d_{out} & d_{in} are the outside and inside diameter from side to side, respectively. For tightly wound spirals, $d_{in} \approx 0$ which makes $\rho \approx 1$. "n" is the number of turns in the spiral while "μ" describes the overall permeability.

Using this method for the 14.7nH inductor designed using ASITIC yields the following values.

$$d_{out} = 250\mu m$$
$$d_{in} = 96.4\mu m$$
$$d_{avg} = 173.2\mu m$$
$$\rho = 0.4434$$

Assuming $\mu \approx \mu_o = 4 \cdot \pi \cdot 10^{-7}$ yields

$$L_{SQ} = 14.569nH$$

This value is extremely close to the results of 14.7nH obtained using ASITIC. One drawback of this method is the lack of parasitic knowledge. ASITIC provides an equivalent two-port network to aid in design while this method simply gives the inductance value. For this reason, all of the impedance matching inductors were laid out using ASITIC. The current sheet approximation is very useful in other applications since it is much quicker than designing with ASITIC.

4.2 Capacitors

4.2.1 Typical

There are numerous ways to layout a capacitor on an integrated circuit. These layouts will depend on the process being used. The capacitors described in this subsection were created using a poly-to-poly layout. Poly1 is used as the bottom plate while poly2 acts as the upper plate. Metal1 is used to connect to both the top and bottom plates. This is easily seen in the following layout.

The layouts provided in the subsequent figures are by no means unique. However, they do incorporate the fundamentals of design that are needed for radio frequency matching capacitors. The main goal of the layouts is to provide a basis that can be extended to individual designs.

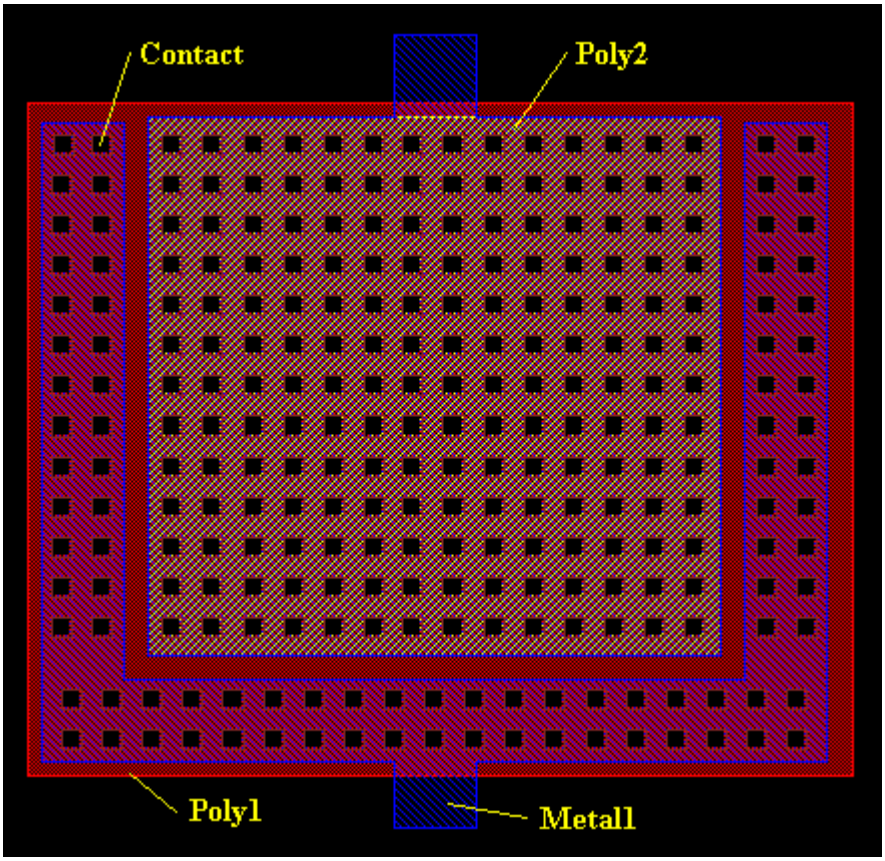


Figure 4.5 – Typical 1pF Capacitor Layout

The previous figure shows how metal1 can directly connect to poly2 using contacts. In order for metal1 to connect to poly1, the poly1 sheet must be stretched from underneath poly2 to allow the contacts to drop from the metal1 layer all the way to poly1. The capacitance is dictated by the overlap of poly1 and poly2. For the ABN AMI process, the capacitance is given by 584aF/um^2 [7].

For impedance matching networks, it is very important to minimize the loss in the matching components. This is done by keeping the series resistance as small as possible. As explained above, the plates of the capacitor are formed with polysilicon. For the ABN AMI process, the sheet resistance for poly1 and poly2 are 23.3Ω/□ and 21.4 Ω/□, respectively. There is also a contact resistance between a polysilicon layer and a metal layer. For this process, the poly1-metal1 contact resistance is 27.6Ω while it is only 14.7Ω for a poly2-metal1 connection [7].

These resistances add considerable series resistance to the capacitor layout. To minimize this value, fingers are used to provide parallel resistive paths.

4.2.2 Fingers

On a typical square capacitor, the top plate will have little resistance because metal1 can contact the plate everywhere. However, the bottom plate is only connected to metal1 on the outside. Since there are no connections from poly1 to metal1 in the center of the capacitor, a large resistance is seen between the actual dielectric and metal trace. This can be seen in Figure 4.5.

To reduce this resistance, the capacitor needs to be rectangular. Figure 4.6 shows an example of a rectangular capacitor.

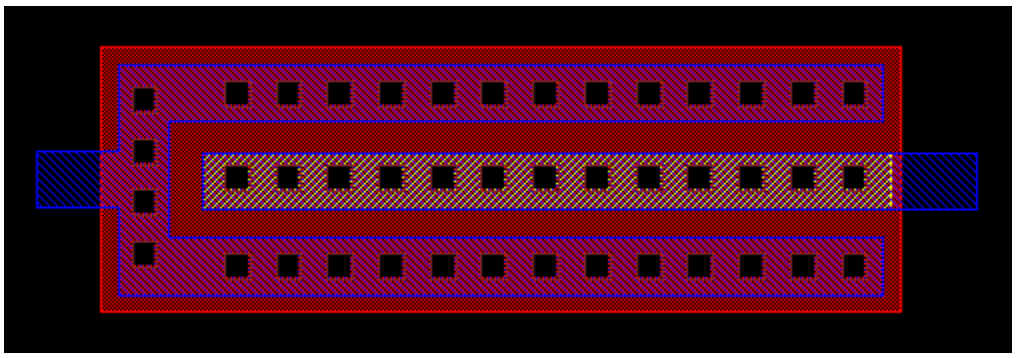


Figure 4.6 – Rectangular Capacitor

It can easily be seen that the resistance is minimized when the fingers are at minimum feature size. Using the minimum size gives the minimum resistance, but it also gives the minimum capacitance for a given area. Most of the area is required for the spacing between the fingers. By doubling the minimum feature size to a double row of contacts, the capacitance is doubled without a significant increase in resistance. The spacing between fingers, however, remains at minimum feature size. A layout using this technique can be seen in Figure 4.7.

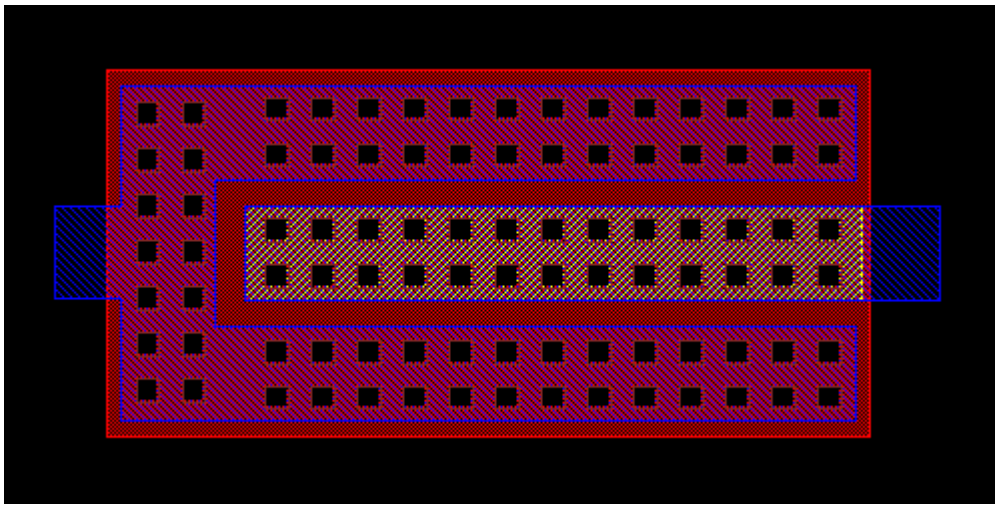


Figure 4.7 – Double Row of Contacts

By placing multiple fingers in parallel, a coarse value can be obtained. To acquire the exact value, the length of the fingers must be adjusted. An example layout can be seen in Figure 4.8.

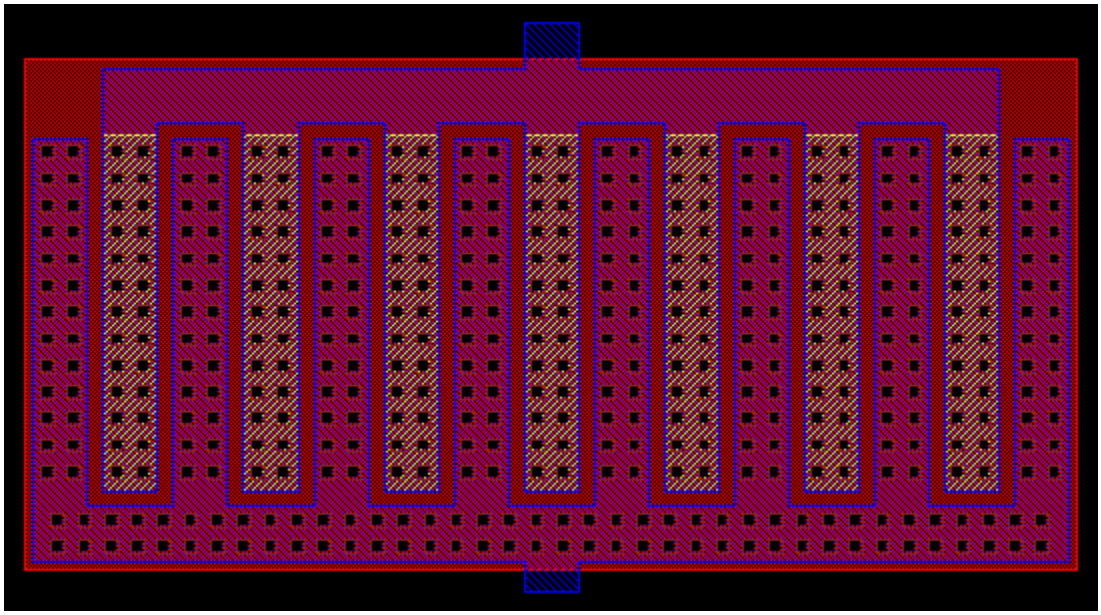


Figure 4.8 – 1pF Capacitor Layout

4.3 Parasitics and Tolerances

4.3.1 Equivalent Circuits

As with all components, the inductor and capacitor layouts shown previously are not merely described by ideal components. There will be an equivalent circuit used to represent the component at a particular frequency. These parasitic effects, in general, degrade the performance of the component, which in turn degrades the entire impedance matching network. If the effects become too great, the network will cause more reflection in the case of radio frequency networks than the original unmatched circuit. For this reason, the parasitics of the components must be understood and minimized to insure the least amount of error.

ASITIC calculates an equivalent circuit for a specified inductor layout. More information on ASITIC can be found in Subsection 4.1.1. The two-port equivalent of an on-chip inductor layout can be seen in Figure 4.9 [5].

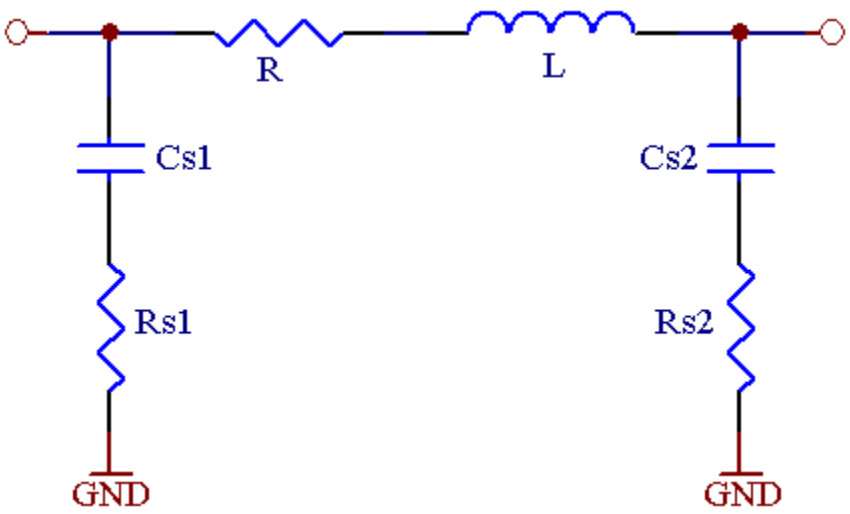


Figure 4.9 – ASITIC Two-port Equivalent Circuit for an Inductor

The values for the variables seen in Figure 4.9 are given in an ASITIC output file. An example output file can be seen in Figure 4.2. This circuit directly replaces the fabricated inductor in the matching network. For cases where Cs1 and Rs1 or Cs2 and Rs2 are small, as is the case for the 14.7nH layout in Subsection 4.1.1, they can typically be ignored if the other components in the matching network have comparably larger impedances. This can be seen more easily by

converting Cs1 and Rs1 to their parallel form, Cp1 and Rp1. The value of Rp1 becomes 3.42KΩ while Cp1 is approximately equal to Cs1. Since Rp1 is large, it can be neglected for most networks, which are usually operating around fifty ohms. This leaves only Cp1, or Cs1 because the values are similar. Neglecting this value will depend on the network's components and configuration.

Capacitors have a very similar equivalent circuit as can be seen in Figure 4.10.

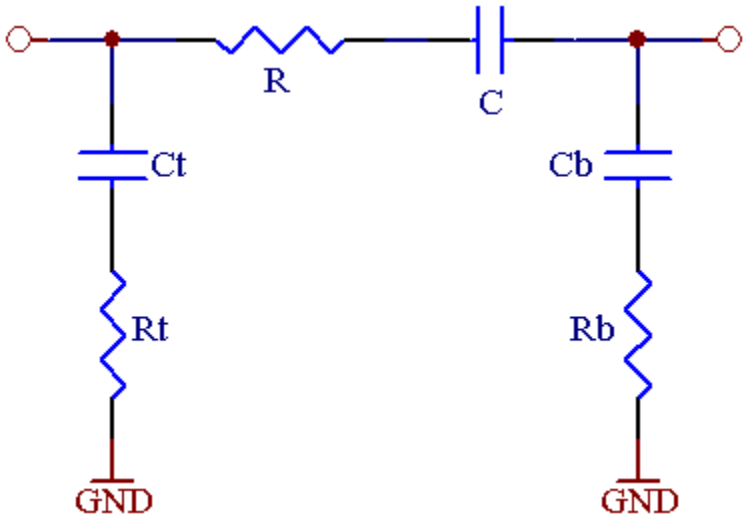


Figure 4.10 – Capacitor Equivalent Circuit

As described in Subsection 4.2.1, the capacitors used for the tested matching networks were formed using the poly1 and poly2 layers. Because the poly2 layer is above the poly1 layer and the poly1 sheet is larger to accommodate the metal1 connections, the top plate will have very little coupling to the substrate. This means the capacitor and resistor to ground associated with the top plate can be neglected as long as no traces are run over the top of the capacitor layout [8]. The simplified circuit can be seen in Figure 4.11.

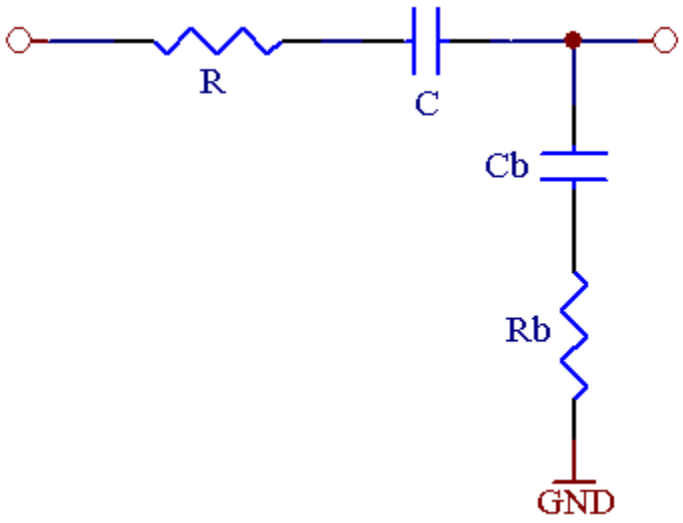


Figure 4.11 – Simplified Capacitor Equivalent Circuit

The parasitic branch associated with the bottom plate will be dependant on the area of poly1. The poly1 to substrate coupling is around 37aF/um^2 [7]. As was explained for the inductor, the resistance value, Rb, can generally be neglected because of its large parallel value. The circuit can be simplified even more when the capacitor has a grounded terminal. The bottom plate of the capacitor can be run to ground which eliminates the other parasitic branch leaving only the series resistance, R. To minimize the value of R, the capacitor can be laid out using the finger technique described in Subsection 4.2.2. Using this method, the value of R can be minimized to a value that can be neglected leaving only the desired capacitance.

Obviously, the bottom terminal of the capacitor cannot always be grounded. In these cases, the analysis must take into account the additional elements, and the performance of the component will be degraded.

4.3.2 Effects on Ideal Networks

Impedance matching networks are never ideal. Even with discrete components, there are tolerances associated with each element. There is also the problem of choosing standard values. On-chip networks do not have this problem because any reasonable value can be fabricated which would again have some tolerance. The main problem, as described in this subsection, is parasitic effects caused by the substrate. In either case, the network performance will be

degraded to some extent. These effects must be considered to determine if the network will cause more mismatch in the circuit than it corrects.

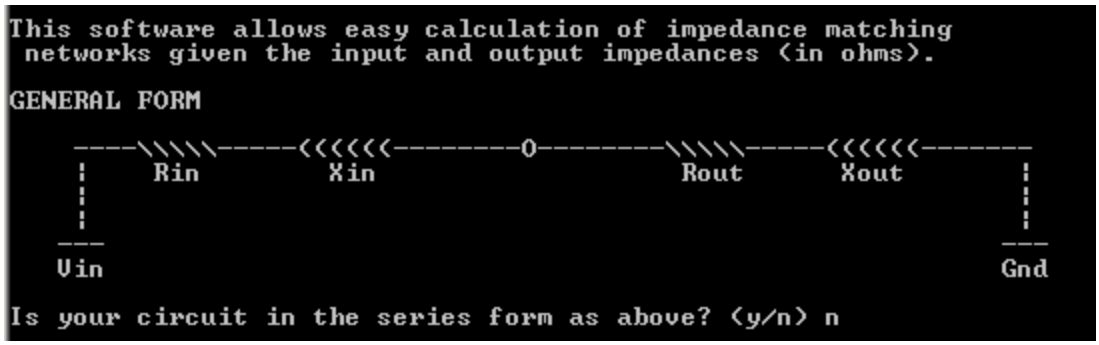
5.0 IMPEDANCE MATCHING SOFTWARE

5.1 Software Functions

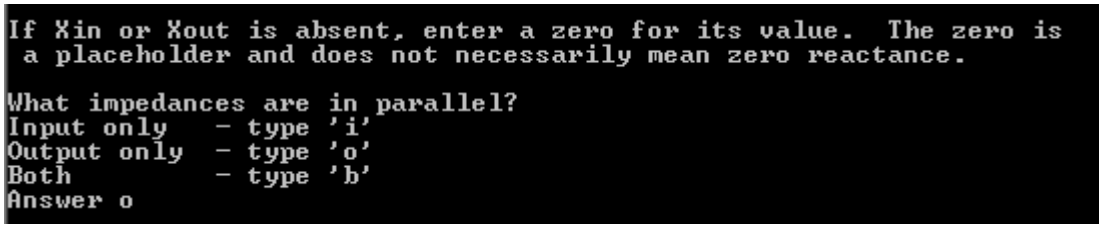
The calculation of impedance matching networks can be time consuming and monotonous. For this reason, software was created to speed the process. The purpose of the software is to combine the ideal networks presented in Section 3.0 with the parasitics caused by the common substrate. The program was written in C++. The source code is included in Appendix C. The first function of the software is to allow a choice among multiple topologies each having a variety of selections. These selections are created using positive and negative Q-values. The user is then asked to enter the parasitic values as described in Section 4.3. These values can be taken from ASITIC or can be calculated using the process parameters. The program uses these values to calculate the total power transferred and the power lost. The program also separates the lost power into categories so the dominant effect can be identified. In most cases, it will be the series resistance in the network's inductors. These resistances tend to be high due to the low quality factors of inductors formed on a silicon substrate [9]. Finally, the program will make a decision on whether the network should or should not be used by comparing the power received with and without the network. If the network is too lossy, more power is received without the matching network than when it is used.

5.2 Software Example for a Fabricated Network

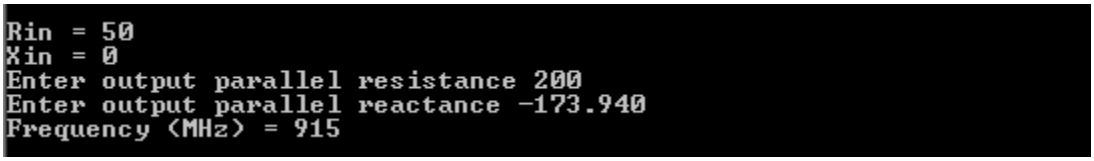
To better illustrate the performance of the impedance matching software, a fabricated network from Section 6.2 was entered into the program. Since the program returns results for the L –, T –, and Π – Networks, it was possible to verify the fabricated L – Network and one of the Π – Networks in a single run of the program. Π – Network (1) was chosen for the sample given below. For a step-by-step explanation of the program, the display window has been split into various sections with a description following each.



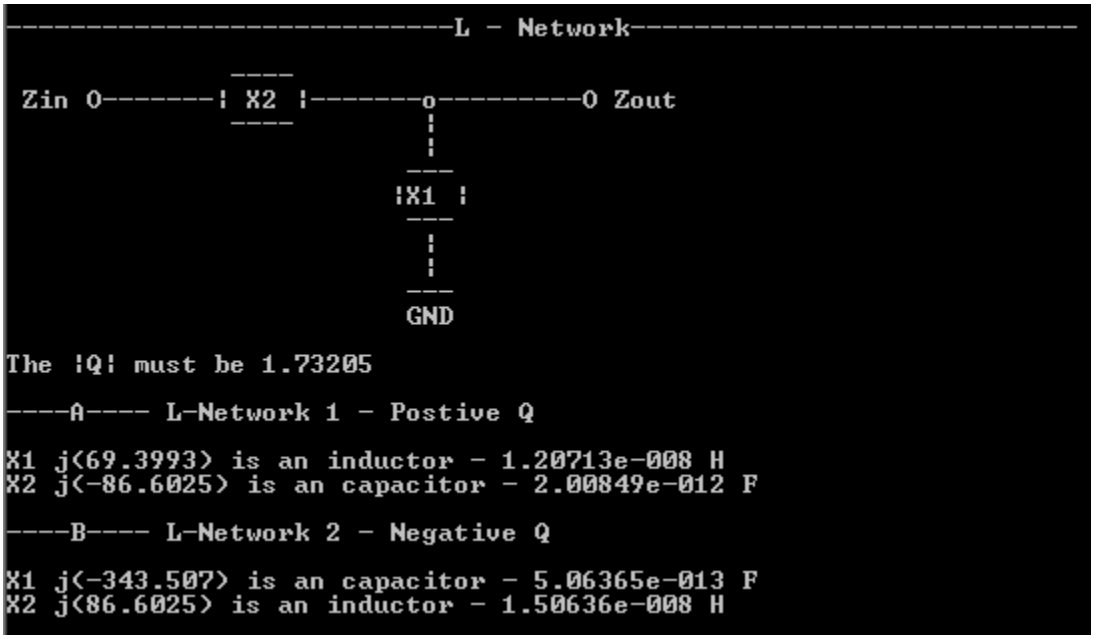
The user is first asked about the circuit's form. For the fabricated networks, the load was in the parallel form, so the choice was "n" for no.



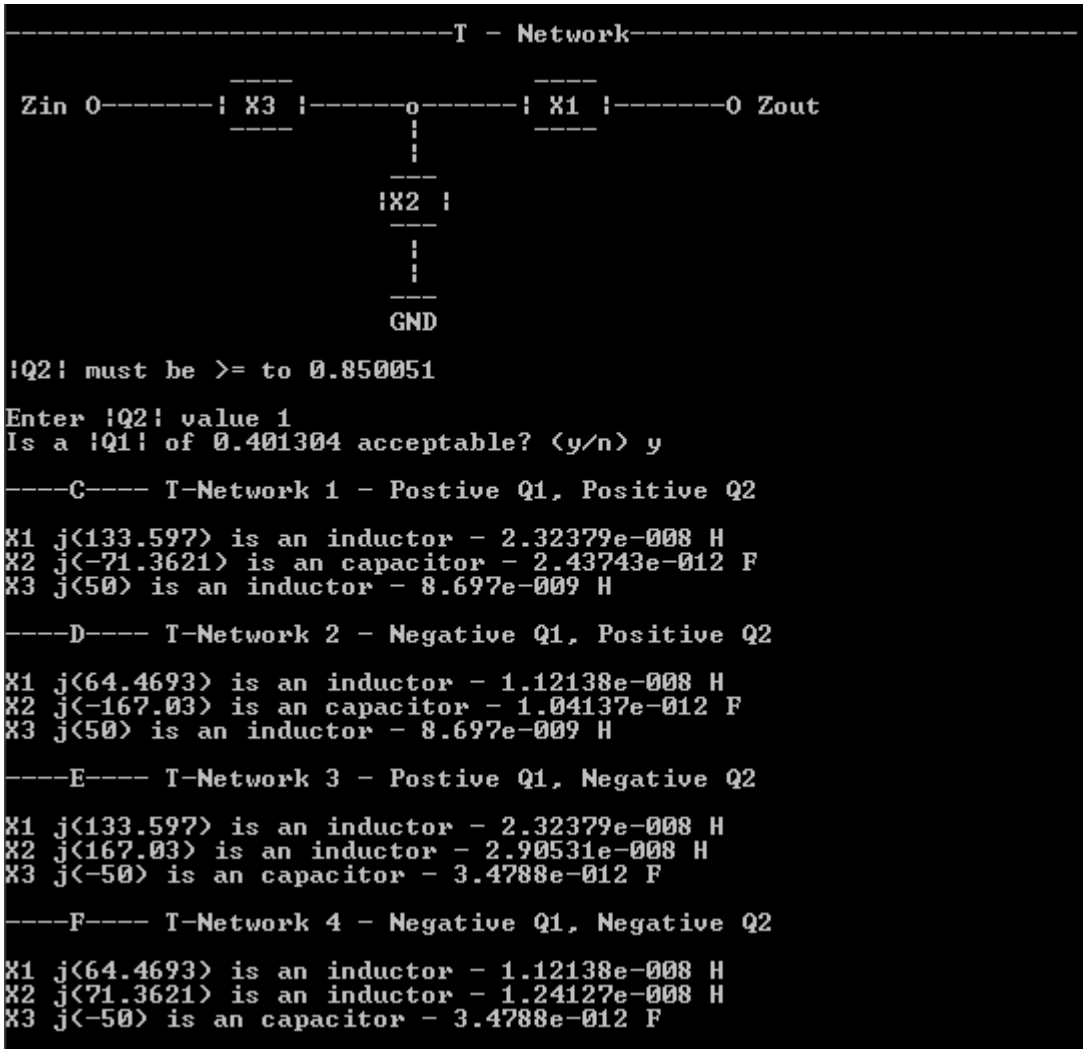
Next, the program informs the user that an absent reactance must be specified as a zero even though it may be infinite. This is required for calculation purposes since infinity cannot be entered. An "o" is entered to specify that that the output impedance is in the parallel form.



The user is then prompted to enter the source and load impedances in ohms. The frequency of operation is also entered.

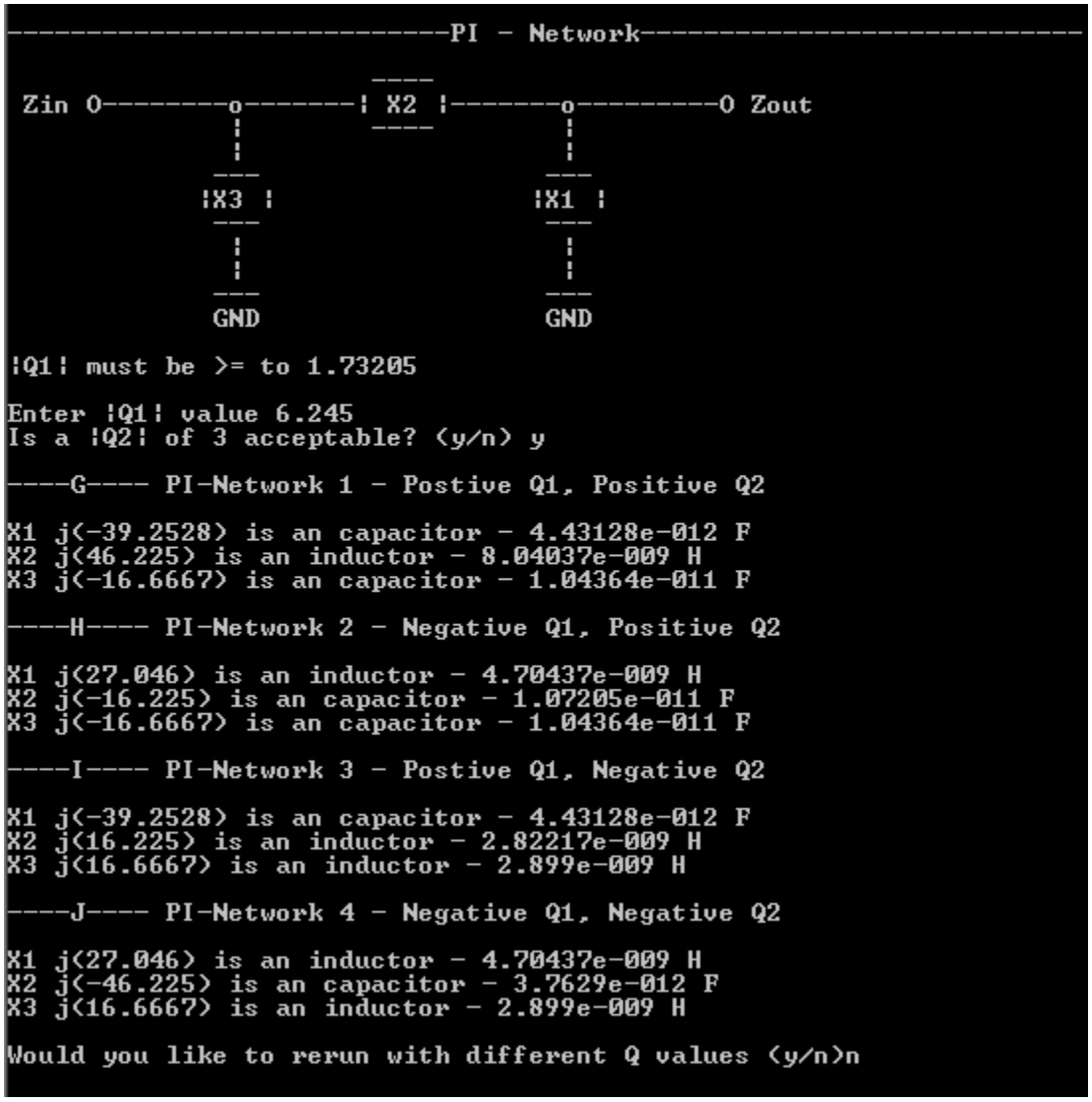


Because the L – Network does not have a degree of freedom, the two possible matching networks are returned. The first network, labeled A, has a positive Q while network B has a negative Q-value. A and B will be used later in the program to specify which network will be used.



The T – Network is calculated next. Because it has one degree of freedom, a variable must be chosen. This program allows the user to choose the magnitude of Q_2. The program also displays the minimum value of Q_2. Generally the designer will pick the lowest Q-value that provides reasonable component values. A low Q-value is less sensitive to component variations because it has a wider bandwidth.

After a value is entered, the program returns the value of Q_1, which will always be less than Q_2 for this case. The user is then prompted to accept the value of Q_1. If the value is not accepted, the program will ask for a new Q_2. Otherwise, the program will display the four possible matching networks for these Q-values. They are labeled C through F for easy reference later in the program.



Next, the Π – Network is calculated in the same way as the T – Network. In this case, however, Q_1 must be specified while Q_2 is calculated. The designer will again try to find the lowest Q-value that yields reasonable component values. The program then uses these Q-values to return the four possible matching networks, which are labeled G through J. The user now has the option to rerun the program with different Q-values. This will only affect the T – and Π – Networks because the L – Network must have the displayed Q-value.

The next part of the program is used to characterize the networks performance when fabricated on a semiconductor chip. It uses the equivalent circuits described in Section 4.3 to replace the ideal components with a model that better describes their behavior. This section only calculates

the parasitic effects for one network because there are numerous variables for each element. The network described below will be "G" which is the first fabricated Π – network from Section 6.2.

```
------------------------PARASITIC EFFECTS-------------------------

The following calculations will examine the effects of the
 substrate's parasitics.  For inductors, the parasitic elements
 can be obtained using the output file from ASITIC.  For capacitors
 to ground, the bottom plate is assume to be run to ground, which
 minimizes the parasitics.  In this case, the parasitic capacitor
 can be assumed in the low femto farads.  The series resistance of
 capacitors can be assumed less than 0.5 ohms if the layout was
 done using the method outlined in the accompanying paper.  For
 floating capacitors, the bottom plate parasitics can be calculated
 using the method outlined in the accompanying paper.


Which Network from above are you using? (A/B/C/D/E/F/G/H/I/J) G

Enter the following parameters for the capacitor X1
R = 0.5
C (pF) = 4.43
Rt = 0
Ct (pF) = 0.1
Enter the following parameters for the inductor X2
R = 13.1
L (nH) = 8.15
Rs1 = 103
Cs1 (pF) = 0.295
Rs2 = 68.7
Cs2 (pF) = 0.304
Enter the following parameters for the capacitor X3
R = 0.5
C (pF) = 10.44
Rt = 0
Ct (pF) = 0.1
```

After specifying the chosen network, the user is asked to enter the parasitic elements for each component. As the program explanation states, these values are outlined in Section 4.3.

```
The old load including the ideal PI - matching network was

         Z = 50 + j(-2.55796e-015)

The load including the parasitics of the PI - matching network is

         Z = 14.5357 + j(-15.9738)

The input impedance was given or calculated to be

         Z = 50 + j(0)

Amount of Maximum Power Transferred  --> 65.7723%
```

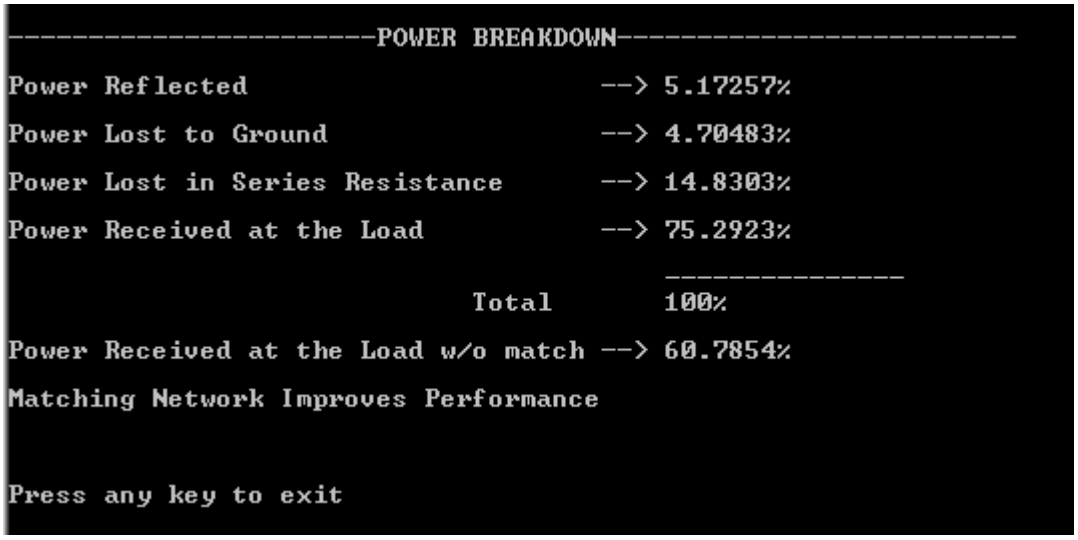
Using the parasitic information, the program calculates the equivalent load impedance including the nonideal network. It also calculates the load with the ideal network. As can be seen, the

reactive part is not zero because there is round-off error in the program. The given input impedance is also displayed for comparison purposes. The last line above shows the amount of power transferred to the input of the matching network for the nonideal case. This number should be as close to one hundred percent as possible. Here, the parasitics have caused a mismatch, which is reflecting a large portion of the available power.

```
----------------------POWER BREAKDOWN-------------------------
Power Reflected                         --> 34.2277%
Power Lost to Ground                    --> 1.22157%
Power Lost in Series Resistance         --> 49.6912%
Power Received at the Load              --> 14.8595%
                                            ---------------
                              Total         100%
Power Received at the Load w/o match --> 60.7854%
Matching Network Degrades Performance -- DO NOT USE NETWORK


Press any key to exit
```

The program concludes by explaining where the available power has gone. The previous program segment showed that 65.7723% of the power was received at the matching network. This means that 34.2277% must be reflected to obtain 100%. Of the 65.7723%, some is lost to ground while most is lost in the series resistances. As the segment above shows, only 14.8595% of the maximum available power is received at the load. From this power breakdown it can be seen that this is a bad matching network because the unmatched network receives 60.7854%. As the program states, this network should not be used. Instead, the source should be directly connected to the load.



As an example of a good network, the same case was rerun neglecting the series resistance of the inductor. From this, it can be seen that the network allows the load to receive approximately 15% more power than in the unmatched case. This example also shows that the series resistance of the inductor is a major part of the loss in the network. This will be discussed in more detail in the succeeding sections.

This program is extremely useful in developing impedance matching networks. It significantly reduces the design time. It also allows the user to examine the effects that the parasitics will have on the network. As was shown above, some impedance matching networks actually degrade the performance of the system.

6.0 RESULTS

6.1 Discrete Components

The first test network was built using discrete components on a breadboard in order to prove the concepts explained in previous sections. Because the performance of thru-hole components degrades as frequency is increased, this network was done at a relatively low frequency [4]. The Agilent 8712ET RF Network analyzer was using to perform the measurements on the discrete network. Because the analyzer will only perform measurements down to a frequency of 300kHz, this was chosen as the test frequency.

Although the simplest example would be to match a load to the 50-ohm source impedance of the network analyzer, this method was not used. The calculated values would be required to correspond to the standard values for discrete components. Instead, the load and network were chosen from the list of standard component values. The overall impedance of the network was then measured to determine if it agrees with the calculated value.

The L – Network was chosen to show how the concepts of impedance matching work. The T – Network and Π – Network were not tested in this section because they are formed on the same principals as the L – Network. The network that was tested can be seen in Figure 6.1.

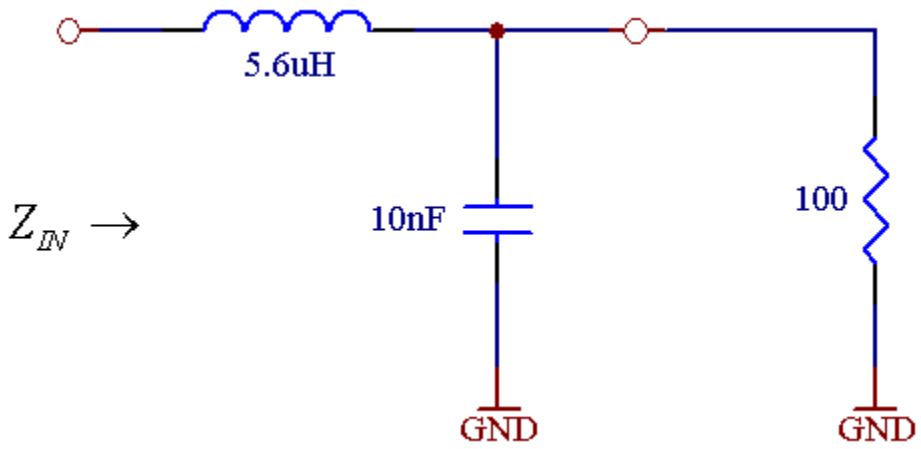


Figure 6.1 – Discrete Component L – Network

A smith chart obtained from the network analyzer can be seen in Figure 6.2. It shows the input impedance of the network as a function of frequency. The test setup used to obtain these results can be seen in Appendix D.1.

The following table summarizes the results obtained in this section.

Table 6.1 – Results for Discrete Component Network

	Impedance	Components (In series)
Calculated	**21.96 – j30.84**	**21.96Ω & 17.20nF**
Measured	**24.10 – j30.11**	**24.10Ω & 17.62nF**

As can be seen from Figure 6.2 and Table 6.1, the calculated and measured results agree within ten percent. There is a minor difference in the resistive part of the solution. This can be partially explained by the absence of the inductor's series resistance in the calculated network. Also, the components used in the matching network all had a tolerance of ± 5%. This can also explain the minor differences in the results.

The L – Network shown in Figure 6.1 would be used to match a source impedance of 21.96 + j30.84 to the load of 100 ohms. Without the matching network, the load only receives 55.5% of the available power. Adding the matching network from above increases the power to a near maximum at 99.8%. The minor deviation from 100% is caused by the slight mismatch seen in Table 6.1.

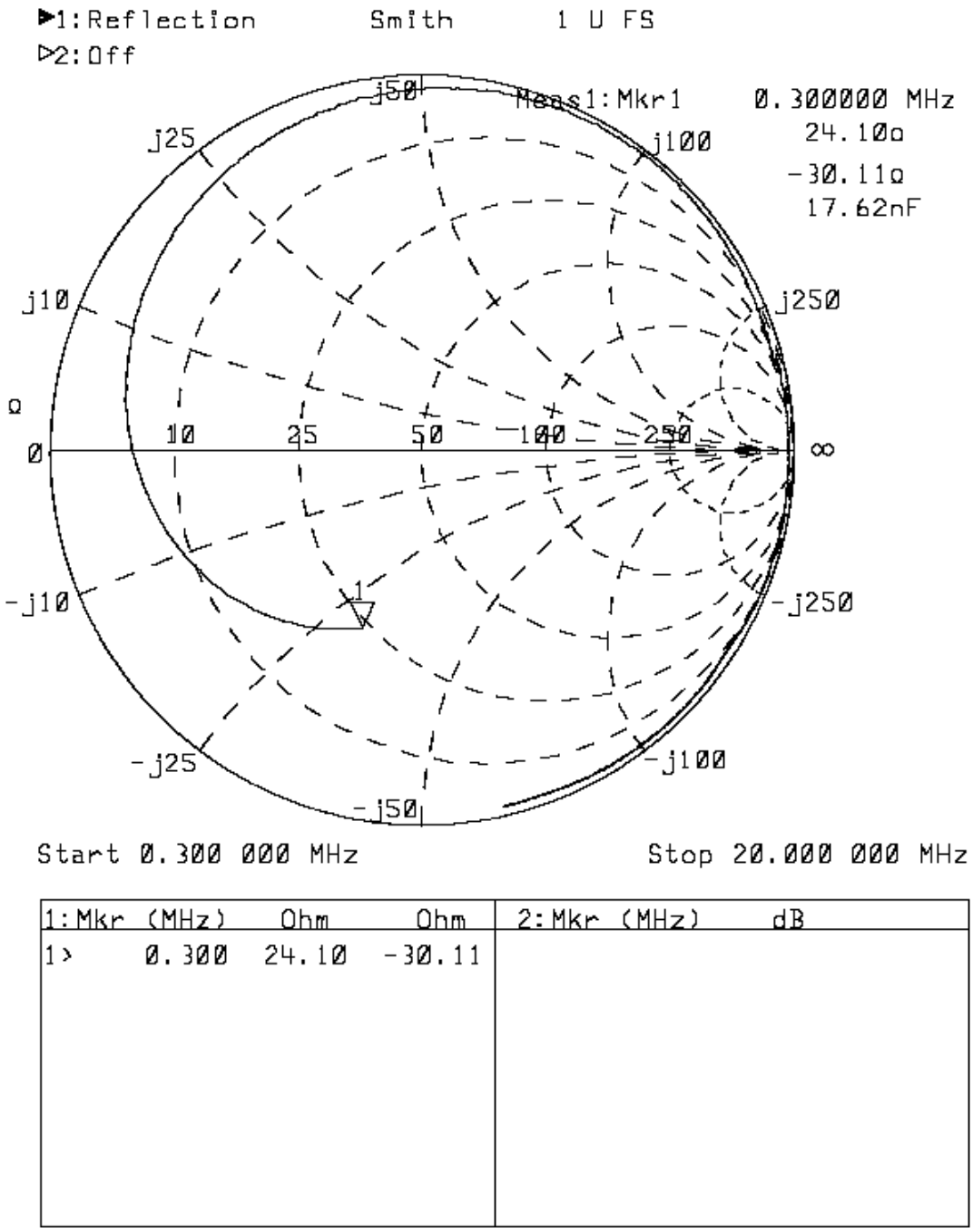


Figure 6.2 – Smith Chart of Impedance Results

As this example shows, impedance matching will work with thru-hole discrete components, but only at frequencies where the components are close to ideal. For printed circuit board (PCB) designs at high frequencies, surface mount devices must be used to minimize the parasitic effects.

Although this example was successful in showing the application of impedance matching, the primary focus of this research is on-chip analog fabrication discussed in the following section.

6.2 On-Chip

To help demonstrate impedance matching techniques on an integrated circuit (IC), several networks were fabricated. All of the networks were designed to match a 200-ohm resistor in parallel with a 1pF capacitor to a source impedance of 50 ohms at 915MHz. A 50-ohm source impedance was chosen so an RF network analyzer could be used to measure the impedance looking into the network. For the ideal case, the network analyzer should see a 50-ohm load. Unlike the discrete case, the network can be designed to match the 50-ohm source impedance because, theoretically, any practical value can be designed on-chip.

Due to space constraints on the chip and budget considerations, only three networks were fabricated. These networks are summarized in the following table. The schematic and layout of each network including the load can also be seen following the table.

Table 6.2 – On-Chip Networks and Components

Type	Layout	Q_1	Q_2	X_1	X_2	X_3
L – Network	Figure 6.3	1.732	-----	12.07nH	2.01pF	-----
Π – Network (1)	Figure 6.4	6.245	3	4.43pF	8.04nH	10.44pF
Π – Network (2)	Figure 6.5	3	1.225	1.61pF	14.70nH	4.26pF

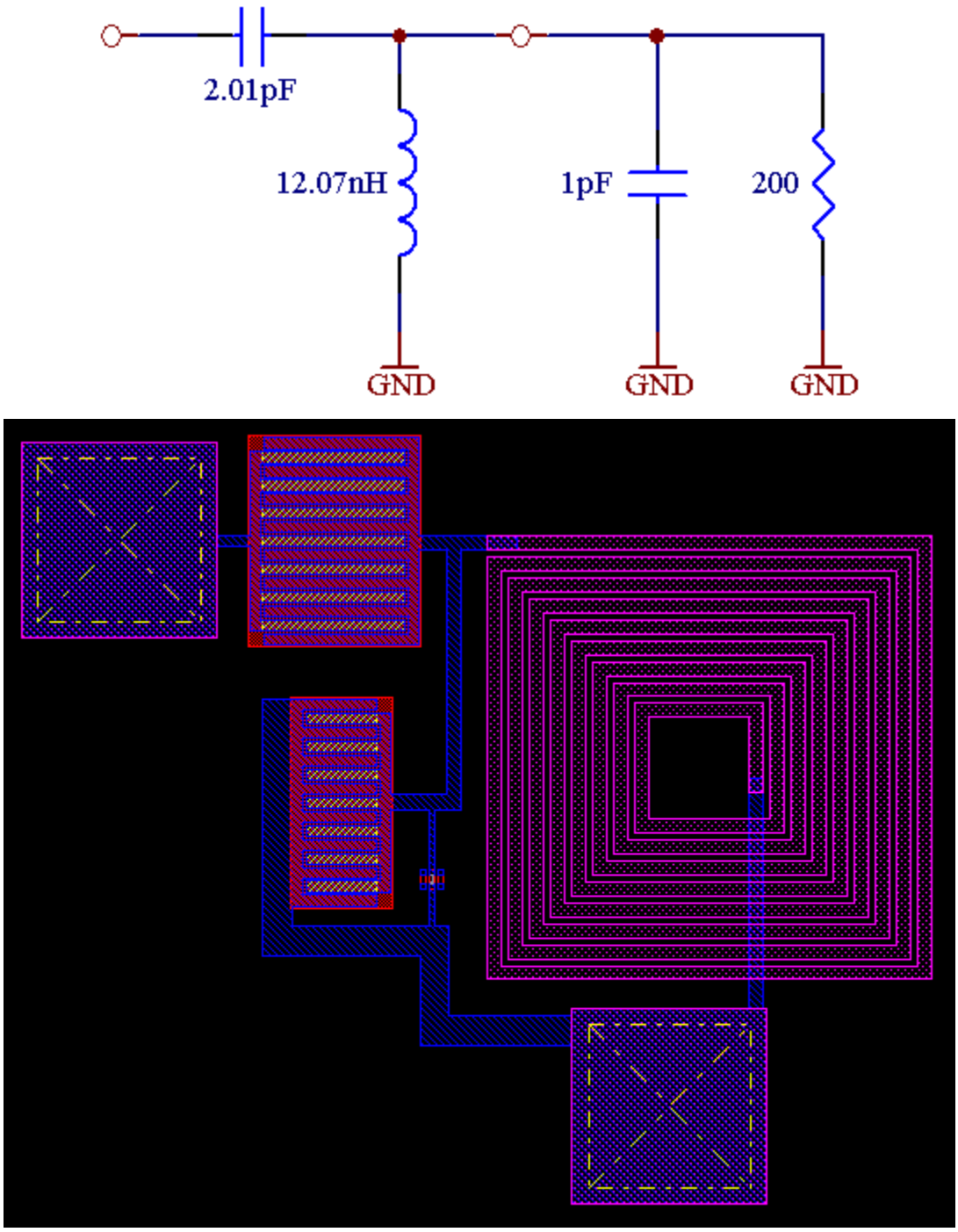


Figure 6.3 – L – Network Schematic and Layout

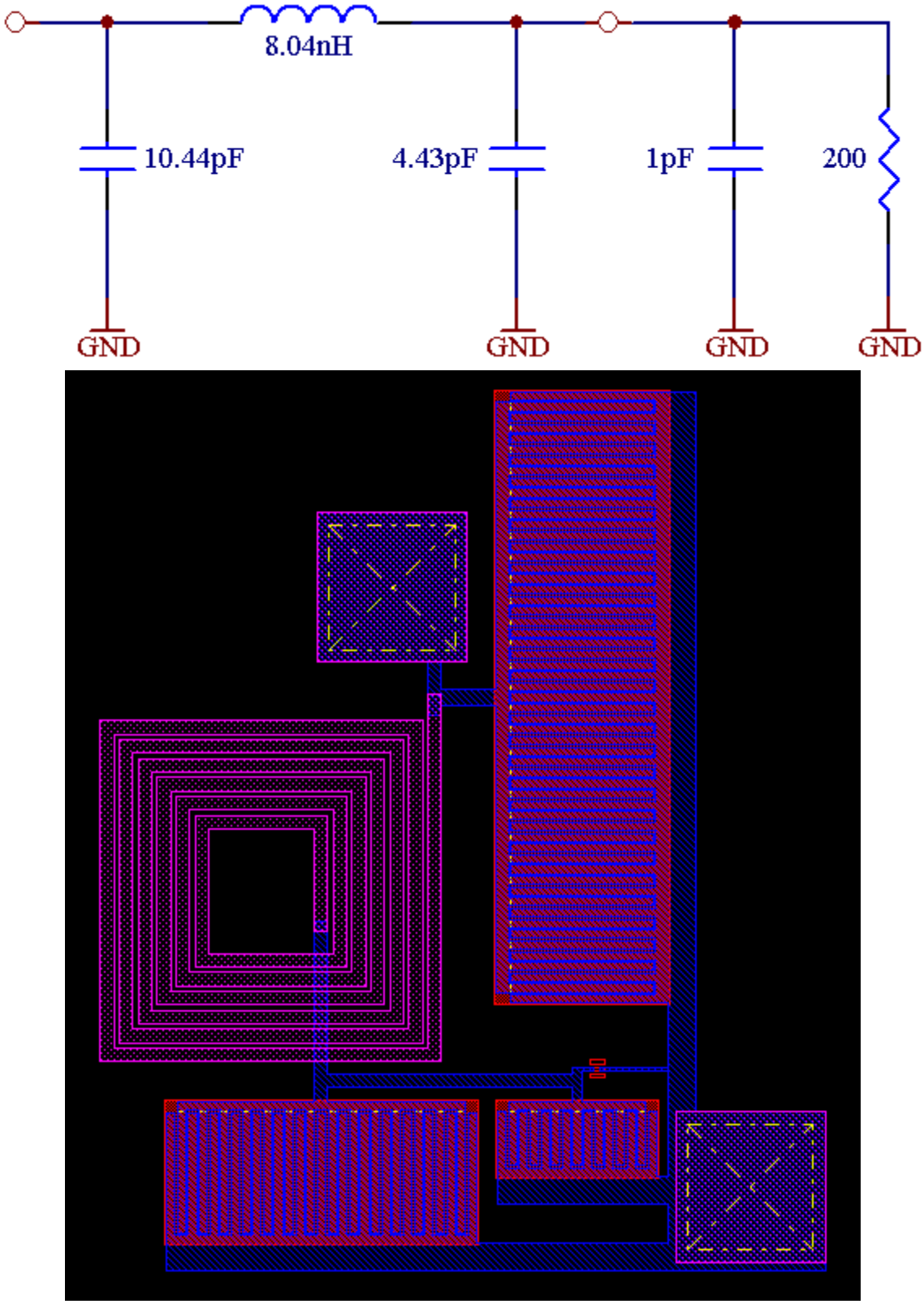


Figure 6.4 – Π – Network (1) Schematic and Layout

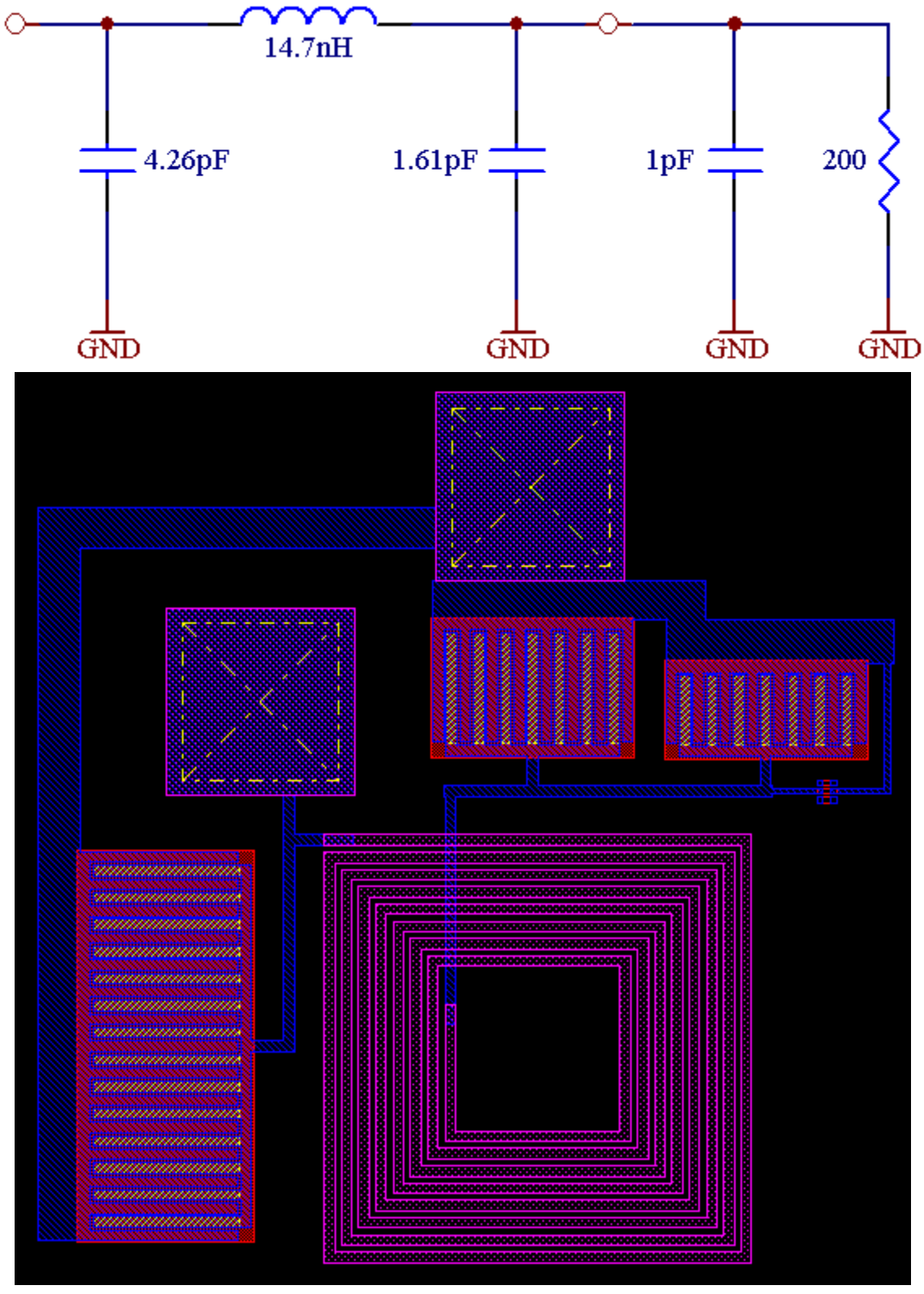


Figure 6.5 – Π – Network (2) Schematic and Layout

The T – Network was not chosen as a candidate for test because it has the worst performance for on-chip networks and also because of space constraints on the fabricated chip. The poor performance is caused by the additional floating element, which adds more parasitic branches to the substrate. By examining Figure 3.9 and Figure 3.14, it can be seen that the T – Network has five parasitic branches to ground (number of component nodes not grounded), while the Π – Network only has four. The result, in general, suggests less coupling to the substrate for the Π – Network. Refer to Section 4.3 for more details.

To better understand the test networks shown in Figures 6.3–6.5, all of the components were also fabricated separately on a chip for individual analysis. The following table compares the predicted value to the actual value of the on-chip components. These measurements were done using the Agilent 4284A Precision LCR Meter, which was connected to the chip using a four-lead probe station. The test frequency was 1MHz with a level of one volt. The picture in Appendix D.2 shows the experimental setup.

Table 6.3 – On-Chip Networks and Components at 1MHz

Network/Load	Predicted Value	Measured Value	% Error
Load Resistor	200 Ω	159.75 Ω	20.125 ↓
Load Capacitor	1 pF	1.354 pF	35.4 ↑
L – Network	12.07 nH	11.84 nH	1.906 ↓
Inductor R_S	20.7 Ω	20.55 Ω	0.725 ↓
L – Network	2.01 pF	2.43 pF	20.896 ↑
Π – Network (1)	4.43 pF	4.833 pF	9.097 ↑
Π – Network (1)	8.04 nH	12.5 nH	55.473 ↑
Inductor R_S	13.3 Ω	13.09 Ω	1.579 ↓
Π – Network (1)	10.44 pF	11.319 pF	8.420 ↑
Π – Network (2)	1.61 pF	2.056 pF	27.702 ↑
Π – Network (2)	14.7 nH	11.7 nH	20.408 ↓
Inductor R_S	28.1 Ω	27.88 Ω	0.783 ↓
Π – Network (2)	4.26 pF	4.791 pF	12.465 ↑

The fabricated components were mostly in a ± 20% range of the target values. This is consistent with analog CMOS design due to the die location on the wafer and other common process variations [7]. All of the capacitors were larger than the design value. This would suggest that the capacitance per unit area for this fabrication run was higher than in previous runs on which

the process parameters were based. Another possibility is that the test setup may be adding some stray capacitance to the value. Each value seems to be consistently high by about 0.3p-0.4p farads, which is shown by the decrease in the error percentage as the values are increased.

The inductors all measured about the same value. Even the 8.04nH inductor measured around 12nH. This was surprising because the inductor layouts were done using ASITIC [5]. These errors could be caused by the test setup. The probe station that was used to test the components was calibrated only with an open-short test. The accuracy would have been increased if a load calibration had been performed. The enhanced calibration would have been completed if the appropriate equipment had been available.

After it was determined by the measurements above that the components were close to the predicted values, the overall networks could be measured. The tests were first performed using the Agilent 8712ET RF Network Analyzer at 915MHz. This test setup can be seen in Appendix D.3. The results obtained from these measurements did not seem to agree with the calculated values. To check the validity of the test setup, the impedances of the matching networks were calculated at a lower frequency. This was done because it was shown in the previous section that this piece of equipment could accurately measure the results for a discrete network at low frequency. The results obtained were invalid. This can be seen by examining the results obtained from the L – Network shown in Figure 6.3. The following figure illustrates in invalid data.

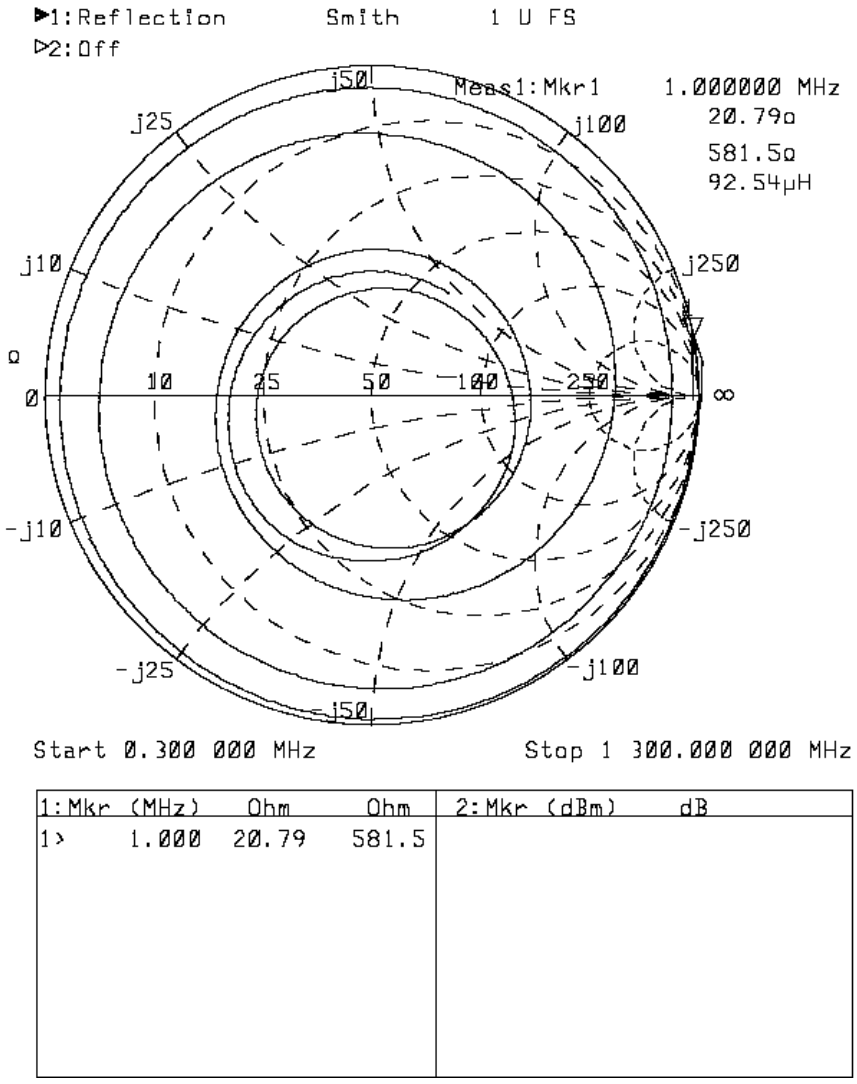


Figure 6.6 – L – Network Results using the Agilent 8712ET

The Smith Chart above shows that the reactance of this network converts to a 92.54µH inductor. This value is extremely large and could not be possible on a small chip. The ideal value showed that the impedance should be around 28.9µ – j79.18k ohms. Using this data and invalid data from the other networks, it was concluded that the ten-inch coaxial cable and coaxial probe were causing large amounts of error in the measurements. This was not the same setup as for the measurements of the discrete network described in the previous section, which explains the valid results obtained.

Because the Agilent 8712ET was the only instrument available to measure at the desired frequency of 915MHz, the scope of the measurements had to be changed. The existing networks

were evaluated at 1MHz and 10MHz. Two LCR meters were used to measure the impedances of the matching networks at these frequencies. The LCR meters utilized a different probe setup than the Agilent Network Analyzer. This setup was the same one used to measure each individual component. It is described in Appendix D.2.

The first set of measurements was done with the substrate floating to characterize the series resistances of the inductors. Without the substrate being grounded, there will be very little coupling to ground meaning these parasitic branches will have negligible effects. The following table summarizes the results. Because the original planned test setup was not used, only the L – Network and Π – Network (1) could be measured. The pad spacing on Π – Network (2) was too close for the four-terminal probes. The pad location requirement is a function of the probe station. All calculations can be seen in Appendix E.

Table 6.4 – Comparison of Measured Values to the Values Calculated using the Predicted Values

@1MHz in ohms	L – Network	Π – Network (1)
Calculated with predicted values, no parasitics	28.9μ - j79.18k	199.922 – j3.936
Calculated with predicted values, with series resistance of inductor	18.759 – j79.18k	213.012 – j4.291
Measured value	-310 – j65.495k	182.29 – j6.55

Examining the results above, it can be seen that the inductor's series resistance mainly affects the resistance of the overall network. It did seem to have an effect on the reactance of the Π – Network, but it was small compared to the resistive change. When comparing the measured and calculated values of the L – Network, it is interesting to see that the measured resistance is negative. Although this is possible with active circuits, the passive devices here cannot generate a negative resistance [10]. Upon speaking to Agilent, it was concluded that the error was caused by the large reactance in the network. Converting the network to its parallel form yields a large resistance of 334MΩ, which is larger than the upper limit of 100MΩ [11]. This means the series resistance of the L – Network cannot be measured with the available equipment. It can, however, be concluded that the resistance is small in comparison to the reactive component.

To get more accurate results, the impedances of the networks were calculated using the measured values for each component. The results are summarized in the following table. All calculations can be seen in Appendix E.

Table 6.5 – Comparison of Measured Values to the Values Calculated using the Predicted and Measured Values

@1MHz in ohms	L – Network	Π – Network (1)
Calculated with predicted values, no parasitics	28.9μ - j79.18k	199.922 – j3.936
Calculated with predicted values, with series resistance of inductor	18.759 – j79.18k	213.012 – j4.291
Calculated with measured values, no parasitics	34.64μ - j65.5k	159.702 – j2.728
Calculated with measured values, with series resistance of inductor	18.208 – j65.5k	172.785 – j3.037
Measured value	-310 – j65.495k	182.29 – j6.55

The calculated values using the measured components were very close to the overall network measurements. The reactance of the L – Network was almost exactly the calculated value. For the Π – Network, the resistive component was only off by five percent while the reactive part seemed to have more variation. The inductor most likely caused this deviation, because it was significantly off the target value of 8.04nH. Although the percentage of error for each component was larger than desired, these results show that a network can be calculated with minimal error as long as accurate component values are know.

To help verify the results, the same process explained previously was performed at 10MHz. The frequency was chosen to be at least an order of magnitude higher than the previous experiment but low enough that the probes would not cause significant error. Each component was again measured individually. The results are shown in the following table.

Table 6.6 – On-Chip Networks and Components at 10MHz

Network/Load	Predicted Value	Measured Value	% Error
Load Resistor	200 Ω	192.074 Ω	3.963 ↓
Load Capacitor	1 pF	1.237 pF	23.7 ↑
L – Network	12.07 nH	2.944 nH	75.609 ↓
Inductor R_S	20.7 Ω	25.455 Ω	22.971 ↑
L – Network	2.01 pF	2.05 pF	1.990 ↑
Π – Network (1)	4.43 pF	4.073 pF	8.059 ↓
Π – Network (1)	8.04 nH	8.975 nH	11.629 ↑
Inductor R_S	13.3 Ω	16.197 Ω	21.782 ↑
Π – Network (1)	10.44 pF	9.314 pF	10.785 ↓
Π – Network (2)	1.61 pF	1.753 pF	8.882 ↑
Π – Network (2)	14.7 nH	3.128 nH	78.721 ↓
Inductor R_S	28.1 Ω	34.4 Ω	22.420 ↑
Π – Network (2)	4.26 pF	4.011 pF	5.845 ↓

The error at 10MHz was much higher than that obtained at 1MHz. This would suggest that the cables and probes are starting to have negative effects on the measurements. To fix this problem, an open-short-load calibration could be performed at the desired frequency. Only the open-short calibration could be performed in this case do to equipment limitations. These results were used to calculate the same values as the 1MHz case. The network impedances are summarized in the following table. These calculations can be seen in Appendix E.

Table 6.7 – Comparison of Measured Values to the Values Calculated using the Predicted and Measured Values

@10MHz in ohms	L – Network	Π – Network (1)
Calculated with predicted values, no parasitics	2.89m – j7.917k	192.474 – j37.888
Calculated with predicted values, with series resistance of inductor	18.762 – j7.918k	204.681 – j41.212
Calculated with measured values, no parasitics	178.1μ – j7.763k	186.44 – j32.208
Calculated with measured values, with series resistance of inductor	22.477 – j7.764k	201.79 – j35.815
Measured value	-629.44 – j7.638k	186.308 – j64.456

The calculated results seen in Table 6.7 are close to the measured values. Again, the resistive part of the L – Network was a negative value meaning it is small compared to the reactive part of

-j7.638k. The reactive part of the Π – Network is off by about fifty percent, which is similar to the results obtained at 1MHz.

To further quantify the effects of the parasitics, the substrate was grounded to include the other parasitic branches. This was done by bonding the ground pad to a ground plane under the chip. The chip was connected to the ground plane using Circuit Works conductive epoxy (CW2400) by Chemtronics. The volume resistivity is given at <0.001 ohm-cm, which provides a low resistive connection [12]. A picture of the bonded chip can be seen in Appendix D.4. The result of these measurements can be seen in the following table.

Table 6.8 – Comparison of Network Impedance for a Floating and Grounded Substrate

@1MHz in ohms	L – Network	Π – Network (1)
Substrate Floating	-310 – j65.495k	182.29 – j6.55
Substrate Grounded	-172.2 – j49.112k	184 – j4.415

The networks were only measured at 1MHz because of limited time available with the higher frequency Agilent LCR Meter. The results obtained showed a decrease in the reactive components with minimal change in the resistance. The reactance of the Π – Network decreased, but it will be shown later that the –j6.55 value is larger than the average for the five chips measured. The average value was –j4.759, which is extremely close to the measured value in Table 6.8, and also the calculated value in Table 6.5. The resistive part of the L – Network was not evaluated for the reason that both numbers are invalid because of their negative nature. For this data, it can be seen that the Π – Network is affected less by the substrate than the L – Network. The main reason for this is the floating capacitor in the L – Network. A capacitor is much closer to the substrate than an inductor. This means its parasitic capacitance to ground will be larger.

One source of error in these measurements is the bond wire used to ground the substrate. It should have minimal effect at lower frequencies, but this test would have to be modified to measure at 915MHz. At this frequency, the wire will have significant reactance since bond wire acts as an inductor [13].

To prove that the substrate parasitics are small in comparison to the series resistance of the inductors, the equivalent impedance of both measured networks was calculated. The results are presented in the following table. As can be seen, the values with and without the substrate grounded are practically equal. This data confirms that the substrate effects are negligible for these networks. These calculations can be seen in Appendix E.

Table 6.9 – Comparison of Network Impedance Including Only the Series Resistance of the Inductor to the Network with all Parasitics Included

@1MHz in ohms	L – Network	Π – Network (1)
Only Inductor Resistance	18.208 – j65.5k	172.785 – j3.037
With all Parasitics	18.208 – j65.5k	172.784 – j3.084

As was stated previously, five chips were measured at 1MHz. This was done to ensure that a substandard chip did not skew the results. The measurements are listed in the following table. The chips were not measured at 10MHz due to limited access to the Agilent LCR Meters.

Table 6.10 – Network Impedances from all the Measured Chips

Chip @1MHz	L – Network	Π – Network (1)
1	-310 – j65.495k	182.29 – j6.55
2	-1.194k – j64.752k	177.097 – j4.979
3	-1.29k – j65.17k	182.837 – j4.152
4	-739 – j64.445k	166.86 – j3.532
5	-1.270k – j65.207k	179.03 – j4.584
Average	-960.6 – j65.014k	177.623 – j4.759

Comparing the data, it can be seen that the negative resistance of the L – Network appears in all the fabricated networks, which means it is small in comparison to the large reactance. The reactive part for the L – Network remained relatively constant for all of the measurements. This means there is low variability between chips for the L – Network. The Π – Network, however, did seem to vary between chips. These variations could be caused by nonuniform oxide thicknesses and etching effects. The average result is very close to the calculated value of 172.785 – j3.037.

7.0 CONCLUSIONS

The theoretical basis for impedance matching has been presented including the fundamental circuit topologies for alternative implementations. This analysis is valid for elements that can be defined as commercially available discrete devices. However, System on a Chip (SoC) implementations of impedance matching circuits require a CMOS realization of inductors and capacitors. These CMOS analog components will have inherent parasitic resistance and capacitance. To show the effects of the parasitics on impedance matching, a group of networks was designed, fabricated, and tested to illustrate the theoretical versus practical consequences of parasitics.

The results obtained in these experiments were used to better understand the effects of parasitics on impedance matching networks. All matching networks suffer from these parasitics, but on-chip networks see the most effect because of the common substrate. To improve the performance, a clear understanding of each component's parasitics must be obtained. This knowledge allows the designer to manipulate the parasitic values to his or her benefit. As an example, the evaluated Π – Network shown in Table 6.8 was not affected by the inductor's two parasitic branches to ground, because the capacitor values of X_1 and X_3 were chosen to be approximately a factor of ten larger. From this, it can be seen that the effects caused by the parasitic branches to ground can be minimized by sizing the components appropriately. For the case where a component must have a value comparable to this branch, the parasitic capacitance can actually be used as part of the matching element. When the network requires a capacitor, the two values will be in parallel which means they simply add, and the required component capacitance is designed at a lesser value. When an inductor is needed, the component value can easily be calculated so it cancels the parasitic capacitance to ground and yields an equivalent value equal to the desired component.

As was just explained, generally the parasitic branches to ground can be neglected. The most dominant effect is caused by the series resistance of the inductor. This value is easily obtained using ASITIC as was explained in Subsection 4.1.1. Unlike the parasitic branches to ground, there is no convenient way to include or cancel this value with a modified design. The physical nature of a resistor is to dissipate power, which is the exact opposite of the overall goal. As was

shown in the software example of Section 5.2, the series resistance is almost entirely responsible for power loss in the circuit. The inductor not only dissipates power but also causes reflection of power by creating an impedance mismatch.

There is a massive amount of research being done to improve the quality factor of inductors by optimizing the layout to minimize the series resistance [9], [14], [15]. These processes range from simply adjusting the trace width to removing the substrate under the inductor. Either way, the qualify factor of on-chip inductors is still much less than that of discrete components. The best way to deal with this problem is to use all-capacitor matching.

All-capacitor matching can be performed when certain criteria are meet. The criteria can be found by examining the network equations and developing a system of equalities. Obviously, these criteria will depend mostly on the source and load impedances. This means the system must have a certain impedance, or this technique will not work. The use of all-capacitor matching is a valuable method but not practical in most applications.

Overall, this research has successfully demonstrated that the parasitics of a particular network can be obtained using analysis tools and hand calculations. The measured results were able to verify the validity of these calculations within the limitations of the test equipment. Despite these limitations, the research presented has proven the described concepts in the low mega hertz frequency range. These results could be extended to include the entire frequency spectrum with the addition of higher frequency test equipment. The presented findings, however, provide valuable information that can be used to design more robust networks for impedance matching applications.

8.0 FUTURE DIRECTIONS

8.1 Research Continuation

The overall goal of this research was to gain insight into parasitic effects on impedance matching networks. This research has demonstrated that it is possible to calculate and include these effects at frequencies around 1MHz. The original objective was to quantify the parasitics at a frequency of 915MHz. Due to limitations in the available test equipment, only theoretical solutions could be achieved at 915MHz. This research has, however, provided valuable information that could be used to continue quantifying the effect of parasitics at frequencies around 1GHz. At these frequencies, component placement also becomes an important consideration. The availability of a chip simulator would be a valuable resource in designing high-speed analog matching networks.

8.2 Chip Simulator

As computers become faster and more powerful, the ability to simulate an entire chip will be feasible. ASITIC currently evaluates a metal structure at a given frequency and uses a two-port network to describe its behavior. This network, however, is only valid at that frequency and is not easily scalable since it is a nonlinear system. To solve this problem, ASITIC can simply perform a frequency sweep to better characterize the structure. This is a minor step in obtaining a complete chip simulator. The difficult question is, "How do different structures interact with each other?" Then the problem becomes the massive amount of data and computational time. What data are negligible, and what data must be considered? Obviously, the data must be categorized as significant or negligible otherwise the computations would be overwhelming. These are only a few questions that must be answered before a complete chip simulator can be constructed.

8.3 Complete System on a Chip

The ultimate goal is to construct a complete System on a Chip (SoC). As an example, it would be desirable in Radio Frequency Identification (RFID) applications to integrate all components including the antenna onto a single chip. By including the antenna on the chip, you minimize manufacturing time and cost, but you also lose the ability to completely characterize the antenna with existing design automation tools. This poses a problem for impedance matching and also

component placement. Does an inductor near the antenna affect its performance? Obviously, it will, but at what distance do the effects diminish? These are questions that could be answered with a complete chip simulator. By answering these difficult questions, we move one step closer to the goal of a complete system on a chip.

APPENDICES

Appendix A – Graph Code

Matlab Code used for Graphs 2.1, 2.3, & 2.4

```
%Charlie Greene

%Graph 2.1
clear all
close all
home

i=sqrt(-1);
zs=50-i*1592;
rl=[1:500];
figure(1)

for xl=-3006:22:3000
   for rd=1:500;
      pout(rd)=0.5/10^-3*((1/(zs+rd+xl*i)).^2)*rd;
   end
   if xl==1592
      plot(rl,abs(pout),'r','LineWidth',2)
   else
   plot(rl,abs(pout))
   hold on
   end
   title('Output Power as a Function of Load Resistance and Reactance')
   ylabel('Output Power (mW)')
   xlabel('Resistance (ohms)')
end

%Graph 2.3
clear all
close all
home

i=sqrt(-1);
zs=50-i*1592;
rl=[1:2000];
figure(1)

for xl=-2504:128:2504
   for rd=1:2000;
      vout(rd)=rd./(rd+i*xl+zs);
   end
   if xl==1592
      plot(rl,abs(vout),'r','LineWidth',2)
```

```
    else
    plot(rl,abs(vout))
    hold on
    end
    title('Resistive Load Voltage as a Function of Load Resistance and Reactance')
    ylabel('Voltage Across Load Resistor (V)')
    xlabel('Resistance (ohms)')
end

%Graph 2.4
clear all
close all
home

i=sqrt(-1);
zs=50-i*1592;
rl=[1:200];
figure(1)

for xl=-2504:128:2504
    for rd=1:200;
        vout(rd)=(rd+i*xl)./(rd+i*xl+zs);
    end
    if xl==1592
        plot(rl,abs(vout),'r','LineWidth',2)
    else
    plot(rl,abs(vout))
    hold on
    end
    title('Load Voltage as a Function of Load Resistance and Reactance')
    ylabel('Voltage Across Load (V)')
     xlabel('Resistance (ohms)')
end
```

Appendix B – Impedance Transformation Example

L – Network Special Case: Upward and Downward Transformation

Transform a 1+j1 ohm load to match a source impedance of 1.5 ohms.

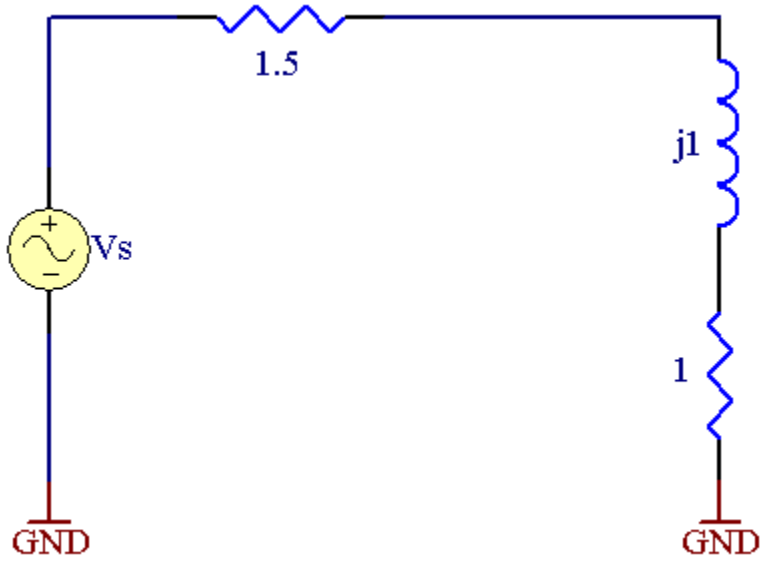


Figure B.1 – Circuit Diagram

Solution:

Downward transformation

The load in its parallel format is 2 // j2 (see Section 2.3). This means the load resistance is greater than the 1.5-ohm resistance of the driving stage, and a downward transformation should be used. The variables for this problem have the following values.

$$R = 2\Omega$$
$$X = 2\Omega$$
$$R_{IN} = 1.5\Omega$$
$$X_{IN} = 0$$

Using the equations given in Subsection 3.1.1:

$$Q = \pm\sqrt{\frac{R}{R_{IN}} - 1} = \pm\sqrt{\frac{2}{1.5} - 1} = \pm 0.577$$

Choosing a positive Q-value yields:

$$X_1 = \frac{X \cdot R}{Q \cdot X - R} = \frac{2 \cdot 2}{0.577 \cdot 2 - 2} = -4.728\Omega$$

$$X_2 = -Q \cdot R_{IN} = -0.577 \cdot 1.5 = -0.866\Omega$$

Simplifying the network as a double check:

$$Z_{IN} = (R \,//\, jX \,//\, jX_1) + jX_2 = (1/R + 1/(jX) + 1/(jX_1))^{-1} + jX_2$$
$$Z_{IN} = (1/2 + 1/(j2) + 1/(-j4.728))^{-1} - j0.866$$
$$Z_{IN} = (1.5 + j0.866) - j0.866$$
$$Z_{IN} = 1.5\Omega \quad \leftarrow \textit{Checks}$$

The completed circuit is given in the following figure.

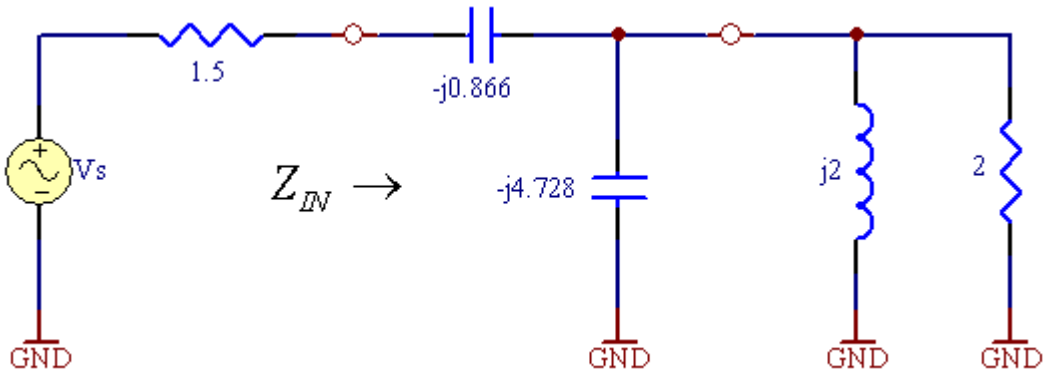


Figure B.2 – Completed Circuit with Matching Network

Upward Transformation

The load impedance in its series form is 1+j1. This means that the load resistance is less than the desired impedance of 1.5 ohms, and an upward transformation should be used. The variables for this problem have the following values.

$$R = 1\Omega$$
$$X = 1\Omega$$
$$R_{IN} = 1.5\Omega$$
$$X_{IN} = \infty$$

Using the equations given in Subsection 3.1.2:

$$Q = \pm\sqrt{\frac{R_{IN}}{R} - 1} = \pm\sqrt{\frac{1.5}{1} - 1} = \pm 0.707$$

Selecting a positive Q-value yields:

$$X_1 = Q \cdot R - X = 0.707 \cdot 1 - 1 = -0.293\Omega$$

$$X_2 = -\frac{R_{IN}}{Q} = -\frac{1.5}{0.707} = -2.122\Omega$$

Simplifying the network as a double check:

$$Z_{IN} = (R + jX + jX_1) // jX_2 = (1 + j1 - j0.293) // -j2.122$$
$$Z_{IN} = (1 + j0.707) // -j2.122 = \frac{(1 + j0.707) \cdot -j2.122}{1 + j0.707 - j2.122}$$
$$Z_{IN} = 1.5\Omega \quad \leftarrow \textit{Checks}$$

The completed circuit is shown in the following figure.

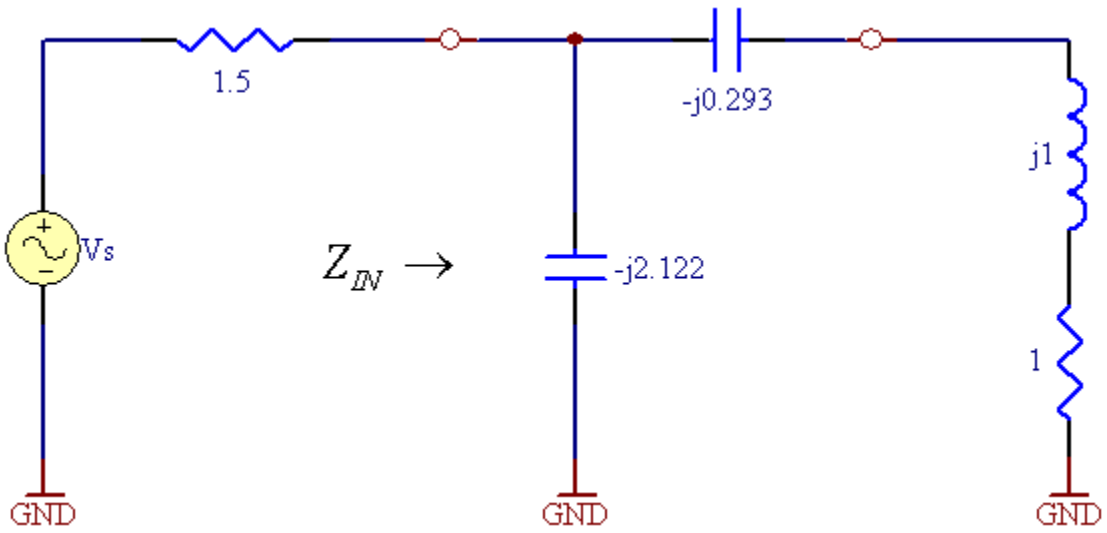


Figure B.3 – Completed Circuit with Matching Network

As shown above, both the downward and upward transformation will produce a valid matching network. This is a special case and will not always be possible. If the transformation Q has a complex value, the wrong transformation direction has been chosen and the other must be used.

Appendix C – Impedance Matching Software

C++ Software Code

```
/*                      Impedance Matching Software
                        Charlie Greene
                        University of Pittsburgh
                        9-17-02                                                 */

#include <complex>
#include <iostream>
#include <stdio.h>
#include <math.h>
using namespace std;


//Global Variables
double rin=0,xin=0,rout=0,xout=0;
double rinL=0,xinL=0,routL=0,xoutL=0;
double rinP=0,xinP=0,routP=0,xoutP=0;
double q1min=0,q2min=0;
double f=0,p=0,n=0;
double imp=0;
double q1=0,q2=0,rp=0;
double x1=0,y2=0,x2=0,x3=0;
double z1=0,z2=0,z3=0;
double rinp=0,xinp=0,routp=0,xoutp=0;
double r11=0,rs11=0,rs21=0,cs11=0,cs21=0,L11=0,C11=0;
double r22=0,rs12=0,rs22=0,cs12=0,cs22=0,L22=0,C22=0;
double r33=0,rs13=0,rs23=0,cs13=0,cs23=0,L33=0,C33=0;
double I=0,max_power=0,power=0;
double x1L,x2L,q=0;
double x1A,x2A,x1B,x2B,x1C,x2C,x3C,x1D,x2D,x3D,x1E,x2E,x3E;
double x1F,x2F,x3F,x1G,x2G,x3G,x1H,x2H,x3H,x1I,x2I,x3I;
double x1J,x2J,x3J;
char ans;
int i=0;
std::complex<double> pzt(0,0);


void print(char type,double x1,double x2,double x3){
        if(x1>=0){
                cout<<"X1 j("<<x1<<") is an inductor - "<<(x1/(2.0*3.14159*f))<<" H"<<endl;
        }
        else{
                x1=-x1;
                cout<<"X1 j("<<-x1<<") is an capacitor - "<<(1/(2.0*3.14159*f*x1))<<" F"<<endl;
                x1=-x1;
        }
        if(x2>=0){
                cout<<"X2 j("<<x2<<") is an inductor - "<<(x2/(2.0*3.14159*f))<<" H"<<endl;
        }
        else{
                x2=-x2;
                cout<<"X2 j("<<-x2<<") is an capacitor - "<<(1/(2.0*3.14159*f*x2))<<" F"<<endl;
                x2=-x2;
        }
```

```
	if(type=='p' || type=='t'){
		if(x3>=0){
			cout<<"X3 j("<<x3<<") is an inductor - "<<(x3/(2.0*3.14159*f))<<" H"<<endl;
		}
		else{
			x3=-x3;
			cout<<"X3 j("<<-x3<<") is an capacitor - "<<(1/(2.0*3.14159*f*x3))<<" F"<<endl;
			x3=-x3;
		}
	}
	cout<<endl;
}

void ldown_cal(double &x1, double &x2){

	if(xoutL<=0.000000001 && xoutL>=-0.000000001){
		x1=routL/q;
	}
	else{
		x1=xoutL*routL/(q*xoutL-routL);
	}

	if(xinL<=0.000000001 && xinL>=-0.000000001){
		x2=-q*rinL;
	}
	else{
		x2=xinL-q*rinL;
	}
}

void lup_cal(double &x1, double &x2){
	if(xoutL<=0.000000001 && xoutL>=-0.000000001){
		x1=q*routL;
	}
	else{
		x1=q*routL-xoutL;
	}

	if(xinL<=0.000000001 && xinL>=-0.000000001){
		x2=-rinL/q;
	}
	else{
		x2=xinL*rinL/(rinL-q*xinL);
	}
}

void ldown(){
	cout<<endl;
	cout<<"----------------------------L - Network----------------------------"<<endl<<endl;
	cout<<"                    ----                              "<<endl;
	cout<<" Zin O-------| X2 |-------o---------O Zout"<<endl;
	cout<<"                    ----          |                   "<<endl;
	cout<<"                                  |                   "<<endl;
	cout<<"                                 ---                  "<<endl;
	cout<<"                                |X1 |                 "<<endl;
	cout<<"                                 ---                  "<<endl;
	cout<<"                                  |                   "<<endl;
	cout<<"                                  |                   "<<endl;
	cout<<"                                 ---                  "<<endl;
	cout<<"                                GND                   "<<endl<<endl;

	rinL=rin;
	xinL=xin;
	routL=(rout*rout+xout*xout)/rout;
	xoutL=xout;
	if(xoutL>=0.000000001 || xoutL<=-0.000000001){                    //double never exactly zero
		xoutL=(rout*rout+xout*xout)/xoutL;
	}

	q=sqrt(routL/rinL-1);
```

```
		cout<<"The |Q| must be "<<q<<endl<<endl;
		cout<<"----A---- L-Network 1 - Postive Q"<<endl<<endl;

		ldown_cal(x1A,x2A);

		print('l',x1A,x2A,0);

		cout<<"----B---- L-Network 2 - Negative Q"<<endl<<endl;
		q=-q;

		ldown_cal(x1B,x2B);

		print('l',x1B,x2B,0);

}

void lup(){
		cout<<endl;
		cout<<"----------------------------L - Network----------------------------"<<endl<<endl;
		cout<<"                              ----                    "<<endl;
		cout<<" Zin O--------o-------| X1 |-------O Zout"<<endl;
		cout<<"                       |      ----                    "<<endl;
		cout<<"                       |                              "<<endl;
		cout<<"                      ---                             "<<endl;
		cout<<"                     |X2 |                            "<<endl;
		cout<<"                      ---                             "<<endl;
		cout<<"                       |                              "<<endl;
		cout<<"                       |                              "<<endl;
		cout<<"                      ---                             "<<endl;
		cout<<"                     GND                              "<<endl<<endl;

		rinL=(rin*rin+xin*xin)/rin;
		xinL=xin;
		routL=rout;
		xoutL=xout;
		if(xinL>=0.000000001 || xinL<=-0.000000001){
			xinL=(rin*rin+xin*xin)/xin;
		}

		q=sqrt(rinL/routL-1);
		cout<<"The |Q| must be "<<q<<endl<<endl;
		cout<<"----A---- L-Network 1 - Postive Q"<<endl<<endl;

		lup_cal(x1A,x2A);

		print('l',x1A,x2A,0);

		cout<<"----B---- L-Network 2 - Negative Q"<<endl<<endl;
		q=-sqrt(rinL/routL-1);

		lup_cal(x1B,x2B);

		print('l',x1B,x2B,0);
}

void tpic(){
		cout<<endl<<endl;
		cout<<"---------------------------T - Network----------------------------"<<endl<<endl;
		cout<<"                    ----          ----                    "<<endl;
		cout<<" Zin O-------| X3 |------o------| X1 |-------O Zout"<<endl;
		cout<<"                    ----    |     ----                    "<<endl;
		cout<<"                            |                             "<<endl;
		cout<<"                           ---                            "<<endl;
		cout<<"                          |X2 |                           "<<endl;
		cout<<"                           ---                            "<<endl;
		cout<<"                            |                             "<<endl;
		cout<<"                            |                             "<<endl;
		cout<<"                           ---                            "<<endl;
		cout<<"                          GND                             "<<endl<<endl;
}
```

```
void t_cal(double q1,double q2,double &x1T,double &x2T,double &x3T){
	x1T=q1*rout-xout;
	x2T=-rp/(q1+q2);
	x3T=q2*rin+xin;
}

void tdown(){

	tpic();

	do{
		q2min=sqrt(rout/rin-1);
		cout<<"|Q2| must be >= to "<<q2min<<endl<<endl;
		cout<<"Enter |Q2| value ";
		cin>>q2;
		rp=rin*(1+q2*q2);
		if(q2>=q2min){
			q1=sqrt(rp/rout-1);
			cout<<"Is a |Q1| of "<<q1<<" acceptable? (y/n) ";
			cin>>ans;
			cout<<endl;
		}
		else{
			cout<<"Q2 too low"<<endl;
			ans='n';
		}
	}
	while(ans=='n');

	t_cal(q1,q2,x1C,x2C,x3C);
	cout<<"----C---- T-Network 1 - Postive Q1, Positive Q2"<<endl<<endl;
	print('t',x1C,x2C,x3C);

	t_cal(-q1,q2,x1D,x2D,x3D);
	cout<<"----D---- T-Network 2 - Negative Q1, Positive Q2"<<endl<<endl;
	print('t',x1D,x2D,x3D);

	t_cal(q1,-q2,x1E,x2E,x3E);
	cout<<"----E---- T-Network 3 - Postive Q1, Negative Q2"<<endl<<endl;
	print('t',x1E,x2E,x3E);

	t_cal(-q1,-q2,x1F,x2F,x3F);
	cout<<"----F---- T-Network 4 - Negative Q1, Negative Q2"<<endl<<endl;
	print('t',x1F,x2F,x3F);

}

void tup(){

	tpic();

	do{
		q1min=sqrt(rin/rout-1);
		cout<<"|Q1| must be >= to "<<q1min<<endl<<endl;
		cout<<"Enter |Q1| value ";
		cin>>q1;
		rp=rout*(1+q1*q1);
		if(q1>=q1min){
			q2=sqrt(rp/rin-1);
			cout<<"Is a |Q2| of "<<q2<<" acceptable? (y/n) ";
			cin>>ans;
			cout<<endl;
		}
		else{
			cout<<"Q1 too low"<<endl;
			ans='n';
		}
	}
	while(ans=='n');
```

```
	t_cal(q1,q2,x1C,x2C,x3C);
	cout<<"----C---- T-Network 1 - Postive Q1, Positive Q2"<<endl<<endl;
	print('t',x1C,x2C,x3C);

	t_cal(-q1,q2,x1D,x2D,x3D);
	cout<<"----D---- T-Network 2 - Negative Q1, Positive Q2"<<endl<<endl;
	print('t',x1D,x2D,x3D);

	t_cal(q1,-q2,x1E,x2E,x3E);
	cout<<"----E---- T-Network 3 - Postive Q1, Negative Q2"<<endl<<endl;
	print('t',x1E,x2E,x3E);

	t_cal(-q1,-q2,x1F,x2F,x3F);
	cout<<"----F---- T-Network 4 - Negative Q1, Negative Q2"<<endl<<endl;
	print('t',x1F,x2F,x3F);

}

void ppic(){
	cout<<endl<<endl;
	cout<<"----------------------------PI - Network----------------------------"<<endl<<endl;
	cout<<"                              ----                              "<<endl;
	cout<<" Zin O--------o-------| X2 |-------o---------O Zout"<<endl;
	cout<<"                      |    ----    |                      "<<endl;
	cout<<"                      |            |                      "<<endl;
	cout<<"                     ---          ---                     "<<endl;
	cout<<"                    |X3 |        |X1 |                    "<<endl;
	cout<<"                     ---          ---                     "<<endl;
	cout<<"                      |            |                      "<<endl;
	cout<<"                      |            |                      "<<endl;
	cout<<"                     ---          ---                     "<<endl;
	cout<<"                    GND          GND                     "<<endl<<endl;
}

void p_cal(double q1,double q2,double &x1P,double &x2P,double &x3P){
	if(xoutP<=0.000000001 && xoutP>=-0.000000001){
		x1P=-routP/q1;
	}
	else{
		x1P=-xoutP*routP/(q1*xoutP+routP);
	}

	x2P=rp*(q1+q2);

	if(xinP<=0.000000001 && xinP>=-0.000000001){
		x3P=-rinP/q2;
	}
	else{
		x3P=xinP*rinP/(rinP-q2*xinP);
	}

}

void pidown(){
	ppic();

	rinP=(rin*rin + xin*xin)/rin;
	xinP=xin;
	if(xinP>=0.000000001 || xinP<=-0.000000001){
		xinP=(rin*rin + xin*xin)/xin;
	}

	routP=(rout*rout + xout*xout)/rout;
	xoutP=xout;
	if(xoutP>=0.000000001 || xoutP<=-0.000000001){
		xoutP=(rout*rout + xout*xout)/xout;
	}

	do{
```

```
                q1min=sqrt(routP/rinP-1);
                cout<<"|Q1| must be >= to "<<q1min<<endl<<endl;
                cout<<"Enter |Q1| value ";
                cin>>q1;
                rp=routP/(1+q1*q1);
                if(q1>=q1min){
                        q2=sqrt(rinP/rp-1);
                        cout<<"Is a |Q2| of "<<q2<<" acceptable? (y/n) ";
                        cin>>ans;
                        cout<<endl;
                }
                else{
                        cout<<"Q1 too low"<<endl;
                        ans='n';
                }
        }
        while(ans=='n');

        p_cal(q1,q2,x1G,x2G,x3G);
        cout<<"----G---- PI-Network 1 - Postive Q1, Positive Q2"<<endl<<endl;
        print('p',x1G,x2G,x3G);

        p_cal(-q1,q2,x1H,x2H,x3H);
        cout<<"----H---- PI-Network 2 - Negative Q1, Positive Q2"<<endl<<endl;
        print('p',x1H,x2H,x3H);

        p_cal(q1,-q2,x1I,x2I,x3I);
        cout<<"----I---- PI-Network 3 - Postive Q1, Negative Q2"<<endl<<endl;
        print('p',x1I,x2I,x3I);

        p_cal(-q1,-q2,x1J,x2J,x3J);
        cout<<"----J---- PI-Network 4 - Negative Q1, Negative Q2"<<endl<<endl;
        print('p',x1J,x2J,x3J);

}

void piup(){
        ppic();

        rinP=(rin*rin + xin*xin)/rin;
        xinP=xin;
        if(xinP>=0.000000001 || xinP<=-0.000000001){
                xinP=(rin*rin + xin*xin)/xin;
        }

        routP=(rout*rout + xout*xout)/rout;
        xoutP=xout;
        if(xoutP>=0.000000001 || xoutP<=-0.000000001){
                xinP=(rout*rout + xout*xout)/xout;
        }

        do{
                q2min=sqrt(rinP/routP-1);
                cout<<"|Q2| must be >= to "<<q2min<<endl<<endl;
                cout<<"Enter |Q2| value ";
                cin>>q2;
                rp=rinP/(1+q2*q2);
                if(q2>=q2min){
                        q1=sqrt(routP/rp-1);
                        cout<<"Is a |Q1| of "<<q1<<" acceptable? (y/n) ";
                        cin>>ans;
                        cout<<endl;
                }
                else{
                        cout<<"Q2 too low"<<endl;
                        ans='n';
                }
        }
        while(ans=='n');
```

```
		p_cal(q1,q2,x1G,x2G,x3G);
		cout<<"----G---- PI-Network 1 - Postive Q1, Positive Q2"<<endl<<endl;
		print('p',x1G,x2G,x3G);

		p_cal(-q1,q2,x1H,x2H,x3H);
		cout<<"----H---- PI-Network 2 - Negative Q1, Positive Q2"<<endl<<endl;
		print('p',x1H,x2H,x3H);

		p_cal(q1,-q2,x1I,x2I,x3I);
		cout<<"----I---- PI-Network 3 - Postive Q1, Negative Q2"<<endl<<endl;
		print('p',x1I,x2I,x3I);

		p_cal(-q1,-q2,x1J,x2J,x3J);
		cout<<"----J---- PI-Network 4 - Negative Q1, Negative Q2"<<endl<<endl;
		print('p',x1J,x2J,x3J);

}

void l_par_down(double x1,double x2){
		if(x1>=0){
				cout<<"Enter the follow parameters for the inductor X1"<<endl;
				cout<<"R = ";
				cin>>r11;
				cout<<"L (nH) = ";
				cin>>L11;
				cout<<"Rs1 = ";
				cin>>rs11;
				cout<<"Cs1 (pF) = ";
				cin>>cs11;
			}
			else{
				cout<<"Enter the follow parameters for the capacitor X1"<<endl;
				cout<<"R = ";
				cin>>r11;
				cout<<"C (pF) = ";
				cin>>C11;
				cout<<"Rt = ";
				cin>>rs11;
				cout<<"Ct (pF) = ";
				cin>>cs11;
			}
			if(x2>=0){
				cout<<"Enter the follow parameters for the inductor X2"<<endl;
				cout<<"R = ";
				cin>>r22;
				cout<<"L (nH) = ";
				cin>>L22;
				cout<<"Rs1 = ";
				cin>>rs12;
				cout<<"Cs1 (pF) = ";
				cin>>cs12;
				cout<<"Rs2 = ";
				cin>>rs22;
				cout<<"Cs2 (pF) = ";
				cin>>cs22;
			}
			else{
				cout<<"Enter the follow parameters for the capacitor X2"<<endl;
				cout<<"R = ";
				cin>>r22;
				cout<<"C (pF) = ";
				cin>>C22;
				cout<<"Rt = ";
				cin>>rs12;
				cout<<"Ct (pF) = ";
				cin>>cs12;
				cout<<"Rb = ";
				cin>>rs22;
				cout<<"Cb (pF) = ";
				cin>>cs22;
```

```
}

//Calculate Mismatch for L-Network Downward Transformation

n=0.000000001;
p=0.000000000001;
L11=L11*n;
L22=L22*n;
C11=C11*p;
C22=C22*p;
cs11=cs11*p;
cs21=cs21*p;
cs12=cs12*p;
cs22=cs22*p;

std::complex<double> x2s1(rs12,-imp/cs12), x1s1(rs11,-imp/cs11), nz2(r22,x2);
std::complex<double> zout(rout,xout), x2s2(rs22,-imp/cs22), nz1(r11,x1);
std::complex<double> zright=1.0/((1.0/x2s2)+(1.0/x1s1)+(1.0/nz1)+(1.0/zout));
std::complex<double> zeq1=zright+nz2;
std::complex<double> ztotal=zeq1*x2s1/(zeq1+x2s1);

//Ideal Calculations
std::complex<double> lzout(rout,xout), lz1(0,x1), lz2(0,x2);
std::complex<double> lzt=lzout*lz1/(lzout+lz1)+lz2;

cout<<"The old load including the ideal L - matching network was"<<endl<<endl;
cout<<"        Z = "<<std::real(lzt)<<" + j("<<std::imag(lzt)<<")"<<endl<<endl;

cout<<"The load including the parasitics of the L - matching network is"<<endl<<endl;
cout<<"        Z = "<<std::real(ztotal)<<" + j("<<std::imag(ztotal)<<")"<<endl<<endl;

//Original input impedance
cout<<"The input impedance was given or calculated to be"<<endl<<endl;
cout<<"        Z = "<<rin<<" + j("<<-xin<<")"<<endl<<endl;

//Power Transferred
std::complex<double> zin(rin,-xin);
I=1.0/(2*rin);
max_power=I*I*rin/2;
std::complex<double> I=1.0/(zin+ztotal);
std::complex<double> power=std::abs(I)*std::abs(I)*std::real(ztotal)/2.0/max_power*100;

//Calculate branch currents
std::complex<double> Ig1=zeq1/(zeq1+x2s1)*I;
std::complex<double> Ix2=x2s1/(zeq1+x2s1)*I;
std::complex<double> Ig2=(1.0/(1.0/zout+1.0/nz1))/((1.0/(1.0/x1s1+1.0/x2s2))+(1.0/(1.0/zout+1.0/nz1)))*Ix2;
std::complex<double> Ix1=(1.0/(1.0/zout+1.0/x2s2+1.0/x1s1))/((1.0/(1.0/zout+1.0/x2s2+1.0/x1s1))+nz1)*Ix2;
std::complex<double> Ir=(1.0/(1.0/nz1+1.0/x2s2+1.0/x1s1))/((1.0/(1.0/nz1+1.0/x2s2+1.0/x1s1))+zout)*Ix2;

//Power Calculations
std::complex<double> Pr=std::abs(Ir)*std::abs(Ir)*std::real(zout)/2.0/max_power*100;
std::complex<double> Pg1=std::abs(Ig1)*std::abs(Ig1)*std::real(x2s1)/2.0/max_power*100;
std::complex<double> Pg2=std::abs(Ig2)*std::abs(Ig2)*std::real(1.0/(1.0/x2s2+1.0/x1s1))/2.0/max_power*100;
std::complex<double> Pg=Pg1+Pg2;
std::complex<double> Ps1=std::abs(Ix1)*std::abs(Ix1)*std::real(nz1)/2.0/max_power*100;
std::complex<double> Ps2=std::abs(Ix2)*std::abs(Ix2)*std::real(nz2)/2.0/max_power*100;
std::complex<double> Ps=Ps1+Ps2;
std::complex<double> Max(100,0);
std::complex<double> Pref=Max-power;
std::complex<double> Pt=Pref+Pg+Ps+Pr;

//Unmatched Power Calculations
std::complex<double> Iu=1.0/(zin+zout);
std::complex<double> Pu=std::abs(Iu)*std::abs(Iu)*std::real(zout)/2.0/max_power*100;

//Display Results
cout<<"Amount of Maximum Power Transferred  --> "<<std::real(power)<<"%"<<endl<<endl<<endl;
cout<<"-----------------------POWER BREAKDOWN-------------------------"<<endl<<endl;
cout<<"Power Reflected                  --> "<<std::real(Pref)<<"%"<<endl<<endl;
cout<<"Power Lost to Ground             --> "<<std::real(Pg)<<"%"<<endl<<endl;
```

```
		cout<<"Power Lost in Series Resistance	--> "<<std::real(Ps)<<"%"<<endl<<endl;
		cout<<"Power Received at the Load	--> "<<std::real(Pr)<<"%"<<endl<<endl;
		cout<<"				----------"<<endl;
		cout<<"			Total	"<<std::real(Pt)<<"%"<<endl<<endl;
		cout<<"Power Received at the Load w/o match --> "<<std::real(Pu)<<"%"<<endl<<endl;

		if(std::real(Pu)>=std::real(Pr)){
			cout<<"Matching Network Degrades Performance -- DO NOT USE NETWORK"<<endl<<endl;
		}
		else{
			cout<<"Matching Network Improves Performance"<<endl<<endl;
		}

}

void l_par_up(double x1,double x2){
		if(x1>=0){
			cout<<"Enter the following parameters for the inductor X1"<<endl;
			cout<<"R = ";
			cin>>r11;
			cout<<"L (nH) = ";
			cin>>L11;
			cout<<"Rs1 = ";
			cin>>rs11;
			cout<<"Cs1 (pF) = ";
			cin>>cs11;
			cout<<"Rs2 = ";
			cin>>rs21;
			cout<<"Cs2 (pF) = ";
			cin>>cs21;
		}
		else{
			cout<<"Enter the following parameters for the capacitor X1"<<endl;
			cout<<"R = ";
			cin>>r11;
			cout<<"C (pF) = ";
			cin>>C11;
			cout<<"Rt = ";
			cin>>rs11;
			cout<<"Ct (pF) = ";
			cin>>cs11;
			cout<<"Rb = ";
			cin>>rs21;
			cout<<"Cb (pF) = ";
			cin>>cs21;
		}
		if(x2<=0){
			cout<<"Enter the following parameters for the capacitor X2"<<endl;
			cout<<"R = ";
			cin>>r22;
			cout<<"C (pF) = ";
			cin>>C22;
			cout<<"Rt = ";
			cin>>rs12;
			cout<<"Ct (pF) = ";
			cin>>cs12;
		}
		else{
			cout<<"Enter the following parameters for the inductor X2"<<endl;
			cout<<"R = ";
			cin>>r22;
			cout<<"L (nH) = ";
			cin>>L22;
			cout<<"Rs1 = ";
			cin>>rs12;
			cout<<"Cs1 (pF) = ";
			cin>>cs12;
		}

		//Calculate Mismatch for L-Network Upward Transformation
```

```
n=0.000000001;
p=0.000000000001;
L11=L11*n;
L22=L22*n;
C11=C11*p;
C22=C22*p;
cs11=cs11*p;
cs21=cs21*p;
cs12=cs12*p;
cs22=cs22*p;

std::complex<double> x2s1(rs12,-imp/cs12), x1s1(rs11,-imp/cs11), nz2(r22,x2);
std::complex<double> zout(rout,xout), x1s2(rs21,-imp/cs21), nz1(r11,x1);
std::complex<double> zright=(1.0/(1.0/zout+1.0/x1s2))+nz1;
std::complex<double> zleft=(1.0/(1.0/nz2+1.0/x1s1+1.0/x2s1));
std::complex<double> ztotal=1.0/(1.0/zright+1.0/nz2+1.0/x2s1+1.0/x1s1);

//Ideal Calculations
cout<<rout<<" "<<xout<<" "<<x1<<" "<<x2<<endl;
std::complex<double> lzout(rout,xout), lz1(0,x1), lz2(0,x2);
std::complex<double> lzt=(lzout+lz1)*lz2/(lzout+lz1+lz2);

cout<<"The old load including the ideal L - matching network was"<<endl<<endl;
cout<<"        Z = "<<std::real(lzt)<<" + j("<<std::imag(lzt)<<")"<<endl<<endl;

cout<<"The load including the parasitics of the L - matching network is"<<endl<<endl;
cout<<"        Z = "<<std::real(ztotal)<<" + j("<<std::imag(ztotal)<<")"<<endl<<endl;

//Original input impedance
cout<<"The input impedance was given or calculated to be"<<endl<<endl;
cout<<"        Z = "<<rin<<" + j("<<-xin<<")"<<endl<<endl;

//Power Transferred
std::complex<double> zin(rin,-xin);
I=1.0/(2*rin);
max_power=I*I*rin/2;
std::complex<double> I=1.0/(zin+ztotal);
std::complex<double> power=std::abs(I)*std::abs(I)*std::real(ztotal)/2.0/max_power*100;

//Calculate branch currents
std::complex<double> Ig1=(1.0/(1.0/nz2+1.0/zright))/((1.0/(1.0/x1s1+1.0/x2s1))+(1.0/(1.0/nz2+1.0/zright)))*I;
std::complex<double> Ix2=(1.0/(1.0/x1s1+1.0/x2s1+1.0/zright))/((1.0/(1.0/x1s1+1.0/x2s1+1.0/zright))+nz2)*I;
std::complex<double> Ix1=zleft/(zright+zleft)*I;
std::complex<double> Ig2=zout/(zout+x1s2)*Ix1;
std::complex<double> Ir=x1s2/(zout+x1s2)*Ix1;

//Power Calculations
std::complex<double> Pr=std::abs(Ir)*std::abs(Ir)*std::real(zout)/2.0/max_power*100;
std::complex<double> Pg1=std::abs(Ig1)*std::abs(Ig1)*std::real(1.0/(1.0/x1s1+1.0/x2s1))/2.0/max_power*100;
std::complex<double> Pg2=std::abs(Ig2)*std::abs(Ig2)*std::real(x1s2)/2.0/max_power*100;
std::complex<double> Pg=Pg1+Pg2;
std::complex<double> Ps1=std::abs(Ix1)*std::abs(Ix1)*std::real(nz1)/2.0/max_power*100;
std::complex<double> Ps2=std::abs(Ix2)*std::abs(Ix2)*std::real(nz2)/2.0/max_power*100;
std::complex<double> Ps=Ps1+Ps2;
std::complex<double> Max(100,0);
std::complex<double> Pref=Max-power;
std::complex<double> Pt=Pref+Pg+Ps+Pr;

//Unmatched Power Calculations
std::complex<double> Iu=1.0/(zin+zout);
std::complex<double> Pu=std::abs(Iu)*std::abs(Iu)*std::real(zout)/2.0/max_power*100;

//Display Results
cout<<"Amount of Maximum Power Transferred  --> "<<std::real(power)<<"%"<<endl<<endl<<endl;
cout<<"----------------------POWER BREAKDOWN-------------------------"<<endl<<endl;
cout<<"Power Reflected                  --> "<<std::real(Pref)<<"%"<<endl<<endl;
cout<<"Power Lost to Ground             --> "<<std::real(Pg)<<"%"<<endl<<endl;
cout<<"Power Lost in Series Resistance  --> "<<std::real(Ps)<<"%"<<endl<<endl;
cout<<"Power Received at the Load       --> "<<std::real(Pr)<<"%"<<endl<<endl;
```

```
                cout<<"                                 ----------"<<endl;
                cout<<"                         Total       "<<std::real(Pt)<<"%"<<endl<<endl;
                cout<<"Power Received at the Load w/o match --> "<<std::real(Pu)<<"%"<<endl<<endl;

                if(std::real(Pu)>=std::real(Pr)){
                        cout<<"Matching Network Degrades Performance -- DO NOT USE NETWORK"<<endl<<endl;
                }
                else{
                        cout<<"Matching Network Improves Performance"<<endl<<endl;
                }


}

void t_par(double x1,double x2,double x3){
                if(x1>=0){
                        cout<<"Enter the following parameters for the inductor X1"<<endl;
                        cout<<"R = ";
                        cin>>r11;
                        cout<<"L (nH) = ";
                        cin>>L11;
                        cout<<"Rs1 = ";
                        cin>>rs11;
                        cout<<"Cs1 (pF) = ";
                        cin>>cs11;
                        cout<<"Rs2 = ";
                        cin>>rs21;
                        cout<<"Cs2 (pF) = ";
                        cin>>cs21;
                }
                else{
                        cout<<"Enter the following parameters for the capacitor X1"<<endl;
                        cout<<"R = ";
                        cin>>r11;
                        cout<<"C (pF) = ";
                        cin>>C11;
                        cout<<"Rt = ";
                        cin>>rs11;
                        cout<<"Ct (pF) = ";
                        cin>>cs11;
                        cout<<"Rb = ";
                        cin>>rs21;
                        cout<<"Cb (pF) = ";
                        cin>>cs21;
                }
                if(x2<=0){
                        cout<<"Enter the following parameters for the capacitor X2"<<endl;
                        cout<<"R = ";
                        cin>>r22;
                        cout<<"C (pF) = ";
                        cin>>C22;
                        cout<<"Rt = ";
                        cin>>rs12;
                        cout<<"Ct (pF) = ";
                        cin>>cs12;
                }
                else{
                        cout<<"Enter the following parameters for the inductor X2"<<endl;
                        cout<<"R = ";
                        cin>>r22;
                        cout<<"L (nH) = ";
                        cin>>L22;
                        cout<<"Rs1 = ";
                        cin>>rs12;
                        cout<<"Cs1 (pF) = ";
                        cin>>cs12;
                }
                if(x3>=0){
                        cout<<"Enter the following parameters for the inductor X3"<<endl;
                        cout<<"R = ";
```

```
                        cin>>r33;
                        cout<<"L (nH) = ";
                        cin>>L33;
                        cout<<"Rs1 = ";
                        cin>>rs13;
                        cout<<"Cs1 (pF) = ";
                        cin>>cs13;
                        cout<<"Rs2 = ";
                        cin>>rs23;
                        cout<<"Cs2 (pF) = ";
                        cin>>cs23;
                }
                else{
                        cout<<"Enter the following parameters for the capacitor X3"<<endl;
                        cout<<"R = ";
                        cin>>r33;
                        cout<<"C (pF) = ";
                        cin>>C33;
                        cout<<"Rt = ";
                        cin>>rs13;
                        cout<<"Ct (pF) = ";
                        cin>>cs13;
                        cout<<"Rb = ";
                        cin>>rs23;
                        cout<<"Cb (pF) = ";
                        cin>>cs23;
                }

                z1=x1;
                z2=x2;
                z3=x3;

}

void p_par(double x1,double x2,double x3){
                if(x1>=0){
                        cout<<"Enter the following parameters for the inductor X1"<<endl;
                        cout<<"R = ";
                        cin>>r11;
                        cout<<"L (nH) = ";
                        cin>>L11;
                        cout<<"Rs1 = ";
                        cin>>rs11;
                        cout<<"Cs1 (pF) = ";
                        cin>>cs11;
                }
                else{
                        cout<<"Enter the following parameters for the capacitor X1"<<endl;
                        cout<<"R = ";
                        cin>>r11;
                        cout<<"C (pF) = ";
                        cin>>C11;
                        cout<<"Rt = ";
                        cin>>rs11;
                        cout<<"Ct (pF) = ";
                        cin>>cs11;
                }
                if(x2>=0){
                        cout<<"Enter the following parameters for the inductor X2"<<endl;
                        cout<<"R = ";
                        cin>>r22;
                        cout<<"L (nH) = ";
                        cin>>L22;
                        cout<<"Rs1 = ";
                        cin>>rs12;
                        cout<<"Cs1 (pF) = ";
                        cin>>cs12;
                        cout<<"Rs2 = ";
                        cin>>rs22;
                        cout<<"Cs2 (pF) = ";
```

```
                cin>>cs22;
        }
        else{
                cout<<"Enter the following parameters for the capacitor X2"<<endl;
                cout<<"R = ";
                cin>>r22;
                cout<<"C (pF) = ";
                cin>>C22;
                cout<<"Rt = ";
                cin>>rs12;
                cout<<"Ct (pF) = ";
                cin>>cs12;
                cout<<"Rb = ";
                cin>>rs22;
                cout<<"Cb (pF) = ";
                cin>>cs22;
        }
        if(x3>=0){
                cout<<"Enter the following parameters for the inductor X3"<<endl;
                cout<<"R = ";
                cin>>r33;
                cout<<"L (nH) = ";
                cin>>L33;
                cout<<"Rs1 = ";
                cin>>rs13;
                cout<<"Cs1 (pF) = ";
                cin>>cs13;
        }
        else{
                cout<<"Enter the following parameters for the capacitor X3"<<endl;
                cout<<"R = ";
                cin>>r33;
                cout<<"C (pF) = ";
                cin>>C33;
                cout<<"Rt = ";
                cin>>rs13;
                cout<<"Ct (pF) = ";
                cin>>cs13;
        }

        z1=x1;
        z2=x2;
        z3=x3;

}

void parasitic(){
        cout<<endl;
        cout<<"-------------------------PARASITIC EFFECTS-------------------------"<<endl<<endl;
        cout<<"The following calculations will examine the effects of the "<<endl;
        cout<<" substrate's parasitics.  For inductors, the parasitic elements "<<endl;
        cout<<" can be obtained using the output file from ASITIC.  For capacitors "<<endl;
        cout<<" to ground, the bottom plate is assume to be run to ground, which "<<endl;
        cout<<" minimizes the parasitics.  In this case, the parasitic capacitor "<<endl;
        cout<<" can be assumed in the low femto farads.  The series resistance of "<<endl;
        cout<<" capacitors can be assumed less than 0.5 ohms if the layout was "<<endl;
        cout<<" done using the method outlined in the accompanying paper.  For "<<endl;
        cout<<" floating capacitors, the bottom plate parasitics can be calculated "<<endl;
        cout<<" using the method outlined in the accompanying paper."<<endl<<endl;
        imp=1.0/(2.0*3.14159*f);
        cout<<endl<<"Which Network from above are you using? (A/B/C/D/E/F/G/H/I/J) ";
        cin>>ans;
        cout<<endl;

        if(ans=='A'||ans=='B'){
                if(ans=='A'){
                        if(rin<rout){
                                l_par_down(x1A,x2A);
                        }
                        else{
```

```
                                        l_par_up(x1A,x2A);
                                }

                        }
                        else{
                                if(rin<rout){
                                        l_par_down(x1B,x2B);
                                }
                                else{
                                        l_par_up(x1B,x2B);
                                }
                        }
}

else if(ans=='C'||ans=='D'||ans=='E'||ans=='F'){
                        if(ans=='C'){
                                t_par(x1C,x2C,x3C);
                        }
                        else if (ans=='D'){
                                t_par(x1D,x2D,x3D);
                        }
                        else if (ans=='E'){
                                t_par(x1E,x2E,x3E);
                        }
                        else{
                                t_par(x1F,x2F,x3F);
                        }

                        n=0.000000001;
                        p=0.000000000001;
                        L11=L11*n;
                        L22=L22*n;
                        L33=L33*n;
                        C11=C11*p;
                        C22=C22*p;
                        C33=C33*p;
                        cs11=cs11*p;
                        cs21=cs21*p;
                        cs12=cs12*p;
                        cs22=cs22*p;
                        cs13=cs13*p;
                        cs23=cs23*p;


                        //Calculate Mismatch for T-Network
                        std::complex<double> x3s2(rs23,-imp/cs23), x2s1(rs12,-imp/cs12), x1s1(rs11,-imp/cs11), nz2(r22,x2);
                        std::complex<double> zmid=1.0/((1.0/x3s2)+(1.0/x2s1)+(1.0/x1s1)+(1.0/nz2));
                        std::complex<double> zout(rout,xout), x1s2(rs21,-imp/cs21), nz1(r11,x1);
                        std::complex<double> zright=(zout*x1s2/(zout+x1s2))+nz1;
                        std::complex<double> nz3(r33,x3), x3s1(rs13,-imp/cs13);
                        std::complex<double> zeq1=zright*zmid/(zright+zmid)+nz3;
                        std::complex<double> ztotal=zeq1*x3s1/(zeq1+x3s1);

                        cout<<"The old load including the ideal PI - matching network was"<<endl<<endl;
                        cout<<"         Z = "<<std::real(pzt)<<" + j("<<std::imag(pzt)<<")"<<endl<<endl;

                        cout<<"The load including the parasitics of the PI - matching network is"<<endl<<endl;
                        cout<<"         Z = "<<std::real(ztotal)<<" + j("<<std::imag(ztotal)<<")"<<endl<<endl;

                        //Original input impedance
                        cout<<"The input impedance was given or calculated to be"<<endl<<endl;
                        cout<<"         Z = "<<rin<<" + j("<<-xin<<")"<<endl<<endl;

                        //Power Transferred
                        std::complex<double> zin(rin,-xin);
                        I=1.0/(2*rin);
                        max_power=I*I*rin/2;
                        std::complex<double> I=1.0/(zin+ztotal);
                        std::complex<double> power=std::abs(I)*std::abs(I)*std::real(ztotal)/2.0/max_power*100;
```

```
		//Calculate branch currents
		std::complex<double> Ig1=zeq1/(zeq1+x3s1)*I;
		std::complex<double> Ix3=x3s1/(zeq1+x3s1)*I;
		std::complex<double> Ix1=zmid/(zmid+zright)*Ix3;
		std::complex<double> Imid=Ix3-Ix1;
		std::complex<double> Ig2=nz2/(nz2+(1.0/((1.0/x3s2)+(1.0/x2s1)+(1.0/x1s1))))*Imid;
		std::complex<double> Ix2=Imid-Ig2;
		std::complex<double> Ig3=zout/(zout+x1s2)*Ix1;
		std::complex<double> Ir=x1s2/(zout+x1s2)*Ix1;

		//Power Calculations
		std::complex<double> Pr=std::abs(Ir)*std::abs(Ir)*std::real(zout)/2.0/max_power*100;
		std::complex<double> Pg1=std::abs(Ig1)*std::abs(Ig1)*std::real(x3s1)/2.0/max_power*100;
		std::complex<double>
Pg2=std::abs(Ig2)*std::abs(Ig2)*std::real(1.0/((1.0/(x3s2)+1.0/(x2s1)+1.0/(x1s1))))/2.0/max_power*100;
		std::complex<double> Pg3=std::abs(Ig3)*std::abs(Ig3)*std::real(x1s2)/2.0/max_power*100;
		std::complex<double> Pg=Pg1+Pg2+Pg3;
		std::complex<double> Ps1=std::abs(Ix1)*std::abs(Ix1)*std::real(nz1)/2.0/max_power*100;
		std::complex<double> Ps2=std::abs(Ix2)*std::abs(Ix2)*std::real(nz2)/2.0/max_power*100;
		std::complex<double> Ps3=std::abs(Ix3)*std::abs(Ix3)*std::real(nz3)/2.0/max_power*100;
		std::complex<double> Ps=Ps1+Ps2+Ps3;
		std::complex<double> Max(100,0);
		std::complex<double> Pref=Max-power;
		std::complex<double> Pt=Pref+Pg+Ps+Pr;

		//Unmatched Power Calculations
		std::complex<double> Iu=1.0/(zin+zout);
		std::complex<double> Pu=std::abs(Iu)*std::abs(Iu)*std::real(zout)/2.0/max_power*100;

		//Display Results
		cout<<"Amount of Maximum Power Transferred  --> "<<std::real(power)<<"%"<<endl<<endl<<endl;
		cout<<"-----------------------POWER BREAKDOWN-------------------------"<<endl<<endl;
		cout<<"Power Reflected                  --> "<<std::real(Pref)<<"%"<<endl<<endl;
		cout<<"Power Lost to Ground             --> "<<std::real(Pg)<<"%"<<endl<<endl;
		cout<<"Power Lost in Series Resistance  --> "<<std::real(Ps)<<"%"<<endl<<endl;
		cout<<"Power Received at the Load       --> "<<std::real(Pr)<<"%"<<endl<<endl;
		cout<<"                                 ---------------"<<endl;
		cout<<"                        Total      "<<std::real(Pt)<<"%"<<endl<<endl;
		cout<<"Power Received at the Load w/o match --> "<<std::real(Pu)<<"%"<<endl<<endl;

		if(std::real(Pu)>=std::real(Pr)){
			cout<<"Matching Network Degrades Performance -- DO NOT USE NETWORK"<<endl<<endl;
		}
		else{
			cout<<"Matching Network Improves Performance"<<endl<<endl;
		}

	}
	else if(ans=='G'||ans=='H'||ans=='I'||ans=='J'){

		if(ans=='G'){
			p_par(x1G,x2G,x3G);
		}
		else if (ans=='H'){
			p_par(x1H,x2H,x3H);
		}
		else if (ans=='I'){
			p_par(x1I,x2I,x3I);
		}
		else{
			p_par(x1J,x2J,x3J);
		}

		n=0.000000001;
		p=0.000000000001;
		L11=L11*n;
		L22=L22*n;
		L33=L33*n;
		C11=C11*p;
		C22=C22*p;
```

```
C33=C33*p;
cs11=cs11*p;
cs21=cs21*p;
cs12=cs12*p;
cs22=cs22*p;
cs13=cs13*p;
cs23=cs23*p;


//Calculate Mismatch for PI-Network

std::complex<double> x3s2(rs23,-imp/cs23), x2s1(rs12,-imp/cs12), x1s1(rs11,-imp/cs11), nz2(r22,z2);
std::complex<double> zout(rout,xout), x2s2(rs22,-imp/cs22), x1s2(rs21,-imp/cs21), nz1(r11,z1);
std::complex<double> nz3(r33,z3), x3s1(rs13,-imp/cs13);
std::complex<double> zright=1.0/((1.0/x2s2)+(1.0/x1s1)+(1.0/nz1)+(1.0/zout));
std::complex<double> zleft=1.0/((1.0/x3s1)+(1.0/x2s1)+(1.0/nz3));
std::complex<double> zeq1=zright+nz2;
std::complex<double> ztotal=zeq1*zleft/(zeq1+zleft);

std::complex<double> pzout(rout,xout), pz1(0,z1), pz2(0,z2), pz3(0,z3);
std::complex<double> pzt=(((pzout*pz1)/(pzout+pz1)+pz2)*pz3)/((pzout*pz1)/(pzout+pz1)+pz2+pz3);

cout<<"The old load including the ideal PI - matching network was"<<endl<<endl;
cout<<"        Z = "<<std::real(pzt)<<" + j("<<std::imag(pzt)<<")"<<endl<<endl;

cout<<"The load including the parasitics of the PI - matching network is"<<endl<<endl;
cout<<"        Z = "<<std::real(ztotal)<<" + j("<<std::imag(ztotal)<<")"<<endl<<endl;

//Original input impedance
cout<<"The input impedance was given or calculated to be"<<endl<<endl;
cout<<"        Z = "<<rin<<" + j("<<-xin<<")"<<endl<<endl;

//Power Transferred
std::complex<double> zin(rin,-xin);
I=1.0/(2*rin);
max_power=I*I*rin/2;
std::complex<double> I=1.0/(zin+ztotal);
std::complex<double> power=std::abs(I)*std::abs(I)*std::real(ztotal)/2.0/max_power*100;

//Calculate branch currents
std::complex<double> Ileft=zeq1/(zeq1+zleft)*I;
std::complex<double> Ix2=zleft/(zeq1+zleft)*I;
std::complex<double> Ig1=nz3/(nz3+(1.0/(1.0/x3s1+1.0/x2s1)))*Ileft;
std::complex<double> Ix3=(1.0/(1.0/x3s1+1.0/x2s1))/(nz3+(1.0/(1.0/x3s1+1.0/x2s1)))*Ileft;
std::complex<double> Ig2=(1.0/(1.0/nz1+1.0/zout))/(1.0/(1.0/nz1+1.0/zout)+1.0/(1.0/x2s2+1.0/x1s1))*Ix2;
std::complex<double> Ix1=(1.0/(1.0/x2s2+1.0/x1s1+1.0/zout))/(nz1+1.0/(1.0/x2s2+1.0/x1s1+1.0/zout))*Ix2;
std::complex<double> Ir=(1.0/(1.0/x2s2+1.0/x1s1+1.0/nz1))/(zout+1.0/(1.0/x2s2+1.0/x1s1+1.0/nz1))*Ix2;

//Power Calculations
std::complex<double> Pr=std::abs(Ir)*std::abs(Ir)*std::real(zout)/2.0/max_power*100;
std::complex<double> Pg1=std::abs(Ig1)*std::abs(Ig1)*std::real(1.0/((1.0/(x3s1)+1.0/(x2s1))))/2.0/max_power*100;
std::complex<double> Pg2=std::abs(Ig2)*std::abs(Ig2)*std::real(1.0/((1.0/(x2s2)+1.0/(x1s1))))/2.0/max_power*100;
std::complex<double> Pg=Pg1+Pg2;
std::complex<double> Ps1=std::abs(Ix1)*std::abs(Ix1)*std::real(nz1)/2.0/max_power*100;
std::complex<double> Ps2=std::abs(Ix2)*std::abs(Ix2)*std::real(nz2)/2.0/max_power*100;
std::complex<double> Ps3=std::abs(Ix3)*std::abs(Ix3)*std::real(nz3)/2.0/max_power*100;
std::complex<double> Ps=Ps1+Ps2+Ps3;
std::complex<double> Max(100,0);
std::complex<double> Pref=Max-power;
std::complex<double> Pt=Pref+Pg+Ps+Pr;

//Unmatched Power Calculations
std::complex<double> Iu=1.0/(zin+zout);
std::complex<double> Pu=std::abs(Iu)*std::abs(Iu)*std::real(zout)/2.0/max_power*100;

//Display Results
cout<<"Amount of Maximum Power Transferred  --> "<<std::real(power)<<"%"<<endl<<endl<<endl;
cout<<"-----------------------POWER BREAKDOWN-------------------------"<<endl<<endl;
cout<<"Power Reflected                  --> "<<std::real(Pref)<<"%"<<endl<<endl;
cout<<"Power Lost to Ground             --> "<<std::real(Pg)<<"%"<<endl<<endl;
```

```
                cout<<"Power Lost in Series Resistance      --> "<<std::real(Ps)<<"%"<<endl<<endl;
                cout<<"Power Received at the Load           --> "<<std::real(Pr)<<"%"<<endl<<endl;
                cout<<"                                    ----------"<<endl;
                cout<<"                          Total      "<<std::real(Pt)<<"%"<<endl<<endl;
                cout<<"Power Received at the Load w/o match --> "<<std::real(Pu)<<"%"<<endl<<endl;

                if(std::real(Pu)>=std::real(Pr)){
                        cout<<"Matching Network Degrades Performance -- DO NOT USE NETWORK"<<endl<<endl;
                }
                else{
                        cout<<"Matching Network Improves Performance"<<endl<<endl;
                }
        }
}

void main(){

cout<<endl;
cout<<"This software allows easy calculation of impedance matching"<<endl;
cout<<" networks given the input and output impedances (in ohms)."<<endl;
cout<<endl;
cout<<"GENERAL FORM"<<endl;
cout<<endl;
cout<<"    ----\\\\\\\\\\-----((((((--------O--------\\\\\\\\\\-----((((((-------"<<endl;
cout<<"   |      Rin         Xin                        Rout        Xout      |  "<<endl;
cout<<"   |                                                                   |  "<<endl;
cout<<"   |                                                                   |  "<<endl;
cout<<"  ---                                                                 --- "<<endl;
cout<<"  Vin                                                                Gnd "<<endl;
cout<<endl;

cout<<"Is your circuit in the series form as above? (y/n) ";
cin>>ans;
cout<<endl;
cout<<"If Xin or Xout is absent, enter a zero for its value.  The zero is"<<endl;
cout<<" a placeholder and does not necessarily mean zero reactance."<<endl<<endl;

        if(ans=='n'){
                cout<<"What impedances are in parallel?"<<endl;
                cout<<"Input only   - type 'i'"<<endl;
                cout<<"Output only  - type 'o'"<<endl;
                cout<<"Both         - type 'b'"<<endl;
                cout<<"Answer ";
                cin>>ans;
                cout<<endl;
                if(ans=='i'){
                        cout<<"Enter input parallel resistance ";
                        cin>>rinp;
                        cout<<"Enter input parallel reactance ";
                        cin>>xinp;
                        //Calculate rin and xin
                        rin=rinp*xinp*xinp/(rinp*rinp + xinp*xinp);
                        xin=rinp*rinp*xinp/(rinp*rinp + xinp*xinp);
                        //Make complex conjugate
                        xin=-xin;
                        cout<<"Rout = ";
                        cin>>rout;
                        cout<<"Xout = ";
                        cin>>xout;
                        cout<<"Frequency (MHz) = ";
                        cin>>f;
                }
                else if(ans=='o'){
                        cout<<"Rin = ";
                        cin>>rin;
                        cout<<"Xin = ";
                        cin>>xin;
                        //Make complex conjugate
                        xin=-xin;
                        cout<<"Enter output parallel resistance ";
```

```
                cin>>routp;
                cout<<"Enter output parallel reactance ";
                cin>>xoutp;
                //Calculate rout and xout
                rout=routp*xoutp*xoutp/(routp*routp + xoutp*xoutp);
                xout=routp*routp*xoutp/(routp*routp + xoutp*xoutp);
                cout<<"Frequency (MHz) = ";
                cin>>f;
        }
        else{
                cout<<"Enter input parallel resistance ";
                cin>>rinp;
                cout<<"Enter input parallel reactance ";
                cin>>xinp;
                //Calculate rin and xin
                rin=rinp*xinp*xinp/(rinp*rinp + xinp*xinp);
                xin=rinp*rinp*xinp/(rinp*rinp + xinp*xinp);
                //Make complex conjugate
                xin=-xin;
                cout<<"Enter output parallel resistance ";
                cin>>routp;
                cout<<"Enter output parallel reactance ";
                cin>>xoutp;
                //Calculate rout and xout
                rout=routp*xoutp*xoutp/(routp*routp + xoutp*xoutp);
                xout=routp*routp*xoutp/(routp*routp + xoutp*xoutp);
                cout<<"Frequency (MHz) = ";
                cin>>f;
        }

}
else{
        cout<<"Enter the following values"<<endl;
        cout<<"Rin = ";
        cin>>rin;
        cout<<"Xin = ";
        cin>>xin;
        xin=-xin;                                       //Make complex conjugate
        cout<<"Rout = ";
        cin>>rout;
        cout<<"Xout = ";
        cin>>xout;
        cout<<"Frequency (MHz) = ";
        cin>>f;
}

f=f*1000000;

//Checks
        while(i==0){

                if(rin<rout){
                        ldown();
                        tdown();
                        pidown();
                }
                else{
                        lup();
                        tup();
                        piup();
                }

                cout<<"Would you like to rerun with different Q-values (y/n)";
                cin>>ans;
                cout<<endl;
                if(ans=='n'){
                        i=1;
                }
        }
```

```
        parasitic();

        cout<<endl<<endl;
        cout<<"Press any key to exit";
        cin>>ans;
}
```

Appendix D – Test Setups

D.1 Agilent 8712ET RF Network Analyzer Setup for Discrete Network

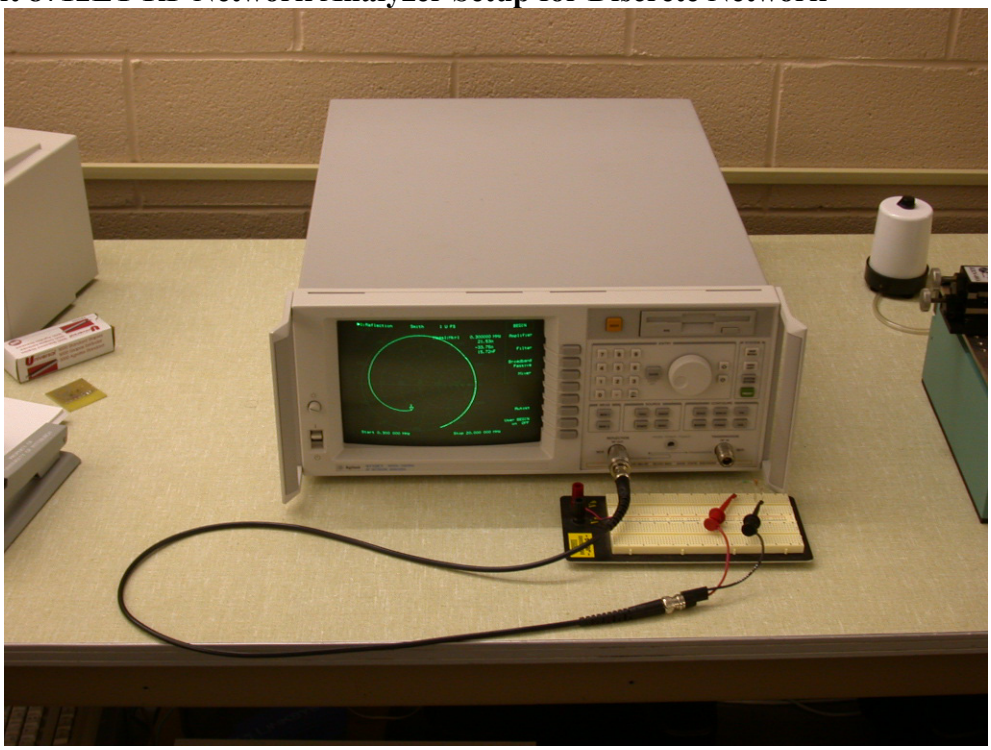


Figure D.1 – Agilent 8712ET RF Network Analyzer Connected to the Discrete Network using Fifty-ohm Coaxial Cable

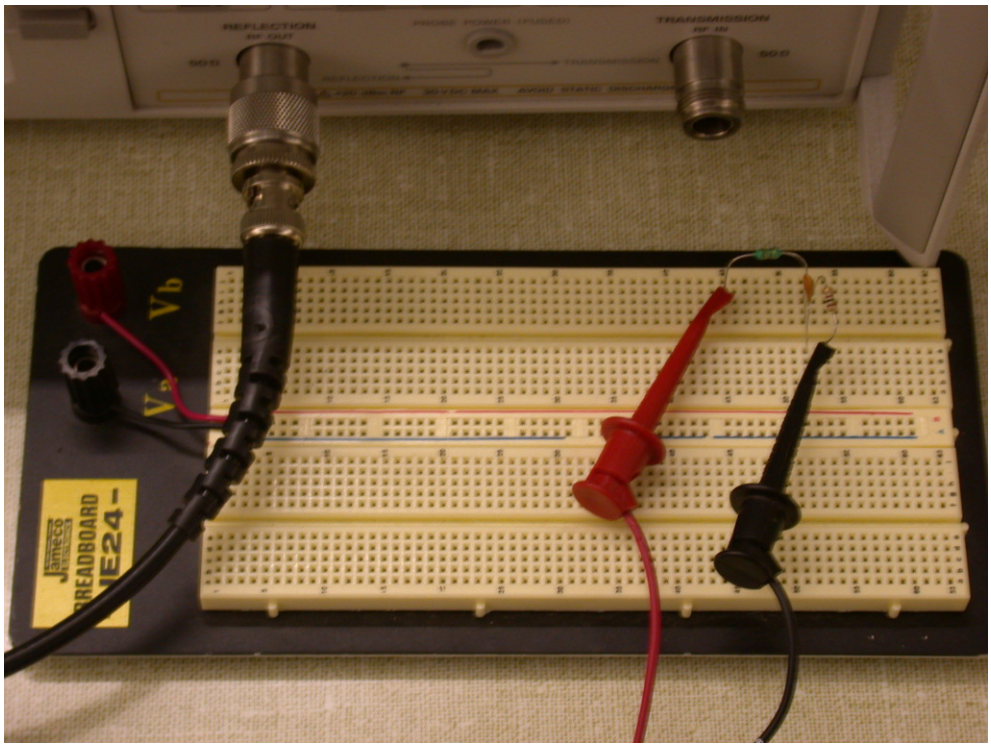


Figure D.2 – Close-up of Discrete Network Connection

D.2 Agilent 4284A Precision LCR Meter Setup with Four-Lead Probe

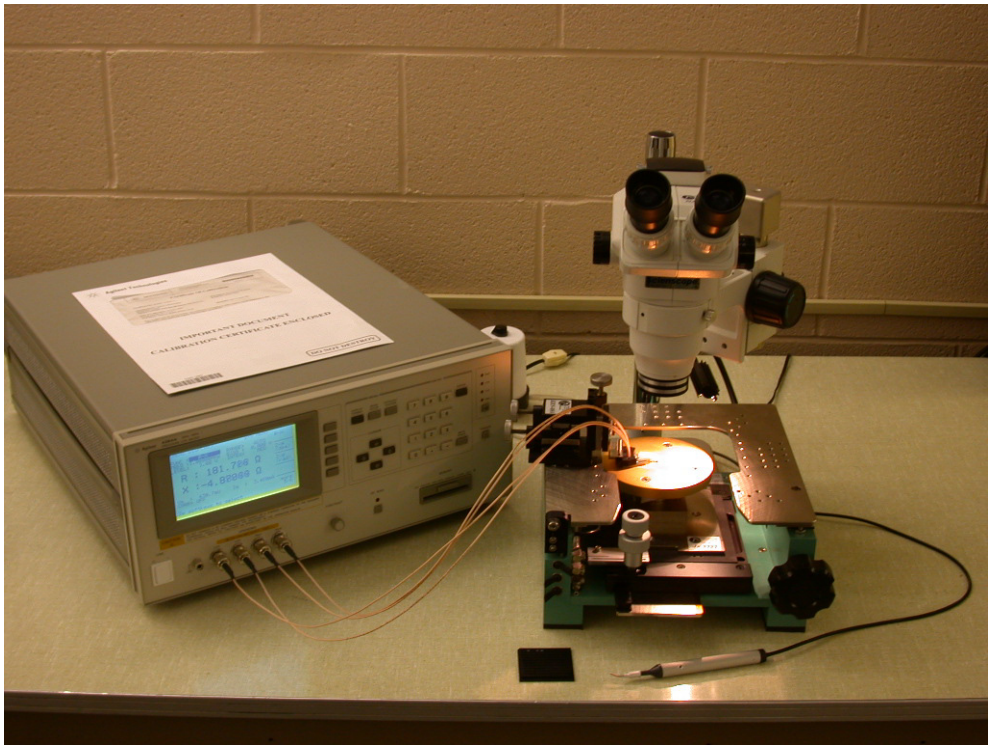


Figure D.3 – Agilent 4284A Connected to the J micro Technology JR-2727 Probe Station through Four Fifty-ohm Coaxial Cables

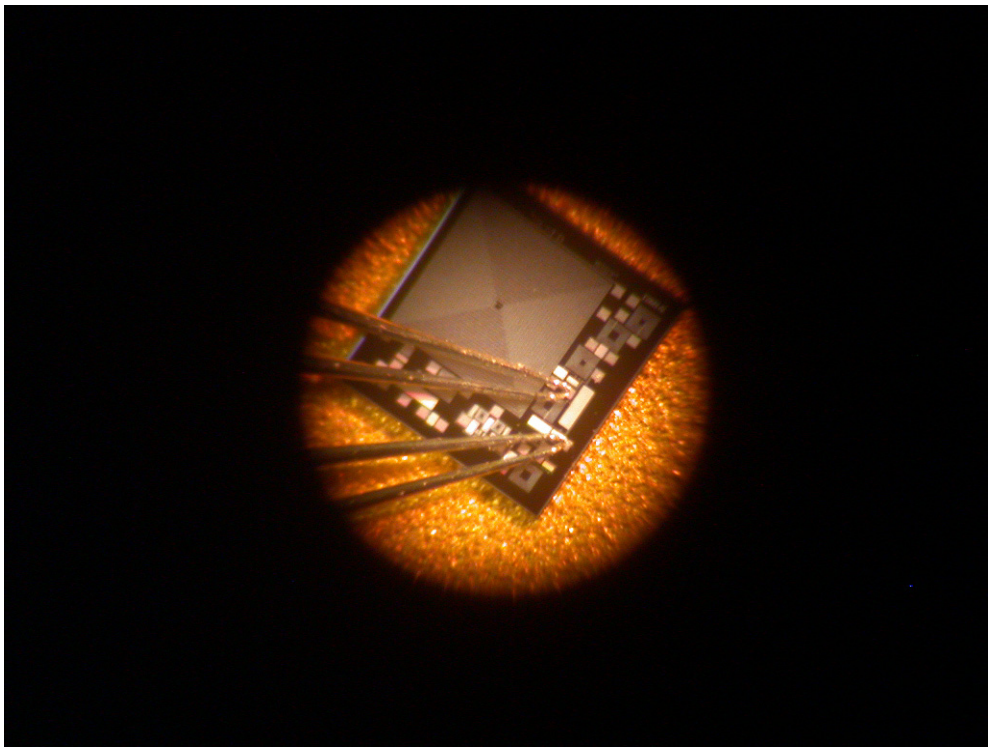


Figure D.4 – Close-up of Four-lead Probing

D.3 Agilent 8712ET RF Network Analyzer Setup with Coaxial Probe

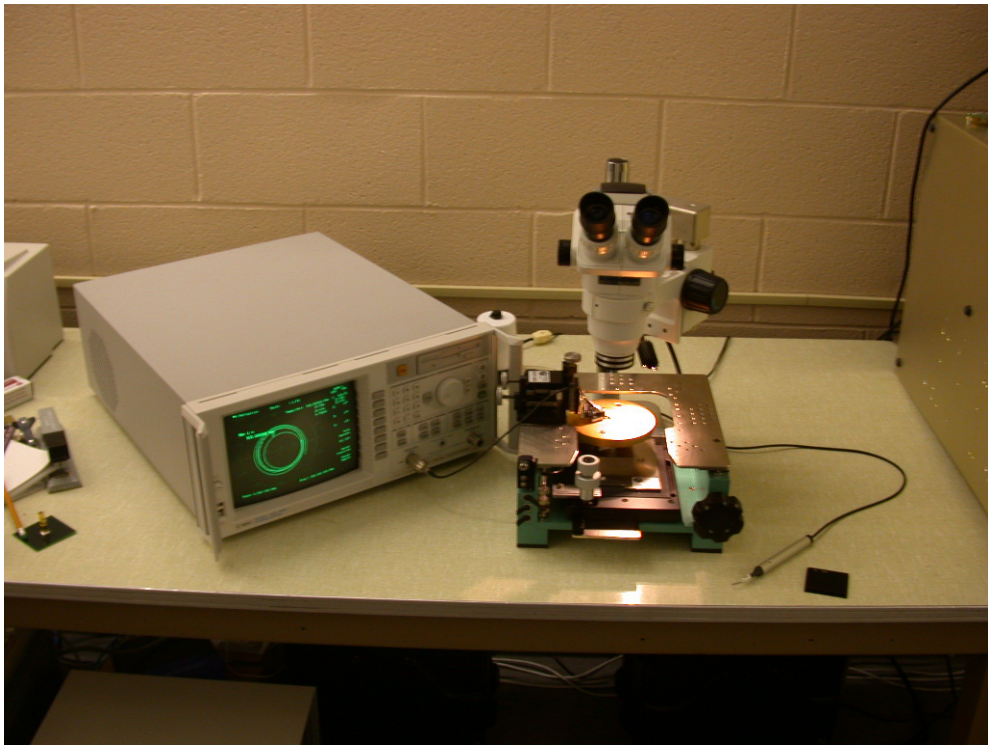


Figure D.5 – Agilent 8712ET Connected to the J micro Technology JR-2727 Probe Station through Fifty-ohm Coaxial Cable

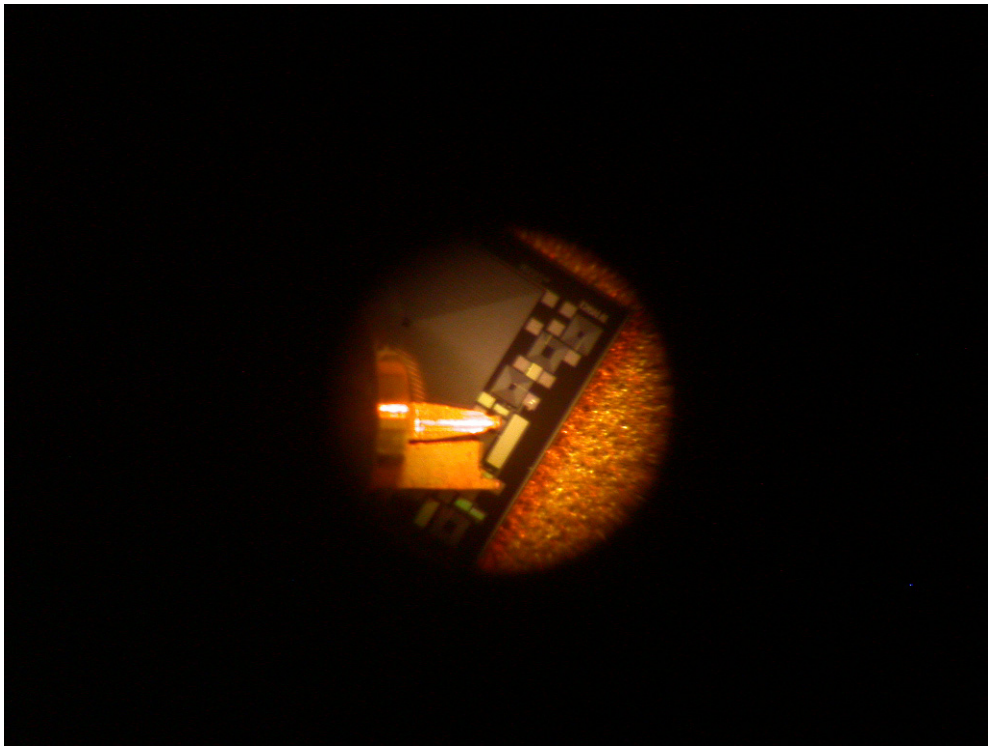


Figure D.6 – Close-up of Coaxial Probing

D.4 Substrate Bonded to Ground Plane

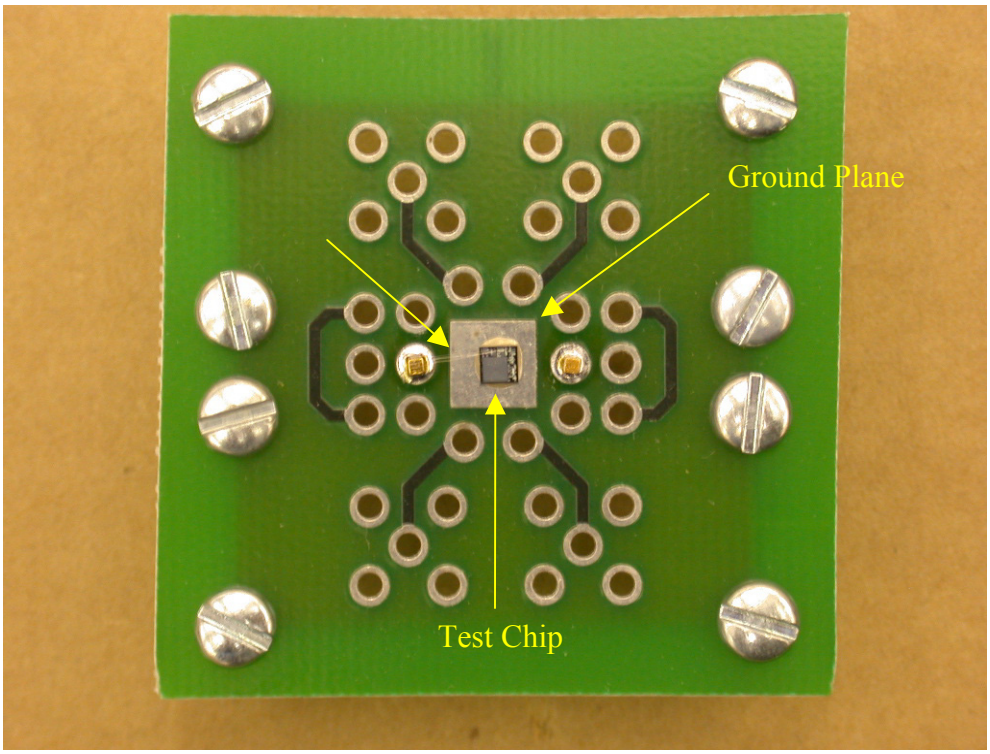


Figure D.7 – Test Board for Grounded Substrate

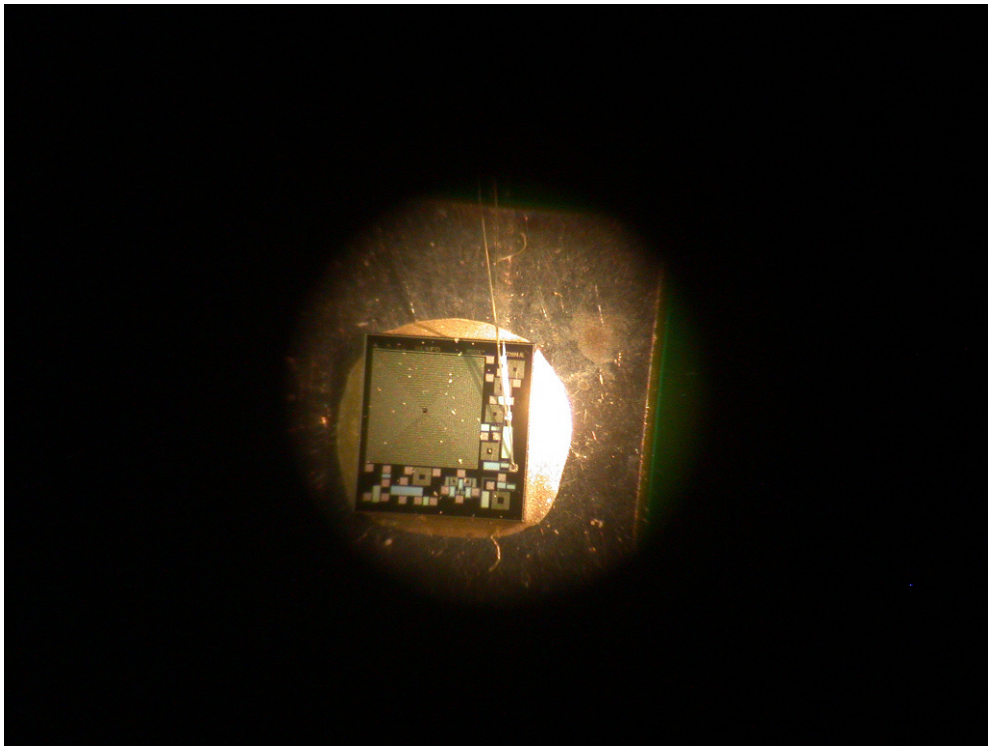


Figure D.8 – Close-up of Bonded Chip

Appendix E – Mathcad Calculations

Network Equivalent Impedances

Calculations for Floating Substrate

Target Values

$C_{p3} := 10.44p$ $R_{load} := 200$ $C_{L2} := 2.01p$ $L_{L1} := 12.1n$

$C_{p1} := 4.43p$ $C_{load} := 1p$ $L_{p2} := 8.04n$

L-Network

$f := 915M$ $\omega := 2\cdot\pi\cdot f$

$$Z_{IN} := \left[\left(\frac{1}{R_{load}}\right) + j\cdot\omega\cdot C_{load} + \frac{1}{j\cdot\omega\cdot L_{L1}}\right]^{-1} + \frac{1}{j\cdot\omega\cdot C_{L2}}$$

$Z_{IN} = 50.297 + 0.236i$

$f := 1M$ $\omega := 2\cdot\pi\cdot f$

$$Z_{IN} := \left[\left(\frac{1}{R_{load}}\right) + j\cdot\omega\cdot C_{load} + \frac{1}{j\cdot\omega\cdot L_{L1}}\right]^{-1} + \frac{1}{j\cdot\omega\cdot C_{L2}}$$

$Z_{IN} = 2.89\times 10^{-5} - 7.918i\times 10^{4}$

$f := 10M$ $\omega := 2\cdot\pi\cdot f$

$$Z_{IN} := \left[\left(\frac{1}{R_{load}}\right) + j\cdot\omega\cdot C_{load} + \frac{1}{j\cdot\omega\cdot L_{L1}}\right]^{-1} + \frac{1}{j\cdot\omega\cdot C_{L2}}$$

$Z_{IN} = 2.89\times 10^{-3} - 7.917i\times 10^{3}$

PI-Network

$f := 915M \qquad \omega := 2\cdot\pi\cdot f$

$$Z_{IN} := \frac{\left[\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + j\cdot\omega\cdot L_{p2}\right]\cdot\frac{1}{j\cdot\omega\cdot C_{p3}}}{\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + \left(j\cdot\omega\cdot L_{p2}\right) + \frac{1}{j\cdot\omega\cdot C_{p3}}}$$

$$Z_{IN} = 49.928 + 6.767i\times 10^{-3}$$

$f := 1M \qquad \omega := 2\cdot\pi\cdot f$

$$Z_{IN} := \frac{\left[\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + j\cdot\omega\cdot L_{p2}\right]\cdot\frac{1}{j\cdot\omega\cdot C_{p3}}}{\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + \left(j\cdot\omega\cdot L_{p2}\right) + \frac{1}{j\cdot\omega\cdot C_{p3}}}$$

$$Z_{IN} = 199.922 - 3.936i$$

$f := 10M \qquad \omega := 2\cdot\pi\cdot f$

$$Z_{IN} := \frac{\left[\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + j\cdot\omega\cdot L_{p2}\right]\cdot\frac{1}{j\cdot\omega\cdot C_{p3}}}{\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + \left(j\cdot\omega\cdot L_{p2}\right) + \frac{1}{j\cdot\omega\cdot C_{p3}}}$$

$$Z_{IN} = 192.474 - 37.888i$$

Including the Inductors' Resistances

Target Parasitics

$R_{LL1} := 20.7 \qquad R_{Lp2} := 13.1$

L-Network

$f := 1M \qquad \omega := 2\cdot\pi\cdot f$

$$Z_{IN} := \left[\left(\frac{1}{R_{load}}\right) + j\cdot\omega\cdot C_{load} + \frac{1}{R_{LL1} + j\cdot\omega\cdot L_{L1}}\right]^{-1} + \frac{1}{j\cdot\omega\cdot C_{L2}}$$

$$Z_{IN} = 18.759 - 7.918i \times 10^{4}$$

$f := 10M \qquad \omega := 2\cdot\pi\cdot f$

$$Z_{IN} := \left[\left(\frac{1}{R_{load}}\right) + j\cdot\omega\cdot C_{load} + \frac{1}{R_{LL1} + j\cdot\omega\cdot L_{L1}}\right]^{-1} + \frac{1}{j\cdot\omega\cdot C_{L2}}$$

$$Z_{IN} = 18.762 - 7.918i \times 10^{3}$$

PI-Network

$f := 1M \qquad \omega := 2\cdot\pi\cdot f$

$$Z_{IN} := \frac{\left[\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + R_{Lp2} + j\cdot\omega\cdot L_{p2}\right]\cdot\frac{1}{j\cdot\omega\cdot C_{p3}}}{\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + \left(R_{Lp2} + j\cdot\omega\cdot L_{p2}\right) + \frac{1}{j\cdot\omega\cdot C_{p3}}}$$

$$Z_{IN} = 213.012 - 4.291i$$

$f := 10M \qquad \omega := 2\cdot\pi\cdot f$

$$Z_{IN} := \frac{\left[\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + R_{Lp2} + j\cdot\omega\cdot L_{p2}\right]\cdot\frac{1}{j\cdot\omega\cdot C_{p3}}}{\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + \left(R_{Lp2} + j\cdot\omega\cdot L_{p2}\right) + \frac{1}{j\cdot\omega\cdot C_{p3}}}$$

$$Z_{IN} = 204.681 - 41.212i$$

Measured Values @ 1MHz

$C_{p3} := 11.319p$ $R_{load} := 159.75$ $C_{L2} := 2.43p$ $L_{L1} := 11.84n$

$C_{p1} := 4.833p$ $C_{load} := 1.354p$ $L_{p2} := 12.5n$

Measured Parasitics

$R_{LL1} := 20.55$ $R_{Lp2} := 13.09$

L-Network

$f := 915M$ $\omega := 2 \cdot \pi \cdot f$

$$Z_{IN} := \left[\left(\frac{1}{R_{load}}\right) + j \cdot \omega \cdot C_{load} + \frac{1}{j \cdot \omega \cdot L_{L1}}\right]^{-1} + \frac{1}{j \cdot \omega \cdot C_{L2}}$$

$$Z_{IN} = 72.046 + 7.91i$$

$f := 1M$ $\omega := 2 \cdot \pi \cdot f$

$$Z_{IN} := \left[\left(\frac{1}{R_{load}}\right) + j \cdot \omega \cdot C_{load} + \frac{1}{j \cdot \omega \cdot L_{L1}}\right]^{-1} + \frac{1}{j \cdot \omega \cdot C_{L2}}$$

$$Z_{IN} = 3.464 \times 10^{-5} - 6.55i \times 10^{4}$$

$f := 10M$ $\omega := 2 \cdot \pi \cdot f$

$$Z_{IN} := \left[\left(\frac{1}{R_{load}}\right) + j \cdot \omega \cdot C_{load} + \frac{1}{j \cdot \omega \cdot L_{L1}}\right]^{-1} + \frac{1}{j \cdot \omega \cdot C_{L2}}$$

$$Z_{IN} = 3.465 \times 10^{-3} - 6.549i \times 10^{3}$$

PI-Network

$f := 915M \qquad \omega := 2\cdot\pi\cdot f$

$$Z_{IN} := \frac{\left[\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + j\cdot\omega\cdot L_{p2}\right]\cdot\frac{1}{j\cdot\omega\cdot C_{p3}}}{\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + \left(j\cdot\omega\cdot L_{p2}\right) + \frac{1}{j\cdot\omega\cdot C_{p3}}}$$

$Z_{IN} = 1.292 - 23.235i$

$f := 1M \qquad \omega := 2\cdot\pi\cdot f$

$$Z_{IN} := \frac{\left[\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + j\cdot\omega\cdot L_{p2}\right]\cdot\frac{1}{j\cdot\omega\cdot C_{p3}}}{\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + \left(j\cdot\omega\cdot L_{p2}\right) + \frac{1}{j\cdot\omega\cdot C_{p3}}}$$

$Z_{IN} = 159.702 - 2.728i$

$f := 10M \qquad \omega := 2\cdot\pi\cdot f$

$$Z_{IN} := \frac{\left[\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + j\cdot\omega\cdot L_{p2}\right]\cdot\frac{1}{j\cdot\omega\cdot C_{p3}}}{\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + \left(j\cdot\omega\cdot L_{p2}\right) + \frac{1}{j\cdot\omega\cdot C_{p3}}}$$

$Z_{IN} = 155.135 - 26.484i$

Including the Inductors' Resistances

L-Network

$f := 1M \qquad \omega := 2\cdot\pi\cdot f$

$$Z_{IN} := \left[\left(\frac{1}{R_{load}}\right) + j\cdot\omega\cdot C_{load} + \frac{1}{R_{LL1} + j\cdot\omega\cdot L_{L1}}\right]^{-1} + \frac{1}{j\cdot\omega\cdot C_{L2}}$$

$$Z_{IN} = 18.208 - 6.55i\times 10^{4}$$

$f := 10M \qquad \omega := 2\cdot\pi\cdot f$

$$Z_{IN} := \left[\left(\frac{1}{R_{load}}\right) + j\cdot\omega\cdot C_{load} + \frac{1}{R_{LL1} + j\cdot\omega\cdot L_{L1}}\right]^{-1} + \frac{1}{j\cdot\omega\cdot C_{L2}}$$

$$Z_{IN} = 18.212 - 6.549i\times 10^{3}$$

PI-Network

$f := 1M \qquad \omega := 2\cdot\pi\cdot f$

$$Z_{IN} := \frac{\left[\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + R_{Lp2} + j\cdot\omega\cdot L_{p2}\right]\cdot\frac{1}{j\cdot\omega\cdot C_{p3}}}{\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + \left(R_{Lp2} + j\cdot\omega\cdot L_{p2}\right) + \frac{1}{j\cdot\omega\cdot C_{p3}}}$$

$$Z_{IN} = 172.785 - 3.037i$$

$f := 10M \qquad \omega := 2\cdot\pi\cdot f$

$$Z_{IN} := \frac{\left[\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + R_{Lp2} + j\cdot\omega\cdot L_{p2}\right]\cdot\frac{1}{j\cdot\omega\cdot C_{p3}}}{\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + \left(R_{Lp2} + j\cdot\omega\cdot L_{p2}\right) + \frac{1}{j\cdot\omega\cdot C_{p3}}}$$

$$Z_{IN} = 167.538 - 29.428i$$

Measured Values @ 10MHz

$C_{p3} := 9.314p$ $R_{load} := 192.074$ $C_{L2} := 2.05p$ $L_{L1} := 2.944n$

$C_{p1} := 4.011p$ $C_{load} := 1.237p$ $L_{p2} := 8.975n$

Measured Parasitics

$R_{LL1} := 25.455$ $R_{Lp2} := 16.197$

L-Network

$f := 10M$ $\omega := 2\cdot\pi\cdot f$

$$Z_{IN} := \left[\left(\frac{1}{R_{load}}\right) + j\cdot\omega\cdot C_{load} + \frac{1}{j\cdot\omega\cdot L_{L1}}\right]^{-1} + \frac{1}{j\cdot\omega\cdot C_{L2}}$$

$$Z_{IN} = 1.781\times 10^{-4} - 7.763i\times 10^{3}$$

PI-Network

$f := 10M$ $\omega := 2\cdot\pi\cdot f$

$$Z_{IN} := \frac{\left[\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + j\cdot\omega\cdot L_{p2}\right]\cdot\frac{1}{j\cdot\omega\cdot C_{p3}}}{\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1}\right)^{-1} + \left(j\cdot\omega\cdot L_{p2}\right) + \frac{1}{j\cdot\omega\cdot C_{p3}}}$$

$$Z_{IN} = 186.44 - 32.208i$$

Including the Inductor Resistances

L-Network

$f := 10M \qquad \omega := 2 \cdot \pi \cdot f$

$$Z_{IN} := \left[\left(\frac{1}{R_{load}} \right) + j \cdot \omega \cdot C_{load} + \frac{1}{R_{LL1} + j \cdot \omega \cdot L_{L1}} \right]^{-1} + \frac{1}{j \cdot \omega \cdot C_{L2}}$$

$$Z_{IN} = 22.477 - 7.764i \times 10^{3}$$

PI-Network

$f := 10M \qquad \omega := 2 \cdot \pi \cdot f$

$$Z_{IN} := \frac{\left[\left(\frac{1}{R_{load}} + j \cdot \omega \cdot C_{load} + j \cdot \omega \cdot C_{p1} \right)^{-1} + R_{Lp2} + j \cdot \omega \cdot L_{p2} \right] \cdot \frac{1}{j \cdot \omega \cdot C_{p3}}}{\left(\frac{1}{R_{load}} + j \cdot \omega \cdot C_{load} + j \cdot \omega \cdot C_{p1} \right)^{-1} + \left(R_{Lp2} + j \cdot \omega \cdot L_{p2} \right) + \frac{1}{j \cdot \omega \cdot C_{p3}}}$$

$$Z_{IN} = 201.79 - 35.815i$$

Calculations for Grounded Substrate

Measured Values @ 1MHz

$C_{p3} := 11.319p$ $R_{load} := 159.75$ $C_{L2} := 2.43p$ $L_{L1} := 11.84n$ $C_{par\_12.1nH} := 0.305p$

$C_{p1} := 4.833p$ $C_{load} := 1.354p$ $L_{p2} := 12.5n$ $C_{par\_2.01pf} := 0.43p$ $R_{par\_12.1nH} := 151$

$C_{par2\_8.04nH} := 0.294p$ $R_{par2\_8.04nH} := 17.7$ $C_{par1\_8.04nH} := 0.298p$ $R_{par1\_8.04nH} := 151$

Measured Parasitics

$R_{LL1} := 20.55$ $R_{Lp2} := 13.09$

L-Network

$f := 1M$ $\omega := 2\cdot\pi\cdot f$

$$Z_{IN} := \left[\left[\left(\frac{1}{R_{load}}\right) + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{par\_2.01pf} + \frac{1}{\frac{1}{j\cdot\omega\cdot C_{par\_12.1nH}} + R_{par\_12.1nH}} + \frac{1}{R_{LL1} + j\cdot\omega\cdot L_{L1}}\right]\right]^{-1} + \frac{1}{j\cdot\omega\cdot C_{L2}}$$

$Z_{IN} = 18.208 - 6.55i \times 10^{4}$

PI-Network

$f := 1M$ $\omega := 2\cdot\pi\cdot f$

$$Z_{IN} := \frac{\left[\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1} + \frac{1}{\frac{1}{j\cdot\omega\cdot C_{par2\_8.04nH}} + R_{par2\_8.04nH}}\right)^{-1} + R_{Lp2} + j\cdot\omega\cdot L_{p2}\right]\cdot\frac{1}{j\cdot\omega\cdot C_{p3}}}{\left(\frac{1}{R_{load}} + j\cdot\omega\cdot C_{load} + j\cdot\omega\cdot C_{p1} + \frac{1}{\frac{1}{j\cdot\omega\cdot C_{par2\_8.04nH}} + R_{par2\_8.04nH}}\right)^{-1} + \left(R_{Lp2} + j\cdot\omega\cdot L_{p2}\right) + \frac{1}{j\cdot\omega\cdot C_{p3}}}$$

$Z_{IN} = 172.784 - 3.084i$

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Dorf, Richard C. and James A. Svoboda. Introduction to Electronic Circuits. 5th ed. John Wiley & Sons, Inc. New York: New York, 2001. 179-184.

[2] Robbins, Allan H. and Wilhelm C. Miller. Circuit Analysis: Theory and Practice. 2nd ed. Delmar. New York: Albany, 2000. 353, 758-762, 839, 851-855, 860.

[3] Abrie, Pieter L.D. The Design of Impedance-Matching Network For Radio-Frequency and Microwave Amplifiers. Artech House, Inc. 1985. 69-92,

[4] Bowick, Chris. RF Circuit Design. Newnes. Maryland: Newton, 1982. 10-17, 66-97.

[5] Niknejad, Ali M. "ASITIC documentation," 572 Cory Hall #1770, University of California at Berkeley, Berkeley, CA 94720-1770 USA. http://formosa.eecs.berkeley.edu/~niknejad/doc-05-28-01/create.html#CREATE_SQUARE

[6] Mohan, Sunderarajan S. "Design, Modeling and Optimization of On-chip Inductor and Transformer Circuits," Diss. Stanford University, 1999.

[7] MOSIS, "The MOSIS Service," 4676 Admiralty Way, Marina del Rey, California 90292-6695 USA. http://www.mosis.org/

[8] Hastings, Alan. The Art of Analog Layout. Prentice Hall. New Jersey: Upper Saddle River, 2001. 194-212.

[9] Lopez-Villegas, Jose M., et al. "Improvement of the Quality Factor of RF Integrated Inductors by Layout Optimization," IEEE Transactions on Microwave Theory and Techniques, Vol. 48, No. 1, January 2000.

[10] Sedra, Adel S., and Kenneth C. Smith. Microelectronic Circuits. 4th Ed. New York: Oxford University Press, 1998. 984-986.

[11] Agilent. "Agilent 4284A Precision LCR Meter: Data Sheet," August 13, 2002. http://cp.literature.agilent.com/litweb/pdf/5963-5390E.pdf

[12] Chemtronics. "CircuitWorks Conductive Epoxy TDS# CW2400," 8125 Cobb Center Drive, Kennesaw, GA 30152-4386 USA. http://www.chemtronics.com/tds2/CW2400.pdf

[13] Sutono, Albert, et al. "Experimental Modeling, Repeatability Investigation and Optimization of Microwave Bond Wire Interconnects," IEEE Transactions on Advanced Packaging, Vol. 24, No. 4, November 2001.

[14] Zu, Larry, et al. "High Q-Factor Inductors Integrated on MCM Si Substrates," IEEE Transactions on Components, Packaging, and Manufacturing Technology – Part B, Vol. 19, No. 3, August 1996.

[15] Chang, J.Y.C., et al. "Large Suspended Inductors on Silicon and Their Use in a 2-μm CMOS RF Amplifier," IEEE Electron Device Letters, Vol. 14, No. 5, May 1993.