

WIDEBAND LOW-PHASE SHIFT DIGITAL STEP ATTENUATORS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ARASH EBRAHIMI JARIHANI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2016

Approval of the thesis:

WIDEBAND LOW-PHASE SHIFT DIGITAL STEP ATTENUATORS

submitted by **ARASH EBRAHIMI JARIHANI** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Tolga Çiloğlu
Head of Department, **Electrical and Electronics Engineering**

Assist. Prof. Dr. Fatih Koçer
Supervisor, **Electrical and Electronics Eng. Dept., METU**

Examining Committee Members:

Prof. Dr. Tayfun Akın
Electrical and Electronics Engineering Dept., METU

Assist. Prof. Dr. Fatih Koçer
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Haluk Külâh
Electrical and Electronics Engineering Dept., METU

Assoc. Prof. Dr. Oğuz Ergin
Computer Engineering Dept., TOBB ETÜ

Assist. Prof. Dr. Serdar Kocaman
Electrical and Electronics Engineering Dept., METU

Date:

02.09.2016



I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: ARASH EBRAHIMI JARIHANI

Signature :

ABSTRACT

WIDEBAND LOW-PHASE SHIFT DIGITAL STEP ATTENUATORS

Arash Ebrahimi Jarihani,

M.S., Department of Electrical and Electronics Engineering

Supervisor : Assist. Prof. Dr. Fatih Koçer

September 2016, 128 pages

Digital Step attenuators are used in many communication systems to regulate the signal level to achieve maximum system performance. The figure of merit of these attenuators are linearity, accuracy, and their power handling capabilities. Attenuators currently offered in the industry performs well to meet stringent requirements of the modern communications systems. However, all step attenuators currently offered introduces a phase-shift with respect to the attenuation setting and frequency. This phase-shift creates problems in communications systems, where multiple antennas and phased array systems are employed. In phased array systems, in order to create desired beam formations, the amplitude excitation and the phase excitation of each element in the array should be controlled accurately. In most applications, the amplitude excitation value is controlled with a digital step attenuator, whereas the phase excitation is controlled with a digital phase shifter in the transmitter path. In such systems, it is very important for the amplitude and the phase excitation values to be controlled independent of each other to simplify the design of a phased array system. Therefore, the development of phase coherent digital step attenuators is crucial for communication systems.

This thesis presents novel digital step attenuators (DSAs) with very low phase variation under attenuation state and frequency changes. Thanks to their phase coherency, the presented attenuators are very useful in phased array systems, where attenuation and phase of each element should be adjusted accurately to achieve desired beam

formation and null points. Several methods to address the phase shift introduced by the low-pass, high-pass filter response of the attenuator core under insertion loss and attenuation states respectively are investigated, while keeping all other specifications (linearity, accuracy, etc.) on par with the state of the art. Compensation circuit include inductive and capacitive switchable blocks.

Firstly, an ideal attenuator is designed and implemented for the proof of concept. Then, 3D EM simulations are utilized to analysis the effects of the wirebonds inside a commercial package. With these data, a mathematical model is implemented for an attenuation block. Based on the mathematical model, an automated MATLAB program with a graphical user interface (GUI) is developed which is used to determine the initial values for compensation elements. With the initial values gathered from the MATLAB program, the attenuators are implemented in Cadence Design System and compensation elements are further optimized. The attenuators are integrated with the digital control circuitry on the same chip. Finally, the fabricated chips (encapsulated in commercial packages) are soldered on an printed circuit evaluation board for measurements.

Unlike previous studies on phase coherent DSAs, the presented work achieves very low phase variations in a quad-flat no-leads (QFN) package, where wirebond effects are significant. The presented attenuators have 6-bit and 7-bit control range and achieves accurate attenuation settings in 0.5 dB and 0.25 dB steps respectively, in a very wide frequency range covering LTE (Long-Term Evaluation) 2600 and all GSM bands (100 MHz to 2.7 GHz). The attenuators are fabricated on a commercial RF Silicon-On-Insulator (SOI) process. These attenuators demonstrate an performance far better than the any attenuator developed to date (both commercial and research). The measurements results show that the attenuators exhibit amplitude and phase errors of less than 1.5 dB and $\pm 3^\circ$, respectively up to 2.5 GHz, where the insertion loss is less than 3 dB. Worst case return loss is -14 dB across the complete frequency band. Moreover, very low phase shift is achieved without sacrificing linearity. The input 1 dB compression point and the input third order intermodulation point are measured at 33 dBm and 49 dBm respectively. Total chip size, including pads, is 1.95 mm $\times$ 0.95 mm.

Keywords: Digital Step Attenuator, Phase Coherent Attenuator, SOI, High power and high linearity Attenuator, Digital Control Circuitry

ÖZ

GENİŞ BANTLI DÜŞÜK FAZ KAYMALI SAYISAL SİNYAL ZAYIFLATICILARI

Arash Ebrahimi Jarihani,
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Yrd. Doç. Dr. Fatih Koçer

Eylül 2016 , 128 sayfa

Sayısal zayıflatıcılar maksimum sistem performansı elde etmek ve sinyal seviyesini düzenlemek için hemen hemen tüm haberleşme sistemlerinde kullanılmaktadır. Bu zayıflatıcılar için önemli isterler arasında lineerlik, hassaslık ve güç dayanırlığı yetenekleri vardır. Şu anda sektörde sunulan zayıflatıcılar modern iletişim sistemlerinin zor gereksinimlerini karşılamak için iyi bir performans sergilemektedirler. Ancak, şu anda sunulan tüm zayıflatıcılar zayıflama ayarı ve frekans değerine bağlı olan bir faz kayması yaratırlar. Bu faz kayması çoklu anten ve dizi anten sistemlerinin kullanıldığı yeni nesil iletişim sistemlerinde (4G ve sonrasında) sorunlar oluşturur. Bu dizi anten sistemlerinde, istenilen huzmeyi oluşturmak için her elemanın genlik ve fazı ayrı ayrı kontrol edilmelidir. Bir çok uygulamada, genlik sayısal zayıflatıcılar ile kontrol edilirken, faz bir faz değiştiricisi ile kontrol edilir. Bu gibi sistemlerde, genlik ve fazın birbirinden bağımsız olarak kontrol edilmesi çok önemlidir. Bu nedenle, faz uyumlu sayısal zayıflatıcıların geliştirilmesi özellikle yeni nesil iletişim sistemleri için çok önemlidir.

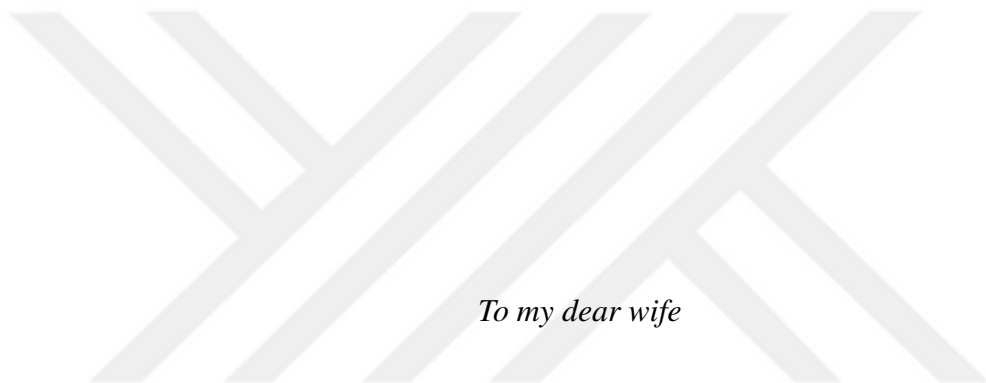
Bu tezde zayıflama seviyesi ve kullanım frekansından etkilenmeden çok düşük faz kayması sunan sayısal zayıflatıcılar tasarlanmıştır. Sayısal zayıflatıcıların araya girme ve zayıflatma durumlarındaki sırasıyla alçak ve yüksek geçirgen filtre gibi davranmasından kaynaklanan faz kaymasını engellemek için değişik teknikler geliştirilmiştir. Geliştirilen bu tekniklerle halihazırda piyasada sunulan sayısal zayıflatıcılarla aynı

özellikleri (doğrusallık, doğruluk, vb) sağlayabilen aynı zamanda faz kaymasının düşük seviyelerde tutulduğu yeni zayıflatıcı prototipleri gerçekleştirilmiştir.

Öncelikle, zayıflatıcıların çalışma prensiplerinin ve faz kaymasının nedeninin anlaşılması için ideal bir zayıflatıcı geliştirilmiştir. Daha sonra, üç boyutlu elektromanyetik benzetimler ile lehim tellerinin ticari paket içinde etkisi analiz edilmiştir. Bu bilgiler kullanılarak, sayısal zayıflatıcı bloğunun matematiksel modeli yaratılmıştır. Yaratılan matematiksel model kullanıcı arayüzü olan bir MATLAB programında derlenmiş ve bu program kullanılarak düşük faz kaymalı sayısal sinyal zayıflatıcının öncül tasarımı tamamlanmıştır. Öncül tasarımdan gelen devre elemanları değerleri kullanılarak tasarlanan zayıflatıcılar Cadence Design Systems paket programına şematik düzeyde aktarılmış ve Spectre benzetimleriyle hassas ayarları yapılmıştır. Tasarlanan zayıflatıcıyı kontrol etmek için bir sayısal kontrol devresi tasarlanıp, zayıflatıcılar ile aynı taban üzerinde entegre edilmiştir.

Önceki araştırmaların aksine, gerçekleştirilen zayıflatıcılar lehim tellerinin çok önem taşıdığı ticari bir QFN paket içinde çok az faz kayması sunmuştur. Gerçeklenen zayıflatıcılar 6-bit ve 7-bit kontrol kapasitesinde sırasıyla 0.5 dB ve 0.25 dB adım aralığı sunarken LTE 2600 ve tüm GSM bantlarını (100 MHz - 2.7 GHz) kapsamaktadırlar. Tasarlanan zayıflatıcılar ticari bir SOI tabanda üretilmiştir. Bu zayıflatıcılar literatürde geliştirilen herhangi bir zayıflatıcıdan elde edilen en iyi performanslardan daha iyi bir performans göstermektedir. Ölçüm sonuçları genlik ve faz hatalarının 2.5 GHz frekansına kadar sırasıyla 1.5 dB ve $\pm 3^\circ$ mertebesinde olduğunu aynı frekansta araya girme kaybının ise sadece 3 dB olduğunu göstermiştir. 100 MHz den 2.5 GHz frekans aralığında en kötü geri dönüş kaybı -14 dB dir. Düşük faz kayması, zayıflatıcıların lineerliğini bozmadan elde edilebilmiştir. Giriş 1 dB sıkıştırma noktası ve üçüncü dereceden giriş iç girişim noktası sırasıyla 33 dBm ve 49 dBm mertebesinde ölçülmüştür. Bağlantı noktaları dahil toplam çip boyutu, 1.95 mm $\times$ 0.95 mm dir.

Anahtar Kelimeler: Sayısal Zayıflatıcı, Faz-uyumlu sayısal zayıflatıcı, SOI, Yüksek güç ve çok doğrusal zayıflatıcı



To my dear wife

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere thanks to my supervisor Assist. Prof. Dr. Fatih Koer for his guidance, support, and for his effort to motivate me throughout my master study. I would like to thank him, who is the originator of this thesis, for a chance to introduce me to the world of RF, and his time devoted me for the development of this thesis.

I would like to thank The Scientific and Technological Research Council of Turkey (TÜBİTAK) for the scholarship and support for me.

I am thankful to all of my previous and present lab friends for their kind friendship. Moreover, I would like to express my particular gratitude to my dear friends, for sharing their knowledge with me and motivating me in every difficulty I have faced. I also express my appreciation to Oğuz Kazan for his helps and contribution in automated measurement setup.

I am very grateful to my parents for their encouragement, patience, and endless support throughout my entire life. My appreciation also goes to my father and mother-in-law for their prayers and encouragement.

Last but not least, I would like to thank to my dear wife, Mahzad, for her endless love, for always standing by me, comforting me, and believing in me. I thank her for giving me the courage and motivation to keep going!

TABLE OF CONTENTS

ABSTRACT	v
ÖZ	vii
ACKNOWLEDGMENTS	x
TABLE OF CONTENTS	xi
LIST OF TABLES	xiv
LIST OF FIGURES	xv
LIST OF ABBREVIATIONS	xx

CHAPTERS

1	INTRODUCTION	1
1.1	Overview of Attenuators	3
1.2	Previous Studies	5
1.3	Research Objectives and Thesis Organization	7
2	THEORY AND DESIGN OF DIGITAL STEP ATTENUATORS	9
2.1	Ideal Digital Step Attenuator Design	9
2.2	Proposed Digital Step Attenuator Design	12
2.2.1	Global Foundries CSOI7RF Technology	12

2.2.2	Proposed Attenuator Core Structure	13
3	DESIGN OF PHASE COHERENT ATTENUATOR	21
3.1	Theory Of Design and Mathematical Modeling	26
3.2	Implementation in Cadence	31
3.3	Gate and Body Floating Technique and Control Voltages . . .	44
3.4	Digital Control Circuitry	47
3.4.1	Monte Carlo and corner simulations	52
3.4.2	Supply voltage variation simulations	53
4	TEST SETUP AND MEASUREMENT RESULTS	57
4.1	S-parameters Measurement Setup	59
4.1.1	7-bit Attenuator Measurements Results	60
4.1.2	6-bit Attenuator Measurements Results	66
4.2	Linearity Measurement Setup	71
4.2.1	7-bit Attenuator Linearity Measurements	71
4.2.2	6-bit Attenuator Linearity Measurements	74
4.3	Performance Summary	75
5	CONCLUSION AND FUTURE WORK	77
	REFERENCES	81
	APPENDICES	
A	AUTOMATED MATLAB ALGORITHM	87
B	ARDUINO CONTROL CODE	95

C	C # CODE IN MICROSOFT VISUAL STUDIO	97
---	---	----



LIST OF TABLES

TABLES

Table 2.1	Maximum current values passing through the resistors at each block.	16
Table 2.2	summary of design requirements for components of each block. . . .	17
Table 3.1	Optimized components values for designed 6 bits of phase coherent attenuator.	43
Table 3.2	Exact values for R_{Large} resistor at different attenuation blocks respect to width of the transistors.	47
Table 4.1	Parallel Truth Table of 7-bit Attenuator	58
Table 4.2	Electrical Specifications: 0.25 dB steps, 7-bit Attenuator	75
Table 4.3	Electrical Specifications: 0.5 dB steps, 6-bit Attenuator	76
Table 4.4	Performance comparison of digital step attenuators	76

LIST OF FIGURES

FIGURES

Figure 1.1	Typical T/R module block diagram	1
Figure 1.2	Relative phase error vs. Frequency [4].	2
Figure 1.3	Three main attenuator core: (a) Pi, (b) T ,and (c) bridge-T.	3
Figure 2.1	6-bit DSA block diagram with 0.5 dB attenuation steps	10
Figure 2.2	Three main attenuator core structures: (a) Pi, (b) T and (c) bridge-T.	10
Figure 2.3	Ideal switched bridge-T attenuator	11
Figure 2.4	Relative attenuation of bridge-T type DSA with 0.5 dB steps for 64 different states.	12
Figure 2.5	Typical schematic of switched bridge-T attenuator	13
Figure 2.6	Switched attenuator in: (a) attenuation and (b) insertion loss states.	14
Figure 2.7	Voltage drop across each attenuation block.	15
Figure 2.8	Voltage drop across shunt transistor of each attenuation block.	16
Figure 2.9	Simulation results of conventional attenuator for 8 dB attenuation: (a) Attenuation and insertion loss. (b) Relative attenuation.	18
Figure 2.10	Schematic of compensated attenuator block.	19
Figure 2.11	Simulation results of compensated (with C_{comp}) attenuator for 8 dB attenuation: (a) Attenuation and insertion loss. (b) Relative attenuation.	19
Figure 2.12	Simulation results of conventional attenuator for 8 dB attenuation: (a) Transmission phase of reference and attenuation states. (b) Relative phase.	20
Figure 2.13	Simulation results of compensated attenuator for 8 dB attenuation: (a) Transmission phase of reference and attenuation states. (b) Relative phase.	20

Figure 3.1	Structure of QFN package	23
Figure 3.2	Schematic of attenuator including wire bonds	24
Figure 3.3	Simulation results of attenuator for 8 dB attenuation including wire bonds: (a) Attenuation and insertion loss. (b) Relative attenuation.	25
Figure 3.4	Simulation results of attenuator for 8 dB attenuation including wire bonds: (a) Transmission phase of reference and attenuation states. (b) Relative phase.	25
Figure 3.5	Switched: (a) Pi attenuator and (b) bridged-T attenuator [34].	26
Figure 3.6	The schematic of the p-i-n diode attenuator.	27
Figure 3.7	The schematic of phase coherent digital step attenuator.	28
Figure 3.8	Schematic of the reference and attenuation states of Mathematical Model.	29
Figure 3.9	Input panel and output plots for MATLAB program.	31
Figure 3.10	Designed attenuator block that provides low phase shift.	32
Figure 3.11	Simulation results of proposed Bridge-T attenuator for 8 dB attenuation with phase compensation circuit: (a) Relative attenuation and (b) phase difference with various C_{comp}	33
Figure 3.12	Simulation results of proposed Bridge-T attenuator for 8 dB attenuation with phase compensation circuit: (a)Relative attenuation and (b) phase difference with various C_p	34
Figure 3.13	Simulation results of proposed Bridge-T attenuator for 8 dB attenuation with phase compensation circuit: (a)Relative attenuation and (b) phase difference with various L_1	35
Figure 3.14	Simulation results of proposed Bridge-T attenuator for 8 dB attenuation with phase compensation circuit: (a)Relative attenuation and (b) phase difference with various L_2	36
Figure 3.15	Simulation results of optimized 8 dB attenuation block: (a) Attenuation and insertion loss. (b) Relative attenuation.	37
Figure 3.16	Simulation results of optimized 8 dB attenuation block: (a) Transmission phase of reference and attenuation states. (b) Relative phase.	37
Figure 3.17	Attenuation characteristic of the designed phase coherent attenuator compared with the conventional attenuator.	38

Figure 3.18 Transmission phase characteristic of the designed phase coherent attenuator compared with the conventional attenuator.	38
Figure 3.19 Figure of modeled chip, inside the QFN package by AWR Analyst.	39
Figure 3.20 Wire bonds inductive values vs. frequency.	39
Figure 3.21 Attenuation results of the 3D EM simulation.	40
Figure 3.22 0.5 dB Step Attenuation vs. Attenuation Setting.	40
Figure 3.23 Attenuation Error vs. Attenuation Setting.	41
Figure 3.24 Insertion Loss vs. Frequency.	41
Figure 3.25 Attenuation Error vs. Attenuation Setting.	42
Figure 3.26 Step attenuation vs. Attenuation Setting.	42
Figure 3.27 Layout of designed phase coherent attenuator.	43
Figure 3.28 Structure of a switch for control the attenuation blocks.	44
Figure 3.29 Structure of a switch including gate-body resistor.	45
Figure 3.30 Transient simulation results for different R_{Large} values	46
Figure 3.31 Schematic diagram of digital control circuitry.	48
Figure 3.32 Schematic diagram and symbol of the input driver.	48
Figure 3.33 Schematic diagram of control circuit.	49
Figure 3.34 Schematic of level shifter circuit.	50
Figure 3.35 Schematic of buffer circuit.	50
Figure 3.36 Schematic of low current voltage divider circuit.	51
Figure 3.37 Simulation results of digital control circuitry.	52
Figure 3.38 Monte carlo simulation for body voltage of transistors at -40°C	53
Figure 3.39 Corner simulation result when both of the inverter transistors are fast.	54
Figure 3.40 Corner simulation result when both of the inverter transistors are slow.	54
Figure 3.41 Supply variation (10% tolerance) simulation results.	55
Figure 3.42 Layout of digital control circuitry.	56

Figure 4.1	Evaluation board schematic.	57
Figure 4.2	Evaluation board layout.	58
Figure 4.3	Photograph of the test setup used for automatic s-parameters measurements.	59
Figure 4.4	Microphotograph of the 7-bits digital step attenuator wire-bonded to the QFN package.	61
Figure 4.5	Measured relative attenuations of the DSA (all states).	62
Figure 4.6	Major state bit error vs. Frequency.	63
Figure 4.7	Measured: (a) Step and (b) Attenuation error vs. Attenuation setting.	63
Figure 4.8	Measured: (a) Input and (b) Output return loss vs. Frequency.	64
Figure 4.9	Measured relative phases of the DSA (all states).	64
Figure 4.10	Relative phase error vs. Frequency.	65
Figure 4.11	Relative phase error vs. Attenuation setting.	65
Figure 4.12	Measured RMS phase error vs. Frequency.	66
Figure 4.13	Measured relative attenuations of the DSA (all states).	67
Figure 4.14	Major state bit error vs. Frequency.	68
Figure 4.15	Measured: (a) Step and (b) Attenuation error vs. Attenuation setting.	68
Figure 4.16	Measured: (a) Input and (b) Output return loss vs. Frequency.	68
Figure 4.17	Measured relative phases of the DSA (all states).	69
Figure 4.18	Relative phase error vs. Frequency.	69
Figure 4.19	Relative phase error vs. Attenuation setting.	70
Figure 4.20	Measured RMS phase error vs. Frequency.	70
Figure 4.21	P_{1dB} compression point of the proposed DSA	71
Figure 4.22	Block diagram of IIP3 measurement setup.	72
Figure 4.23	Setup photograph of IIP3 measurements.	72
Figure 4.24	Measured IIP3 for major states of 7-bit attenuator at different frequencies.	73

Figure 4.25 P_{1dB} compression point of the proposed 6-bit DSA.	74
Figure 4.26 Measured IIP3 for major states of 6-bit attenuator at different frequencies.	74



LIST OF ABBREVIATIONS

AGC	Amplitude Gain Control
CAD	Computer Aided Diagnosis
CMOS	Complementary Metal Oxide Semiconductor
DSA	Digital Step Attenuator
DUT	Device Under Test
FET	Field Effect Transistor
GaAs	Gallium Arsenide
GSM	Global Systems for Mobile Communications
HEMT	High Electron Mobility Transistor
IC	Integrated Circuit
LTE-A	Long Term Evolution-Advanced
LSB	Least Significant Bit
MMIC	Microwave Monolithic Integrated Circuit
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
PCB	Printed Circuit Board
SOI	Silicon On Insulator
SPI	Serial Peripheral Interface
TTL	Transistor-Transistor Logic

CHAPTER 1

INTRODUCTION

Demand for wireless data is increasing exponentially. To address this demand transceiver/receiver systems employing phased array antennas with their high gain and directivity and beam steering capabilities, have gained popularity. For example, LTE-A (Long Term Evolution-Advanced) communication backbone uses antenna arrays to dynamically steer the beam into crowded areas to address the increase in the bandwidth demand [1], [2]. In phased array systems, in order to create desired beam formations, the amplitude excitation and the phase excitation of each element in the array should be controlled accurately. In most applications, the amplitude excitation value is controlled with a digital step attenuator, whereas the phase excitation is controlled with a digital phase shifter in the transmitter path as shown in Figure 1.1.

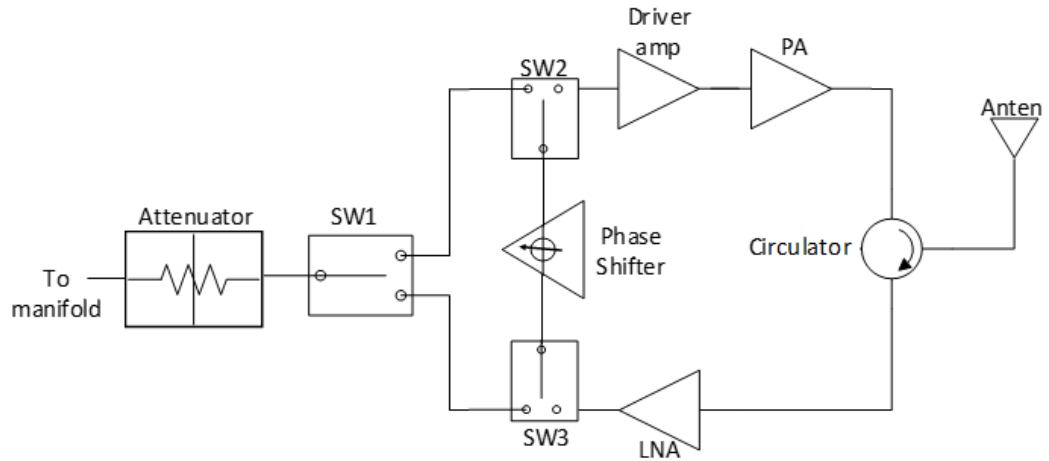


Figure 1.1: Typical T/R module block diagram

In such systems, the width and the sidelobe level of the beam are tailored by apply-

ing amplitude weighting with the attenuator (gain control circuit), while the beam direction is controlled by the phase shifter. In such systems, it is very important for the amplitude and the phase excitation values to be controlled independent of each other. In other words, the step attenuator must have a very low insertion phase variation over different attenuation states at different operation frequencies to avoid tracking errors. There are numerous off-the-shelf digital step attenuators offered by semiconductor companies supporting various frequencies and communication protocols [3], [4]. However, all of these attenuators introduce a large phase shift that varies considerably with the attenuation state and frequency. The total phase shift of these attenuators, regardless of their fabrication technology, is typically in the order of 30° to 40° at 4 GHz [4]. For example, Figure 1.2 shows the phase shift of a state of the art DSA sold by peregrine semiconductor (Product No. PE43704).

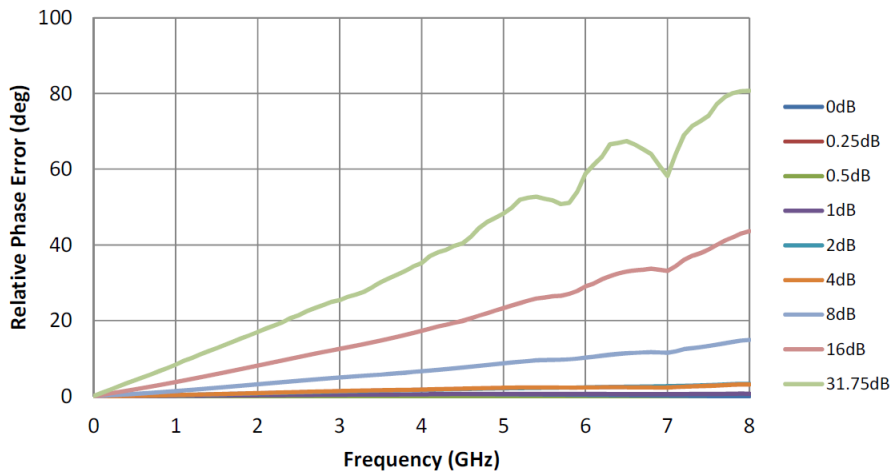


Figure 1.2: Relative phase error vs. Frequency [4].

This intrinsic phase shift affects the successful formation of the beam and thus should be avoided or compensated. The phase shift can be compensated with the help of the phase shifter via a look-up table, which complicates the design of a phased array system considerably [5–7]. Therefore, the development of phase coherent digital step attenuators is crucial for next generation communication systems. In this work, several methods are investigated to compensate the phase shift of digital step attenuators while keeping all other specifications (linearity, accuracy, etc.) on par with the state of the art.

1.1 Overview of Attenuators

An attenuator is an electronic device that reduces the power of the input signal. Many topologies have been proposed and most of them are based on the simple implementation of the attenuator using the resistive-network. Among these topologies, there are three different forms of resistive networks that are used as basic configuration for the attenuator (Pi, Tee, and Bridge-Tee) [8–12]. These topologies are illustrated in Figure 1.3.

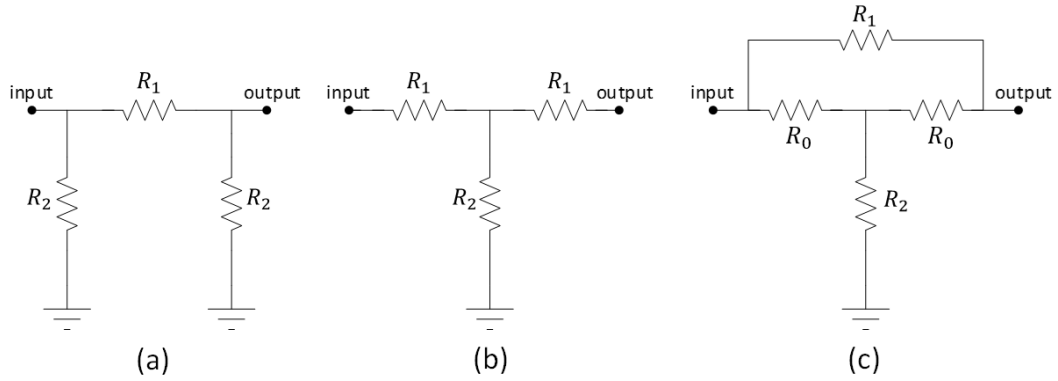


Figure 1.3: Three main attenuator core: (a) Pi, (b) T, and (c) bridge-T.

Attenuators are used in communications systems to control signal amplitude (e.g., variable gain amplifiers, amplitude gain control (AGC) loops) both in the receive and the transmit path [13], [14].

Attenuators can be divided into two main categories depending on their control type; analog and digital. An analog attenuator requires an analog voltage to adjust its attenuation level [15–18]. They have a number of shortcomings, which limit their use in many applications. For example, their control voltage versus attenuation slope is usually nonlinear, exhibiting varying gain slopes through the control range [19], [20]. This makes the control circuit difficult to design and also may introduce oscillations in the AGC loop, if not designed carefully. They usually have worse linearity when compared to DSAs. This is due to the fact that transistors used in the attenuator core are used as varying resistor elements, introducing large distortion characteristics, especially around the transistor pinch-off voltages [21–23].

Digital attenuators accept a digital code instead of an analog input (binary code).

They are classified with respect to the number of control bits (e.g., 6-bit, 7-bit control) with different resolutions. The resolution of DSA's are determined with their minimum attenuation step (i.e., least significant bit LSB)). As an example, a 7-bit 0.25 dB LSB DSA provides 0 to 31.75 dB attenuation in 128 steps separated from each other by 0.25 dB. These types of attenuators are mostly designed with High Electron Mobility Transistors (HEMT) and Metal Oxide Semiconductor Field Effect Transistors (MOSFET), where transistors are used as switching elements in their most linear regime: cut-off or linear [24], [25]. Therefore, digital step attenuators can be designed to be very linear. The input third order intermodulation distortion (IIP3) value for a typical DSA can be larger than 50 dBm [26], [27].

A digitally controlled step attenuator is preferable to a variable gain amplifier or an analog attenuator in terms of control complexity, power consumption, phase variation, and linearity. However, DSA's have some disadvantages as well. When used in an AGC loop, they require an analog-to-digital converter (ADC) to convert the analog output of the power detector to a digital code. This is usually a minor problem since many systems already include ADCs and microcontrollers to perform various tasks.

In the literature, different designs of DSAs have been demonstrated. CMOS switches (as a varistor), distributed attenuators using p-i-n diodes, switched path attenuators, and switched Pi/T attenuators are examples of these designs [28] – [32]. Switched attenuators may have different processes consist of CMOS, GaAs (Gallium Arsenide), and SOI (Silicon On Insulator) at different frequency bands.

In order to build a good switched attenuator, it is obvious that one should chose a technology that can provide transistors with good switching characteristics. Historically GaAs processes with their high electron mobility (HEMT) transistors have been the technology of choice due to their suitability for high frequency and high power applications. There are international companies such as Hittite Microwave, Skyworks, RFMD and M/A-COM still use GaAs processes to produce attenuators.

Recently, it can be seen that there is a growing number of attenuators, which are designed on SOI process [33]. The popularity of SOI processes in attenuator products is due to the fact that these processes offer complementary devices which can be used to build complex digital functions along with the RF core on the same substrate. The

lack of complementary devices along with the finite gate current leakage of HEMT devices on GaAs substrate makes it very hard to build digital circuitry without dissipating static power. Moreover, RF SOI processes are winning over their GaAs counterparts due to their robustness against ESD shocks and lower cost due to larger wafer size.

In this study an RF SOI process has been chosen. This is due to the fact that, compared to standard CMOS FET devices, SOI devices have higher drive currents, less junction capacitance, reduced short channel effects, and better sub-threshold slope. Moreover, the chosen technology has intrinsic immunity to radiation induced latch-up effects. The advantages of this process are discussed in detail in the following chapters.

1.2 Previous Studies

Attenuators currently offered in the industry perform well to meet stringent requirements of the modern communications systems. However, all step attenuators currently offered introduces a phase-shift with respect to the attenuation setting and frequency. This phase-shift creates problems in next generation communications systems (4G and above) where multiple antennas and phased array systems are employed.

There are numerous studies that can be found in the literature tackling the inherent phase shift of attenuators. Ku et al. [34] provide a very detailed analysis of the resistive attenuator core, to show the root cause of the phase shift. They propose using a low pass filter in the attenuation state to lag the phase of the system. Although simple in implementation the proposed method has some serious flaws. First of all, their study ignores the effect of bondwires, which have significant effect in the input/output pins, and as will be shown, more importantly in the ground downbond connections. The authors used a technology that offers substrate vias, to minimize the downbond bondwire effect, and measured their system using a calibrated probe station. However, substrate vias are only commercially available in III-IV technologies, and are not currently offered in RF SOI processes. Therefore, it is important to develop an attenuator which is phase compensated in the presence of downbond wires

and a commercial package.

A similar study by Ciccognani [31] proposes using a shunt phase compensation element in parallel with the attenuator core. In addition to the negligence of the bondwire effects, this technique achieves phase compensation in a very narrow bandwidth. Moreover, attenuation errors and linearity are not reported. Jeong [33] designed an attenuator on an SOI process which enhanced power handling capability in the REF mode and improved the isolation performance in the attenuation mode. It was tested by on-wafer measurements. The measured attenuation performance from 1.4 to 2.4 GHz is presented which is very narrow band without any compensation for phase shift.

Another study by Cho [35], presents 5-bit DSA on 0.18 μm IBM 7RF SOI technology. Similar to the previous researches, on-wafer measurement is performed, ignoring all bondwire and packaging effects. This work have high insertion loss due to using SPDT switches and low resolution. Moreover, linearity performance of the attenuator is poor, achieving an input P_{1dB} of approximately 10 dBm at 10 GHz.

Dai (2007) [36], tried to solve the problem by employing a number of MESFET transistors as linearly changing resistive elements, very much like an analog attenuator. Although this technique achieves stable phase, due to using transistors in their most non-linear saturation regime (as opposed to cut-off and linear), the linearity is low.

Using transistors in their non-linear regime is employed by Wagner [37] in an X-type attenuator. Due to the differential architecture, the effects of bondwire inductors are successfully avoided. However, similar to [36] not constraining transistors to their most linear regime results in low linearity. Digital attenuator implementations built with p-i-n diodes [38], [39], provide promising phase shift results. However, their control is hard and they cannot be integrated with the control block, and thus are not preferred.

Dogan et al. [40] in 2008 designed an attenuator, which is fabricated in 0.13m CMOS process, and did a general research about the analysis and optimization of the attenuators. However, the designed chip can not be used for many GSM applications because of the narrow band characteristics and low power handling caused by the

CMOS devices of the chosen technology.

1.3 Research Objectives and Thesis Organization

The aim of the proposed thesis is to develop phase coherent (low phase-shift) digital step attenuators on a commercial SOI technology with attenuation accuracy and power handling capabilities on par with the state of the art. The fabricated chips will ease the design of communication systems that employs phased array antennas. Within this study, mathematical model of the phase coherent attenuator with the bond wires is created. This model is used in the design of phase compensation circuits to be integrated with the attenuator. As well as a MATLAB program based on the model is provided as a byproduct of the work done. By using this model, an initial design of the attenuator is completed and used as a starting point for more detailed circuit analysis by a SPICE simulator. Designs are completed on simulation and layout tools provided by Cadence Design Systems. Prototypes are fabricated on an SOI technology offered by Global Foundries.

The specific objectives of this study are listed as follows:

- 1) The need for phase coherent attenuators could not be met by the industry yet. Although there are some work done to address phase shift problem in the literature, the solutions offered ignore the effect of bond wires. Therefore, these solutions cannot be used in the field where a packaged part is required. The microchips in this research are designed to work in a commercial package. The aim of this research is to keep the phase shift between any state lower than $\pm 3^\circ$ at 4 GHz. Therefore, design of attenuators that have an order of magnitude, lower phase shift than that of any state of the art attenuator offered by the industry is the primary research goal.
- 2) Since the attenuators being developed are designed to be used in the backbone and base station applications, linearity of the part is an important figure of merit. The aim of this research is to have P_{1dB} compression point to be higher than 30 dBm and input third order intermodulation product (IIP3) value to be higher than 50 dBm for all attenuation levels. To the best of author's knowledge there are no attenuators with

this much linearity and power handling capability while offering very low phase shift.

3) Thanks to the complementary devices provided by the technology, the chip includes a digital control circuitry to control the attenuator. This control circuit has only parallel control mode and consists of seven TTL compatible control lines to select the desired attenuation state.

4) The prototypes are measured on a printed circuit board designed to work at high frequencies. It is used as an evaluation kit for designed attenuators, which can be controlled manually or using a computer program to do automatic measurements.

This thesis consists of 5 chapters as summarized below:

In chapter 2, a brief background information about an ideal attenuator is provided. The simulation results of the ideal attenuator are discussed, which are used to design and implement the proposed attenuator.

Chapter 3 deals with the design theory of the proposed phase coherent attenuator and describes the methods utilized for the attenuation and phase compensation. Then, using compensation methods of the phase coherent attenuator at one attenuation block, an automated MATLAB algorithm is presented. This MATLAB program is used for calculating the initial values for compensation elements. Moreover, modeled attenuator is implemented in Cadence and simulation results are illustrated. In addition, digital control circuitry to control the attenuator is presented.

Chapter 4 presents the measurement results of the fabricated phase coherent attenuators. Measurements presented are small signal and large signal measurements and their comparison with the results that are achieved in simulations.

Chapter 5 is the conclusion and provides suggestions for the related studies in the future.

CHAPTER 2

THEORY AND DESIGN OF DIGITAL STEP ATTENUATORS

This chapter presents design details and analysis results of a 6-bit 0.5 dB LSB Digital Step Attenuator built with ideal circuit elements.

The reasons of this study can be listed as follows:

- 1) To gather and develop a better understanding of the working principles of DSAs.
- 2) To check the maximum power dissipation of all resistive elements in the attenuator core via transient simulations. With this knowledge minimum width of the resistors will be set when the design is migrated to include resistors provided by the fabrication process. Maximum current through resistors is set by the maximum allowable power level of the attenuator which is set at 25 dBm.
- 3) To check the maximum voltage swing across all switches. With this knowledge, the number of series transistors required to replace each ideal switch will be calculated. Maximum voltage swing will be set by the input P_{1dB} specification of the attenuator, which is specified at 30 dBm.

2.1 Ideal Digital Step Attenuator Design

Generally, DSA's are created by cascading individual switchable attenuator blocks with different attenuation values. Figure 2.1 shows how a 6-bit DSA with 0.5 dB steps (from 0.5 dB to 16 dB attenuation blocks) can be built.

Each attenuator block shown in the Figure 2.1 can be set to either 0 dB (insertion loss

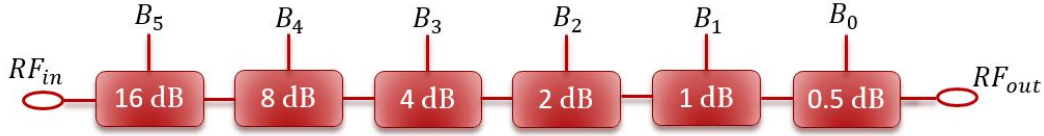


Figure 2.1: 6-bit DSA block diagram with 0.5 dB attenuation steps

or reference state) or X dB (attenuation state) according to the digital control signal. For example, if “zero” digital code is applied to all digital control bits, attenuator will be in the insertion loss state. Similarly, it will be in the 4 dB attenuation state if “one” digital code is applied only to B_3 , while the remaining bits are set to “zero”. Thus, the proposed DSA can be adjusted to reduce the power of the input signal with attenuation range from 0 dB to 31.5 dB with 0.5 dB steps which can be controlled by 2^6 different binary code combinations.

Digital step attenuators are usually passive devices made from simple resistive network structures (voltage divider networks). Each attenuator block shown above can be designed with various topologies. The most common topologies used in integrated circuit implementations are Pi, T, and bridge-T types due to their better matching characteristics. These topologies are illustrated in Figure 2.2.

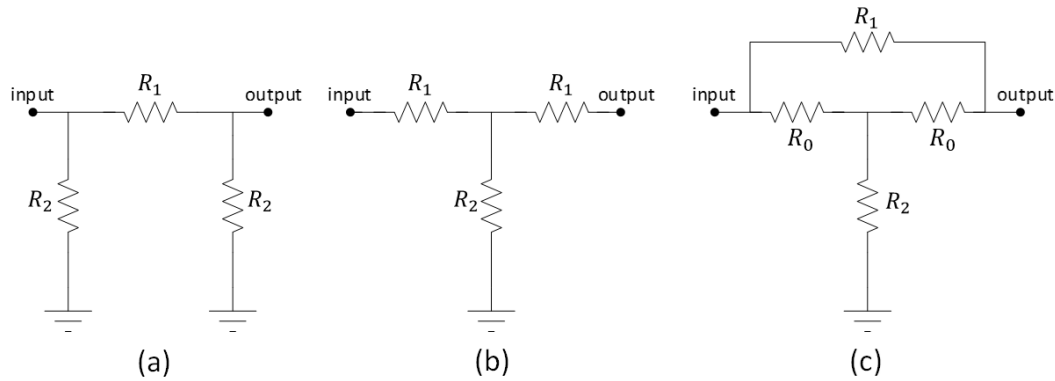


Figure 2.2: Three main attenuator core structures: (a) Pi, (b) T and (c) bridge-T.

These fixed attenuators can be used to construct switchable attenuators with the addition of switchable elements. For example, Figure 2.3 illustrates how a switchable attenuator can be implemented with the bridge-T attenuator core. If SW_1 is OFF and SW_2 is ON, the resistive network is placed in the signal path reducing the power of

the input signal. The effect of resistive network is nullified when SW_1 is ON and SW_2 is OFF (i.e., signal bypasses attenuator core).

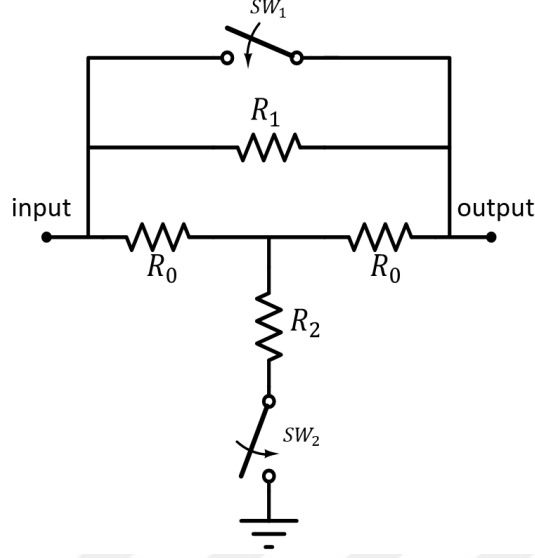


Figure 2.3: Ideal switched bridge-T attenuator

R_0 in Figure 2.3 is the characteristic impedance of the system. Since, the attenuator developed in this work is designed to work in wireless communication systems it is selected as 50Ω . The values of R_1 and R_2 with respect to the required attenuation value can be calculated by using equations (2.1) and (2.2) for the bridge-T topology [41]. In both equations, A is the desired attenuation value in dB.

$$R_1 = \frac{R_0}{(10^{\frac{A}{20}} - 1)} \quad (2.1)$$

$$R_2 = R_0 \times (10^{\frac{A}{20}} - 1) \quad (2.2)$$

A complete digital step attenuator with bridge-T attenuator cores is implemented and simulated in Cadence[®] by using idealcircuit elements (resistors and switches). Simulation results can be seen in Figure 2.4.

In the simulation shown in Figure 2.4, each line shows a different attenuation setting from DC to 8 GHz. Total attenuation is from 0 dB (insertion loss) to 31.5 dB increasing with 0.5 dB steps.

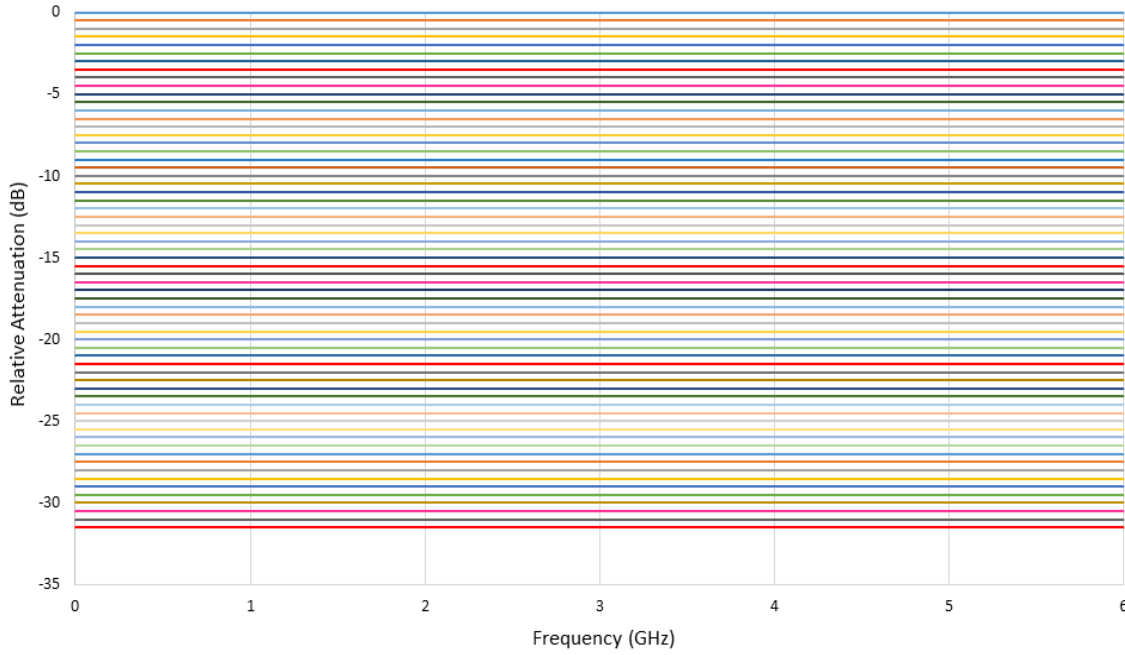


Figure 2.4: Relative attenuation of bridge-T type DSA with 0.5 dB steps for 64 different states.

The implemented ideal attenuator is used in the transient analysis to check the maximum power dissipation and the voltage swing across various elements. This information is used to adjust the width of resistors and the number of series transistors in the stack as will be detailed later in the following sections.

2.2 Proposed Digital Step Attenuator Design

This section presents the basic information about proposed digital step attenuator. Section 2.2.1 briefly explains the reasons for preferring the relatively expensive SOI technology (when compared to CMOS). Section 2.2.2 describes the design steps of switched bridge-T attenuator.

2.2.1 Global Foundries CSOI7RF Technology

Silicon-On-Insulator (SOI) technology was chosen to minimize parasitic capacitances associated with junction-to-well diodes, and to reduce substrate coupling. Because

devices are isolated from substrate, switch elements may be stacked to handle high power signals without requiring triple-well structures for devices isolation. For any switch process, it is important to have the minimum insertion loss (set by FET R_{on}) and highest isolation (set by FET C_{off}). A product of R_{on} and C_{off} is used as a figure of merit (FOM) for switches to compare different technologies. The selected SOI technology has excellent FOM that is approximately 250 fs. This figure of merit is comparable to that of state of the art GaAs technologies and much better than CMOS technologies [42], [43].

2.2.2 Proposed Attenuator Core Structure

The bridge-T attenuator core has been chosen in the design of the proposed attenuator, since it requires only two switching elements. This leads to simplicity in control and adjusting the attenuation value. Furthermore, it has a tendency to match itself to R_0 at high attenuation values, because of its two fixed resistors. The ideal attenuator built in the previous section is used as starting point to build the DSA to be fabricated in the selected technology. To do this, all ideal resistors are replaced with poly-silicon resistors offered by technology. Additionally, all ideal switches are replaced with n-type FET transistors that are optimized for switching applications. These changes can be seen in Figure 2.5, which illustrates a switched bridge-T attenuator.

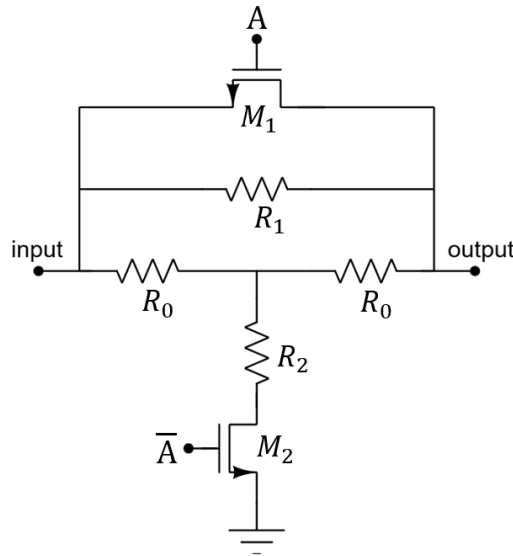


Figure 2.5: Typical schematic of switched bridge-T attenuator

Since the FET transistors act as a switches, they are used in their two regimes, OFF and Linear (ON). While the switch is in the OFF regime, it acts as open circuit and signal cannot pass through the switch. On the other hand, when the switch works in the linear regime, source and drain of the transistor are connected together and it acts as a short circuit (except for the channel resistance). Therefore, transistors that have been used as switches can be modeled by a resistor (R_{on}) in the ON state and a capacitor (C_{off}) in the OFF state. Schematics of both states are shown in Figure 2.6.

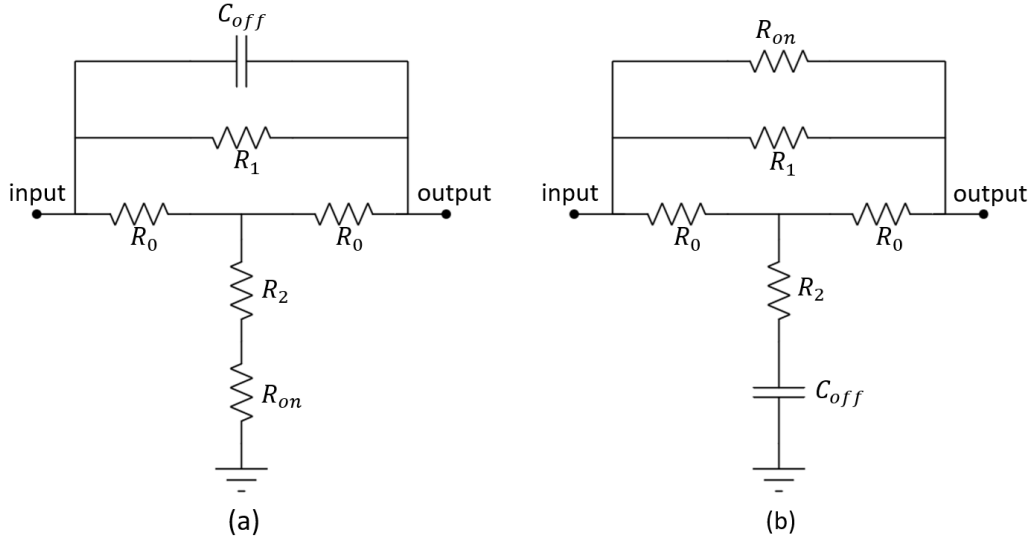


Figure 2.6: Switched attenuator in: (a) attenuation and (b) insertion loss states.

Referring to Figure 2.6, since R_{on} of M_2 is in series with R_2 in the attenuation state (Figure 2.6 (a)) the impedance of the shunt path change from its ideal value modifying the attenuation value. Likewise, in the reference state (Figure 2.6 (b)) insertion loss will be increased due to the finite ON resistance of M_1 . Moreover, C_{off} of M_1 causes signal leakage between the input and the output. Due to this signal leakage, attenuation will decrease with increasing frequency. Thus, the values of R_1 and R_2 should be optimized to achieve required attenuation value for each block during the migration from the ideal attenuator to the attenuator built with real elements.

Once the resistor values are selected to achieve desired attenuation level for each block. Physical properties of circuit elements (i.e., resistor and transistor) should be chosen with respect to the circuit design specifications. The design constraints that set the physical properties of the circuit elements are the input P_{1dB} compression and the maximum allowable input power of the attenuator.

For this design, 1 dB compression point is targeted at 30 dBm input power whereas the maximum allowable input power is limited at 25 dBm.

The P_{1dB} requirement is used to check the maximum voltage swing across each switch (transistor) for each attenuation case. Figure 2.7 is the transient analysis results showing the maximum voltage swing across each attenuator block. For this simulation, a sinusoidal signal with a power of 30 dBm is applied to the input of the attenuator while the attenuator is switched between various attenuation states.

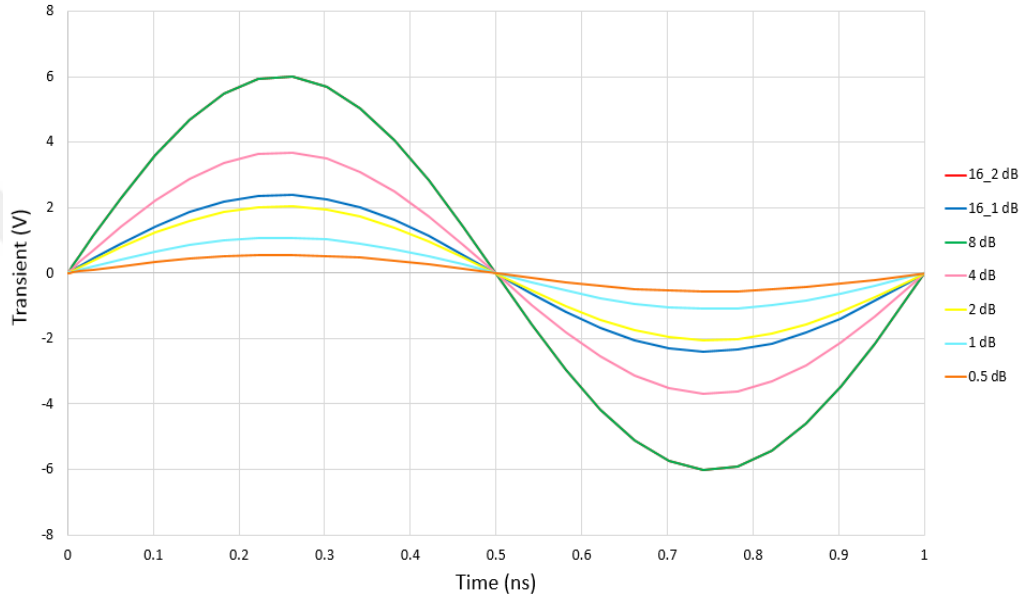


Figure 2.7: Voltage drop across each attenuation block.

This simulation result is used to determine the number of transistors that should be used in series to construct the switching elements in each block. As an example, 8 dB block experiences 6 V peak voltage drop across its terminals. Since the selected technology offers nFET transistors that can withstand a maximum of 2.5 V between its terminals, three series transistors are used for this particular block. By this way, sinusoidal swing with its 6 V peak is divided equally across 3 stacked nFET devices (when the transistors are OFF), guaranteeing no breakdown occurs.

Similarly, shunt path transistors in all the attenuation blocks experience 10 V peak voltage due to the 30 dBm input power at reference state (All shunt transistors are OFF), as shown in Figure 2.8. Therefore, four stacked transistors are used for the shunt path of all blocks.

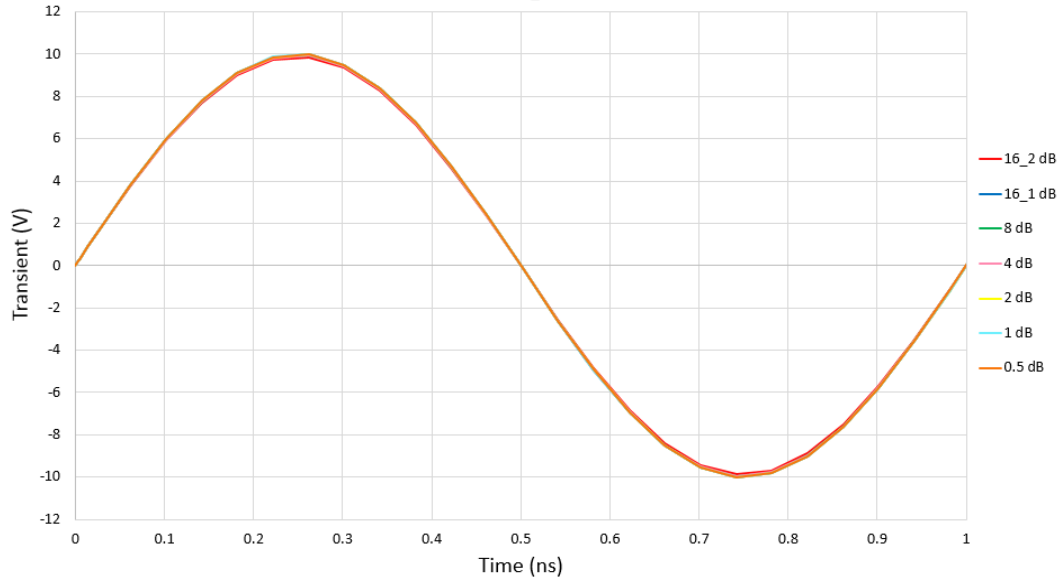


Figure 2.8: Voltage drop across shunt transistor of each attenuation block.

By stacking required number of nFET devices, transistor breakdown under large voltage is avoided, successfully increasing the power handling capability of the attenuator. Another important specification of the transistors are their total widths. Current passing through each transistor when they are ON is monitored at 30 dBm input power. With this knowledge the width of transistors are chosen accordingly.

In order not to push the circuit to compression below 30 dBm input power, the width of the transistors are chosen such that the AC current passing through each is at least 10 times smaller than their linear current (I_{lin}). This ensures small V_{ds} drop across transistors, increasing linearity.

Maximum allowable input power specification is used to choose the width of the resistors. Table 2.1 shows the maximum current passing through each one of the resistors.

Table 2.1: Maximum current values passing through the resistors at each block.

	8 dB (16/2)	8 dB (16/1)	8 dB	4 dB	2 dB	1 dB	0.5 dB
R_1 current (mA)	30.6	11.7	28.2	44.9	56.7	65.2	69.5
R_2 current (mA)	48.8	18.6	46.2	27.7	14.4	7.8	3.9
R_0 current (mA)	48.6	18.6	47	28.1	15.8	8.5	4.4

The maximum allowable current through the poly-silicon resistors offered by the

technology is limited by the power dissipation and the thermal resistance of them.

Equations (2.3), (2.4), and (2.5) which are provided by the factory, are used to calculate the minimum width of each resistor, so that the heat generated does not increase the temperature of the resistor higher than 125°C at 85°C ambient temperature (which is the junction temperature of the technology).

$$\Delta T = \text{ThermalResistance} \times \text{Power} \quad (2.3)$$

where:

$$\text{Power} = (\text{width} \times \text{allowed current})^2 \times R \quad (2.4)$$

$$\text{ThermalResistance} = 1.1847 \times 10^5 \times (\text{Area})^{-0.72433} \quad (2.5)$$

It should be noted that if 16 dB attenuation was achieved in a single attenuator block, there would be too much power dissipated in a single resistor necessitating the usage of very wide resistors. This in turn would increase the parasitics, decreasing the bandwidth. Therefore, 16 dB stage is constructed by the cascade connection of two 8 dB stages.

Design requirements gathered from the transient simulations are listed in Table 2.2.

Table 2.2: summary of design requirements for components of each block.

	8 dB (16/2)	8 dB (16/1)	8 dB	4 dB	2 dB	1 dB	0.5 dB
# Seris transistor	3	1	3	2	1	1	1
# Shunt transistor	4	4	4	4	4	4	4
R_1 width (μm)	94	25	84	133	160	167	160
R_2 width (μm)	148	40	139	82	40	20	9
R_0 width (μm)	160	43	154	76	34	17	10

Figure 2.9 (a) shows the simulation results of the 8 dB block with the design requirements as listed above (i.e., resistor widths, number of stacked transistors, etc.). It shows S_{21} of the block when it is set to the insertion loss and the attenuation states.

Figure 2.9 (b) shows the relative attenuation of the same block. It is created by subtracting the attenuation state from the insertion loss state.

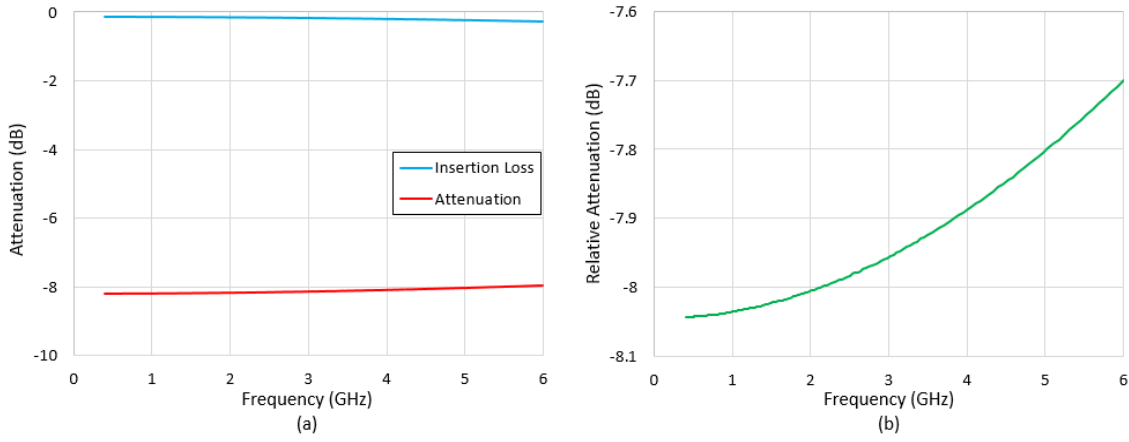


Figure 2.9: Simulation results of conventional attenuator for 8 dB attenuation: (a) Attenuation and insertion loss. (b) Relative attenuation.

As can be seen from the figure, 8 dB block presents a finite insertion loss value. This is expected due to the finite ON resistance of transistors. Moreover, insertion loss value increases with increasing frequency. This is due to the parasitic shunt capacitors of circuit elements, which were lacking in the ideal simulations.

Besides, attenuation value at the attenuation state decreases with respect to frequency. Although this may seem counterintuitive at first look, it is also expected. It should be noted that the off capacitance, C_{off} , of M_1 is in parallel with the attenuator core in the attenuation state. With increasing frequency, impedance of C_{off} decreases, creates a leakage path from input to output effectively decreasing the attenuation value.

With these two opposing effects, relative attenuation of the block changes with respect to the frequency and creates an error in the attenuation. This effect should be compensated. One way to compensate for the attenuation error is to add a capacitance in parallel with R_2 . This capacitance decreases the impedance of shunt path, effectively increasing the attenuation value with the increasing frequency. By carefully adjusting the value of the compensation capacitor, frequency roll-off characteristics of the insertion loss state can be matched with the attenuation state. The schematic diagram of a compensated attenuator is given in Figure 2.10.

It should be noted that values of compensation capacitance C_{comp} should be optimized

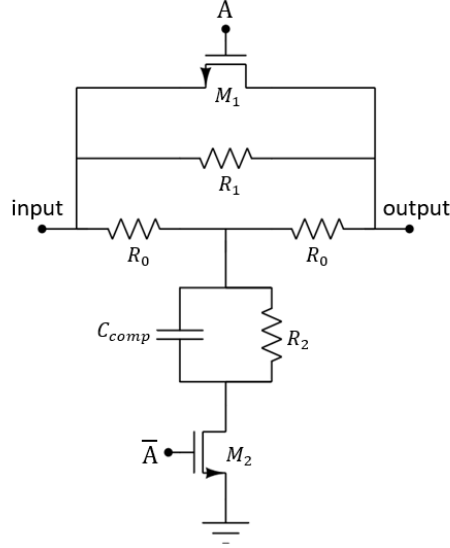


Figure 2.10: Schematic of compensated attenuator block.

for all individual attenuation blocks. The simulation result of the compensated 8 dB attenuation block is shown in Figure 2.11.

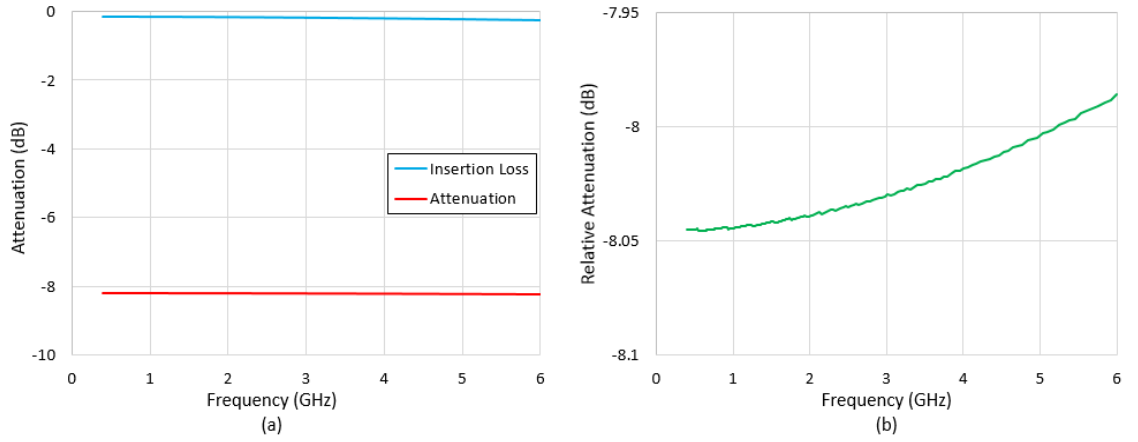


Figure 2.11: Simulation results of compensated (with C_{comp}) attenuator for 8 dB attenuation: (a) Attenuation and insertion loss. (b) Relative attenuation.

The purpose of this work is to design a phase coherent (low phase shift) DSA. For this reason, the phase shift of the attenuator should be investigated. It is observed during the simulations that due to the parasitic capacitances not only the attenuation value is affected (Figure 2.9), but also a high phase shift is occurred between the reference and attenuation states. Figures 2.12 and 2.13 illustrate transmission phase of the attenuation block before and after adding C_{comp} , respectively.

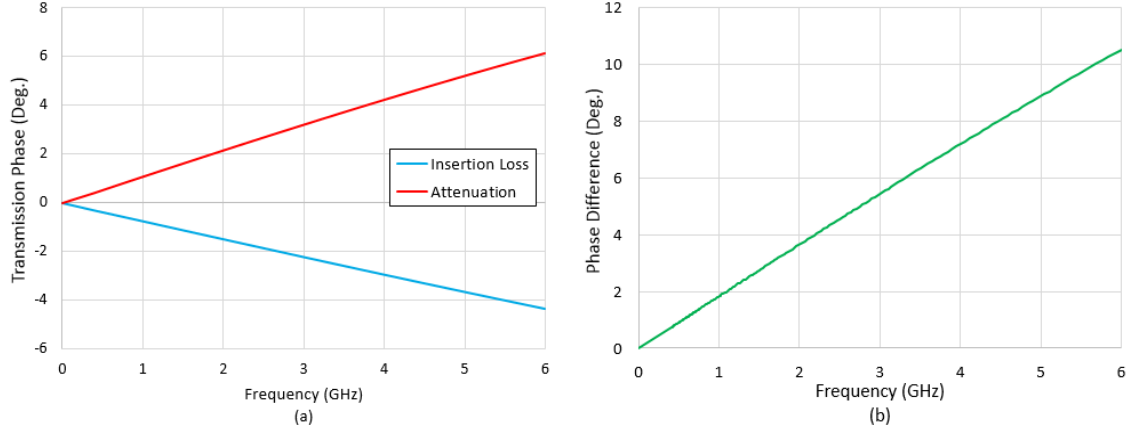


Figure 2.12: Simulation results of conventional attenuator for 8 dB attenuation: (a) Transmission phase of reference and attenuation states. (b) Relative phase.

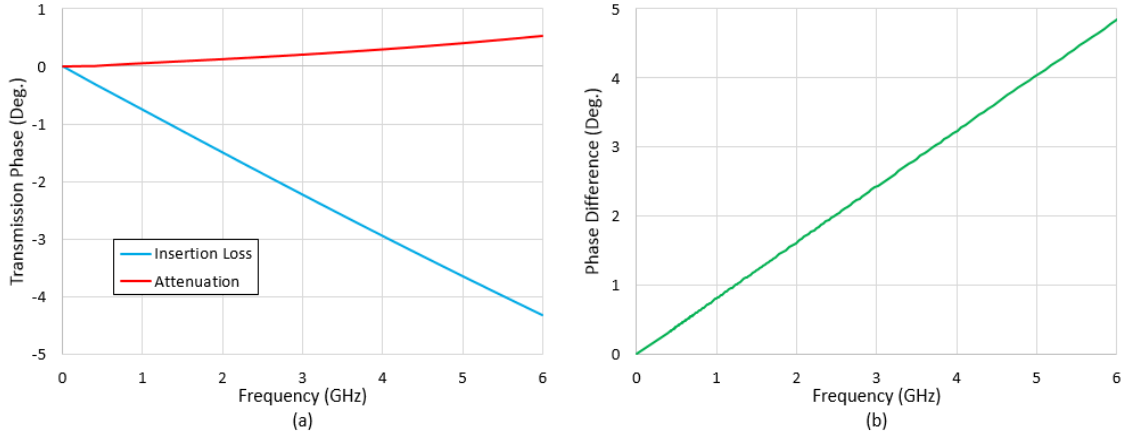


Figure 2.13: Simulation results of compensated attenuator for 8 dB attenuation: (a) Transmission phase of reference and attenuation states. (b) Relative phase.

In both situations, the attenuation state has a phase-lead compared to the reference state phase. In other words, attenuator work as a high-pass and a low-pass filter in attenuation and reference states, respectively. The errors of attenuation were corrected by the compensation capacitance with respect to frequency. The same capacitance reduced the phase shift in half, but did not get rid of it completely. Phase shift can be reduced to zero by increasing the value of C_{comp} . However, it would increase the attenuation error. Ignored up to this point is the bondwire effects, which will make the matter worse as will be shown later. Therefore, other methods are developed to correct the phase shift occurring in the system while keeping the attenuation error at acceptable levels. These methods are explained thoroughly in the following chapter.

CHAPTER 3

DESIGN OF PHASE COHERENT ATTENUATOR

This chapter presents the basic theory behind the design of a phase coherent DSA. First, the reasons of the phase shift is explained in detail. Then, the methods of compensating the phase shift are discussed. In section 3.1.1, a mathematical model is developed for the amplitude/phase compensation methods. In section 3.1.2, these compensation methods are implemented in Cadence®. Finally, section 3.2 describes the digital control circuit that is integrated in the same chip with the attenuator blocks.

As mentioned in the previous chapter, attenuation error due to the effect of the OFF state capacitance of transistors, C_{off} , in the attenuation state can be corrected by a compensation capacitance, C_{comp} . However, it cannot correct the phase shift problem completely because of two important factors that are neglected in literature to this day. These factors are detailed below.

1) The parasitic capacitance effects of circuit elements;

One of the targets of this study is to construct a phase coherent digital step attenuator with very good linearity. As discussed in the previous chapter, transistors are used as switching elements in DSAs. In the ON condition, these transistors are in their linear regime following the well know CMOS formula:

$$I_{DS} = \mu_n C_{ox} \frac{W}{L} ((V_{GS} - V_T)V_{DS} - \frac{1}{2}V_{DS}^2) \quad (3.1)$$

In order to achieve high linearity specifications the nonlinear term in Equation (3.1),

which is $\frac{1}{2}V_{DS}^2$, should be made as small as possible. This can only be achieved by using very wide transistors so that drain-to-source voltage drop across them is minimal under large AC current swing. A good rule of thumb is to use transistors at least 10 times wider than the width set by the linear current requirement of the transistor (I_{lin}). As an example, transistors as wide as 1.5 mm were used in this study. Using very large transistors, on the other hand, has a side-effect; significant parasitic capacitors.

This effect was neglected (by using small size devices) in the previous studies that have been examined [25], [40], since linearity had not been the main focus. And thus, poor linearity numbers were reported.

Since the attenuator developed in this research is designed to have very good linearity, its power handling should also be very high. Assuming a 4 dB back-off from the P_{1dB} compression point, an input power of 26 dBm could be applied to the input of the attenuator during normal operation. Power dissipated in the 16 dB attenuation stage for this level of input power is approximately 390 mW, most of which is dissipated by the attenuator resistors. In order not to exceed the maximum allowable temperature of resistors, the resistors should have very large surface areas governed by the formula as shown below (provided by the fab) and as explained earlier:

$$ThermalResistance = 1.1847 \times 10^5 \times (AREA)^{-0.72433} \quad (3.2)$$

As resistors would create similar parasitic effects to that of transistors, increasing their area would similarly result in increased parasitic. Due to inferior power handling specifications of the work reported in previous studies, parasitic effects caused by resistors were also neglected.

The main purpose of this work is to develop a phase coherent attenuator with comparable linearity and power handling specifications to that of a commercial digital step attenuator. Therefore, wide transistors and large resistors are employed in the design and all parasitic effects are addressed and taken care of. In order to design an attenuator with respectable linearity figures a state of the art RF process is chosen to decrease substrate parasitic and coupling effects (CSOI 7RF process from Global Foundries).

2) Inductive effects of bond wires;

Fabricated dies are usually encapsulated in packages (e.g., plastic, ceramic, etc.) for the end use. There are a large number of packages available for various applications. QFN (quad flat no leads) packages are generally used for high frequency applications since they offer small parasitic effects and an exposed ground paddle, which is crucial for high frequency applications. These packages offer a good grounding performance since their large metal paddle under them can be connected to the PCB (Printed Circuit Board) ground. An isometric view of a typical 4×4 QFN package along with the silicon die is shown in Figure 3.1.

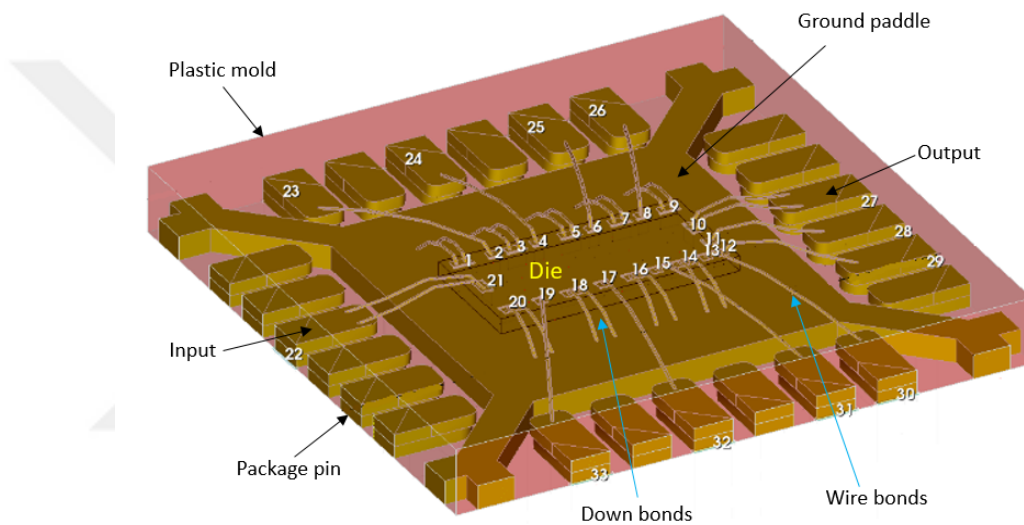


Figure 3.1: Structure of QFN package

The silicon die is connected to the package leads by thin wires, called wire bonds. These wire bonds (gold or aluminum) are very thin (typical 10 mils) and thus exhibit significant inductive effects. Referring back to Figure 3.1, wirebonds connecting the die to the input/output port and ground paddle of the package manifest themselves as inductors connected to the I/O and ground pads. It is known that, 10 mil diameter wirebond exhibits about 1 nH of inductance for each 1 mm of length. To minimize this effect, they should be as short as possible. To reduce inductance of the ground connections, wire bonds from the chip to the exposed paddle of the QFN package, called down bond, are used to connect ground pads on the die to the metal paddle. Moreover, to decrease the inductive effects of the wire bonds even more, double wires are used in addition to the short down bonds. However, even with these methods a

typical downbond would exhibit approximately 0.4 nH of inductance in this kind of package. This inductance value is significant especially for high frequency applications. Nevertheless, this important effect was ignored in all of the studies investigated [20], and [37]. To include this important effect, a new schematic of the attenuator with wire bonds that are modeled by inductors is given in Figure 3.2.

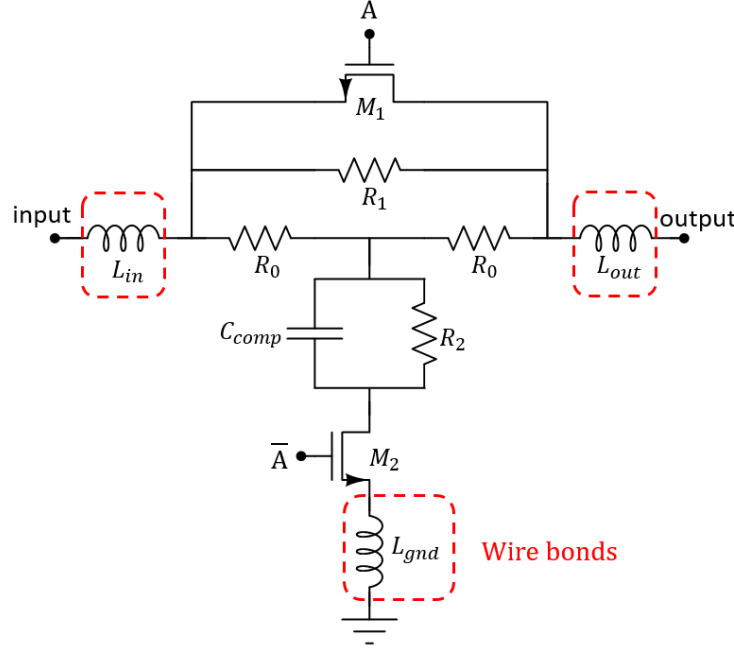


Figure 3.2: Schematic of attenuator including wire bonds

Inductors L_{in} and L_{out} in Figure 3.2 increase the insertion loss of the reference and attenuation states at the same time and with the same values and thus does not affect the relative attenuation or the phase shift. However, the downbond inductor L_{gnd} , which is in series with the shunt resistor R_2 , increases the impedance of the shunt path, decreasing the attenuation value with respect to frequency. Therefore, all attenuator stages should be tuned and compensated with L_{gnd} included in the schematic. The effect of wire bonds on the attenuation and the phase shift of the 8 dB stage which is attenuation compensated with L_{gnd} in the schematic can be seen in Figures 3.3 and 3.4 wire bonds, respectively.

The compensation is done by increasing the C_{comp} value with respect to the attenuator without the wire bonds effects. However, the phase shift of attenuator is further increased at the attenuation state creating a larger relative phase shift between the reference and the attenuation states.

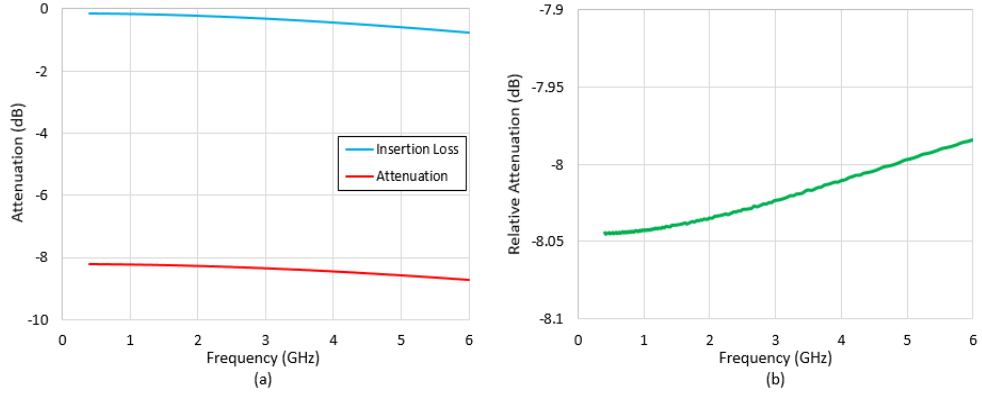


Figure 3.3: Simulation results of attenuator for 8 dB attenuation including wire bonds: (a) Attenuation and insertion loss. (b) Relative attenuation.

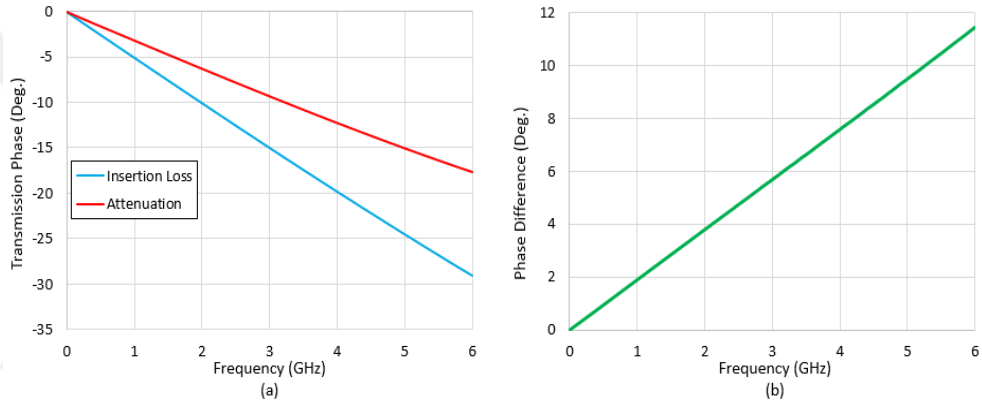


Figure 3.4: Simulation results of attenuator for 8 dB attenuation including wire bonds: (a) Transmission phase of reference and attenuation states. (b) Relative phase.

It should be noted that, the negligence of these wire bond effects are well justified for some of the previous studies [24], [25], and [34] where the authors used III-IV technologies that offer substrate vias. However, since substrate vias are not available for many technologies including the chosen SOI technology, the solutions presented in these studies for phase compensation are not suitable for many applications. To compensate these flaws, the effects of wire bonds, especially the down bond, which is connected to the ground paddle have been examined. With this study a universal solution to the phase shift problem of digital step attenuators is developed and presented without the need of specialized processing technologies, such as substrate vias. In the next sections, design of phase coherent circuit is discussed.

3.1 Theory Of Design and Mathematical Modeling

As mentioned in Chapter 2, as well as analyzed by previous workers [5], [34], insertion phase of the attenuation state is relatively lead-phase in comparison with reference state. This is due to the phase-shift introduced by high-pass filter (HPF) created by the attenuator in the attenuation state (Refer to Figure 2.6). In contrast, a low-pass filter (LPF) has a phase lag to the reference phase. From the research conducted by Ku (2010), it can be concluded that to reduce the transmission phase between two states (reference and attenuation states) of the attenuator, LPF structures can be added to the circuit at the attenuation state [34]. Therefore, by using the phase-lag characteristic of a low-pass filter, the transmission phase difference of a conventional switched Bridge-T attenuator can effectively be decreased. Figure 3.5 shows how inductive and capacitive LPF structures are used at the attenuator circuit.

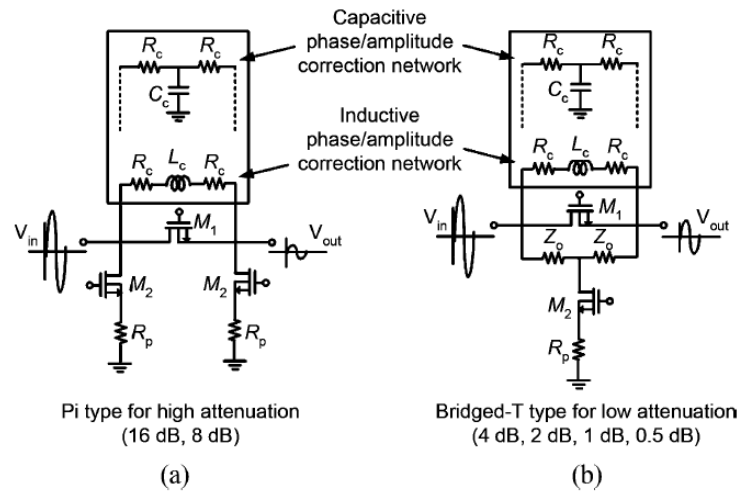


Figure 3.5: Switched: (a) Pi attenuator and (b) bridged-T attenuator [34].

This study ignores the effect of bondwires, which will have significant effect in the input/output pins, and as it is discussed in the previous section, more importantly in the ground downbond connections. O.V.Stukach et al. [38] designed a variable attenuator with p-i-n diodes, which has low phase shift. Figure 3.6 shows the schematic of the p-i-n diode variable attenuator. As reported, “in the attenuator without L_1 and L_2 the phase shift difference in same band reaches 57° . The inductances L_1 and L_2 provide phase stability, which combined with the diode’s parasitic capacities and compensated for the phase shift variation when the attenuation is adjusted”.

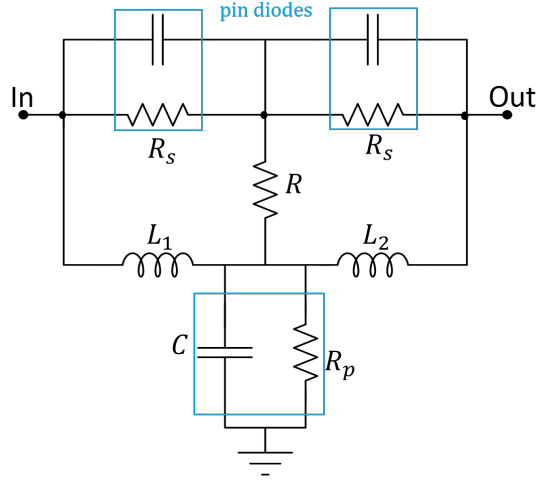


Figure 3.6: The schematic of the p-i-n diode attenuator.

In other words, Stukach also used inductive LPF (series inductor with attenuator) to reduce the phase shift of the attenuation state.

According to above mentioned studies, since capacitive LPF structure has high insertion loss and needs to be connected to ground (additional downbond is required) inductive LPF is preferred in this work. Inductances at series branches are used to compensate the phase lead of the attenuation state to the reference state.

All the other attenuation blocks will be influenced with the usage of a series inductance. In addition, insertion loss will be increased. To prevent these problems, series inductance should be used where it is required. Thus, switchable inductors that can be controlled to work in attenuation and reference states, are used between the individual attenuation blocks and the attenuator circuit.

In brief, attenuators are composed with switchable inductors, whose inductors resonate with the FETs capacitances (C_{off}) at frequencies much higher than operating frequency. By employing this configuration, it can be considered that not only amplitude but also phase compensation performances are improved [47].

The most important method for changing lead phase to lag is adding a shunt capacitance to the attenuator circuit. Because only attenuation state needs to shift phase from lead to lag state, it should be switchable similar to series inductances to work at the attenuation state.

It is concluded that using both series inductances and a shunt capacitance as phase compensation circuit can solve attenuation and phase shifting problem with acceptable error. Finally, amplitude and phase of the attenuator can be modified by tuning the inductors and the capacitor to obtain optimized values.

Several methods have been developed to minimize the amplitude and phase errors of the proposed attenuator. To achieve minimal phase shift and good amplitude accuracy at the same time, the circuit shown in Figure 3.7 is designed and implemented.

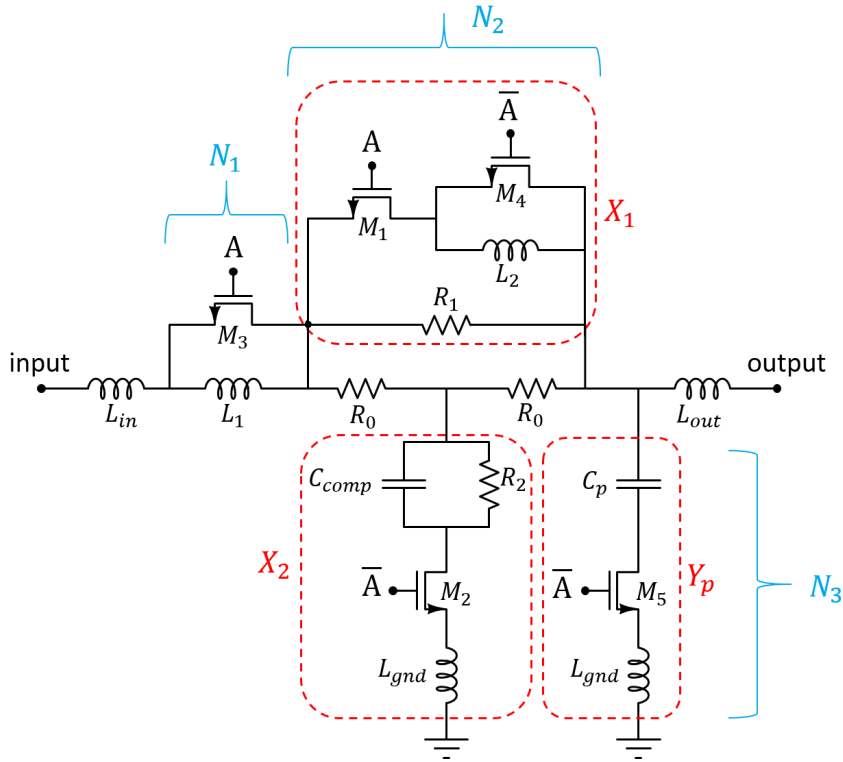


Figure 3.7: The schematic of phase coherent digital step attenuator.

This schematic is divided into different blocks and networks to ease the mathematical calculations. Figure 3.8 illustrates the schematic of designed phase coherent digital step attenuator in both reference and attenuation states.

To explore the condition of having no attenuation error and transmission phase difference, which is the condition of the ideal switched bridge-T attenuator, the transmission coefficient S_{21} is calculated from the transmission matrices (ABCD) at the reference and attenuation states.

The inductors L_1 and L_2 have been neglected for simplicity due to their small effects on reference and attenuation states, respectively.

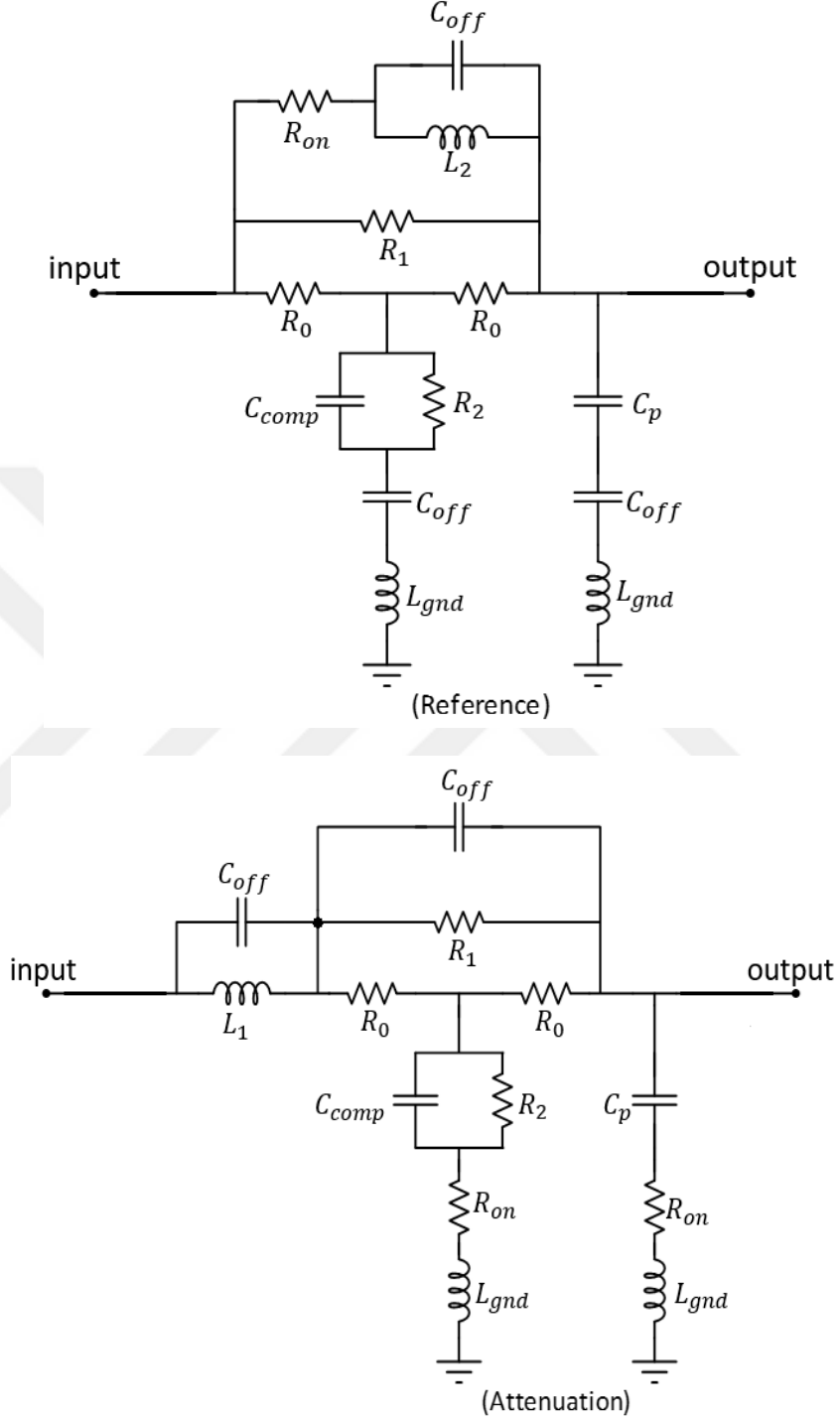


Figure 3.8: Schematic of the reference and attenuation states of Mathematical Model.

Impedance and admittance of different blocks (for both reference and attenuation states), which are used in the calculation of S_{21} , are calculated with some approxima-

tions within a single stage attenuator as:

$$X_{1-ref} = \frac{SL_2R_1 + R_{on}R_1}{SL_2 + R_{on} + R_1} \quad (3.3)$$

$$X_{1-att} = \frac{R_1}{SR_1C_{off} + 1} \quad (3.4)$$

$$X_{2-ref} = \frac{SR_2(C_{off} + C_{comp}) + 1}{S^2R_2C_{comp}C_{off} + SC_{off}} \quad (3.5)$$

$$X_{2-att} = \frac{S^2R_2C_{comp}L_{gnd} + S(L_{gnd} + R_{on}R_2C_{comp}) + (R_{on} + R_2)}{SR_2C_{comp} + 1} \quad (3.6)$$

$$Y_{P-ref} = \frac{SC_pC_{off}}{S^2L_{gnd}C_pC_{off} + C_p + C_{off}} \quad (3.7)$$

$$Y_{P-att} = \frac{SC_p}{S^2L_{gnd}C_p + SC_pR_{on} + 1} \quad (3.8)$$

Since, compensation elements are related to each other, determination of the initial values for them needs complex mathematical operations. Therefore, use of hand calculation is time consuming and can lead to errors. To overcome these difficulties, according to design theory that has been explained and using above equations, an automated MATLAB algorithm with graphical user interface (GUI) is developed for the calculation of the initial values. As it can be seen in the Figure 3.9, this program uses the initial values for compensation circuit elements as the input. Then, it calculates and plots the attenuation and phase of designed circuit as the output for both reference and attenuation states. In addition, it plots the relative attenuation and transmission phase difference. The code of the algorithm is presented in Appendix A.

At the input panel, values of R_{on} and C_{off} are known and are related to the process technology. Moreover, the values of R_1 and R_2 can be calculated for specific attenuation (e.g., the values of Figure 3.9 are for 8 dB attenuation).

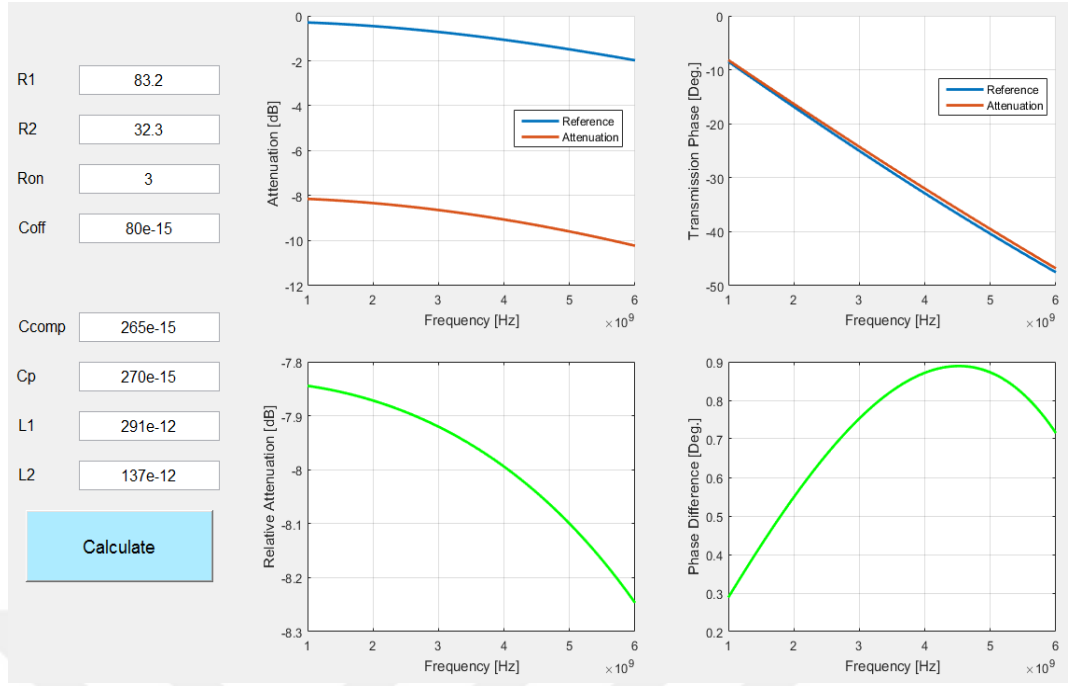


Figure 3.9: Input panel and output plots for MATLAB program.

These values can help to calculate the initial values with minimum iteration. Calculated values are for ideal elements in a single attenuation block and they should be optimized for technology elements in one attenuation block when it is cascaded with others.

3.2 Implementation in Cadence

This section provides the optimization studies of the phase coherent DSA using circuit analysis tools(Virtuoso Design Environment provided by Cadence Design Systems). Initial values for the optimization were gathered using the mathematical tool as described in the previous section. Figure 3.10, redraws the designed phase coherent attenuator block.

Development steps can be summarized as follows:

- 1) Typically, the attenuation of conventional bridge-T attenuator decreases as the frequency of the signal applied to the attenuator increases, producing an upward slope in the frequency response that is attributed to impedance of the L_{gnd} and C_{off} of

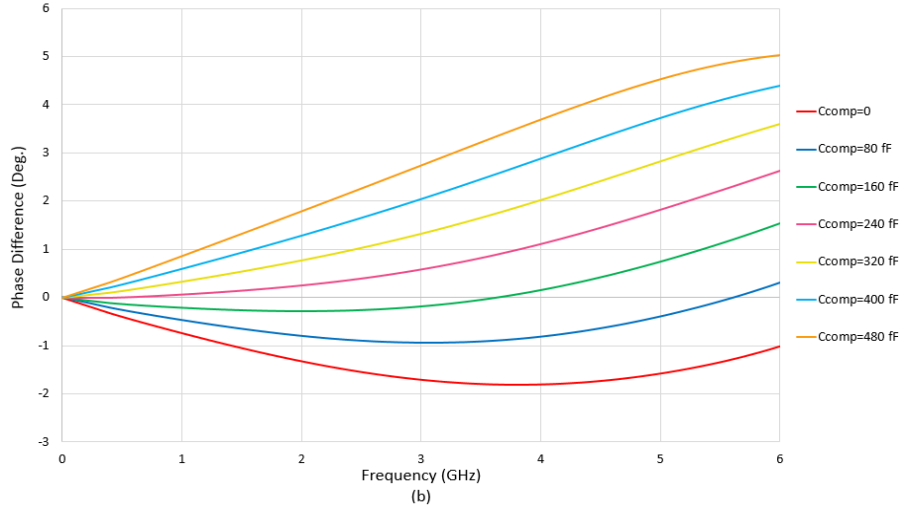
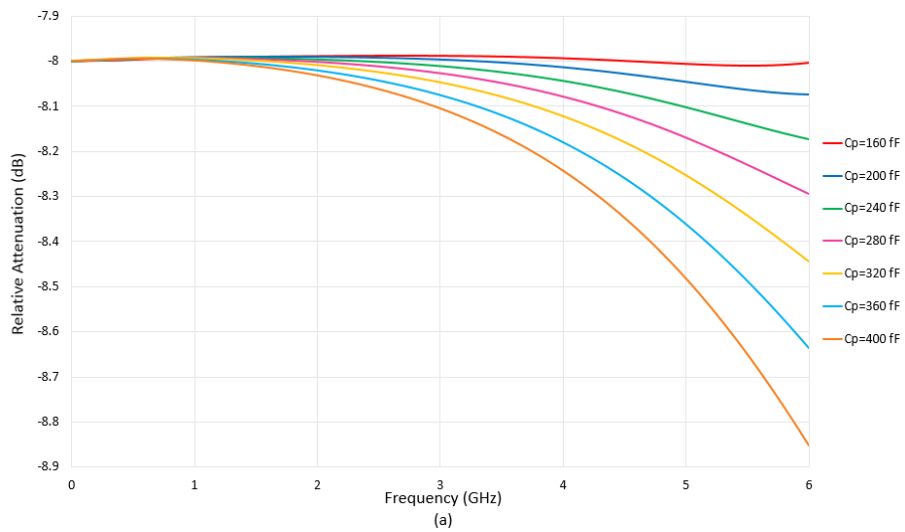


Figure 3.11: Simulation results of proposed Bridge-T attenuator for 8 dB attenuation with phase compensation circuit: (a) Relative attenuation and (b) phase difference with various C_{comp} .

As can be seen from the plots, careful optimization of the compensation capacitance, C_{comp} , along with the other design variables already optimized (as will be shown later) leads to good attenuation accuracy while achieving low phase shift.

2) Attenuator block has phase-lead in the attenuation state with respect to the reference state. This difference can be corrected by connecting a switchable shunt capacitor to the circuit. This method is applied to the circuit by adding block X_2 . The effect of C_p for various capacitance values while other circuit blocks are kept at their optimized values can be seen in the Figure 3.12.



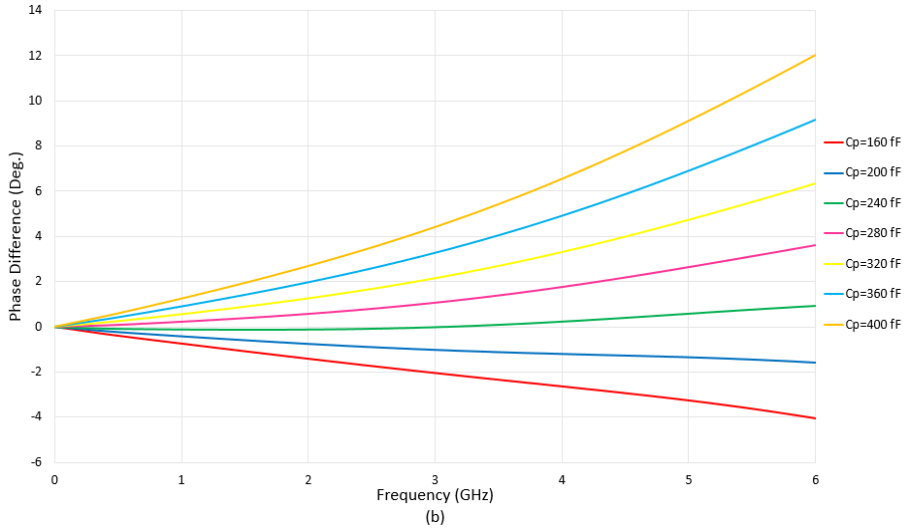


Figure 3.12: Simulation results of proposed Bridge-T attenuator for 8 dB attenuation with phase compensation circuit: (a)Relative attenuation and (b) phase difference with various C_p .

This capacitance slightly increases the attenuation error, because parasitics of shunt capacitance C_p have effect on reference state. However, its effect at attenuation state is dominant. According to the results, this switchable shunt capacitance has excellent effect on phase shifting and compensated the difference between the REF and ON states.

3) Finally, switchable L_1 and L_2 inductors have been added in series to the attenuator circuit (X_3 and X_4 blocks). They have effect on the attenuation and the phase shift of attenuation and reference states, respectively. Thus, the attenuation error due to adding C_p has been corrected, while the corrected phase shift is unharmed (They compensate phase error of each other). The effect of L_1 is illustrated in Figure 3.13.

As expected, simulation results show that L_1 acts similar to C_p . Increasing the value of L_1 , increases the phase shift of the attenuator in attenuation state. Additionally, because of L_1 the attenuation value increases with respect to the frequency since it is in series with the attenuator.

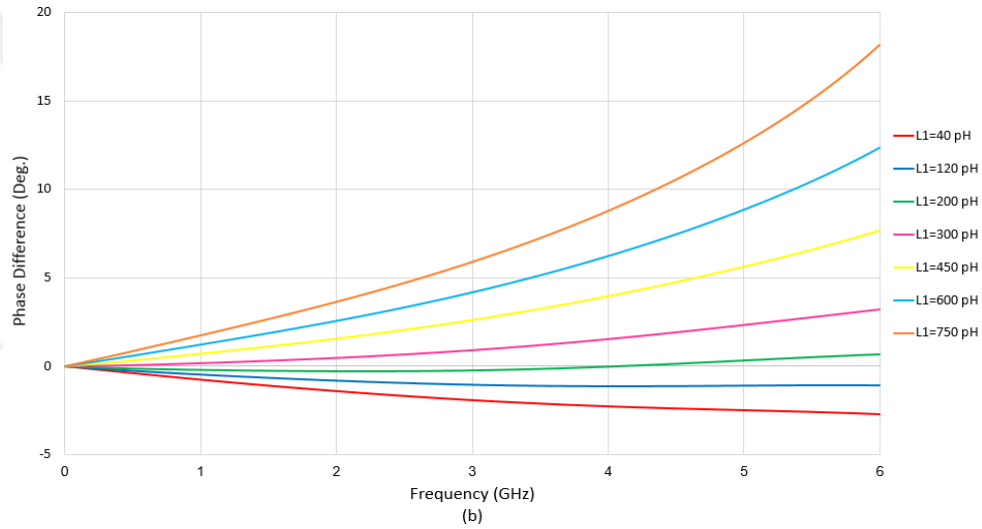
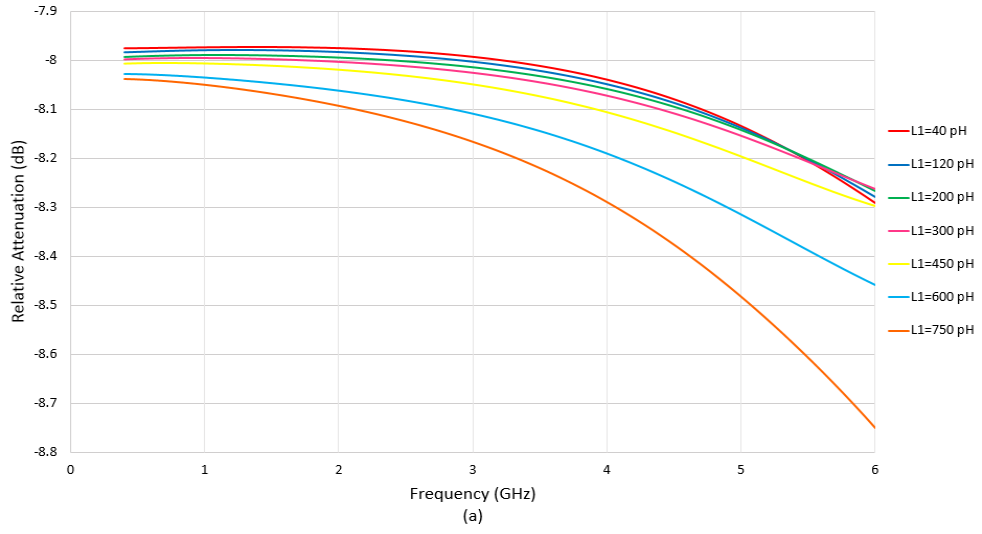


Figure 3.13: Simulation results of proposed Bridge-T attenuator for 8 dB attenuation with phase compensation circuit: (a)Relative attenuation and (b) phase difference with various L_1 .

Since, both C_p and L_1 work together in the attenuation state, the attenuation error will be increased. To solve this problem, switchable inductor, L_2 is added to the circuit which works at reference state and leads to an increase in the insertion loss state with respect to frequency. In other words, it is used to adjust the flatness of the attenuation with frequency without contributing in the phase shift. The effect of L_2 is illustrated in Figure 3.14.

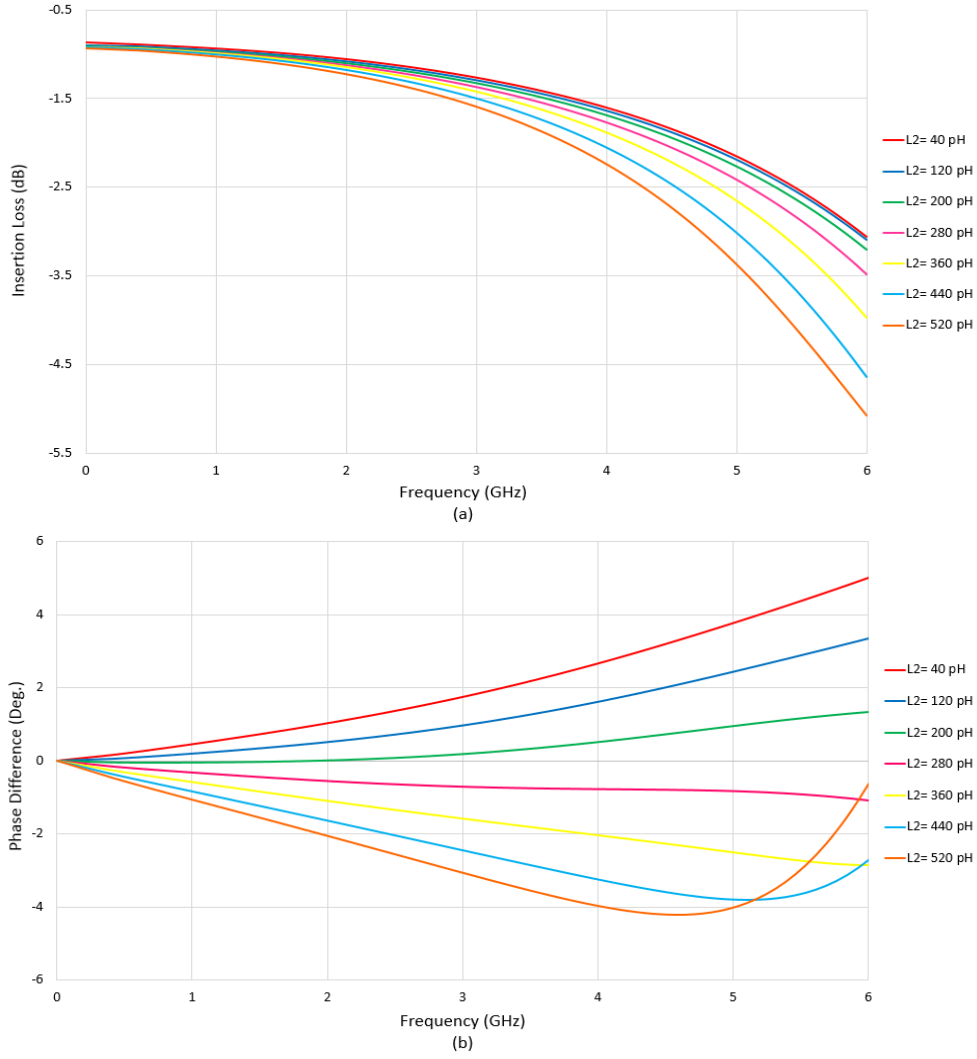


Figure 3.14: Simulation results of proposed Bridge-T attenuator for 8 dB attenuation with phase compensation circuit: (a)Relative attenuation and (b) phase difference with various L_2 .

As expected, physical concept of the circuit is phase lag, which is caused by the series branch's inductive reactance. Implementation of series inductances decrease phase difference between the reference (OFF) and attenuation (ON) states.

The elements of the phase coherent circuit are optimized for each attenuation block. Simulation results of the optimized attenuator circuit for 8 dB attenuation block can be seen in Figures 3.15 and 3.16.

As can be seen from Figure 3.15 and 3.16, attenuator has a finite attenuation error and a phase shift. However, the phase shift is much more improved when compared

to the other studies and commercial products, without sacrificing too much on the attenuation accuracy.

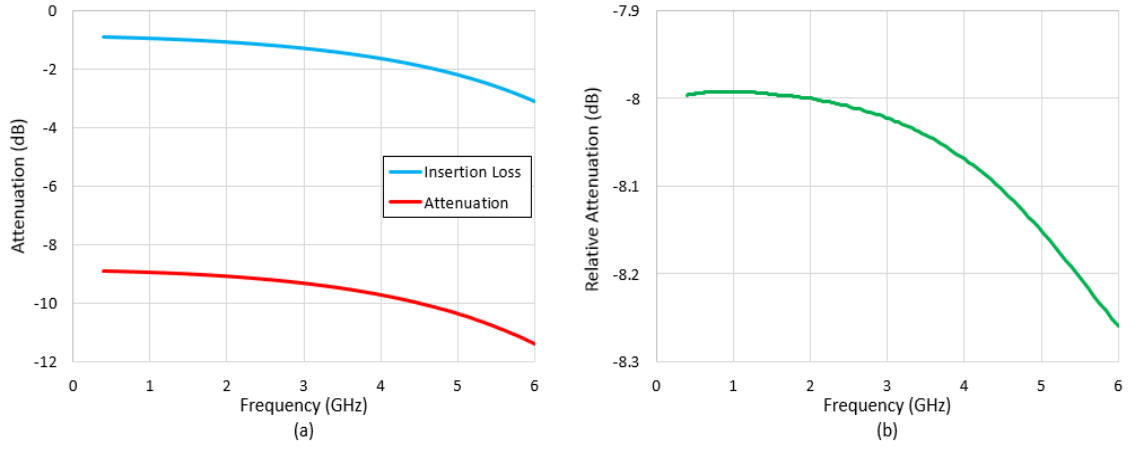


Figure 3.15: Simulation results of optimized 8 dB attenuation block: (a) Attenuation and insertion loss. (b) Relative attenuation.

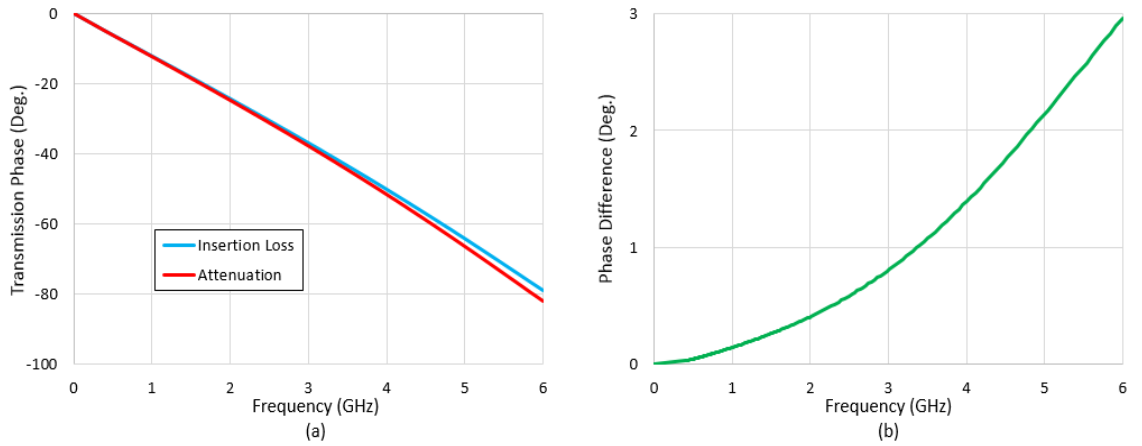


Figure 3.16: Simulation results of optimized 8 dB attenuation block: (a) Transmission phase of reference and attenuation states. (b) Relative phase.

Figure 3.17 illustrates the attenuation of the proposed phase coherent attenuator compared with the conventional attenuator. As can be seen from the plot the insertion loss of the proposed attenuator is increased by about 1 dB at 4 GHz.

Figure 3.18 compares the phase shift of the conventional attenuator with the proposed attenuator. As can be seen from the plot, the relative phase shift of the phase coherent attenuator is smaller than 5 degrees, whereas the conventional attenuator exhibits a phase shift of larger than 30 degrees at 4 GHz.

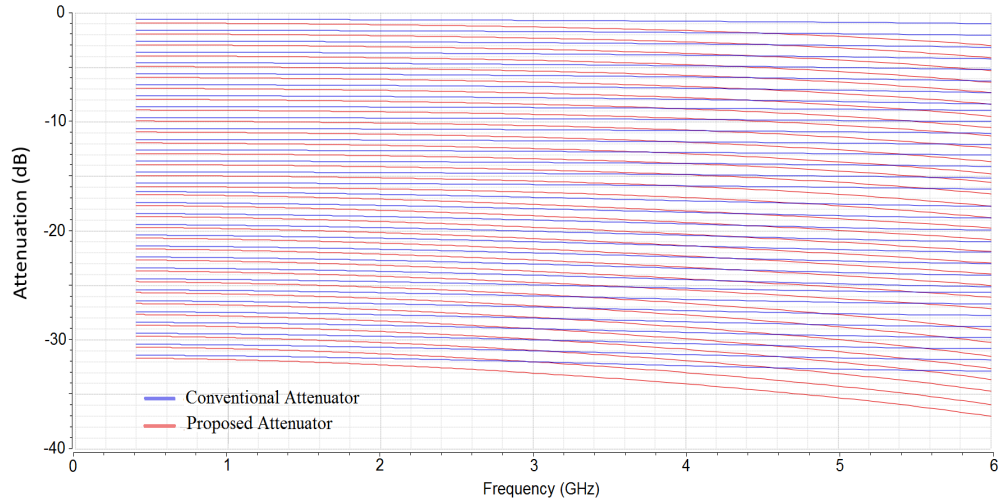


Figure 3.17: Attenuation characteristic of the designed phase coherent attenuator compared with the conventional attenuator.

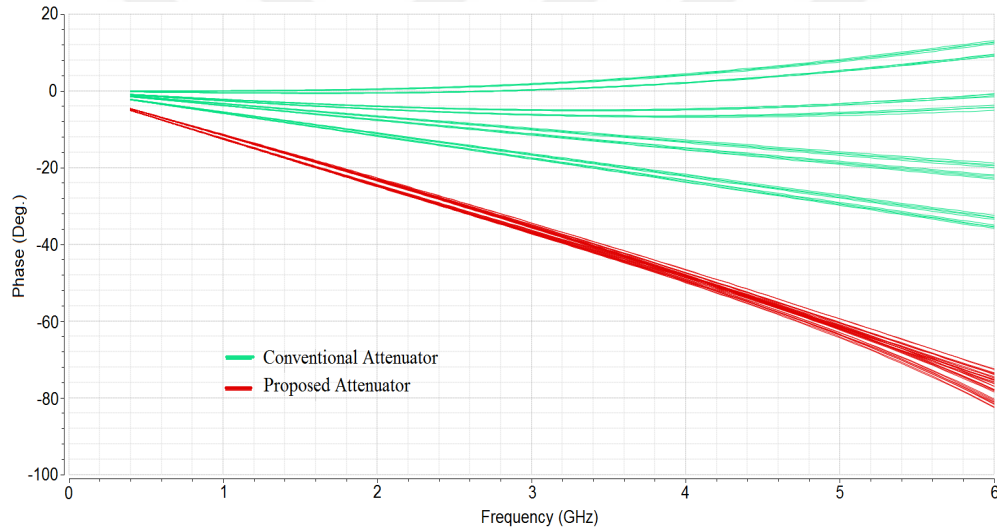


Figure 3.18: Transmission phase characteristic of the designed phase coherent attenuator compared with the conventional attenuator.

As mentioned previously, the produced integrated circuits is encapsulated in a commercial QFN package. Thus, to see the package and wire bonds effects on the designed circuit and to do further optimization, 3D EM simulations are performed using AWR Analyst package program. An isometric view of system is given in Figure 3.19.

It should be noted that, 3D EM simulation results for the package model, which is provided by AWR is consistent up to 3 GHz. Therefore, optimization is performed

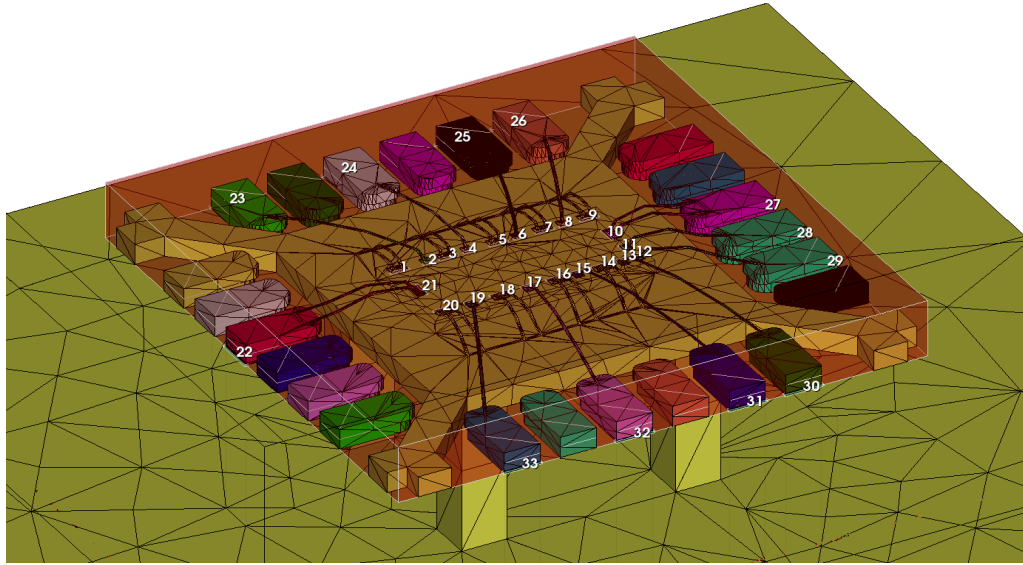


Figure 3.19: Figure of modeled chip, inside the QFN package by AWR Analyst.

using 3D EM simulation results up to 3 GHz. However, approximate inductive values of wire bonds are used for higher frequencies. The rule of thumb is approximately 800 pH for input/output wire bonds and 400 pH for down bonds. According to 3D simulations that can be seen in Figure 3.20, the inductive values of down bonds are approximately 420 pH at the operating frequency band. This value is very close to the rule of thumb (400 pH) that is used for optimization of the attenuator for frequencies higher than 3 GHz. 3D EM simulation results of all the attenuation states are shown in Figure 3.21.

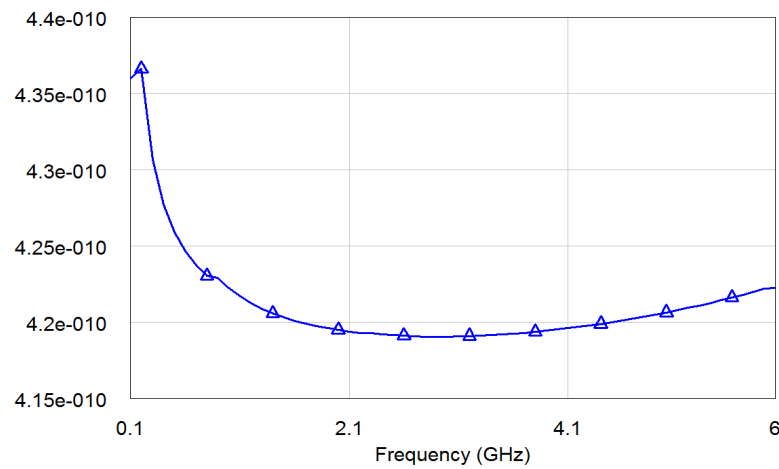


Figure 3.20: Wire bonds inductive values vs. frequency.

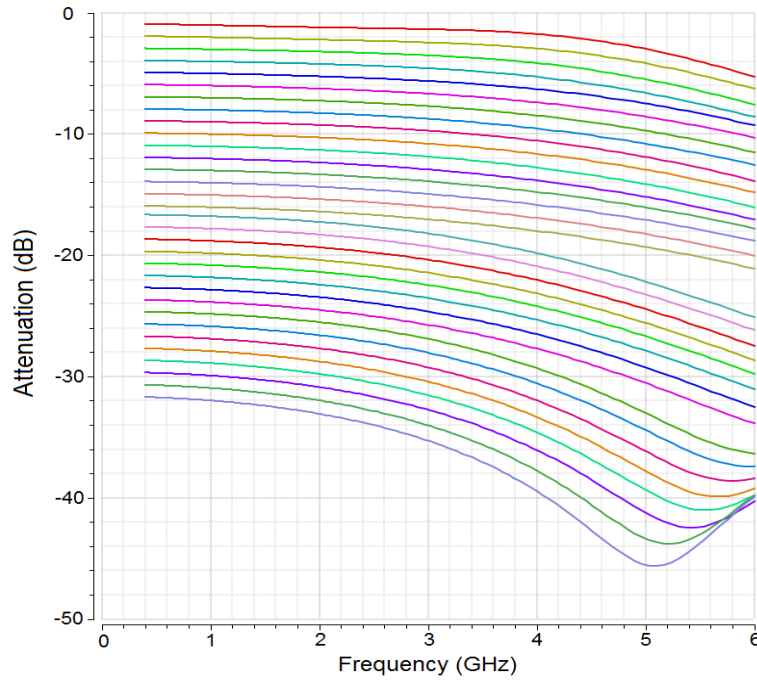


Figure 3.21: Attenuation results of the 3D EM simulation.

Step and state errors are important specifications of attenuators. Figure 3.22 shows the step error of the 6-bit attenuator at different frequencies. The plot shows the error between each consecutive state with respect to the ideal difference over the attenuation setting.

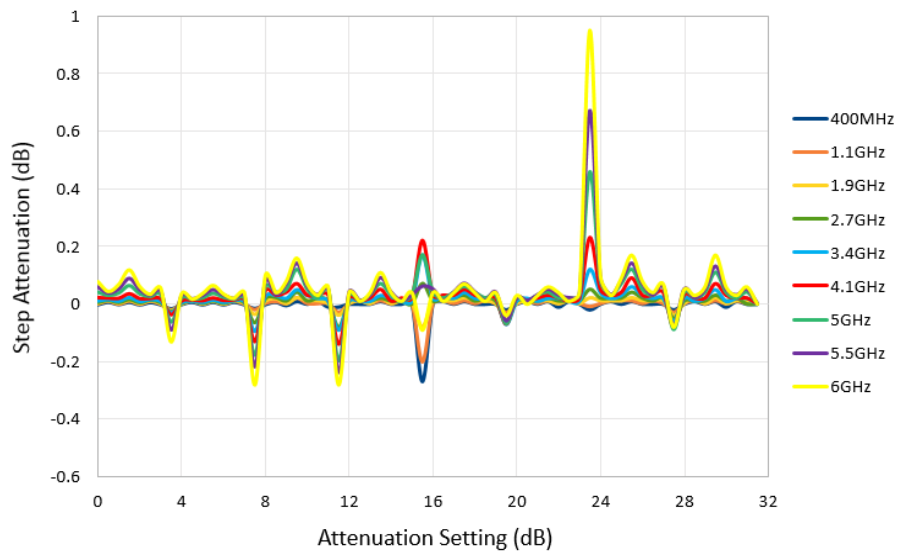


Figure 3.22: 0.5 dB Step Attenuation vs. Attenuation Setting.

As can be seen the attenuator can provide attenuation steps with an error less than 0.5 dB for frequencies smaller than 5 GHz.

Figure 3.23 shows the state error of the same attenuator. The plot shows the error of each state referenced to the insertion loss state.

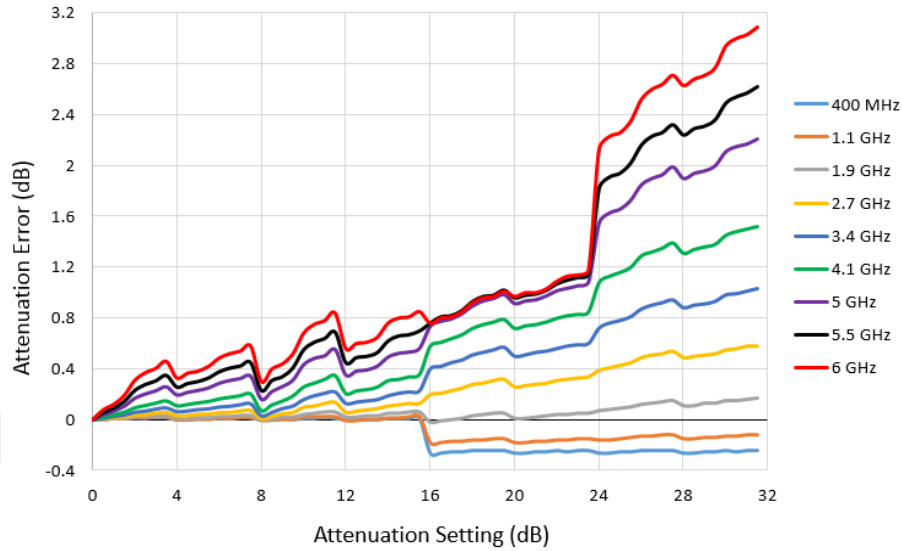


Figure 3.23: Attenuation Error vs. Attenuation Setting.

To assess the effect of temperature variation on the digital step attenuator performance, the circuit is simulated from -40°C to 85°C . Figure 3.24 shows the insertion loss variation of the attenuator.

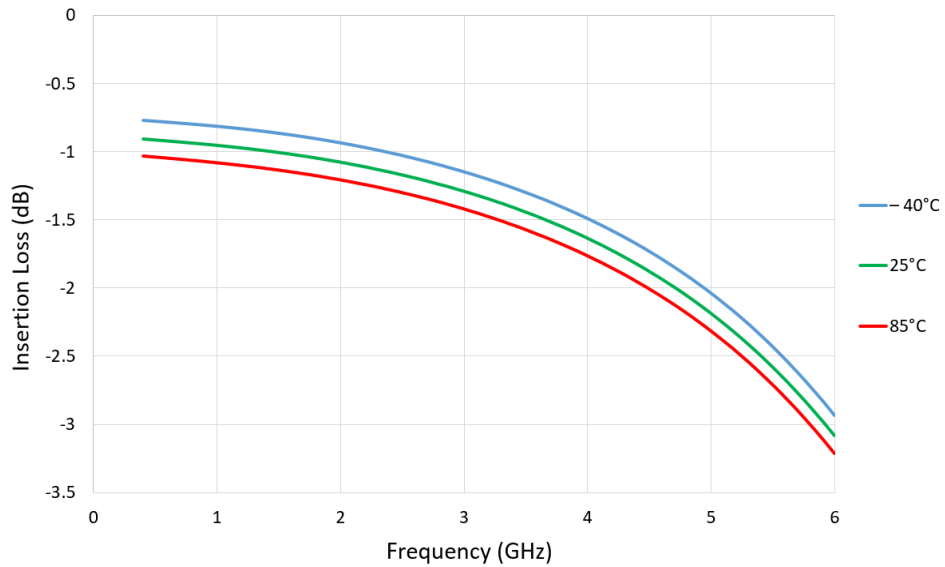


Figure 3.24: Insertion Loss vs. Frequency.

As can be seen from the figure, insertion loss increases with respect to temperature. This is expected since the channel resistance of switch transistors increases with the temperature.

Figure 3.25 and Figure 3.26 shows the state and step errors of the attenuator with respect to temperature, respectively. The input frequency is kept at 4 GHz for this simulation sweep. As can be seen from the figures the attenuator keeps its monotonicity over changing temperature experiencing less than 0.5 dB variation in the state error.

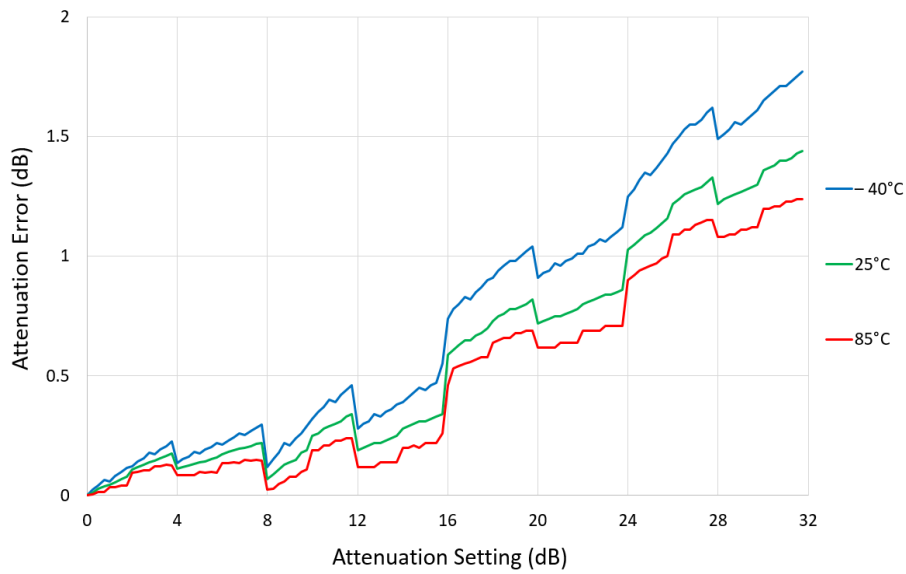


Figure 3.25: Attenuation Error vs. Attenuation Setting.

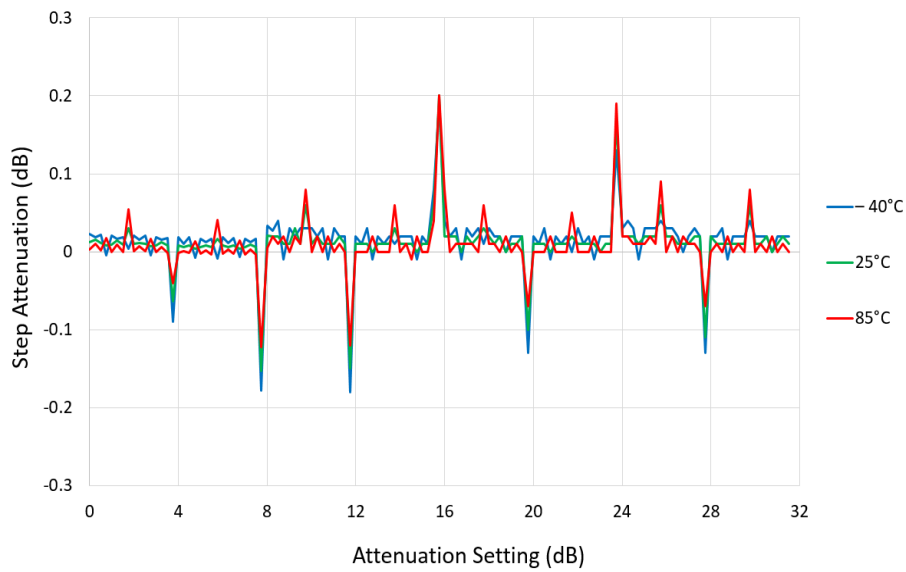


Figure 3.26: Step attenuation vs. Attenuation Setting.

This result is expected since the attenuation value of each block is mainly set by the resistor ratios which change at exactly the same amount with changing temperature.

Table 3.1 indicates the summary of the optimized values for the compensation elements of each attenuation block.

Table 3.1: Optimized components values for designed 6 bits of phase coherent attenuator.

	8 dB (16/2)	8 dB (16/1)	8 dB	4 dB	2 dB	1 dB	0.5 dB
C_{comp} (fF)	800	785	265	140	95	20	17
C_p (fF)	350	350	270	-	-	-	-
L_1 (pH)	600		290	150	-	-	-
L_2 (pH)	425	308	137	-	-	-	-

Figure 3.27 shows the layout of the designed integrated circuit. The total chip size, including pads, is $950\mu\text{m} \times 1950\mu\text{m}$. This size is selected specifically to fit the chip inside a QFN 4×4 mm package.

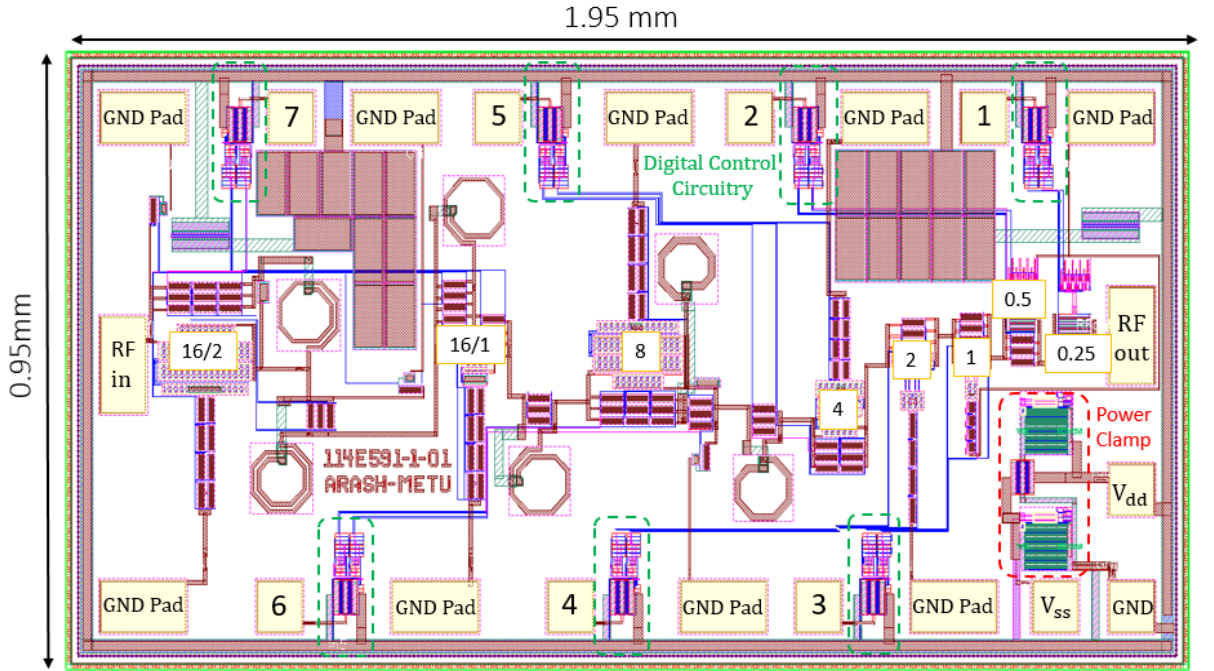


Figure 3.27: Layout of designed phase coherent attenuator.

ESD protection circuits are used at all digital inputs. Expected classification of the circuit is Class-2. In addition, power clamp (RC-triggered) ESD protection are used between the V_{dd} , V_{ss} and GND lines.

It is important to note that the added compensation circuitry comes with an area penalty as can be seen from the layout. However, an uncompensated attenuator would also have all its attenuation blocks separated from each other with some distance to reduce RF and thermal coupling between stages. Therefore, real area penalty due to the added compensation circuitry is minimal.

3.3 Gate and Body Floating Technique and Control Voltages

As detailed in the previous sections, nFET devices are used as switches to control the attenuation blocks. In order for transistors to approximate ideal switches they should remain at their set states (i.e., ON or OFF) regardless of the voltage swing applied to their inputs. As an example, assume a transistor whose gate voltage is taken to 2.5 V, and its body to 0 V to approximate a switch in its ON position as shown in Figure 3.28.

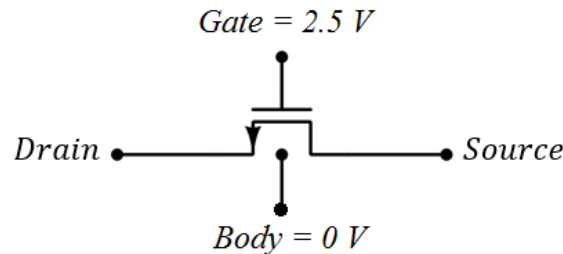


Figure 3.28: Structure of a switch for control the attenuation blocks.

If the gate and body is kept constant at their respective DC voltages with AC grounds (e.g., gate and body is connected to DC sources, or buffers directly), any AC voltage applied to the source of the transistor will modulate the gate to source voltage of the transistor. This, in turn will modulate the ON resistance of the transistor (please refer to equation (3.1)) and will degrade the linearity of the switch. At the far extreme, if the voltage swing is large enough (e.g., larger than 2.5 V), the transistor will turn off when the gate to drain voltage is smaller than the threshold of the device.

Similar analysis can be applied to a transistor in the OFF case as well, for which the transistor can turn on under enough AC swing. Moreover, even larger AC swing will create too much voltage swing across the transistor terminals, resulting a permanent damage to the transistor. Same discussion is true for the body terminal as well. For an nFET device to work properly, body should be kept at a lower voltage than the source and the drain terminals so that the junction diodes (source-body and drain-body) would not turn on.

In order to address this problem, switches used in this design make use of the gate and body floating technique. A large floating resistor is connected to the gate and body terminals creating a high-pass filter looking in from the input of the switch as shown in Figure 3.29.

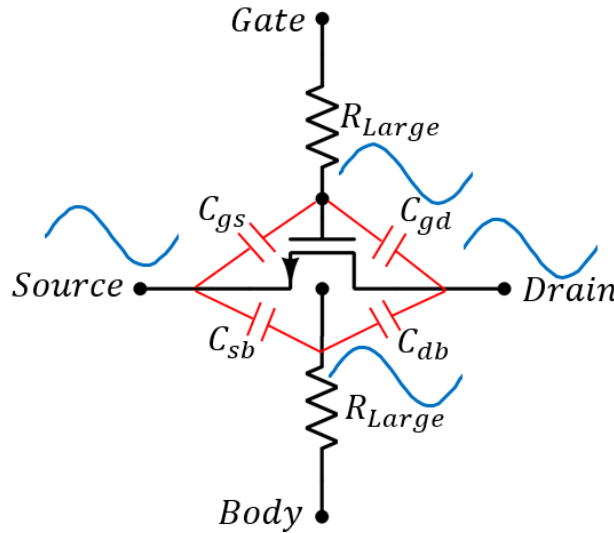


Figure 3.29: Structure of a switch including gate-body resistor.

With correct selection of the resistance value of R_{large} , the high-pass filter created by this resistor and the source to gate capacitance (C_{gs}) (and source to body capacitance (C_{sb})) successfully replicates the AC swing applied to the input. By this way, the DC operating point of transistors are not modified under AC input. Another advantage of using this technique is the improvement of the bandwidth of the transistors. Since gate and body terminals are following the input AC swing, C_{gs} and C_{sb} capacitors are bootstrapped and are effectively nullified.

The downside of this technique is the slow transient response of the switching ele-

ments. Looking in from the control side, $R_{large}-C_{gs}$ duo (similarly, $R_{large}-C_{gb}$ duo) presents a low pass filter response. Therefore, the digital signal applied to the control terminal will be slow to propagate to the gate (and body) terminal of devices. Thus, there is a trade-off between the lowest operational frequency of such circuits (e.g., DSAs) and the switching speed of them. Thus, one should select a minimum gate (and body) floating resistance value depending on the minimum operating frequency of the system. For this design, the minimum operating frequency of the DSA is chosen to be 400 MHz.

The analysis shown in Figure 3.30, shows the ratio of the source voltage swing to the drain voltage swing under varying R_{large} resistor values at 100 MHz (4X is chosen as a safety factor).

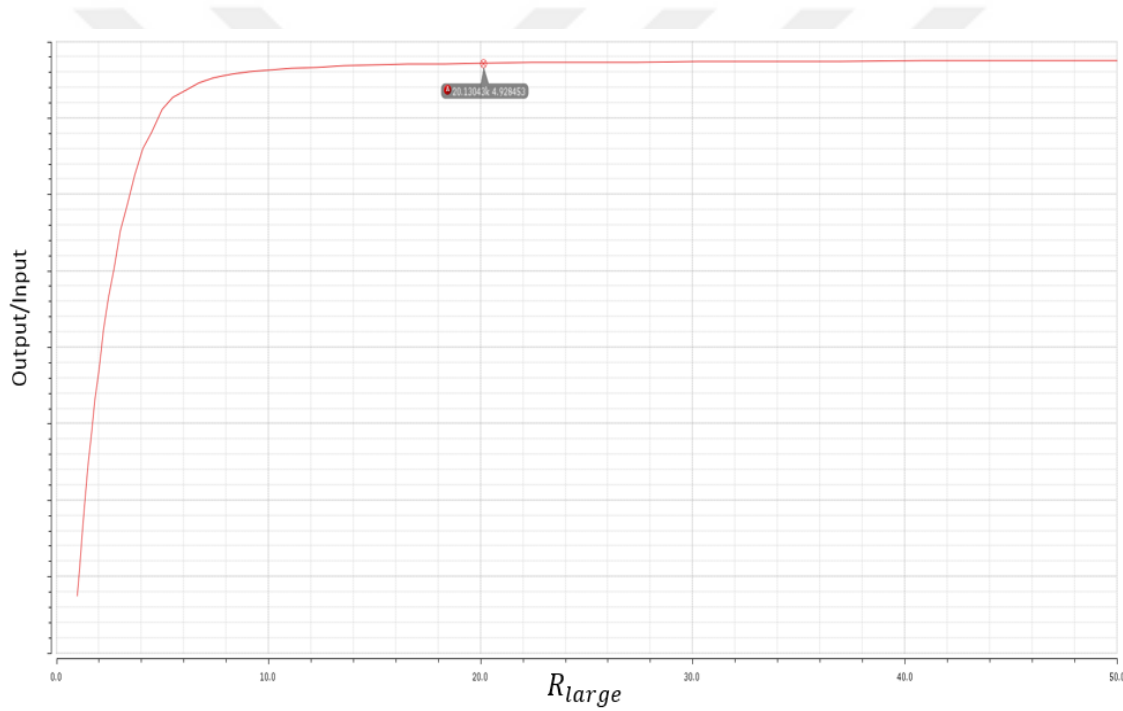


Figure 3.30: Transient simulation results for different R_{Large} values .

As can be seen from the plot, the ratio is almost constant after 20 kΩ, suggesting any resistor value larger than 20 kΩ would effectively float the transistor. The value of R_{large} will change depending on the value of C_{gs} and thus with the width of the transistor.

Table 3.2 gives a summary of the R_{large} values and the width of the transistors used in this design.

Table 3.2: Exact values for R_{Large} resistor at different attenuation blocks respect to width of the transistors.

		8 dB (16/2)	8 dB (16/1)	8 dB	4 dB	2 dB	1 dB	0.5 dB
M_1	width (mm)	1.5	1.5	1.5	1.5	1.5	1.5	1.5
	R_{Large} (K Ω)	100	100	100	100	100	100	100
M_2	width (mm)	1	1	1	0.7	0.5	0.2	0.05
	R_{Large} (K Ω)	150	150	150	200	300	700	1000

As previously discussed, for the realization of this project 0.18 μm SOI CMOS technology is selected which is offered by Global Foundries. In this technology, each switch has its own silicon oxide area. Therefore, body connection of each switch can be floated individually, which is otherwise impossible to do in a standard CMOS process. Although floating gate and body terminal is adequate enough to achieve respectable linearity another technique is employed to achieve even better linearity numbers; applying varying voltages to gate and body. In this technique the following is done:

- 1) ON case: 2.5 V and 0 V is applied to the gate and body of transistors, respectively. By this way maximum rated voltage of transistors is not exceeded (i.e., 2.5 V).
- 2) OFF case: -2.5 V is applied to both gate and body of transistors. By this way, the gate and body terminals are kept 2.5 V lower than the DC value of the RF input (0 V) providing a good headroom for transistors to stay OFF.

As a summary, the gate voltages of transistors will vary between 2.5 V and -2.5 V, while the body terminal will vary between 0 V and -2.5 V in ON and OFF cases, respectively. Since the chosen technology offers only 2.5 V PMOS and NMOS transistors, special digital buffers should be implemented to provide the required 5 V dynamic range. The following section provides the design details of the digital control circuitry.

3.4 Digital Control Circuitry

In order to make the circuit easy to control the digital control inputs to the circuit is chosen to be TTL compatible (varying between 0 and 2.5 V).

The digital control circuit that is used to convert the incoming 0 V and 2.5 V digital inputs to the required voltages at the gate and body of transistors is given in Figure 3.31 [44]. The required negative voltage (V_{ss}) is supplied to the chip externally.

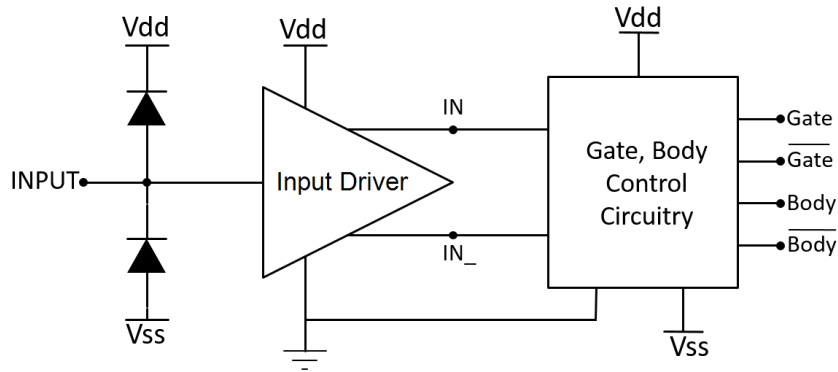


Figure 3.31: Schematic diagram of digital control circuitry.

It can be seen that the digital control circuitry includes two ESD protection diodes, the input driver and the gate-body control circuit. The input driver of Figure 3.31, which is created from two cascaded inverters (as shown in Figure 3.32) acts as buffer and also creates the complement of the input signal.

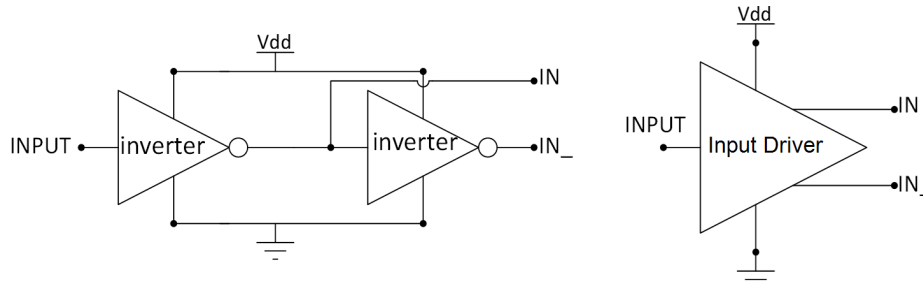


Figure 3.32: Schematic diagram and symbol of the input driver.

These signal are used as the input of the gate-body control circuit, which is the most important part of the digital control circuitry. Figure 3.33 shows the gate-body control circuit.

As mentioned before, the gate voltages of switch transistors vary between 2.5 V (V_{dd}) and -2.5 V ($-V_{ss}$), while the body terminal vary between 0 V and -2.5 V in ON and OFF cases, respectively. To create these control voltages, the gate-body control circuit is implemented. This circuit is used to convert TTL compatible digital input signal to a logic level with 5 V dynamic range with 2.5 V devices.

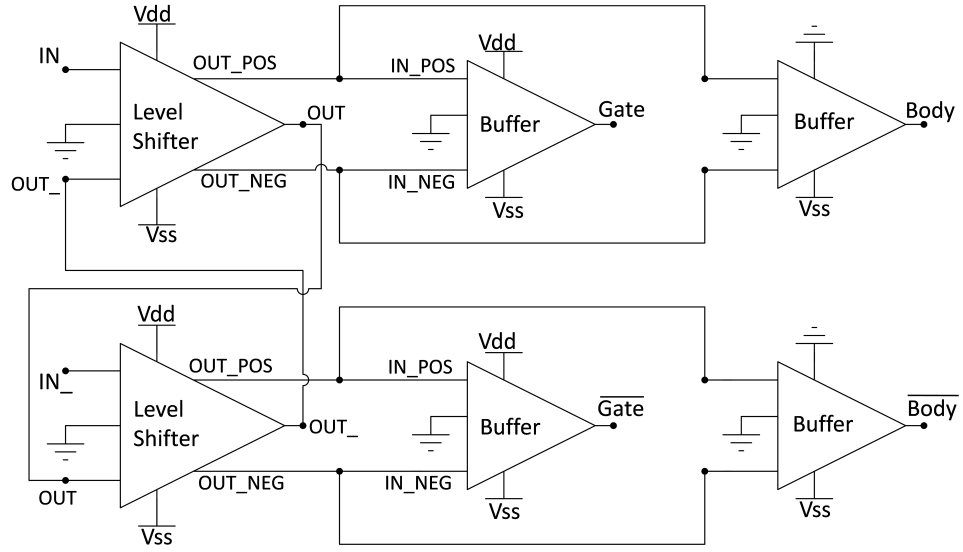


Figure 3.33: Schematic diagram of control circuit.

When the digital input signal on the INPUT node of the input driver reaches a logical high state (V_{dd}), the IN signal and the IN_- signal go to GND and V_{dd} , respectively. The OUT signal is driven to V_{dd} . At the same time, the OUT_- signal is driven towards $-V_{ss}$. The level shifter works similarly when the INPUT signal transitions from a logic high state to a logic low state. When the digital input signal on the INPUT node of the input driver reaches a logical low state (GND), the IN signal and the IN_- signal go to V_{dd} and GND , respectively. The OUT signal is driven to $-V_{ss}$. At the same time, the OUT_- signal is driven towards V_{dd} . These outputs are limited by using a LCVD to increase the reliability of the transistors in the level shifter. The reduced voltage swing does not alter the function of the level shifter. This control circuit includes a large number of inverters coupled in a feedback configuration. The feedback provides the quick change in state of the level shifter. Moreover, the grounding contribution provides that the level shifter inverters never see more than a V_{dd} voltage drop across the source/drain nodes of the transistors of the inverters.

RF coupling will occur if level shifter output are directly used to control the RF switch transistors. Thus, RF buffers are used to prevent the undesired RF coupling to the digital circuitry. They act as driver stages to inhibit dropping of the control voltages (gate, body and their complementary). Larger dimension transistors are used in the RF buffer circuit to achieve an effective AC ground at the control signals.

The schematic of the level shifter and buffer circuits, which are used in the control circuit shown in Figures 3.34 and 3.35, respectively.

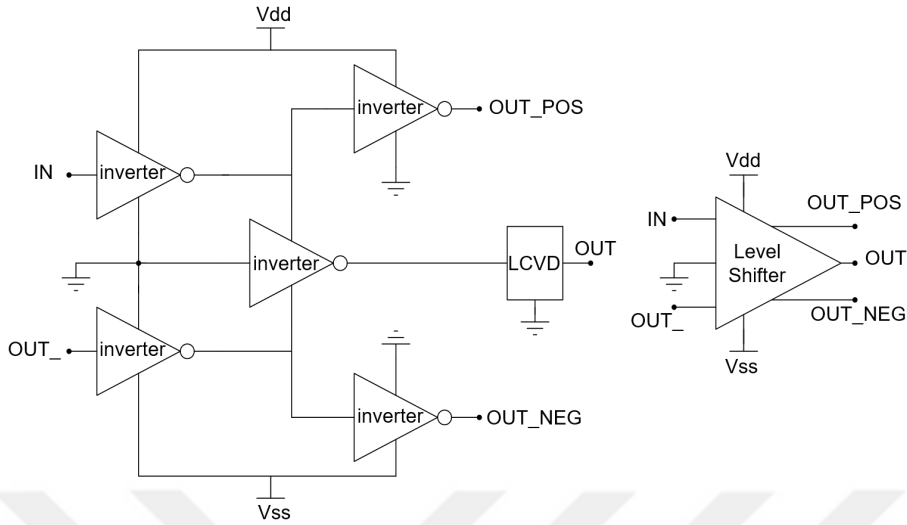


Figure 3.34: Schematic of level shifter circuit.

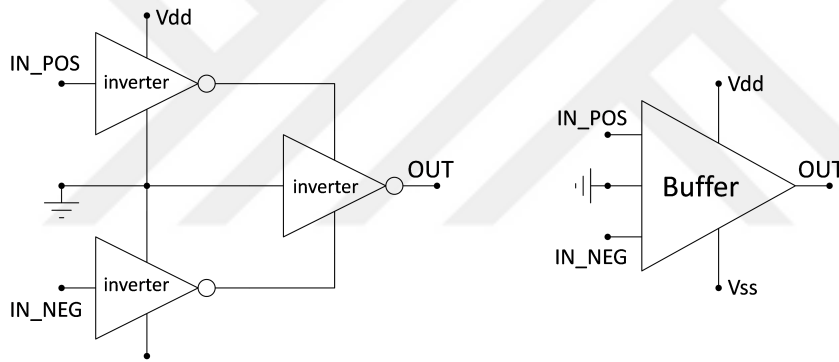


Figure 3.35: Schematic of buffer circuit.

As can be seen from the Figure 3.34, the outputs of the input inverters are tapped and provided as input to two output inverters to generate OUT_POS and OUT_NEG output signals ranges from GND to V_{dd} and from $-V_{ss}$ to GND , respectively (level shifter output signals never exceed V_{dd} or $-V_{ss}$). The output signals are used as input signals for RF buffers to produce required control voltages for the gate and the body of the switch transistors in addition to address potential gate oxide reliability problems in the RF buffer circuits. Referring to Figure 3.35, the RF buffer input inverters are used to control the power supply voltages of the RF buffer output inverter.

To prevent stability problems of transistors in the level shifter circuit due to feedback, a low current voltage divider (LCVD) is used at the output of level shifters in the

feedback path. LCVD is used to address potential gate oxide reliability issues related to excessive voltage swings across the gate oxide of the feedback inverter transistors. This circuit limits the voltages applied to the gates of the level shifter's feedback inverters and increases the reliability of the transistors in both of the level shifters. Figure 3.36 illustrates the transistor level schematic of LCVD.

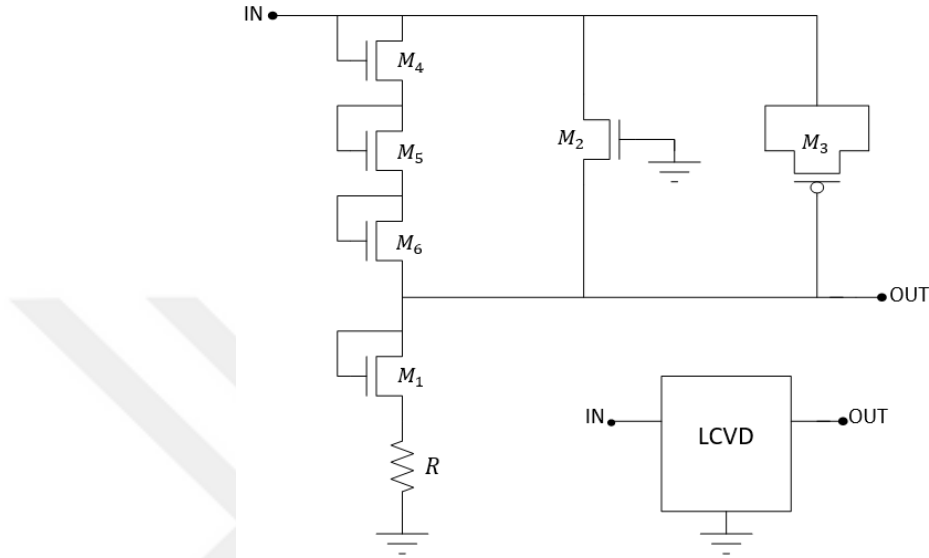


Figure 3.36: Schematic of low current voltage divider circuit.

In this circuit, M_1 reversed biased during negative voltage swings of input signal assuring that no DC current is drained from the negative power supply. M_2 assures that the output node swings to the potential of the input signal at IN node when input goes to a negative potential (M_2 turning ON). M_3 is configured as a capacitor and used to assist in accelerating the switching time of the voltage divider. M_4 , M_5 , and M_6 (diode devices) are used to divide the voltage of an input signal provided to the voltage divider at input node. The resistance R is ballast resistor which drops any additional input voltages that exceed the value of $[\text{input voltage} - (n \cdot V_{th})]$, since n is the number of diodes.

Figure 3.37 illustrates the simulation results of the digital control circuitry. As can be seen from the transient simulation results, the control circuitry successfully receives the digital code and generates the required digital voltages to be applied to the gate and the body. It should be noted that there are as many control circuitry blocks as required. As an example for a 6-bit attenuator, there will be 6 of these circuits

controlling 6 attenuation stages individually.

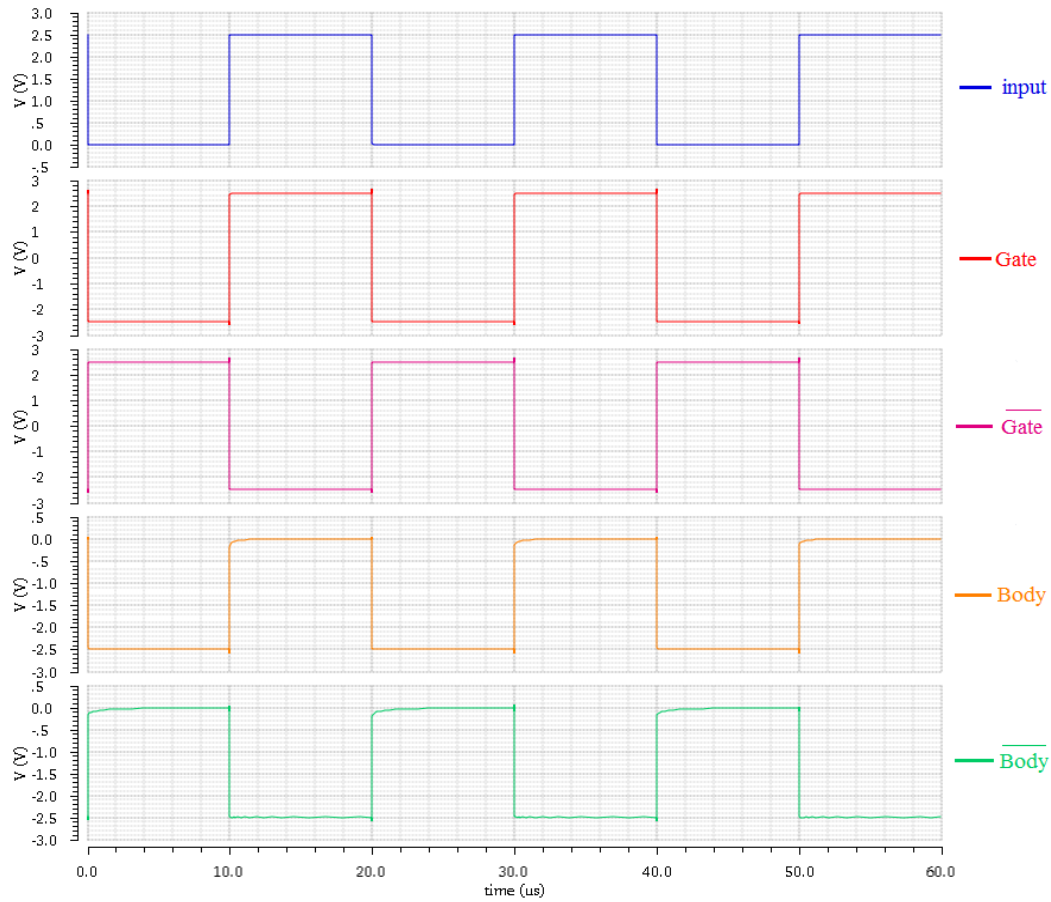


Figure 3.37: Simulation results of digital control circuitry.

3.4.1 Monte Carlo and corner simulations

Monte Carlo (statistical) analysis is used to estimate parametric yields and generate information about the performance characteristics of the designed circuit. A Monte Carlo statistical simulation requires many iterations in order to see the effect of multiple process extremes. Thus, Monte Carlo simulation included 100 steps over temperature and supply variations with global process variation is performed.

Figure 3.38, shows the Monte Carlo simulation result for one of the multiple iterations with different temperatures (-40°C , 27°C and 85°C). It shows Body control voltage for both ON and OFF cases of a switch. As it can be seen, for all the variation, the circuit can function and the body voltage value does not exceed 0 V and -2.5 V for ON and OFF cases, respectively.

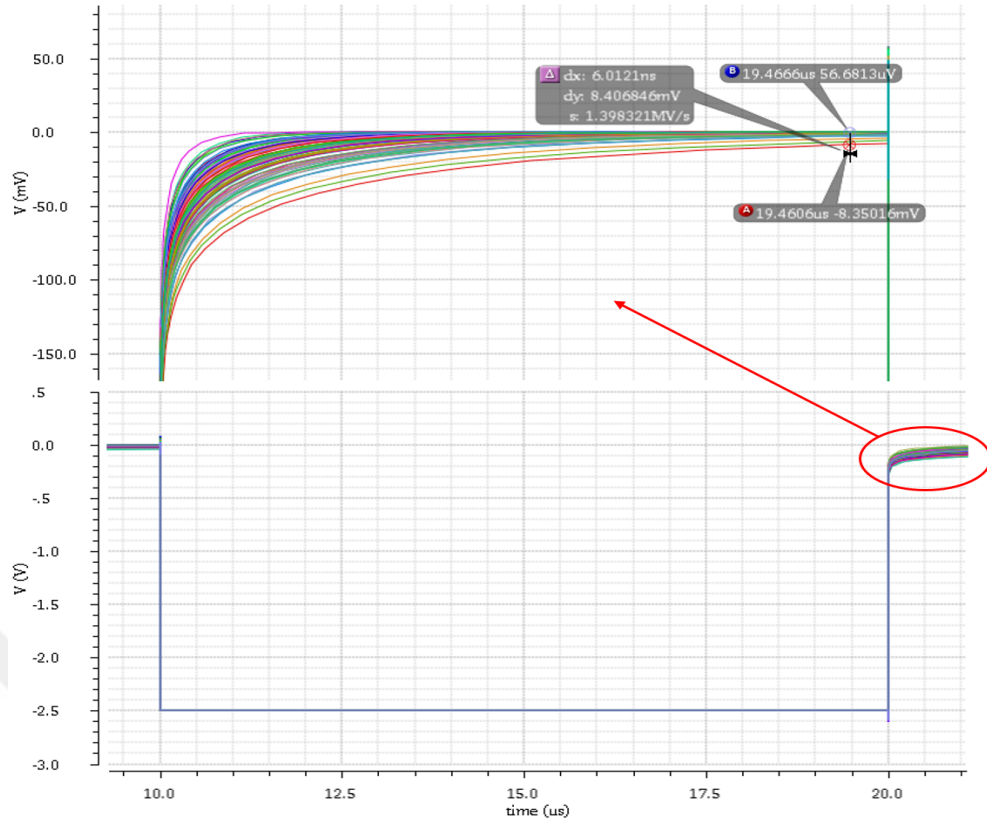


Figure 3.38: Monte carlo simulation for body voltage of transistors at -40°C .

The process variation (Monte Carlo and corner models) is set to match the spec tolerance. The results of the most important corner simulations for inverter transistors (FF and SS) are given in the Figures 3.39 and 3.40, respectively. The plots show that there is not a lot of delay and transistors work well.

3.4.2 Supply voltage variation simulations

The control circuit is simulated for 10% supply variation. Figure 3.41, shows the simulation results for both (0.25 V, 2.25 V) and (-0.25 V, 2.75 V) input voltages instead of (0 V, 2.5 V).

The simulation results demonstrate that control circuit works well and required voltage values for control the gate and body of transistors in both ON and OFF cases are achieved.

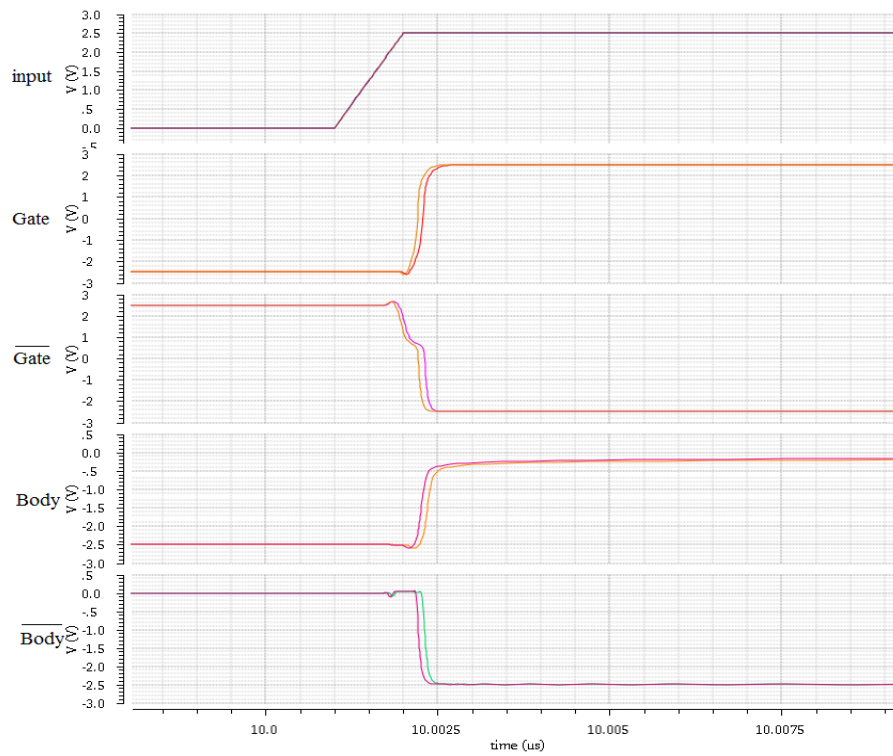


Figure 3.39: Corner simulation result when both of the inverter transistors are fast.

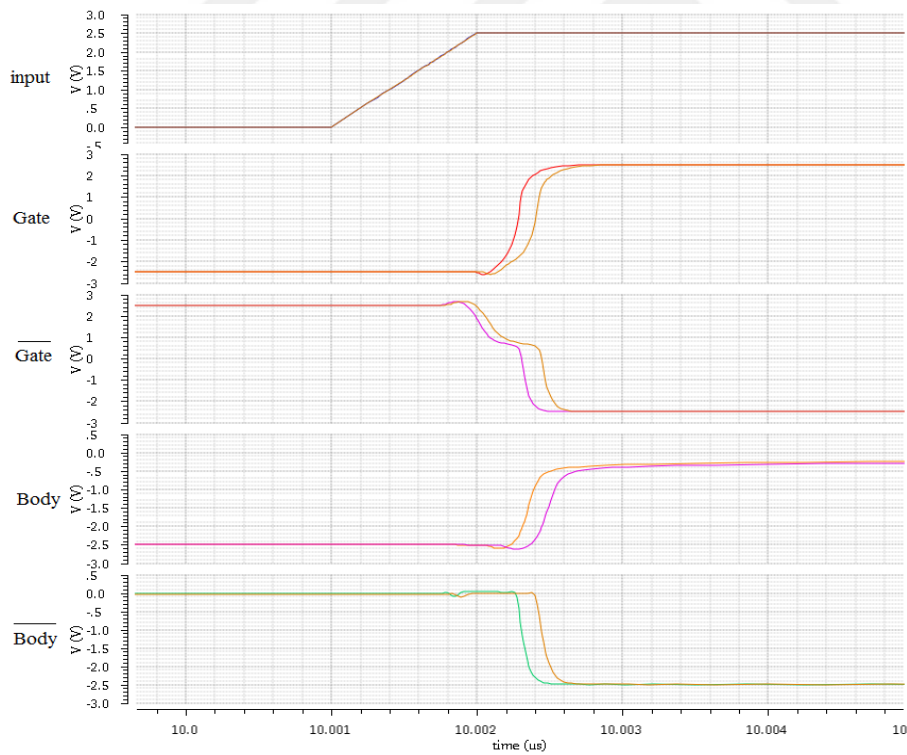


Figure 3.40: Corner simulation result when both of the inverter transistors are slow.

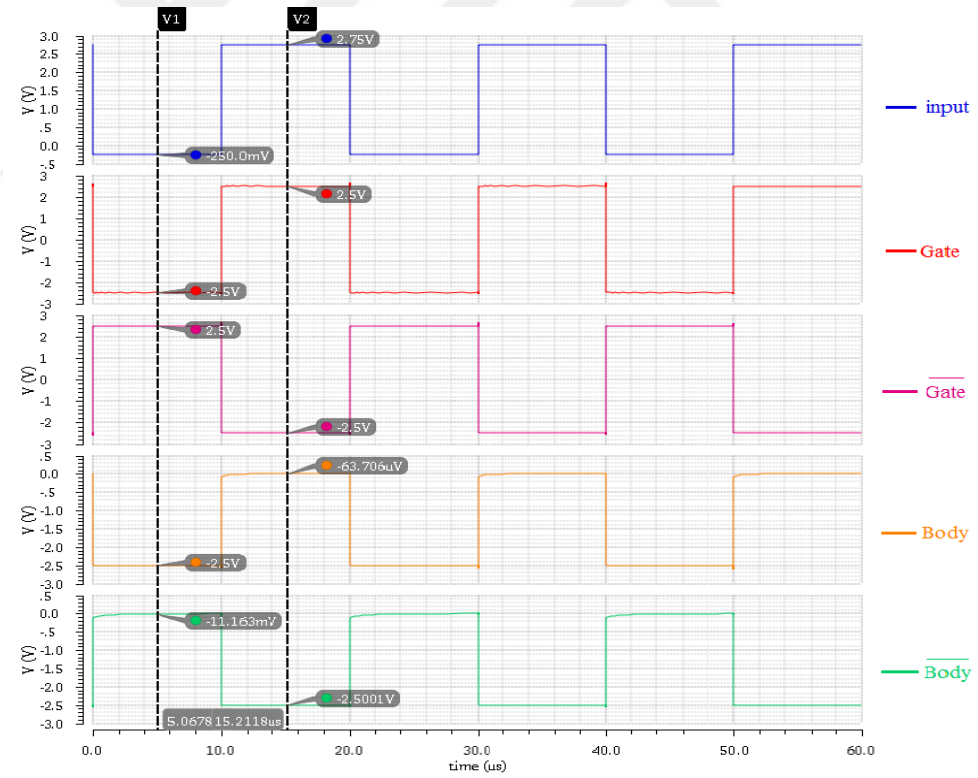
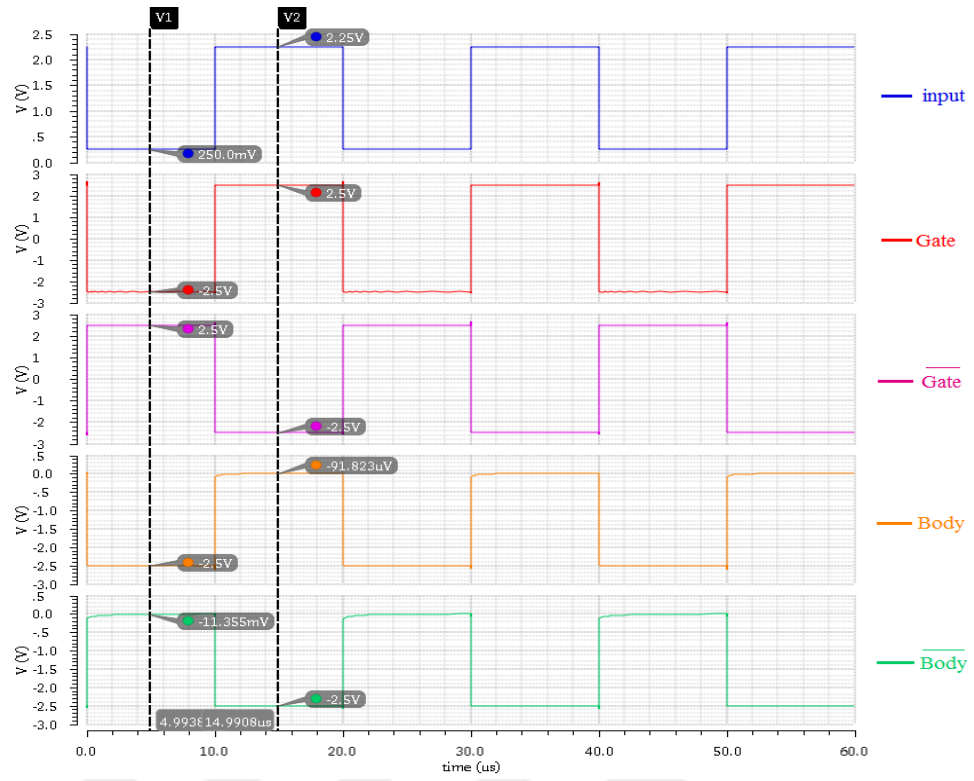


Figure 3.41: Supply variation (10% tolerance) simulation results.

The layout of the digital control circuitry is illustrated in the Figure 3.42.

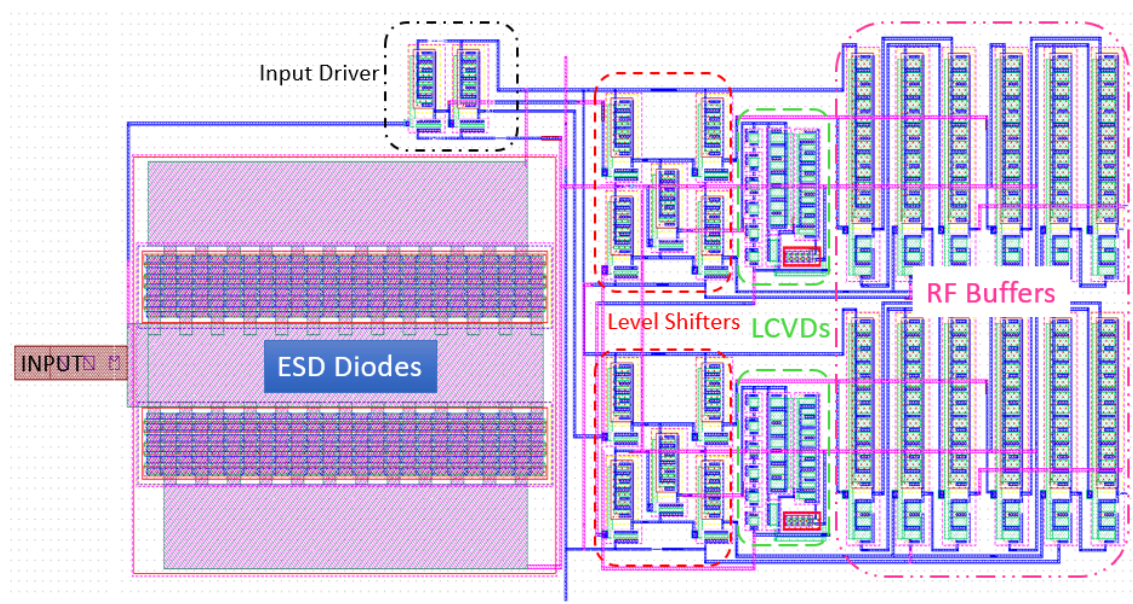


Figure 3.42: Layout of digital control circuitry.

CHAPTER 4

TEST SETUP AND MEASUREMENT RESULTS

This chapter presents the test results of the designed and fabricated 7-bit (covers 31.75 dB attenuation range with 0.25 dB steps) and 6-bit (covers 31.5 dB attenuation range with 0.5 dB steps) phase coherent DSAs. The measured results of the attenuators performance are demonstrated in terms of relative amplitude and phase, attenuation and phase errors, step and state errors, insertion loss, return losses, and linearity.

To do measurements, a compact printed circuit board (PCB) is designed and fabricated on 20 mil Rogers 4003 material. Schematic of the evaluation board (EVB) and PCB layout are shown in Figures 4.1 and 4.2, respectively.

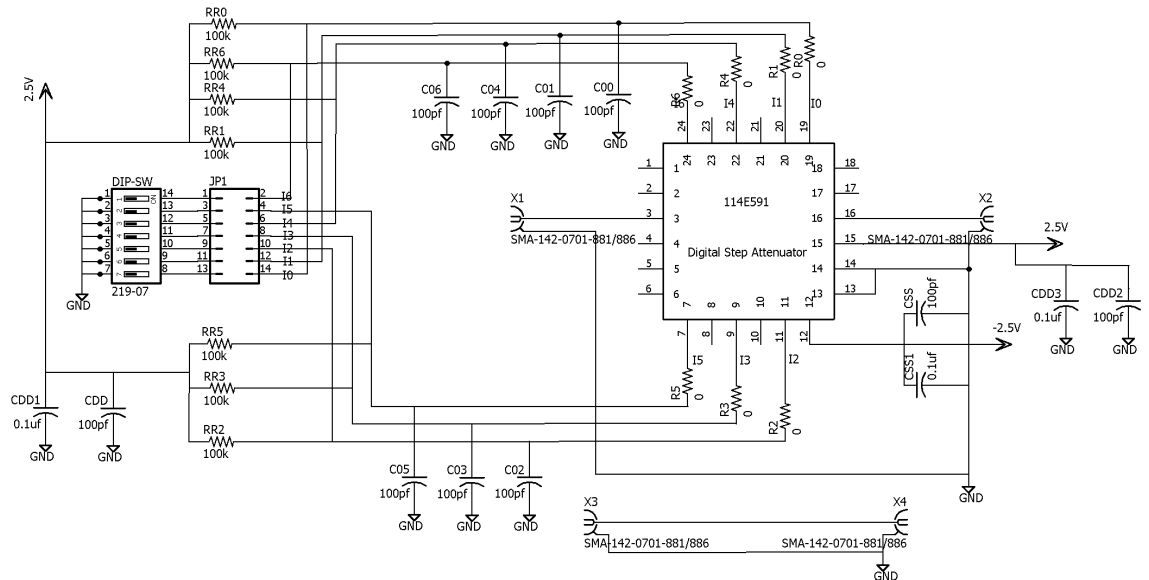


Figure 4.1: Evaluation board schematic.

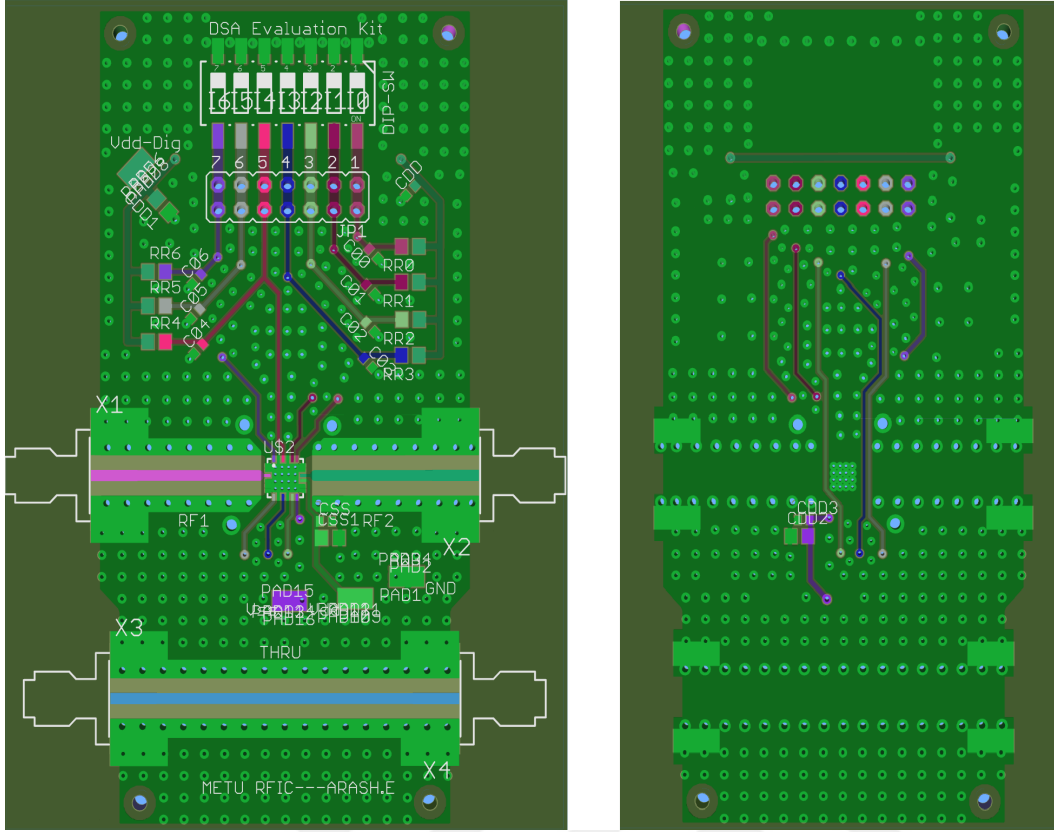


Figure 4.2: Evaluation board layout.

This EVB can be controlled manually or using a computer program to do automatic S-parameter measurements in parallel mode. The parallel control mode consists of seven (or six) control lines. Automatic measurement program can select the desired attenuation state, as shown in Table 4.1. (L = LOW = 0 V and H = HIGH = 2.5 V)

Table 4.1: Parallel Truth Table of 7-bit Attenuator

Parallel Control Setting							Attenuation Setting
C_7	C_6	C_5	C_4	C_3	C_2	C_1	
L	L	L	L	L	L	L	Reference I.L.
L	L	L	L	L	L	H	0.25 dB
L	L	L	L	L	H	L	0.5 dB
L	L	L	L	H	L	L	1 dB
L	L	L	H	L	L	L	2 dB
L	L	H	L	L	L	L	4 dB
L	H	L	L	L	L	L	8 dB
H	L	L	L	L	L	L	16 dB
H	H	H	H	H	H	H	31.75 dB

4.1 S-parameters Measurement Setup

Manual Control: In manual control, the control line of the attenuator is changed using the DIP-switch. Manual control to measure the S-parameter results can take a lot of time. For example, a 7-bits attenuator have $2^6 = 128$ different states. To ease the measurement, an automatic measurement setup is developed.

Automatic Control: In this method, S-parameters can be measured by using a computer and a programmable device (Arduino Mega). In order to evaluate the performance of the proposed attenuators the setup shown in Figure 4.3 is used.

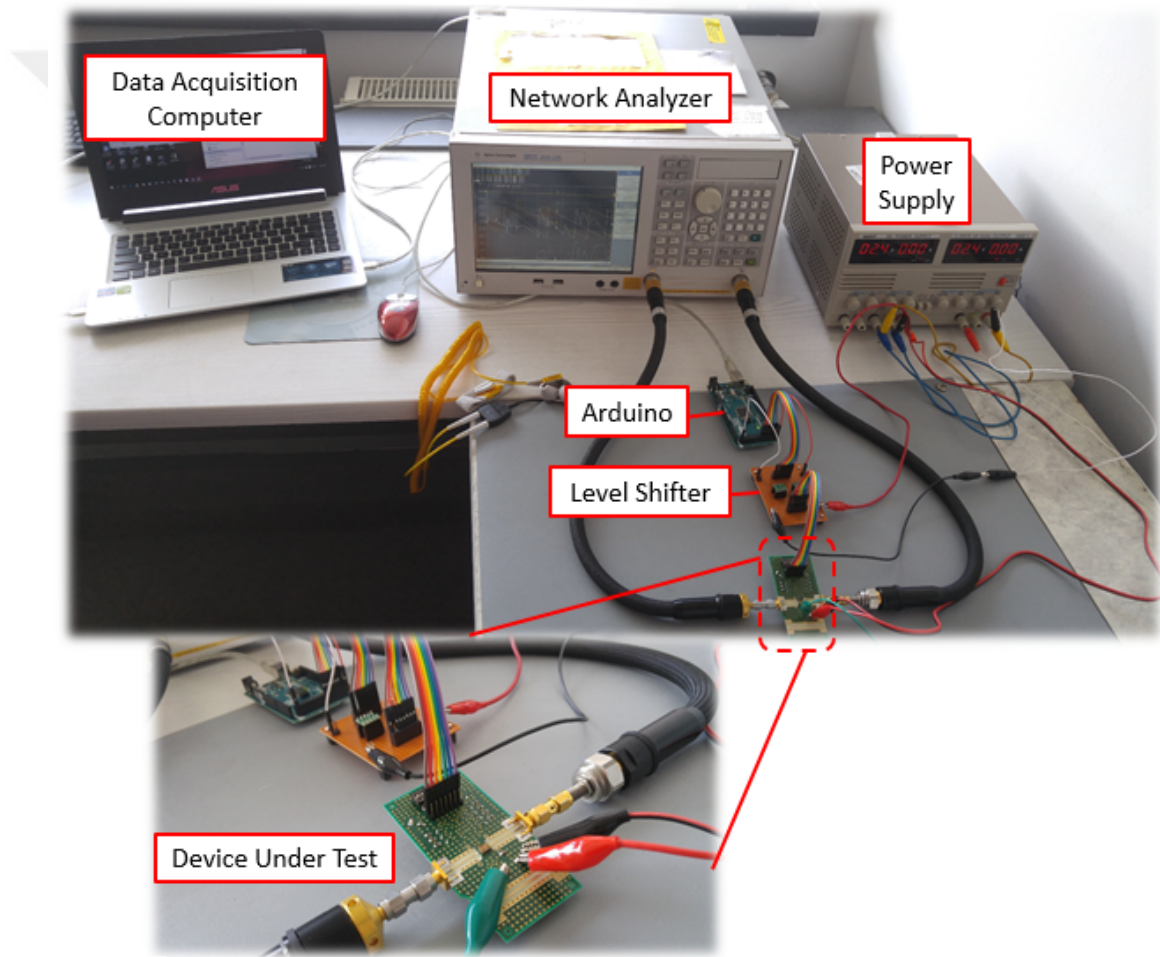


Figure 4.3: Photograph of the test setup used for automatic s-parameters measurements.

In this setup, computer is connected to the network analyzer and the arduino device. Arduino is controlled by computer to generate 7 (or 6) bits binary code from 0000000

to 1111111 (or 000000 to 111111) to select all the attenuation states one by one. S-parameters (amplitude and phase) data of the each attenuation state related to binary code is measured by the network analyzer (Agilent E5071C) and saved into an excel file.

Device under test require 0 V and 2.5 V in each control line to select the desired attenuation state. Since arduino outputs are 5 V, level shifters are used to convert 5 V to 2.5 V. Additionally, the required supply voltages (V_{dd} , V_{ss} and GND) is applied to the chip externally by power supply.

Arduino Mega control code is presented in Appendix B. In addition, the written C# code in the visual studio environment to control the network analyzer and save data at an excel file is presented in Appendix C.

4.1.1 7-bit Attenuator Measurements Results

Figure 4.4 shows the chip photograph of the fabricated attenuator encapsulated in a 4×4 mm QFN package. The chip size, including pads, is $950\mu\text{m} \times 1950\mu\text{m}$. The S-parameters are obtained using an Agilent E5071C network analyzer via the automatic measurement setup.

Figure 4.5 shows the measured relative attenuation of the 7-bit DSA. As it can be seen, attenuation curves have a dip that started for frequencies higher than 2.7 GHz, especially for attenuation states higher than 16 dB. The same thing is observed during the 3D EM simulations for frequencies higher than 3 GHz (Referring to Figure 3.21). In addition, attenuation ripples is created at frequencies higher than 3 GHz.

The reasons of mentioned flaws will be investigated in the future but most probability this problem is coming from the wirebonds effects and coupling between them or impedance matching problem between the PCB and the IC.

Due to the mentioned problems, all the measured results are reported up to 2.7 GHz. Even with this limitation, the developed IC can work for all GSM and LTE 2600 frequencies Covering a frequency band between 100 MHz and 2.7 GHz.

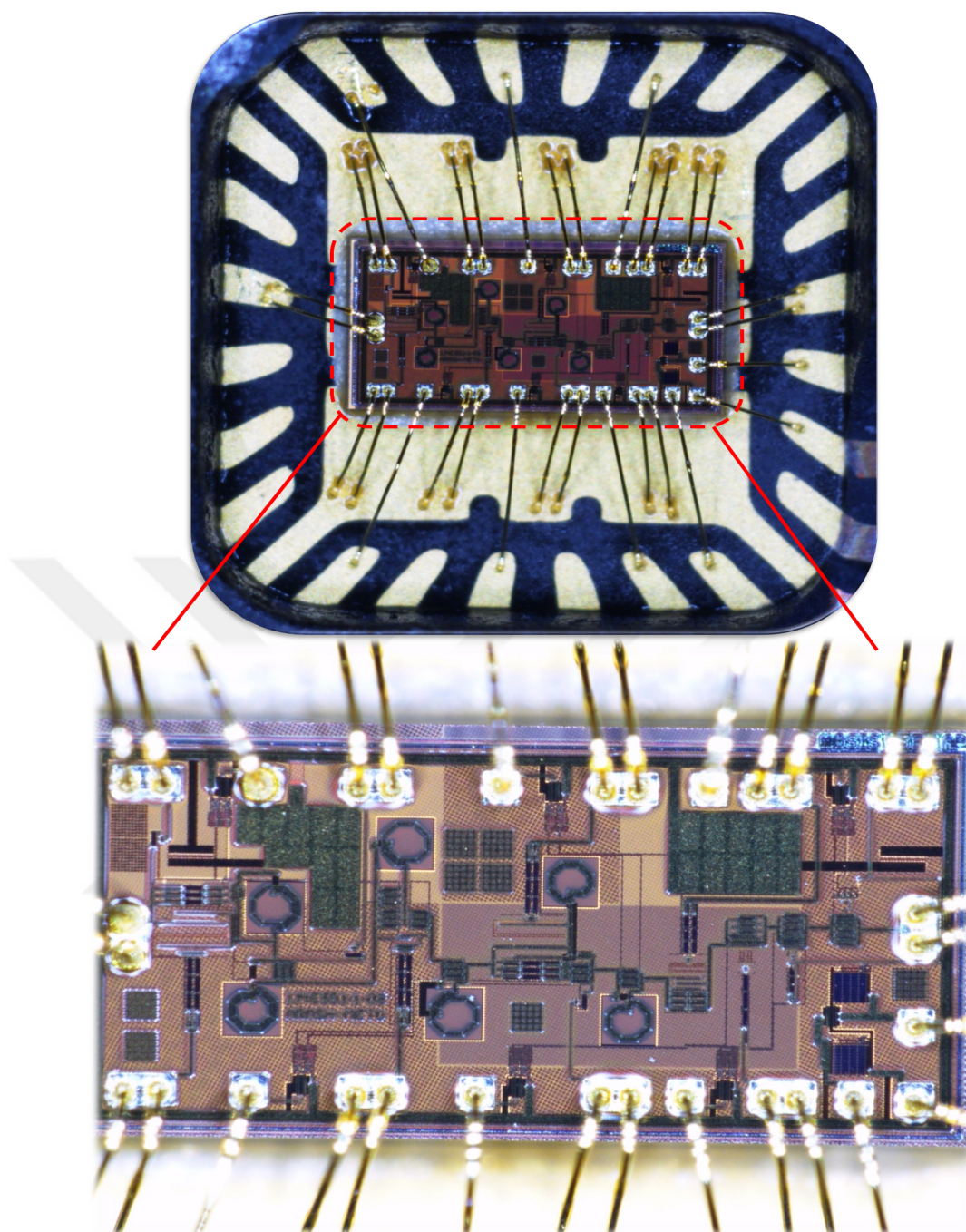


Figure 4.4: Microphotograph of the 7-bits digital step attenuator wire-bonded to the QFN package.

Referring to Figure 4.5, the measured results show flat attenuation performances up to 2.7 GHz with excellent accuracy. The measured minimum and maximum insertion loss are 1.3 dB and 3.5 dB, respectively.

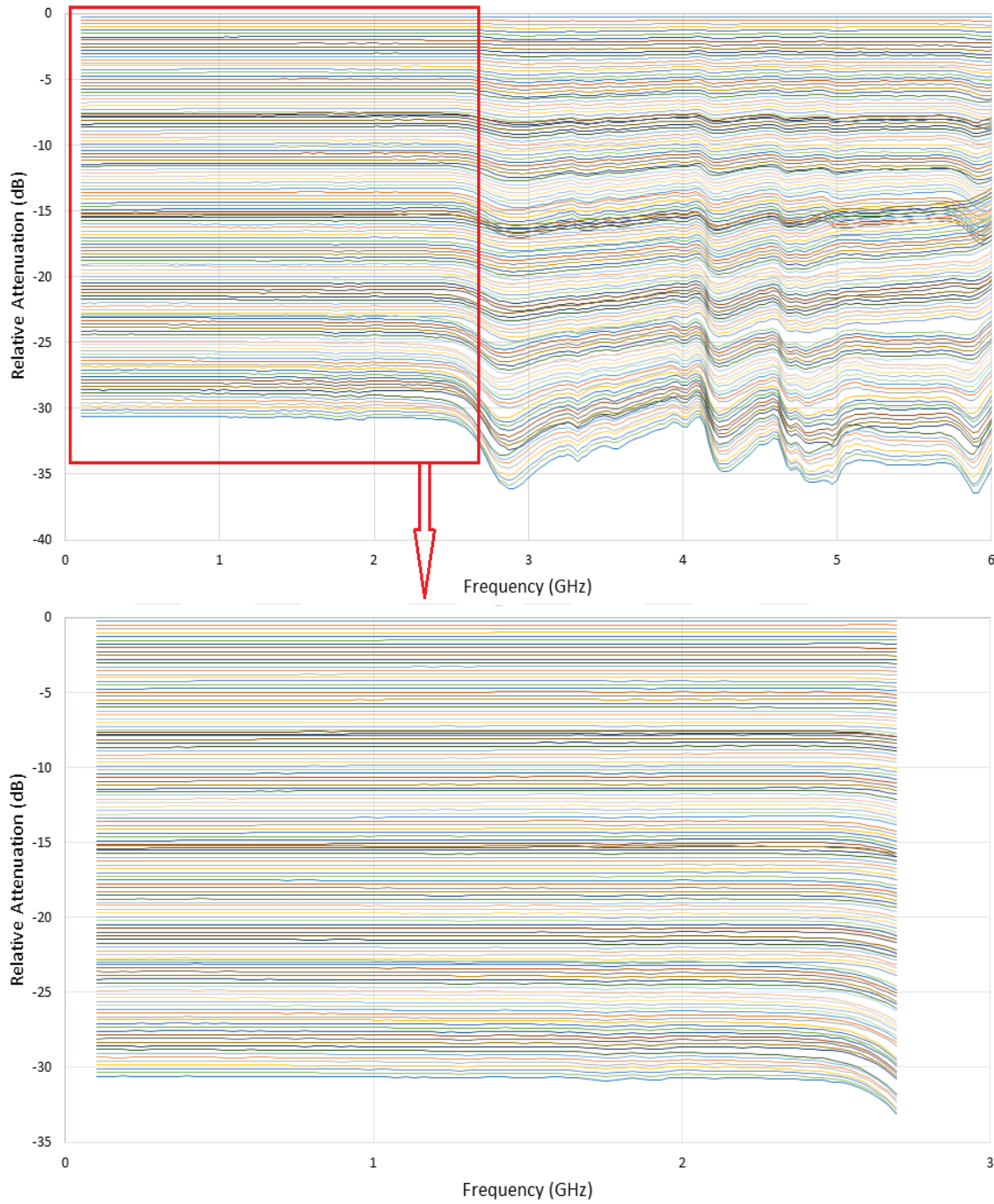


Figure 4.5: Measured relative attenuations of the DSA (all states).

Figure 4.6 shows attenuation errors of the major states. 8 dB and 16 dB states are intentional adjusted to have attenuation lower than ideal values. These states are sacrificed to decrease the step and attenuation errors at high frequencies. Figure 4.7 shows the step and state errors of the 7-bit attenuator. Monotonicity is held so long as step-attenuation does not cross below -0.4 dB. These errors are plotted over the

attenuation setting at different frequencies. As can be seen from the plots, attenuation accuracy is very good for all frequency points.

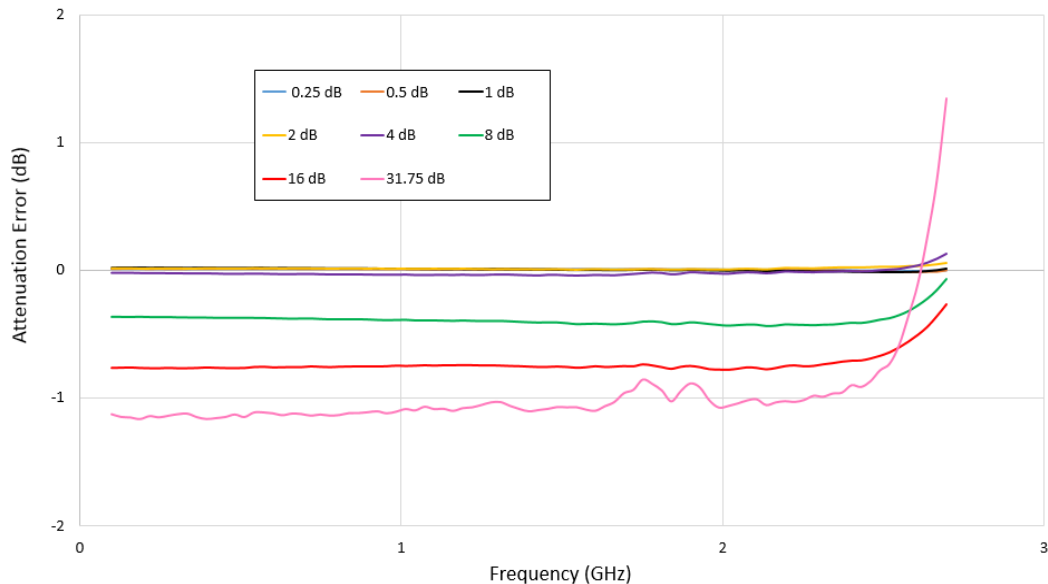


Figure 4.6: Major state bit error vs. Frequency.

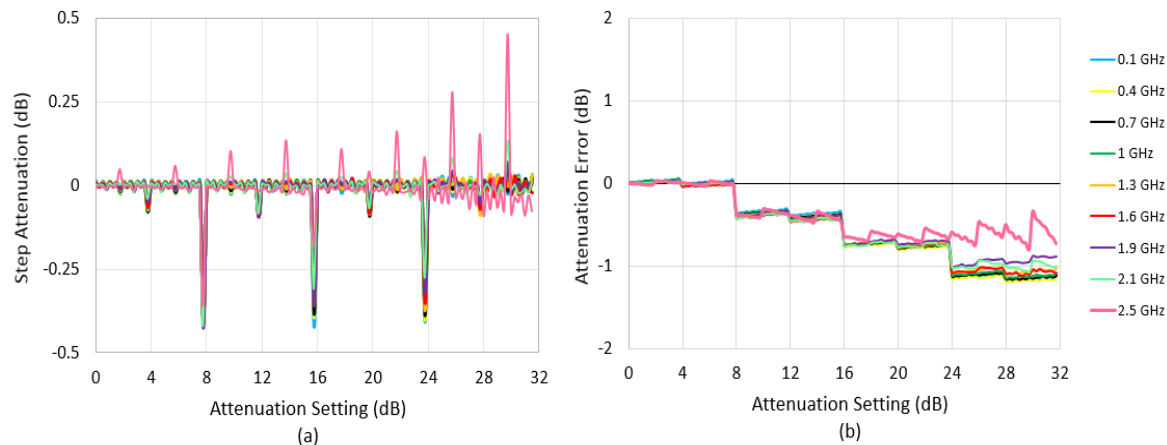


Figure 4.7: Measured: (a) Step and (b) Attenuation error vs. Attenuation setting.

An important parameter in characterizing digital attenuators is the voltage standing wave ratio (VSWR), which is often shown as return loss. Return loss is the ratio of the reflected power to the incident power that varies as a function of frequency. It is essentially a measure of how well the attenuator will transfer RF signal with minimal reflections and power loss. Figure 4.8 shows typical input and output return loss versus frequency. Note the input and output return losses are better than 14.5 dB.

Figure 4.9 shows the measured insertion phase of the all attenuation states relative to

the phase of the minimum attenuation state. The maximum phase shift is less than $\pm 3^\circ$ up to 2.5 GHz frequency.

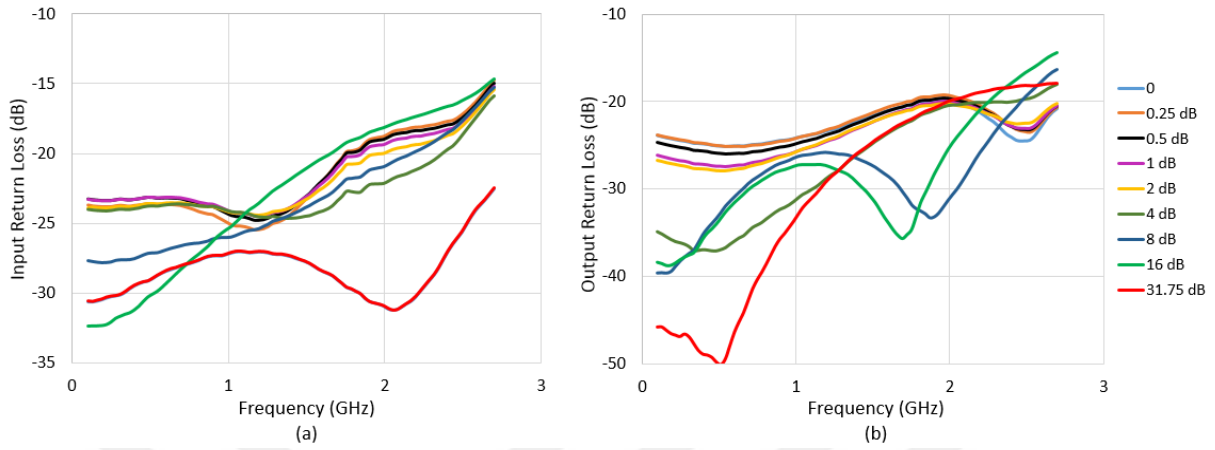


Figure 4.8: Measured: (a) Input and (b) Output return loss vs. Frequency.

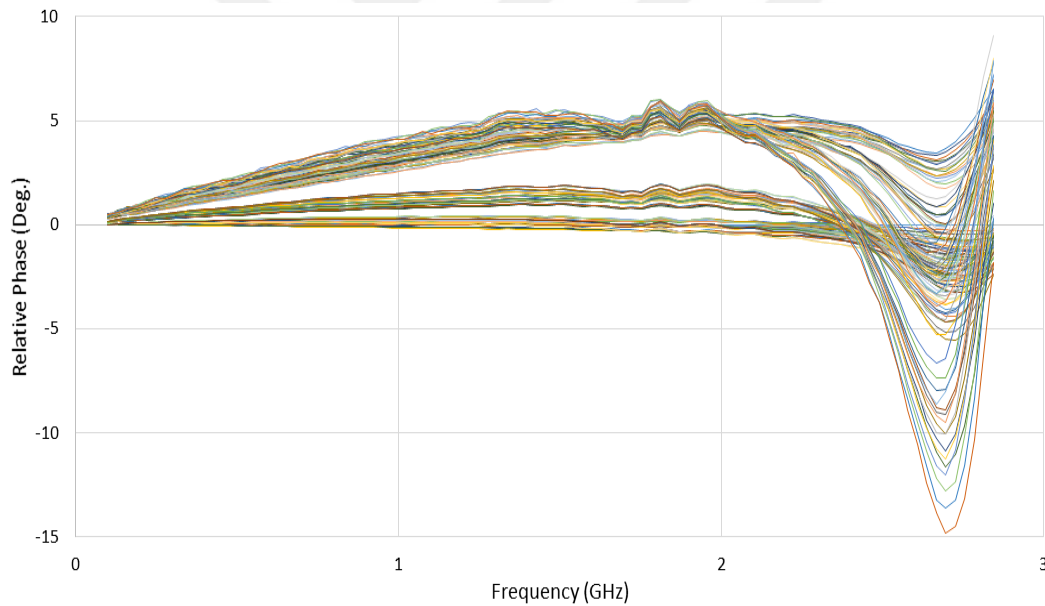


Figure 4.9: Measured relative phases of the DSA (all states).

Figures 4.10 and 4.11 show the relative phase error of the DSA versus frequency and attenuation setting, respectively. It can be seen that phase error is compensated up to 2.5 GHz. The maximum phase error is 6.5° in operation frequency range.

In comparison with industrial attenuator [4], the proposed attenuator is improved the phase shift problem approximately 4 times.

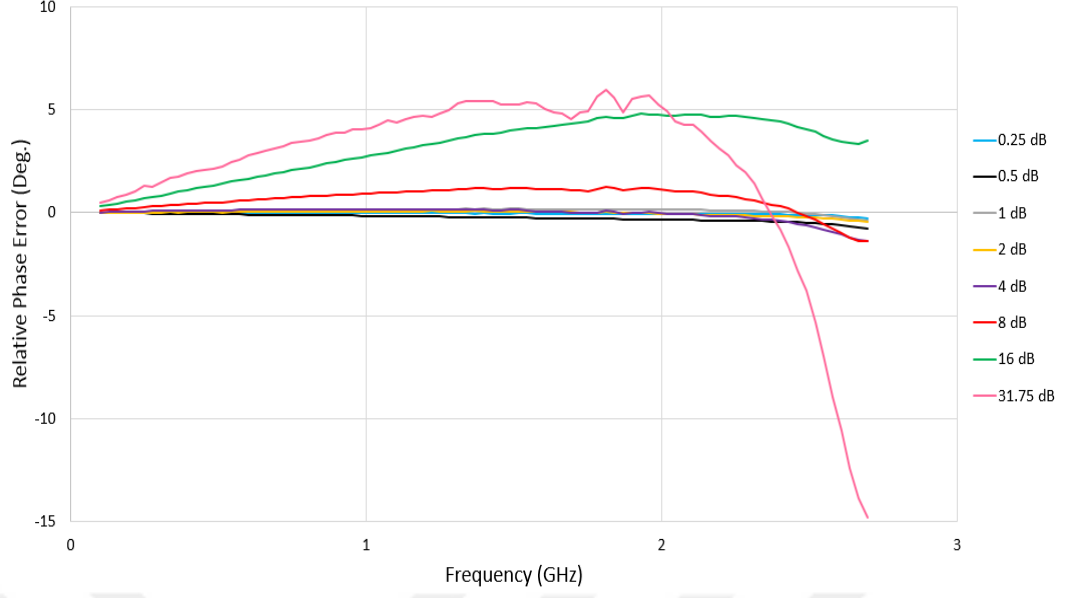


Figure 4.10: Relative phase error vs. Frequency.

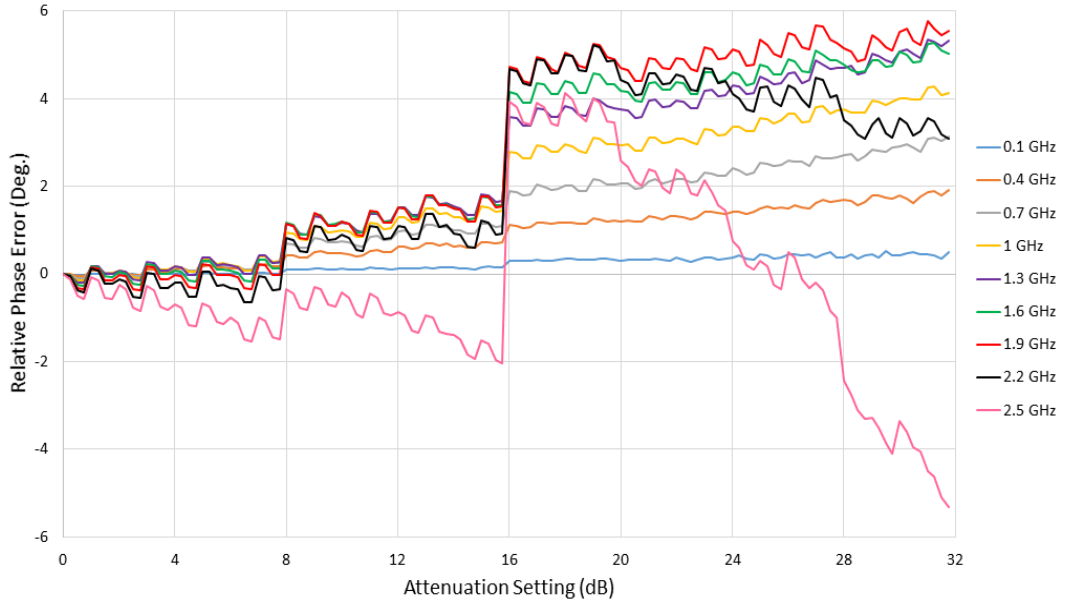


Figure 4.11: Relative phase error vs. Attenuation setting.

Moreover, the rms imbalance is calculated using the standard deviation of the insertion phase as:

$$RMS\ Phase\ Error = \sqrt{\frac{1}{128} \sum_{i=0}^{127} (S_{21ph,i} - \overline{S_{21ph}})^2} \quad (4.1)$$

where $S_{21ph,i}$ is the insertion phase of i^{th} attenuation state and $\overline{S_{21ph}}$ is the mean of all the insertion phases. RMS phase error is less than 3.6° up to 2.5 GHz, as shown in Figure 4.12.

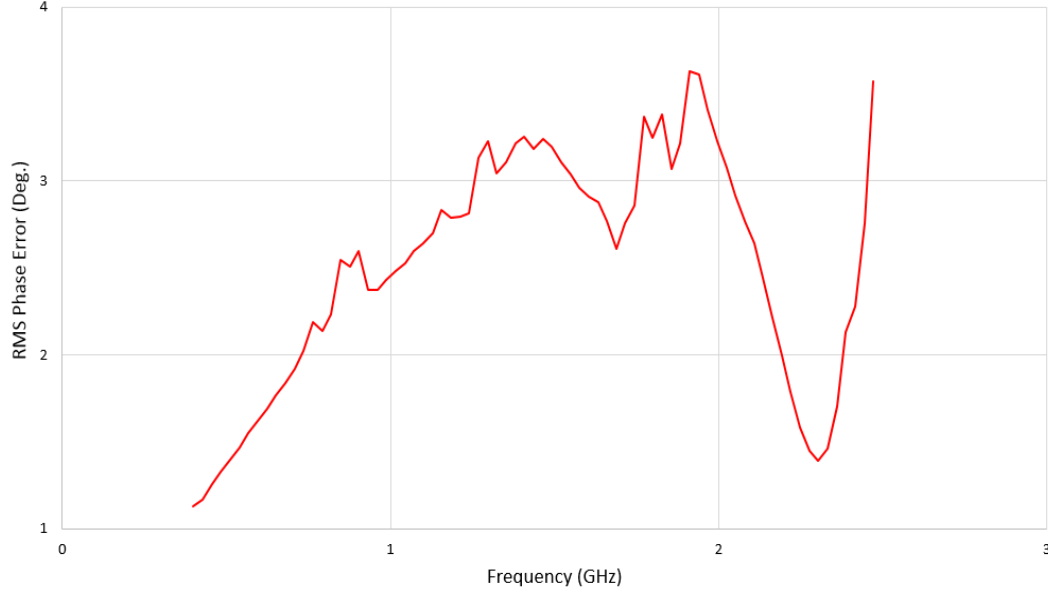


Figure 4.12: Measured RMS phase error vs. Frequency.

4.1.2 6-bit Attenuator Measurements Results

Similar to 7-bit attenuator, the S-parameters are obtained using an Agilent E5071C network analyzer and the automatic measurement setup.

Figure 4.13 shows the measured relative attenuations of the 6-bit DSA. Similar to 7-bit version, attenuation curves have a dip that started for frequencies higher than 2.7 GHz, especially for attenuation states higher than 16 dB in addition to attenuation ripples at frequencies higher than 3 GHz.

Due to the mentioned problems, all the measured results are reported up to 2.7 GHz.

Figure 4.14 shows attenuation errors of the major states. Step and state errors of the 6-bit attenuator are illustrated in Figure 4.15. Monotonicity is held so long as step-attenuation does not cross below -0.5 dB.

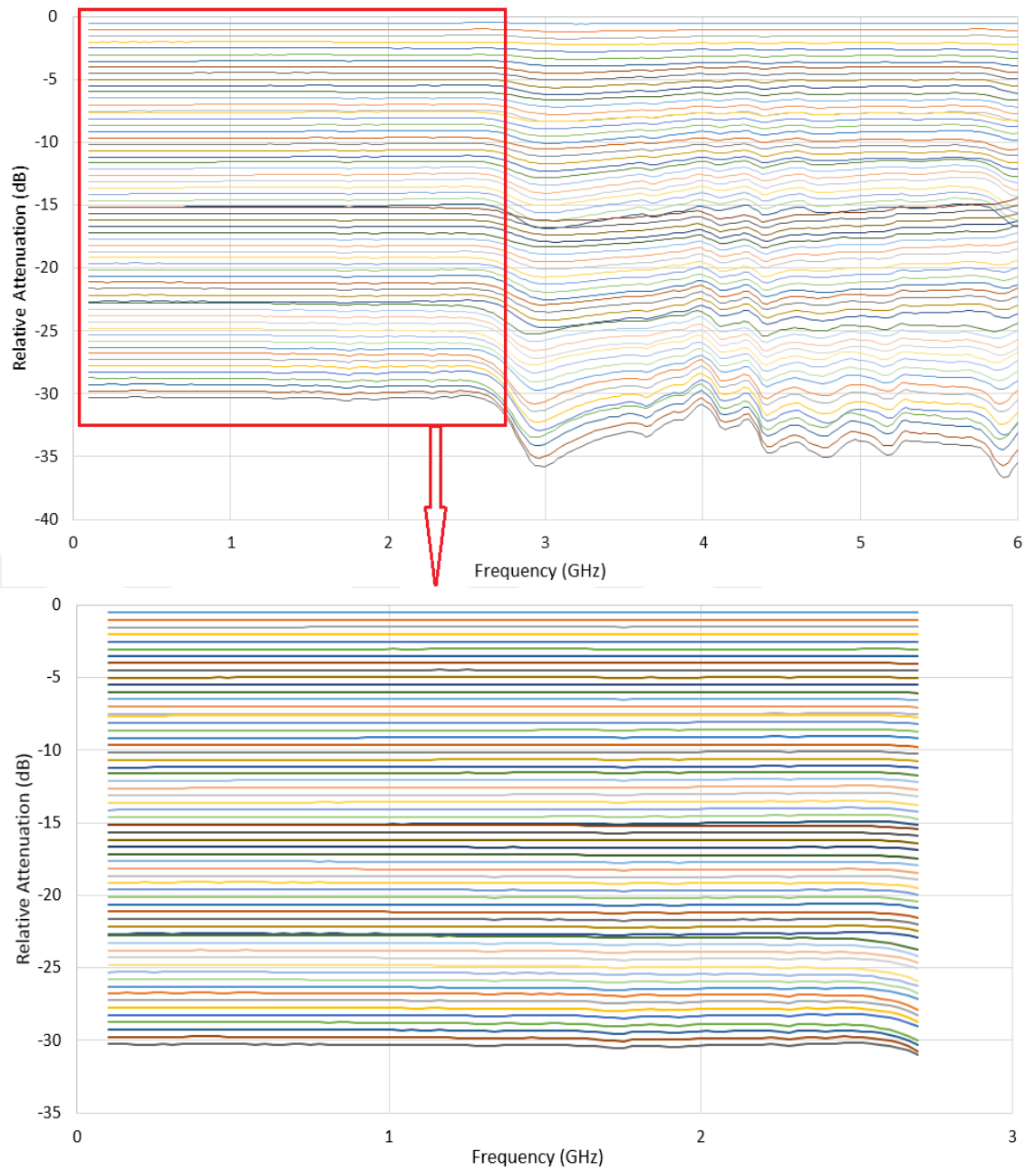


Figure 4.13: Measured relative attenuations of the DSA (all states).

Figure 4.16 shows input and output return loss versus frequency. Note the input and output return losses are typically as good as 15 dB up to 2.7 GHz for all attenuation states.

Figure 4.17 shows the measured relative phase of the 6-bit DSA. The maximum phase imbalance is approximately 5° up to 2.5 GHz.

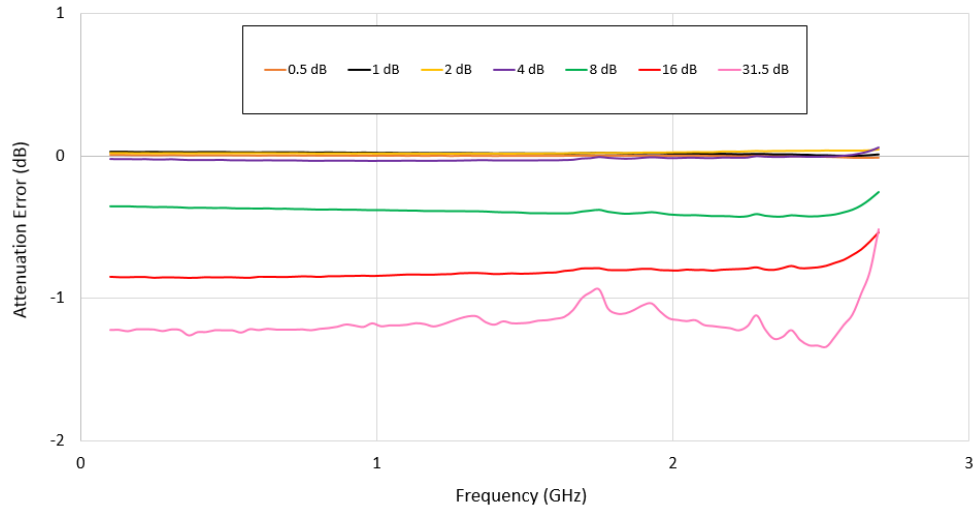


Figure 4.14: Major state bit error vs. Frequency.

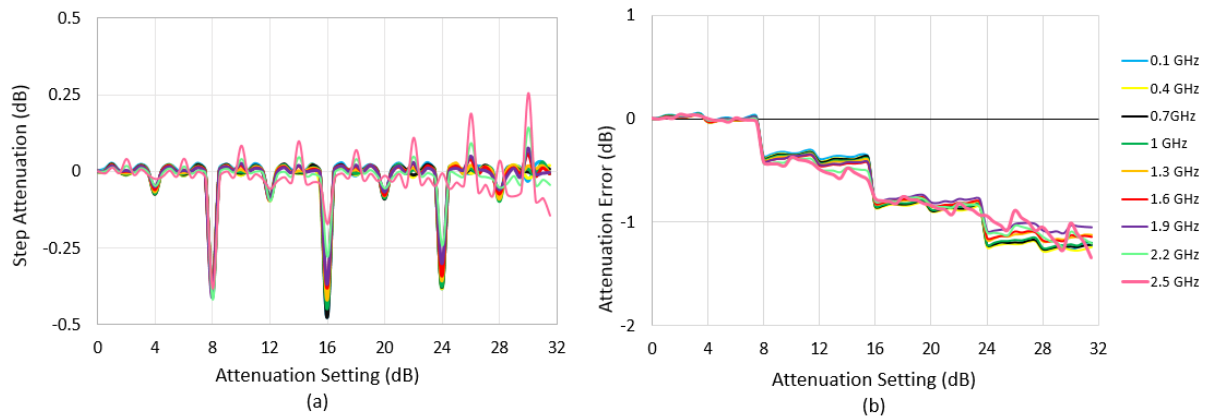


Figure 4.15: Measured: (a) Step and (b) Attenuation error vs. Attenuation setting.

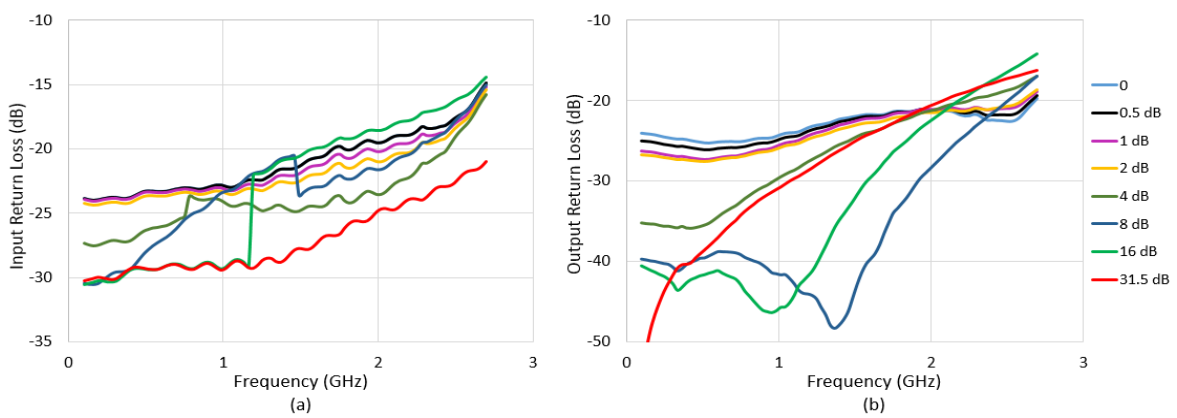


Figure 4.16: Measured: (a) Input and (b) Output return loss vs. Frequency.

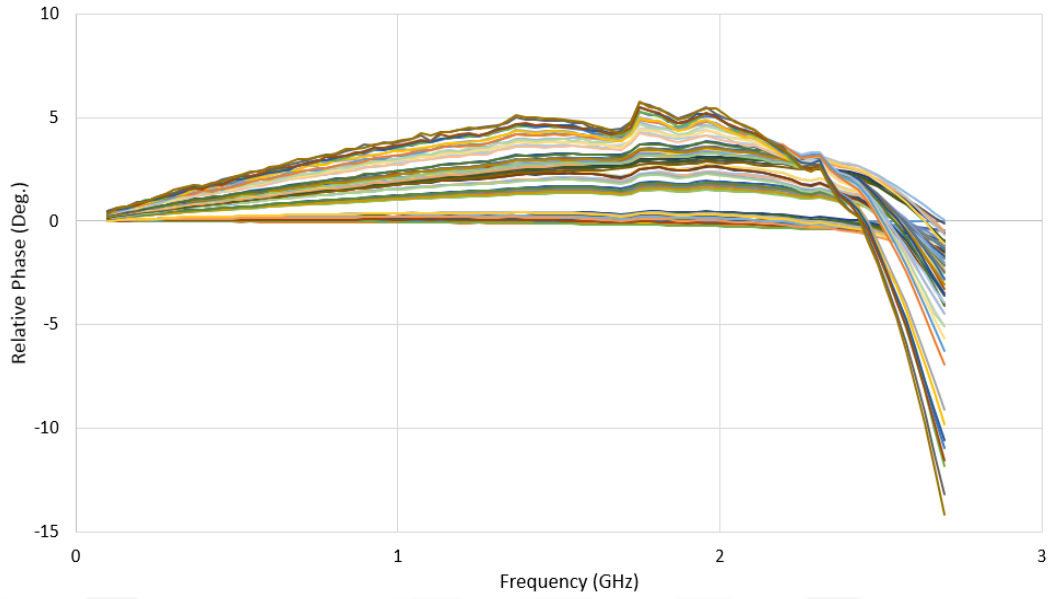


Figure 4.17: Measured relative phases of the DSA (all states).

Figures 4.18 and 4.19 show the relative phase error of the DSA versus frequency and attenuation setting, respectively. The maximum phase error is 5.2° in operation frequency range. In comparison with industrial attenuator [4], the proposed attenuator decreased the phase shift approximately 5 times.

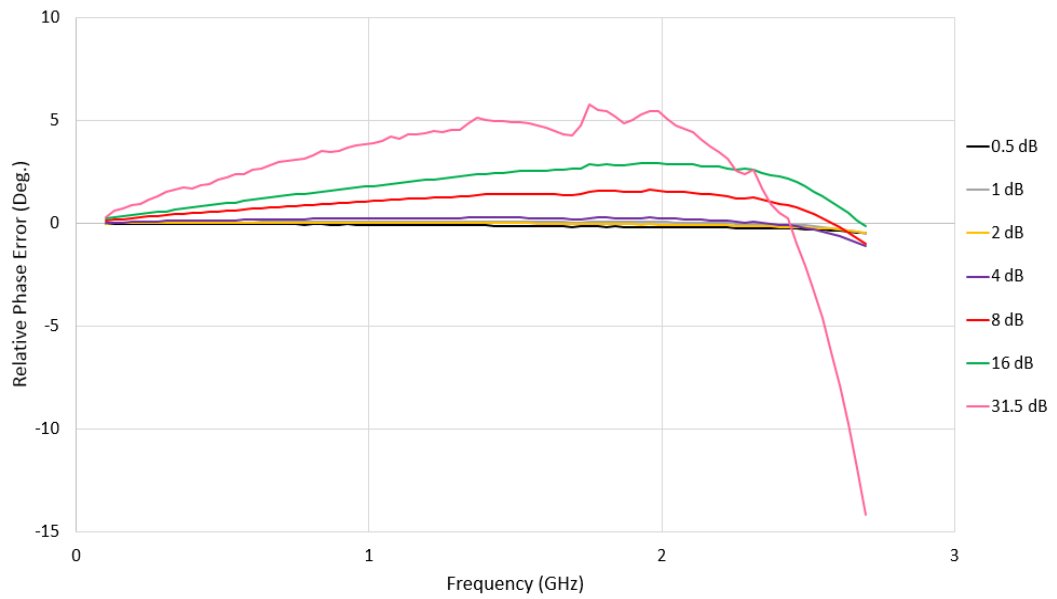


Figure 4.18: Relative phase error vs. Frequency.

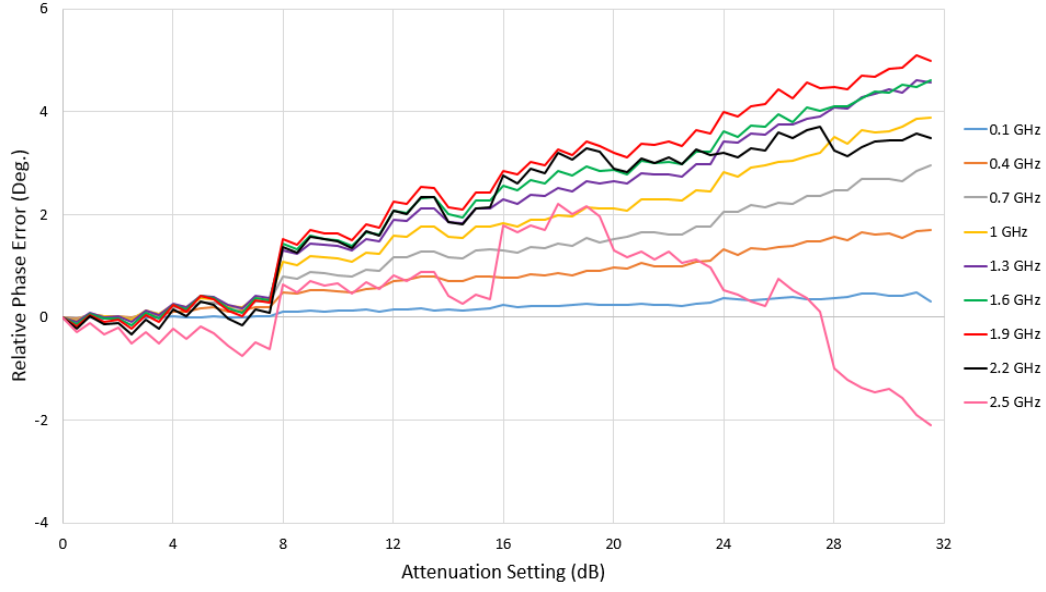


Figure 4.19: Relative phase error vs. Attenuation setting.

Moreover, the rms imbalance is also calculated as:

$$RMS \text{ Phase Error} = \sqrt{\frac{1}{64} \sum_{i=0}^{63} (S_{21ph,i} - \overline{S_{21ph}})^2} \quad (4.2)$$

Figure 4.12 shows the RMS phase error that is less than 3.2° up to 2.5 GHz.

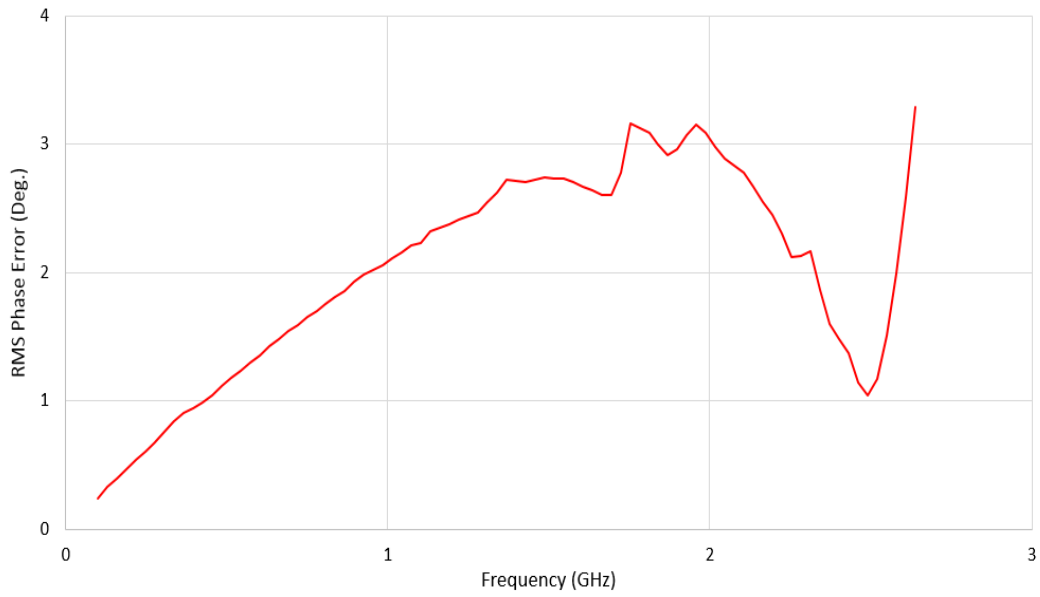


Figure 4.20: Measured RMS phase error vs. Frequency.

4.2 Linearity Measurement Setup

4.2.1 7-bit Attenuator Linearity Measurements

Digital attenuators are rated for acceptable RF input power, usually in terms of the 1 dB compression point and occasionally in terms of the maximum acceptable input power level that can be tolerated without damage to the attenuator. Exceeding a digital attenuator's 1 dB compression point generally results in the generation of elevated output harmonic and spurious signal levels due to the nonlinear behavior of FETs used in DSAs.

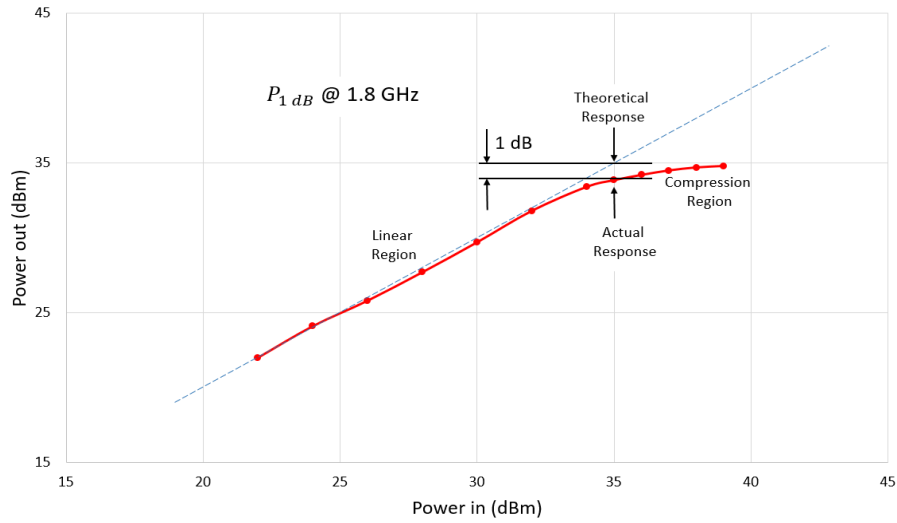


Figure 4.21: P_{1dB} compression point of the proposed DSA .

As can be seen from the above figure, proposed attenuator has very high 1 dB compression point that is measured at 35 dBm. To measure the P_{1dB} compression point, very high input powers are applied to the attenuator, while the attenuator is designed to have maximum 30 dBm input power. As a result, input attenuation block (16 dB) is damaged. Therefore, for the rest of the measurements 0.1 dB compression are reported instead of 1 dB. $P_{0.1dB}$ is better than 32 dBm.

Because of the linearity requirements of modern applications, digital step attenuators are also characterized in terms of their input third-order intercept point (Input IP3).

Figures 4.22 and 4.23 show block diagram and measurement setup photograph for the IIP3 measurements.

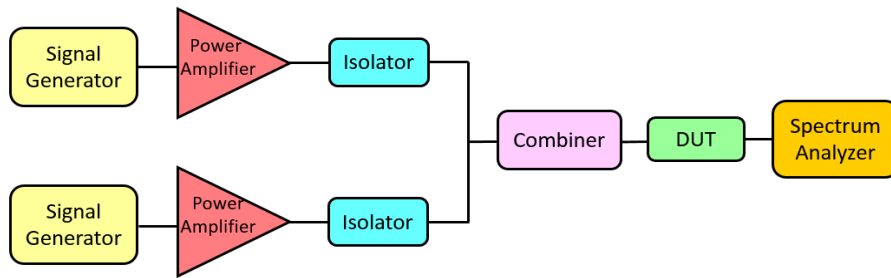


Figure 4.22: Block diagram of IIP3 measurement setup.

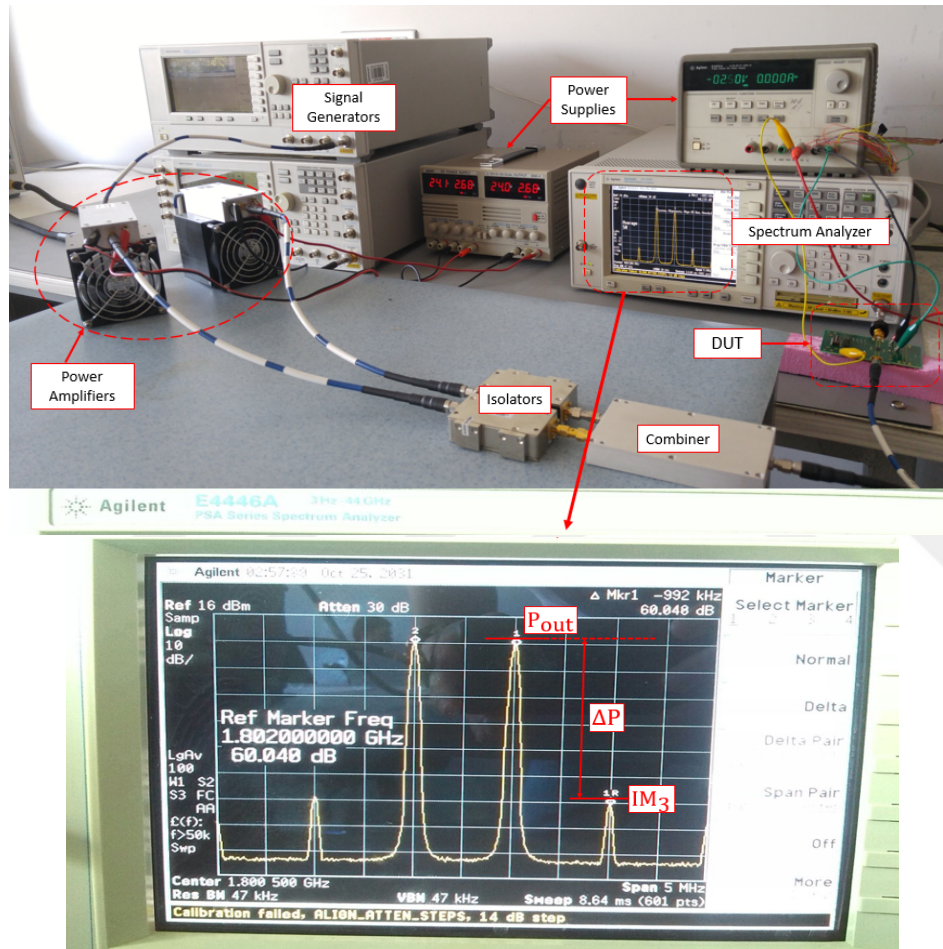


Figure 4.23: Setup photograph of IIP3 measurements.

The IIP3 of the attenuator is measured by applying two fundamental tones separated by 1 MHz at the input. These two tones are generated by two signal sources followed by external high linearity, high gain power amplifiers. This ensures small harmonic content of the fundamental tones. Two isolators are used at the output of the amplifiers to isolate them from each other, so that they do not mix with each other creating

intermodulation tones and reducing the measurement limit of the setup. These amplified and isolated fundamental tones are then combined with a power combiner and applied to the input of the DUT. The output of the DUT is measured using a Spectrum Analyzer.

IIP3 of the attenuator is then calculated with the following formula:

$$IIP3 = P_{out} + \frac{\Delta P}{2} + Attenuation \quad (4.3)$$

where P_{out} is the fundamental power. Using this equation, IIP3 of the proposed attenuator are measured for major states at four different frequencies, as shown in Figure 4.24.

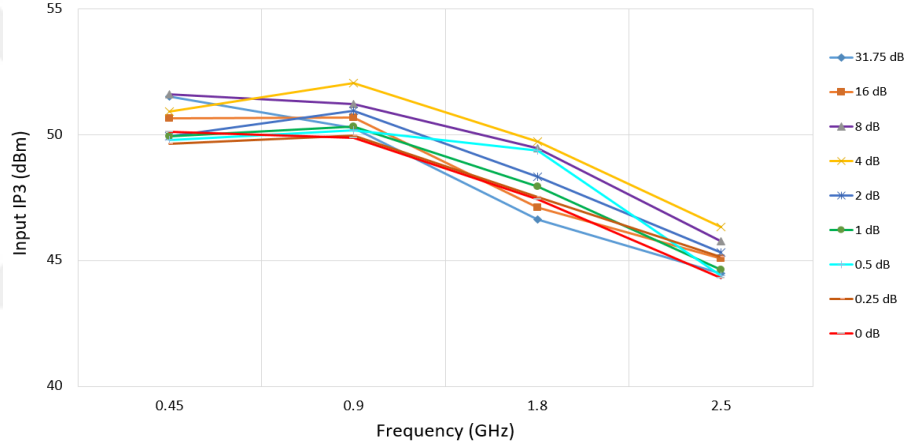


Figure 4.24: Measured IIP3 for major states of 7-bit attenuator at different frequencies.

The noise floor of a typical Spectrum Analyzer is around -100 dBm. Therefore, in order to measure high IIP3 values the fundamental tones should be very large. As an example if fundamental tones are 10 dBm each, the system IIP3 measurement limit is 65 dBm ($10 + \frac{10 - (-100)}{2}$). Since the attenuators are designed to have large IIP3 values, they are measured with high fundamental powers (two tones at +18 dBm) in order not to be limited by the the measurement setup IP3.

P_{1dB} value of the attenuator as linearity figure of merit is very high, but the aim of this research was to have IIP3 value to be higher than 50 dBm for all attenuation levels. The measured IIP3 values are less than expected values for all states at higher

frequencies. The reason of these should be investigated.

4.2.2 6-bit Attenuator Linearity Measurements

Measured P_{1dB} compression point for different attenuation states at 2.5 GHz is illustrated in Figure 4.25. In addition, input IP3 values are calculated using equation (4.3) for all attenuation states at different frequencies. The IIP3 results are presented in Figure 4.26.

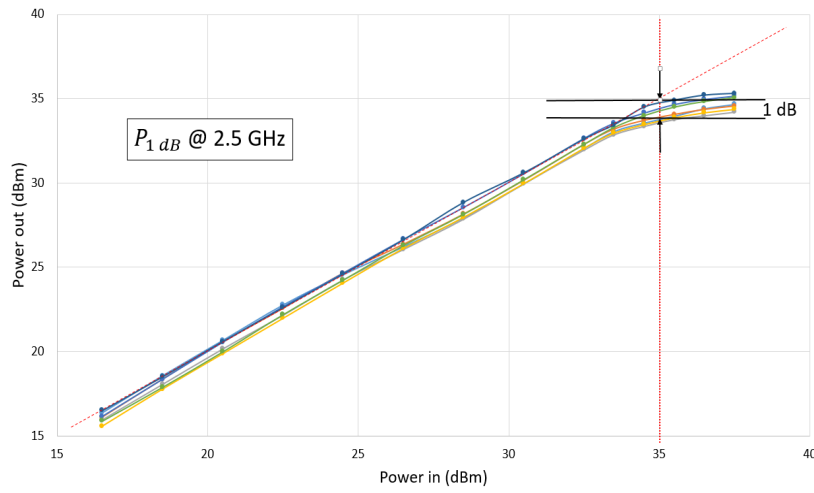


Figure 4.25: P_{1dB} compression point of the proposed 6-bit DSA.

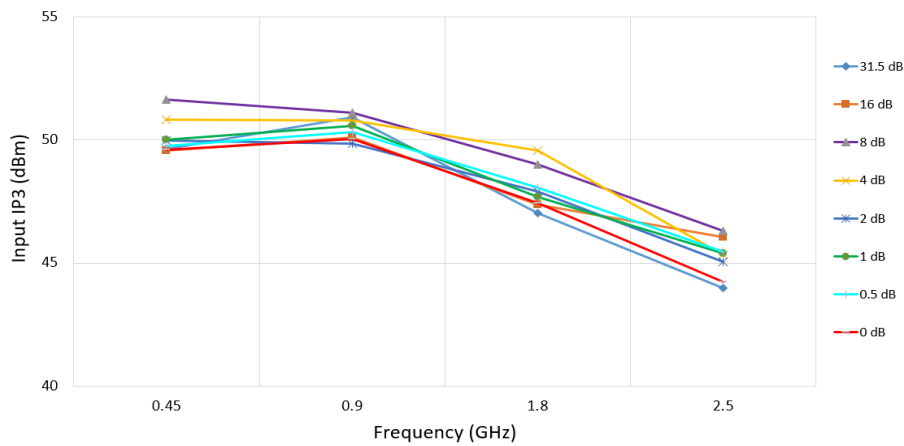


Figure 4.26: Measured IIP3 for major states of 6-bit attenuator at different frequencies.

4.3 Performance Summary

Tables 4.2 and 4.3 show the performance summary of the 7-bit and 6-bit proposed digital step attenuators, respectively.

Table 4.4 shows the performance of the proposed attenuators in comparison with the state of the art attenuators.

Table 4.2: Electrical Specifications: 0.25 dB steps, 7-bit Attenuator

Parameter	Condition	Frequency	Min	Typ	Max	Unit
Operating frequency			100 MHz		2.7 GHz	As shown
Attenuation range	0.25 dB Step			0 - 31.75		dB
Insertion Loss		0.1 GHz - 1.5 GHz		1.7	2	dB
		1.5 GHz - 2.7 GHz		1.8	3.3	dB
Attenuation Error	0 dB - 15.75 dB Attenuation Setting	0.1 GHz - 1.5 GHz				dB
		1.5 GHz - 2.7 GHz				dB
	16 dB - 31.75 dB Attenuation Setting	0.1 GHz - 1.5 GHz				dB
		1.5 GHz - 2.7 GHz				dB
Return Loss	Input port	0.1 GHz - 1.6 GHz		20		dB
		1.6 GHz - 2.7 GHz		15		dB
Return Loss	Output port	0.1 GHz - 1.6 GHz		20		dB
		1.6 GHz - 2.7 GHz		15		dB
Relative Phase	0 dB - 31.75 dB Attenuation Setting	100 MHz - 2.5 GHz		6.2		deg
Input P1 dB		100 MHz - 2.7 GHz	33	35		dBm
Input IP3	Two tones at +18 dBm, 1 MHz spacing		44		51	dBm

Table 4.3: Electrical Specifications: 0.5 dB steps, 6-bit Attenuator

Parameter	Condition	Frequency	Min	Typ	Max	Unit
Operating frequency			100 MHz		2.7 GHz	As shown
Attenuation range	0.5 dB Step			0 - 31.5		dB
Insertion Loss		0.1 GHz - 1.7 GHz		1.7	2	dB
		1.7 GHz - 2.7 GHz		1.8	3.2	dB
Attenuation Error	0 dB - 15.5 dB Attenuation Setting	0.1 GHz - 1.7 GHz				dB
		1.7 GHz - 2.7 GHz				dB
	16 dB - 31.5 dB Attenuation Setting	0.1 GHz - 1.5 GHz				dB
		1.5 GHz - 2.7 GHz				dB
Return Loss	Input port	0.1 GHz - 1.5 GHz		20		dB
		1.5 GHz - 2.7 GHz		15		dB
Return Loss	Output port	0.1 GHz - 2 GHz		20		dB
		2 GHz - 2.7 GHz		15		dB
Relative Phase	0 dB - 31.5 dB Attenuation Setting	100 MHz - 2.5 GHz		5.1		deg
Input P1 dB		100 MHz - 2.7 GHz	33	35		dBm
Input IP3	Two tones at +18 dBm, 1 MHz spacing		45		52	dBm

Table 4.4: Performance comparison of digital step attenuators

	[49]	[50]	[51]	[52]	This work (1)	This work (2)
Technology	GaAs	GaAs	SOS	SOS	SOI	SOI
Operating Frequency	0.7 – 3.8 (GHz)	DC – 3 (GHz)	0.01 – 4 (GHz)	0.01 – 2.5 (GHz)	0.1 – 2.7 (GHz)	0.1 – 2.7 (GHz)
Attenuation Range	31 dB 5-bit, LSB=1	31.5 dB 6-bit, LSB=0.5	31.5 dB 5-bit, LSB=0.5	31.5 dB 6-bit, LSB=0.5	31.5 dB 6-bit, LSB=0.5	31.75 dB 7-bit, LSB=0.25
Insertion Loss (dB)	1.8 – 3.5	3	1.3 – 2.3	1 – 1.9	1.7 – 3.2	1.7 – 3.2
Return Loss (dB)	>10	>12	>10	>14	>15	>15
Attenuation Accuracy (dB) max	$\pm(0.5+5\%$ Atten. Setting)	$\pm(0.5+5\%$ Atten. Setting)	$\pm(0.15+8\%$ Atten. Setting)	$\pm(0.15+8\%$ Atten. Setting)	$-(0.3+3\%$ Atten. Setting)	$-(0.15+3\%$ Atten. Setting)
$P_{0.1dB}$ (dBm)	26	27	30	30	33	33
Input IP3 (dBm)	45	+34	59	+55	45 – 52	44 – 51
Relative Phase Error	35°	25°	< 20°	< 20°	< 5.1°	< 6.2°

CHAPTER 5

CONCLUSION AND FUTURE WORK

Digital Step attenuators are used in many communication systems. In phased array systems, in order to create desired beam formations, the amplitude excitation and the phase excitation of each element in the array should be controlled accurately. In such systems, it is very important for the amplitude and the phase excitation values to be controlled independent of each other to simplify the design of the phased array system. Thus, the development of phase coherent digital step attenuators is crucial for communication systems. In this work, several methods were developed to decrease the phase shift problem of the DSAs.

Mathematical model of the phase coherent attenuator with the bond wires is presented. As well as a MATLAB algorithm based on the model is provided to calculate the initial values of the phase compensation elements.

This thesis presents 7-bit and 6-bit digital step attenuators that are compensated to achieve very low phase shift with respect to the attenuation and frequency changes. The proposed attenuators are fabricated on a commercial $0.18\mu\text{m}$ SOI process. The attenuators employ amplitude/phase compensation circuits that include switchable series inductors and shunt capacitors. In the proposed attenuators the digital control circuitry is integrated on the same die, reducing the package size, cost and complexity of the system. The fabricated chips are packaged in a commercial 4×4 mm QFN package and mounted on a test-fixture for measurements.

The proposed 7-bit and 6-bit digital step attenuators have high attenuation ranges of 31.75 dB and 31.5 dB with the high resolution of 0.25 dB and 0.5 dB steps, respec-

tively. In critical applications such as cellular code-division-multiple-access (CDMA) base-station transceivers, which rely on precise amplitude control, a minimum attenuation step of 0.5 dB may be required. Attenuators provide an MSB up to 16 dB, which is comparable with the best available attenuators in the industry. The capacitors, resistors, inductors and size of transistors are optimized for accurate attenuation and reasonable input and output return losses and linearity. The digital step attenuators exhibit amplitude and phase errors less than 1.2 dB and 6 degree in 0.1–2.5 GHz band, respectively. The phase shift is reduced to 5.1° and 6.2° and the rms phase imbalance is less than 3.1° and 3.6° for 6-bit and 7-bit attenuators, respectively. Return losses are higher than 15 dB. Moreover, the attenuators have high linearity with input third order intermodulation point (IIP_3) higher than 48 dBm and P_{1dB} compression point of 35 dBm, on par with commercial high phase shift attenuators.

As a summary, the proposed attenuators and phase compensation methods are novel and no similar work exists in the literature. It is believed that the results demonstrated throughout this study will be helpful for the future research on attenuators for industrial and commercial applications. Yet, further work could be done to improve some aspects.

Future work:

Although major research objectives are accomplished, there is still need for further research to improve the bandwidth and linearity performances of the attenuators. Some of the possible future studies can be listed as follows:

- 1) A digital attenuator's frequency range should be matched to the application of interest. When the frequency range of a component is broad, it can be used in many systems. This reduces the number of different components to be stocked by the user, and reduces the design time for new system, by eliminating or reducing the qualification requirement of a new component. The operating frequency of the presented attenuators are limited to 2.7 GHz. Moreover, measured IIP3 values are less than expected and simulated values. The reasons of these should be investigated in the future.

2) The proposed attenuators requires both positive and negative supply voltages. An internal negative voltage generator could be designed.

3) These attenuators only support parallel control mode. A serial peripheral interface (SPI) could be designed to have DSAs that can support both parallel and serial data communications.





REFERENCES

- [1] Seidel, E. "Progress on LTE Advanced - The future 4G standard". Nomor Research. Available online: <http://www.nomor.de/home/technology/white-papers/progress-on-lte-advanced---the-future-4g-standard>
Last access: 19-December-2015
- [2] Bleicer, A. "LTE-Advanced Is the Real 4G". IEEE Spectrum. Available online: <http://spectrum.ieee.org/telecom/standards/lte-advanced-is-the-real-4g>
Last access: 19-December-2015
- [3] "Hittite Microwave - HMC542ALP4, Digital Attenuators". Available online: <http://www.hittite.com/products/view.html/view/HMC542ALP4>
Last access: 20-September-2015
- [4] "Part Number PE43704 - Peregrine Semiconductor". Available online: <http://www.psemi.com/products/digital-step-attenuators-dsa/pe43704>
Last access: 20-September-2015
- [5] Walker, S., "A low phase shift attenuator," in *Microwave Theory and Techniques, IEEE Transactions on*, vol. 42, no. 2, pp. 182-185, Feb 1994.
- [6] D. Roques, J. Cazaux, and M. Pouysegur, "A new concept to cancel insertion phase variation in MMIC amplitude controller," in *IEEE Microw. and Millimeter-Wave Monolithic Circuits Symp. Dig., Dallas, TX*, , pp. 59-62, May. 1990.
- [7] H. Takasu, C. Sakakibara, M. Okumura, and S. Kamihashi, "S-band MMIC digital attenuator with small phase variation," in *Proc. IEEE Asia-Pacific Microw. Conf., Singapore*, , pp. 421-424, Dec. 1999.
- [8] L. Sjogren, D. Ingram, M. Biedenbender, R. Lai, B. Allen, and K. Hubbard, "A low phase-error 44-GHz HEMT attenuator," in *IEEE Microw. Guided Wave Lett.*, vol. 8, no. 5, pp. 194-195, May 1998.
- [9] B. Khabbaz, A. Pospishil, and H. P. Singh, "DC-to-20-GHz MMIC multibit digital attenuators with on-chip TTL control," in *IEEE J. Solid-State Circuits*, vol. 27, no. 10, pp. 1457-1462, Oct. 1992.
- [10] D. Krafcsik, F. Ali, and S. Bishop, "Broadband, low-loss 5- and 6-bit digital attenuators," in *IEEE MTT-S Int. Microw. Symp. Dig., Orlando, FL*, , pp. 1627-1630, May. 1995.

- [11] J. C. Sarkissian, M. Delmond, E. Laporte, E. Rogeaux, and R. Soulard, "A Ku-band 6-bit digital attenuator with integrated serial to parallel converter," in *IEEE MTT-S Int. Microw. Symp. Dig., Anaheim, CA*, pp. 1915–1918, Jun. 1999.
- [12] I.-K. Ju, Y.-S. Noh, and I.-B. Yom, "Ultra broadband DC to 40 GHz 5-bit pHEMT MMIC digital attenuator," in *Proc. Eur. Microw. Conf., Paris, France*, pp. 995–998, Oct. 2005.
- [13] U. L. Rohde, J. Whitaker, and T. T. N. Bucher, *Communications receivers : principles and design*. New York : McGraw-Hill, c1997., 1997.
- [14] J. Smith, *Modern communication circuits*. Boston, Mass : WCB/McGraw-Hill, c1998., 1998.
- [15] Suzuki, Y.; Ohira, T.; Ogawa, H., "Voltage-controlled biphas attenuator and vector synthesizer for monolithic microwave signal processors," in *Microwave Theory and Techniques, IEEE Transactions on* , , vol.46, no.11, pp.1982-1985, Nov 1998.
- [16] Komisarczuk, J.; Rychert, L., "High power variable attenuator with PIN diodes," in *Microwaves and Radar, 1998. MIKON '98., 12th International Conference on* , , vol.3, pp.843-847, 20-22 May 1998.
- [17] Saavedra, C.E.; Jackson, B.R., "Voltage-variable attenuator MMIC using phase cancellation," in *Circuits, Devices and Systems, IEE Proceedings* , , vol.153, no.5, pp.442-446, October 2006.
- [18] Sun, H.J.; Ewan, J., "A 2-18 GHz monolithic variable attenuator using novel triple-gate MESFETs," in *Microwave Symposium Digest, 1990., IEEE MTT-S International* , , pp.777-780 vol.2, 8-10 May 1990.
- [19] H. Dogan, R. G. Meyer, and A. M. Niknejad, "A DC-10GHz linear-in-dB attenuator in 0.13 m CMOS technology," in *Custom Integrated Circuits Conference, 2004. Proceedings of the IEEE 2004* , , pp. 609–612, 2014.
- [20] RFMD, "Voltage Controlled Attenuator," RFSA2013 datasheet.
- [21] E. Franzwa, A. Ellis, B. Nelson, M. Granger-Jones, and G. Valenti, "A 50MHz-16GHz low distortion SOI voltage controlled attenuator IC with IIP3 > +38dBm and control range of > 25dB," in *Microwave Symposium Digest (MTT), 2012 IEEE MTT-S International* , , 2012, pp. 1–3.
- [22] B. Maoz, "A novel, linear voltage variable MMIC attenuator," [6] *IEEE Trans. Microw. Theory Tech.* , , vol. 38, no. 11, pp. 1675–1683, Nov. 1990.
- [23] Maxim Integrated, "250MHz to 4000MHz Dual, Analog Voltage Variable Attenuator," MAX19790 datasheet.

- [24] Doddamani, N.D.; Harishchandra; Nandi, A.V., "Design of SPDT Switch, 6 Bit Digital Attenuator, 6 Bit Digital Phase Shifter for L-Band T/R Module using 0.7 M GaAs MMIC Technology," in *Signal Processing, Communications and Networking, 2007. ICSCN '07. International Conference on*, pp.302-307, 22-24 Feb. 2007.
- [25] Xing Wen; Fa-Xin Yu; Ling-Ling Sun, "A Novel Ultra Broadband 8-45 GHz 4-Bit GaAs Pseudomorphic High Electron Mobility Transistors (pHEMT) Monolithic Digital Attenuator Used for Gain Control of Transceivers," in *Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference on*, pp.519-522, 7-9 Dec. 2009.
- [26] T. Tsushima, H. Takeuchi, and M. Kimishima, "A 10 MHz-6 GHz high power high linearity 35 dB digital step attenuator MMIC using GaN HEMTs with TaON passivation," in *2015 IEEE International Symposium on Radio-Frequency Integration Technology (RFIT)*, , 2015, pp. 247–249.
- [27] Analog Devices, "0.5 dB LSB 6-Bit Digital Attenuator SMT, DC - 6 GHz," HMC624A datasheet.
- [28] J. K. Hunton and A. G. Ryals, "Microwave variable attenuators and modulators using PIN diodes," in *IEEE Trans. Microw. Theory Tech.* , vol. MTT-10, no. 7, pp. 262–273, Jul. 1962.
- [29] B.-W. Min and G. M. Rebeiz, "A 10-50-GHz CMOS distributed step attenuator with low loss and low phase imbalance," in *IEEE J. Solid-State Circuits* , vol. 42, no. 11, pp. 2547–2554, Nov. 2007.
- [30] Y. S. Dai, D. G. Fang, and Y.-X. Guo, "A novel UWB (0.045–50 GHz) digital/analog compatible MMIC variable attenuator with low insertion phase shift and large dynamic range," in *IEEE Microw. Wireless Compon. Lett.*, vol. 17, no. 1, pp. 61–63, Jan. 2007.
- [31] W. Ciccognani, F. Giannini, E. Limiti, and P. E. Longhi, "Compensating for parasitic phase shift in microwave digitally controlled attenuators," in *Electron. Lett.*, vol. 44, no. 12, pp. 743–744, Jun. 2008.
- [32] R. Baker, "CMOS-based digital step attenuator designs," in *Wireless Design Dev.*, pp. 46–50, May 2004.
- [33] Jeong, Jinho; Pornpromlikit, S.; Scuderi, Ant.; Presti, C.; Asbeck, P., "High Power Digitally- Controlled SOI CMOS Attenuator With Wide Attenuation Range," in *Microwave and Wireless Components Letters, IEEE* , vol.21, no.8, pp.433-435, Aug. 2011.
- [34] Bon-Hyun Ku; Songcheol Hong, "6-bit CMOS Digital Attenuators With Low Phase Variations for X -Band Phased-Array Systems," in *Microwave Theory and Techniques, IEEE Transactions on* , vol.58, no.7, pp.1651-1663, July 2010.

- [35] Moon-Kyu Cho, Jeong-Geun Kim, and Donghyun Baek, "A Broadband Digital Step Attenuator with Low Phase Error and Low Insertion Loss in 0.18- μ m SOI CMOS Technology" in *ETRI Journal*, vol. 35, No. 4, pp. 638-643, Aug. 2013.
- [36] Yong-Sheng Dai; Ping Li; Qun-Fei Han; Shao-Bo Chen; Xi Chen; Li-Jie Wang; Li Xu; Ri-Qing Chen, "Research on a novel 218 GHz PHEMT MMIC digital attenuator with low insertion phase shift," in *Microwave and Millimeter Wave Technology (ICMMT), 2012 International Conference on* , vol.4, pp. 1-4, 5-8 May 2012.
- [37] Wagner, J.; Mayer, U.; Wickert, M.; Wolf, R.; Joram, N.; Strobel, A.; Ellinger, F., "X-type attenuator in CMOS with novel control linearization, very low phase variations and automatic matching," in *Microwave Integrated Circuits Conference (EuMIC), 2013 European* , pp.200-203, 6-8 Oct. 2013.
- [38] Stukach, O.V., "Variable Attenuator with Low Phase Shift," in *Wireless Technology, 2006. The 9th European Conference on*., pp. 241-244, 10-12 Sept. 2006.
- [39] O. V. Stoukatch and L. P. Ligthart, "Attenuator with Small Phase Shift for Ultra-Wideband Pulse Subsurface Geo-Radar," in *Microwave Conference, 2002. 32nd European, Milan, Italy*., pp.1-3, 23-26 Sept. 2002.
- [40] H. Dogan, R. G. Meyer and A. M. Niknejad, "Analysis and Design of RF CMOS Attenuators," in *IEEE Journal of Solid-State Circuits* , vol. 43, no. 10, pp. 2269-2283, Oct. 2008.
- [41] Available online: <http://www.microwaves101.com/calculators/858-attenuator-calculator>
Last access: 10-July-2016
- [42] Available online: <http://www.microwavejournal.com/articles/print/10269-the-state-of-rf-and-microwave-switches>
Last access: 20-August-2016
- [43] Available online: https://www.components.omron.com/components/web/pdflib.nsf/0/64722CE7CCE28EAA86257855006F064B/file/StateofRFSwitchwDiags_Nov2010.pdf
Last access: 20-August-2016
- [44] Burgener, Mark L., and James S. Cable. "Switch circuit and method of switching radio frequency signals." U.S. Patent No. 7,123,898. 17 Oct. 2006.
- [45] Hyunchul Eom; Kyoungsoon Yang, "A 6-20 GHz compact multi-bit digital attenuator using InP/InGaAs PIN Diodes," in *Indium Phosphide and Related Materials, 2008. IPRM 2008. 20th International Conference on* , pp.1-3, 25-29 May 2008.

- [46] Zhao, Li; Liang, Wen-Feng; Xu, Xiao-Jie; Jiang, Xin; Zhou, Jian-Yi, "An integrated Q-band 6-bit digital attenuator with low insertion loss," in *Microwave Conference (APMC), 2014 Asia-Pacific*, , pp.1196-1198, 4-7 Nov. 2014.
- [47] Hangai, M.; Asao, H.; Hieda, M.; Yamaguchi, M.; Miyazaki, M., "Amplitude/Phase Temperature Compensation Attenuators With Variable- Q FET Resonators," in *Microwave Theory and Techniques, IEEE Transactions on*,, vol.56, no.12, pp.3058-3065, Dec. 2008.
- [48] P. Sun, "Analysis of phase variation of CMOS digital attenuator," in *Electronics Letters*, , vol. 50, no. 25, pp. 1912-1914, 12 4 2014.
- [49] "Analog Devices - HMC273A, Digital Step Attenuators". Available online: <http://www.analog.com/en/products/rf-microwave/attenuators/digital-step-attenuators/hmc273a.html>
Last access: 25-August-2016
- [50] "Analog Devices - HMC424AG16, Digital Step Attenuators". Available online: <http://www.analog.com/en/products/rf-microwave/attenuators/digital-step-attenuators/hmc424ag16.html>
Last access: 25-August-2016
- [51] "Peregrine Semiconductor – PE4312, Digital Step Attenuators". Available online: <http://www.psemi.com/products/digital-step-attenuators-dsa/pe4312>
Last access: 25-August-2016
- [52] "Peregrine Semiconductor – PE4314, Digital Step Attenuators". Available online: <http://www.psemi.com/products/digital-step-attenuators-dsa/pe4314>
Last access: 25-August-2016



APPENDIX A

AUTOMATED MATLAB ALGORITHM

The utilized MATLAB algorithm with graphical user interface (GUI) to calculate the initial values of the compensation elements of the phase coherent digital step attenuator is as below:

```
function varargout = MatCalculator(varargin)

% MATCALCULATOR MATLAB code for MatCalculator.fig
% MATCALCULATOR, by itself, creates a new MATCALCULATOR or raises
the existing
%singleton*.
%
% H = MATCALCULATOR returns the handle to a new MATCALCULATOR
or the handle to the existing singleton*.
%
% MATCALCULATOR('CALLBACK',hObject,eventData,handles,...)
calls the local
% function named CALLBACK in MATCALCULATOR.M with the given
input arguments.
%
% MATCALCULATOR('Property','Value',...) creates a new
MATCALCULATOR or raises
% the existing singleton*. Starting from the left, property
value pairs are
% applied to the GUI before MatCalculator_OpeningFcn gets
called. An
% unrecognized property name or invalid value makes
property application
% stop. All inputs are passed to MatCalculator_OpeningFcn
via varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI
allows only one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help
MatCalculator
% Last Modified by GUIDE v2.5 12-Jul-2016 15:11:05
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn',  @MatCalculator_OpeningFcn, ...
                  'gui_OutputFcn',   @MatCalculator_OutputFcn, ...
                  'gui_LayoutFcn',   [], ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
```

```

        gui_State.gui_Callback = str2func(varargin{1});
    end
    if nargout
        [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
    else
        gui_mainfcn(gui_State, varargin{:});
    end
% End initialization code - DO NOT EDIT
% --- Executes just before MatCalculator is made visible.
function MatCalculator_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles     structure with handles and user data (see
GUIDATA)
% varargin    command line arguments to MatCalculator (see
VARARGIN)
% Choose default command line output for MatCalculator
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes MatCalculator wait for user response (see
UIRESUME)
% uiwait(handles.MathCalculator);
% --- Outputs from this function are returned to the command
line.
function varargout = MatCalculator_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles     structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;

function R1Edit_Callback(hObject, eventdata, handles)
% hObject    handle to R1Edit (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles     structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of R1Edit as
text
%         str2double(get(hObject,'String')) returns contents of
R1Edit as a double
% --- Executes during object creation, after setting all
properties.
function R1Edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to R1Edit (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles empty - handles not created until after all
CreateFcns called
% Hint: edit controls usually have a white background on
Windows.

```

```

%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function R2Edit_Callback(hObject, eventdata, handles)
% hObject      handle to R2Edit (see GCBO)
% eventdata    reserved - to be defined in a future version of
MATLAB
% handles      structure with handles and user data (see
GUIDATA)
% Hints: get(hObject,'String') returns contents of R2Edit as
text
%         str2double(get(hObject,'String')) returns contents of
R2Edit as a double
% --- Executes during object creation, after setting all
properties.

function R2Edit_CreateFcn(hObject, eventdata, handles)

% hObject      handle to R2Edit (see GCBO)
% eventdata    reserved - to be defined in a future version of
MATLAB
% handles      empty - handles not created until after all
CreateFcns called
% Hint: edit controls usually have a white background on
Windows.
%         See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function RonEdit_Callback(hObject, eventdata, handles)
% hObject      handle to RonEdit (see GCBO)
% eventdata    reserved - to be defined in a future version of
MATLAB
% handles      structure with handles and user data (see
GUIDATA)
% Hints: get(hObject,'String') returns contents of RonEdit as
text
%         str2double(get(hObject,'String')) returns contents of
RonEdit as a double
% --- Executes during object creation, after setting all
properties.

function RonEdit_CreateFcn(hObject, eventdata, handles)

% hObject      handle to RonEdit (see GCBO)
% eventdata    reserved - to be defined in a future version of
MATLAB
% handles      empty - handles not created until after all
CreateFcns called
% Hint: edit controls usually have a white background on
Windows.
%         See ISPC and COMPUTER.

```

```

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function CoffEdit_Callback(hObject, eventdata, handles)

% hObject      handle to CoffEdit (see GCBO)
% eventdata    reserved - to be defined in a future version of
MATLAB
% handles      structure with handles and user data (see
GUIDATA)
% Hints: get(hObject,'String') returns contents of CoffEdit as
text
%      str2double(get(hObject,'String')) returns contents of
CoffEdit as a double
% --- Executes during object creation, after setting all
properties.

function CoffEdit_CreateFcn(hObject, eventdata, handles)

% hObject handle to CoffEdit (see GCBO)
% eventdata    reserved - to be defined in a future version of
MATLAB
% handles empty - handles not created until after all
CreateFcns called
% Hint: edit controls usually have a white background on
Windows.
%      See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function CcompEdit_Callback(hObject, eventdata, handles)
% hObject      handle to CcompEdit (see GCBO)
% eventdata    reserved - to be defined in a future version of
MATLAB
% handles      structure with handles and user data (see
GUIDATA)
% Hints: get(hObject,'String') returns contents of CcompEdit
as text
%      str2double(get(hObject,'String')) returns contents of
CcompEdit as a double
% --- Executes during object creation, after setting all
properties.

function CcompEdit_CreateFcn(hObject, eventdata, handles)

% hObject      handle to CcompEdit (see GCBO)
% eventdata    reserved - to be defined in a future version of
MATLAB
% handles      empty - handles not created until after all
CreateFcns called
% Hint: edit controls usually have a white background on
Windows.
%      See ISPC and COMPUTER.

```

```

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function CpEdit_Callback(hObject, eventdata, handles)
% hObject      handle to CpEdit (see GCBO)
% eventdata    reserved - to be defined in a future version of
MATLAB
% handles      structure with handles and user data (see
GUIDATA)
% Hints: get(hObject,'String') returns contents of CpEdit as
text
%            str2double(get(hObject,'String')) returns contents of
CpEdit as a double
% --- Executes during object creation, after setting all
properties.

function CpEdit_CreateFcn(hObject, eventdata, handles)

% hObject      handle to CpEdit (see GCBO)
% eventdata    reserved - to be defined in a future version of
MATLAB
% handles      empty - handles not created until after all
CreateFcns called
% Hint: edit controls usually have a white background on
Windows.
%            See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function L1Edit_Callback(hObject, eventdata, handles)

% hObject      handle to L1Edit (see GCBO)
% eventdata    reserved - to be defined in a future version of
MATLAB
% handles      structure with handles and user data (see
GUIDATA)
% Hints: get(hObject,'String') returns contents of L1Edit as
text
%            str2double(get(hObject,'String')) returns contents of
L1Edit as a double
% --- Executes during object creation, after setting all
properties.
function L1Edit_CreateFcn(hObject, eventdata, handles)
% hObject      handle to L1Edit (see GCBO)
% eventdata    reserved - to be defined in a future version of
MATLAB
% handles      empty - handles not created until after all
CreateFcns called
% Hint: edit controls usually have a white background on
Windows.
%            See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))

```

```

        set(hObject,'BackgroundColor','white');
end

function L2Edit_Callback(hObject, eventdata, handles)

% hObject      handle to L2Edit (see GCBO)
% eventdata    reserved - to be defined in a future version of
MATLAB
% handles      structure with handles and user data (see
GUIDATA)
% Hints: get(hObject,'String') returns contents of L2Edit as
text
%            str2double(get(hObject,'String')) returns contents of
L2Edit as a double
% --- Executes during object creation, after setting all
properties.
function L2Edit_CreateFcn(hObject, eventdata, handles)
% hObject      handle to L2Edit (see GCBO)
% eventdata    reserved - to be defined in a future version of
MATLAB
% handles      empty - handles not created until after all
CreateFcns called
% Hint: edit controls usually have a white background on
Windows.
%            See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in CalculateButton.
function CalculateButton_Callback(hObject, eventdata, handles)
% hObject      handle to CalculateButton (see GCBO)
% eventdata    reserved - to be defined in a future version of
MATLAB
% handles      structure with handles and user data (see
GUIDATA)
%%
clc
syms  f w S Coff C2 Cp Ron R1 R2 Zo Yo L L1 L2 Lg

f=1e9:.01e9:6e9;
%f=4*10^9;
w=2*pi*f;
S=i*w;
Zo=50;
Yo=0.02;
L=900*10^-12;
Lg=400*10^-12;

Coff=str2double(get(handles.CoffEdit , 'String'));
Ccomp=str2double(get(handles.CcompEdit , 'String'));
Cp=str2double(get(handles.CpEdit , 'String'));
L1=str2double(get(handles.L1Edit , 'String'));
L2=str2double(get(handles.L2Edit , 'String'));
Ron=str2double(get(handles.RonEdit , 'String'));
R1=str2double(get(handles.R1Edit , 'String'));

```

```

R2=str2double(get(handles.R2Edit , 'String'));
% figure(1);
% hold on
% grid on
ZL2= S.*L2;
Yp_ref=((Cp + Coff + S.^2.*Lg.*Cp.*Coff)./(S.*Coff.*Cp)).^-1;
Yp_att=((1 + S.*Cp.*Ron + S.^2.*Lg.*Cp)./(S.*Cp)).^-1;
X1_ref=((Ron+(S.*L1)).*R1)./(R1+(Ron+(S.*L1)));
X1_att=R1./(1+(S.*R1.*Coff));
X2_ref=((S.*Ccomp.*R2 + S.*R2.*Coff + 1)./((S.*Ccomp.*R2
+1).*S.*Coff)) + S.*Lg;
X2_att=((S.*Ccomp.*R2.*Ron + R2 + Ron)./(S.*Ccomp.*R2 + 1))+
S.*Lg;
DeltaZ_ref_N2=Zo.^2 + 2.*Zo.*X2_ref;
DeltaZ_att_N2=Zo.^2 + 2.*Zo.*X2_att;
Y11_ref_N2=(Zo + X2_ref)./DeltaZ_ref_N2;
Y11_att_N2=(Zo + X2_att)./DeltaZ_att_N2;
Y21_ref_N2=(-1 .* X2_ref)./DeltaZ_ref_N2;
Y21_att_N2=(-1 .* X2_att)./DeltaZ_att_N2;
Y11_ref_N1=1./X1_ref;
Y11_att_N1=1./X1_att;
Y21_ref_N1=-1./X1_ref;
Y21_att_N1=-1./X1_att;
Y11eq_ref=Y11_ref_N1 + Y11_ref_N2;
Y11eq_att=Y11_att_N1 + Y11_att_N2;
Y21eq_ref=Y21_ref_N1 + Y21_ref_N2;
Y21eq_att=Y21_att_N1 + Y21_att_N2;
DY_ref=(Y11eq_ref + Yo).^2 - Y21eq_ref.^2;
DY_att=(Y11eq_att + Yo).^2 - Y21eq_att.^2;
A2_D2_ref=(-1.*Y11eq_ref)./Y21eq_ref;
A2_D2_att=(-1.*Y11eq_att)./Y21eq_att;
B2_ref=-1./Y21eq_ref;
B2_att=-1./Y21eq_att;
Y_ref=Y11eq_ref.^2 - Y21eq_ref.^2;
Y_att=Y11eq_att.^2 - Y21eq_att.^2;
C2_ref=-1.*Y_ref./Y21eq_ref;
C2_att=-1.*Y_att./Y21eq_att;
A123_ref=2.*S.*L.*C2_ref + A2_D2_ref;
A123_att=2.*S.*L.*C2_att + A2_D2_att;
B123_ref=2.*S.*L.*A2_D2_ref + B2_ref;
B123_att=2.*S.*L.*A2_D2_att + B2_att;
C123_ref=C2_ref;
C123_att=C2_att;
D123_ref=A2_D2_ref;
D123_att=A2_D2_att;
A1234_ref= A123_ref;
A1234_att= A123_att;
B1234_ref= B123_ref ;
B1234_att= B123_att + ZL2.*A123_att;
C1234_ref= C123_ref;
C1234_att= C123_att;
D1234_ref= D123_ref;
D1234_att= D123_att + ZL2.*C123_att;
A_ref= A1234_ref + Yp_ref.*B123_ref;
A_att= A1234_att + Yp_att.*B123_att;
B_ref= B1234_ref ;
B_att= B1234_att ;
C_ref= C1234_ref + Yp_ref.*D123_ref;
C_att= C1234_att + Yp_att.*D123_att;

```

```

D_ref= D1234_ref ;
D_att= D1234_att ;

S21_ref=2./(A_ref + B_ref./Zo + C_ref.*Zo + D_ref);
S21_att=2./(A_att + B_att./Zo + C_att.*Zo + D_att);
S21dBref=20.*log10(abs(S21_ref));
S21dBatt=20.*log10(abs(S21_att));
S21degref=angle(S21_ref).*180/pi;
S21degatt=angle(S21_att).*180/pi;
DeltaS21dB=(S21dBatt - S21dBref);
DeltaS21deg=S21degatt - S21degref;

cla(handles.AttenAxes , 'reset');
set(handles.AttenAxes , 'Visible' , 'on');
axes(handles.AttenAxes);
hold on;
plot(f , S21dBref , 'LineWidth' , 2);
plot(f , S21dBatt , 'LineWidth' , 2);
hold off;
%title('Attenuation');
xlabel('Frequency [Hz]');
ylabel('Relative Attenuation [dB]');
legend('Reference' , 'Attenuation');
grid on;

cla(handles.DeltaAttenAxes , 'reset');
set(handles.DeltaAttenAxes , 'Visible' , 'on');
axes(handles.DeltaAttenAxes);
plot(f , DeltaS21dB , 'g' , 'LineWidth' , 2);
%title('Attenuation');
xlabel('Frequency [Hz]');
ylabel('Attenuation [dB]');
grid on;

cla(handles.PhaseAxes , 'reset');
set(handles.PhaseAxes , 'Visible' , 'on');
axes(handles.PhaseAxes);
hold on;
plot(f , S21degref , 'LineWidth' , 2);
plot(f , S21degatt , 'LineWidth' , 2);
hold off;
%title('Phase');
xlabel('Frequency [Hz]');
ylabel('Transmission Phase [Deg.]');
legend('Reference' , 'Attenuation');
grid on;

cla(handles.DeltaPhaseAxes , 'reset');
set(handles.DeltaPhaseAxes , 'Visible' , 'on');
axes(handles.DeltaPhaseAxes);
plot(f , DeltaS21deg , 'g' , 'LineWidth' , 2);
%title('Phase');
xlabel('Frequency [Hz]');
ylabel('Phase Difference [Deg.]');
grid on;

%%

```

APPENDIX B

ARDUINO CONTROL CODE

Below Arduino Mega code is utilized to generate binary codes for control the attenuators. Outputs of the Arduino Mega respect to the command that coming from computer (connected to the network analyzer) can be zero or one digital codes.

The utilized Arduino mega code for generate

```
char receivedChar;

int state;    // used for HIGH or LOW

byte pins[] = {34, 32, 30, 28, 26, 24, 22};

const byte numPins = 7; // how many bits needed for the control

char receivedNumberInBinary[numPins];

int pinNo;

void setup()
{
    // initialize serial communication at 9600 bits per second:

    Serial.begin(9600);           /* we setup all led pins as OUTPUT */

    for(int i = 0; i < numPins; i++) {
        pinMode(pins[i], OUTPUT);
    }
}

void loop()
{
    //

    receivedChar = Serial.read();

    if(receivedChar == 'a') {
        receivedChar = Serial.read();

        while(receivedChar == -1) {
            receivedChar = Serial.read();
        }
    }
}
```

```

}

pinNo = 0;

while(receivedChar != 'b') {

    receivedNumberInBinary[pinNo] = receivedChar;

    pinNo = pinNo + 1;

    receivedChar = Serial.read();

    while(receivedChar == -1) {

        receivedChar = Serial.read();

    }

}

/* convert presses to binary and store it as a string */
String binNumber(receivedNumberInBinary);    /* get the length of the string
*/

int binLength = binNumber.length();

for(int i = 0, x = 1; i < binLength; i++, x+=2) {

    if(binNumber[i] == '0') state = LOW;

    if(binNumber[i] == '1') state = HIGH;

    digitalWrite(pins[i], state);

}    //Serial.write(receivedNumberInBinary);

}

if(receivedChar == 'c') {

    Serial.write("d", 1);

    Serial.write(receivedNumberInBinary, numPins);

    Serial.write("e", 1);

}

delay(1);    // delay for stability

}

```

APPENDIX C

C# CODE IN VISUAL STUDIO

A C# code was written in visual studio. This code sends a command to Arduino. Arduino generates the binary code to select the required attenuation state and S-parameters data from network analyzer saves in excel file.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

// For Arduino
using System.IO.Ports;
using System.Threading;
using System.Threading.Tasks;

// For Agilent
using Ivi.Driver.Interop;
using Agilent.AgilentNA.Interop;

// For Excel
using Excel = Microsoft.Office.Interop.Excel;

namespace Data_Collector_for_Arash
{
    public partial class Form1 : Form
    {
        int trigms = 2000;

        bool otherTabForArduino = false;
        bool otherTabForAgilent = false;

        // For Arduino
        SerialPort port;
        string[] portnames;

        char[] inputRead = new char[7];
        char[] inputWrite;

        string ArduinoSentData;
        string ArduinoSentData1;

        // For Agilent
        AgilentNA driver = null;
```

```

        string resourceDesc =
"USB0::0x0957::0x0D09::MY46107373::0::INSTR";
        string initOptions = "QueryInstrStatus=true,
Simulate=false, DriverSetup= Model=E5071C, Trace=false,
TraceName=C:\\temp\\traceOut";

        bool idquery = true;
        bool reset = false;

        public Form1()
        {
            InitializeComponent();

            //Disables the other tabs
            (tabControl1.TabPages[1] as Control).Enabled =
false;
            (tabControl1.TabPages[2] as Control).Enabled =
false;

            //Checks if a serial port is connected (for
Arduino)
            portnames = SerialPort.GetPortNames();

            for (int i = 0; i < portnames.Length; i++)
            {
                comboBox1.Items.Add(portnames[i]);
            }

            comboBox1.Items.Add("Simulate");

            comboBox1.DropDownStyle =
ComboBoxStyle.DropDownList;
            comboBox2.DropDownStyle =
ComboBoxStyle.DropDownList;
            comboBox3.DropDownStyle =
ComboBoxStyle.DropDownList;
            comboBox4.DropDownStyle =
ComboBoxStyle.DropDownList;
            comboBox5.DropDownStyle =
ComboBoxStyle.DropDownList;
            comboBox6.DropDownStyle =
ComboBoxStyle.DropDownList;
            comboBox7.DropDownStyle =
ComboBoxStyle.DropDownList;

            //if (portnames.Length == 0)
            //{
            //    button1.Enabled = false;
            //    comboBox1.Enabled = false;
            //    textBox2.Enabled = false;
            //    textBox3.Enabled = false;
            //}
            //

        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (comboBox1.Text != "Simulate")
            {

```

```

        port = new SerialPort(comboBox1.Text,
Convert.ToInt32(textBox2.Text), Parity.None, 8, StopBits.One);

        port.Open();

        if (port.IsOpen)
        {
            textBox3.Text = "Successful";

            otherTabForArduino = true;

            (tabControl1.TabPages[1] as
Control).Enabled = otherTabForArduino && otherTabForAgilent;
            (tabControl1.TabPages[2] as
Control).Enabled = otherTabForArduino && otherTabForAgilent;
        }
        else
        {
            textBox3.Text = "Error";
        }
    }
    else
    {
        textBox3.Text = "Simulate";

        otherTabForArduino = true;

        (tabControl1.TabPages[1] as Control).Enabled =
otherTabForArduino && otherTabForAgilent;
        (tabControl1.TabPages[2] as Control).Enabled =
otherTabForArduino && otherTabForAgilent;
    }
}

private void extBtn_Click(object sender, EventArgs e)
{
    if (driver != null && driver.Initialized)
    {
        // Close the driver
        driver.Close();
        System.Windows.Forms.MessageBox.Show("Agilent
Driver Closed");
    }

    if (System.Windows.Forms.Application.MessageLoop)
    {
        // WinForms app
        System.Windows.Forms.Application.Exit();
    }
    else
    {
        // Console app
        System.Environment.Exit(1);
    }
}

private void button2_Click(object sender, EventArgs e)
{
    if (driver != null && driver.Initialized)

```

```

    {
        // Close the driver
        driver.Close();
    }

    try
    {
        driver = new AgilentNA();

        resourceDesc = textBox1.Text;

        reset = checkBox1.Checked;

        if (checkBox2.Checked)
        {
            initOptions = "QueryInstrStatus=true,
Simulate=true, DriverSetup= Model=E5071C, Trace=false,
TraceName=C:\\temp\\traceOut";
        }
        else
        {
            initOptions = "QueryInstrStatus=true,
Simulate=false, DriverSetup= Model=E5071C, Trace=false,
TraceName=C:\\temp\\traceOut";
        }

        // Initialize the driver. See driver help
topic "Initializing the IVI-COM Driver" for additional
information
        driver.Initialize(resourceDesc, idquery,
reset, initOptions);

        // Print a few IIviDriverIdentity properties
        textBox4.Text = "";
        textBox4.AppendText("Driver Initialized\r\n");
        textBox4.AppendText("\r\nIdentifier: " +
driver.Identity.Identifier);
        textBox4.AppendText("\r\nRevision: " +
driver.Identity.Revision);
        textBox4.AppendText("\r\nVendor: " +
driver.Identity.Vendor);
        textBox4.AppendText("\r\nDescription: " +
driver.Identity.Description);
        textBox4.AppendText("\r\nModel: " +
driver.Identity.InstrumentModel);
        textBox4.AppendText("\r\nFirmwareRev: " +
driver.Identity.InstrumentFirmwareRevision);
        textBox4.AppendText("\r\nSerial #: " +
driver.System.SerialNumber);
        textBox4.AppendText("\r\nSimulate: " +
driver.DriverOperation.Simulate);

        otherTabForAgilent = true;

        (tabControl1.TabPages[1] as Control).Enabled =
otherTabForArduino && otherTabForAgilent;
        (tabControl1.TabPages[2] as
Control).Enabled = otherTabForArduino &&
otherTabForAgilent;
    }
}

```

```

    }
    catch (Exception ex)
    {
        System.Windows.Forms.MessageBox.Show(ex.Message);
    }
}

private void button3_Click(object sender, EventArgs e)
{
    string tempSS;
    bool errorBit = false;

    if (textBox3.Text != "Simulate")
    {
        if (comboBox2.Text == "6")
        {
            tempSS = 'a' + textBox6.Text + '0' + 'b';
            inputWrite = tempSS.ToCharArray();

            ArduinoSentData = textBox6.Text + '0';
        }
        else if (comboBox2.Text == "5")
        {
            tempSS = 'a' + textBox6.Text + "00" + 'b';
            inputWrite = tempSS.ToCharArray();

            ArduinoSentData = textBox6.Text + "00";
        }
        else
        {
            tempSS = 'a' + textBox6.Text + 'b';
            inputWrite = tempSS.ToCharArray();

            ArduinoSentData = textBox6.Text;
        }

        if (textBox6.Text.Length !=
Convert.ToInt32(comboBox2.Text))
        {
            errorBit = true;
        }

        if (!errorBit)
        {
            port.Write(inputWrite, 0, 9);

            // Arduino data check
            checkSentToArduino();
            checkSentToArduino();
            checkSentToArduino();
        }
        else
        {
            textBox5.Text = "Error: Bit Number not match";
        }
    }
}

```

```

        else
        {
            if (comboBox2.Text == "6")
            {
                tempSS = 'a' + textBox6.Text + '0' + 'b';
                inputWrite = tempSS.ToCharArray();

                ArduinoSentData = textBox6.Text + '0';
            }
            else if (comboBox2.Text == "5")
            {
                tempSS = 'a' + textBox6.Text + "00" + 'b';
                inputWrite = tempSS.ToCharArray();

                ArduinoSentData = textBox6.Text + "00";
            }
            else
            {
                tempSS = 'a' + textBox6.Text + 'b';
                inputWrite = tempSS.ToCharArray();

                ArduinoSentData = textBox6.Text;
            }

            textBox5.Text = "Successful [Simulate]";
        }
    }
    private void checkSentToArduino()
    {
        string tempSS = "c";
        inputWrite = tempSS.ToCharArray();

        port.Write(inputWrite, 0, 1);

        if (port.ReadChar() != 'd')
        {
            textBox5.Text = "Error";
        }

        char tempc;
        tempc = Convert.ToChar(port.ReadChar());
        inputRead[0] = tempc;

        if (!(tempc == '1' || tempc == '0'))
        {
            textBox5.Text = "Error";
        }

        tempc = Convert.ToChar(port.ReadChar());
        inputRead[1] = tempc;

        if (!(tempc == '1' || tempc == '0'))
        {
            textBox5.Text = "Error";
        }

        tempc = Convert.ToChar(port.ReadChar());
        inputRead[2] = tempc;

        if (!(tempc == '1' || tempc == '0'))

```

```

{
    textBox5.Text = "Error";
}

tempc = Convert.ToChar(port.ReadChar());
inputRead[3] = tempc;

if (!(tempc == '1' || tempc == '0'))
{
    textBox5.Text = "Error";
}

tempc = Convert.ToChar(port.ReadChar());
inputRead[4] = tempc;

if (!(tempc == '1' || tempc == '0'))
{
    textBox5.Text = "Error";
}

tempc = Convert.ToChar(port.ReadChar());
inputRead[5] = tempc;

if (!(tempc == '1' || tempc == '0'))
{
    textBox5.Text = "Error";
}

tempc = Convert.ToChar(port.ReadChar());
inputRead[6] = tempc;

if (!(tempc == '1' || tempc == '0'))
{
    textBox5.Text = "Error";
}

if (port.ReadChar() != 'e')
{
    textBox5.Text = "Error";
}

string tempS = new string(inputRead);

if (ArduinoSentData != tempS)
{
    textBox5.Text = "Error";
}
else
{
    textBox5.Text = "Successful";
}
}

private void button4_Click(object sender, EventArgs e)
{
    try
    {
        trigms = Convert.ToInt32(textBox11.Text);

        // Excel Offsets
    }
    catch { }
}

```

```

        int xFreqOffset = 1;
        int yFreqOffset = 2;

        int xDataOffset = 2;
        int yDataOffset = 2;

        // Excel App Opened

        Excel.Application xlApp = new
Microsoft.Office.Interop.Excel.Application();

        if (xlApp == null)
        {
            MessageBox.Show("Excel is not properly
installed!!");
            return;
        }
        // Excel Files are created

        //S11_LogMag
        Excel.Workbook xlWorkbookS11_LogMag;
        Excel.Worksheet xlWorkSheetS11_LogMag;
        object misValueS11_LogMag =
System.Reflection.Missing.Value;

        xlWorkbookS11_LogMag =
xlApp.Workbooks.Add(misValueS11_LogMag);
        xlWorkSheetS11_LogMag =
(Excel.Worksheet)xlWorkbookS11_LogMag.Worksheets.get_Item(1);

        //S12_LogMag
        Excel.Workbook xlWorkbookS12_LogMag;
        Excel.Worksheet xlWorkSheetS12_LogMag;
        object misValueS12_LogMag =
System.Reflection.Missing.Value;

        xlWorkbookS12_LogMag =
xlApp.Workbooks.Add(misValueS12_LogMag);
        xlWorkSheetS12_LogMag =
(Excel.Worksheet)xlWorkbookS12_LogMag.Worksheets.get_Item(1);

        //S21_LogMag
        Excel.Workbook xlWorkbookS21_LogMag;
        Excel.Worksheet xlWorkSheetS21_LogMag;
        object misValueS21_LogMag =
System.Reflection.Missing.Value;

        xlWorkbookS21_LogMag =
xlApp.Workbooks.Add(misValueS21_LogMag);
        xlWorkSheetS21_LogMag =
(Excel.Worksheet)xlWorkbookS21_LogMag.Worksheets.get_Item(1);

        //S22_LogMag
        Excel.Workbook xlWorkbookS22_LogMag;
        Excel.Worksheet xlWorkSheetS22_LogMag;
        object misValueS22_LogMag =
System.Reflection.Missing.Value;

```

```

        xlWorkBookS22_LogMag =
xlApp.Workbooks.Add(misValueS22_LogMag);
        xlWorksheetsS22_LogMag =
(Excel.Worksheet)xlWorkBookS22_LogMag.Worksheets.get_Item(1);
        //S11_Phase
        Excel.Workbook xlWorkBookS11_Phase;
        Excel.Worksheet xlWorksheetS11_Phase;
        object misValueS11_Phase =
System.Reflection.Missing.Value;

        xlWorkBookS11_Phase =
xlApp.Workbooks.Add(misValueS11_Phase);
        xlWorksheetsS11_Phase =
(Excel.Worksheet)xlWorkBookS11_Phase.Worksheets.get_Item(1);
        //S12_Phase
        Excel.Workbook xlWorkBookS12_Phase;
        Excel.Worksheet xlWorksheetS12_Phase;
        object misValueS12_Phase =
System.Reflection.Missing.Value;

        xlWorkBookS12_Phase =
xlApp.Workbooks.Add(misValueS12_Phase);
        xlWorksheetsS12_Phase =
(Excel.Worksheet)xlWorkBookS12_Phase.Worksheets.get_Item(1);
        //S21_Phase
        Excel.Workbook xlWorkBookS21_Phase;
        Excel.Worksheet xlWorksheetS21_Phase;
        object misValueS21_Phase =
System.Reflection.Missing.Value;

        xlWorkBookS21_Phase =
xlApp.Workbooks.Add(misValueS21_Phase);
        xlWorksheetsS21_Phase =
(Excel.Worksheet)xlWorkBookS21_Phase.Worksheets.get_Item(1);
        //S22_Phase
        Excel.Workbook xlWorkBookS22_Phase;
        Excel.Worksheet xlWorksheetS22_Phase;
        object misValueS22_Phase =
System.Reflection.Missing.Value;

        xlWorkBookS22_Phase =
xlApp.Workbooks.Add(misValueS22_Phase);
        xlWorksheetsS22_Phase =
(Excel.Worksheet)xlWorkBookS22_Phase.Worksheets.get_Item(1);

        // Message box cleared
        textBox10.Text = "";

        // Determine current date and time
        string dateTimeNow =
DateTime.Now.ToString("yyyyMMdd_HH:mm:ss");

        //Creates measurements folder
        string folderPath = @"C:\Measurements\" +
dateTimeNow + @"_" + comboBox2.Text + @"_" + trigs + @"ms" +
@"_";

        if (comboBox3.Text == "GHz")
        {

```

```

        folderPath = folderPath +
Convert.ToDouble(Convert.ToDouble(textBox7.Text) *
Math.Pow(10, 9)) + @"_";
    }
    else if (comboBox3.Text == "MHz")
    {
        folderPath = folderPath +
Convert.ToDouble(Convert.ToDouble(textBox7.Text) *
Math.Pow(10, 6)) + @"_";
    }
    else
    {
        folderPath = folderPath +
Convert.ToDouble(textBox7.Text) + @"_";
    }

    if (comboBox4.Text == "GHz")
    {
        folderPath = folderPath +
Convert.ToDouble(Convert.ToDouble(textBox8.Text) *
Math.Pow(10, 9)) + @"_";
    }
    else if (comboBox4.Text == "MHz")
    {
        folderPath = folderPath +
Convert.ToDouble(Convert.ToDouble(textBox8.Text) *
Math.Pow(10, 6)) + @"_";
    }
    else
    {
        folderPath = folderPath +
Convert.ToDouble(textBox8.Text) + @"_";
    }

    folderPath = folderPath + textBox9.Text +
@"_Single_";

    folderPath = folderPath + textBox6.Text;

System.IO.Directory.CreateDirectory(folderPath);

// Setup a measurement and output data
double[] data0 = null;
double[] freq0 = null;

double[] data1 = null;
double[] freq1 = null;

double[] data2 = null;
double[] freq2 = null;

double[] data3 = null;
double[] freq3 = null;

double[] data4 = null;
double[] freq4 = null;

```

```

        double[] data5 = null;
        double[] freq5 = null;

        double[] data6 = null;
        double[] freq6 = null;

        double[] data7 = null;
        double[] freq7 = null;

        int points;

// Setup convinient pointers to Channels and Measurements

        IAgilentNChannel Ch1 =
driver.Channels.get_Item("Channel1");
        IAgilentNMeasurement Ch1S11 =
Ch1.Measurements.get_Item("Measurement1");
        IAgilentNMeasurement Ch1S12 =
Ch1.Measurements.get_Item("Measurement2");
        IAgilentNMeasurement Ch1S21 =
Ch1.Measurements.get_Item("Measurement3");
        IAgilentNMeasurement Ch1S22 =
Ch1.Measurements.get_Item("Measurement4");

        IAgilentNMeasurement Ch1S11P =
Ch1.Measurements.get_Item("Measurement5");
        IAgilentNMeasurement Ch1S12P =
Ch1.Measurements.get_Item("Measurement6");
        IAgilentNMeasurement Ch1S21P =
Ch1.Measurements.get_Item("Measurement7");
        IAgilentNMeasurement Ch1S22P =
Ch1.Measurements.get_Item("Measurement8");

        // Channnel Setting
        Ch1.Points = Convert.ToInt32(textBox9.Text);
        points = Convert.ToInt32(textBox9.Text);

        if (comboBox3.Text == "GHz")
        {
            Ch1.StimulusRange.Start =
Convert.ToDouble(Convert.ToDouble(textBox7.Text) *
Math.Pow(10, 9));
        }
        else if (comboBox3.Text == "MHz")
        {
            Ch1.StimulusRange.Start =
Convert.ToDouble(Convert.ToDouble(textBox7.Text) *
Math.Pow(10, 6));
        }
        else
        {
            Ch1.StimulusRange.Start =
Convert.ToDouble(textBox7.Text);
        }

        if (comboBox4.Text == "GHz")
        {

```

```

        Ch1.StimulusRange.Stop =
Convert.ToDouble(Convert.ToDouble(textBox8.Text) *
Math.Pow(10, 9));
    }
    else if (comboBox4.Text == "MHz")
    {
        Ch1.StimulusRange.Stop =
Convert.ToDouble(Convert.ToDouble(textBox8.Text) *
Math.Pow(10, 6));
    }
    else
    {
        Ch1.StimulusRange.Stop =
Convert.ToDouble(textBox8.Text);
    }

    Ch1.TriggerMode =
AgilentNATriggerModeEnum.AgilentNATriggerModeContinuous;

    driver.Trigger.Source =
AgilentNATriggerSourceEnum.AgilentNATriggerSourceManual;

    // Setup S11 LogMag data measurement
    Ch1S11.Create(1, 1); // Define S11 measurement
    Ch1S11.Format =
AgilentNAMEasurementFormatEnum.AgilentNAMEasurementLogMag;

    // Setup S12 LogMag data measurement
    Ch1S12.Create(1, 2); // Define S12 measurement
    Ch1S12.Format =
AgilentNAMEasurementFormatEnum.AgilentNAMEasurementLogMag;

    // Setup S21 LogMag data measurement
    Ch1S21.Create(2, 1); // Define S21 measurement
    Ch1S21.Format =
AgilentNAMEasurementFormatEnum.AgilentNAMEasurementLogMag;

    // Setup S22 LogMag data measurement
    Ch1S22.Create(2, 2); // Define S22 measurement
    Ch1S22.Format =
AgilentNAMEasurementFormatEnum.AgilentNAMEasurementLogMag;

    textBox10.AppendText("Measuring Channel1
LogMag Data...\r\n");

    Ch1.TriggerSweep(trigms); // Take sweep and
wait up to 2 seconds for sweep to complete

    // Read and output data
    data0 = Ch1S11.FetchFormatted();
    freq0 = Ch1S11.FetchX();

    data1 = Ch1S12.FetchFormatted();
    freq1 = Ch1S12.FetchX();

    data2 = Ch1S21.FetchFormatted();
    freq2 = Ch1S21.FetchX();

    data3 = Ch1S22.FetchFormatted();

```

```

freq3 = Ch1S22.FetchX();

// Excel Files are filled

//S11_LogMag
xlWorkSheetS11_LogMag.Cells[1, xDataOffset] =
textBox6.Text;

for (int i = 0; i < data0.Length; i++)
{
    xlWorkSheetS11_LogMag.Cells[yDataOffset +
i, xDataOffset] = data0[i];
}

for (int i = 0; i < freq0.Length; i++)
{
    xlWorkSheetS11_LogMag.Cells[yFreqOffset +
i, xFreqOffset] = freq0[i];
}

//S12_LogMag
xlWorkSheetS12_LogMag.Cells[1, xDataOffset] =
textBox6.Text;

for (int i = 0; i < data1.Length; i++)
{
    xlWorkSheetS12_LogMag.Cells[yDataOffset +
i, xDataOffset] = data1[i];
}

for (int i = 0; i < freq1.Length; i++)
{
    xlWorkSheetS12_LogMag.Cells[yFreqOffset +
i, xFreqOffset] = freq1[i];
}

//S21_LogMag
xlWorkSheetS21_LogMag.Cells[1, xDataOffset] =
textBox6.Text;

for (int i = 0; i < data2.Length; i++)
{
    xlWorkSheetS21_LogMag.Cells[yDataOffset +
i, xDataOffset] = data2[i];
}

for (int i = 0; i < freq2.Length; i++)
{
    xlWorkSheetS21_LogMag.Cells[yFreqOffset +
i, xFreqOffset] = freq2[i];
}

//S22_LogMag
xlWorkSheetS22_LogMag.Cells[1, xDataOffset] =
textBox6.Text;

```

```

        for (int i = 0; i < data3.Length; i++)
        {
            xlWorkSheetS22_LogMag.Cells[yDataOffset +
i, xDataOffset] = data3[i];
        }
        for (int i = 0; i < freq3.Length; i++)
        {
            xlWorkSheetS22_LogMag.Cells[yFreqOffset +
i, xFreqOffset] = freq3[i];
        }
        // Setup S11 Phase data measurement
        Ch1S11P.Create(1, 1); // Define S11
measurement
        Ch1S11P.Format =
AgilentNAMEasurementFormatEnum.AgilentNAMEasurementPhase;

        // Setup S12 Phase data measurement
        Ch1S12P.Create(1, 2); // Define S12
measurement
        Ch1S12P.Format =
AgilentNAMEasurementFormatEnum.AgilentNAMEasurementPhase;

        // Setup S21 Phase data measurement
        Ch1S21P.Create(2, 1); // Define S21
measurement
        Ch1S21P.Format =
AgilentNAMEasurementFormatEnum.AgilentNAMEasurementPhase;

        // Setup S22 Phase data measurement
        Ch1S22P.Create(2, 2); // Define S22
measurement
        Ch1S22P.Format =
AgilentNAMEasurementFormatEnum.AgilentNAMEasurementPhase;

        textBox10.AppendText("Measuring Channell1 Phase
Data...\r\n");
        Ch1.TriggerSweep(trigms); // Take sweep and
wait up to 2 seconds for sweep to complete

        // Read and output data
        data4 = Ch1S11P.FetchFormatted();
        freq4 = Ch1S11P.FetchX();

        data5 = Ch1S12P.FetchFormatted();
        freq5 = Ch1S12P.FetchX();

        data6 = Ch1S21P.FetchFormatted();
        freq6 = Ch1S21P.FetchX();

        data7 = Ch1S22P.FetchFormatted();
        freq7 = Ch1S22P.FetchX();

        // Excel Files are filled

        //S11_Phase
        xlWorkSheetS11_Phase.Cells[1, xDataOffset] =
textBox6.Text;

```

```

        for (int i = 0; i < data4.Length; i++)
        {
            xlWorkSheetS11_Phase.Cells[yDataOffset +
i, xDataOffset] = data4[i];
        }

        for (int i = 0; i < freq4.Length; i++)
        {
            xlWorkSheetS11_Phase.Cells[yFreqOffset +
i, xFreqOffset] = freq4[i];
        }

        //S12 Phase
        xlWorkSheetS12_Phase.Cells[1, xDataOffset] =
textBox6.Text;

        for (int i = 0; i < data5.Length; i++)
        {
            xlWorkSheetS12_Phase.Cells[yDataOffset +
i, xDataOffset] = data5[i];
        }

        for (int i = 0; i < freq5.Length; i++)
        {
            xlWorkSheetS12_Phase.Cells[yFreqOffset +
i, xFreqOffset] = freq5[i];
        }

        //S21 Phase
        xlWorkSheetS21_Phase.Cells[1, xDataOffset] =
textBox6.Text;

        for (int i = 0; i < data6.Length; i++)
        {
            xlWorkSheetS21_Phase.Cells[yDataOffset +
i, xDataOffset] = data6[i];
        }

        for (int i = 0; i < freq6.Length; i++)
        {
            xlWorkSheetS21_Phase.Cells[yFreqOffset +
i, xFreqOffset] = freq6[i];
        }

        //S22 Phase
        xlWorkSheetS22_Phase.Cells[1, xDataOffset] =
textBox6.Text;

        for (int i = 0; i < data7.Length; i++)
        {
            xlWorkSheetS22_Phase.Cells[yDataOffset +
i, xDataOffset] = data7[i];
        }

        for (int i = 0; i < freq7.Length; i++)
        {
            xlWorkSheetS22_Phase.Cells[yFreqOffset +
i, xFreqOffset] = freq7[i];
        }
    }

```

```

//

// Excel Files are saved

xlWorkBookS11_LogMag.SaveAs(folderPath +
@"\S11_LogMag.xls", Excel.XlFileFormat.xlWorkbookNormal,
misValueS11_LogMag, misValueS11_LogMag, misValueS11_LogMag,
misValueS11_LogMag, Excel.XlSaveAsAccessMode.xlExclusive,
misValueS11_LogMag, misValueS11_LogMag, misValueS11_LogMag,
misValueS11_LogMag, misValueS11_LogMag);
xlWorkBookS11_LogMag.Close(true,
misValueS11_LogMag, misValueS11_LogMag);

xlWorkBookS12_LogMag.SaveAs(folderPath +
@"\S12_LogMag.xls", Excel.XlFileFormat.xlWorkbookNormal,
misValueS12_LogMag, misValueS12_LogMag, misValueS12_LogMag,
misValueS12_LogMag, Excel.XlSaveAsAccessMode.xlExclusive,
misValueS12_LogMag, misValueS12_LogMag, misValueS12_LogMag,
misValueS12_LogMag, misValueS12_LogMag);
xlWorkBookS12_LogMag.Close(true,
misValueS12_LogMag, misValueS12_LogMag);

xlWorkBookS21_LogMag.SaveAs(folderPath +
@"\S21_LogMag.xls", Excel.XlFileFormat.xlWorkbookNormal,
misValueS21_LogMag, misValueS21_LogMag, misValueS21_LogMag,
misValueS21_LogMag, Excel.XlSaveAsAccessMode.xlExclusive,
misValueS21_LogMag, misValueS21_LogMag, misValueS21_LogMag,
misValueS21_LogMag, misValueS21_LogMag);
xlWorkBookS21_LogMag.Close(true,
misValueS21_LogMag, misValueS21_LogMag);

xlWorkBookS22_LogMag.SaveAs(folderPath +
@"\S22_LogMag.xls", Excel.XlFileFormat.xlWorkbookNormal,
misValueS22_LogMag, misValueS22_LogMag, misValueS22_LogMag,
misValueS22_LogMag, Excel.XlSaveAsAccessMode.xlExclusive,
misValueS22_LogMag, misValueS22_LogMag, misValueS22_LogMag,
misValueS22_LogMag, misValueS22_LogMag);
xlWorkBookS22_LogMag.Close(true,
misValueS22_LogMag, misValueS22_LogMag);

xlWorkBookS11_Phase.SaveAs(folderPath +
@"\S11_Phase.xls", Excel.XlFileFormat.xlWorkbookNormal,
misValueS11_Phase, misValueS11_Phase, misValueS11_Phase,
misValueS11_Phase, Excel.XlSaveAsAccessMode.xlExclusive,
misValueS11_Phase, misValueS11_Phase, misValueS11_Phase,
misValueS11_Phase, misValueS11_Phase);
xlWorkBookS11_Phase.Close(true,
misValueS11_Phase, misValueS11_Phase);

xlWorkBookS12_Phase.SaveAs(folderPath +
@"\S12_Phase.xls", Excel.XlFileFormat.xlWorkbookNormal,
misValueS12_Phase, misValueS12_Phase, misValueS12_Phase,
misValueS12_Phase, Excel.XlSaveAsAccessMode.xlExclusive,
misValueS12_Phase, misValueS12_Phase, misValueS12_Phase,
misValueS12_Phase, misValueS12_Phase);
xlWorkBookS12_Phase.Close(true,
misValueS12_Phase, misValueS12_Phase);

```

```

        xlWorkBookS21_Phase.SaveAs(folderPath +
@"\S21_Phase.xls", Excel.XlFileFormat.xlWorkbookNormal,
misValueS21_Phase, misValueS21_Phase, misValueS21_Phase,
misValueS21_Phase, Excel.XlSaveAsAccessMode.xlExclusive,
misValueS21_Phase, misValueS21_Phase, misValueS21_Phase,
misValueS21_Phase, misValueS21_Phase);
        xlWorkBookS21_Phase.Close(true,
misValueS21_Phase, misValueS21_Phase);

        xlWorkBookS22_Phase.SaveAs(folderPath +
@"\S22_Phase.xls", Excel.XlFileFormat.xlWorkbookNormal,
misValueS22_Phase, misValueS22_Phase, misValueS22_Phase,
misValueS22_Phase, Excel.XlSaveAsAccessMode.xlExclusive,
misValueS22_Phase, misValueS22_Phase, misValueS22_Phase,
misValueS22_Phase, misValueS22_Phase);
        xlWorkBookS22_Phase.Close(true,
misValueS22_Phase, misValueS22_Phase);

        // Excel App Closed
xlApp.Quit();

        // Check instrument for errors
int errorNum = -1;
string errorMsg = null;
textBox10.AppendText("\r\n");
while (errorNum != 0)
{
    driver.Utility.ErrorQuery(ref errorNum,
ref errorMsg);
    textBox10.AppendText("ErrorQuery: " +
errorNum + ", " + errorMsg + "\r\n");
}

        System.Windows.Forms.MessageBox.Show("Single
Measurement Completed");
    }
    catch (Exception ex)
    {
        System.Windows.Forms.MessageBox.Show(ex.Message);
    }
}

private void button6_Click(object sender, EventArgs e)
{
    try
    {
        trigms = Convert.ToInt32(textBox12.Text);

        // Excel Offsets
int xFreqOffset = 1;
int yFreqOffset = 2;

int xDataOffset = 2;
int yDataOffset = 2;

        // Excel App Opened

```

```

        Excel.Application xlApp = new
Microsoft.Office.Interop.Excel.Application();

        if (xlApp == null)
        {
            MessageBox.Show("Excel is not properly
installed!!");
            return;
        }

        // Excel Files are created

        //S11 LogMag
        Excel.Workbook xlWorkBookS11_LogMag;
        Excel.Worksheet xlWorkSheetS11_LogMag;
        object misValueS11_LogMag =
System.Reflection.Missing.Value;

        xlWorkBookS11_LogMag =
xlApp.Workbooks.Add(misValueS11_LogMag);
        xlWorkSheetS11_LogMag =
(Excel.Worksheet)xlWorkBookS11_LogMag.Worksheets.get_Item(1);

        //S12_LogMag
        Excel.Workbook xlWorkBookS12_LogMag;
        Excel.Worksheet xlWorkSheetS12_LogMag;
        object misValueS12_LogMag =
System.Reflection.Missing.Value;

        xlWorkBookS12_LogMag =
xlApp.Workbooks.Add(misValueS12_LogMag);
        xlWorkSheetS12_LogMag =
(Excel.Worksheet)xlWorkBookS12_LogMag.Worksheets.get_Item(1);

        //S21_LogMag
        Excel.Workbook xlWorkBookS21_LogMag;
        Excel.Worksheet xlWorkSheetS21_LogMag;
        object misValueS21_LogMag =
System.Reflection.Missing.Value;

        xlWorkBookS21_LogMag =
xlApp.Workbooks.Add(misValueS21_LogMag);
        xlWorkSheetS21_LogMag =
(Excel.Worksheet)xlWorkBookS21_LogMag.Worksheets.get_Item(1);

        //S22_LogMag
        Excel.Workbook xlWorkBookS22_LogMag;
        Excel.Worksheet xlWorkSheetS22_LogMag;
        object misValueS22_LogMag =
System.Reflection.Missing.Value;

        xlWorkBookS22_LogMag =
xlApp.Workbooks.Add(misValueS22_LogMag);
        xlWorkSheetS22_LogMag =
(Excel.Worksheet)xlWorkBookS22_LogMag.Worksheets.get_Item(1);

        //S11_Phase
        Excel.Workbook xlWorkBookS11_Phase;
        Excel.Worksheet xlWorkSheetS11_Phase;

```

```

        object misValueS11_Phase =
System.Reflection.Missing.Value;

        xlWorkBookS11_Phase =
xlApp.Workbooks.Add(misValueS11_Phase);
        xlWorkSheetS11_Phase =
(Excel.Worksheet)xlWorkBookS11_Phase.Worksheets.get_Item(1);

        //S12_Phase
        Excel.Workbook xlWorkBookS12_Phase;
        Excel.Worksheet xlWorkSheetS12_Phase;
        object misValueS12_Phase =
System.Reflection.Missing.Value;

        xlWorkBookS12_Phase =
xlApp.Workbooks.Add(misValueS12_Phase);
        xlWorkSheetS12_Phase =
(Excel.Worksheet)xlWorkBookS12_Phase.Worksheets.get_Item(1);

        //S21_Phase
        Excel.Workbook xlWorkBookS21_Phase;
        Excel.Worksheet xlWorkSheetS21_Phase;
        object misValueS21_Phase =
System.Reflection.Missing.Value;

        xlWorkBookS21_Phase =
xlApp.Workbooks.Add(misValueS21_Phase);
        xlWorkSheetS21_Phase =
(Excel.Worksheet)xlWorkBookS21_Phase.Worksheets.get_Item(1);

        //S22_Phase
        Excel.Workbook xlWorkBookS22_Phase;
        Excel.Worksheet xlWorkSheetS22_Phase;
        object misValueS22_Phase =
System.Reflection.Missing.Value;

        xlWorkBookS22_Phase =
xlApp.Workbooks.Add(misValueS22_Phase);
        xlWorkSheetS22_Phase =
(Excel.Worksheet)xlWorkBookS22_Phase.Worksheets.get_Item(1);

        // Message box cleared
        textBox13.Text = "";

        // Determine current date and time
        string dateTimeNow =
DateTime.Now.ToString("yyyyMMdd_HH:mm:ss");

        //Creates measurements folder
        string folderPath = @"C:\Measurements\" +
dateTimeNow + @"_" + comboBox5.Text + @"_" + trigms + @"ms" +
@"_";

        if (comboBox7.Text == "GHz")
        {
            folderPath = folderPath +
Convert.ToDouble(Convert.ToDouble(textBox16.Text) *
Math.Pow(10, 9)) + @"_";
        }

```

```

        else if (comboBox7.Text == "MHz")
        {
            folderPath = folderPath +
Convert.ToDouble(Convert.ToDouble(textBox16.Text) *
Math.Pow(10, 6)) + @"_";
        }
        else
        {
            folderPath = folderPath +
Convert.ToDouble(textBox16.Text) + @"_";
        }

        if (comboBox6.Text == "GHz")
        {
            folderPath = folderPath +
Convert.ToDouble(Convert.ToDouble(textBox15.Text) *
Math.Pow(10, 9)) + @"_";
        }
        else if (comboBox6.Text == "MHz")
        {
            folderPath = folderPath +
Convert.ToDouble(Convert.ToDouble(textBox15.Text) *
Math.Pow(10, 6)) + @"_";
        }
        else
        {
            folderPath = folderPath +
Convert.ToDouble(textBox15.Text) + @"_";
        }

        folderPath = folderPath + textBox14.Text +
@"_Multiple";

        System.IO.Directory.CreateDirectory(folderPath);

        // Setup a measurement and output data
        double[] data0 = null;
        double[] freq0 = null;
        double[] freq0init = null;

        double[] data1 = null;
        double[] freq1 = null;
        double[] freq1init = null;

        double[] data2 = null;
        double[] freq2 = null;
        double[] freq2init = null;

        double[] data3 = null;
        double[] freq3 = null;
        double[] freq3init = null;

        double[] data4 = null;
        double[] freq4 = null;
        double[] freq4init = null;

        double[] data5 = null;
        double[] freq5 = null;
        double[] freq5init = null;

```

```

double[] data6 = null;
double[] freq6 = null;
double[] freq6init = null;

double[] data7 = null;
double[] freq7 = null;
double[] freq7init = null;

int points;

// Setup convenient pointers to Channels and
Measurements
IAgilentNAChannel Ch1 =
driver.Channels.get_Item("Channel1");
IAgilentNAMeasurement Ch1S11 =
Ch1.Measurements.get_Item("Measurement1");
IAgilentNAMeasurement Ch1S12 =
Ch1.Measurements.get_Item("Measurement2");
IAgilentNAMeasurement Ch1S21 =
Ch1.Measurements.get_Item("Measurement3");
IAgilentNAMeasurement Ch1S22 =
Ch1.Measurements.get_Item("Measurement4");

IAgilentNAMeasurement Ch1S11P =
Ch1.Measurements.get_Item("Measurement5");
IAgilentNAMeasurement Ch1S12P =
Ch1.Measurements.get_Item("Measurement6");
IAgilentNAMeasurement Ch1S21P =
Ch1.Measurements.get_Item("Measurement7");
IAgilentNAMeasurement Ch1S22P =
Ch1.Measurements.get_Item("Measurement8");

// Channel Setting
Ch1.Points = Convert.ToInt32(textBox14.Text);
points = Convert.ToInt32(textBox14.Text);

if (comboBox7.Text == "GHz")
{
    Ch1.StimulusRange.Start =
Convert.ToDouble(Convert.ToDouble(textBox16.Text) *
Math.Pow(10, 9));
}
else if (comboBox7.Text == "MHz")
{
    Ch1.StimulusRange.Start =
Convert.ToDouble(Convert.ToDouble(textBox16.Text) *
Math.Pow(10, 6));
}
else
{
    Ch1.StimulusRange.Start =
Convert.ToDouble(textBox16.Text);
}

if (comboBox6.Text == "GHz")
{

```

```

        Ch1.StimulusRange.Stop =
Convert.ToDouble(Convert.ToDouble(textBox15.Text) *
Math.Pow(10, 9));
    }
    else if (comboBox6.Text == "MHz")
    {
        Ch1.StimulusRange.Stop =
Convert.ToDouble(Convert.ToDouble(textBox15.Text) *
Math.Pow(10, 6));
    }
    else
    {
        Ch1.StimulusRange.Stop =
Convert.ToDouble(textBox15.Text);
    }
    Ch1.TriggerMode =
AgilentNATriggerModeEnum.AgilentNATriggerModeContinuous;

    driver.Trigger.Source =
AgilentNATriggerSourceEnum.AgilentNATriggerSourceManual;

    //////////////////////////////////////
    // Measurement Loop Starts here //
    //////////////////////////////////////

    int ArduinoData = 0;

    string binaryData;

    int loopIterationNumber;

    if(comboBox5.Text == "6")
    {
        loopIterationNumber = 64;
    }
    else if (comboBox5.Text == "5")
    {
        loopIterationNumber = 32;
    }
    else
    {
        loopIterationNumber = 128;
    }
    for(int y = 0; y < loopIterationNumber; y++)
    {
        progressBar1.Value =
Convert.ToInt32((Convert.ToDouble(y) /
Convert.ToDouble(loopIterationNumber-1))*100);

        // Arduino data changed here

        string tempSS;
        bool errorBit = false;

        if (textBox3.Text != "Simulate")
        {

            if (comboBox5.Text == "6")
            {

```

```

        binaryData = ToBin(ArduinoData, 6);
        tempSS = 'a' + binaryData + '0' + 'b';
        inputWrite = tempSS.ToCharArray();

        ArduinoSentData1 = ToBin(ArduinoData, 6) + '0';
    }
    else if (comboBox5.Text == "5")
    {
        binaryData = ToBin(ArduinoData, 5);
        tempSS = 'a' + binaryData + "00" + 'b';
        inputWrite = tempSS.ToCharArray();

        ArduinoSentData1 = ToBin(ArduinoData, 5) + "00";
    }
    else
    {
        binaryData = ToBin(ArduinoData, 7);
        tempSS = 'a' + binaryData + 'b';
        inputWrite = tempSS.ToCharArray();

        ArduinoSentData1 = ToBin(ArduinoData, 7);
    }

    if (binaryData.Length !=
Convert.ToInt32(comboBox5.Text))
    {
        errorBit = true;
    }

    if (!errorBit)
    {
        port.Write(inputWrite, 0, 9);

        // Arduino data check
        checkSentToArduinol();
        checkSentToArduinol();
        checkSentToArduinol();
    }
    else
    {
        textBox13.Text = textBox13.Text +
"Error: Bit Number not match\r\n";
    }
}
else
{
    if (comboBox5.Text == "6")
    {
        binaryData = ToBin(ArduinoData, 6);
        tempSS = 'a' + binaryData + '0' + 'b';
        inputWrite = tempSS.ToCharArray();

        ArduinoSentData1 = ToBin(ArduinoData, 6) + '0';
    }
    else if (comboBox5.Text == "5")
    {
        binaryData = ToBin(ArduinoData, 5);
        tempSS = 'a' + binaryData + "00" + 'b';

```

```

        inputWrite = tempSS.ToCharArray();

    ArduinoSentData1 = ToBin(ArduinoData, 5) + "00";
    }
    else
    {
        binaryData = ToBin(ArduinoData, 7);
        tempSS = 'a' + binaryData + 'b';
        inputWrite = tempSS.ToCharArray();

        ArduinoSentData1 = ToBin(ArduinoData, 7);
    }
}

    ArduinoData++;

    //

    // Define S11 measurement
    Ch1S11.Create(1, 1);
    // Setup S11 LogMag data measurement
    Ch1S11.Format =
AgilentNAMEasurementFormatEnum.AgilentNAMEasurementLogMag;

    // Define S12 measurement
    Ch1S12.Create(1, 2);
    // Setup S12 LogMag data measurement
    Ch1S12.Format =
AgilentNAMEasurementFormatEnum.AgilentNAMEasurementLogMag;

    // Define S21 measurement
    Ch1S21.Create(2, 1);
    // Setup S21 LogMag data measurement
    Ch1S21.Format =
AgilentNAMEasurementFormatEnum.AgilentNAMEasurementLogMag;

    // Define S22 measurement
    Ch1S22.Create(2, 2);
    // Setup S22 LogMag data measurement
    Ch1S22.Format =
AgilentNAMEasurementFormatEnum.AgilentNAMEasurementLogMag;

    textBox13.AppendText("Measuring Channel1
LogMag Data...\r\n");
    Ch1.TriggerSweep(trigms); // Take sweep
and wait up to 2 seconds for sweep to complete

    // Read and output data
    data0 = Ch1S11.FetchFormatted();
    freq0 = Ch1S11.FetchX();

    data1 = Ch1S12.FetchFormatted();
    freq1 = Ch1S12.FetchX();

    data2 = Ch1S21.FetchFormatted();
    freq2 = Ch1S21.FetchX();

    data3 = Ch1S22.FetchFormatted();
    freq3 = Ch1S22.FetchX();

```

```

        // Excel Files are filled
        //S11_LogMag
        xlWorkSheetS11_LogMag.Cells[1, xDataOffset] =
binaryData;

        for (int i = 0; i < data0.Length; i++)
        {
            xlWorkSheetS11_LogMag.Cells[yDataOffset + i,
xDataOffset] = data0[i];
        }

        if (y == 0)
        {
            for (int i = 0; i < freq0.Length; i++)
            {
                xlWorkSheetS11_LogMag.Cells[yFreqOffset + i, xFreqOffset] =
freq0[i];
            }

            freq0init = new double[freq0.Length];
        }
        else
        {
            if (!freq0init.SequenceEqual(freq0))
            {
                textBox13.AppendText("Error:
Frequency data do not match\r\n");
            }
        }
        //S12_LogMag
        xlWorkSheetS12_LogMag.Cells[1,
xDataOffset] = binaryData;

        for (int i = 0; i < data1.Length; i++)
        {
            xlWorkSheetS12_LogMag.Cells[yDataOffset + i, xDataOffset] =
data1[i];
        }

        if (y == 0)
        {
            for (int i = 0; i < freq1.Length; i++)
            {
                xlWorkSheetS12_LogMag.Cells[yFreqOffset + i, xFreqOffset] =
freq1[i];
            }

            freq1init = new double[freq1.Length];
        }
        else
        {
            if (!freq1init.SequenceEqual(freq1))
            {
                textBox13.AppendText("Error:
Frequency data do not match\r\n");
            }
        }
    }
}

```

```

        }
    }
    //S21_LogMag
xlWorkSheetS21_LogMag.Cells[1, xDataOffset] = binaryData;

    for (int i = 0; i < data2.Length; i++)
    {

xlWorkSheetS21_LogMag.Cells[yDataOffset + i, xDataOffset] =
data2[i];

    }

    if (y == 0)
    {
        for (int i = 0; i < freq2.Length; i++)
        {

xlWorkSheetS21_LogMag.Cells[yFreqOffset + i,
xFreqOffset] = freq2[i];

        }

        freq2init = new double[freq2.Length];
    }
    else
    {
        if (!freq2init.SequenceEqual(freq2))
        {
            textBox13.AppendText("Error:
Frequency data do not match\r\n");
        }
    }
    //S22_LogMag
xlWorkSheetS22_LogMag.Cells[1, xDataOffset] = binaryData;

    for (int i = 0; i < data3.Length; i++)
    {

xlWorkSheetS22_LogMag.Cells[yDataOffset + i, xDataOffset] =
data3[i];

    }
    if (y == 0)
    {
        for (int i = 0; i < freq3.Length; i++)
        {

xlWorkSheetS22_LogMag.Cells[yFreqOffset + i, xFreqOffset] =
freq3[i];

        }
        freq3init = new double[freq3.Length];
    }
    else
    {
        if (!freq3init.SequenceEqual(freq3))
        {
            textBox13.AppendText("Error:
Frequency data do not match\r\n");
        }
    }
    //

```

```

        // Define S11 measurement
        Ch1S11P.Create(1, 1);
        // Setup S11 Phase data measurement
        Ch1S11P.Format =
AgilentNAMEasurementFormatEnum.AgilentNAMEasurementPhase;

        // Define S12 measurement
        Ch1S12P.Create(1, 2);
        // Setup S12 Phase data measurement
        Ch1S12P.Format =
AgilentNAMEasurementFormatEnum.AgilentNAMEasurementPhase;

        // Define S21 measurement
        Ch1S21P.Create(2, 1);
        // Setup S21 Phase data measurement
        Ch1S21P.Format =
AgilentNAMEasurementFormatEnum.AgilentNAMEasurementPhase;

        // Define S22 measurement
        Ch1S22P.Create(2, 2);
        // Setup S22 Phase data measurement
        Ch1S22P.Format =
AgilentNAMEasurementFormatEnum.AgilentNAMEasurementPhase;

        textBox13.AppendText("Measuring Channel1
Phase Data...\r\n");
        Ch1.TriggerSweep(trigms); // Take sweep
and wait up to 2 seconds for sweep to complete

        // Read and output data
        data4 = Ch1S11P.FetchFormatted();
        freq4 = Ch1S11P.FetchX();

        data5 = Ch1S12P.FetchFormatted();
        freq5 = Ch1S12P.FetchX();

        data6 = Ch1S21P.FetchFormatted();
        freq6 = Ch1S21P.FetchX();

        data7 = Ch1S22P.FetchFormatted();
        freq7 = Ch1S22P.FetchX();

        // Excel Files are filled

        //S11_Phase
        xlWorkSheetS11_Phase.Cells[1, xDataOffset]
= binaryData;

        for (int i = 0; i < data4.Length; i++)
        {
            xlWorkSheetS11_Phase.Cells[yDataOffset + i, xDataOffset] =
            data4[i];
        }

        if (y == 0)
        {
            for (int i = 0; i < freq4.Length; i++)

```

```

        {
xlWorkSheetS11_Phase.Cells[yFreqOffset + i, xFreqOffset] =
freq4[i];
        }

        freq4init = new double[freq4.Length];
    }
    else
    {
        if (!freq4init.SequenceEqual(freq4))
        {
            textBox13.AppendText("Error:
Frequency data do not match\r\n");
        }
    }

    //S12_Phase
xlWorkSheetS12_Phase.Cells[1, xDataOffset] = binaryData;

    for (int i = 0; i < data5.Length; i++)
    {
xlWorkSheetS12_Phase.Cells[yDataOffset + i, xDataOffset] =
data5[i];
    }

    if (y == 0)
    {
        for (int i = 0; i < freq5.Length; i++)
        {

xlWorkSheetS12_Phase.Cells[yFreqOffset + i, xFreqOffset] =
freq5[i];
        }
        freq5init = new double[freq5.Length];
    }
    else
    {
        if (!freq5init.SequenceEqual(freq5))
        {
            textBox13.AppendText("Error:
Frequency data do not match\r\n");
        }
    }

    //S21_Phase
xlWorkSheetS21_Phase.Cells[1, xDataOffset] = binaryData;

    for (int i = 0; i < data6.Length; i++)
    {
        xlWorkSheetS21_Phase.Cells[yDataOffset
+ i, xDataOffset] = data6[i];
    }
    if (y == 0)
    {
        for (int i = 0; i < freq6.Length; i++)
        {
xlWorkSheetS21_Phase.Cells[yFreqOffset + i, xFreqOffset] =
freq6[i];
        }
    }

```

```

        freq6init = new double[freq6.Length];
    }
    else
    {
        if (!freq6init.SequenceEqual(freq6))
        {
            textBox13.AppendText("Error:
Frequency data do not match\r\n");
        }
    }

    //S22_Phase
xlWorkSheetS22_Phase.Cells[1, xDataOffset] = binaryData;

    for (int i = 0; i < data7.Length; i++)
    {
        xlWorkSheetS22_Phase.Cells[yDataOffset
+ i, xDataOffset] = data7[i];
    }

    if (y == 0)
    {
        for (int i = 0; i < freq7.Length; i++)
        {

xlWorkSheetS22_Phase.Cells[yFreqOffset + i, xFreqOffset] =
freq7[i];

        }
        freq7init = new double[freq7.Length];
    }
    else
    {
        if (!freq7init.SequenceEqual(freq7))
        {
            textBox13.AppendText("Error:
Frequency data do not match\r\n");
        }
    }

    xDataOffset++;
}

//////////////////////////
// Measurement Loop Ends here //
//////////////////////////

// Excel Files are saved
xlWorkbookS11_LogMag.SaveAs(folderPath +
@"\S11_LogMag.xls", Excel.XlFileFormat.xlWorkbookNormal,
misValueS11_LogMag, misValueS11_LogMag, misValueS11_LogMag,
misValueS11_LogMag, Excel.XlSaveAsAccessMode.xlExclusive,
misValueS11_LogMag, misValueS11_LogMag, misValueS11_LogMag,
misValueS11_LogMag, misValueS11_LogMag);
xlWorkbookS11_LogMag.Close(true,
misValueS11_LogMag, misValueS11_LogMag);

```

```

xlWorkBookS12_LogMag.SaveAs(folderPath +
@"\S12_LogMag.xls", Excel.XlFileFormat.xlWorkbookNormal,
misValueS12_LogMag, misValueS12_LogMag, misValueS12_LogMag,
misValueS12_LogMag, Excel.XlSaveAsAccessMode.xlExclusive,
misValueS12_LogMag, misValueS12_LogMag, misValueS12_LogMag,
misValueS12_LogMag, misValueS12_LogMag);
xlWorkBookS12_LogMag.Close(true,
misValueS12_LogMag, misValueS12_LogMag);

```

```

xlWorkBookS21_LogMag.SaveAs(folderPath +
@"\S21_LogMag.xls", Excel.XlFileFormat.xlWorkbookNormal,
misValueS21_LogMag, misValueS21_LogMag, misValueS21_LogMag,
misValueS21_LogMag, Excel.XlSaveAsAccessMode.xlExclusive,
misValueS21_LogMag, misValueS21_LogMag, misValueS21_LogMag,
misValueS21_LogMag, misValueS21_LogMag);
xlWorkBookS21_LogMag.Close(true,
misValueS21_LogMag, misValueS21_LogMag);

```

```

xlWorkBookS22_LogMag.SaveAs(folderPath +
@"\S22_LogMag.xls", Excel.XlFileFormat.xlWorkbookNormal,
misValueS22_LogMag, misValueS22_LogMag, misValueS22_LogMag,
misValueS22_LogMag, Excel.XlSaveAsAccessMode.xlExclusive,
misValueS22_LogMag, misValueS22_LogMag, misValueS22_LogMag,
misValueS22_LogMag, misValueS22_LogMag);
xlWorkBookS22_LogMag.Close(true,
misValueS22_LogMag, misValueS22_LogMag);

```

```

xlWorkBookS11_Phase.SaveAs(folderPath +
@"\S11_Phase.xls", Excel.XlFileFormat.xlWorkbookNormal,
misValueS11_Phase, misValueS11_Phase, misValueS11_Phase,
misValueS11_Phase, Excel.XlSaveAsAccessMode.xlExclusive,
misValueS11_Phase, misValueS11_Phase, misValueS11_Phase,
misValueS11_Phase, misValueS11_Phase);
xlWorkBookS11_Phase.Close(true,
misValueS11_Phase, misValueS11_Phase);

```

```

xlWorkBookS12_Phase.SaveAs(folderPath +
@"\S12_Phase.xls", Excel.XlFileFormat.xlWorkbookNormal,
misValueS12_Phase, misValueS12_Phase, misValueS12_Phase,
misValueS12_Phase, Excel.XlSaveAsAccessMode.xlExclusive,
misValueS12_Phase, misValueS12_Phase, misValueS12_Phase,
misValueS12_Phase, misValueS12_Phase);
xlWorkBookS12_Phase.Close(true,
misValueS12_Phase, misValueS12_Phase);

```

```

xlWorkBookS21_Phase.SaveAs(folderPath +
@"\S21_Phase.xls", Excel.XlFileFormat.xlWorkbookNormal,
misValueS21_Phase, misValueS21_Phase, misValueS21_Phase,
misValueS21_Phase, Excel.XlSaveAsAccessMode.xlExclusive,
misValueS21_Phase, misValueS21_Phase, misValueS21_Phase,
misValueS21_Phase, misValueS21_Phase);

```

```

        xlWorkBookS21_Phase.Close(true,
misValueS21_Phase, misValueS21_Phase);

        xlWorkBookS22_Phase.SaveAs(folderPath +
@"\S22_Phase.xls", Excel.XlFileFormat.xlWorkbookNormal,
misValueS22_Phase, misValueS22_Phase, misValueS22_Phase,
misValueS22_Phase, Excel.XlSaveAsAccessMode.xlExclusive,
misValueS22_Phase, misValueS22_Phase, misValueS22_Phase,
misValueS22_Phase, misValueS22_Phase);
        xlWorkBookS22_Phase.Close(true,
misValueS22_Phase, misValueS22_Phase);

        // Excel App Closed
xlApp.Quit();

        // Check instrument for errors
int errorNum = -1;
string errorMsg = null;
textBox13.AppendText("\r\n");
while (errorNum != 0)
{
driver.Utility.ErrorQuery(ref errorNum, ref errorMsg);
textBox13.AppendText("ErrorQuery: " + errorNum + ", " +
errorMsg + "\r\n");
}

System.Windows.Forms.MessageBox.Show("Multiple Measurements
Completed");
}
catch (Exception ex)
{
System.Windows.Forms.MessageBox.Show(ex.Message);
}
}

private void checkSentToArduino1()
{
    string tempSS = "c";
    inputWrite = tempSS.ToCharArray();

    port.Write(inputWrite, 0, 1);

    if (port.ReadChar() != 'd')
    {
        textBox13.AppendText("Error [Arduino]\r\n");
    }

    char tempc;
    tempc = Convert.ToChar(port.ReadChar());
    inputRead[0] = tempc;

    if (!(tempc == '1' || tempc == '0'))
    {
        textBox13.AppendText("Error [Arduino]\r\n");
    }

    tempc = Convert.ToChar(port.ReadChar());

```

```

        inputRead[1] = tempc;
        if (!(tempc == '1' || tempc == '0'))
        {
            textBox13.AppendText("Error [Arduino]\r\n");
        }
        tempc = Convert.ToChar(port.ReadChar());
        inputRead[2] = tempc;

        if (!(tempc == '1' || tempc == '0'))
        {
            textBox13.AppendText("Error [Arduino]\r\n");
        }
        tempc = Convert.ToChar(port.ReadChar());
        inputRead[3] = tempc;
        if (!(tempc == '1' || tempc == '0'))
        {
            textBox13.AppendText("Error [Arduino]\r\n");
        }
        tempc = Convert.ToChar(port.ReadChar());
        inputRead[4] = tempc;
        if (!(tempc == '1' || tempc == '0'))
        {
            textBox13.AppendText("Error [Arduino]\r\n");
        }
        tempc = Convert.ToChar(port.ReadChar());
        inputRead[5] = tempc;
        if (!(tempc == '1' || tempc == '0'))
        {
            textBox13.AppendText("Error [Arduino]\r\n");
        }
        tempc = Convert.ToChar(port.ReadChar());
        inputRead[6] = tempc;
        if (!(tempc == '1' || tempc == '0'))
        {
            textBox13.AppendText("Error [Arduino]\r\n");
        }
        if (port.ReadChar() != 'e')
        {
            textBox13.AppendText("Error [Arduino]\r\n");
        }
        string tempS = new string(inputRead);
        if (ArduinoSentData1 != tempS)
        {
            textBox13.AppendText("Error [Arduino]\r\n");
        }
        else
        {
            textBox13.AppendText("Successful
[Arduino]\r\n");
        }
    }
    public static string ToBin(int value, int len)
    {
        return (len > 1 ? ToBin(value >> 1, len - 1) :
null) + "01"[value & 1];
    }
}
}

```