UNIVERSITY OF CALIFORNIA,
IRVINE

Design, Analysis and Implementation of an Area-Efficient Beam-Steerable Sub-Terahertz Imaging Receiver Array Architecture

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Electrical Engineering

by

Francis M. Caster II

Dissertation Committee:
Professor Payam Heydari, Chair
Professor Kumar Wickramasinghe
Professor Lee Swindlehurst

2014

Dedication

To

My wife,

Cindy,

whose patience, support and understanding

throughout this journey,

have made this possible.

And to my daughter,

Kealani,

who would always,

when circumstance required,

happily accompany me

to the university

throughout the course of my research.

Table of Contents

Page

List of Symbols

GHz : gigahertz

a_1 : current gain

a_2 : current gains

$A_{k,l}^{(m,n)}$: the gain weights associated with element (k, l) going to super-pixel (m,n)

BiCMOS : bipolar CMOS

BW : system bandwidth

C : capacitor

C++ : programming language

c_π : transistor input capacitance

C_{CAS} : cascode node parasitic capacitance

CMOS : complementary metal-oxide semiconductor

CPW : ground-shielded coplanar waveguide

DAQ : data acquisition

dB : decibels

DC : direct current

DFF : D flip flop

EM : electro-magnetic

f : Frequency in Hertz

FOV : field of view

f_{MAX} : transistor maximum oscillation frequency

f_T : transistor unity gain frequency

G : pre-detection gain

ΔG : rms variation of G

GaAs : gallium arsenide

gm : transistor transconductance

GPIO : general purpose input output

GUI : graphical user interface

HBT : heterojunction bipolar transistor

HEMT :

I_C : Collector current

InP : indium phosphide

k_B : the Boltzmann constant

λ : wavelength

L : inductor

L_{BASE} : base inductance

L_{EFF} : efeective inductance

LNA : low noise amplifier

LO : local oscillator

LTCC : low temperature co-fired ceramic

MAG : maximum available gain

MIM : metal-insulator-metal

NAND : logic gate

NEP : noise equivalent power

NETD : noise equivalent temperature difference

NF : noise figure

NOR : logic gate

Ω : Ohm

P : power

PAF : phased array feeds

PCB : printed circuit board

PL : propagation loss

Q : quality factor

R : resistor

r_π : transistor input resistance

r_b : transistor base resistance

R_{EFF} : effective resistance

RF : radio frequency

$\Re(Z_{in})$: real part of Z_{in}

$\Re_S$: one-stage power detector responsivity

$\Re_T$: two-stage power detector responsivity

Sb : antimony

SiGe : silicon germanium

SNR : signal to noise ratio

SPDT : single pole double-throw

T_{BASE} : base T-line

T_{delay} : group delay

TQFP : thin quad flat pack

τ : back-end integration time

τ_d : unit delay

$\tau_{k,l}^{(m,n)}$: the delay weights associated with element (k, l) going to super-pixel (m,n)

T_{SYS} : the noise temperature of the receiver

TTD : true time delay

USB : universal serial bus

V : voltage

$V_C(m,n,t)$: the signal at the combiner's output (m,n)

VGA : variable gain amplifier

$V_{in}(k,l,t)$: the signal received by each element (k,l) with respect to time

V_T : Thermal Voltage

ω : Frequency in radians

ω_t : Transistor cutoff frequency in radians

Z_{in} : input impedance

Z_0 : Source impedance.

2-D : 2 dimensional

3-D : 3 dimensional

List of Figures

List of Tables

Acknowledgements

I would like to thank my advisor, Professor Payam Heydari, for his guidance, constructive criticism and encouragement throughout my graduate career here at UCI. His drive, enthusiasm and passion for excellence in engineering research have been a constant source of inspiration to me.

I would like to acknowledge Professor Filippo Capolino. I have enjoyed the occasional collaboration between his EM research group and the NCIC Lab during the course of my research.

I also would like to thank my committee members, Professor Kumar Wickramasinghe and Professor Lee Swindlehurst for their willing participation in this process.

I would like to acknowledge several people for their contributions to my research. They are Leland Gilreath for his many informative discussions on all things imaging-related during the course of my research, Zheng Wang for his contribution to the 1:4 active power splitter and TTD section, Shiji Pan for his contribution to the antenna section, and Mehdi Veysi for useful discussions on focal systems.

I gratefully acknowledge the support provided by TowerJazz for manufacturing my test chips, and Anritsu for the loan of the test equipment needed to measure the performance of my circuits.

Curriculum Vitae

Francis M. Caster II

Education

1984	B.S. in Electrical Engineering California State University, Fullerton
2012	M.S. in Electrical Engineering University of California, Irvine
2014	Ph.D. in Electrical Engineering University of California, Irvine

Professional Experience

2002-2010	Owner, AFC Designs, Inc.
2000-2002	Manager, Globespan
1998-2002	Project Lead, VTC
1990-1998	Owner, AFC Designs, Inc.
1984-1990	Design Engineer, Rockwell International

Publications

- F. Caster II, L. Gilreath, S. Pan, Z. Wang, F. Capolino, P. Heydari, "Design and Analysis of a Wband 9-Element Imaging Array Receiver Using Spatial-Overlapping Super-Pixels in Silicon" *IEEE Journal of Solid-State Circuits*, vol. 49, no. 6, pp. 1317 – 1332, June 2014
- S. Pan, F. Caster II, P. Heydari, F. Capolino, "A 94 GHz Extremely Thin Metasurface-Based BiCMOS On-chip Antenna" *IEEE Transactions on Antennas and Propagation*, 2014
- F. Caster II, L. Gilreath, S. Pan, Z. Wang, F. Capolino, P. Heydari, "A 93-113GHz BiCMOS 9-element Imaging Array Receiver Utilizing Spatial-Overlapping Pixels with Wideband

Phase and Amplitude Control" *Solid-State Circuits Conference Digest of Technical Papers* (ISSCC), 2013, pp. 144 – 145

- N. Tan, F. Caster II, C. Eichrodt, S. George, B. Horng, J. Zhao, "A universal quad AFE with integrated filters for VDSL, ADSL, and G.SHDSL", *Custom Integrated Circuits Conference*, 2003, pp. 599 – 602

Patents

- Programmable analog-to-digital converter with bit conversion optimization – Patent 6329938
- Wideband analog front-end for DSL applications - Patent 7061987
- Complementary mixer – patent pending

Abstact of the Dissertation

Design, Analysis and Implementation of an Area-Efficient Beam-Steerable Sub-Terahertz Imaging Receiver Array Architecture

By

Francis M. Caster II

Doctor of Philosophy in Electrical Engineering

University of California, Irvine

Professor Payam Heydari Irvine, Chair

Multi-pixel passive imaging arrays are presently fabricated in expensive processes with low integration levels. Imaging arrays designed in inexpensive processes with higher levels of integration would result in cheaper larger arrays. The objective of this work is twofold: 1) to develop a highly integrated silicon-based passive imaging receiver integrating the entire receiver frontend from antenna to detector; and 2) to develop a highly scalable architecture for use in focal plane arrays of arbitrary dimensions, incorporating new features to address large-array issues while retaining the pixel density of conventional focal plane arrays. A new area-efficient beam-steerable imaging array receiver architecture suitable to integrated circuit realization up to sub-THz frequencies is presented. A direct-detection-based receiver array utilizing the proposed architecture is designed and implemented in an advanced 0.18μm BiCMOS process. The receiver chip achieves a peak measured coherent responsivity of 1,150MV/W, a measured

incoherent responsivity of 1,000MV/W, a minimum NEP of 0.28fW/Hz$^{1/2}$ and a front-end 3-dB bandwidth from 87–108GHz, while consuming 225mW per receiver element. The measured NETD of the SiGe receiver chip is 0.45K with a 20ms integration time. The proposed architecture uses a new concept of spatial-overlapping super-pixels which results in (1) improved SNR at the pixel level through a reduction of spillover losses, (2) partially correlated adjacent super-pixels, (3) a 2×2 window averaging function in the RF domain, (4) the ability to compensate for the systematic time delay and amplitude variations due to the off-focal-point effect for antennas away from the focal point, and (5) the ability to compensate for mutual coupling effects among the array elements.

Introduction

Imaging systems are increasingly finding their way into our daily lives in areas as diverse as medical diagnostic imaging, full-body security scanning at airports, building safety inspection and reduced-visibility navigation.

Several factors are slowing the development rate of imaging systems into yet more aspects of society. The main factors are cost and scan time. The imaging receivers in widespread use today are fabricated in expensive processes with very low integration levels, often resulting in scanners with low numbers of receivers to control cost. Many times the receivers are arrayed in only one dimension. This leads to scanners with long scan times, as the scanner has to be mechanically scanned across the area of interest to obtain a full scan.

As integration level goes up, cost generally goes down. With decreasing cost and increasing integration levels, scanners can include more receivers. Cheaper processes supporting high integration levels would therefore enable the development of larger arrays, resulting in more affordable scanners with reduced scan times.

Large arrays of receivers, while minimizing scan time, pose new challenges for imaging receiver designers. Issues such as coupling and mismatch among the receivers, along with receiver distance from the center of the focusing optics, affect the performance of the larger arrays.

This work explores the possibility of designing an imaging receiver in a silicon-based process to address the cost and integration concerns. This work also addresses the large array

concerns by developing a new area efficient, highly scalable, imaging receiver array architecture with the capability to mitigate some of the large array effects.

This work is organized as follows. Chapter 1 introduces background concepts related to imaging, discusses some active and passive imaging principles, provides an overview of prior work and presents the goals for this work. Chapter 2 discusses large array issues and proposes an architecture that addresses these issues. Next, block-level circuit design, analysis and measurement results are presented in Chapter 3. Chapter 4 presents the system-level test setup and measurement results, while conclusions are drawn in Chapter 5.

Chapter 1
Background

Introduction

The main purpose of this chapter is to provide the frame of reference for the motivation behind this research. The first section presents some general background information on concepts related to imaging. The second section presents some important passive and active imaging principles. The next section provides an overview of the current state-of-the-art in imaging receivers. In the last part of the chapter the goals of this work are discussed.

1.1 Background

The millimeter-wave frequency range from 30-300GHz has been a dynamic area of research in the field of passive and active imaging and sensing for several decades. At millimeter-wave frequencies, black body radiation is emitted at a nearly constant power spectral density (i.e., white spectrum), which is directly proportional to the temperature and emissivity of the radiating object. Low-attenuation atmospheric windows centered at frequencies of 35, 94, 140, and 220GHz exist (Figure 1), which provide transmission through obstacles such as clothing, smoke, dust, fog, and clouds. Applications such as concealed weapon detection, aircraft navigation in low visibility conditions, and satellite surveillance have been targeted for imaging systems operating at these frequencies [1]. In recent years, silicon technologies have achieved the required imaging system performance [2], [3], [4], [5], [6] that had previously only been obtained using III-V technologies in multi-chip module based systems [7], [8], [9]. While silicon-based imaging receivers require a lens or other focusing system just as their III-V

counterparts, due to high integration levels and low cost at mass production, silicon-based imaging receiver systems can be especially advantageous when a large number of array elements is required. Combined with the high yield of silicon processes, concepts such as wafer-level arrays become a very exciting and real possibility.

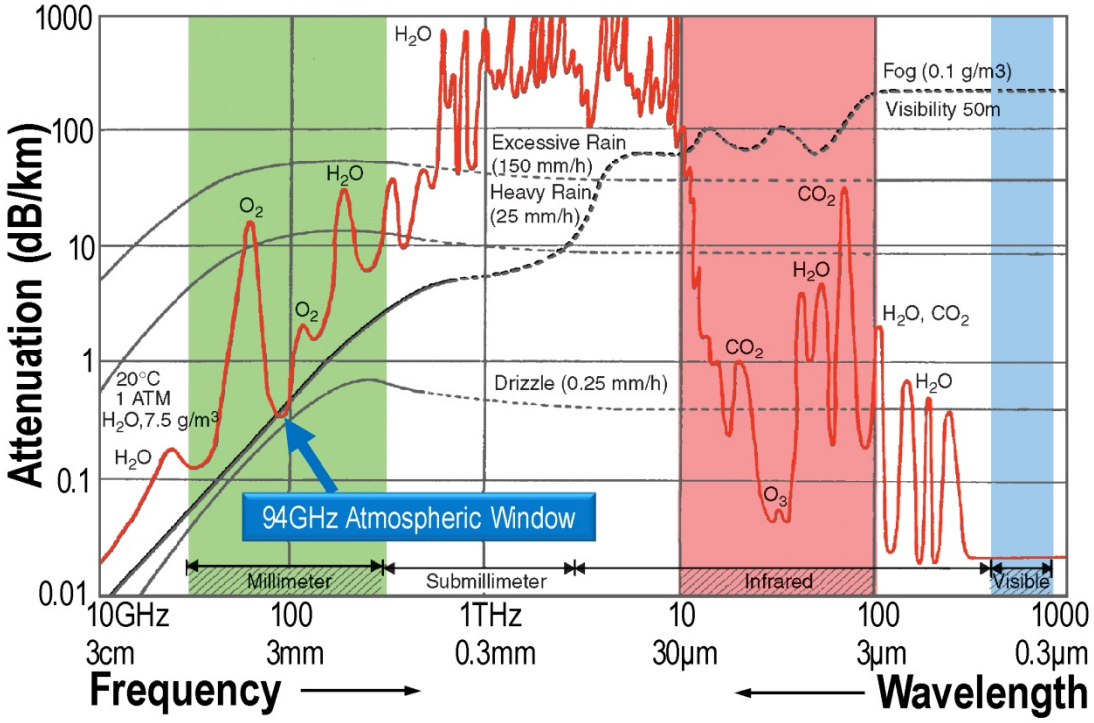


Figure 1. Attenuation profile of the electromagnetic spectrum[1]

1.2 Passive and Active Imaging Principles

The essential performance characteristic of a passive imaging system is its thermal resolution, also known as its noise equivalent temperature difference (NETD). The general equation for the NETD of a passive millimeter-wave receiver is given by [5], [6], [10],

$$NETD = T_{SYS}\sqrt{\frac{1}{BW\cdot\tau}+\left(\frac{\Delta G}{G}\right)^2+\frac{1}{2\tau}\cdot\left(\frac{NEP}{k_B\cdot G\cdot BW\cdot T_{SYS}}\right)^2} \quad (1)$$

where T_{SYS} is the noise temperature of the receiver, BW denotes the system bandwidth, τ is the back-end integration time, G is the pre-detection gain, ΔG is the rms variation of G, k_B is the Boltzmann constant and NEP is the noise-equivalent power of the detector. As discussed in prior work [4], [5], [6], ΔG is often the dominant term in (1), and can degrade the NETD significantly. A major source of the gain variation in many of the technologies used to construct imaging receivers is the $1/f$ noise of the LNA. A common solution to this problem is to use a Dicke switch architecture [11], which employs a single pole double-throw (SPDT) switch at the input of the LNA in order to continuously switch the LNA input between the antenna and a calibrated reference load. The switch is toggled at a frequency higher than the $1/f$ noise corner frequency of the receiver front-end. In this case the two detected signals (antenna and reference) are continuously subtracted, and the detector's output voltage contributions due to low-frequency gain fluctuations are canceled ($\Delta G \rightarrow 0$). With the assumption that the LNA dominates the system noise temperature, the NETD for a Dicke switch system can be reduced to [10]:

$$NETD = 2 \cdot T_{SYS} \sqrt{\frac{1}{BW \cdot \tau}} \tag{2}$$

where the factor of 2 is present because the Dicke switch is connected to the antenna for half of each cycle. The integration time is determined by the receiver's specific application, and can range from 1 to 40 ms depending on the required frame rate of the imaging system. Attaining a low NETD therefore largely requires minimizing the system noise temperature. An NETD of 0.5K or less is typically cited as the acceptable threshold for passive imaging systems [7], [8].

Unlike passive imaging systems, active imaging systems do not rely on the detection of an object's naturally emitted thermal radiation. Instead a transmitter illuminates the source object and detects the reflected power. In these systems [5], the thermal radiation becomes insignificant

compared to the reflected power of the illuminating source, and the SNR of the imaging receiver is vastly increased. For such cases, NETD is no longer a strong figure of merit, and small receiver gain fluctuations are much less of a concern.

1.3 Overview of Prior Work

W-band imaging systems have been designed and implemented in compound semiconductors since the early 1990's [9], [12]. These III-V imaging solutions are typically in the form of multi-chip modules with discrete antennas. For example, [9] uses GaAs HEMT technology to implement an LNA+detector front-end chip for a Dicke switch system, whereas [7] implements a module consisting of an InP LNA and a zero-biased Sb-heterostructure diode detector with low enough gain variation that a Dicke switch is not needed to achieve NETD < 0.5K.

Silicon processes are evolving as shown in Figure 2 and are taking over territory previously held by III-V technologies. A number of silicon-based imaging systems have been developed since 2008. These systems have been evolving quickly to include higher levels of integration and more functionality. Three examples from the NCIC Lab at the University of Irvine, California demonstrate this trend. In [4] a CMOS W-band heterodyne solution (Figure 3) is proposed to achieve higher front-end gain and to be able to implement the power detector at an IF frequency (rather than at W-band), at the expense of the need for LO generation/distribution. In [5] the Dicke switch functionality embedded within the LNA (Figure 4) is combined and integrated with back-end circuits into a single chip in order to avoid pre-gain switch insertion loss. In [6] a BiCMOS W-band 4-element focal plane array (Figure 5) is presented, including slot-folded dipole antennas, the heterodyne front-ends, and an LO synthesizer and distribution network all integrated on a single chip.

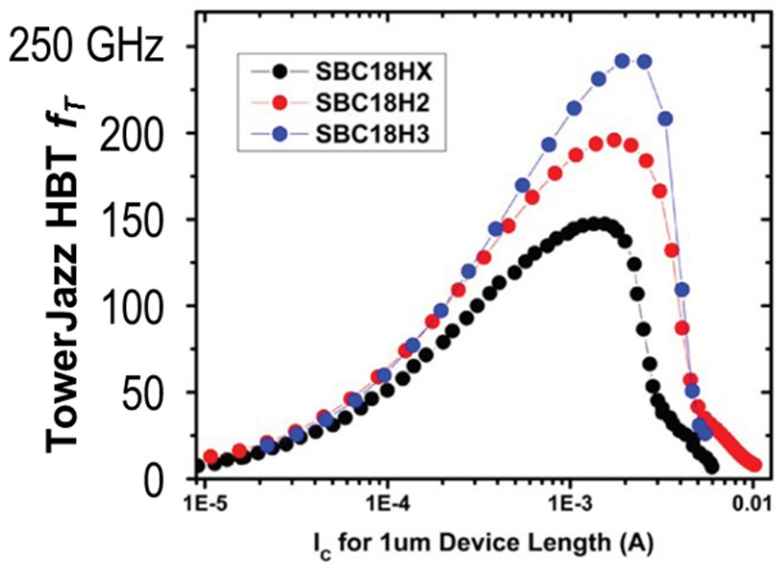


Figure 2. Evolution of HBT f_T in TowerJazz SiGe process

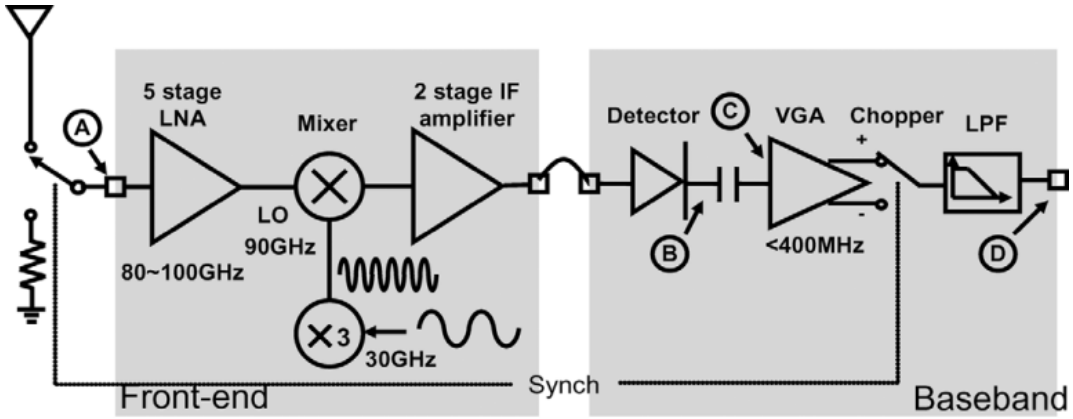


Figure 3. A W-band 65nm CMOS receiver chipset [4]

Table 1. Performance summary for receiver in [4]

Responsivity	16 MV/W	Avg. NEP	8.8 fW/Hz$^{1/2}$
Predetector Gain	35 dB	3dB Bandwidth	10 GHz
Integration Time	30 ms	NETD	1 K
Power Dissipation	102 mW	Size (mm x mm)	1.5 x 2
Technology	65nm CMOS		

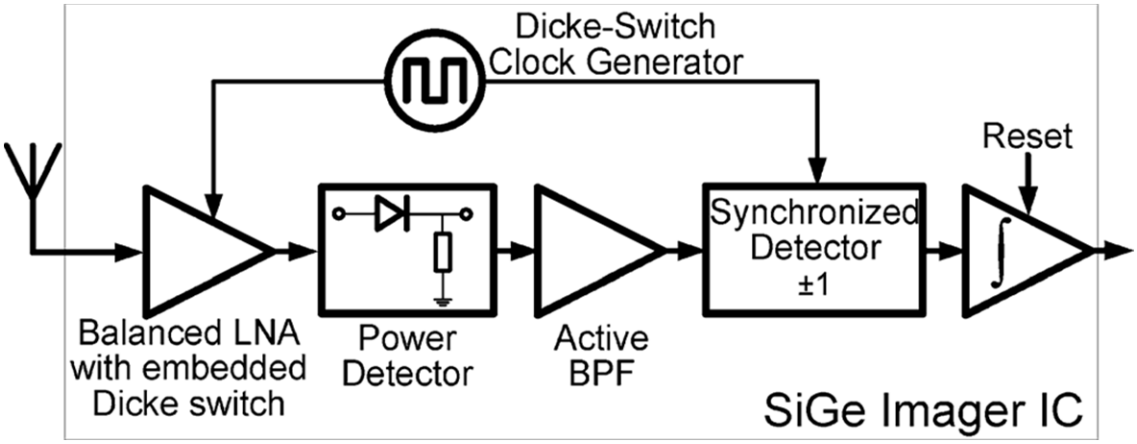


Figure 4. W-band SiGe direct-detection-based passive imager [5]

Table 2. Performance summary for receiver in [5]

Responsivity	**20-43 MV/W**	**Min. NEP**	**10 fW/Hz$^{1/2}$**
Predetector Gain	**30 dB**	**3dB Bandwidth**	**26 GHz**
Integration Time	**30 ms**	**NETD**	**0.4 K**
Power Dissipation	**200 mW**	**Size (mm x mm)**	**5 x 2.5**
Technology	**TowerJazz 0.18µm SiGe BiCMOS**		

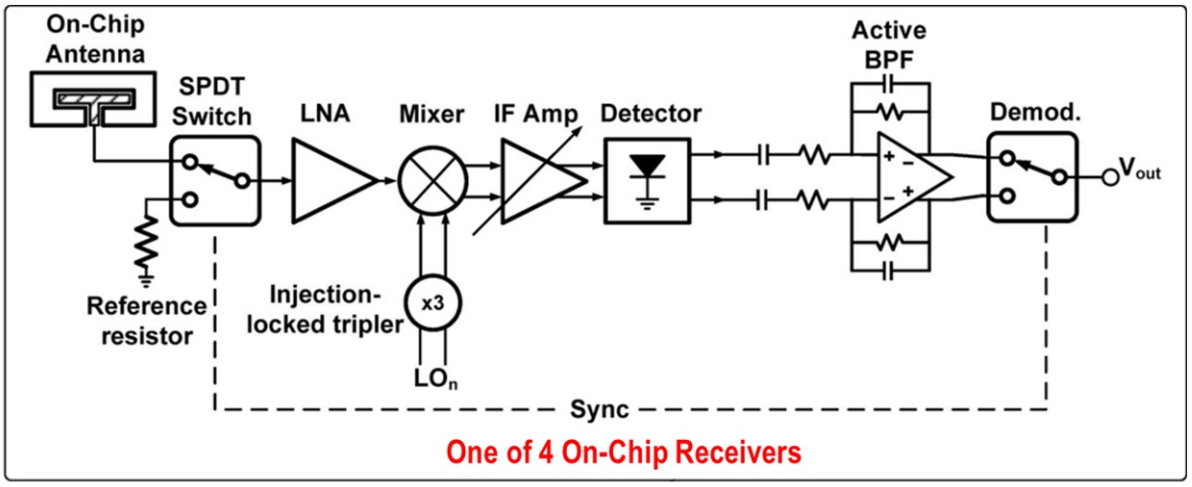


Figure 5. A BiCMOS W-band 2×2 focal-plane array with on-chip antenna [6]

Table 3. Performance summary for receiver in [6]

Responsivity	48 MV/W	NEP	50.6 fW/Hz$^{1/2}$
Predetector Gain	25 dB	3dB Bandwidth	20 GHz
Integration Time	30 ms	NETD	3 K
Power Dissipation	787 mW	Size (mm x mm)	3.5 x 3
Technology	TowerJazz 0.18µm SiGe BiCMOS		

1.4 Motivation for This Work

Receiver cost and scan time are two factors slowing the development rate of affordable practical imaging systems. The imaging receivers in widespread use today are fabricated in expensive III-V processes with very low integration levels, often resulting in scanners with low numbers of receivers to control cost. Many times the individual receivers, referred to subsequently as pixels, are arrayed in only one dimension. This leads to scanners with long scan times, as the scanner has to be mechanically scanned across the area of interest to obtain a full scan.

As the integration level goes up, the cost is expected to go down. With decreasing cost and increasing integration levels, scanners can include more pixels. Moving to cheaper silicon processes supporting high integration levels would therefore enable the development of larger, more affordable pixel arrays, effectively reducing both the scan times and cost of imaging systems. Large arrays of pixels, while minimizing scan time and reducing cost, pose new challenges for imaging receiver designers. New issues[13][14][15][16][17][18] such as coupling

among the antennas, mismatch among the RF paths and the pixel distance to the center of the focusing optics all affect the performance of the larger arrays.

The most recent prior work from NCIC lab [6] comprises a BiCMOS W-band 2×2 focal-plane array with on-chip antennas. This receiver successfully demonstrates that integrating more than one pixel per chip is possible. However, the completely integrated receiver is not sensitive enough for passive imaging when using the integrated antennas due to their low efficiency, and while the architecture in [6] works well for 2×2 focal-plane arrays, it is not amenable to arbitrarily large arrays due to the complexity of the clock distribution required for the intermediate frequency conversion. Additionally, as the size of the array increases, this architecture does nothing to address the large array issues previously mentioned.

This work continues the trends of innovation and increasingly higher levels of integration and performance in silicon-based imaging systems. A new architecture [19] addressing the integration and cost concerns, while simultaneously mitigating the issues arising from large arrays is presented. Specifically, the pre-detector gain in each RF path is increased and a new 2-stage power detector is designed which together increase the receiver responsivity to a level where integrated antennas can be used successfully. Additionally, gain and time delay controls are added to each RF path to address the large array issues.

Chapter 2

Proposed Architecture

Introduction

This chapter develops the architecture of the proposed imaging receiver array.

2.1 Pixel Development

2.1.1 Conventional Pixel and Traditional Focal Plane Array

A traditional focal plane array is implemented as an $M \times N$ array of pixels. A circuit representing one of these pixels as used in a typical imaging system is shown in Figure 6, depicting an element comprising an antenna and its associated LNA feeding a detector. A traditional focal plane array implemented as a 2×2 array of these pixels is depicted in Figure 7. Image acquisition time is an important figure of merit used to rate imaging systems. Image acquisition time for systems employing focal plane arrays is determined by several factors, including the array size, the pixel spacing, the focusing system and the desired spatial resolution of the object. If the object cannot be imaged with one scan, then decreasing image aquisition time requires one or more of the following actions such as, increasing the array size, reducing the pixel spacing, or changing the focusing system.

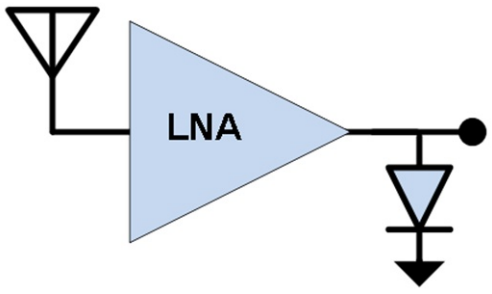


Figure 6. Typical pixel

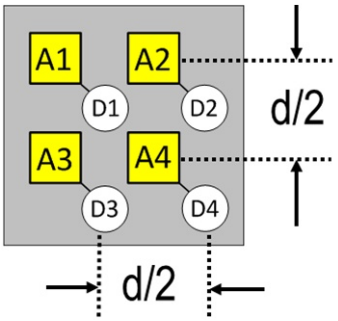


Figure 7. Traditional focal plane array

2.1.2 Improved Pixel

In the array of typical pixels shown in Figure 7, there is no control mechanism to adjust for the path gain variation or the path delay variation among the pixels, regardless of the source of the variation. An improved pixel can be created by incorporating wideband delay and amplitude control into the signal path of each element prior to the detector as shown in Figure 8. This allows both path gain and path delay adjustment of each pixel of the array to compensate for the systematic path gain and path delay variation among the pixels, ensuring equal path gain and path delay among the pixels. Although each pixel's path gain and path delay are controllable in this architecture, this architecture does not allow for beam steering of the individual pixels.

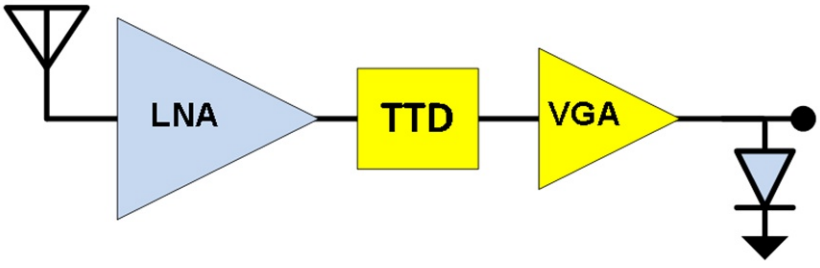


Figure 8. Pixel with wideband delay and gain control

2.1.3 Super-Pixel

A new pixel comprising a 2×2 array of elements feeding just one detector can be constructed with the pre-detector components from Figure 8. This new pixel (Figure 9) now becomes a super-pixel with unique time delay and gain weightings for each RF path which enables the super-pixel to function as a phased array, capable of beam steering in two dimensions. This allows the super-pixel to constrict the field of view of the feed antenna so that more energy illuminates the lens and less is lost to spillover, with a resulting SNR improvement dependent upon the original spillover losses.

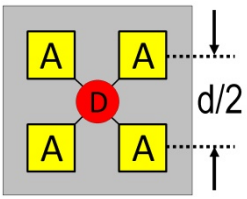


Figure 9. A 2×2 array of elements feeding one detector

2.2 Array Development

2.2.1 Conventional Method to Align Super-Pixels

A straightforward way of aligning super-pixels to form a larger array is to place each super-pixel such that the element-to-element spacing of adjacent super-pixels remains at *d*/2, as shown in Figure 10. In this case, although the element spacing remains the same when compared to the traditional focal plane array of Figure 7, the pixel spacing has doubled resulting in a physically larger array for the same number of pixels. While the optics can be chosen to match this array design, the increased physical size of the array may be a problem in some applications.

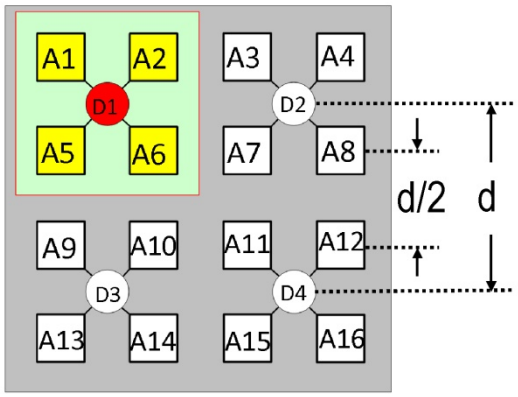


Figure 10. A 2×2 array of non-overlapping super-pixels

2.2.2 New Method to Array Super-Pixels

A hybrid architecture has been developed to take advantage of the traditional focal plane array pixel spacing and the super-pixel capability. In this new architecture (Figure 11), the super-pixels overlap and share neighboring elements, and hence, are referred to as spatial-overlapping super-pixels. Each detector in this new architecture is still fed by four elements as in a conventional array of super-pixels (Figure 10), but now each element feeds up to four detectors depending on the element's location within the array (Figure 11). This new array retains the pixel and element spacing of the traditional focal plane while incorporating super-pixel functionality.

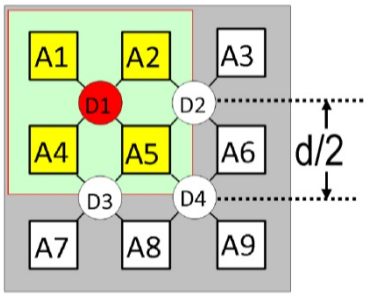


Figure 11. A 2×2 array of overlapping super-pixels

2.3 Large Arrays

In an imaging receiver, trade-off exists between scene coverage in the far field and efficiency due to spillover losses. Ideally, one would like to produce patterns that exactly fill the primary optics without losing energy on the sides (i.e., spillover) while simultaneously covering the field of view without gaps. While the feed element spacing depends on the geometry of the primary optics, specifically the F-number (defined as the ratio of the focal length to the lens diameter), the use of overlapped super-pixels will reduce spillover losses of the primary optics. Moreover, spillover and taper efficiency can be controlled by coherently combining the outputs of a number of array elements [13](e.g., four in this work), hence, the concept of overlapping super-pixels. The spatial-overlapping super-pixels can provide subarray patterns that cover the field of view (FOV), thereby reducing spillover losses of the primary optics. The spatial-overlapping super-pixels will help improve feed efficiency, while maintaining good far field coverage.

Suppose that in an $M\times N$-element array, $V_{in}(k,l,t)$ denotes the signal received by each element (k,l), $1\le k\le M$ and $1\le l\le N$; and $V_C(m,n,t)$ is the signal at the combiner's output (m,n), $1\le m\le M-1$ and $1\le n\le N-1$ (*cf.* Figure 12). From each element (k, l), four paths will be emanating to four separate super-pixels (μ,ν) where $k-1\le\mu\le k$ and $l-1\le\nu\le l$, each of which has its own amplitude and time-delay weighting coefficient, as shown in Figure 12. V_C is related to V_{in} of each element, as shown in Eq. (3):

$$V_C(m,n,t)=\sum_{k=m}^{m+1}\sum_{l=n}^{n+1}A_{k,l}^{(m,n)}V_{in}(k,l,t-\tau_{k,l}^{(m,n)}) \tag{3}$$

where $A_{k,l}^{(m,n)}$ and $\tau_{k,l}^{(m,n)}$ represent the gain and delay weights associated with element (k, l) going to super-pixel (m,n). As it is clear from (3) and Figure 12, each super-pixel V_C shares two signals

V_{in} with adjacent super-pixels. This means that each super-pixel is now "intentionally" correlated with its adjacent super-pixels, which leads to an image smoothing mechanism in the RF domain prior to power detection. This averaging mechanism can be turned off with proper selection of the element weightings. More precisely, turning off three RF paths forces the detector to process only one of the elements, thereby disabling the super-pixel functionality. In addition to an element's RF path being either on or off, there are multiple gain and delay states, which allow flexibility in signal combining.

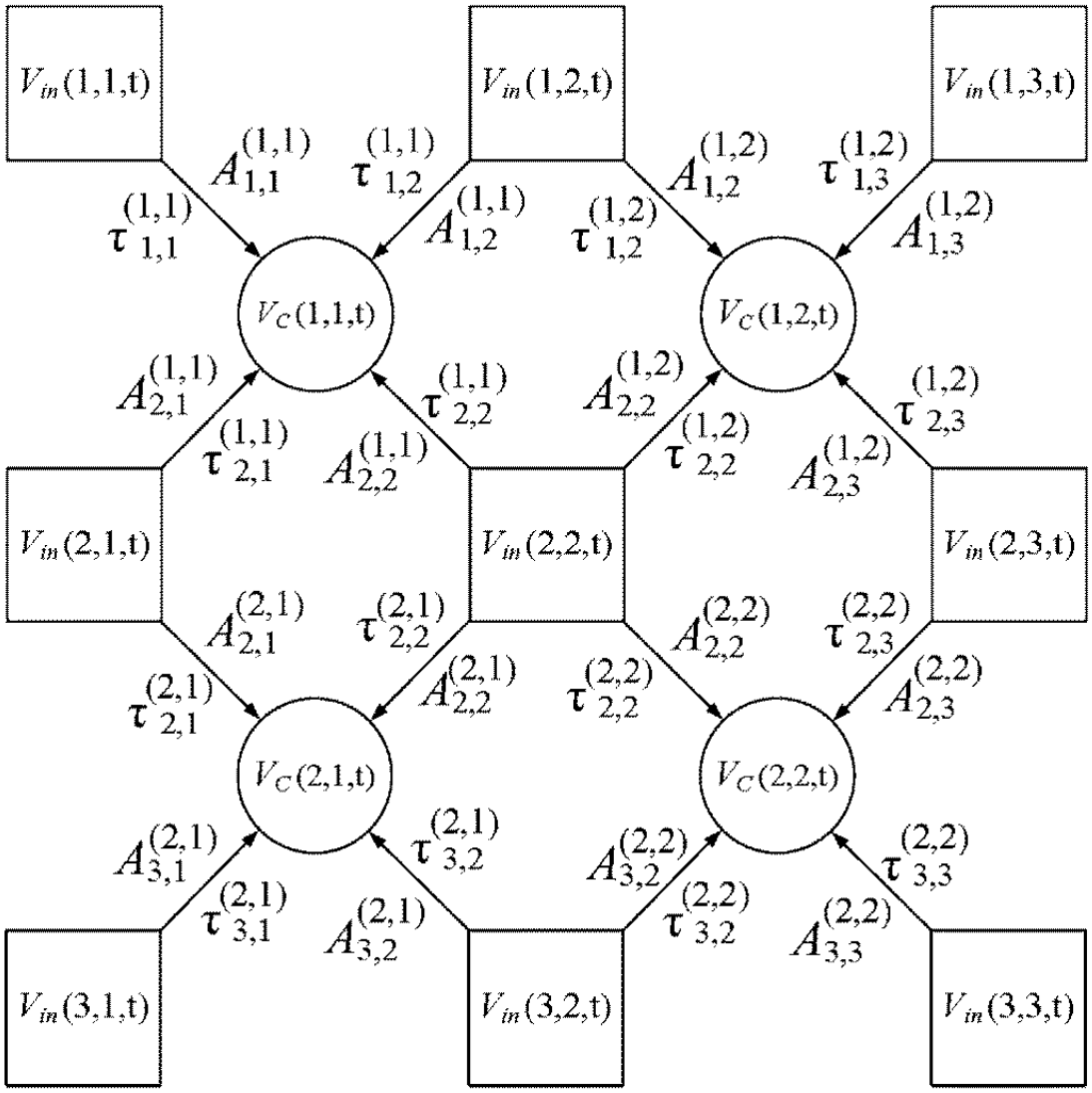


Figure 12. Amplitude and delay weighting coefficient diagram

The use of amplitude and time delay (for a wideband system) controls offer a variety of application-specific advantages: (1) In an imaging array receiver with a reflecting lens, the focus spot starts broadening out as the received beam angle moves away from the reflector boresight, resulting in a wider spot size [14]-[17]. Therefore, it is not possible to collect all the energy by just stepping through feed antennas successively further from the boresight. However, it is possible to collect the energy in the broadened focus spot by applying proper time delay and amplitude weightings to the array elements [14]-[18]. More precisely, if the focal system tries to image an object away from the main axis, the corresponding image appears distorted and loss of spatial resolution may occur. “Super-pixels” are able to compensate for these effects with time delay and amplitude adjustments and potentially increase the FOV of the focal plane array with a focusing system without resorting to mechanical scans [14], [15], [17]. (2) The concept of using a phased array within the focal plane of a reflector has been used in radio astronomy [14], [19] and array-fed reflectors [18].The primary motivation for focal plane phased array feeds (PAFs) in reflector antennas in these systems is to achieve wide fields of view with multiple, electronically steered beams so as to form a radio image with a single dish pointing at the target [14]. (3) The use of time delay and amplitude control in radio astronomy systems reduces side lobes and positions the location of nulls to reject interfering signals, which leads to an improvement of SNR. (4) The use of time delay and amplitude control allows for compensation of mutual couplings between elements, as stated in [14]. (5) For a pixel realized using a single element, directivity decreases with increasing scan angle (relative to pixels located away from the focal point). Instead, using super-pixels with the ability to control amplitude and time delay, the directivity can be kept approximately constant with increasing scan angle. Finally, to successfully leverage any correlation present among the element signals, the gain and delay of

each RF path from element to detector must be wideband and adjustable. This requires an adjustable wideband true time delay (TTD) circuit and a wideband VGA in each RF path.

2.4 Proposed Architecture

Taking all of this into consideration, a single RF path can be defined as shown in Figure 13. The 1:4 power splitter is required so that each element may feed up to four power detectors, and the 4:1 power combiner is required so that each power detector is fed by four elements. The entire RF path before the detector is constructed as a ground-shielded coplanar waveguide (CPW). The TTD circuit has been incorporated into the splitter, providing a significant area savings. A single super-pixel is therefore constructed by combining four RF paths to feed one detector. A 3-D diagram of a complete super-pixel circuit is shown in Figure 14. Noticeably missing from Figure 14 is a Dicke switch. Taking into account (1) the high pre-detector gain (i.e., 40–48dB for LNA+VGA), (2) the low 1/f noise corner typical of SiGe radiometers [20], and (3) the use of a cooling system to thermally stabilize the chip, the traditional Dicke architecture was not used in this design. However, the ability to electronically chop noise by modulating the input to the detector through software control was included as a backup. Knowing the advantage of a Dicke switch based on prior research [5]-[6], this work focuses on introducing the concept of spatially overlapping super-pixels and on designing a completely integrated system capable of passive imaging.

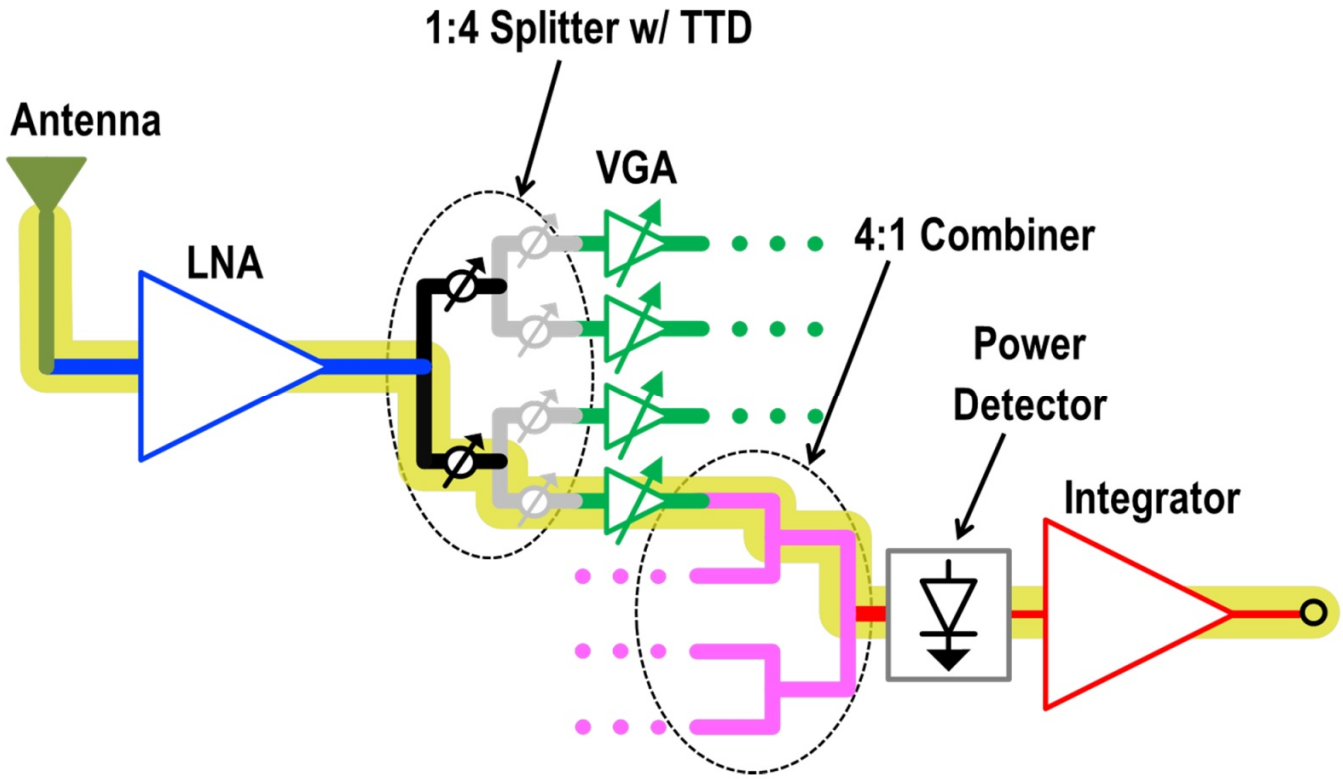


Figure 13. Single RF path circuit diagram

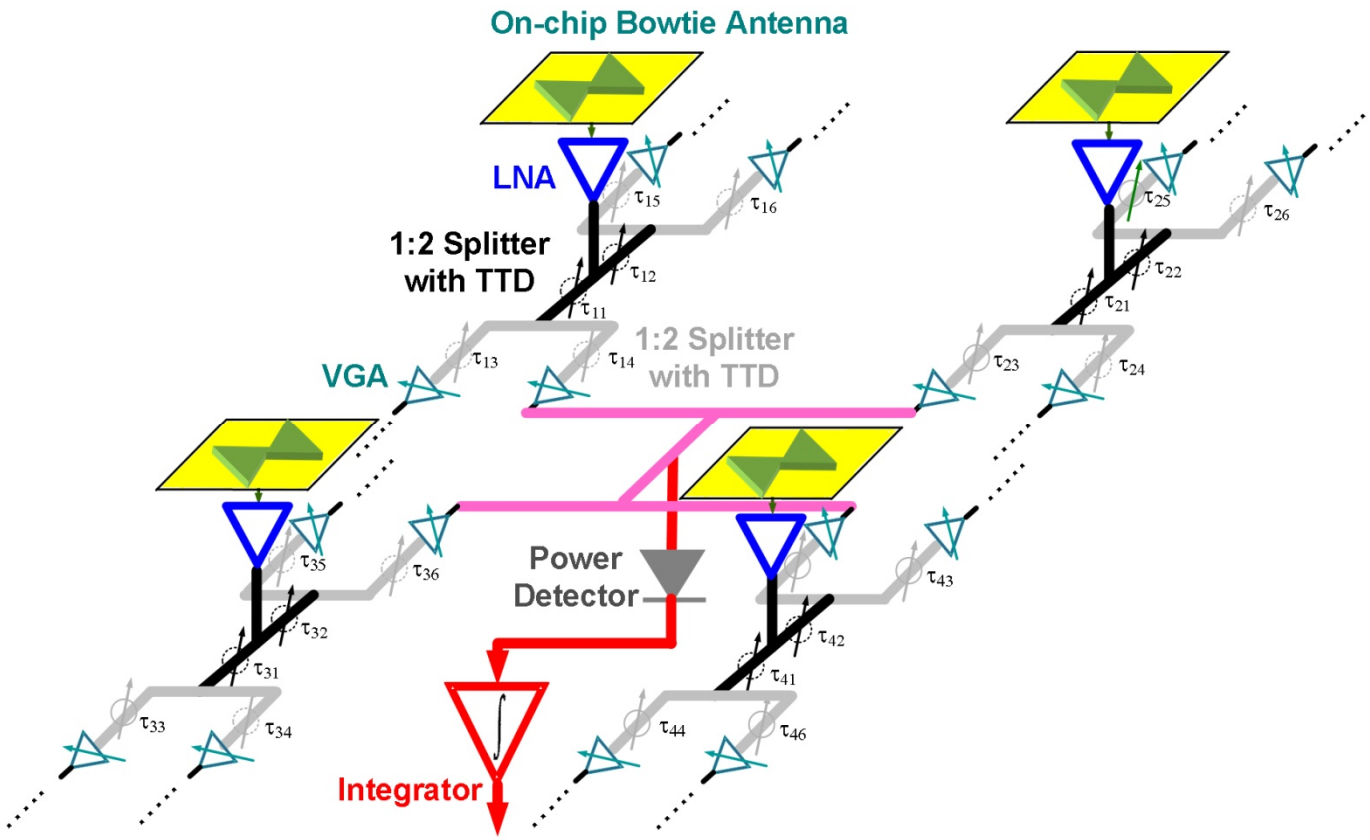


Figure 14. Single super-pixel circuit diagram

The proposed architecture is extendible to any arbitrarily sized $M \times N$ array of elements. This design implements a 3×3-element array corresponding to a 2×2 super-pixel array with spatial-overlapping super-pixels and is shown in Figure 15. The detectors corresponding to the four super-pixels are labeled 1-4. Two unused groups of combiners and detectors have been circled. They are not used in the 3×3 design, but are included for layout symmetry and to demonstrate the architecture extendability. Although the proposed architecture is suitable for implementation of imaging receivers operating at frequencies into the hundreds of GHz range, the atmospheric window centered around 94GHz is the target frequency of the proposed imaging receiver primarily due to process and test equipment considerations.

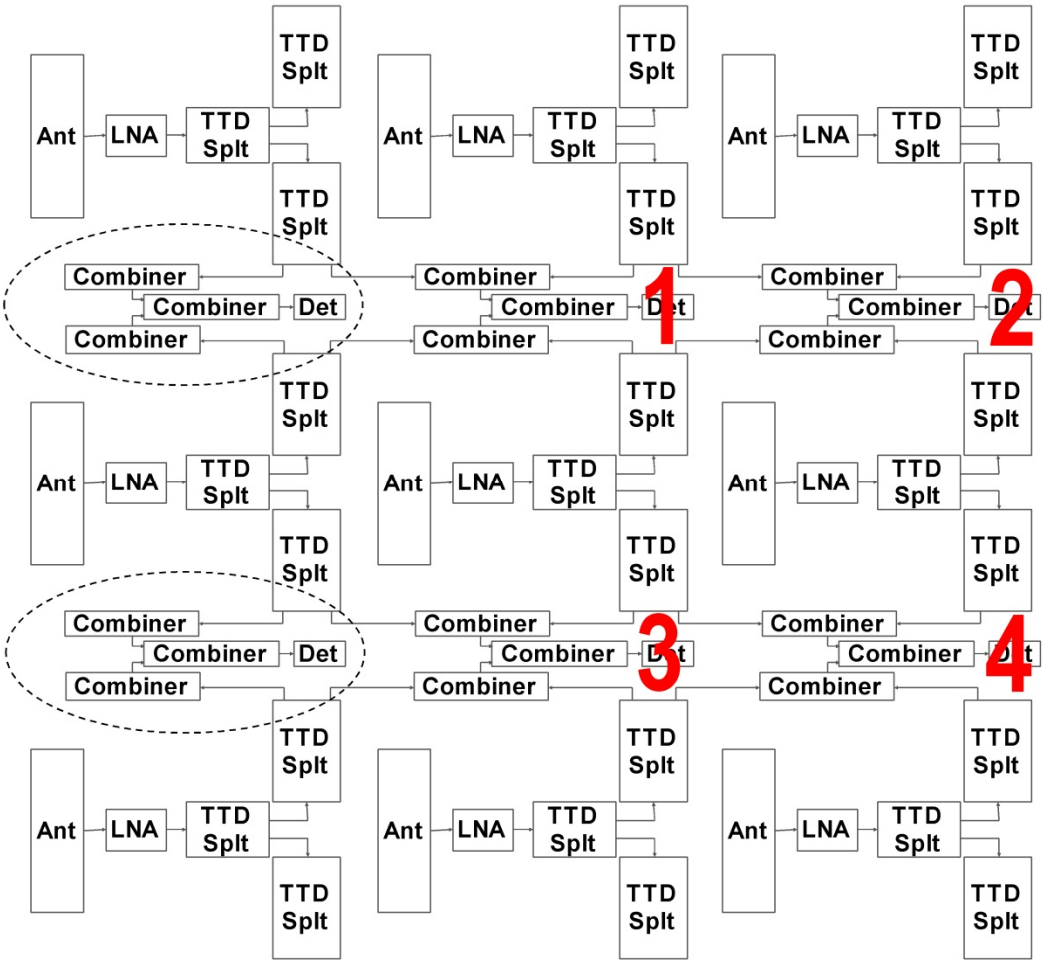


Figure 15. Complete array block diagram

Chapter 3

Circuit Design, Analysis and Testing

Introduction

This chapter details the design, analysis and measured results of the individual circuit blocks.

3.3 Fabrication Process

The TowerJazz SBC18H3 SiGe BiCMOS process was selected for this design. The SBC18H3 process technology platform integrates NPNs with f_T/f_{MAX} of up to 240/260GHz onto the dual gate 1.8V/3.3V 0.18µm RF/mixed-signal CMOS process, incorporating six metal layers and MIM capacitors. Figure 16 shows a schematic of the SBC18 process.

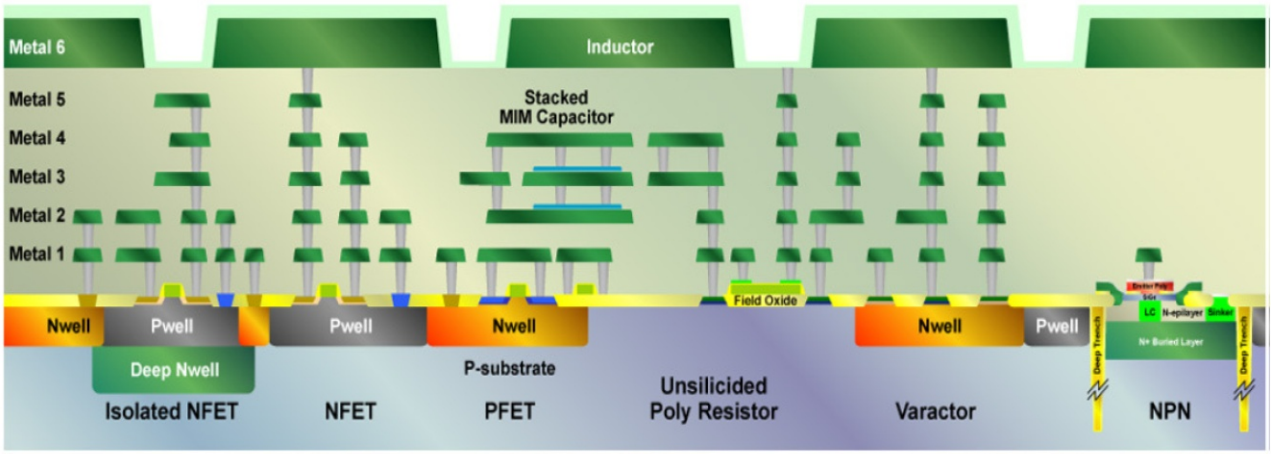


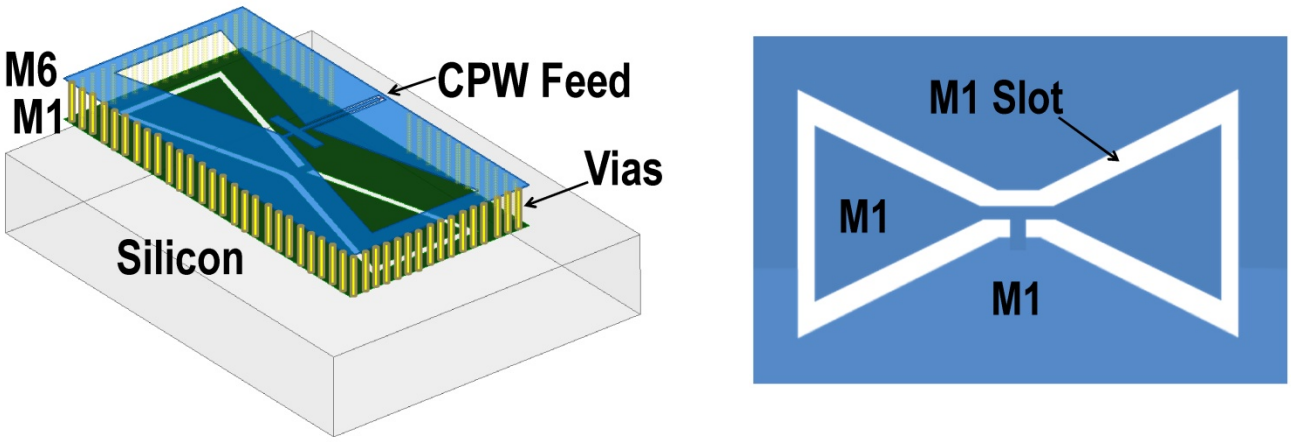
Figure 16. TowerJazz SBC18 process schematic

3.2 Design of a W-band On-chip Antenna and 9-element Antenna Array

Each receive path begins with an antenna. The 9 on-chip antennas are placed as one 3×3 array with a separation distance of 1.9mm, or 0.57λ at 90 GHz. One of the major concerns of

using silicon-based on-chip antennas [21] is the risk of interference between antennas and the on-chip circuitry, especially the passive structures (i.e., inductors). This concern is based on the assumption that the antenna is using the silicon as a substrate; however, the situation is different when the antenna is fully backed with a ground plane at the bottom metal layer (M1) and uses the interlayer dielectric as the antenna substrate. Moreover, for phased arrays, the mutual coupling between antenna elements could be destructive to the radiation pattern. In this design, the LNA provides high gain (~40dB), and the antenna's unwanted mutual coupling could adversely modulate the received signal and overwhelm (saturate) the LNA's output. The most straightforward way to reduce the coupling is a fully shielded antenna ground plane at M1. However, this could significantly affect the antenna input bandwidth [22], [23]. Therefore, a patterned ground plane at M1 is used to reduce the coupling between adjacent antennas in the array configuration. The exact dimensions of the die as fabricated were used in the full wave simulation showing negligible electromagnetic (EM) coupling between the antenna and the nearest LNA inductor.

Based on a bowtie-shape slot antenna, the on-chip antenna is built on the top metal layer of the process (M6). Beneath the slot antenna, there is a shielding structure composed of the ground plane on M1 and the vias between M1 and M6. Figure 17(a) shows the 3-D view of the proposed antenna. Figure 17(b) shows the patterned ground plane on M1, formed in the same shape and with similar dimensions as the bowtie on M6. The stress relief slots are not visible. The whole antenna is fed by a 50 Ω CPW line. A metal stub is added in the middle of the bowtie slot to provide better antenna input impedance matching. Figure 18 shows a photograph of the fabricated antenna.



(a) (b)

Figure 17. Proposed antenna. (a) 3-D view; (b) patterned ground plane on M1

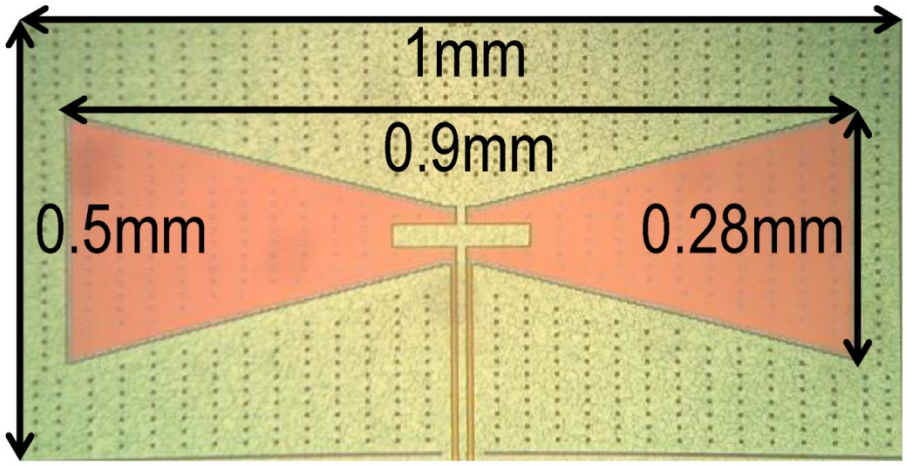


Figure 18. Photo of the bowtie slot antenna

It should be mentioned that in spite of the existence of the M1 shielding layer, EM waves still partially penetrate the shielding, propagate, and establish surface wave modes inside the silicon. To achieve high antenna gain, the silicon substrate was back-ground to a specific thickness

(220μm), which is close to a quarter wavelength at 90GHz in silicon. The quarter-wave reflection from the ground plane improves the radiation efficiency, which is around 22% according to full wave simulation.

Figure 19 shows the mutual coupling between the center antenna and one of its adjacent antennas in both the E- and H-planes. A comparison between the cases with and without shielding is provided. It can be observed that without the additional shielding structure, the mutual coupling is as high as –22dB in the E-plane and –28dB in the H-plane. Meanwhile, with the shielding structure, the coupling between two antennas is 6dB lower for both E- and H-plane cases. Figure 20 shows the positioning details of the 9 antenna elements of this work.

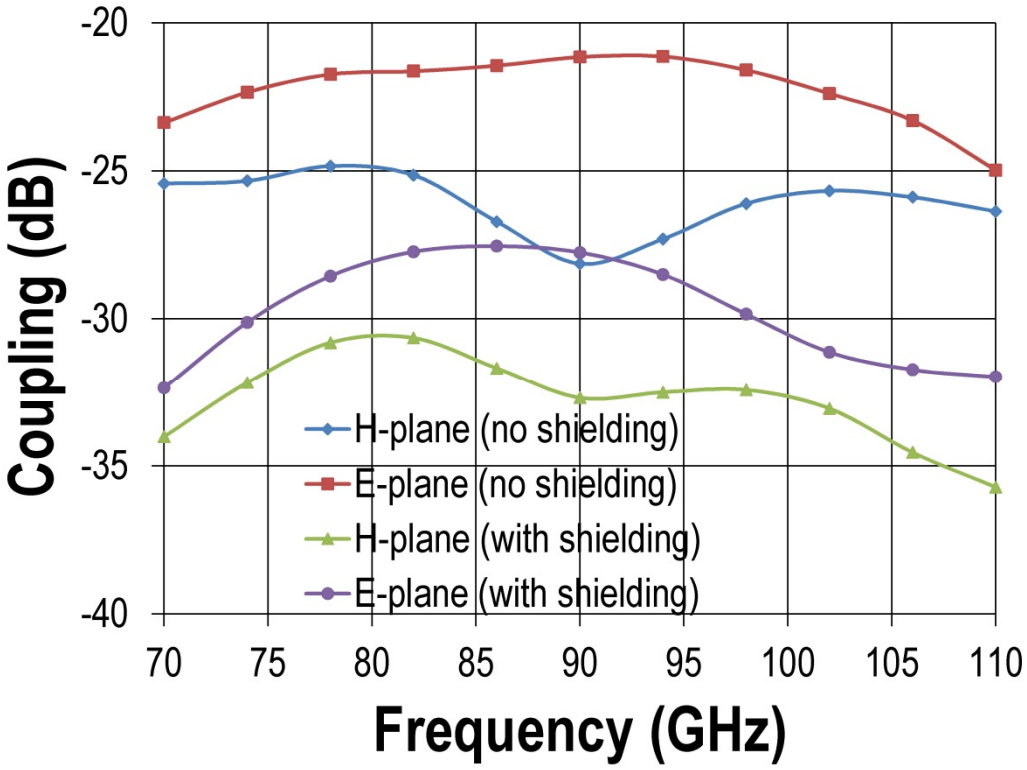


Figure 19. Simulated antenna mutual coupling

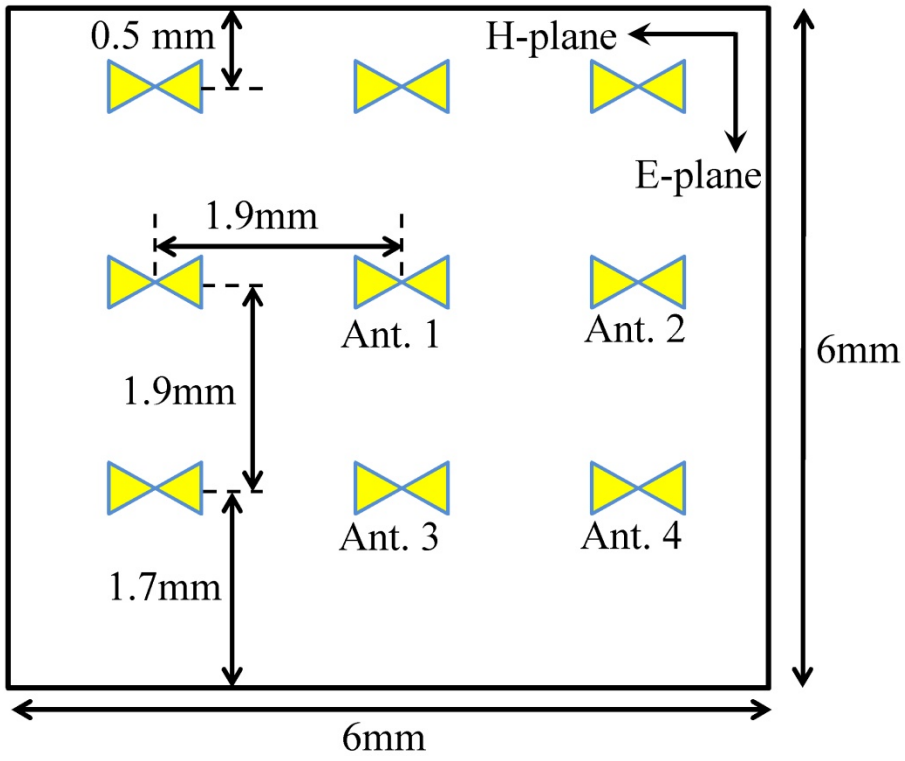


Figure 20. Antenna positioning details

Figure 21 shows the simulated broadside gain versus frequency for the four elements inside one super-pixel, where antenna 1 indicates the antenna closest to the center of the die, antennas 2 and 3 indicate the antennas adjacent to antenna 1 on the H- and E-plane, respectively, and antenna 4 indicates the antenna at the corner of the chip. It can be observed that at 90GHz, the four antennas show a very similar gain, which is close to 0dBi. Additionally, over the bandwidth from 80–100GHz, there is at most 4dB gain variation between the four antennas. D. G. Kam, *et al.*, showed examples of 5dB gain variation over different antennas [24], where a phased array was designed on an LTCC substrate at 60GHz. Indeed, due to reflection at the chip edge and interference, the actual antenna gain of four elements could still deviate from the simulation. For the proposed design, the antenna's systematic gain variation is equalized by adjusting the gain setting of the VGA.

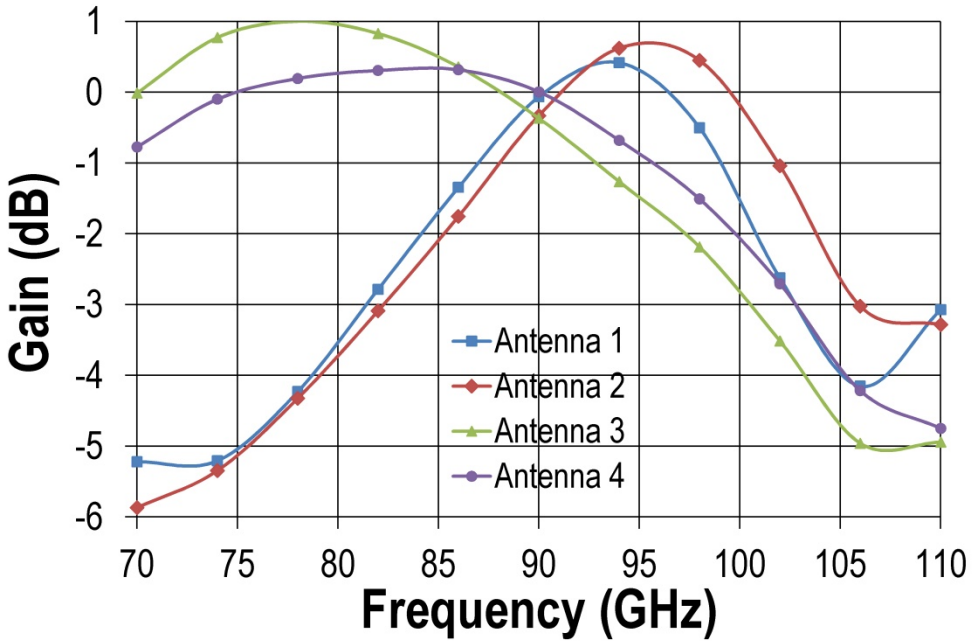


Figure 21. Simulated broadside gain vs frequency for a super-pixel

It is also noteworthy that antenna 1 and antenna 2 share similar gain over frequency patterns while the same phenomenon happens between antenna 3 and antenna 4. This means that the gain of the proposed antenna is more susceptible to the substrate dimension in the E-plane than that in the H-plane. This implies that if the array is implemented in one dimension using the proposed antenna, it is better to align the array along the H-plane such that each antenna provides similar gain regardless of its location on the die. Although the antennas exhibit gain variation, the simulated reflection coefficients of the four antennas (Figure 22) are very similar. Reflection coefficients show good matching from 80–104GHz. Each bow-tie slot antenna occupies 0.5mm^2 of chip area.

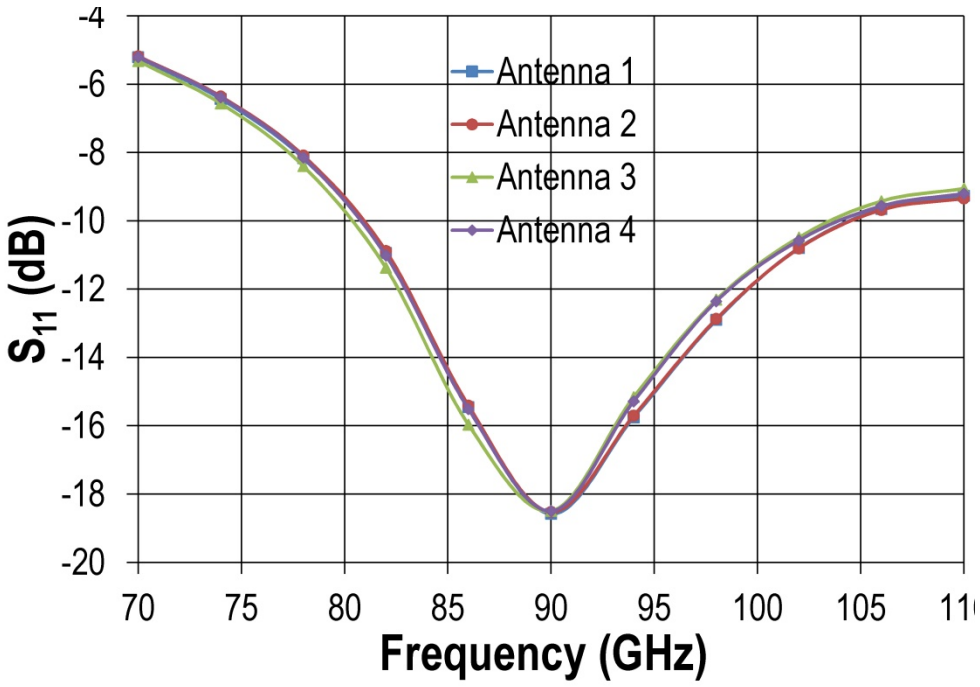


Figure 22. Simulated return loss of the four antennas labeled in Figure 20

3.3 Design of a W-band 4-stage LNA

The next block in the RF path is the LNA. The architecture and technology choices of the proposed imaging system impose several requirements on the LNA pre-detection gain and noise figure (NF). Based on the system link budget analysis designed to accommodate the sensitivity of the available lab equipment, the LNA requires approximately 40 dB of gain, 20 GHz bandwidth, and a NF of less than 8 dB. Such high gain and low NF are not easily attainable at W-band frequencies, thus a careful design approach is required. A multi-stage common emitter design was initially considered. Ultimately a four-stage cascode design was selected due to its superior high frequency response [25] (Figure 23).

The TowerJazz H3 process provides three different configurations for high-speed HBTs. They differ in their physical emitter-base-collector arrangements, specifically the number of emitter, base, and collector fingers in the physical transistor layout. The combinations for

numbers of emitter, base, and collector fingers available for selection are: 1-2-1; 1-2-2; and 2-3-2. All three geometries are evaluated at their optimum bias (emitter current density) points for minimum NF. The minimum NF, bias current, noise resistance, and maximum available gain (MAG) are simulated and compared. Simulations showed that the NFs of all three transistors are within 0.2 dB of each other, while the MAG of the 1-2-2 transistor is 0.6 dB higher than the others. Therefore, the 1-2-2 transistor is selected for the design with an emitter area of $0.13\times5\mu m^2$ based on simulation results. The transistor is laid out in Cadence and the parasitics are extracted in Calibre with the R+C+CC options. This extracted transistor is used in all amplifier simulations.

Additionally, several techniques are used that, collectively, further increase the gain, widen the bandwidth and reduce the NF: (1) transmission line (T-line) inductors T_{CAS} are used to lower the noise contribution of the common-base transistor in each cascode cell and widen the bandwidth[26]; (2) the base T-line T_{BASE} is designed carefully to widen the bandwidth while maintaining unconditional stability (k-factor>20); and (3) cascaded two high-pass L-match interstage matching networks with two resonant frequencies are designed to achieve wideband inter-stage matching.

\

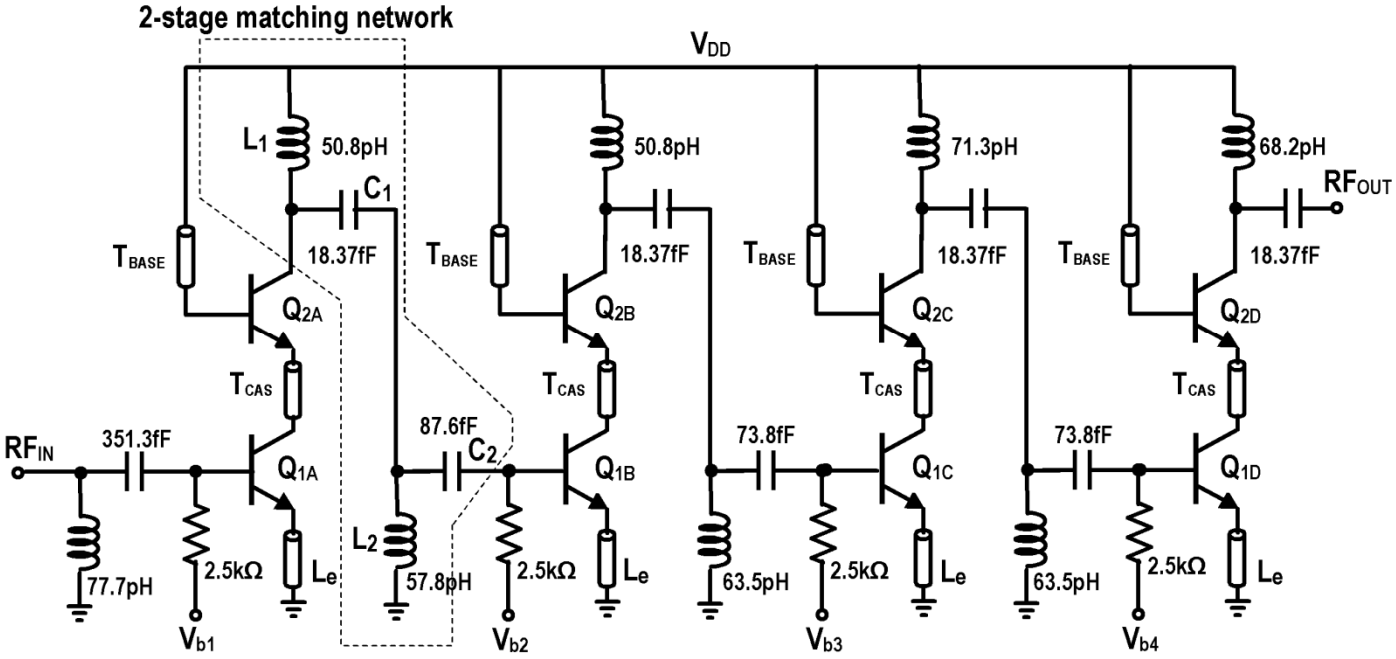


Figure 23. Four-stage LNA schematic

Figure 24(a) shows the common base stage of the cascode amplifier with base inductor L_{BASE} and base resistor r_b. In [27], the input impedance seen looking into the emitter terminal of a transistor with an inductor-degenerated base is

$$Z_{in} = \frac{r_b + r_\pi + j\omega(L_{BASE} + r_\pi C_\pi r_b) - \omega^2 r_\pi C_\pi L_{BASE}}{1 + g_m r_\pi + j\omega r_\pi C_\pi} \tag{4}$$

which can be rewritten as

$$Z_{in} \cong \frac{1}{g_m} - \frac{\omega^2 L_{BASE}}{\omega_T} + \frac{\omega^2 r_b}{\omega_T^2} + j\omega\left[\frac{r_b}{\omega_T} - \frac{1}{g_m \omega_T} + \frac{\omega^2 L_{BASE}^2}{\omega_T^2}\right]$$

$$= R_{EFF} + j\omega L_{EFF} \tag{5}$$

Using (5), Figure 24(a) is redrawn for analysis. The resulting circuit in Figure 24(b), which includes the cascode node parasitic capacitance C_{CAS}, resembles a shunt-peaked amplifier [28].

The extra bandwidth obtained from shunt-peaking comes at the cost of stability as this inductor introduces a negative resistance term to the real part of Z_{in}. The condition for stability then is given by

$$\Re(Z_{in}) = \frac{1}{g_m} - \frac{\omega^2 L_{BASE}}{\omega_T} + \frac{\omega^2 r_b}{\omega_T^2} \geq 0 \tag{6}$$

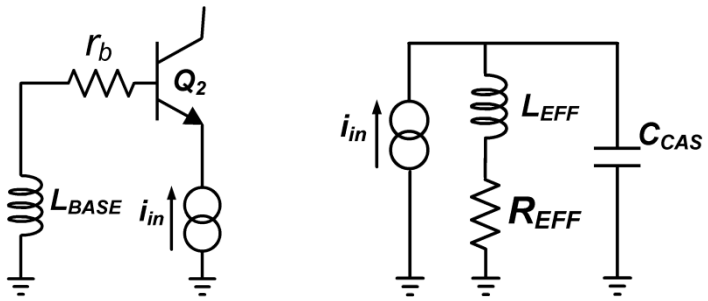
Or equivalently

$$L_{BASE} \leq \frac{1}{\omega_T}\left(\left(\frac{\omega_T}{\omega}\right)^2 \frac{1}{g_m} + r_b\right) \tag{7}$$

With f_T =240GHz, g_m=0.2mA/V and r_b=10Ω, L_{BASE} is then found to be 26.2pH at the onset of instability. The inductance value providing maximum bandwidth enhancement is calculated as [28]:

$$L_{EFF} = \frac{R_{EFF} C_{CAS}}{\sqrt{2} / R_{EFF}} \tag{8}$$

Again using f_T =240GHz, g_m=0.2mA/V, r_b=10Ω and C_{CAS}=10fF, $L_{BASE} \cong 51.6\,\text{pH}$ at the maximum bandwidth. Comparing the two values of L_{BASE} shows that the amplifier will enter the instability region before achieving maximum gain and bandwidth.



(a) (b)

Figure 24. Common base amplifier (a) Transistor-level schematic (b) Analysis schematic

Wide bandwidth multistage amplifiers require wide bandwidth low Q interstage matching networks. Single-stage L-match networks do not allow specification of the Q when the transformation ratio and resonant frequency are specified [28]. However, cascading these networks and transforming to an intermediate impedance allows enough degree of freedom to design a matching network with the desired bandwidth since the Q, transformation ratio and resonant frequency of each stage can be independently adjusted. One-stage high-pass L-match networks were cascaded in this design to form 4th-order high-pass networks (Figure 23). The two-stage matching network analysis schematic is shown in Figure 25.

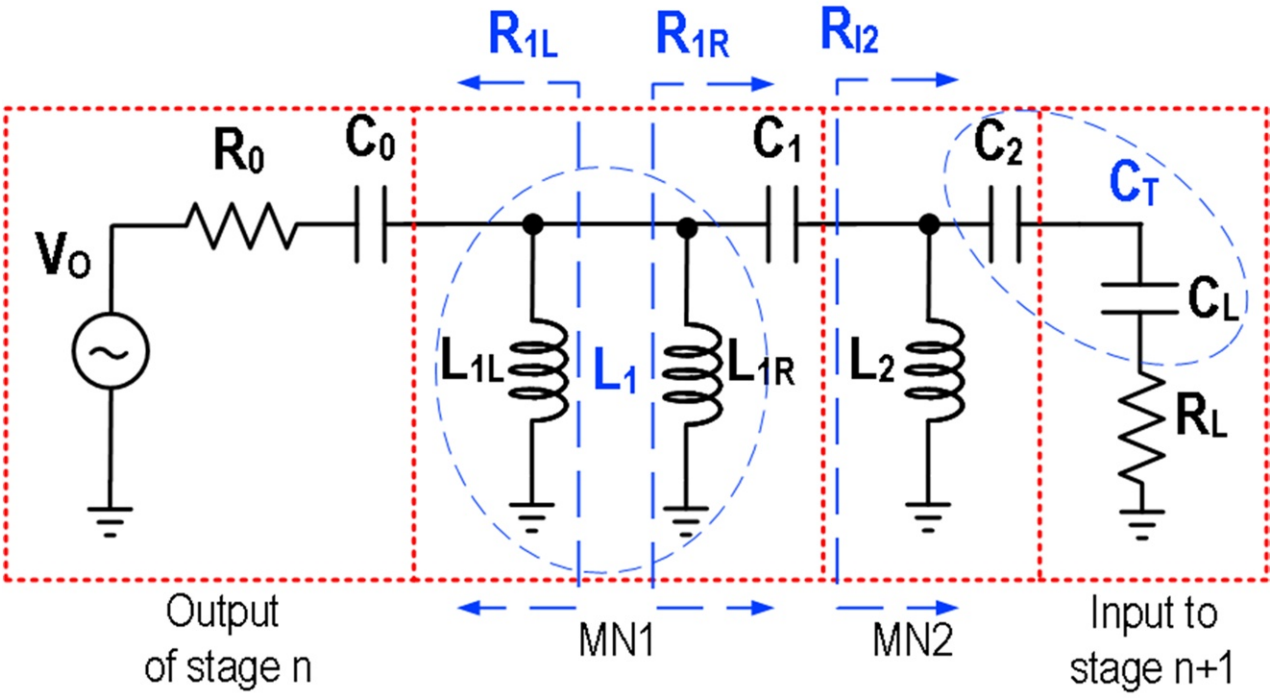


Figure 25 Two-stage matching network analysis schematic

In the analysis schematic above, inductor L_1 has been decomposed into two parallel inductors L_{1L} and L_{1R}. Capacitors C_2 and C_L have been combined in series to form C_T. The resonant frequencies of each matching network, ω_1 and ω_2, are defined by

$\omega_1 = \frac{1}{\sqrt{L_1(C_O + C_1)}} = \frac{1}{\sqrt{L_{1L}C_O}} = \frac{1}{\sqrt{L_{1R}C_1}}$ and $\omega_2 = \frac{1}{\sqrt{L_2(C_2 \| C_L)}} = \frac{1}{\sqrt{L_2 C_T}}$. Q_2 of matching network 2 is defined by $Q_2 = \frac{\sqrt{L_2/C_T}}{R_L}$. For matching network 1, looking left, $Q_{1L} = \frac{\sqrt{L_{1L}/C_O}}{R_O}$. Looking right, $Q_{1R} = \frac{\sqrt{L_{1R}/C_O}}{R_{I2}}$, where $R_{I2} = R_L(1+Q_2^2)$. Putting this all together the first stage Q looking right is given by

$$Q_{1R} = \frac{\sqrt{(C_o + C_1)L_1}}{C_1 R_L\left(1 + \frac{L_2}{C_T R_L^2}\right)} \tag{9}$$

the second stage Q is

$$Q_2 = \sqrt{\frac{L_2}{C_T}} \Big/ R_L \tag{10}$$

Plots of one- and two-stage interstage matching networks' frequency responses are shown in Figure 26. As verified in Figure 26, with appropriate selection of component values the two-stage interstage matching network bandwidth can be improved by over 100% compared to one-stage interstage matching network bandwidth. These components control ripple in the pass band and can provide a bump in gain at one band edge or the other. High-pass networks were chosen for the L-match networks in the belief that transistor and interconnect parasitics would ultimately limit the high corner frequency of the LNA.

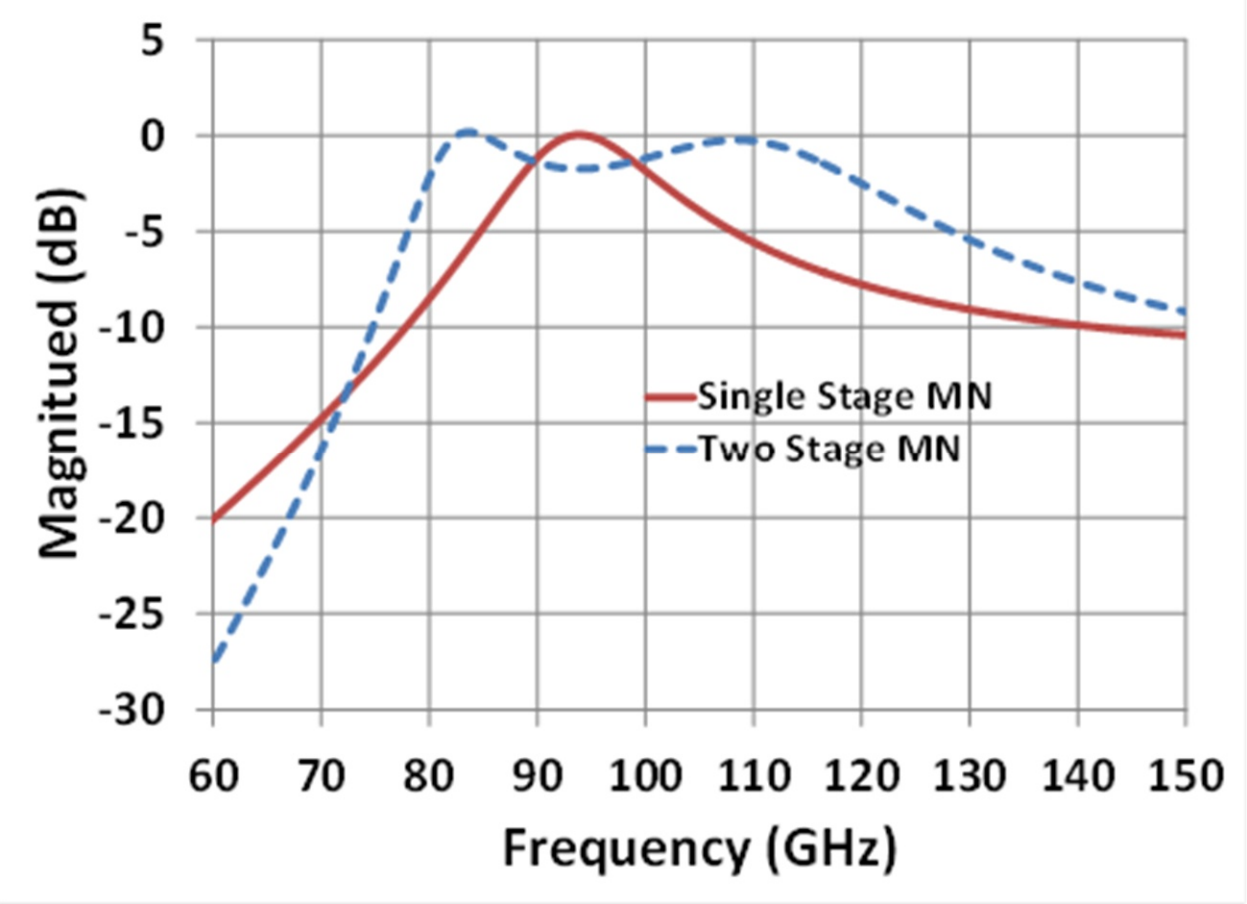


Figure 26. Frequency responses of one-stage and two-stage matching networks

The LNA's simulation and measurement results are shown in Figure 27. The measured LNA gain is 39dB with a 20GHz bandwidth from 93-113GHz. The measured in-band S_{11} and S_{22} are better than -14dB and –8dB across the band, with the exception of an S_{22} peak at 110 GHz. However, no oscillations are detected at this frequency and the measured *k*-factor is above unity. Simulation results indicate that the LNA's NF varies from 7 to 9dB and previously measured NFs of LNAs designed in this process [5], [6] have all shown great agreement between simulation and measurement. Figure 28 shows the die photo of an LNA breakout. The LNA occupies 320×200μm^2 and draws 19mA from a 1.8V power supply.

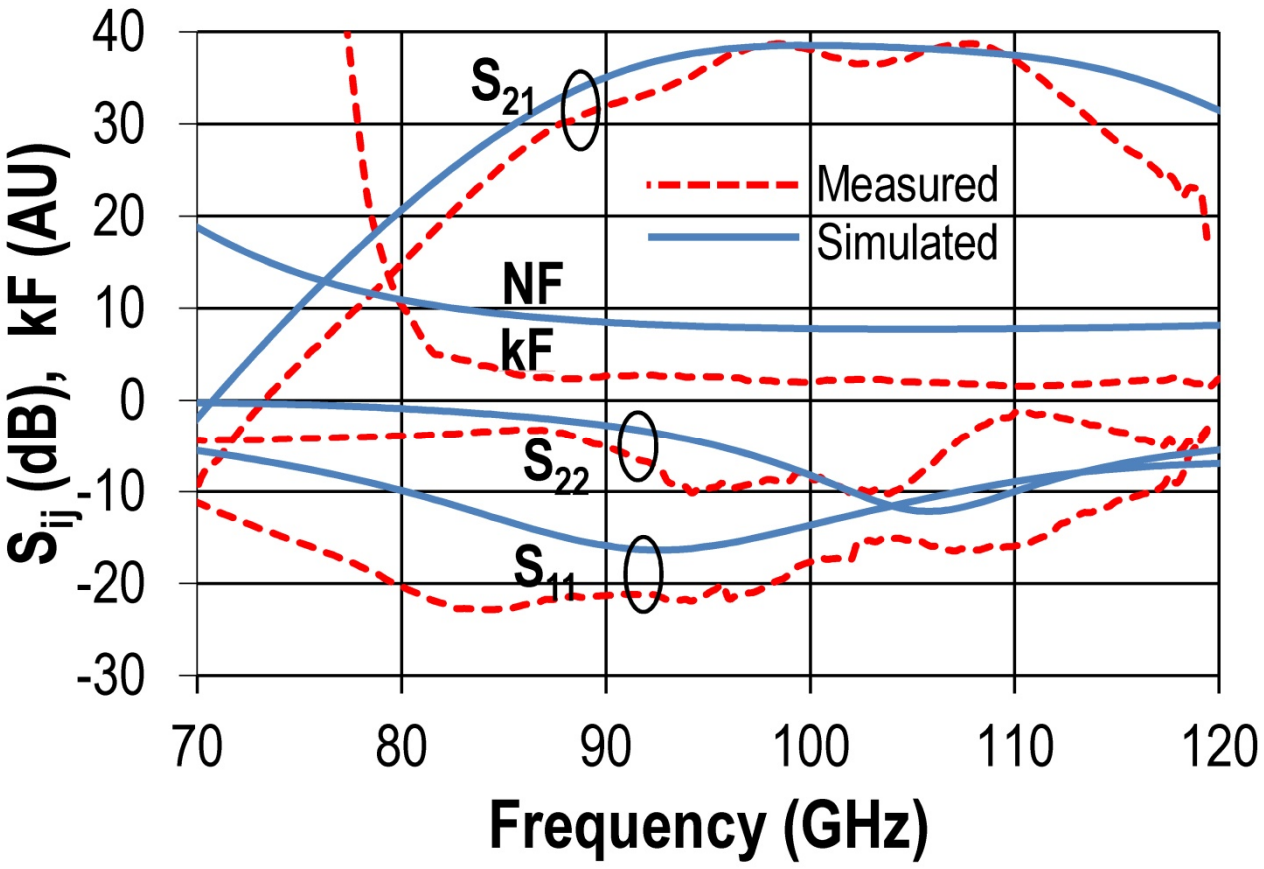


Figure 27. Measured and simulated LNA s-parameters

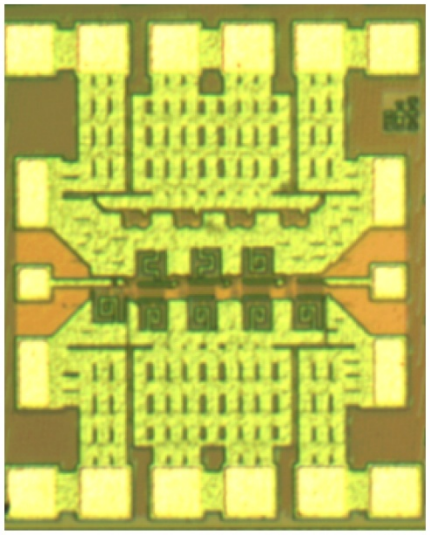


Figure 28. Die photo of the LNA breakout circuit

3.4 Design of a W-band Power Splitter with Built-in True Time Delay Circuit

As illustrated in Section II, each element of the proposed imaging array receiver of Figure 13 feeds up to four detectors via a wideband 1:4 power splitter. Therefore, the super-pixels in this imaging system need to provide a wideband delay control mechanism. Owing to inherently wideband characteristics, a distributed architecture is considered as a favorable choice for the realization of the TTD circuit [26], [29]. Shown in Figure 29 is an example of the distributed TTD circuit comprising three unit cells with variable delay distributed along CPW-based T-lines. Although the prototype unit cell is based on [29], there are distinct differences between them, namely, the inductors are replaced with CPW T-lines, a series capacitor is added at the base of the g_m transistor Q_3, and an additional T-Line T_e is added at the common emitter nodes of the differential pair to increase bandwidth by resonating out the undesired parasitic capacitance at these nodes. The unit cell's output phase is [29]:

$$\angle V_{o1} = -\tan^{-1}\frac{\sin(\omega\tau_d)}{a_2/a_1+\cos(\omega\tau_d)} \tag{11}$$

with

$$\tau_d = \sqrt{L_{eff}C_{eff}}\ ,\ a_1 = \frac{g_{m1}}{g_{m1}+g_{m2}}\ ,\ a_2 = \frac{g_{m2}}{g_{m1}+g_{m2}} \tag{12}$$

where τ_d denotes the unit delay of each unit cell [L_{eff} and C_{eff} are the inductance and capacitance of the T-line section within the cell, respectively], a_1 and a_2 are current gains of the differential pair transistors ($a_1 + a_2$=1). Hence, the group delay T_{delay} of this unit cell is

$$T_{delay}(\omega) = -\frac{d\angle V_{o1}(\omega)}{d\omega} = \tau_d\frac{1+a_2/a_1\cdot\cos(\omega\tau_d)}{1+(a_2/a_1)^2+2a_2/a_1\cdot\cos(\omega\tau_d)} \tag{13}$$

For flat delay across frequency, the group delay should be frequency independent, i.e.,

$$\frac{dT_{delay}(\omega)}{d\omega} = 0 \tag{14}$$

$$\Rightarrow \tau_d^2 \frac{-a_2/a_1 \sin(\omega\tau_d)\cdot\left(a_2^2/a_1^2 - 1\right)}{\left[1+\left(a_2/a_1\right)^2 + 2a_2/a_1 \cdot \cos(\omega\tau_d)\right]^2} = 0 \tag{15}$$

$$\Rightarrow a_1 = 0, \text{ or } a_2 = 0, \text{ or } a_1 = a_2.$$

Eq. (15) indicates that only three bias conditions (a_1=0, a_2=0 or a_1=a_2) for the differential pair of Figure 29 will lead to a broadband flat group delay. Furthermore, because the condition a_1=a_2 is more vulnerable to process/voltage/temperature variation, binary tuning (a_1a_2=01 or a_1a_2=10) is used to realize binary broadband delay (0 or τ_d). The delay of each cell within the distributed TTD is controlled by complementary digital signals applied at the inputs of the differential pair in each cell. This ensures that for all delay settings the g_m transistor's DC current and parasitic capacitor C_π will remain constant, and in so doing maintain constant characteristic impedance of the input/output T-lines.

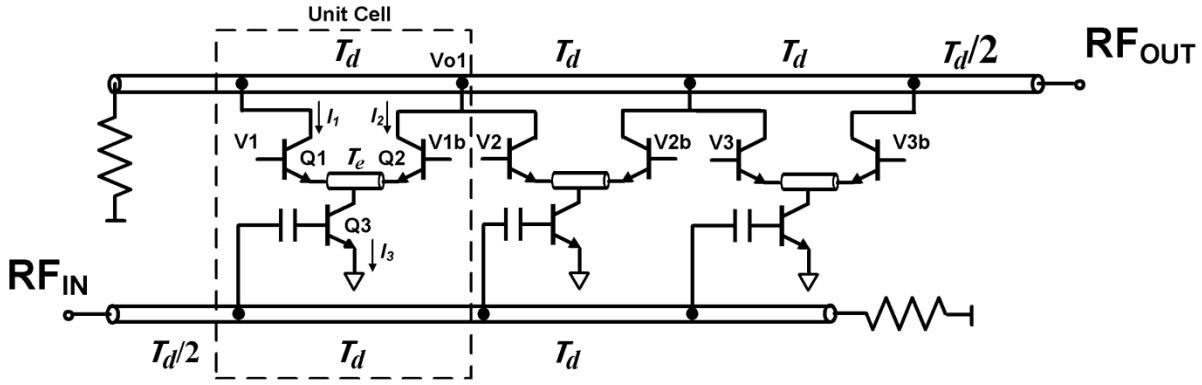


Figure 29. Three-stage distributed true time delay circuit

A 1:2 distributed active power splitter with TTD control (Figure 30) is constructed by sharing the input base T-lines of two identical distributed TTD circuits, while a 1:4 power splitter with TTD control (Figure 31) is constructed by cascading two 1:2 power splitters. The input power is equally split among the 4 paths, with independent delay control of each path. The circuit layout of this power splitter is designed symmetrically; therefore, the routing loss through the splitter is the same for every path.

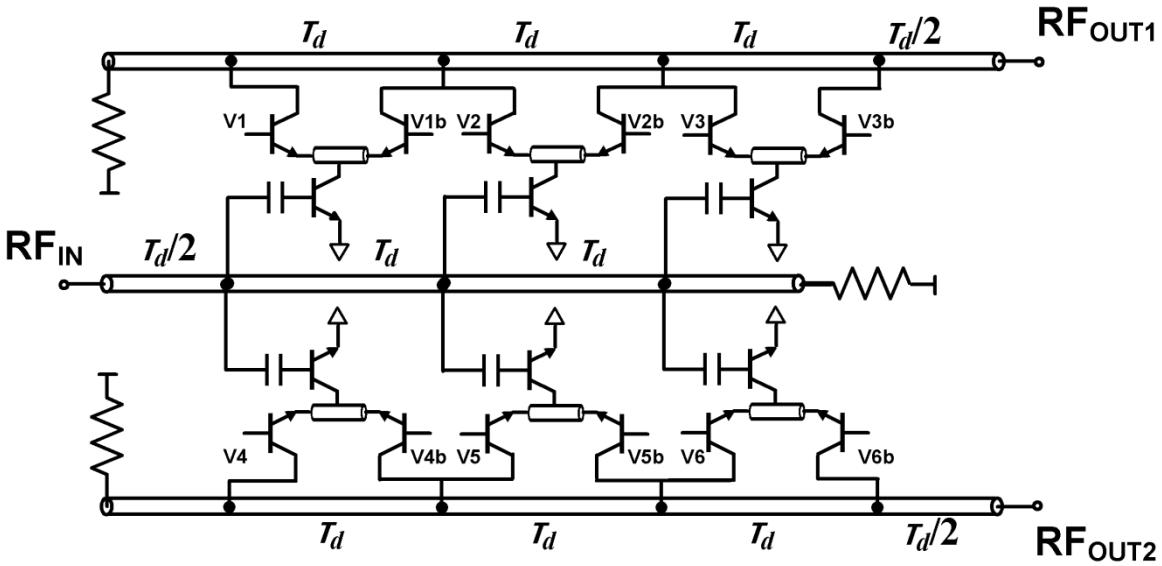


Figure 30. 1:2 Active power splitter with true time delay control

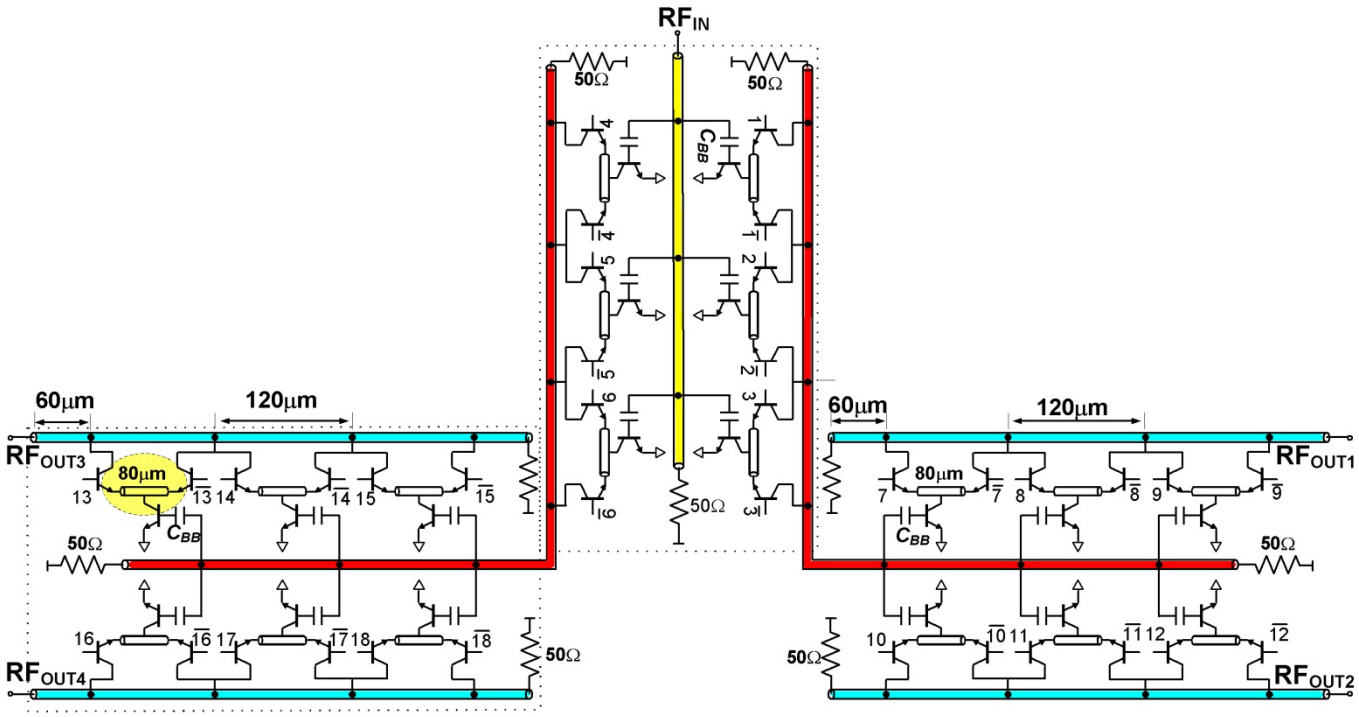


Figure 31. 1:4 Active power splitter with true time delay control

A breakout circuit of the TTD/splitter block is shown in Figure 32. The s-parameter measurements (Figure 33) are done using on-wafer probe testing and a VNA, and show less than 3dB gain roll-off across all the delay settings from 80–105GHz; absolute gain is -2.2dB ±0.6dB at 100GHz. The gain variation across all delay states is compensated by a subsequent VGA stage; therefore the gain variation does not significantly affect the antenna pattern. Figure 34(a) shows the measured phase response with respect to frequency for the distributed active power splitter and true time delay breakout circuit along with best-fit straight lines for 7 different delay settings. This measured linear response for each delay setting across the 80–105GHz range demonstrates the nearly constant group delay versus frequency characteristic for each delay setting. The error of the measured delay with respect to the best-fit straight line data versus frequency across the 80–105GHz range is shown in Figure 34(b). The measured wideband

variable delay of each path is controllable from 0ps to ±2ps with 0.5ps steps, corresponding to 0° to ±18° for beam steering with 4.5° steps.

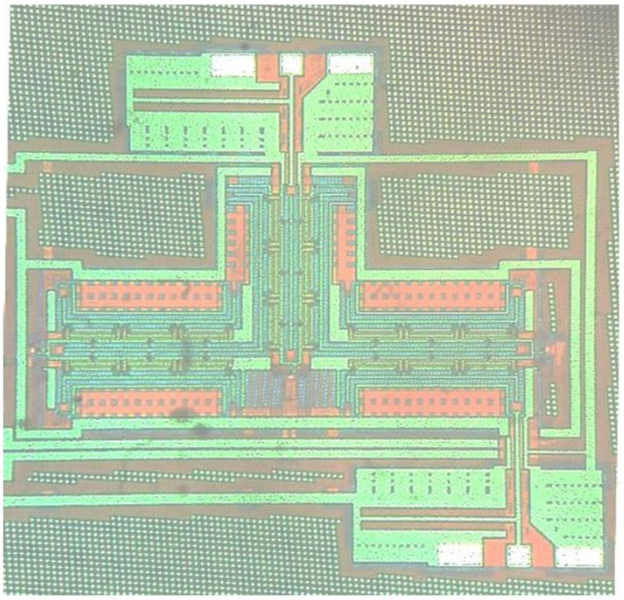
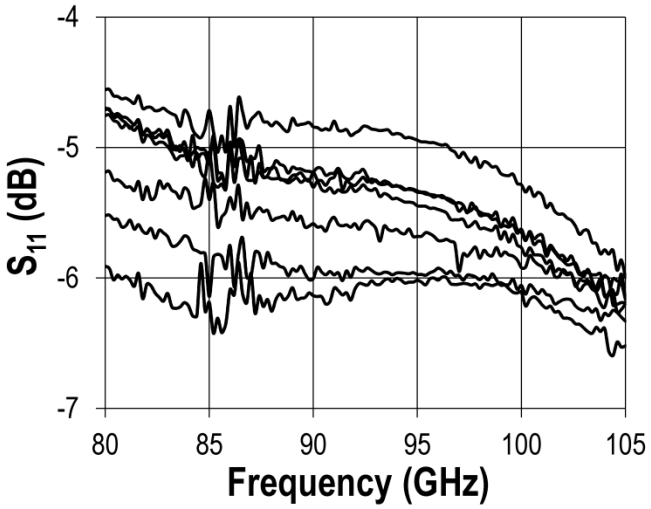
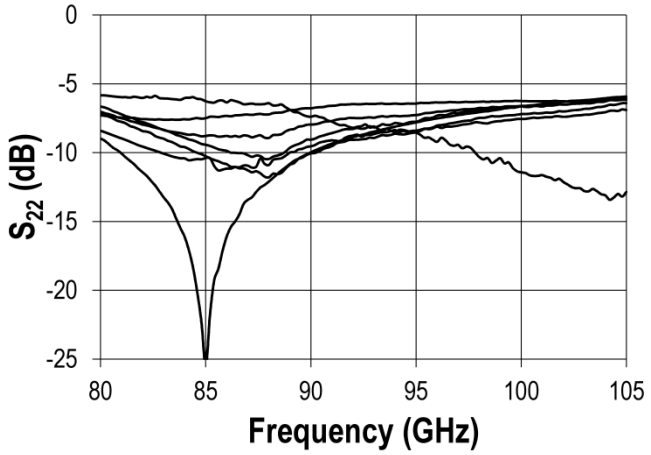


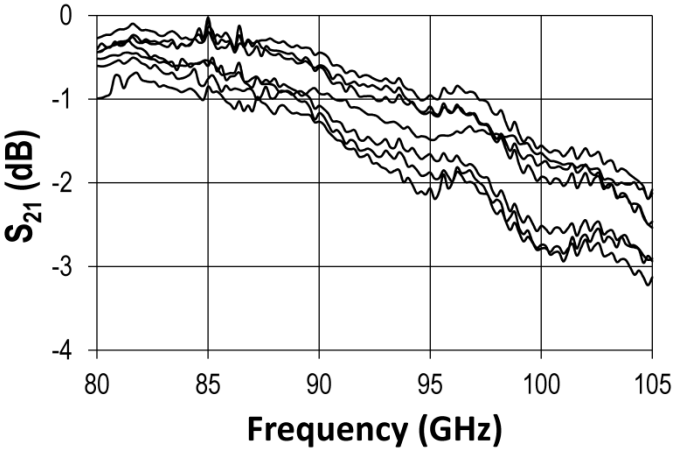
Figure 32. Photograph of the 1:4 active power splitter with true time delay control



(a)

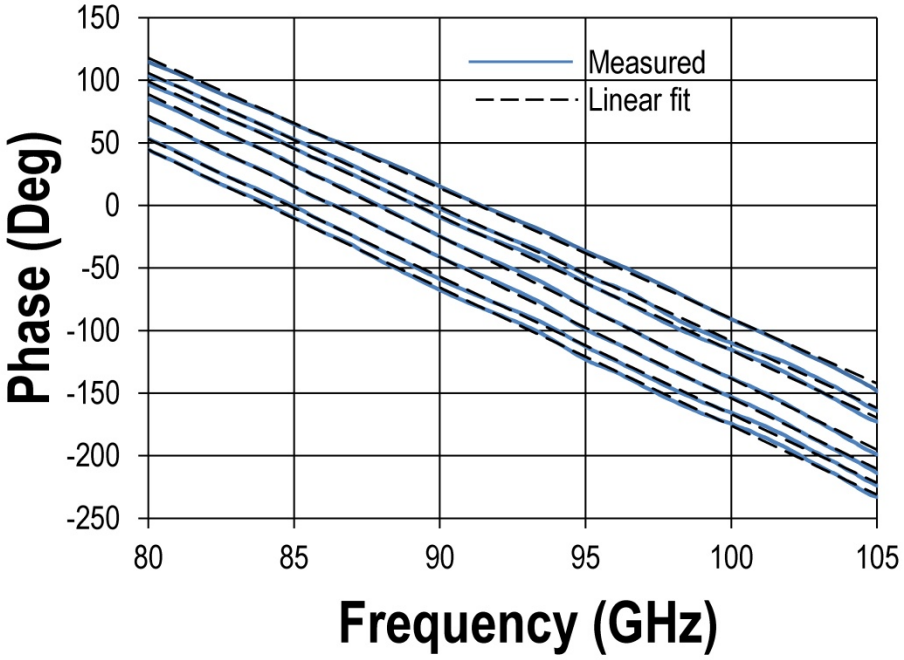


(b)

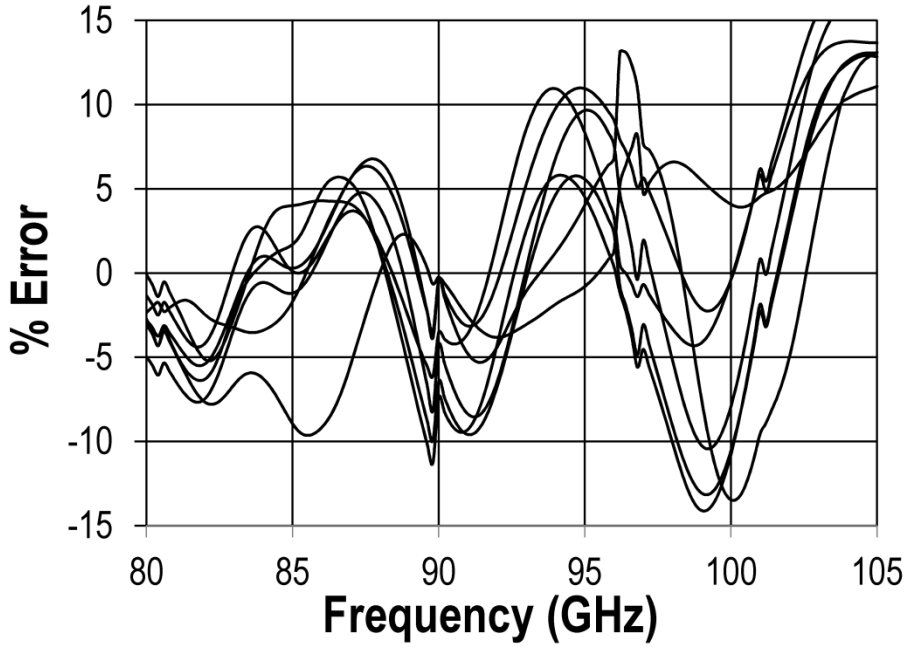


(c)

Figure 33. Measured true time delay circuit s-parameters across all delay states. (a)S11 (b)S22 (c)S21



(a)



(b)

Figure 34. Measured phase vs. frequency response of the distributed active power splitter with true time delay for 7 delay settings. (a) Measured phase vs. frequency response and best-fit straight lines. (b) Error of measured delay with respect to best-fit data versus frequency

3.5 Design of the VGA

Referring again to Figure 13 it is shown that a VGA is included in each RF path after the 1:4 wideband active power splitter and true time delay circuit. As discussed in section C, the VGA must be able to compensate for the gain variation across the delay states of the TTD, ensuring that any gain variation does not significantly affect the antenna pattern. This requires the VGA to have enough gain control to compensate for the expected variation and that it must be wideband. The designed gain control is 3 to 9dB gain in 1dB steps, digitally controlled, while the target bandwidth is greater than 35GHz. An off-state is included to aid in RF path calibration. The VGA is a wideband cascode design based on the LNA gain stage. The VGA schematic is presented in Figure 35, while the simulation results are depicted in Figure 36 and 37.

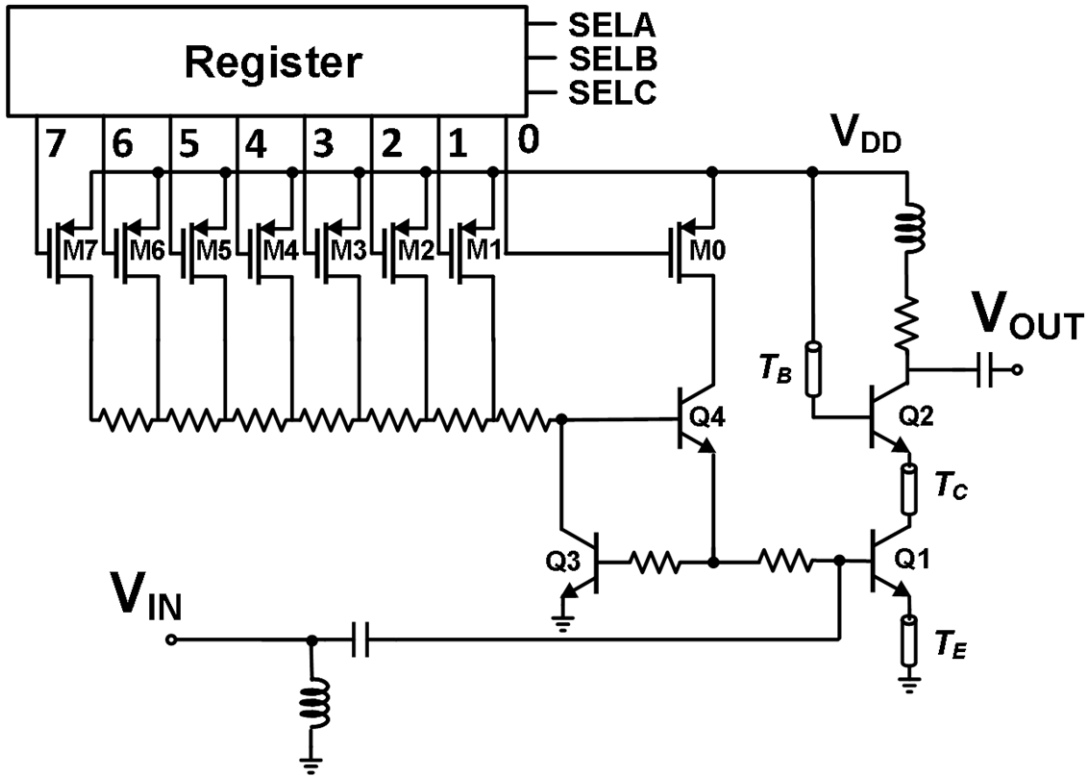


Figure 35. VGA schematic

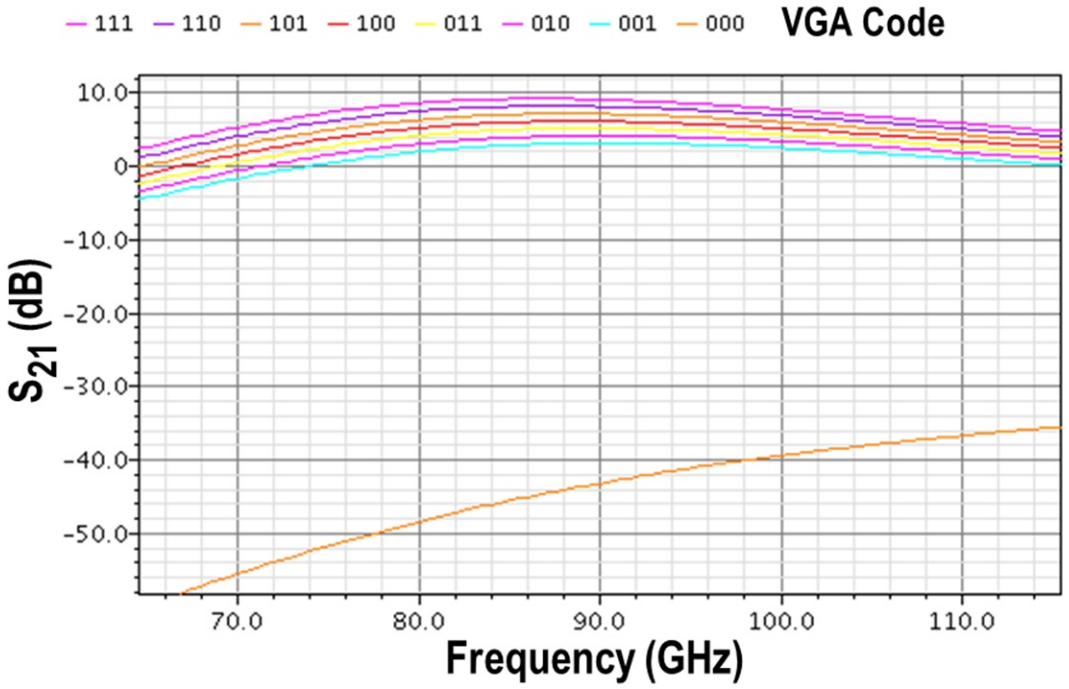


Figure 36. Simulated VGA S_{21} for all 8 gain settings

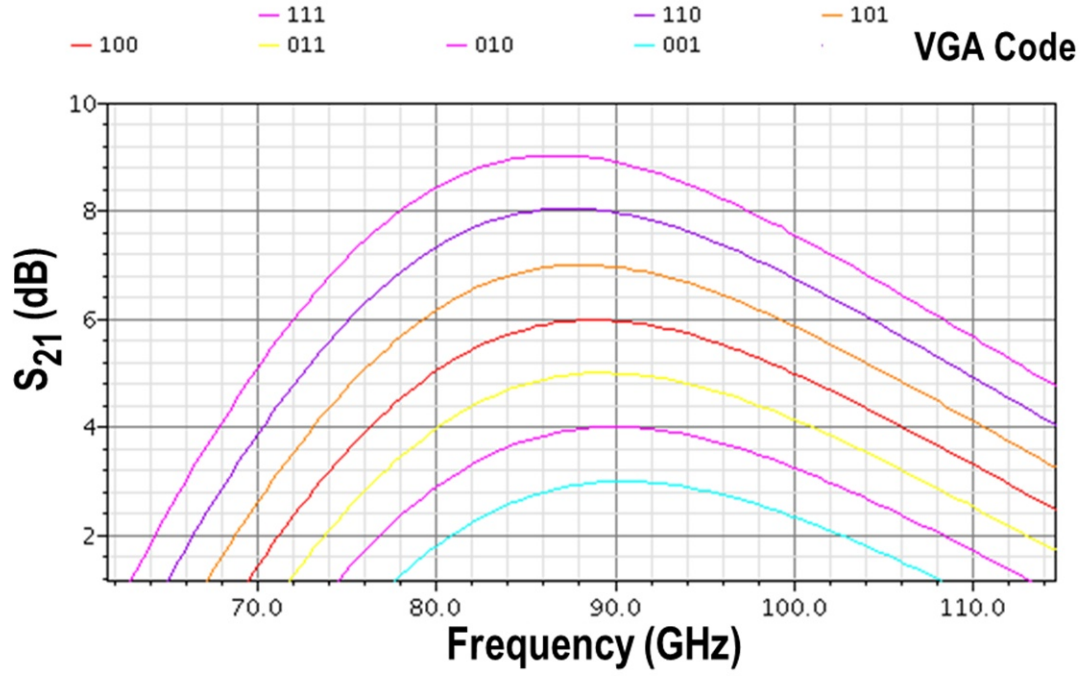


Figure 37. Enlarged plot of the simulated VGA S_{21} for the 7 positive gain settings

A breakout circuit for the VGA is not available, so no direct VGA measurements are possible. However, VGA operation is demonstrated later in the system measurement section. Figure 38 is a photograph of the VGA. The size of the VGA is 115×175μm^2.

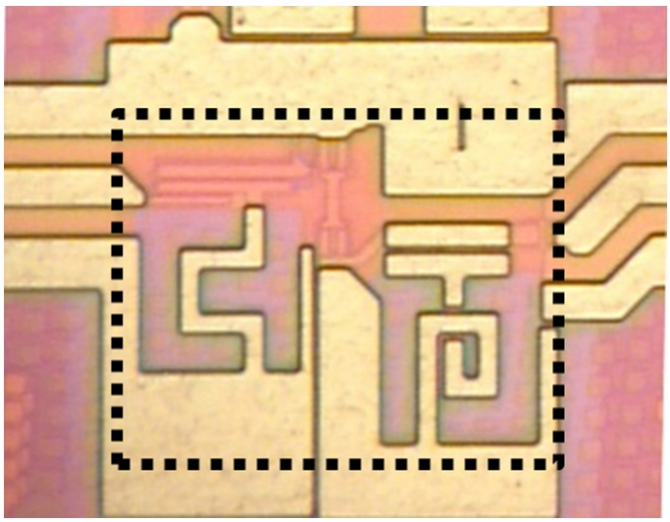


Figure 38. Photograph of the VGA

3.6 Design of the Passive 4-1 Power Combiner

The last circuit in the RF path before the detector is a 4:1 power combiner. This circuit combines the outputs of four elements, enabling super-pixel functionality. A cascade of traditional passive Wilkinson 2:1 combiners are used to create the 4:1 combiners. Passive Wilkinson combiners can pose layout challenges due to their relatively large size, since they require signal traces ¼λ in length. In this design that requirement becomes an advantage. Due to the antenna spacing at approximately .59λ, there is ample room for the combiners. They are designed as ground-shielded CPW with M3 as the lowest level metal (Figure 39) to provide for two levels of metal interconnect underneath the RF signal paths. This allows power, clock and

data signals to be routed under the RF signal paths without any coupling concerns. The combiners' inputs and output are all matched to 50Ω. The simulated insertion loss is less than 0.6dB. A breakout circuit for the 4:1 power combiner is not available, so no direct 4:1 power combiner measurements are possible, although 4:1 power combiner operation is implicitly demonstrated later in the system measurement section. Figure 40 is a schematic of a 2:1 power combiner, while Figure 41 is a schematic of the 4:1 combiner. Figure 42 is a photograph of the 4:1 power combiner. The size of the 4:1 power combiner is 480×720μm^2.

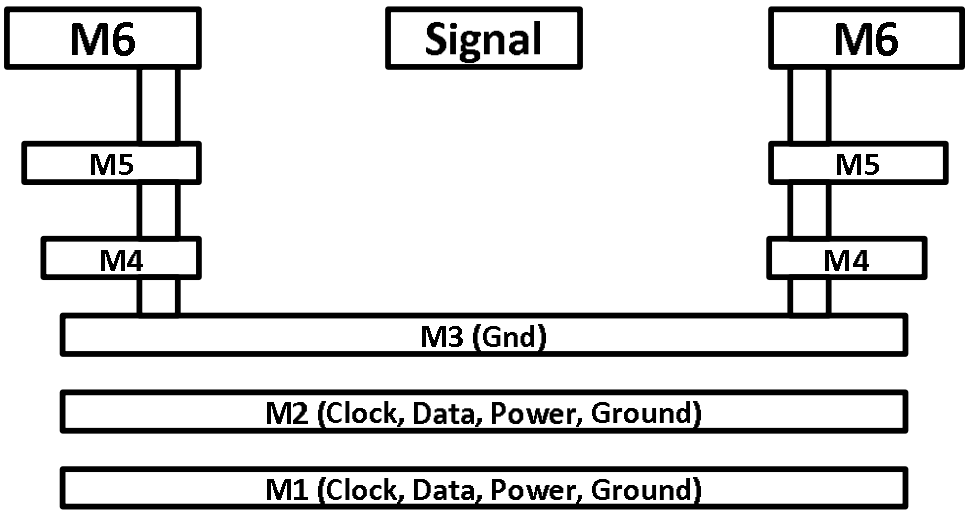


Figure 39. Ground-shielded CPW

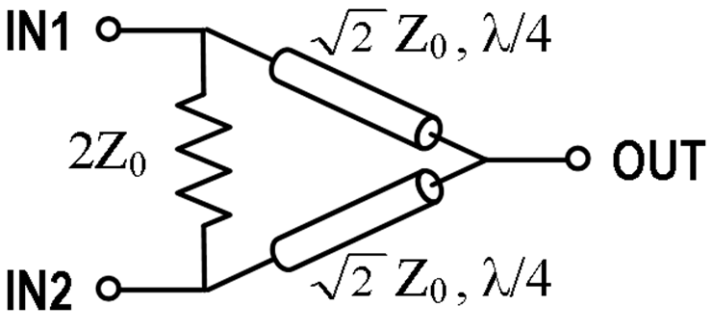


Figure 40. 2:1 Wilkinson power combiner schematic

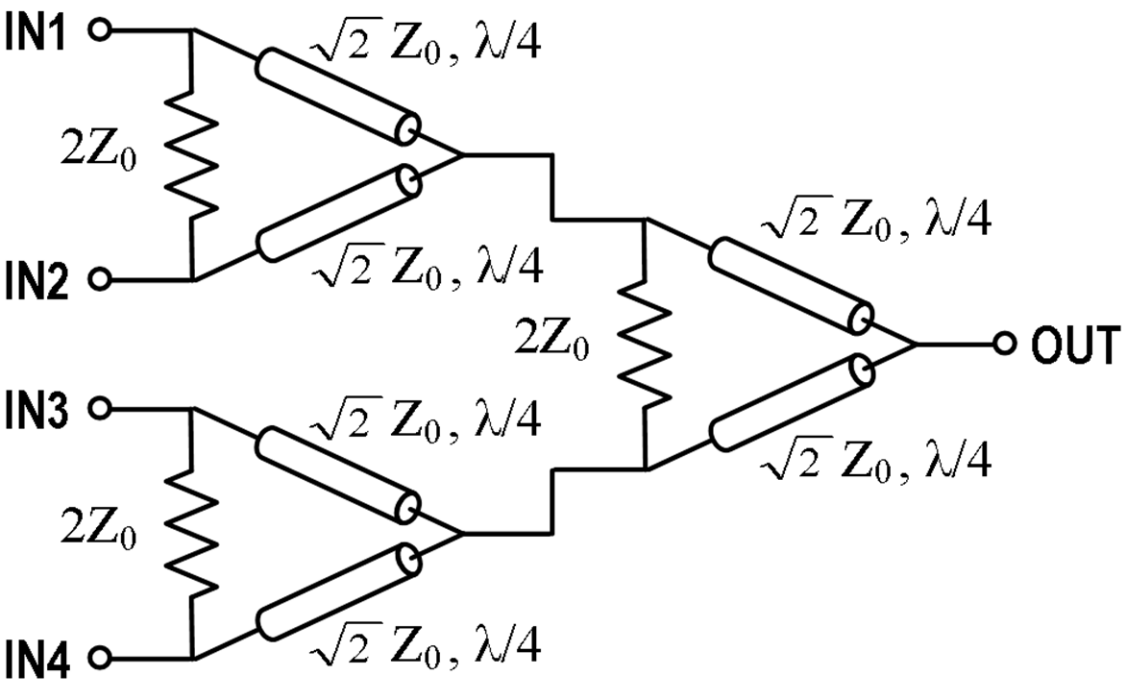


Figure 41. 4:1 Wilkinson power combiner schematic

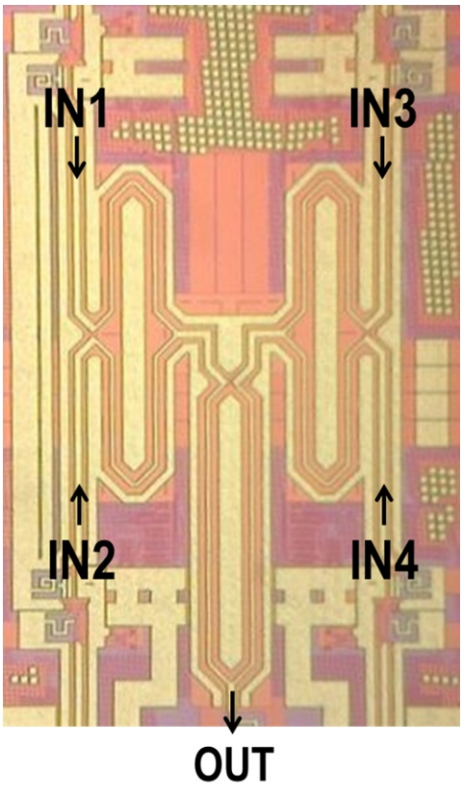


Figure 42. Photograph of the Wilkinson 4:1 power combiner

3.7 Design of a W-band Power Detector

The power detector converts the millimeter-wave power incident upon the receiver into a DC voltage. It should exhibit a linear responsivity, requiring the detector to operate in the square-law region. The detector in this work (Figure 43) consists of a single-ended input, differential output topology. Similar to the designs in [5] and [30], the left half of the circuit, comprised of Q_{1A}, R_{1A}, Q_{2A}, R_{2A} and C_A, performs the power to voltage conversion, while the right half of the circuit, comprised of Q_{1B}, R_{1B}, Q_{2B}, R_{2B} and C_B, provides a DC reference for the differential output. However, in this design, power detection is realized in a two-stage process. The HBT Q_{1A} together with resistor R_{1A} and the overall capacitance seen at the emitter of Q_{1A} perform the first stage of detection, generating a DC output proportional to the input power level along with AC signals at the fundamental and higher-order harmonics of the input. The HBT Q_{2A} together with R_{2A} and C_2 amplifies the DC component of the first stage output, provides the second stage of power detection of the first stage AC output, and low-pass filters the final output.

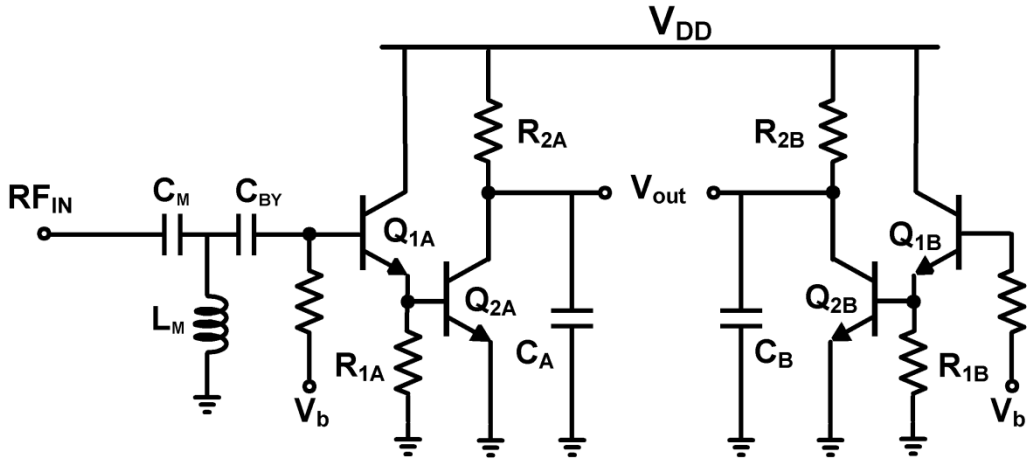


Figure 43. W-band power detector schematic

Assuming a sinusoidal input signal, a Taylor series expansion with truncation of higher order terms is used to determine the differential DC output voltage. In the analysis that follows, the

input voltage at the base of Q_1 is $v_{IN}(t) = V_{bias} + V_{in,m}\cos(\omega t)$, $Vbias = V_{BE} + V_{O1}$, V_{O1} is the voltage across R_1, $v_{O1}(t)$ is the first stage output, $v_{O2}(t)$ is the second stage output, V_T is the thermal voltage, I_{DC1} and I_{DC2} are the DC bias currents of Q_{1A} and Q_{2A}, respectively, and $V_T = kT/q$ where T is temperature, k is the Boltzmann constant, and q is the electron charge.

$$v_{O1}(t) = I_S R_1 e^{(v_{IN}(t) - V_{O1})/V_T}$$

$$v_{O1}(t) = I_S R_1 e^{(V_{BE} + V_{O1} + V_{in,m}\cos(\omega t) - V_{O1})/V_T}$$

$$v_{O1}(t) = I_S R_1 e^{[V_{BE} + V_{in,m}\cos(\omega t)]/V_T}$$

$$v_{O1}(t) = I_{DC1} R_1 e^{[V_{in,m}\cos(\omega t)]/V_T}$$

$$v_{O1}(t) = I_{DC1} R_1 \left[1 + \frac{V_{in,m}\cos(\omega t)}{V_T} + \frac{{V_{in,m}}^2 \cos^2(\omega t)}{2{V_T}^2}\right]$$

$$v_{O1}(t) = I_{DC1} R_1 \left[1 + \frac{V_{in,m}\cos(\omega t)}{V_T} + \left(\frac{{V_{in,m}}^2}{2{V_T}^2}\right) \bullet \left\{\frac{1}{2} + \frac{1}{2}\cos(2\omega t)\right\}\right]$$

$$v_{O1}(t) = I_{DC1} R_1 \left[\left(1 + \frac{{V_{in,m}}^2}{4{V_T}^2}\right) + \left(\frac{V_{in,m}}{V_T}\right)\cos(\omega t)\right]$$

$$v_{O2}(t) = V_{DD} - I_2(t) R_2$$

$$v_{O2}(t) = V_{DD} - R_2 I_{S2} e^{[v01(t)]/vt}$$

$$v_{O2}(t) = V_{DD} - R_2 I_{S2} e^{\{I_{DC1} R_1 [(1 + \frac{{V_{in,m}}^2}{4{V_T}^2}) + (\frac{V_{in,m}}{V_T})\cos(\omega t)]\}/vt}$$

$$v_{O2}(t) = V_{DD} - R_2 I_{S2} e^{\{I_{DC1} R_1 (\frac{1}{V_T} + \frac{{V_{in,m}}^2}{4{V_T}^3})\}} \; e^{\{I_{DC1} R_1 (\frac{V_{in,m}}{{V_T}^2})\cos(\omega t)\}}$$

$$v_{O2}(t) = V_{DD} - R_2 I_{DC2} e^{\{I_{DC1} R_1 (\frac{{V_{in,m}}^2}{4{V_T}^3})\}} \; e^{\{I_{DC1} R_1 (\frac{V_{in,m}}{{V_T}^2})\cos(\omega t)\}}$$

$$v_{O2}(t) = V_{DD} - R_2 I_{DC2} e^{I_{DC1}R_1\left(\frac{V_{in,m}{}^2}{4V_T{}^3}\right)} \left(1 + \{I_{DC1}R_1\left(\frac{V_{in,m}}{V_T{}^2}\right)\cos(\omega t)\} + \frac{\{I_{DC1}R_1\left(\frac{V_{in,m}}{V_T{}^2}\right)\cos(\omega t)\}^2}{2} \right)$$

$$v_{O2}(t) = V_{DD} - R_2 I_{DC2} \left(1 + I_{DC1}R_1\left(\frac{V_{in,m}{}^2}{4V_T{}^3}\right) \right) \left(1 + \{I_{DC1}R_1\left(\frac{V_{in,m}}{V_T{}^2}\right)\cos(\omega t)\} + \frac{\{I_{DC1}R_1\left(\frac{V_{in,m}}{V_T{}^2}\right)\cos(\omega t)\}^2}{2} \right)$$

$$v_{O2}(t) = V_{DD} - R_2 I_{DC2} \left(1 + I_{DC1}R_1\left(\frac{V_{in,m}{}^2}{4V_T{}^3}\right) \right) \left(1 + \frac{\{I_{DC1}R_1\left(\frac{V_{in,m}}{V_T{}^2}\right)\}^2 \left(\frac{1}{2} + \frac{1}{2}\cos(2wt)\right)}{2} \right)$$

$$v_{O2}(t) = V_{DD} - R_2 I_{DC2} \left(1 + I_{DC1}R_1\left(\frac{V_{in,m}{}^2}{4V_T{}^3}\right) \right) \left(1 + \frac{\{I_{DC1}R_1\left(\frac{V_{in,m}}{V_T{}^2}\right)\}^2}{4} \right)$$

$$v_{O2}(t) = V_{DD} - R_2 I_{DC2} \left(1 + \frac{I_{DC1}R_1}{V_T}\left(\frac{V_{in,m}{}^2}{4V_T{}^2}\right) \right) \left(1 + \frac{\{\frac{I_{DC1}R}{V_{T\ 1}}\left(\frac{V_{in,m}}{V_T}\right)\}^2}{4} \right)$$

$$v_{O2}(t) = V_{DD} - R_2 I_{DC2} \left(1 + \frac{I_{DC1}R_1}{V_T}\left(\frac{V_{in,m}{}^2}{4V_T{}^2}\right) \right) \left(1 + \frac{\left(\frac{I_{DC1}R_1}{V_T}\right)^2 \left(\frac{V_{in,m}}{V_T}\right)^2}{4} \right)$$

$$v_{O2}(t) = V_{DD} - R_2 I_{DC2} \left(1 + \left(\frac{I_{DC1}R_1}{V_T}\right)\left(\frac{V_{in,m}{}^2}{4V_T{}^2}\right) \right) \left(1 + \left(\frac{I_{DC1}R_1}{V_T}\right)^2 \left(\frac{V_{in,m}{}^2}{4V_T{}^2}\right) \right)$$

$$v_{O2}(t) = V_{DD} - R_2 I_{DC2} \left(1 + \left(\frac{I_{DC1}R_1}{V_T}\right)\left(\frac{V_{in,m}{}^2}{4V_T{}^2}\right) + \left(\frac{I_{DC1}R_1}{V_T}\right)^2 \left(\frac{V_{in,m}{}^2}{4V_T{}^2}\right) \right)$$

With $V_{in,m}$ replaced by $V_{inT,m}$, $V_{diff,T}$ of the proposed two-stage detector is given by

$$V_{diff,T}(t) = R_2 I_{DC2}\left(\left(\frac{I_{DC1}R_1}{V_T}\right)\left(\frac{{V_{inT,m}}^2}{4{V_T}^2}\right)+\left(\frac{I_{DC1}R_1}{V_T}\right)^2\left(\frac{{V_{inT,m}}^2}{4{V_T}^2}\right)\right)$$

$$V_{diff,T}(t) = R_2 I_{DC2}\frac{V_{BE,on}}{V_T}\left(\left(\frac{{V_{inT,m}}^2}{4{V_T}^2}\right)+\left(\frac{V_{BE,on}}{V_T}\right)\left(\frac{{V_{inT,m}}^2}{4{V_T}^2}\right)\right)$$

$$V_{diff,T}(t) = R_2 I_{DC2}\frac{V_{BE,on}}{V_T}\left(\frac{{V_{inT,m}}^2}{4{V_T}^2}\right)\left(1+\left(\frac{V_{BE,on}}{V_T}\right)\right) \qquad (16)$$

where $V_{BE,on}$ is the turn-on voltage of an HBT, V_T is the thermal voltage, I_{DC1} and I_{DC2} are the DC bias currents of Q_{1A} and Q_{2A}, respectively, and $V_{RF,m}$ denotes the RF input amplitude to the power detector.

With lossless conjugate input matching, the input RF power to the power detectors is $P_{RF} = P_{in} \Rightarrow \frac{{V_{RF,rms}}^2}{R_s} = \frac{{V_{in,rms}}^2}{R_{in}} \Rightarrow \frac{{V_{RF}}^2}{2R_s} = \frac{{V_{inT,m}}^2}{2R_{inT}} = \frac{{V_{inS,m}}^2}{2R_{inS}}$. For the two stage detector $R_S < R_{inT}$, so $R_{inT} = R_s\left(1+Q_T^2\right)$ due to the impedance transformation network. To find P_{RF} in terms of $V_{inT,m}$, $\frac{{V_{RF}}^2}{2R_s} = \frac{{V_{inT,m}}^2}{2R_s\left(1+Q_T^2\right)}$.

Dividing (16) by the power input to the detector, the responsivity $\Re_T$ of the two-stage detector is expressed as:

$$\Re_T = \frac{V_{diff,T}}{P_{in}} = 2R_s\left(1+Q_T^2\right)R_2 I_{DC2}\left(\frac{V_{BE,on}}{V_T}\right)\left(\frac{1}{4{V_T}^2}\right)\left(1+\left(\frac{V_{BE,on}}{V_T}\right)\right)$$

$$\Re_T = \frac{V_{diff,T}}{P_{in}} = \frac{2R_s\left(1+Q_T^2\right)R_2 I_{DC2}}{4{V_T}^2}\left(\frac{V_{BE,on}}{V_T}\right)\left(1+\left(\frac{V_{BE,on}}{V_T}\right)\right)$$

$$\Re_T = \frac{V_{diff,T}}{P_{in}} = \frac{\alpha_T \, 2R_s R_2 I_{DC2}}{4V_T{}^2}\left(\frac{V_{BE,on}}{V_T}\right)\left(1+\left(\frac{V_{BE,on}}{V_T}\right)\right) \tag{17}$$

Upon examination of (17), the responsivity is clearly seen to be directly proportional to the matching gain α_T.

Similarly, for the single stage detector in [5], $V_{diff,S}$ of the single-stage detector is given by

$$V_{diff,S}(t) = R_2 I_{DC}\left(\frac{V_{inS,m}{}^2}{4V_T{}^2}\right) \tag{18}$$

The input impedance of the one-stage detector is much lower than the input impedance of the two-stage detector, with the real part < 50Ω. With $R_{inS} < R_s$, $R_{inS} = R_s/\left(1+Q_S^2\right)$ due to the impedance transformation network. To find P_{RF} in terms of $V_{inS,m}$, $\frac{V_{RF}{}^2}{2R_s} = \frac{V_{inS,m}{}^2}{2R_s/\left(1+Q_S^2\right)}$. The responsivity, $\Re_S$, for the single-stage detector in [5] is given by

$$\Re_S = \frac{V_{diff,S}}{P_{in}} = \alpha_S \, 2R_s R I_{DC} R_s\left(\frac{1}{4V_T{}^2}\right) \tag{19}$$

where α_S accounts for the impedance transformation ratio from the source impedance to the detector's input impedance, and I_{DC} is the detector DC bias current.

To demonstrate the responsivity improvement of the proposed power detector compared to the conventional counterpart [5], the ratio of the responsivities is derived. It is readily proven the responsivity of this detector is always higher than that of the single stage detector, and the lower limit is given by

$$\frac{\Re_T}{\Re_S} \geq \frac{\alpha_T}{\alpha_S} \cdot \frac{V_{BE,on}}{V_T}\left(1+\frac{V_{BE,on}}{V_T}\right) \tag{20}$$

This ratio shows improvement in the proposed detector's responsivity, because (1) $V_{BE,on}/V_T$ is greater than unity, and (2) α_T is greater than α_S due to the higher input impedance of the two-stage detector.

The measured results of the proposed power detector are shown in Figure 44. The detector output voltage increases linearly with the input signal power, until the detector output begins saturating around 800mV, corresponding to a detector input power of –18dBm. The measured responsivity is 54KV/W. A thermal control mechanism was used to maintain the required level of responsivity to achieve an NETD of less than 0.5K. A break out circuit of the proposed detector is shown Figure 45. It occupies 125×160μm^2 of chip area and consumes 4mW of DC power.

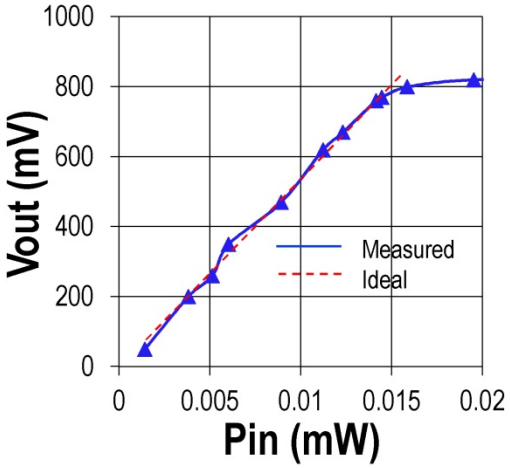


Figure 44. Measured W-band power detector performance

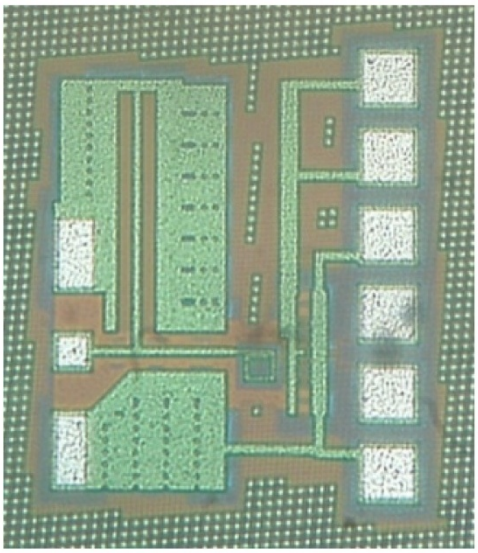


Figure 45. W-band power detector breakout circuit

3.8 Design of the Digital Control Circuits

The wideband true time delay circuit and the VGA require external control signals to set the proper delay and gain coefficients for each RF path. These coefficients are stored locally in each circuit in digital control registers. This chip contains a clock, reset, 3 data and 6 address pins. These signals are routed around the chip to all control registers, where they are locally decoded. Although the control signals cross under the RF signal path in metal 1 and metal 2 and are shielded by metal 3, no digital signals are toggled during chip operation to minimize coupling. The control registers and decoders are composed of standard CMOS logic including inverters, NANDs, NORs and DFFs.

3.9 The Complete Chip

A die photograph of the complete 9-element imaging array receiver is shown in Figure 46. Clearly marked are all of the main circuits, as well as a super-pixel.

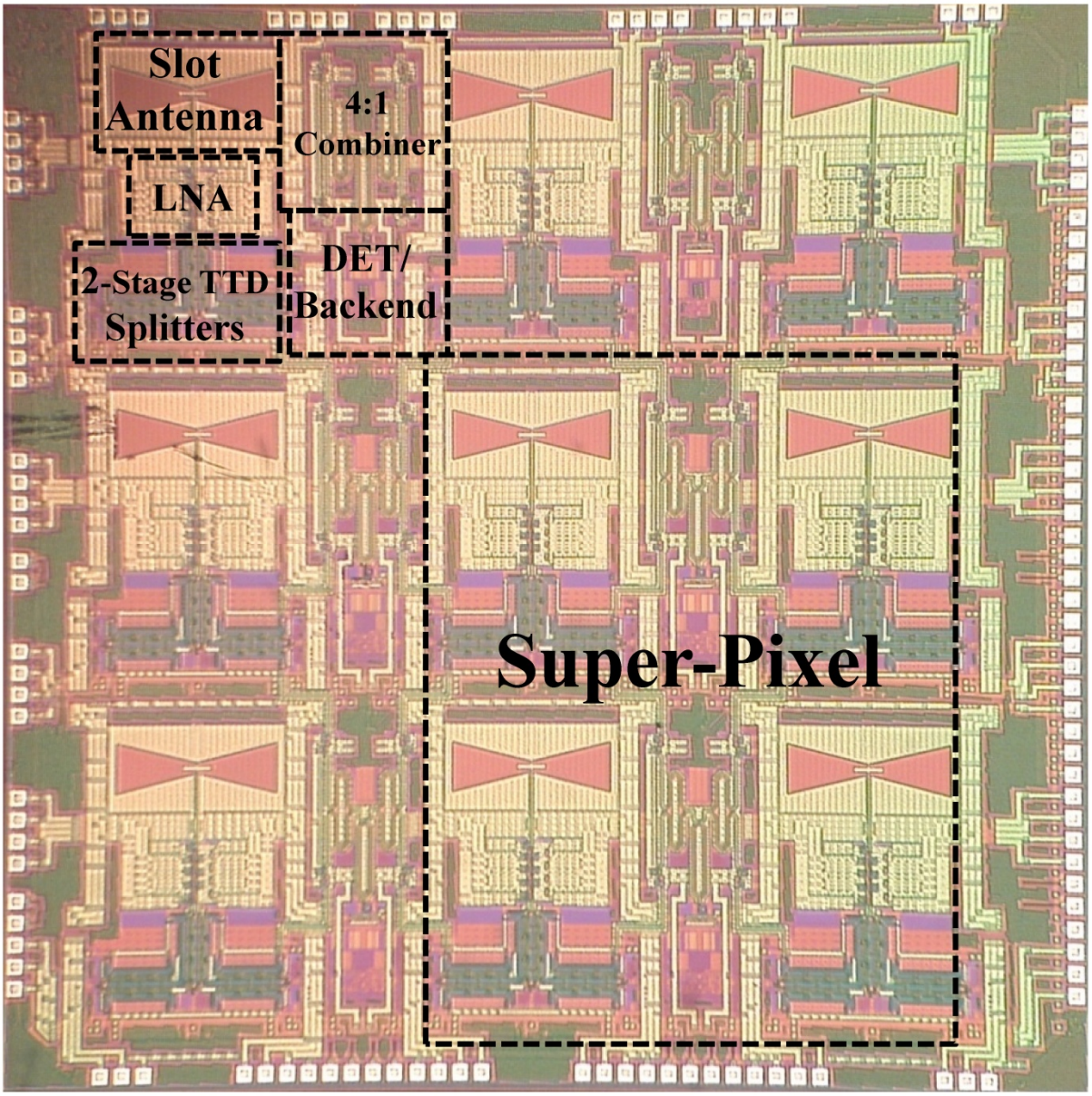


Figure 46. Die photo of the 9-element imaging array receiver

Chapter 4

System Level Test Setup and Measurement Results

Introduction

This section describes the test system, the various system setups and the system measurements.

4.1 Test System Overview

The system level test system consists of a test fixture, a laptop, a thermal control mechanism and the test stimulus. The test fixture comprises two PCBs. The first PCB is a custom 2-layer board (Figure 47). It contains an open-top socket that attaches the packaged receiver chip to the PCB, a parallel interface that allows communication between the receiver chip and the second PCB, manually adjustable voltage regulators that control the various supply voltages and biases used by the receiver chip and the interface circuitry, and the thermal control mechanism used to remove any excess heat generated by the receiver chip. The receiver chip is wire-bonded in a 160-pin thin quad flat pack (TQFP) in which the top of the package has been etched away to expose the die (Figure 48). It is attached to the PCB via the open-top socket. This allows for the testing of multiple receiver chips with a single test fixture, as receiver chips can easily be swapped in and out of the socket. The second PCB is an off-the-shelf DAQ/GPIO board, which communicates with the laptop through a USB connection. To facilitate antenna directivity testing, the test fixture is mounted to a rod that can be rotated ±180° with respect to an incident W-band signal 50cm away. To enable testing with different signal polarizations, the test fixture can be rotated 90° with respect to the rod. Custom software provides a simple user interface to reset the receiver chip, to set and check all receiver chip registers, to initiate data acquisition, and to record all data to the laptop. The custom software is comprised of two parts: the software

written in C++ to control the DAQ/GPIO (Appendix 1) and the graphical user interface (GUI) written in Visual C++ for the laptop. The DAQ/GPIO contains the routines to program and reset the receiver chip's registers, set conversion times, initiate conversions and read output data. The laptop GUI interfaces with the DAQ/GPIO to allow the user to easily and efficiently send data to the DAQ/GPIO, provides continuous status of the receiver chip, and writes all conversion data to files on the laptop. A screen shot of the laptop GUI is shown in Figure 49.

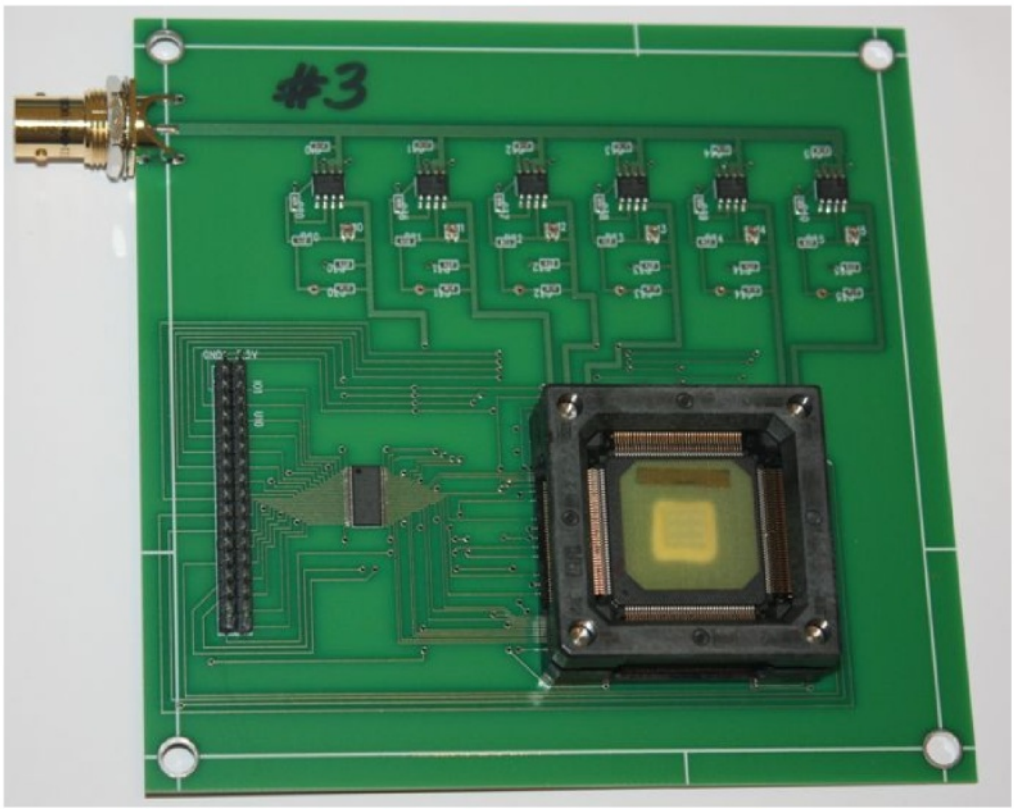


Figure 47. Custom PCB and a packaged chip

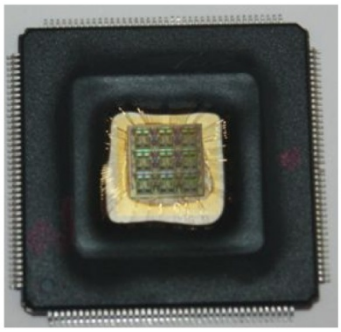


Figure 48. Receiver chip in the open-top TQFP package

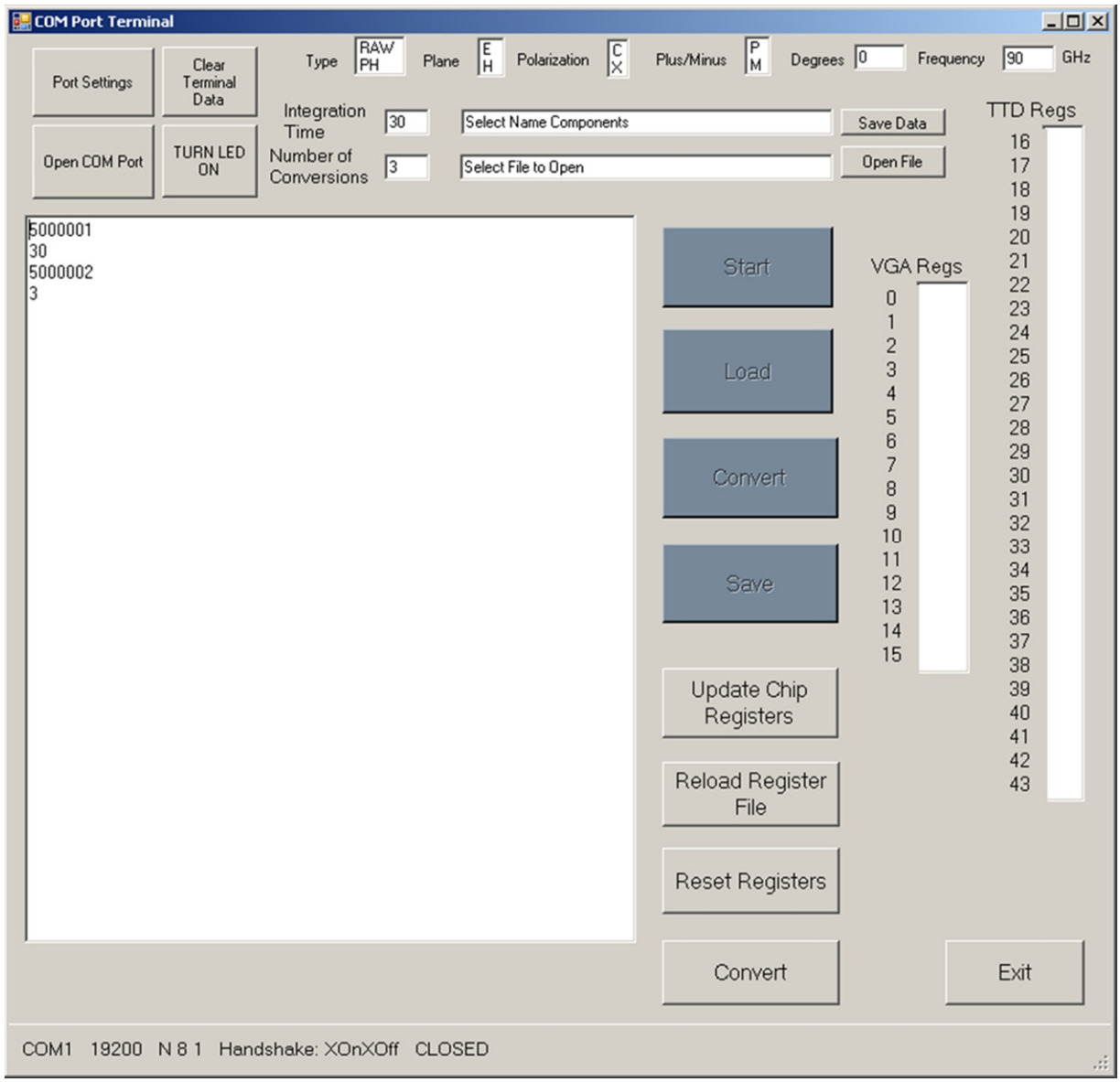


Figure 49. Screen shot of the laptop GUI

4.2 Test Setups

There are three main setups used in the testing of this device. The first setup (Figure 50) contains a W-band signal source, and it is used for antenna testing and active responsivity measurements. The second setup (Figure 51) is used for passive responsivity measurements. It does not contain a W-band signal source; instead EM absorbers at various temperatures are the radiation source. This setup is also used to measure the output noise of the receiver. The third test setup (Figure 52) is used to perform active imaging tests.

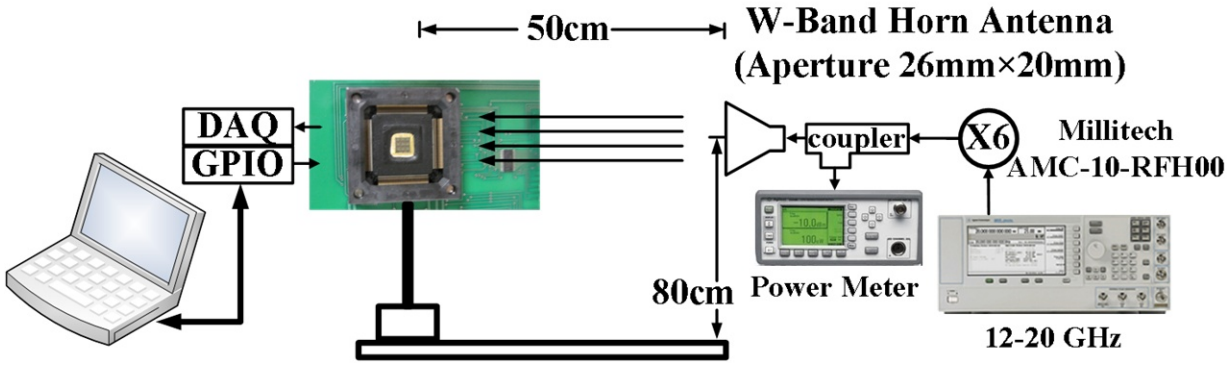


Figure 50. Antenna and active responsivity test setup

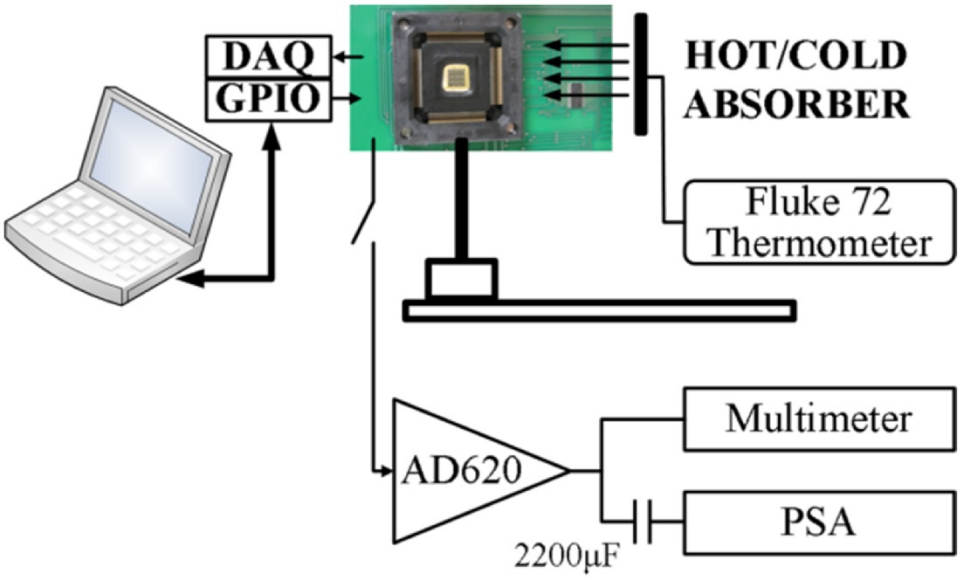


Figure 51. Passive responsivity and output noise test setup

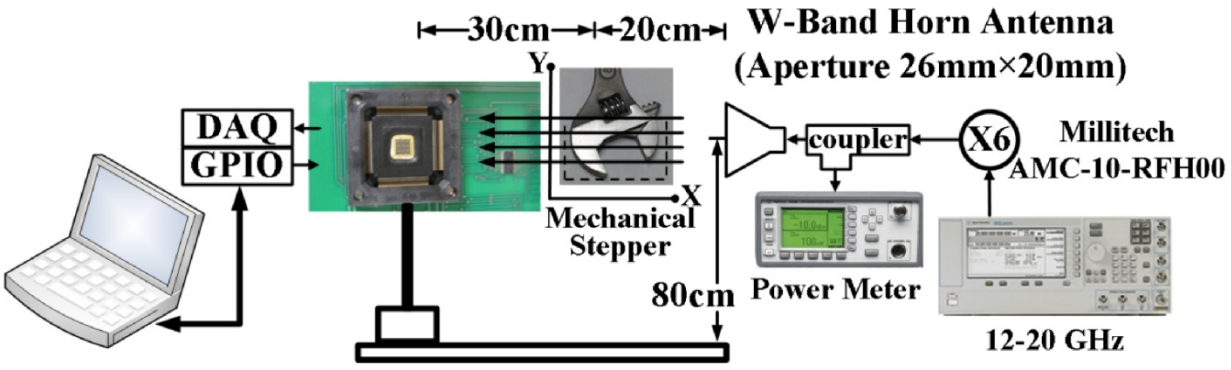


Figure 52. Active imaging test setup

4.3 System Level Test Preparation

The W-band transmitter output power versus frequency needs calibration before commencing any system testing. The transmit power and frequency are varied on the signal generator, and the output power delivered to the transmit horn is recorded for each transmit power and frequency setting. Next, a horn antenna connected to a power meter is placed 50cm away from the transmit horn antenna. The transmit power and frequency are again varied and the received power is recorded for each power and frequency setting. The propagation loss *PL* in air is calculated to be 65.5dB for a distance of 50cm at 90GHz.

Next, prior to beginning system tests, it is important to first verify the functionality of the packaged chips, since the chips are all blind-packaged without any testing. As a first test for functionality, the supply currents for the test fixture and the DC voltage of each super-pixel's detector are measured and compared to expected values. Upon successful measurement results, an abbreviated active responsivity test is run to prescreen the chips for working super-pixels. Using the test setup of Figure 50 the TTD coefficients are set to 0˚ delay, the VGA coefficients are set to no gain and the W-band transmitter power is varied. The receiver output should remain

unchanged for all super-pixels. The test is repeated with the gain coefficients set to maximum gain. This time the super-pixel outputs should vary with the transmit power level. As a final test, the W-band transmit power is fixed and the VGA gain coefficients are varied from no gain to maximum gain. This time the super-pixel outputs should vary with the VGA gain coefficients. If the chip passes all three tests, it is considered functional and fit for further testing. Not all super-pixels work on all chips tested, although all of the working super-pixels have similar characteristics. The receiver chips with four working super-pixels are used for complete system-level testing.

4.4 System Level Antenna Testing

RF path calibration is required for all RF paths prior to receiver measurements due to variation among the RF paths. This variation mainly comprises (1) the gain variation among antennas, (2) amplifier gain variation due to process variation, and (3) variation in path loss through the active power splitter, true time delay and power combiner. Generally, the calibration for each RF path is conducted based on the output of its power detector for the ON and OFF states of the transmitter, with the transmit frequency set to 90GHz, the integration time set to 20ms and the transmit power set to a level to prevent saturation of the integrator. The measurement starts with turning on a single RF path. The gain of the VGA of this path is swept across all settings when the chip is mounted to receive in the broadside direction (Figure 50). This measurement is repeated for the RF path associated with an adjacent element in the same reference plane. VGA settings for the two paths are selected so as to provide the same gain in each RF path.

Antenna testing begins with the two adjacent paths of the two adjacent elements in the same reference plane turned on together. The radiation patterns are measured under a full sweep of

TTD settings combined with the tentative VGA settings. With the transmit horn position fixed, the receiver test fixture is rotated mechanically in 1° steps from –60° to +60° to measure the radiation pattern. At each step, the test program sweeps through all combinations of VGA and TTD settings to eventually generate hundreds of radiation patterns. Based on the measured patterns, the optimal VGA and TTD settings are picked in terms of (1) the peak gain direction, and (2) sustaining similar power levels for different steering angles for each targeted steering angle from –18° to +18° with a step of 4.5°. The setting for each super-pixel is obtained by combining the optimal setting for each 2 element pair in the same reference plane. This data is used to program the gain and delay coefficients for the super-pixels in the antenna beam steering measurements, which demonstrate the wideband delay and amplitude control of the super-pixel. The super-pixel is programmed for beam angles from -18° to 18° in 4.5° steps. For each setting the transmit power and frequency remain constant, the receiver-to-transmitter angle is varied from -60° to +60° in 1° increments and the received power is measured. This test is repeated with a single RF path enabled. Figure 53 shows super-pixel beam steering radiation patterns compared with a single antenna radiation pattern, demonstrating the beam steering ability of the super-pixel.

Next a single RF path is enabled. With constant transmit power and frequency settings, the receiver-to-transmitter angle is varied from -60° to 60° in 1° increments. The VGA gain coefficients are stepped over the entire 6dB tuning range of 3 to 9dB in 1dB increments at each angle and the received power is measured. Figure 54 shows a single antenna's radiation pattern in one quadrant under these conditions. While the absolute value of the gain is unimportant, the 1dB differences in output power corresponding to the VGA gain coefficient settings verifies the operation of the VGA over the target range of 0° to 18°.

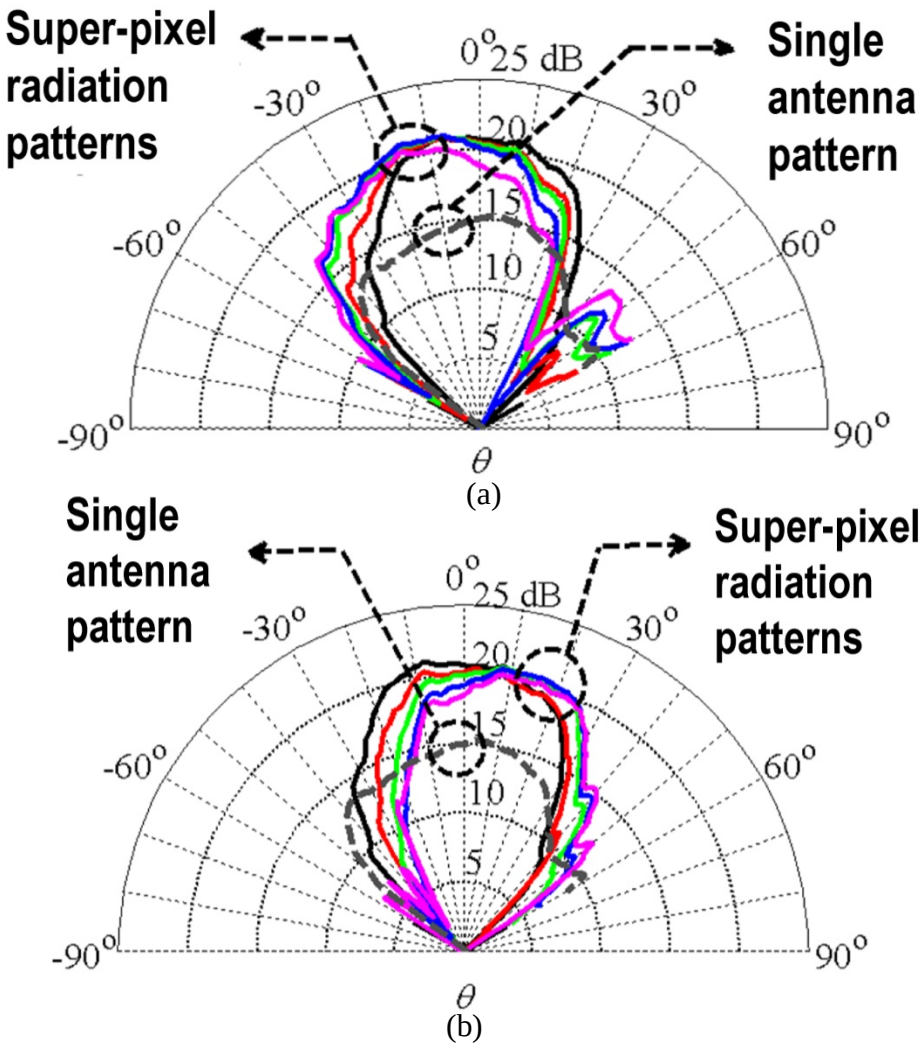


Figure 53. Super-pixel beam steering radiation patterns compared with single antenna radiation pattern. (a) –18° to 0° (b) 0° to 18°

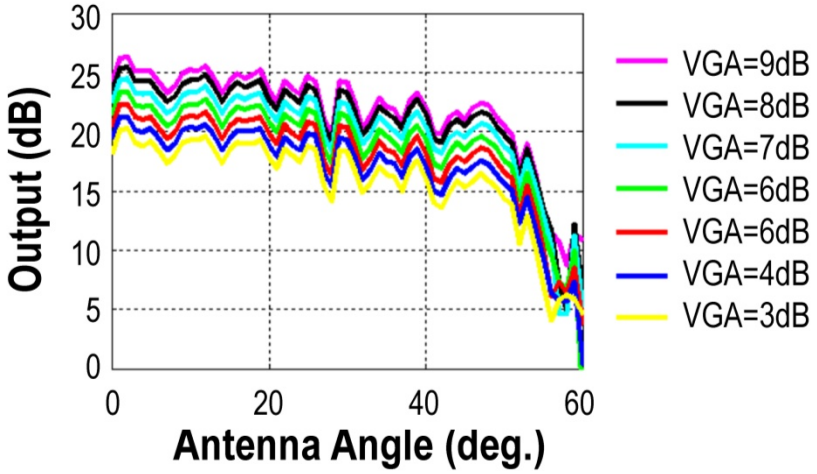


Figure 54. Single antenna beam pattern for various VGA gain settings

4.5 Receiver Noise Testing

The test setup of Figure 51 is used for noise tests, with the AD620 connected to the detector outputs. The receiver is programmed for 0 degree beam steering. The VGAs are all disabled for the detector output noise tests to disconnect the detectors from all signal paths. For the receiver output noise tests, the VGA gain coefficients are set to the maximum gain setting and a room temperature absorber is used as the temperature source. The output noise spectra of both the detector and a complete super-pixel are plotted in Figure 55 along with the simulated detector output noise. The measured detector output noise shows good agreement with the simulation results, exhibiting a minimum output noise at 10 kHz offset of 205nV/Hz$^{1/2}$. The measured receiver output noise at 10 kHz offset is 276nV/Hz$^{1/2}$.

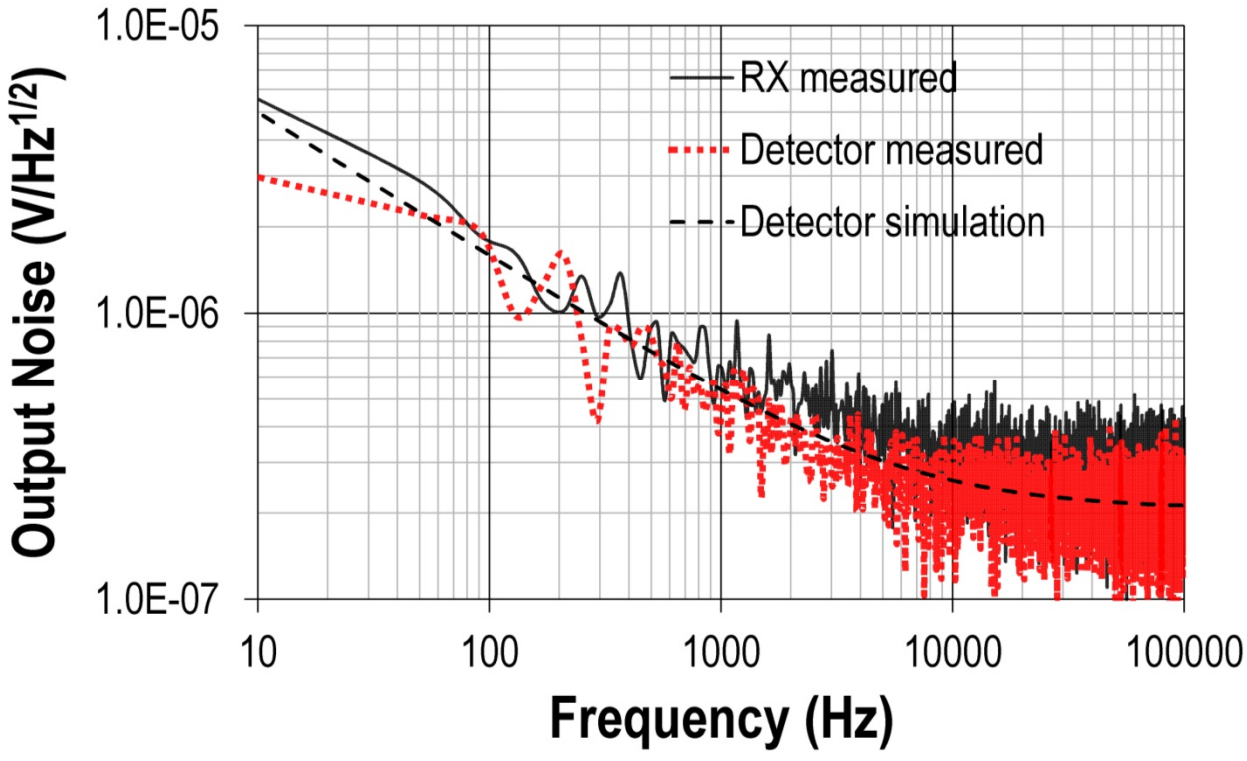


Figure 55. Receiver and detector measured output noise

4.6 Receiver Active Responsivity Testing

The active responsivity tests are performed with the test setup depicted in Figure 50 using a W-band transmitter to illuminate the receiver chip. The receiver-to-transmitter angle is set to 0°, the receiver chip is programmed to 0° beam steering based on data obtained from the earlier antenna testing, and the previously calibrated transmitter power is set manually to a level to prevent saturation of the detector. All nearby metal objects (posts, wires, etc.) are covered by EM absorbers. The transmit frequency is swept over the W-Band frequency range for several values of transmit power. The received power is recorded at all frequency and power combinations for minimum, nominal and maximum VGA gain settings. Figure 56 shows the results of the measured coherent (i.e., active) receiver responsivity, including antenna losses, for the minimum, nominal, and maximum VGA gain settings. In addition to providing SNR improvements in an active imaging mode, the illuminating source allows one to measure the frequency response of the receiver chip. As can be seen in Figure 56, the entire receiver chain has a half-power bandwidth of roughly 87–108GHz.

The measured receiver super-pixel output noise voltage (e.g., 276nV/$Hz^{1/2}$ at 10kHz) and the measured coherent responsivity are used to determine the receiver NEP. This measured coherent NEP is plotted in Figure 56 for the minimum, nominal, and maximum VGA gain settings. The minimum receiver NEP under these conditions, averaged over the receiver bandwidth, is 0.28fW/$Hz^{1/2}$. Using the output noise voltage corresponding to the 20ms sampling time (e.g., 3.5μV/$Hz^{1/2}$), the average NEP is 3.55fW/$Hz^{1/2}$. These results were consistent when measured over multiple super-pixels from multiple chips.

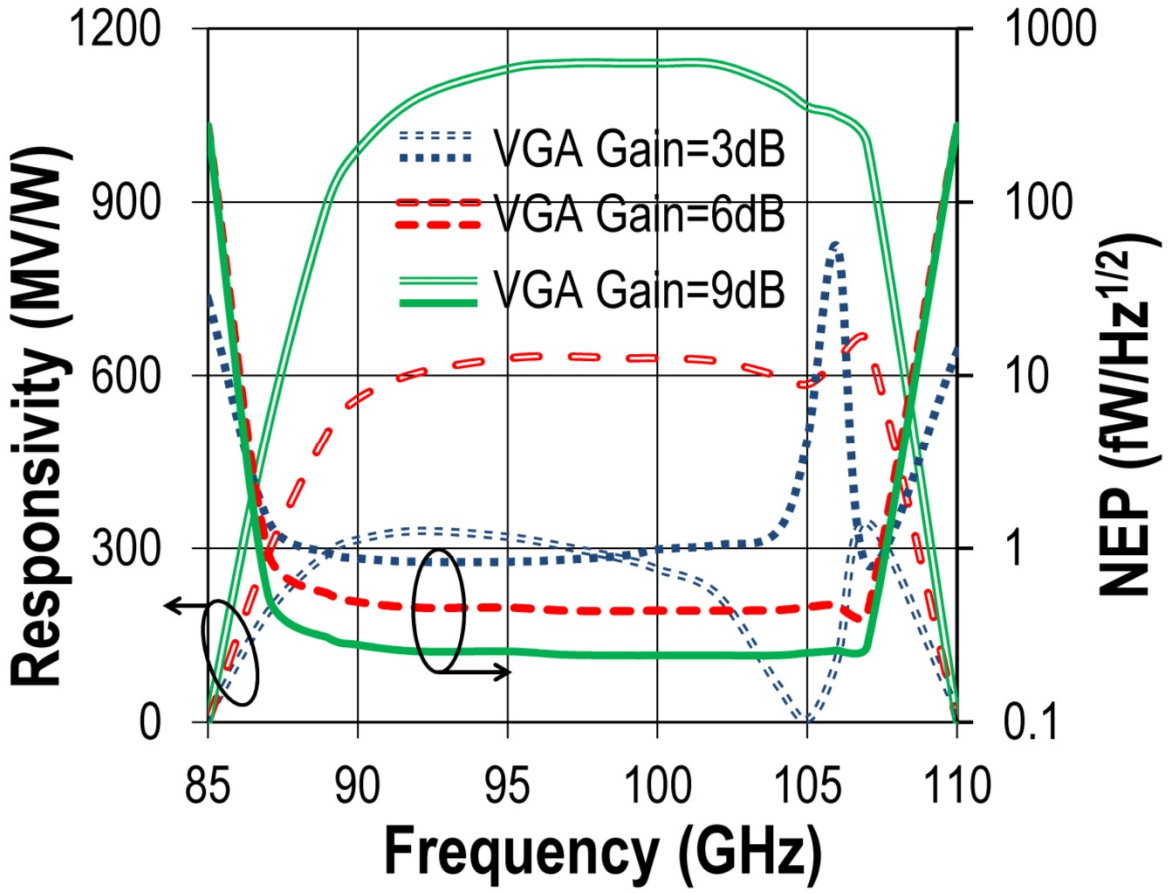


Figure 56. Measured coherent responsivity and measured NEP for minimum, nominal and maximum VGA gain settings of a super-pixel

4.7 Receiver Passive Responsivity Testing

Passive responsivity tests incorporate the test setup of Figure 51 with the AD620 disconnected from the detector outputs. The receiver is programmed for 0 degree beam steering and for maximum VGA gain. A focusing system is not available, so no attempt is made to passively image a small detailed object. Instead, the size of the EM absorber and its spacing to the receiver chip are chosen to completely cover the field of view of the array. EM absorbers are heated to temperatures ranging from 22.3C to 177C using an electric warming plate. An infrared thermometer is used to measure the surface temperature uniformity of the EM absorber. The EM

absorbers are placed approximately 5cm away from the receiver chip. There are no means to generate an extremely cold source, so 22.3C is chosen as the temperature reference. 177C is the temperature limit the EM absorbers can withstand before melting or burning up, so that is the maximum temperature. Multiple measurements are made at each selected temperature and the RMS value of the output voltage fluctuation levels is calculated. From this data, NETD is calculated as described in [7]. Measured results for these passive incoherent measurements are averaged and plotted in Figure 57. With 50K and 90K temperature differences, the receiver shows an incoherent responsivity of approximately 1,000MV/W, which is in good agreement with the measured coherent responsivity. An NETD of 0.45K is directly measured with the passive responsivity test setup. The measured data can also be used to determine the thermal sensitivity of the receiver. It is also plotted in Figure 57.

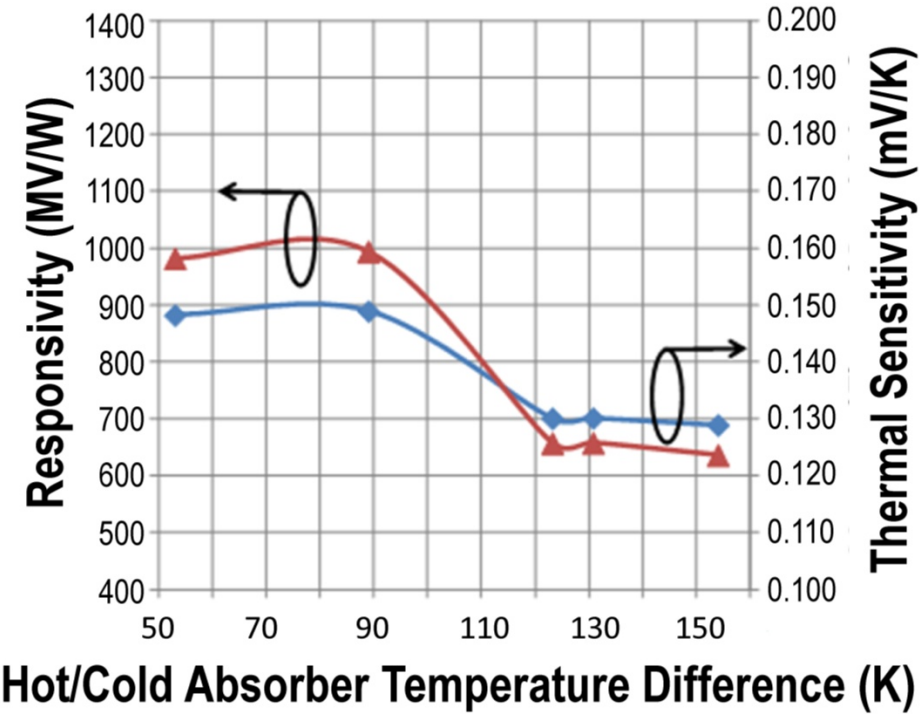


Figure 57. Measured incoherent responsivity and thermal sensitivity of a super-pixel

4.8 Receiver Active Imaging Testing

In the absence of a focusing system, an imaging receiver employing wide beam-width on-chip antennas will have a difficult time forming a passive image of a practical sized object, as the wide antenna beam angle will pick up background noise power from adjacent objects, corrupting the noise power of the desired object to be imaged. To overcome this problem, an active test incorporating an illuminating source with an extremely narrow beam-width is carried out. This source, effectively acting as a lens to "focus" the beam, is used in a setup similar to the ones presented in [5], [6] to test the imaging receiver (Figure 52). In this setup, the illumination frequency is 90 GHz, the output power of the illuminating source is set to a level to prevent saturation of the integrator, the integration time is 20ms, the distance from the illuminating source to the object is 20cm, and the object-to-receiver distance is 30cm. The receiver chip is programmed to 0° beam steering with maximum gain. The object is then stepped in the x and y directions in 4mm increments and the detector outputs are recorded 10 times at each position and then averaged. The total scan time exceeds one hour due to the manual operation of the stepper. The antenna spacing is actually 1.9mm, which requires that the object should ideally be stepped at 3.8mm intervals for equal spaced pixel samples. However, the resolution of the XY-stepper is 1mm resulting in a step size of 4mm. The effect of the 4mm step size is unequal spaced pixel samples of either 1.9mm or 2.1mm. This effect is ignored in the resulting image analysis and equal spaced 2mm samples are assumed.

Figure 58 shows the images from the outputs of the four individual super-pixels, obtained by 4mm spatial sampling as described above. The data for each super-pixel is first smoothed in Matlab using a built-in Matlab 2-D averaging function. Although the super-pixels are all

programmed to 0° beam steering with maximum gain, no attempt is made to equalize the gain differences among the four super-pixels due to variation among circuit components such as antenna gains, LNA gains, TTD losses, signal path losses or detector responsivities. Instead the images are normalized to the same intensity scale to compensate for the super-pixel gain differences. The four individual super-pixel images are then incorporated into one composite image by taking into account the physical location of each pixel in the individual super-pixel images. The combined image is then smoothed in Matlab using a built-in Matlab 2-D averaging function. The subsequent composite image results in a visibly sharper image.

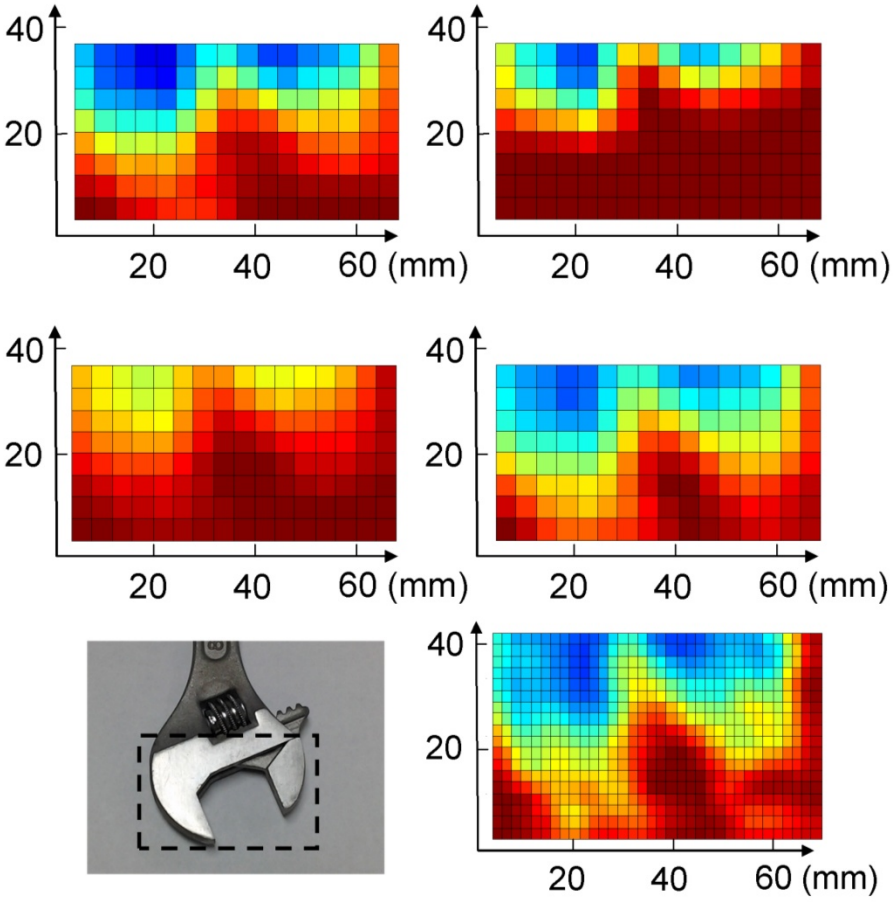


Figure 58. Reconstructed images from the outputs of the four individual super-pixels, a photograph of the object and the composite image obtained by combining the four overlapping super-pixel images

4.9 Receiver Performance Summary

The complete measured performance of the proposed area-efficient beam-steerable sub-terahertz imaging receiver array is summarized in Table I. The proposed 6mm×6mm BiCMOS chip with 9 on-chip antennas forming a 2×2 array of spatial overlapping super-pixels displays a measured NETD of 0.45K and a minimum NEP of 0.28fW/Hz$^{1/2}$. Table II shows comparisons between this work and other W-band imaging receiver chips. The proposed imaging receiver

reports the lowest measured NETD and NEP of silicon-based imaging receivers along with the highest level of integration of any technology.

Table 4. Summary of the Receiver Array Performance

Pre-detection Gain	43dB
Receiver Array 3-dB Bandwidth	87-108GHz
Minimum/Maximum Subarray Beam-Steering Angle	-18°/+18°
Beam-Steering Angular Resolution	4.5°
Measured Average Incoherent Responsivity (Passive Hot/Cold Measurement)	800MV/W
Measured NETD (Averaged Across Multiple Hot/Cold Measurements)	0.45K
Measured NEP	$0.28fW/Hz^{1/2}$
Power Dissipation (W)	2.03W (225mW for each of the 9 elements)
Process Technology	0.18µm SiGe BiCMOS
Level of Integration	9 On-Chip Antennas forming 4 Overlapping 2x2 Super-pixels with Baseband Circuitry
Die Area	6mm×6mm

Table 5. Performance Comparison of W-Band Imagers

Reference	Technology	Level of Integration	Responsivity	Integration Time	Min NEP	NETD
[2]	0.13µm SiGe BiCMOS	Dicke Switch + LNA + Detector	5MV/W	30ms	21fW/Hz$^{1/2}$	0.83K (Calculated)
[3]	65nm CMOS	Dicke Switch + LNA + Detector	0.09MV/W	30ms	200fW/Hz$^{1/2}$	10K (Calculated)
[5]	0.18µm SiGe BiCMOS	Dicke Switch + LNA + Detector + Baseband	45MV/W	30ms	10fW/Hz$^{1/2}$	0.4K (Calculated)
[6]	0.18µm SiGe BiCMOS	4 Element Focal Plane Array with On-chip Antennas and PLL	48MV/W (285MV/W not including on-chip antenna loss)	30ms	8.1fW/Hz$^{1/2}$	3K (Calculated)
[7]	InP HEMT	LNA + Detector Chipset	0.5MV/W	3.125ms	Not Reported	0.45K (Measured)
This work	0.18µm SiGe BiCMOS	9-Element Array with On-chip Antennas and Baseband	800MV/W	20ms	0.28fW/Hz$^{1/2}$	0.45K (Measured)

Chapter 5
Conclusions

The development of a new highly integrated low cost silicon-based passive imaging receiver suitable to integrated circuit realization up to sub-THz frequencies integrating the entire receiver frontend from antenna to detector was pursued, along with the development of a highly scalable architecture for use in focal plane arrays of arbitrary dimensions, incorporating new features to address large-array issues while retaining the pixel density of conventional focal plane arrays.

The proposed imaging receiver, incorporating a new high gain LNA, a new two-stage power detector and a new pixel architecture was designed, fabricated and tested. Receiver output noise tests along with active responsivity tests were performed demonstrating a maximum responsivity of 1150 MV/W and a minimum NEP of 0.28fW/Hz$^{1/2}$. Passive hot/cold tests were also performed and a thermal resolution of 0.45K was measured, indicating receiver sensitivity suitable for passive imaging applications. To the author's knowledge, this is the first reported measured NETD of a fully-integrated silicon radiometer containing integrated antennas directly measuring an object's temperature.

The proposed array architecture incorporating features to address large-array issues was tested. Employing the use of spatially overlapping super-pixels, the new architecture successfully demonstrated a focal plane array capable of beam-steering with the pixel density of a conventional focal plane array. The proposed receiver demonstrated 7-step amplitude and 9-step delay control for the new super-pixel, enabling beam steering angles from -18° to +18° with 4.5° angular resolution. The proposed architecture uses a new concept of spatial-overlapping super-pixels which results in (1) improved SNR at the pixel level through a reduction of spillover losses, (2) partially correlated adjacent super-pixels, (3) a 2×2 window averaging function in the

RF domain, (4) the ability to compensate for the systematic time delay and amplitude variations due to the off-focal-point effect for antennas away from the focal point, and (5) the ability to compensate for mutual coupling effects among the array elements.

Bibliography

[1] L. Yujiri, M. Shoucri, and P. Moffa, “Passive millimeter-wave imaging,” *IEEE Microwave Magazine*, vol. 4, pp. 39-50, Sep. 2003.

[2] J. W. May and G. M. Rebeiz, “Design and Characterization of W-Band SiGe RFICs for Passive Millimeter-Wave Imaging,” *IEEE Trans. Microwave Theory Tech*., vol. 58, no. 5, pp. 1420-1430, May 2010.

[3] A. Tomkins, P. Garcia, and S. P. Voinigescu, “A passive W-band imager in 65nm bulk CMOS,” *IEEE CSICS*, pp. 1-4, Oct. 2009.

[4] L. Zhou, et al., “A W-band CMOS Receiver Chipset for Millimeter-Wave Radiometer Systems,” *IEEE J. Solid-State Circuits,* vol. 46, no. 2, pp. 378-391, Feb. 2011.

[5] L. Gilreath, V. Jain, and P. Heydari, "Design and Analysis of a W-Band SiGe Direct-Detection-Based Passive Imaging Receiver," *IEEE J. Solid-State Circuits*, vol. 46, no. 10, pp. 2240-2252, Oct. 2011.

[6] Z. Chen, et al., "A BiCMOS W-Band 2×2 Focal-Plane Array with On-chip Antenna," *IEEE J. Solid-State Circuits*, vol. 47, no. 10, Oct. 2012.

[7] J. J. Lynch, et al., “Passive millimeter-wave imaging module with preamplified zero-bias detection,” *IEEE Trans. Microwave Theory Tech*., vol. 56, no. 7, pp. 1592–1600, July 2008.

[8] C. Martin, et al., “Rapid passive MMW security screening portal,” *Proceedings of SPIE Defense and Security 2008,* vol. 6948.

[9] D.C.W. Lo, et al., “A monolithic W-band high gain LNA/Detector for millimeter-wave radiometric imaging applications,” *IEEE MTT-S International Symposium*, pp. 1117-1120, June 1995.

[10] M. Tiuri, “Radio Astronomy Receivers,” *IEEE Trans. on Antennas and Propagation,* vol. 12, no. 7, pp. 930-938, Dec. 1964.

[11] R. H. Dicke, “The measurement of thermal radiation at microwave frequencies,” *The Review of Scientific Instruments*, vol. 17, no. 7, pp. 268-275, July 1946.

[12] H.Wang, et al., ”A Monolithic W-band Preamplified Diode Detector,” *Microwave Symposium Digest*, 1993, IEEE MTT-S International, pp. 365-368.

[13] J. Richard Fisher, “Phased Array Feeds for Low Noise Reflector Antennas,” National Radio Astronomy Observatory, Green Bank, West Virginia, 1996.

[14] R. M. Davis, et al., “A Scanning Reflector Using an Off-Axis Space-Fed Phased-Array Feed,” *IEEE Transactions on Antenna and Propagation*, 1991.

[15] A. V. Mrstik, et al., ”Scanning Capabilities of Large Parabolic Cylinder Reflector Antennas with Phased-Array Feeds,” *IEEE Transaction on Antenna and Propagation*, 1981.

[16] S. J. Blank, et al.,"Array Feed Synthesis for Correction of Reflector Distortion and Vernier Beamsteering," *IEEE Transaction on Antennas and Propagation*,1998.

[17] Y. Rahmat-samii, et al., "Directivity of Planar Array Feeds for Satellite Reflector Applications," *IEEE Transactions on Antennas and Propagation*, 1983.

[18] J. Landon, et al., "Phased Array Feed Calibration, Beamforming, and Imaging", *The Astronomical Journal*, 139:1154–1167, 2010 March.

[19] F. Caster, et al,. "A 93-to-113GHz BiCMOS 9-element imaging array receiver utilizing spatial-overlapping pixels with wideband phase and amplitude control", *International Solid-State Circuits Conference Digest of Technical Papers (ISSCC) 2013* , pp. 144-145.

[20] E. Dacquay, et al., "D-Band Total Power Radiometer Performance Optimization in an SiGe HBT Technology," *IEEE Transactions on Microwave Theory and Techniques*, vol. 60, no. 3, pp. 813-826, March 2012

[21] D. Tianwei, C. Zhiming, and Y. P. Zhang, "Coupling Mechanisms and Effects Between On-Chip Antenna and Inductor or Coplanar Waveguide," *IEEE Transactions on Electron Devices*, vol. 60, pp. 20-27, 2013.

[22] J. M. Edwards and G. M. Rebeiz, "High-Efficiency Elliptical Slot Antennas With Quartz Superstrates for Silicon RFICs," *IEEE Transactions on Antennas and Propagation*, vol. 60, pp. 5010-5020, 2012.

[23] S. Pan and F. Capolino, "Design of a CMOS On-Chip Slot Antenna With Extremely Flat Cavity at 140 GHz," *IEEE Antennas and Wireless Propagation Letters*, vol. 10, pp. 827-830, 2011.

[24] D. G. Kam, D. Liu, A. Natarajan, S. Reynolds, and B. A. Floyd, "Organic Packages With Embedded Phased-Array Antennas for 60-GHz Wireless Chipsets," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 1, no. 11, pp. 1806-1814, Nov. 2011.

[25] P. Gray, P. Hurst, S. Lewis, R. Meyer, *Analysis and Design of Analog Integrated Circuits*, John Wiley & Sons, 2009

[26] P. Heydari, "Design and Analysis of a Performance-Optimized CMOS UWB Distributed LNA," *IEEE J. Solid-State Circuits*, vol. 42, no. 9, pp. 1892-1905, Sept. 2007

[27] P. Chiang, et al., "A 200-GHz Inductively Tuned VCO With 7-dBm Output Power in 130-nm SiGe BiCMOS", *IEEE Transactions on Microwave Theory and Techniques*, vol. 61, no. 10, pp. 3666-3673, October 2013

[28] T. Lee, *The Design of CMOS Radio-Frequency Integrated Circuits*, Cambridge University Press, 2004.

[29] A. Safarian, L. Zhou, and P. Heydari, "CMOS Distributed Active Power Combiners and Splitters for Multi-Antenna UWB Beamforming Transceivers," *IEEE J. Solid-State Circuits*, vol. 42, no. 7, pp. 1481-1491, July 2007.

[30] L. Zheng, et al., "Design and Analysis of a W-Band Detector in 0.18-μm SiGe BiCMOS," *IEEE Silicon Monolithic Integrated Circuits in RF Systems*, Jan. 2010.

Appendix A

DAQ/GPIO Code

```
////////////////////////////////////////////////////////////////////////
//
// includes from Coridium
//
////////////////////////////////////////////////////////////////////////
#include "coridium_pcb.h"
#include "mem.h"
#include "string.h"
#include "printf.h"
#include "uart.h"
#include "breakpoint.h"
#include "cor_bitbang.h"
#include "flash.h"
#include "adc.h"
#define USE_FLOAT
#ifdef USE_FLOAT
#include <math.h>
#endif
////////////////////////////////////////////////////////////////////////
//
// New Code
//
////////////////////////////////////////////////////////////////////////

int buffer[ (512 >> 2) ];
int numConversions, intTime[8];

void resetChip(void)
{
  int i; // buffer[ (512 >> 2) ];
  // printf(" %s",instring);
  //printf("\n Reset chip, set all registers to 0.\n\n");
  // Put chip in default state
  // Set all IO to outputs
  for (i=0; i<16; i++) {
    OUTPUT(i);
  }
  // Set clock low
  LOW(9);
  // Reset integrator
  HIGH(8);
  // Reset registers
  HIGH(11);
  WAIT (1); //high 1 ms
  LOW(11);
  for (i=0; i < 44; i++) buffer[i] = 0;
}
```

```
void loadReg(int j, int k) // j is reg, k is data
{
  if (j & 32) HIGH(10) else LOW(10);
  if (j & 16) HIGH(0)  else LOW(0);
  if (j &  8) HIGH(1)  else LOW(1);
  if (j &  4) HIGH(4)  else LOW(4);
  if (j &  2) HIGH(2)  else LOW(2);
  if (j &  1) HIGH(6)  else LOW(6);
  if (k &  4) HIGH(3)  else LOW(3);
  if (k &  2) HIGH(7)  else LOW(7);
  if (k &  1) HIGH(5)  else LOW(5);
  WAIT(2);
  HIGH(9); // Toggle clock
  WAIT(2);
  LOW(9);
  buffer[j] = k;
}

void TTDPATH(int path, int ttdreg1, int ttdreg2, int waitTime)
{
  int conv[8],i,j,k,l;
  conv[0]=0;
  conv[1]=0;
  conv[2]=0;
  conv[3]=0;
  conv[4]=0;
  conv[5]=0;
  conv[6]=0;
  conv[7]=0;
  loadReg(path-1,4);
  //Set TTF reg1
  for (j=3; j>-1; j--){ //write data 3 thru 0 for ttdreg1a
    //address
    loadReg(ttdreg1,j);
    for (i=3; i>-1; i--){ //write data 3 thru 0 for ttdreg1b
      loadReg(ttdreg2,i);
      WAIT(waitTime); // give chip time to settle after reg data change
      for (l=0; l<numConversions; l++){
        for (k=4; k<8; k++){   // only channels 1 and 2
          LOW(8); // start conversion
          WAIT(intTime[k]); // wait 30 ms
          conv[k] = AD(k)>>6; // save data
          HIGH(8)
          WAIT(intTime[k]);
        }
        printf("  %2d      4      %2d     %2d      %2d    %2d   %4d  %4d  %4d
%4d  %4d  %4d  %4d  %4d\n", path-1,
           ttdreg1, j, ttdreg2, i, conv[7], conv[6], conv[5], conv[4],
conv[3], conv[2], conv[1], conv[0]);
      }
    }
  }
  loadReg(path-1,0);
  loadReg(ttdreg1,0);
  loadReg(ttdreg2,0);
}
```

```
void TTDPATH2(int path1, int ttdreg1a, int ttdreg1b, int path2, int ttdreg2a,
int ttdreg2b, int waitTime)
{
  int conv[8],i,j,k,l;
  conv[0]=0;
  conv[1]=0;
  conv[2]=0;
  conv[3]=0;
  conv[4]=0;
  conv[5]=0;
  conv[6]=0;
  conv[7]=0;
  loadReg(path1-1,4);
  loadReg(path2-1,4);
  //Set TTF reg1
  for (j=3; j>-1; j--){ //write data 3 thru 0 for ttdreg1a
    //address
    loadReg(ttdreg1a,j);
    for (i=3; i>-1; i--){ //write data 3 thru 0 for ttdreg1b
      loadReg(ttdreg1b,i);
      WAIT(waitTime); // give chip time to settle after reg data change
      for (l=0; l<numConversions; l++){
        for (k=4; k<8; k++){  // only channels 1 and 2
          LOW(8); // start conversion
          WAIT(intTime[k]); // wait 30 ms
          conv[k] = AD(k)>>6; // save data
          HIGH(8)
          WAIT(intTime[k]);
        }
        printf("  %2d      4      %2d     %2d      %2d    %2d   %4d  %4d  %4d
%4d  %4d  %4d  %4d  %4d\n", path1-1,
           ttdreg1a, j, ttdreg1b, i, conv[7], conv[6], conv[5], conv[4],
conv[3], conv[2], conv[1], conv[0]);
      }
    }
  }
  loadReg(ttdreg1a,0);
  loadReg(ttdreg1b,0);
  for (j=3; j>-1; j--){ //write data 3 thru 0 for ttdreg2a
    loadReg(ttdreg2a,j);
    for (i=3; i>-1; i--){ //write data 3 thru 0 for ttdreg2b
      loadReg(ttdreg2b,i);
      WAIT(waitTime); // give chip time to settle after reg data change
      for (l=0; l<numConversions; l++){
        for (k=4; k<8; k++){  // only channels 1 and 2
          LOW(8); // start conversion
          WAIT(intTime[k]); // wait 30 ms
          conv[k] = AD(k)>>6; // save data
          HIGH(8)
          WAIT(intTime[k]);
        }
        printf("  %2d      4      %2d     %2d      %2d    %2d   %4d  %4d  %4d
%4d  %4d  %4d  %4d  %4d\n", path2-1,
           ttdreg2a, j, ttdreg2b, i, conv[7], conv[6], conv[5], conv[4],
conv[3], conv[2], conv[1], conv[0]);
      }
```

```
    }
  }
  //Set appropriate reg back to 0
  loadReg(path1-1,0);
  loadReg(path2-1,0);
  loadReg(ttdreg2a,0);
  loadReg(ttdreg2b,0);
}

void TTDPATH3(int path1, int ttdreg1a, int ttdreg1b, int code1, int path2,
int ttdreg2a, int ttdreg2b, int code2, int waitTime)
{
  int conv[8],i,j,k,l;
  conv[0]=0;
  conv[1]=0;
  conv[2]=0;
  conv[3]=0;
  conv[4]=0;
  conv[5]=0;
  conv[6]=0;
  conv[7]=0;
  loadReg(path1-1,code1);
  loadReg(path2-1,code2);
  //Set TTF reg1
  for (j=3; j>-1; j--){ //write data 3 thru 0 for ttdreg1a
    //address
    loadReg(ttdreg1a,j);
    for (i=3; i>-1; i--){ //write data 3 thru 0 for ttdreg1b
      loadReg(ttdreg1b,i);
      WAIT(waitTime); // give chip time to settle after reg data change
      for (l=0; l<numConversions; l++){
        for (k=4; k<8; k++){  // only channels 1 and 2
          LOW(8); // start conversion
          WAIT(intTime[k]); // wait 30 ms
          conv[k] = AD(k)>>6; // save data
          HIGH(8)
          WAIT(intTime[k]);
        }
        printf("  %2d %2d  %2d %2d   %2d   %2d      %2d    %2d   %4d  %4d
%4d  %4d  %4d  %4d  %4d  %4d  %4d\n", path1-1,
        code1,path2-1,code2,   ttdreg1a, j, ttdreg1b, i, conv[7], conv[6],
conv[5], conv[4], conv[3], conv[2], conv[1], conv[0]);
      }
    }
  }
  loadReg(ttdreg1a,0);
  loadReg(ttdreg1b,0);
  for (j=3; j>-1; j--){ //write data 3 thru 0 for ttdreg2a
    loadReg(ttdreg2a,j);
    for (i=3; i>-1; i--){ //write data 3 thru 0 for ttdreg2b
      loadReg(ttdreg2b,i);
      WAIT(waitTime); // give chip time to settle after reg data change
      for (l=0; l<numConversions; l++){
        for (k=4; k<8; k++){  // only channels 1 and 2
          LOW(8); // start conversion
          WAIT(intTime[k]); // wait 30 ms
          conv[k] = AD(k)>>6; // save data
```

```
          HIGH(8)
          WAIT(intTime[k]);
        }
        printf("  %2d  %2d  %2d  %2d  %2d   %2d      %2d   %2d   %4d  %4d
%4d  %4d  %4d  %4d  %4d  %4d\n", path1-1, code1,path2-1,
           code2, ttdreg2a, j, ttdreg2b, i, conv[7], conv[6], conv[5],
conv[4], conv[3], conv[2], conv[1], conv[0]);
      }
    }
  }
  //Set appropriate reg back to 0
  loadReg(path1-1,0);
  loadReg(path2-1,0);
  loadReg(ttdreg2a,0);
  loadReg(ttdreg2b,0);
}


void calPath(int angle,int v5,int v6,int v7,int v8,int t18,int t20,int
t21,int t22,int t27,int t30,int t33,int t35,int waitTime)
{
  int conv[8],k,l;
  conv[0]=0;
  conv[1]=0;
  conv[2]=0;
  conv[3]=0;
  conv[4]=0;
  conv[5]=0;
  conv[6]=0;
  conv[7]=0;
  loadReg(4,v5);
  loadReg(5,v6);
  loadReg(6,v7);
  loadReg(7,v8);
  loadReg(18,t18);
  loadReg(20,t20);
  loadReg(21,t21);
  loadReg(22,t22);
  loadReg(27,t27);
  loadReg(30,t30);
  loadReg(33,t33);
  loadReg(35,t35);
  WAIT(waitTime); // give chip time to settle after reg data change
  for (l=0; l<numConversions; l++){
    for (k=4; k<6; k++){  // only channels 1 and 2 if k=4
      LOW(8); // start conversion
      WAIT(intTime[k]); //
      conv[k] = AD(k)>>6; // save data
          HIGH(8)
      WAIT(intTime[k]);
    }
      printf("  %3d   %d %d %d %d    %d %d    %d %d   %d %d   %d %d   %4d
%4d  %4d  %4d  %4d  %4d  %4d  %4d\n", angle,
    v5,v6,v7,v8, t18,t20,t21,t22,t27,t30,t33,t35,
      conv[7], conv[6], conv[5], conv[4], conv[3], conv[2], conv[1],
conv[0]);
  }
```

```
}


void PhAnt(int path1, int path2,int waitTime)
{
  int conv[8],i,k;
  //turn on two VGA REGS
  i=path1-1;
  conv[0]=0;
  conv[1]=0;
  conv[2]=0;
  conv[3]=0;
  conv[4]=0;
  conv[5]=0;
  conv[6]=0;
  conv[7]=0;

  //Set appropriate VGA reg to 4
  buffer[i]=4;
  if ((i & 32)>>5) HIGH(10) else LOW(10);
  if ((i & 16)>>4) HIGH(0)  else LOW(0);
  if ((i &  8)>>3) HIGH(1)  else LOW(1);
  if ((i &  4)>>2) HIGH(4)  else LOW(4);
  if ((i &  2)>>1) HIGH(2)  else LOW(2);
  if (i &  1) HIGH(6)  else LOW(6);
  HIGH(3);
  LOW(7);
  LOW(5);
  HIGH(9); // Toggle clock to write data and address
  WAIT(1);
  LOW(9);
  i=path2-1;
  //Set appropriate VGA reg to 4
  buffer[i]=4;
  if ((i & 32)>>5) HIGH(10) else LOW(10);
  if ((i & 16)>>4) HIGH(0)  else LOW(0);
  if ((i &  8)>>3) HIGH(1)  else LOW(1);
  if ((i &  4)>>2) HIGH(4)  else LOW(4);
  if ((i &  2)>>1) HIGH(2)  else LOW(2);
  if (i &  1) HIGH(6)  else LOW(6);
  HIGH(3);
  LOW(7);
  LOW(5);
  HIGH(9); // Toggle clock to write data and address
  WAIT(1);
  LOW(9);
  WAIT(waitTime); // give chip time to settle after reg data change
  // do conversions
  for (i=0; i<numConversions; i++){
        for (k=6; k<8; k++){
      LOW(8); // start conversion
      WAIT(intTime[k]); // wait 30 ms
      conv[k] = AD(k)>>6; // save data
      HIGH(8);
      WAIT(intTime[k]);
    }
```

```
    printf("  %2d      4      %2d      4       0     0   %4d  %4d  %4d  %4d
%4d  %4d  %4d  %4d\n", path1-1,
      path2-1, conv[7], conv[6], conv[5], conv[4], conv[3], conv[2], conv[1],
conv[0]);
  }
  resetChip();
}

//////////////////////////////////////////////////////////////////////
//
// Main program
//
//////////////////////////////////////////////////////////////////////

int main(void)
{
  char instring[256];
  int   inval,i, save_time, port,bit, waitTime;
  int  conv[8],sum[8], j, k, l; // intTime, numConversions;
#ifdef USE_FLOAT
  float x ;
#endif

      ////////////////////////////////////////////////////////////////////
      //
      //  Coridium code
      //
      ////////////////////////////////////////////////////////////////////
  int present;
  char shortMessage[20];
  char shortResponse[20];
  int  wordMessage[20];
  int  wordResponse[20];
  SystemInit();
  UART_init(0,19200);
  timer_tick_init();
#if defined LPC178x
#warning ADinit problem 178x
#elif (!defined ARMexpress) && (!defined LPC2106)
  initAD();            // turn on the AD converters
#endif

    ////////////////////////////////////////////////////////////////////
    //
    //  Initialize variables
    //
    ////////////////////////////////////////////////////////////////////
    intTime[0]=30;
    intTime[1]=30;
    intTime[2]=30;
    intTime[3]=30;
    intTime[4]=30;
    intTime[5]=30;
    intTime[6]=30;
    intTime[7]=30;
    waitTime=200;
  numConversions=3;
```

```
conv[0]=0;
conv[1]=0;
conv[2]=0;
conv[3]=0;
conv[4]=0;
conv[5]=0;
conv[6]=0;
conv[7]=0;

    ////////////////////////////////////////////////////////////////////////
    //
    //  Begin Main Loop
    //
    ////////////////////////////////////////////////////////////////////////
printf("\n\nBegin Testing\n");
// Put chip in default state
// Set all IO to outputs
for (i=0; i<16; i++) {
  OUTPUT(i);
}
HIGH(15);
resetChip();
while (1) {
      printf("\nEnter option:");
      gets(instring);
      inval = atoi(instring);

      switch (inval) {
  case 1000000:
        printf("LED on\n");
        OUTPUT(15);
        LOW(15);
        break;

  case 2000000:
        printf("LED off\n");
    OUTPUT(15);
        HIGH(15);
        break;

  case 4000000:
        printf("Blink LED for 500ms\n");
        OUTPUT(15);
        HIGH(15);
        for (i=0; i<20; i++) {
          if(i&1) HIGH(15) else LOW(15);
    WAIT (500);
    }
        break;

  case 5000000:
        // Put chip in default state
        // Set all IO to outputs
        for (i=0; i<16; i++) {
      OUTPUT(i);
        }
        resetChip();
```

```
    printf("\n Wiggle a pin for 10 seconds\n");
    printf("\n enter which pin (0 to 11):");
    gets(instring);
    bit = atoi(instring);
    OUTPUT(bit);
    for (i=0; i<1000; i++) {
      HIGH(bit);
  WAIT (waitTime);
      LOW(bit)
      WAIT (waitTime);
    }
    HIGH(bit);
    break;

case 5000001:
    // set int_time
  printf("\n Enter Integration time for all channels:");
  gets(instring);
  bit = atoi(instring);
  intTime[0] = bit;
  printf("\nIntegration time is now %d for all channels", intTime[0]);
  intTime[1]=bit;
  intTime[2]=bit;
  intTime[3]=bit;
  intTime[4]=bit;
  intTime[5]=bit;
  intTime[6]=bit;
  intTime[7]=bit;
  break;

case 5000002:
    // set number of conversions
  gets(instring);
  bit = atoi(instring);
  numConversions = bit;
  printf("\nNumber of Conversions is now %d", numConversions);
  break;

case 5000021:
    // set int_time
  printf("\nEnter Integration time for integrator 1");
  gets(instring);
  bit = atoi(instring);
  intTime[6] = bit;
  intTime[7] = bit;
  printf("\nIntegration time for Integrator 1 is now %d", intTime[6]);
  //break;

//case 5000022:
    // set int_time
  printf("\nEnter Integration time for integrator 2");
  gets(instring);
  bit = atoi(instring);
  intTime[4] = bit;
  intTime[5] = bit;
  printf("\nIntegration time for Integrator 2 is now %d", intTime[4]);
  //break;
```

```
      //case 5000023:
          // set int_time
        printf("\nEnter Integration time for integrator 3");
        gets(instring);
        bit = atoi(instring);
        intTime[2] = bit;
        intTime[3] = bit;
        printf("\nIntegration time for Integrator 3 is now %d", intTime[2]);
        //break;

      //case 5000024:
          // set int_time
        printf("\nEnter Integration time for integrator 4");
        gets(instring);
        bit = atoi(instring);
        intTime[0] = bit;
        intTime[1] = bit;
        printf("\nIntegration time for Integrator 4 is now %d", intTime[0]);
        break;


      case 6000000:
        // Load regs
        for (i=0; i<44; i++) {
          gets(instring);
          bit = atoi(instring);
          buffer[i] = bit;
          loadReg(i,bit);
        }
        break;

    case 7000000:
        // Load regs
        for (i=0; i<44; i++) {
          printf("\n%d",buffer[i]);
        //printf("\n reg%2d=%3d", i,buffer[i]);
        }
        break;

    case 8000000:
        // Read INT output from all super pixels, 1 INT at a time
        printf("\n INT1P INT1N INT2P INT2N INT3P INT3N INT4P INT4N");
        for (k=0; k<8; k++){sum[k]=0;}
        for (i=0; i<numConversions; i++) {
          for (k=0; k<8; k++){
            LOW(8); // start conversion
            WAIT(intTime[k]); // wait 30 ms
            conv[k] = AD(k)>>6; // save data
            sum[k]=sum[k]+conv[k];
            HIGH(8);
            WAIT(intTime[k]);
          }
          printf("\n    %2d     %4d  %4d  %4d  %4d  %4d  %4d  %4d  %4d", i+1,
conv[7], conv[6], conv[5],
              conv[4], conv[3], conv[2], conv[1], conv[0]);
        }
```

```
        printf("\n ============================================= ");
        printf("\n           %4d  %4d  %4d  %4d  %4d  %4d  %4d  %4d",
10*sum[7]/i, 10*sum[6]/i, 10*sum[5]/i,
             10*sum[4]/i, 10*sum[3]/i, 10*sum[2]/i, 10*sum[1]/i,
10*sum[0]/i);
        printf("\n  ");
        printf("\n*   %4d    %4d    %4d    %4d", 10*sum[7]/i-10*sum[6]/i,
            10*sum[5]/i-10*sum[4]/i, 10*sum[3]/i-10*sum[2]/i, 10*sum[1]/i-
10*sum[0]/i);
        printf("\nComplete");
        break;


      case 9000000:
        printf("\n Enter pin(0-11):");
        gets(instring); // printf(" %s",instring);
        bit = atoi(instring);
        printf("\n High or Low(h/l):");
        gets(instring);printf(" %s",instring);
        if (instring == "h")
        {HIGH(bit);
         printf("\nIO%d is now high",bit);
         }
         else
         {LOW(bit);
         printf("\nIO%d is now high",bit);
         }
       // endif
          break;

      case 9000001:
        printf("\n Enter AD pin(0-7):");
        gets(instring); // printf(" %s",instring);
        bit = atoi(instring);
        AD(bit)>>6;
        printf("\nAD%d command entered",bit);
              for (l=0; l<numConversions; l++){
        for (k=0; k<8; k++){
          LOW(8); // start conversion
          WAIT(intTime[k]); // wait 30 ms
          conv[k] = AD(k)>>6; // save data
          HIGH(8)
          WAIT(intTime[k]);
        }
        printf("%2d  %4d  %4d  %4d  %4d  %4d  %4d  %4d  %4d\n", l, conv[7],
conv[6], conv[5], conv[4], conv[3], conv[2], conv[1], conv[0]);
           break;
    }


      case 10000000:
        // printf(" %s",instring);
        printf("\n Reset chip, set all registers to 0.\n\n");
        resetChip();
        break;

      case 11000002:
```

```
        printf("\n Enter State Change Wait Time 0->5000(%d):", waitTime);
        gets(instring); // printf(" %s",instring);
        bit = atoi(instring);
        if ((bit >=0) & (bit <= 50000)) waitTime=bit; else printf("\n Wait
time not changed.");
      break;

      case 11000004:
        printf("\n New Calibration Code -- Only pixel 2 active");
        printf("\n Enter State Change Wait Time (%d):", waitTime);
        gets(instring); // printf(" %s",instring);
        bit = atoi(instring);
        if ((bit >=0) & (bit <= 50000)) waitTime=bit; else printf("\n Wait
time not changed.");
        // Set all IO to outputs
         // only channels 1 and 2 active
        for (i=0; i<16; i++) {
          OUTPUT(i);
        }
        resetChip();
        calPath(0, 4,5,5,4, 0,0, 0,0, 0,0, 0,0, waitTime);
        calPath(4, 4,4,5,4, 1,0, 0,2, 0,0, 0,0, waitTime);
        calPath(4, 4,4,4,4, 0,1, 0,2, 0,0, 0,0, waitTime);
        calPath(4, 4,4,4,4, 2,0, 0,2, 0,0, 0,0, waitTime);
        calPath(4, 4,4,6,4, 3,0, 0,2, 0,0, 0,0, waitTime);
        calPath(4, 4,6,5,4, 1,0, 0,3, 0,0, 0,0, waitTime);
        calPath(4, 4,6,4,4, 0,1, 0,3, 0,0, 0,0, waitTime);
        calPath(4, 4,6,4,4, 2,0, 0,3, 0,0, 0,0, waitTime);
        calPath(4, 4,6,6,4, 3,0, 0,3, 0,0, 0,0, waitTime);
        calPath(4, 4,4,5,4, 1,0, 2,0, 0,0, 0,0, waitTime);
        calPath(4, 4,4,4,4, 0,1, 2,0, 0,0, 0,0, waitTime);
        calPath(4, 4,4,4,4, 2,0, 2,0, 0,0, 0,0, waitTime);
        calPath(4, 4,4,6,4, 3,0, 2,0, 0,0, 0,0, waitTime);
        calPath(8, 4,7,5,4, 1,0, 1,3, 0,0, 0,0, waitTime);
        calPath(8, 4,7,5,4, 0,1, 1,3, 0,0, 0,0, waitTime);
        calPath(8, 4,7,5,4, 2,0, 1,3, 0,0, 0,0, waitTime);
        calPath(8, 4,7,5,4, 3,0, 1,3, 0,0, 0,0, waitTime);
        calPath(8, 4,7,5,4, 1,0, 2,3, 0,0, 0,0, waitTime);
        calPath(8, 4,7,5,4, 0,1, 2,3, 0,0, 0,0, waitTime);
        calPath(8, 4,7,5,4, 2,0, 2,3, 0,0, 0,0, waitTime);
        calPath(8, 4,7,5,4, 3,0, 2,3, 0,0, 0,0, waitTime);
        calPath(8, 4,7,5,4, 1,0, 1,2, 0,0, 0,0, waitTime);
        calPath(8, 4,7,5,4, 0,1, 1,2, 0,0, 0,0, waitTime);
        calPath(8, 4,7,5,4, 2,0, 1,2, 0,0, 0,0, waitTime);
        calPath(8, 4,7,5,4, 3,0, 1,2, 0,0, 0,0, waitTime);
        calPath(8, 4,7,5,4, 1,0, 3,3, 0,0, 0,0, waitTime);
        calPath(8, 4,7,5,4, 0,1, 3,3, 0,0, 0,0, waitTime);
        calPath(8, 4,7,5,4, 2,0, 3,3, 0,0, 0,0, waitTime);
        calPath(8, 4,7,5,4, 2,0, 3,3, 0,0, 0,0, waitTime);
        calPath(13, 4,7,4,4, 1,1, 1,3, 0,0, 0,0, waitTime);
        calPath(13, 4,7,4,4, 2,1, 1,3, 0,0, 0,0, waitTime);
        calPath(13, 4,7,5,4, 3,1, 1,3, 0,0, 0,0, waitTime);
        calPath(13, 4,5,4,5, 1,1, 2,3, 0,0, 0,0, waitTime);
        calPath(13, 4,5,4,5, 2,1, 2,3, 0,0, 0,0, waitTime);
        calPath(13, 4,5,5,5, 3,1, 2,3, 0,0, 0,0, waitTime);
        calPath(13, 4,5,4,4, 1,1, 1,2, 0,0, 0,0, waitTime);
        calPath(13, 4,5,4,4, 2,1, 1,2, 0,0, 0,0, waitTime);
```

```
  calPath(13, 4,5,5,4, 3,1, 1,2, 0,0, 0,0, waitTime);
  calPath(13, 4,6,4,4, 1,1, 3,3, 0,0, 0,0, waitTime);
  calPath(13, 4,6,4,4, 2,1, 3,3, 0,0, 0,0, waitTime);
  calPath(13, 4,6,5,4, 3,1, 3,3, 0,0, 0,0, waitTime);
  calPath(20, 5,7,7,4, 2,3, 1,3, 0,0, 0,0, waitTime);
  calPath(20, 5,7,7,4, 3,3, 1,3, 0,0, 0,0, waitTime);
  calPath(20, 5,7,6,4, 1,3, 1,3, 0,0, 0,0, waitTime);
  calPath(20, 5,5,7,5, 2,3, 2,3, 0,0, 0,0, waitTime);
  calPath(20, 5,5,7,5, 3,3, 2,3, 0,0, 0,0, waitTime);
  calPath(20, 5,5,6,5, 1,3, 2,3, 0,0, 0,0, waitTime);
  calPath(20, 5,5,7,4, 2,3, 1,2, 0,0, 0,0, waitTime);
  calPath(20, 5,5,7,4, 3,3, 1,2, 0,0, 0,0, waitTime);
  calPath(20, 5,5,6,4, 1,3, 1,2, 0,0, 0,0, waitTime);
  calPath(20, 5,6,7,4, 2,3, 3,3, 0,0, 0,0, waitTime);
  calPath(20, 5,6,7,4, 3,3, 3,3, 0,0, 0,0, waitTime);
  calPath(20, 5,6,6,4, 1,3, 3,3, 0,0, 0,0, waitTime);
  calPath(-4, 4,5,5,4, 0,0, 0,0, 0,1, 3,1, waitTime);
  calPath(-4, 4,5,5,4, 0,0, 0,0, 1,0, 3,1, waitTime);
  calPath(-4, 5,5,5,4, 0,0, 0,0, 3,0, 3,1, waitTime);
  calPath(-4, 4,5,5,5, 0,0, 0,0, 0,1, 3,3, waitTime);
  calPath(-4, 4,5,5,5, 0,0, 0,0, 1,0, 3,3, waitTime);
  calPath(-4, 5,5,5,5, 0,0, 0,0, 3,0, 3,3, waitTime);
  calPath(-8, 4,6,5,4, 0,0, 0,0, 0,1, 1,1, waitTime);
  calPath(-8, 4,6,5,4, 0,0, 0,0, 1,0, 1,1, waitTime);
  calPath(-8, 5,6,5,4, 0,0, 0,0, 3,0, 1,1, waitTime);
  calPath(-8, 4,6,5,4, 0,0, 0,0, 0,1, 2,3, waitTime);
  calPath(-8, 4,6,5,4, 0,0, 0,0, 1,0, 2,3, waitTime);
  calPath(-8, 5,6,5,4, 0,0, 0,0, 3,0, 2,3, waitTime);
  calPath(-8, 4,5,5,4, 0,0, 0,0, 0,1, 2,2, waitTime);
  calPath(-8, 4,5,5,4, 0,0, 0,0, 1,0, 2,2, waitTime);
  calPath(-8, 5,5,5,4, 0,0, 0,0, 3,0, 2,2, waitTime);
  calPath(-8, 4,7,5,4, 0,0, 0,0, 0,1, 1,2, waitTime);
  calPath(-8, 4,7,5,4, 0,0, 0,0, 1,0, 1,2, waitTime);
  calPath(-8, 5,7,5,4, 0,0, 0,0, 3,0, 1,2, waitTime);
  calPath(-13, 4,7,6,7, 0,0, 0,0, 2,2, 1,3, waitTime);
  calPath(-13, 4,7,5,7, 0,0, 0,0, 0,3, 1,3, waitTime);
  calPath(-13, 4,7,6,7, 0,0, 0,0, 3,1, 1,3, waitTime);
  calPath(-13, 4,6,6,6, 0,0, 0,0, 2,2, 2,3, waitTime);
  calPath(-13, 4,6,5,6, 0,0, 0,0, 0,3, 2,3, waitTime);
  calPath(-13, 4,6,6,6, 0,0, 0,0, 3,1, 2,3, waitTime);
  calPath(-13, 4,6,6,4, 0,0, 0,0, 2,2, 3,2, waitTime);
  calPath(-13, 4,6,5,4, 0,0, 0,0, 0,3, 3,2, waitTime);
  calPath(-13, 4,6,6,4, 0,0, 0,0, 3,1, 3,2, waitTime);
  calPath(-20, 4,7,6,7, 0,0, 0,0, 1,3, 1,3, waitTime);
  calPath(-20, 4,7,6,7, 0,0, 0,0, 3,3, 1,3, waitTime);
  calPath(-20, 6,6,6,6, 0,0, 0,0, 1,3, 2,3, waitTime);
  calPath(-20, 4,6,6,6, 0,0, 0,0, 3,3, 2,3, waitTime);
  calPath(-20, 6,6,6,4, 0,0, 0,0, 1,3, 3,2, waitTime);
  calPath(-20, 4,6,6,4, 0,0, 0,0, 3,3, 3,2, waitTime);
  //printf("Complete\n\n");
  break;


case 11000000:
  printf("\n Enter State Change Wait Time (%d):", waitTime);
  gets(instring); // printf(" %s",instring);
  bit = atoi(instring);
```

```
        if ((bit >=0) & (bit <= 50000)) waitTime=bit; else printf("\n Wait
time not changed.");
        // Set all IO to outputs
         // only channels 1 and 2 active
        for (i=0; i<16; i++) {
          OUTPUT(i);
        }
        resetChip();
        TTDPATH3(5,18,20,4,7,27,30,4,waitTime);
        TTDPATH3(5,18,20,4,7,27,30,5,waitTime);
        TTDPATH3(5,18,20,4,7,27,30,6,waitTime);
        TTDPATH3(5,18,20,4,7,27,30,7,waitTime);
        TTDPATH3(6,21,22,4,8,33,35,4,waitTime);
        TTDPATH3(6,21,22,5,8,33,35,4,waitTime);
        TTDPATH3(6,21,22,6,8,33,35,4,waitTime);
        TTDPATH3(6,21,22,7,8,33,35,4,waitTime);
        //printf("Complete\n\n");
        break;

      case 11000003:
        printf("\n Enter State Change Wait Time (%d):", waitTime);
        gets(instring); // printf(" %s",instring);
        bit = atoi(instring);
        if ((bit >=0) & (bit <= 50000)) waitTime=bit; else printf("\n Wait
time not changed.");
        // Set all IO to outputs
         // only channels 1 and 2 active
        for (i=0; i<16; i++) {
          OUTPUT(i);
        }
        resetChip();
        TTDPATH(1,16,17,waitTime);
        TTDPATH(2,18,19,waitTime);
        TTDPATH(3,23,25,waitTime);
        TTDPATH(4,27,29,waitTime);
        TTDPATH(5,18,20,waitTime);
        TTDPATH(6,21,22,waitTime);
        TTDPATH(7,27,30,waitTime);
        TTDPATH(8,33,35,waitTime);
        // part 2
        TTDPATH2(1,16,17,3,23,25,waitTime);
        TTDPATH2(2,18,19,4,27,29,waitTime);
        TTDPATH2(5,18,20,7,27,30,waitTime);
        TTDPATH2(6,21,22,8,33,35,waitTime);

        //printf("Complete\n\n");
        break;

      case 11000001:
          // only channels 1 and 2 active
        //printf(" %s",instring);
        //printf("\n Test1: Check gain of each indepent VGA path\n\n");
        // Put chip in default state
        // Set all IO to outputs
        for (i=0; i<16; i++) {
          OUTPUT(i);
        }
```

```
         resetChip();
        // Read INT output from all antennas, 1 path at a time
        // enabling only one VGA at a time
        // All VGAs should be set to 000 due to reset
        //printf("\nPAFPA Calibration Data\n");
        //printf("\n\nPart 1: Single Path Variation due to VGA reg, all TTD
reg=0\n");
        // printf("\n\nPath VGAReg Data INT1P INT1N INT2P INT2N INT3P INT3N
INT4P INT4N\n");
        //printf("\n\n VGAReg Data TTD1  CODE1  TTD2 CODE2  INT1P INT1N INT2P
INT2N INT3P INT3N INT4P INT4N\n");
        //printf("\n\n");
        for (i=0; i<4; i++){ //i for addresses
          if ((i & 32)>>5) HIGH(10) else LOW(10);
          if ((i & 16)>>4) HIGH(0)  else LOW(0);
          if ((i &  8)>>3) HIGH(1)  else LOW(1);
          if ((i &  4)>>2) HIGH(4)  else LOW(4);
          if ((i &  2)>>1) HIGH(2)  else LOW(2);
          if (i &  1) HIGH(6)  else LOW(6);
          for (j=7; j>-1; j--){ //j for data high to low to leave reg off
when loop done
            if ((j &  4)>>2) HIGH(3)  else LOW(3);
            if ((j &  2)>>1) HIGH(7)  else LOW(7);
            if (j &  1) HIGH(5)  else LOW(5);
            HIGH(9); // Toggle clock to write data and address
            WAIT(1);
            LOW(9);
            WAIT(waitTime); // give chip time to settle after reg data change
            for (l=0; l<numConversions; l++){
              for (k=6; k<8; k++){
                LOW(8); // start conversion
                WAIT(intTime[k]); // wait 30 ms
                conv[k] = AD(k)>>6; // save data
                HIGH(8);
                WAIT(intTime[k]);
              }
              printf("   %2d    %2d    0     0     0    0    %4d  %4d  %4d
%4d  %4d  %4d  %4d  %4d\n", i, j, conv[7],
                conv[6], conv[5], conv[4], conv[3], conv[2], conv[1],
conv[0]);
            }
          }
        }
        //printf("\nPart 2: Single Path Variation due to TTD reg, all VGA
reg=4\n");
        // printf("\nPath VGAReg TTDREG1 Data1 TTDREG2 Data2 INT1P INT1N
INT2P INT2N INT3P INT3N INT4P INT4N\n\n");
        //printf("\nVGAReg VGAdata TTDREG1 Data1 TTDREG2 Data2 INT1P INT1N
INT2P INT2N INT3P INT3N INT4P INT4N\n\n");
        resetChip();
        TTDPATH(1,16,17,waitTime);
        TTDPATH(2,18,19,waitTime);
        TTDPATH(3,23,25,waitTime);
        TTDPATH(4,27,29,waitTime);
        // TTDPATH(5,18,20,waitTime);
        // TTDPATH(6,21,22,waitTime);
        // TTDPATH(7,27,30,waitTime);
```

```
        // TTDPATH(8,33,35,waitTime);
        // TTDPATH(9,24,26,waitTime);
        // TTDPATH(10,28,31,waitTime);
        // TTDPATH(11,37,38,waitTime);
        // TTDPATH(12,39,40,waitTime);
        // TTDPATH(13,28,32,waitTime);
        // TTDPATH(14,34,36,waitTime);
        // TTDPATH(15,39,41,waitTime);
        // TTDPATH(16,42,43,waitTime);

        //printf("\n\n\nPart 3: Two Paths on for PhAnt, All TTD=0\n");
        //printf("\nVGAReg VGAdata TTDREG1 Data1 TTDREG2 Data2 INT1P INT1N
INT2P INT2N INT3P INT3N INT4P INT4N\n\n");
        resetChip();
        PhAnt(1,3,waitTime);
        PhAnt(1,2,waitTime);
        PhAnt(2,4,waitTime);
        PhAnt(3,4,waitTime);
        // PhAnt(5,7,waitTime);
        // PhAnt(5,6,waitTime);
        // PhAnt(6,8,waitTime);
        // PhAnt(7,8,waitTime);
        // PhAnt(9,11,waitTime);
        // PhAnt(9,10,waitTime);
        // PhAnt(10,12,waitTime);
        // PhAnt(11,12,waitTime);
        // PhAnt(13,15,waitTime);
        // PhAnt(13,14,waitTime);
        // PhAnt(14,16,waitTime);
        // PhAnt(15,16,waitTime);

        //printf("\n\n\nPart 4: Two Path Variation for PhTTD, Paths 2 and 4
on, sweep TTD18, TTD19\n");
        //printf("\nVGAReg VGAdata TTDREG1 Data1 TTDREG2 Data2 INT1P INT1N
INT2P INT2N INT3P INT3N INT4P INT4N\n\n");
        resetChip();
        //Set appropriate VGA reg to 4
        i=1; // path 2 on
        if ((i & 32)>>5) HIGH(10) else LOW(10);
        if ((i & 16)>>4) HIGH(0)  else LOW(0);
        if ((i &  8)>>3) HIGH(1)  else LOW(1);
        if ((i &  4)>>2) HIGH(4)  else LOW(4);
        if ((i &  2)>>1) HIGH(2)  else LOW(2);
        if (i &  1) HIGH(6)  else LOW(6);
        HIGH(3);
        LOW(7);
        LOW(5);
        HIGH(9); // Toggle clock to write data and address
        WAIT(1);
        LOW(9);
        i=3; // path 4 on
        if ((i & 32)>>5) HIGH(10) else LOW(10);
        if ((i & 16)>>4) HIGH(0)  else LOW(0);
        if ((i &  8)>>3) HIGH(1)  else LOW(1);
        if ((i &  4)>>2) HIGH(4)  else LOW(4);
        if ((i &  2)>>1) HIGH(2)  else LOW(2);
        if (i &  1) HIGH(6)  else LOW(6);
```

```
        HIGH(3);
        LOW(7);
        LOW(5);
        HIGH(9); // Toggle clock to write data and address
        WAIT(1);
        LOW(9);
        //TTF Variation now
        //ttdreg1 = 18;
        //ttdreg2 = 19;
        LOW(10);// add5
        HIGH(0);// add4
        LOW(1); // add3
        LOW(4); // add2
        HIGH(2);// add1
        LOW(3);// data2=0 for TTF regs
        for (j=3; j>-1; j--){ //write data 0 thru 3
          LOW(6); //address - 18
          //data data2=0
          if ((j &  2)>>1) HIGH(7)  else LOW(7); //data1
          if (j &  1) HIGH(5)  else LOW(5); //data0
          HIGH(9); // Toggle clock to write data for ttdreg1
          WAIT(1);
          LOW(9);
          WAIT(1);
          for (i=3; i>-1; i--){ //write data 0 thru 3 for ttdreg2
            HIGH(6); //address 19
            //data
            if ((i & 2)>>1) HIGH(7)  else LOW(7);
            if (i & 1) HIGH(5)  else LOW(5);
            HIGH(9); // Toggle clock to write data for TTD reg 2
            WAIT(1);
            LOW(9);
            WAIT(waitTime); // give chip time to settle after reg data change
            for (l=0; l<numConversions; l++){
              for (k=6; k<8; k++){
                LOW(8); // start conversion
                WAIT(intTime[k]); // wait 30 ms
                conv[k] = AD(k)>>6; // save data
                HIGH(8);
                WAIT(intTime[k]);
              }
              printf("   0      0      %2d     %2d      %2d    %2d   %4d  %4d
%4d  %4d  %4d  %4d  %4d  %4d\n",
                18, j, 19, i, conv[7], conv[6], conv[5], conv[4], conv[3],
conv[2], conv[1], conv[0]);
            }
          }
        }
        resetChip();
        //printf("Complete\n\n");
        break;


      case 12000000:
        // printf(" %s",instring);
        printf("\nTest2: Conversion");
```

```
        printf("\nDo you want to initialize VGA and TTD reg values to
default?");
        printf("\nEnter y if yes:");
        gets(instring);
        // printf(" %s",instring);
        port = atoi(instring);
        if (port == 73 ) // 73 = y
        {
          printf("\n Begin Initialization.\n Set all TTD=01, all VGA=04\n");
          // Put chip in default state
          // Set all IO to outputs
          for (i=0; i<16; i++) {
            OUTPUT(i);
          }
          // Set clock low
          LOW(9);
          // Reset integrator
          HIGH(8);
          // Reset registers
          HIGH(11);
          WAIT (1); //high 1 ms
          LOW(11);
          // Program all TTD regs to 01
          LOW(3); // Data2
          LOW(7); // Data1
          HIGH(5); // Data0
          for  (i=17; i<45; i++) //use 45 to keep address out of range
            {
            if (i & 32) HIGH(10) else LOW(10);
            if (i & 16) HIGH(0)  else LOW(0);
            if (i &  8) HIGH(1)  else LOW(1);
            if (i &  4) HIGH(4)  else LOW(4);
            if (i &  2) HIGH(2)  else LOW(2);
            if (i &  1) HIGH(6)  else LOW(6);
            WAIT(1);
            HIGH(9); // Toggle clock
            WAIT(1);
            LOW(9);
          }
          // Program all VGA regs to 04
          HIGH(3); // Data2
          LOW(7); // Data1
          LOW(5); // Data0
          for  (i=0; i<16; i++) {
            LOW(10);
            LOW(0);
            if (i &  8) HIGH(1)  else LOW(1);
            if (i &  4) HIGH(4)  else LOW(4);
            if (i &  2) HIGH(2)  else LOW(2);
            if (i &  1) HIGH(6)  else LOW(6);
            WAIT(1);
            HIGH(9); // Toggle clock
            WAIT(1);
            LOW(9);
          }
        printf("\n Initialization complete.");
        } // end of if for initialize
```

```
        printf("\n Do you want to program any registers?");
        printf("\n Enter y if yes:");
        gets(instring);printf(" %s",instring);
        port = atoi(instring);
        while (port == 73 ) {
          printf("\n Enter Register (0 to 43):");
          gets(instring); //printf(" %s",instring);
          j = atoi(instring);
          if (j & 32) HIGH(10) else LOW(10);
          if (j & 16) HIGH(0)  else LOW(0);
          if (j &  8) HIGH(1)  else LOW(1);
          if (j &  4) HIGH(4)  else LOW(4);
          if (j &  2) HIGH(2)  else LOW(2);
          if (j &  1) HIGH(6)  else LOW(6);
          if ( j > 15 )
            printf("\n Enter Data (0 to 3):");
          else
            printf("\n Enter Data (0 to 7):");
          gets(instring); // printf(" %s",instring);
          k = atoi(instring);
          if (k &  4) HIGH(3)  else LOW(3);
          if (k &  2) HIGH(7)  else LOW(7);
          if (k &  1) HIGH(5)  else LOW(5);
          WAIT(1);
          HIGH(9); // Toggle clock
          WAIT(1);
          LOW(9);
          printf("\n Register %d programmed to %d.", j, k);
          printf("\n Do you want to program any registers?");
          printf("\n Enter y if yes:");
          gets(instring); // printf(" %s",instring);
          port = atoi(instring);
        }
        // Read INT output from all super pixels, 1 INT at a time
        printf("\nConversion INT1P INT1N INT2P INT2N INT3P INT3N INT4P
INT4N");
        WAIT(waitTime); // give chip time to settle after reg data change
        for (k=0; k<8; k++){sum[k]=0;}
        for (i=0; i<numConversions; i++) {
          for (k=0; k<8; k++){
            LOW(8); // start conversion
            WAIT(intTime[k]); // wait 30 ms
            conv[k] = AD(k)>>6; // save data
            sum[k]=sum[k]+conv[k];
            HIGH(8);
            WAIT(intTime[k]);
          }
          printf("\n    %2d     %4d  %4d  %4d  %4d  %4d  %4d  %4d  %4d", i+1,
conv[7], conv[6], conv[5],
              conv[4], conv[3], conv[2], conv[1], conv[0]);
        }
        printf("\n ***************************************** ");
        printf("\n           %4d  %4d  %4d  %4d  %4d  %4d  %4d  %4d",
10*sum[7]/i, 10*sum[6]/i, 10*sum[5]/i,
             10*sum[4]/i, 10*sum[3]/i, 10*sum[2]/i, 10*sum[1]/i,
10*sum[0]/i);
        printf("\n  ");
```

```
        printf("\n differences:  1: %4d  2:%4d  3:%4d  4:%4d", 10*sum[7]/i-
10*sum[6]/i,
             10*sum[5]/i-10*sum[4]/i, 10*sum[3]/i-10*sum[2]/i, 10*sum[1]/i-
10*sum[0]/i);
        printf("\nComplete");
        break;


        default:
        printf("*** unknown option - %d ***\n", inval);
        break;

    }
  }

  return 0;
}
```