Mithril - FMCW Radar

by

Tomas Esson, Ajay Thakkar, Juan Jimenez

tesson@stevens.edu, athakka5@stevens.edu, jjimene6@stevens.edu

September 7, 2024

tesson@stevens.edu, athakka5@stevens.edu, jjimene6@stevens.edu

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The following includes small biographies on all the authors as well as their research interests and projects.

Authors' Biographies

Ajay Thakkar

Ajay Thakkar Ajay Thakkar is a junior majoring in Computer Engineering. He is interested in signal processing and lower level coding. Below you can find his GitHub: https://github.com/athakkar2.

Tomas Esson

Tomas Esson is an aspiring Computer Engineering at Stevens Institute of technology. He is an avid surfer and enjoys elegant math proofs. Currently pursuing interests in computer chip design, digital systems implementation, mathematical optimization of computer chips, and electrical engineering.

Juan Jimenez

Juan Jimenez is a Junior Computer Engineering student at the Stevens Institute of technology. Interested in the intersection between Artificial Intelligence, embedded electronics, and software engineering. To see more projects visit the following GitHub link: https://github.com/jjimene1

Chapter 2

Project Description

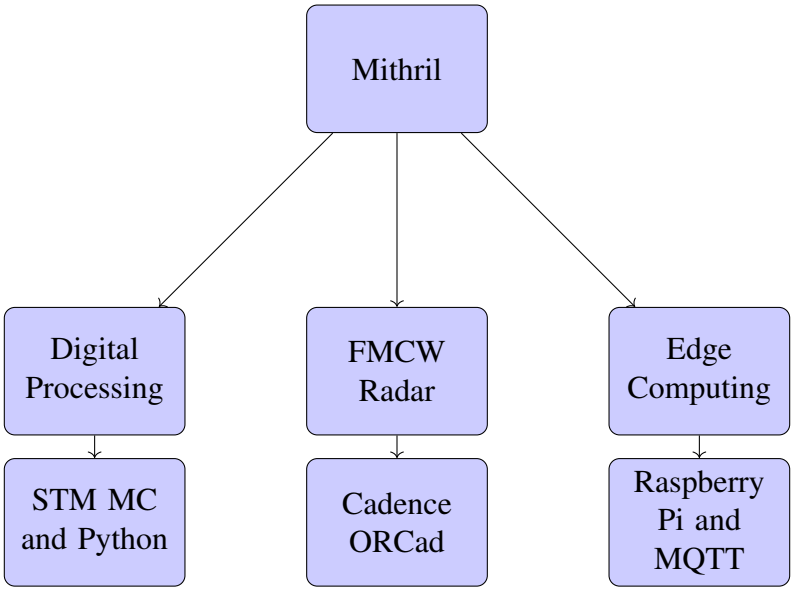

Figure 2.1: Flowchart of the Mithril system

Mithril is a nodal FMCW radar system that incorporates traditional FMCW radar, digital processing, edge computing, and distributed networking. The initial application of this project was for a military context. State of the art missile defense systems cost too much and are too big to be viable in many places, and while our system doesn't compare to systems like the Iron Dome or Patriot System it could allow for at least an alert for civilians to leave possible shelling and bombing sites. The system would include small radar nodes capable of detecting missiles and electronically steering their line of sight via phased array antennas. Multiple nodes could work in tandem to communicate their findings, allowing for a wider range of sight and active tracking of projectiles. The network would be able to function regardless of how many nodes there were, and still continue to function if nodes are destroyed.

As can be seen in Figure 2.1, the radar was designed as a standalone PCB in ORCad, digital processing was handled by STM microcontrollers, and distributed networking is done via Raspberry

Pi's and the MQTT protocol. All of these components were designed, engineered, and interfaced from scratch with a limited budget of 2000 dollars.

2.1 FMCW Radar PCB

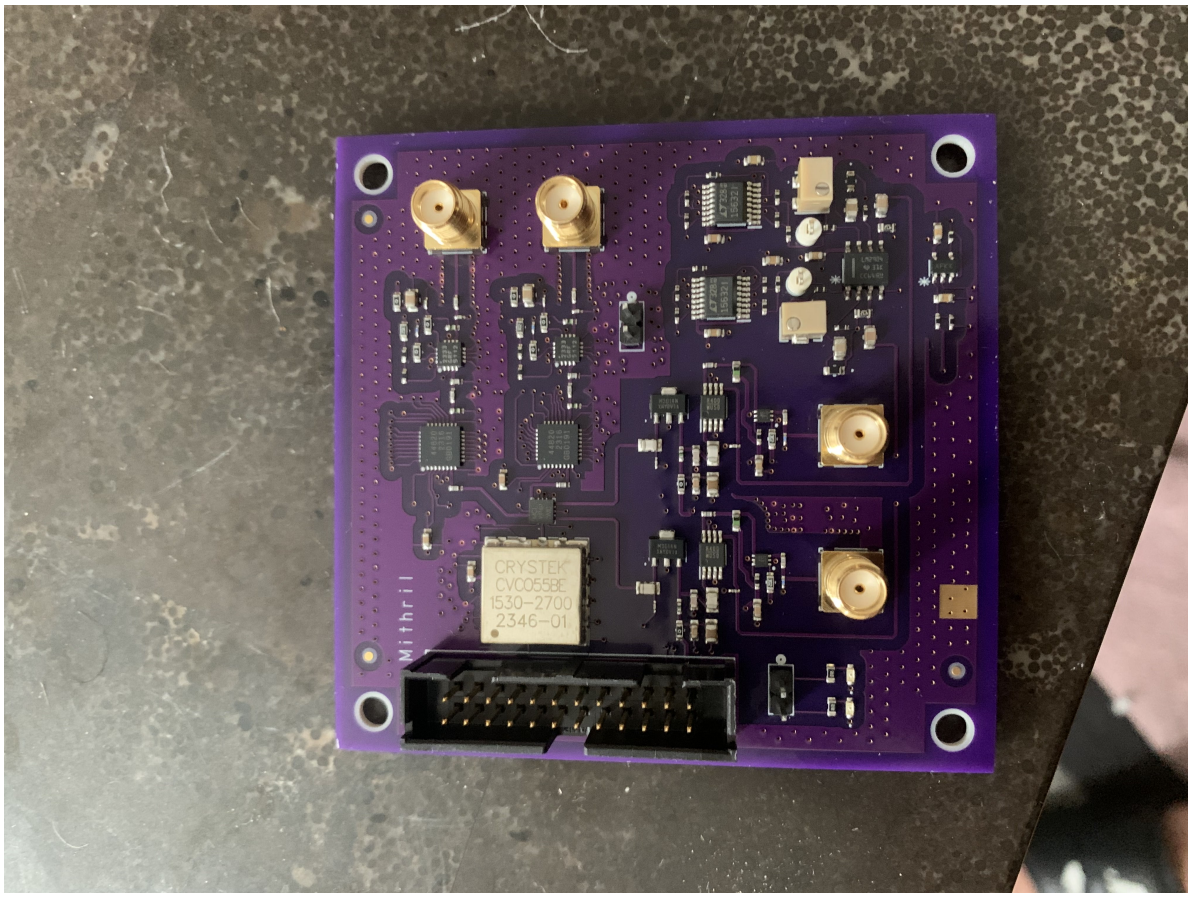

Figure 2.2: Mithril PCB

The heart of the project is a standalone PCB capable of FMCW radar and phased array beamforming, which can be seen in Image 2.2. The PCB has a transmitting and receiving portion, each with several stages. The TX portion has a stage for signal synthesis, phase shifting, and amplification. The RX portion has a stage for amplification, mixing, and baseband filtering/amplifying. There are also GPIO pins for interfacing with the microcontroller, as well as test pins for analyzing the signals we receive. This was foreign ground for the entire team, and was started by first researching common RF chains, picking parts that could handle the afformentioned stages, designing the schematic and layout in ORCad, and then printing and assembling the board. Afterwards, we tested for weeks to find different issues and workarounds which will be mentioned later.

2.2 Digital Processing

The digital processing was handled entirely by the STM-F756ZG. Two of them were used in the final implementation, where one was responsible for generating a ramp voltage which will be discussed later, and the other responsible for sampling the radar's received signal, windowing the signal, taking the FFT, transmitting the FFT over UART, and receiving commands over UART to program the phase shifter on the PCB via its GPIO pins. This was a lot of tasks, and all of it was programmed to run in a superloop with the help of DMA and interrupts.

2.3 Networking

Mithril relies on the idea of using smaller, cheaper radar PCB nodes which work together to achieve the same sort of capabiltiy as a much larger system. To achieve this PCB radar nodes must be able to communciate with one another and be able to coordinate and send data in a safe and secure manner. Furthermore, the network must be able to easily add new nodes into the system as well as be able to handle if and when nodes are destroyed or if communication becomes impossible. To achieve this we have implemeneted a mesh network topology for our defense system.

A mesh network is a type of network topology in which each node in the network can communicate with multiple other nodes in the network, creating a highly redundant and resilient network architecture. In a mesh network, each node acts as both a sender and a receiver, relaying data through the network until it reaches its intended destination. This allows for the creation of a self-configuring and self-healing network that can be quickly deployed and is resilient to failures. Figure 2.3 shows a typical mesh network configuration.

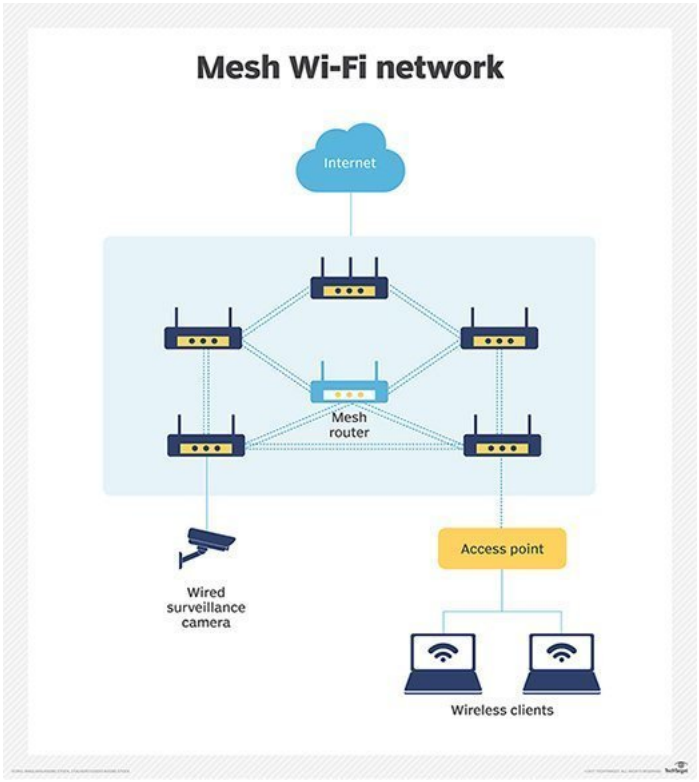

Figure 2.3: Sample Mesh Network Configuration

Mesh networks have numerous applications, and one of the most promising is for Internet of Things (IoT) devices. IoT devices typically rely on wireless communication, but traditional wireless networks may not be sufficient for large-scale deployments. An ad-hoc mesh network can provide the necessary scalability and reliability for IoT devices, allowing them to communicate with each other and with central servers without relying on a single point of failure. This makes them ideal for scenarios where traditional networks may not be available or feasible, such as in disaster response situations or in remote locations both possible locations where Mithril might be employed.

To actually implement the mesh network we will use Raspberry Pi's to handle all the software and wireless connectivity capabilities. Raspberry Pi's are small, low-cost, single-board computers which provide incredible performance despite their small size and cost. Raspberry Pi's are roughly the size of a credit card and come with various input and output ports, including HDMI, USB, Ethernet, and GPIO (General Purpose Input/Output) pins. They can run a variety of operating systems, including Linux-based distributions such as Raspbian and Ubuntu, and can be programmed in various programming languages, including Python, C, and Java. Because of their small size, low power consumption, and versatility, Raspberry Pi's are an excellent platform for the creation of low-cost and flexible ad-hoc mesh networks. Mithril will utilize a series of Raspberry Pi's along with some USB Wi-Fi dongles to implement an ad-hoc network which will trasmit radar data. Mithril achieves this this by utilizing several Linux software libraries together to create a framework for a Raspberry-Pi enabled mesh network.

With the nodes setup for wireless network capabilties Mithril needs some form of protocol to

actually communciate and send information over the mesh network. To this end, Mithril utilizes the MQTT network protocol to send the telemetry information acquired by the PCB radar nodes. MQTT is a lightweight, publish-susbcribe, machine to machine network protocol for message queue/message queuing. Nodes on the mesh network utilizing MQTT can publish their data under a certain ”topic” which other nodes on the network can susbcribe to and receive the ”messages” which are published under the topics. We utilize Eclipse Paho which is a python library which provides functionality for implementing MQTT v5.0. It provides a client class which enables applications to connect to an MQTT broker to publish messages, and to subscribe to topics and receive published messages. It also provides some helper functions to make publishing one off messages to an MQTT server very straightforward.

Chapter 3

Resources

3.1 Radar Theory

MATLAB Radar Theory
- MIT Lincoln Lab Radar Course
- FMCW Radar Design Book
- Principles of Modern Radar Book
- Analog Devices Phased Array Article

3.2 PCB

OrCAD Tutorial
- Altium RF PCB Design
- Analog Devices RF PCB Considerations
- HACKADAY Practical RF Design Tips

3.3 Henrik's Blog (Blogs on Making Radar Systems)

FMCW Radar Iteration 1
- FMCW Radar Iteration 2
- FMCW Radar Iteration 3

3.4 STM Tutorials

Phil's Lab STM FFT Tutorial
- STM ADC with DMA and Timer Tutorial

Chapter 4

Radar Theory

To preface this chapter, we are not experts by any means. Please refer to Chapter 3 for theory from real smart people. This is just for our reference and used as a means of documenting how we've understood the things we learned along the way.

4.1 What is Radar?

4.2 General Radar Theory

4.3 Types of Radar

4.4 FMCW Radar Theory

4.5 Heterodyning/Mixers

4.6 Baseband Filtering

4.7 Radar Architecture

4.8 RF PCB Considerations

include the benefits of using higher frequency signals
heterodyning

Chapter 5

Part Selection

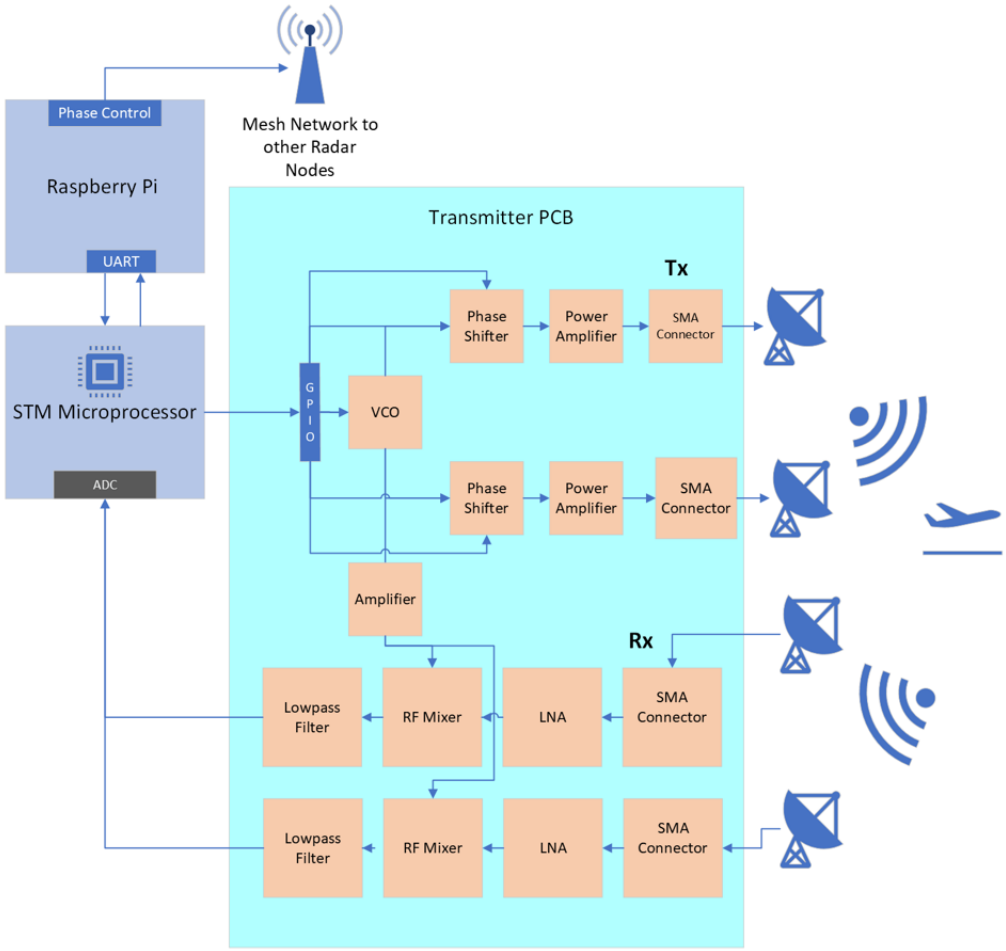

Figure 5.1: Architecture Overview

In this chapter, we will go through the architecture of the PCB and show what parts we picked and what purposes they serve. Figure 5.1 shows the overall architecture of the project.

5.1 Voltage Controlled Oscillator (VCO)

The first step in creating a radar is signal synthesis. You essentially need to create an alternating current signal that can go through your antenna and radiate out into the air. To reiterate from Chapter 4, higher frequency signals are important for a better radar resolution, and so it is desired to have a signal that is high in frequency. Naively, at the beginning of this project we thought we could use a 16 bit DAC to synthesize our signal but after finding out its max clock rate was 1 MHz, we realized a DAC was not fit for this. All we needed was something that could create a simple sinusoid at a very high frequency.

Voltage Controlled Oscillators (VCO) do just this. They take in a ”tuning voltage” which corresponds to a certain frequency of sinusoid which it will output. The circuitry for this is beyond me, but for a good overall guide on VCOs you can check out DigiKey’s article here.

Now that we knew how to synthesize our signal, it was important to find a good VCO that had a high output bandwidth. Whatever bandwidth the VCO has will impact what parts we can get in the other stages of the RF chain since these must be within the VCOs operating regions.

PERFORMANCE SPECIFICATION	MIN	TYP	MAX	UNITS
Lower Frequency:			1530	MHz
Upper Frequency:	2700			MHz
Tuning Voltage:	0.5		10.5	VDC
Supply Voltage:	4.75	5.0	5.25	VDC
Output Power:	+4.0	+7.5	+11.0	dBm
Supply Current:		15	30	mA
Harmonic Suppression (2^{nd} Harmonic):		-15		dBc
Pushing:			25.0	MHz/V
Pulling, all Phases:			35.0	MHz pk-pk
Tuning Sensitivity:		140		MHz/V
Phase Noise @ 10kHz offset:		-87		dBc/Hz
Phase Noise @ 100kHz offset:		-110		dBc/Hz
Load Impedance:		50		Ω
Input Capacitance:			50	pF
Operating Temperature Range:	-40		+75	°C
Storage Temperature Range:	-45		+90	°C

Figure 5.2: VCO Datasheet Table

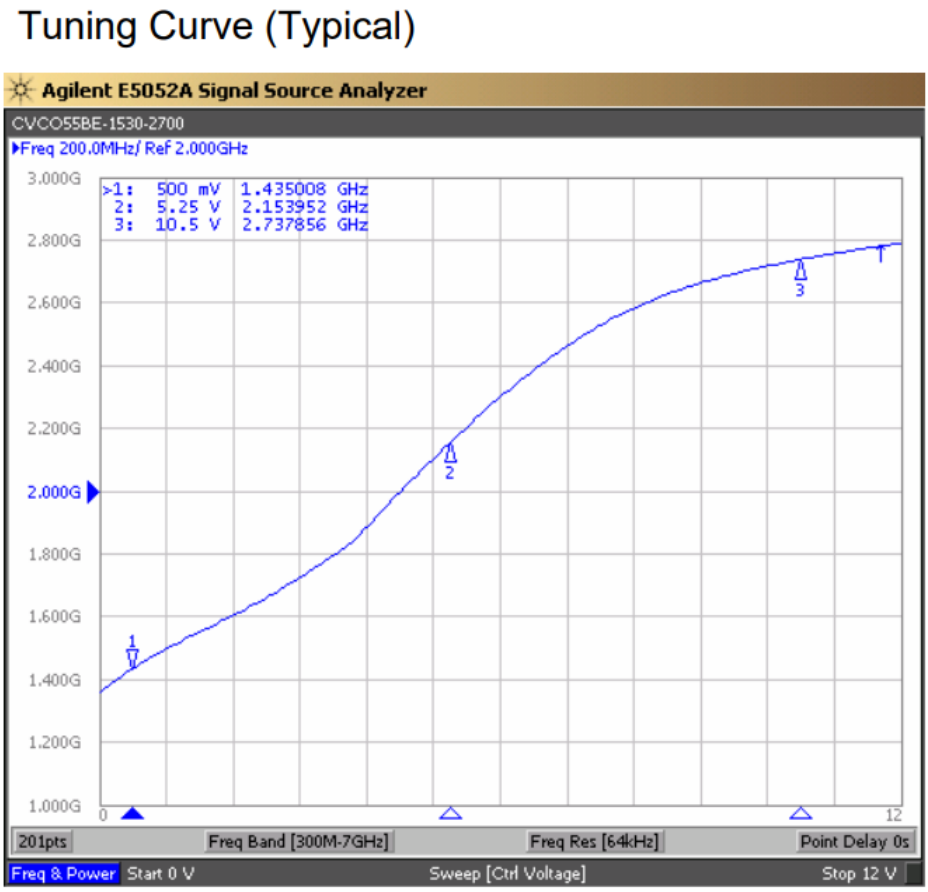

Figure 5.3: Tuning Voltage Graph

We chose a VCO from Crystek which can be found on Digikey here. Looking at the specifications table in Figure 5.2, some good things to look for are first and foremost the frequency range of the part. It goes from about 1.5-2.7 GHz, which is a pretty wide range and would support a lot of other parts as it also covers the Wi-Fi band. The second thing to look for is output power. The output power can be a constraint for other parts, since they might have a absolute limit on their input power, and it is useful to know the output power for finding out how strong your signal will be once it propogates out of your antenna. Here we can see the output power is around 7.5 dBm, and this will be split four ways. Decibels are a logarithmic scale, so we cannot just divide by four but instead use a decibel calculator like this one.

Now, another key metric is the tuning voltage. Luckily, this Crystek provides a tuning voltage graph found in Figure 5.3 that shows us how the output frequency changes with changes in the tuning voltage. One observation is that it is not linear, meaning our ramp voltage will not result in a true linear ramp in frequency. Another observation is that our chosen frequency of 1.8 GHz is around 3.5 volts following the graph. Our STM board that will generate the ramp voltage can by default only go to 3.3 volts, but we found a workaround to go to 5 volts which allowed us to stick with our decided center frequency.

5.2 Power Divider

As mentioned before, we want to divide the VCO's signal four ways because want it to go to two phase shifters, and two mixers. At first we thought it was as simple as having one wire split into four, but as we will go over in Chapter **??**, impedence matching is very important in RF circuits. To put it simply, impedance matching ensures that no power is reflected, and this is important because reflected power means distortions in the signal and power loss. That means we needed a power splitter meant for splitting an RF signal without causing reflected power. There is not much of note with the part we used which can be found here. The main thing is that it contains the frequency of 1.8 GHz we want to use.

5.3 Phase Shifter

Two of the power dividers split paths will go into phase shifters. The phase shifters are used to make our phased array of antennas as explained in Chapter 4. They are able to change the phase (add time delay), to the signal so that when they propogate through the air they can construct and destruct. The phase shifters we chose can be found here. These are digital phase shifters that have 256 different phases it can apply to the signal. They have a parallel or serial interface we can use to transmit 8 bit words That will alter the phase of our signal. The interface and its timing diagrams will be explained more in detail in Chapter 8. All we need to know is that the bandwidth that the chip supports contains 1.8 GHz.

5.4 Power Amplifier

Nominal Operating Parameters – RF (1710 to 1920 MHz)

The following conditions apply unless noted otherwise: Typical Application Schematic using the 1710 to 1920 MHz tuning set, M1 = 6.0 kΩ, V_{DD} = 5 V, I_{DDQ} = 212 mA, 50 Ω system impedance, P_{OUT} = +21 dBm, F_{TEST} = 1805 MHz, $T_{PKG\ HEAT\ SINK}$ = 25 °C. Evaluation board losses are included within the specifications.

Parameter	Symbol	Specification			Unit	Condition
		Min.	Typ.	Max.		
Small Signal Gain	S21		17.1		dB	F_{TEST} = 1805 MHz, $T_{PKG\ HEAT\ SINK}$ = 25 °C, V_{DD} = 5 V, P_{IN} = -25 dBm.
Input Return Loss	S11		< -11		dB	F_{RF} = 1710 to 1920 MHz Small Signal.
Output Return Loss	S22		< -6		dB	F_{RF} = 1710 to 1920 MHz Small Signal.
Reverse Isolation	S12		< -21		dB	F_{RF} = 1710 to 1920 MHz Small Signal.
Noise Figure	NF		1.7		dB	On standard evaluation board.
Output 3rd Order Intercept Point	OIP3		40		dBm	18 dBm P_{OUT} per Tone at 600 kHz Spacing.
Output 1 dB Compression Power	OP1dB		32.2		dBm	Sine wave input, V_{DD} = 5V, $T_{PKG\ HEAT\ SINK}$ = 25 °C.
Adjacent Channel Leakage Ratio	ACLR		-45		dBc	P_{OUT} = +20 dBm, LTE 20MHz 100RB TM1.1 Downlink Waveform with 9.6dB PAR, F_{TEST} = 1805 MHz, $T_{PKG\ HEAT\ SINK}$ = 25 °C, V_{DD} = 5 V.

Figure 5.4: Power Amplifier Table

At this point after the VCO and phase shifter, we want the RF signal to propogate through the air. However, according to the radar range equation the power of the signal when attenuating through the air attenuates at a power of four which is a lot. Therefore, we need to make sure our signal is powerful enough to go pretty far. So, we use an RF power amplifier to amplify the signal. We chose a part from GuerillaRF which can be found here. Looking at the table in Figure 5.4, we can see some key metrics when looking at power amplifiers. First, of course we want to make sure it has a bandwidth that supports our chosen frequency of 1.8 GHz. Second, we want to look at the small-signal gain and Output 1 dB Compression Point or OP1dB. The small-signal gain is the ideal gain that can be reached with a low power signal, and is listed as 17.1 dB. The OP1dB is a metric we did not know about and were thankful to find it. With most amplifiers, gain operates linearly meaning whatever power your input signal is you just add the gain of the amplifier and this will be the resulting power of the signal. However, at a certain point the amplifier saturates and does not operate linearly anymore, and will essentially cap-off its gain at the OP1dB limit. For example, with the OP1dB being 32.2 dB, if I input a signal that was 30 dB I would expect a resulting signal of 47.1 dB but this would not be the case. The amplifier ceases to operate linearly after the 32.2 dB mark, and will both distort the signal and output something weaker than expected. A lot of amplifiers will boast a high gain but have a low OP1dB, so this is definitely something to check for.

5.5 Low Noise Amplifier

Nominal Operating Parameters - RF

The following conditions apply unless noted otherwise: typical measurement schematic using the 0.1 to 2.7 GHz tuning set, V_{DD} = 5 V, V_{ENABLE} = 5 V, I_{DDQ} = 60 mA, F_{TEST} = 1.95 GHz, 50 Ω system impedance, $T_{PKG\ BASE}$ = 25 °C. Evaluation board losses are included within the specifications.

Parameter	Symbol	Specification			Unit	Condition
		Min.	Typ.	Max.		
Gain	S21	26.5	28		dB	
Noise Figure	NF		0.6	0.8	dB	On standard evaluation board.
Output 3rd Order Intercept Point	OIP3		31		dBm	+2 dBm P_{OUT} per tone at 2 MHz spacing (1949 and 1951 MHz)
Output 1 dB Compression Power	OP1dB	18	20		dBm	

Figure 5.5: Low-Noise Amplifier Table

This is the first part that will be placed in the receiver RF chain. According to the Friis equation in Chapter 4, the first stage in the receiver RF chain matters a lot for the noise figure of your system. Therefore, we wanted to find a part with a low noise figure and high gain, as this will impact our SNR the most. Low noise amplifiers are made for this exact purpose, where they amplify a small signal with very low noise. The part we chose was this, which is made by GuerillaRF. By examining the table in Figure 5.5, we can see that it has a gain of 28 dB, and a noise figure of .6 dB. We also can look at the OP1dB which has a figure of 20 dB. Since the LNA will be amplifying a signal straight from an antenna, the signal will be super weak and there is a low likelihood it will reach the OP1dB.

5.6 Mixer/LO Amp

Electrical Specifications, T_A = +25°C, LO = +17 dBm, IF = 200 MHz*

Parameter	Min.	Typ.	Max.	Min.	Typ.	Max.	Min.	Typ.	Max.	Units
Frequency Range, RF	1.7 - 1.8			1.8 - 2.0			2.0 - 2.2			GHz
Frequency Range, LO	1.4 - 1.75			1.5 - 1.95			1.7 - 2.15			GHz
Frequency Range, IF	DC - 300			DC - 300			DC - 300			MHz
Conversion Loss		9	11		8.8	10.5		8.8	10.5	dB
Noise Figure (SSB)		9	11		8.8	10.5		8.8	10.5	dB
LO to RF Isolation	29	33		24	30		20	25		dB
LO to IF Isolation	16	20		17	22		19	25		dB
IP3 (Input)	30	34		32	36		28	32		dBm
1 dB Gain Compression (Input)	18	21		18	21		18	22		dBm
LO Input Drive Level (Typical)	+16 to +18			+16 to +18			+16 to +18			dBm

*Unless otherwise noted, all measurements performed as a downconverter, with low side LO & IF = 200 MHz.

Figure 5.6: Mixer Table

Harmonics of LO

	nLO Spur @ RF Port			
LO Freq (GHz)	1	2	3	4
1.4	42	26	56	46
1.55	33	25	56	53
1.7	29	29	49	50
1.85	26	31	44	53
2	24	36	44	48
2.15	21	38	43	49

LO = +17 dBm
All values are in dBc below input LO level @ RF port.

Figure 5.7: Mixer Harmonics Table

Now that our received signal is amplified we must down-convert it in order to sample and process it. The process for doing this is called heterodyning or mixing, and the theory behind this can be found in detail in Chapter 4. A mixer has three ports, the local oscillator, RF signal, and output. The local oscillator and RF signal are multiplied to produce the sum and difference of the LO and RF signals on the output port. We are mainly interested in the difference, also called the Intermediate Frequency (IF) since it is a low frequency and can be sampled easily. In our case, we take a copy of the VCO as the local oscillator and then mix this with the amplified return signal to produce our intermediate frequency. To achieve this, we used a discontinued mixer from Analog Devices which can be found here. Looking at the mixer's specifications table in Figure 5.6, we can see some new properties. The conversion loss is the output IF power delivered minus the available RF input signal power. The LO to RF Isolation is how much of the local oscillator signal leaks into

the RF port, and the LO to IF Isolation is how much the local oscillator signal leaks into the output port. This is a passive component, meaning it does not require power and solely operates off the power of the LO and RF signals. Therefore we see in the table that the LO Input must be 16-18 dBm to drive the mixer. Essentially the local oscillator powers the mixer, and its harmonics will therefore be prominent in the IF port due to leakage. We can see in Figure 5.7 that at 1.85 GHz the manufacturer tells us the strength of LO harmonics up to the fourth order. At the top it says spur which means spurious (unwanted) outputs due to the nonlinearity of the mixer. Essentially this table tells us that there will be unwanted spectral components in the output, which is something we did not pay enough attention to and will discuss in Chapter 11.

Table 1. Typical Performance (1)

Characteristic	Symbol	900 MHz	2140 MHz	3500 MHz	Unit
Small-Signal Gain (S21)	G_p	19.5	15	10	dB
Input Return Loss (S11)	IRL	-25	-12	-8	dB
Output Return Loss (S22)	ORL	-11	-13	-19	dB
Power Output @1dB Compression	P1dB	25	25.8	25	dBm
Third Order Output Intercept Point	OIP3	40.5	40.5	40	dBm

1. V_{CC} = 5 Vdc, T_A = 25°C, 50 ohm system, application circuit tuned for specified frequency.

Figure 5.8: LO Amp Table

As we said before, the mixer is a passive component which is driven by the LO which needs a power level of 16-18 dBm. After splitting our VCO's output four ways, we have about a 2 dBm signal that we must amplify. So, we chose an RF broadband amplifier from NXP which can be found here. As we can see in Figure 5.8, the amp has a gain of around 15 dB for our frequency and an OP1dB of 25.8 dB which is a lot of headroom.

5.7 IF Amplifiers and Filters

Now, our return signal has been downconverted, but the power of that signal is very weak. As well as this, there are unwanted spectral components in that signal that are bi-products of the mixer that we need to get rid of. This means we need amplifiers and filters to make our signal ready to be sampled and processed in the microcontroller. First we used a super simple op-amp that served as a voltage follower which can be found here. This was used to create a virtual ground for our biasing. Then we use a dual channel op-amp from TI for our variable gain amplifier which can be found here.

OUTPUT							
V_O	Voltage output swing from rail	Positive rail (V+)		I_{OUT} = 50 µA	1.35	1.42	V
				I_{OUT} = 1 mA	1.4	1.48	V
				I_{OUT} = 5 mA[1]	1.5	1.61	V
		Negative rail (V–)		I_{OUT} = 50 µA	100	150	mV
				I_{OUT} = 1 mA	0.75	1	V
			V_S = 5 V, RL ≤ 10 kΩ connected to (V–)	T_A = –40°C to +85°C	5	20	mV

Figure 5.9: Output Swing Characteristics

Honestly, when selecting parts we figured all operational amplifiers just acted the same way. However, after printing and manufacturing we ran into problems which will be discussed in Chapter 11 that could have been remedied if we paid more attention to the Op-Amps we were using. The main problem is that these operational amplifiers are not "Rail-to-Rail". While rail-to-rail amplifiers guarantee that they do not saturate until the positive and negative voltages you've supplied it, other amplifiers have what is called voltage swing. As we can see in Figure 5.9, the range of voltage that the operational amplifier can achieve is not its rails, but specifies how much lower than its rail it will be. This is something to look out for when selecting an Op-Amp.

- **Extremely Easy to Use—A Single Resistor Value Sets the Cutoff Frequency (256Hz < f_C < 256kHz)**
- **Extremely Flexible—Different Resistor Values Allow Arbitrary Transfer Functions with or without Gain (256Hz < f_C < 256kHz)**
- Supports Cutoff Frequencies Up to 360kHz Using FilterCAD™
- LTC1563-2: Unity-Gain Butterworth Response Uses a Single Resistor Value, Different Resistor Values Allow Other Responses with or without Gain
- LTC1563-3: Unity-Gain Bessel Response Uses a Single Resistor Value, Different Resistor Values Allow Other Responses with or without Gain
- Rail-to-Rail Input and Output Voltages
- Operates from a Single 3V (2.7V Min) to ±5V Supply
- Low Noise: 36µV_{RMS} for f_C = 25.6kHz, 60µV_{RMS} for f_C = 256kHz
- f_C Accuracy < ±2% (Typ)
- DC Offset < 1mV
- Cascadable to Form 8th Order Lowpass Filters
- Available in Narrow SSOP-16 Package

Figure 5.10: Filter Features

After amplifying the signal, we wanted to filter out unwanted spectral components by using a strong filter. Instead of designing this ourselves, we decided to use a 4th Order LPF which can be found here. By looking at Figure 5.10, we can see that this is an integrated circuit with a flexible cutoff frequency. It also says that it is a rail-to-rail input and output circuit, which is an important

feature to note as discussed with the amplifiers. A fourth order filter just means that this filter has four filters with the same cutoff frequency cascaded upon each other, which yields a steeper frequency response and attenuates are unwanted signals further.

Chapter 6

Schematic

6.1 ORCad Capture Overview

ORCad comes with a schematic software called Capture CIS which we used to create our schematics. As a reference to both us and others, this section is dedicated to providing a small walkthrough to using the software and some nice-to-know features.

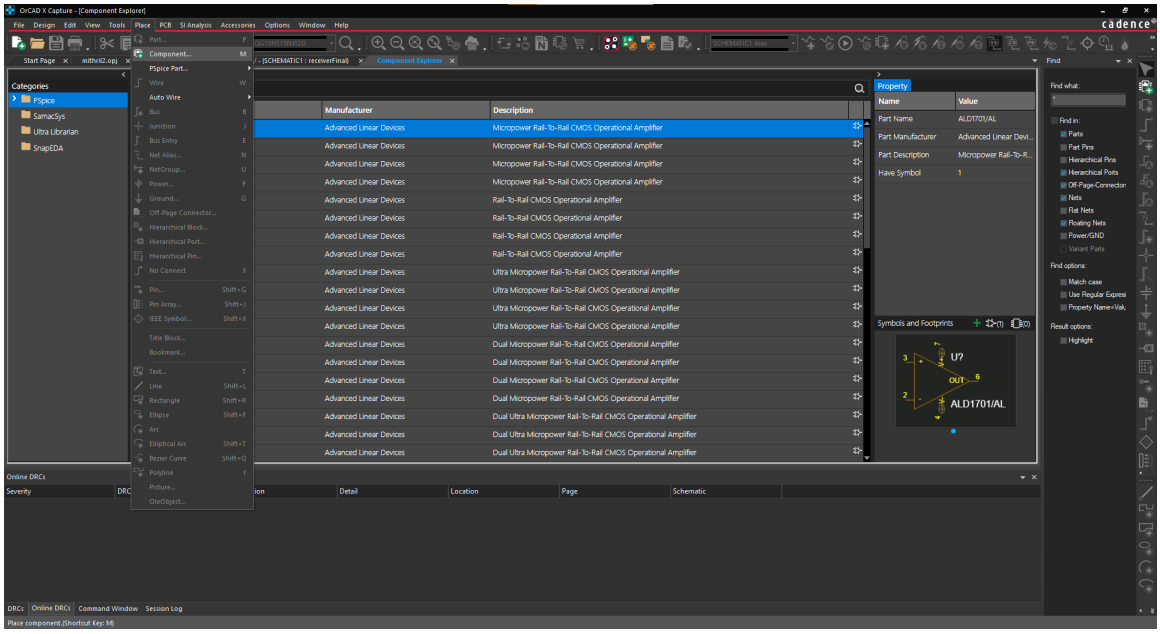

Figure 6.1: Component Database Search

The first important feature we found was the component database, which can be seen in Figure 6.1. By clicking on the icon that has a small chip and cloud will take you to this page which contains subdirectories ”PSpice” ”SamacSys” ”UltraLibrarian” and ”SnapEDA”. PSpice contains parts that have simulation properties, but we ignored this as we did not know how to use PSpice. The other three are different databases containing schematic symbols and layout footprints for parts that can be found on major retailers like DigiKey, Mouser, and Arrow.

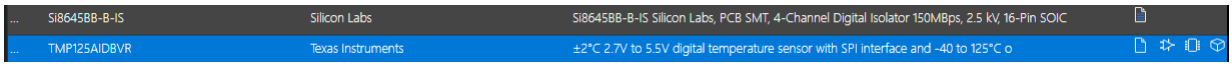

Figure 6.2: Schematic/Footprint Examples

You can search for parts in the search fields for the different databases to try to find a part that has a schematic symbol and footprint available. You can see in Figure 6.2 that parts with the symbol and footprint available will have an amp and chip symbol next to them. The box means there is a 3D CAD model associated with them too. If you cannot find the symbol and footprint for a part, we recommend going on Mouser and finding the part, then requesting the symbol and footprint. Usually, the schematic and footprint will be added to SamacSys after a couple of days.

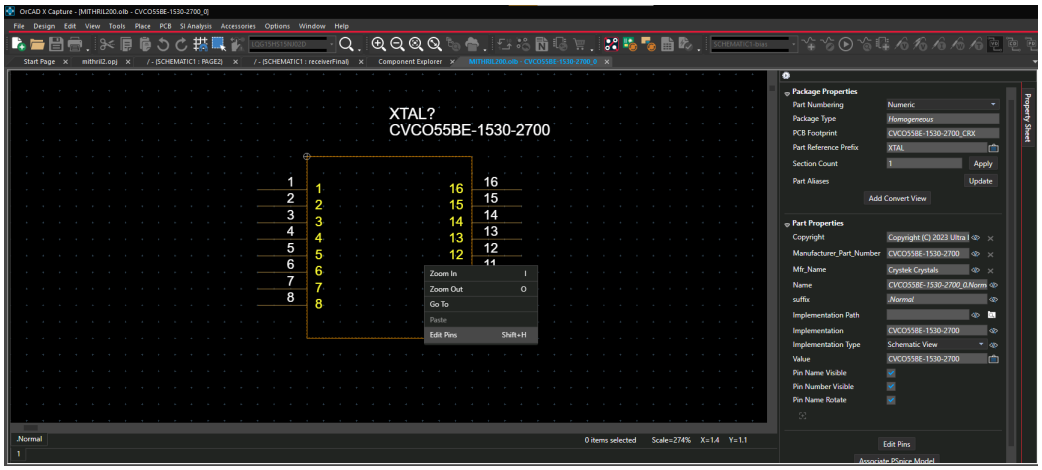

Figure 6.3: Editing Schematic Parts

Sometimes, these symbols and footprints are not correct. If the schematic symbol is incorrectly labeled, you can edit the part, then click edit pins to make sure all the pins are correctly labeled and numbered as can be seen in Figure 6.3.

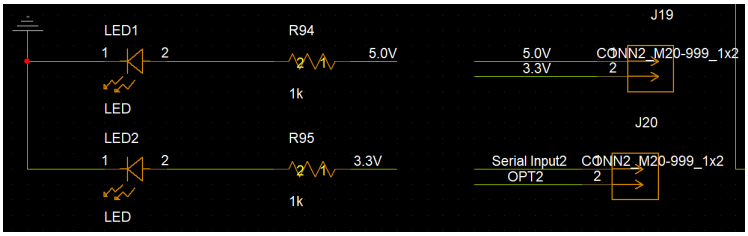

Figure 6.4: Wires and Net Alias

Once you've placed your parts down in the schematic, now comes time to connect everything together. To navigate through the schematic interface, you can use CTRL+Scroll Wheel to zoom in and out, and middle mouse button to scroll left to right.

You can use "w" to enter wiring mode, which lets you place down wires according to your grid size. After wiring, be sure to use "n" to enter net alias mode and assign aliases to your wires.

As can be seen in Figure 6.4, there are two non-connected wires both with the alias ”5.0V”. This effectively connects them since they are under the same alias and will also label them in the layout when routing. Using the net alias helps with things like power where connecting everything that needs power with a wire to your power source would make the schematic a mess. In short, all wires with the same net alias are considered connected.

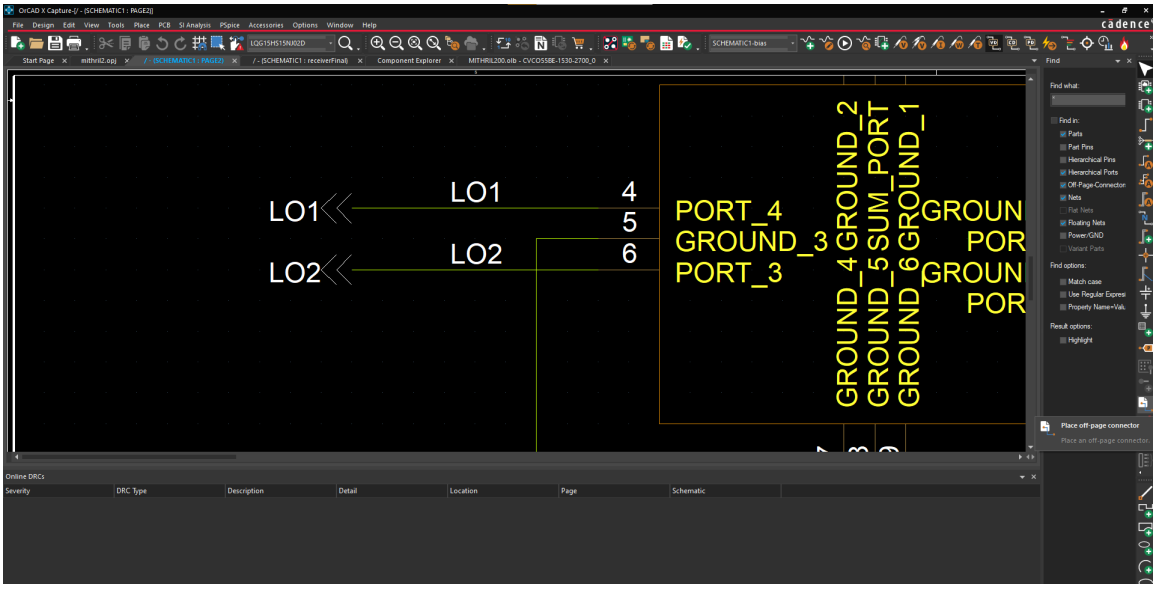

Figure 6.5: Off Page Connectors

If you run out of room or want to separate your schematics into different pages, you can connect wires from different schematic pages using off-page connectors as shown in Figure 6.5. Connect the off-page connector to a wire and name it the same on both schematic pages to have the wires connect.

6.2 Circuitry Good Practices

6.2.1 DC Blocking and Coupling Caps

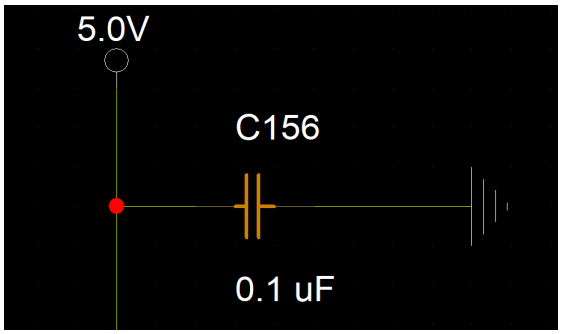

Figure 6.6: Coupling Capacitor

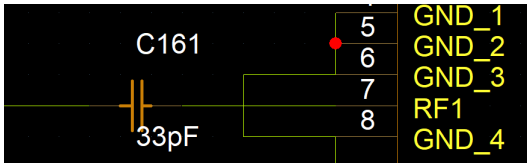

Figure 6.7: DC Blocking Capacitor

Two good practices to include in your schematics are coupling and DC blocking capacitors.

Coupling capacitors are usually used when giving power to some component or chip as can be seen in Figure 6.6. It can be thought of as a mini low-pass filter, where the capacitance of your coupling capacitor will affect the cutoff frequency of the filter. These coupling capacitors should be placed in parallel with the power wire that goes into the chip.

DC Blocking capacitors do the exact opposite, and are usually used when passing a signal from one component to another as can be seen in Figure 6.7. They can be thought of as mini high-pass filters, where the capacitance of the DC Blocking capacitor affects the cutoff frequency. These are placed in series with the signal wire.

$$C = \frac{1}{2\pi X f} \tag{6.1}$$

Figure 6.8: Capacitance of a Coupling or DC Blocking Capacitor

The equation for finding the capacitance of either the coupling or DC blocking capacitor can be seen in Equation 6.8 where C is the capacitance, X is the impedance, and f is the cutoff frequency. Essentially, a higher cutoff frequency corresponds to a lower capacitance value, and the only difference between the coupling and DC blocking capacitor is the coupling capacitor passes everything below that cutoff frequency, and the DC blocking capacitor passes everything above that cutoff frequency.

6.2.2 Test Pins and Grounds

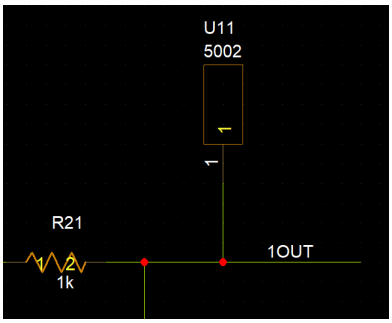

Figure 6.9: Test Pin

Other extremely important things to add are test pins as seen in Figure 6.9 and exposures to ground. Credit to Professor Muresan for telling us to add both of these as they were vital to testing and the operation of the board.

Test pins should be placed in parallel with signal wires so that you can hook in with an oscilloscope and examine what is happening. Really, after every stage in a signal chain it would be good practice to expose the signal via a test pin or if it is RF with an SMA connector. This allows you to debug analog things and find out how each stage is affecting your signal. We only added it in one

place and after the fact wished we had placed more test pins so we could see what was happening from component to component.

Ground pins and pads are also really important. We overlooked adding a ground pin which was extremely inconvenient for us, and wished we had put ample ground pins and pads everywhere on the board. These are necessary if using external components with the board, since they should have a common ground.

6.2.3 Power LEDs

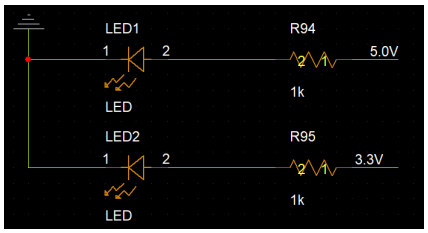

Figure 6.10: Power LEDs

Another nice thing to have are some LEDs connected to power, just so you know at the very least your board is turned on and everything is receiving power. Having a switch of some sort to block power is also something nice to have which we did not add.

6.2.4 Potentiometers

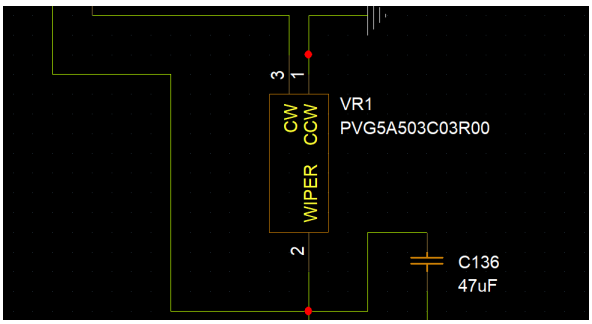

Figure 6.11: Potentiometers

Another extremely useful thing to add is potentiometers, as can be seen in Figure 6.11. In things like amplifying or filtering stages where resistors can change the cutoff frequency or gain of your circuit, potentiometers are very useful. From Figure 6.11, we can see that potentiometers have three ports: CCW, CW, and Wiper. How potentiometers work is that the wiper controls the resistance of the potentiometer, and turning it CW will make more signal go in that port, and turning it CCW will make more signal go into that port (oversimplified explanation). Therefore, it can be set up by just feeding the input to the wiper, the output to CCW, and CW to ground. This can help with changing gains and cutoff frequencies even after the board is printed and assembled.

6.3 Transmitter

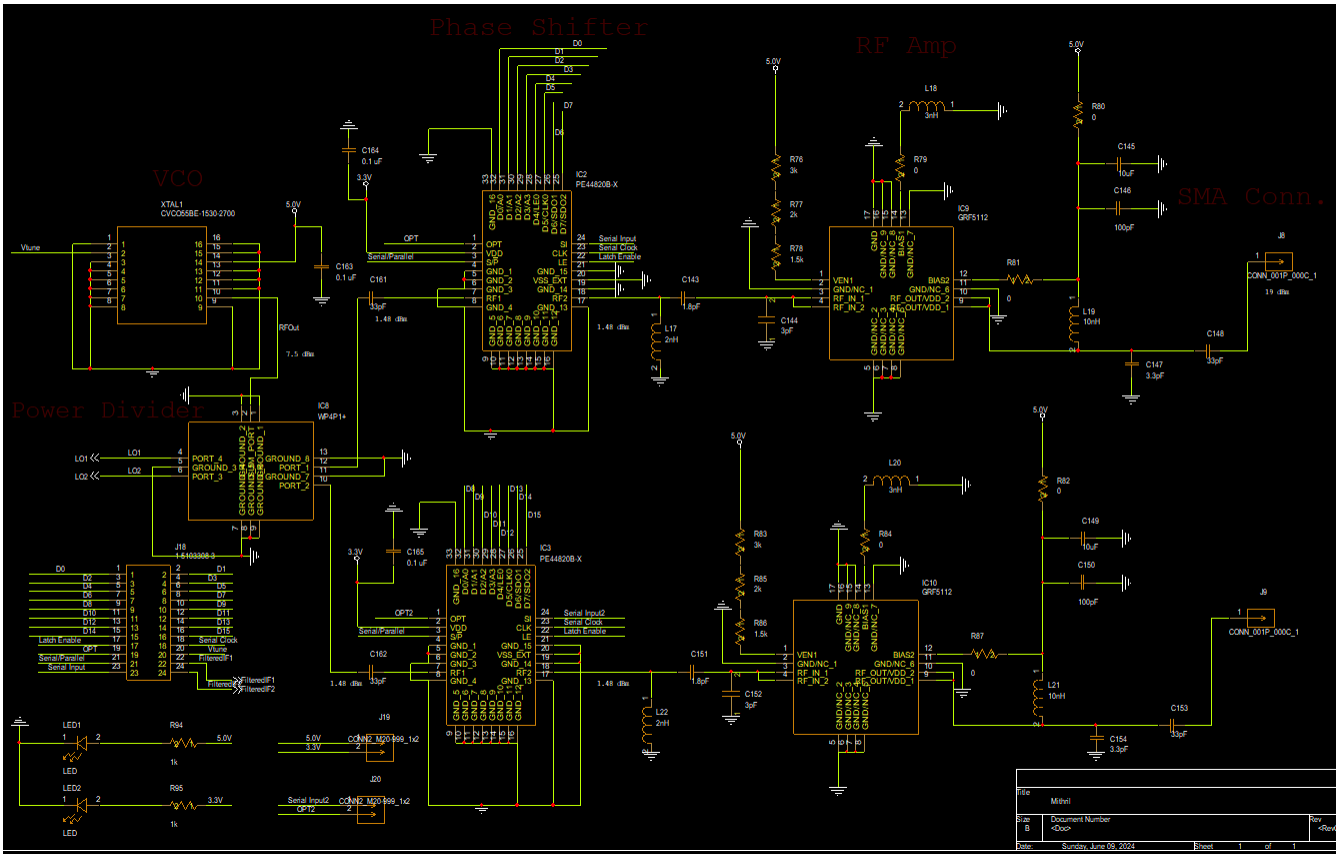

Figure 6.12: Transmitter Schematic

The first step in creating a PCB is fleshing out the schematic. We already decided on what parts to use in Chapter 5, and now was the time to create a circuit which interfaces the parts together to make a fully functioning system. As can be seen in Figure 6.12, the stages of the RF chain are labeled reflecting what was discussed in the Part Selection chapter. Essentially all of the parts shown here are connected to resistors, capacitors, and inductors according to the example schematics in their respective datasheets. In the bottom left of the schematic we can see some LEDs that are connected to power just to make sure the board is being powered and our GPIO connector which will be used to control our parts.

The signal path starts with synthesis at the VCO. The VCO connects to 5 volts and is controlled by a tuning voltage will determine the frequency at the output port of the VCO. This Vtune wire is connected to the GPIO header.

This output signal is then connected to a power divider which splits it four ways. Two of the split signals will go to the next stage in the RF chain, and the other two are used as local oscillators in the receiver chain. This can be seen by two ports of the power divider being connected to off page connectors which can be accessed by the receiver in another schematic page.

The next stage of the RF chain in the transmitter portion are the phase shifters. The phase shifters will change the phase of the two identical signals which essentially means a time delay.

This causes construction and destruction when the two signals propogate through the air, and allows for concentrated energy in one direction. We chose 8 bit digital phase shifters that have both a serial and parallel interface. The part is connected to 3.3 volts, and has a lot of control and data pins that we connected to our GPIO header to allow for control by our microcontroller. The phase shifter and its operation will be discussed more in Chapter 8.

Finally, the signal passes through our power amplifier with a small signal gain of 17.1 dBm which should help it propogate through the air for a longer period of time without dissipating. The amplified signal will then propogate through two transmitting antennas (our small phased array of antennas) to construct and destruct forming a beam of EMF.

6.4 Receiver

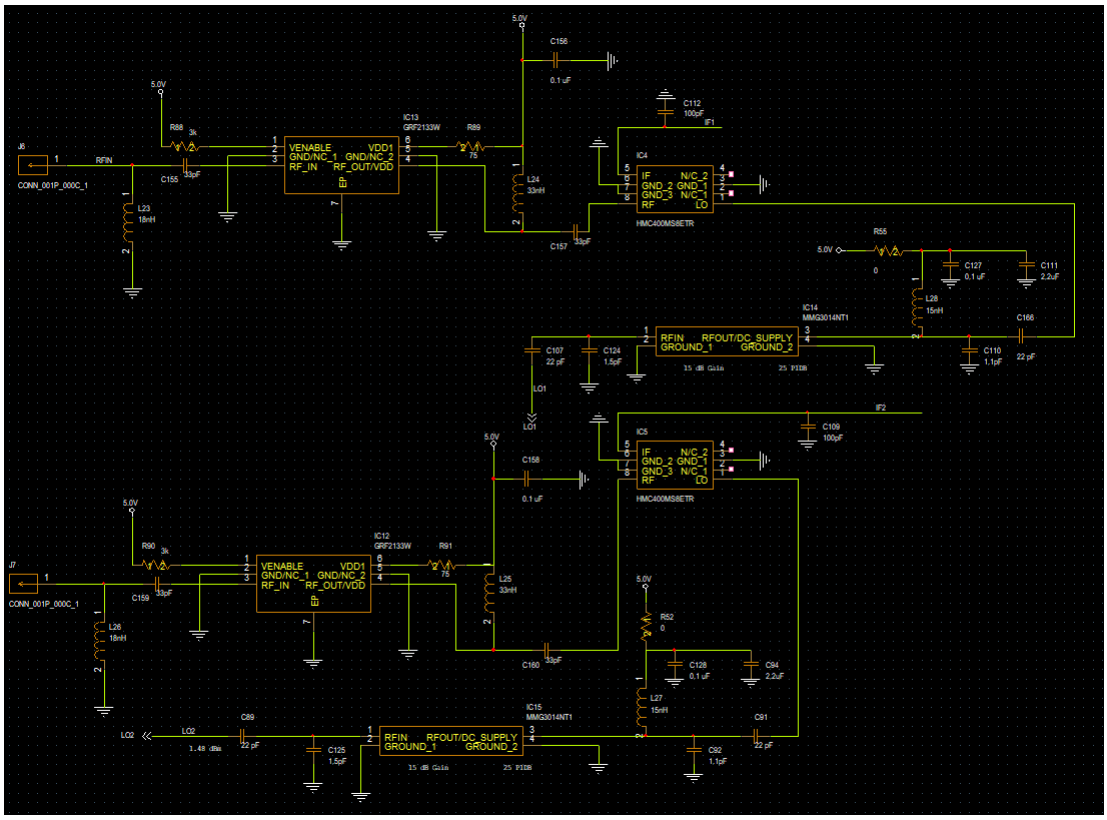

Figure 6.13: RF Portion of Receiver Schematic

After our signal propogates through the air, it should hit some objects and reflect back to the physical system, and the goal is to capture this signal and make it digestible to analyze. The first step is capturing the signal, which is accomplished through the RF portion of the receiving schematic shown in Figure 6.13.

First, we have two receiving antennas where the reflected signal should induce a signal in the antennas. We have two receiving antennas to distinguish the angle of arrival as discussed in 4.

The very small signal induced in the antennas will then pass through a low-noise amplifier which is supposed to amplify our small signal to be used in the later stages of the chain. The reason a low-noise amplifier is used in a receiving RF chain is because the first stage of a receiver is the most impactful on its signal-to-noise ratio, or how strong the signal has to be to be distinguishable from noise. This is discussed more in depth in Chapter 4. This part is powered by 5 volts, and also has an enable pin that can set the gain of the LNA and disable it if need be. There is also a pin called ”EP” which stands for exposed pad. it is essentially a big ground pad that is used as a heat sink, distributing heat to the ground plane.

After being amplified, the signal is fed into a mixer. The mixer is supposed to output the sum and difference of two RF signals. One of these is our amplified return signal, the other is a copy of out transmitted signal (LO). This transmitted signal comes from an off-page connector which is connected to the power divider in the transmitter schematic, and is then amplified by a power amplifier according to the mixer's LO specifications. The reason it is amplified is because the mixer is a passive component, meaning it does not use an external power source for its functionality. The mixer feeds these signals into the RF and LO ports respectively, and outputs the sum and difference of these signals out of the IF or intermediate frequency port.

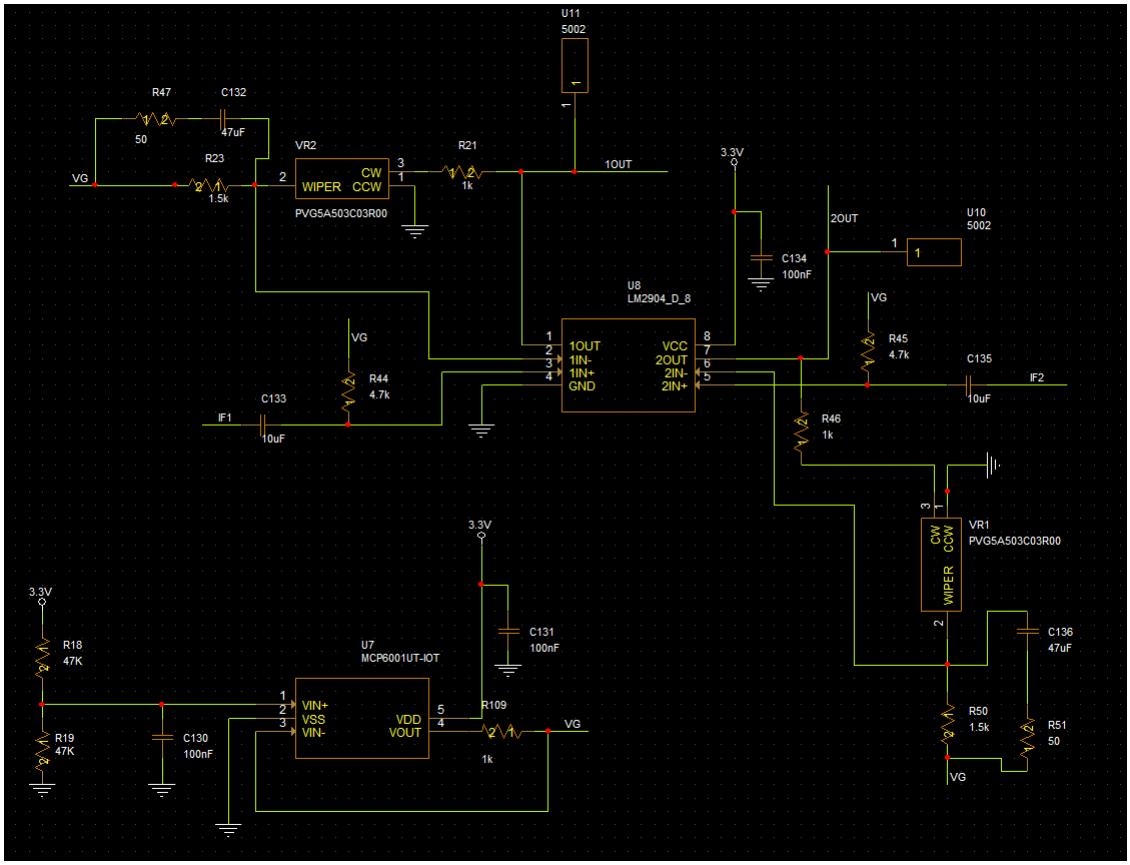

Figure 6.14: IF Amplifier Stage

After the sum and difference of our VCO copy (Local Oscillator) and return signal are outputted from the mixer, we need to do some post-processing to make sense of it. The first stage of the

intermediate frequency processing is in our case amplification (which turned out to be a bad idea as can be seen in Chapter 11). As mentioned in 5, our mixers output has a reference of 0v or ground. This means that it contains positive and negative voltage as part of its signal. However, we decided to use single supply amplifiers. This means its theoretical rails are 0v (ground) to Vcc (3.3v in this case). Therefore, we need to bias our mixer output around the middle of our rails to avoid clipping and to take advantage of the full bandwidth of the amp.

The bottom left of Figure 6.14 shows our biasing circuit, which is a simple voltage division using equal resistors and a voltage follower to keep the voltage steady. This becomes our new ”virtual ground”, or reference voltage when dealing with the amplifier. Looking at the dual amp in the center of Figure 6.14, we can see that the mixer output (labeled IF1 and IF2) is biased with our DC voltage from the biasing circuit, effectively making the new reference 3.3/2 volts.

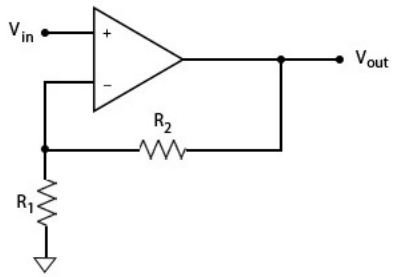

Figure 6.15: Non-Inverting Amplifier Circuit

The amplifier setup itself was taken from a Finnish engineer named Henrik Forsten whose blog we followed closely and can be found here. Essentially, it follows a basic non-inverting amplifier circuit which can be seen in Figure 6.15. The output of the amplifier feeds to a potentiometer (R2), and shares a node with resistance tied to ground (R1) and the negative input of the amplifier. The potentiometer makes this a variable-gain amplifier, since we can change R2 to different resistance values to fine tune the amplification based on how small or large the mixer’s output is. The ground for the power supply in the amplifier is connected to the power supply, but we simply replaced everywhere there would have been ground in the rest of the circuit with our ”virtual ground” from the biasing circuit to make the reference 3.3/2 volts.

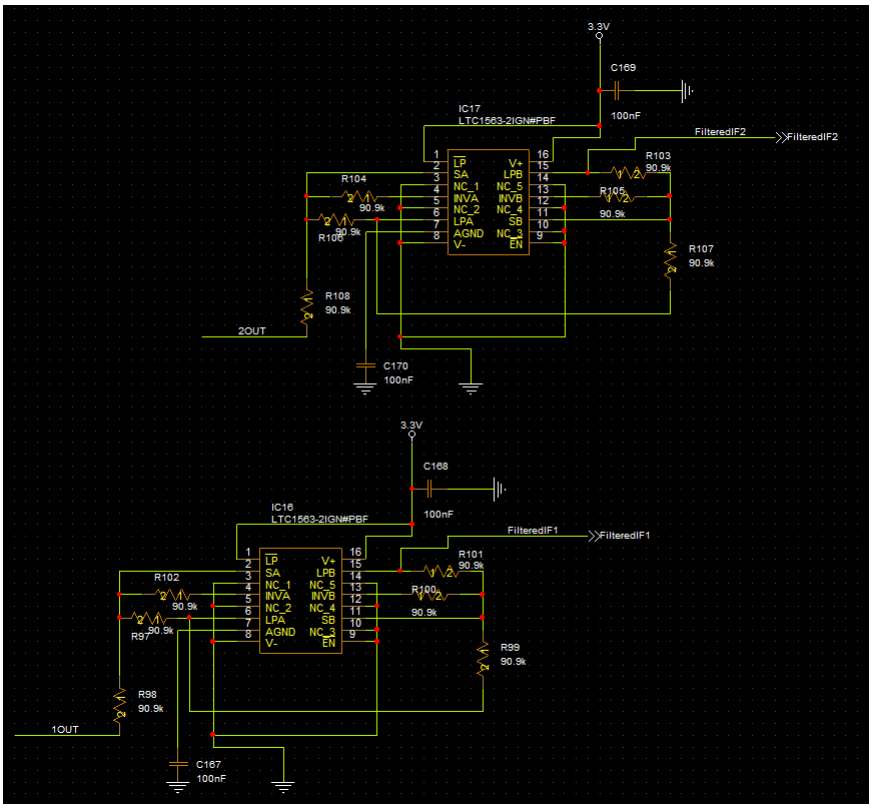

Figure 6.16: Low Pass Filters

At this point, our intermediate frequency signal is amplified, but might still have noise from the antenna or high frequency components from the mixer. In order to remove these unwanted frequencies from our signal, we want to apply some filtering to only leave the intermediate frequency of interest. We added fourth-order butterworth LPFs after the amplifiers to attenuate these other signals. As we can see in 6.16, the chip takes multiple resistance values that determine its cutoff frequency.

$$R = 10k\left(\frac{256kHz}{f_C}\right); \quad f_C = \text{Cutoff Frequency} \tag{6.2}$$

Figure 6.17: Cutoff Frequency Equation for LPF

The resistances can be determined by Equation 6.17, and in our case we used 90.9k resistors to achieve a cutoff frequency of 28 kHz. This filtered signal then goes to a GPIO pin which will be sampled by a microcontroller.

Chapter 7

Layout

7.1 Intro to PCBs

Before getting into the meat and potatoes of how to design a PCB, I think it is worthwhile to go over the terms and basics surrounding PCBs. When starting this project I didn't know anything, and probably still don't, however it helps to know the terms and have context when watching tutorials and reading documentation, etc.

7.1.1 PCB Layers

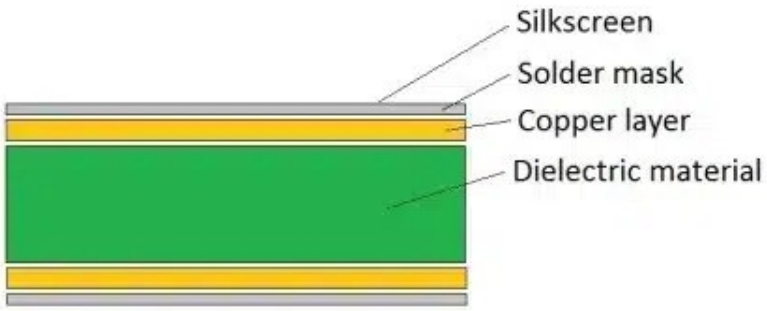

Figure 7.1: PCB Layers

A Printed Circuit Board (PCB) is made up of different materials all stacked up on top each of other, where each of these materials is called a layer of the PCB. You will also hear the term "stackup" used to describe the amount and kind of layers in one PCB. There are essentially only four types of layers you need to know, as shown in Figure 7.1. The first and most important one is the copper layer. Copper is a conductor, and so this is the layer in which we place copper where we want to conduct electricity. Whether its for sending power or signals from one place to another, or placing copper where we want to solder our component pads, this is the place where the actual circuitry is outlined. You can notice however there are two copper layers in Figure 7.1, both of which can have different signals passing through them. In order to separate the two conductors to prevent shorts,

we use a substrate layer, also referred to as a dielectric material. The dielectric layer goes between copper layers, with one purpose being electrical insulation to keep copper layers from shorting or inducing current in one another. While it does not conduct electricity, it has a controlled impedance meaning signals won't have weird reactions in different parts of the board and behavior of signals can be predicted. Different dielectric materials have different dielectric constants, essentially the number you can use to determine the constant impedance across the board. Now that we have a dielectric and copper layer, we can place a bunch of copper layers separated by dielectric layers to fit a lot of circuitry in a small space. The third layer we need is the soldermask. The top and bottom copper layers are exposed to the elements, while all copper layers in the middle are safe within the board. These top and bottom layers have exposed copper that are subject to things like oxidation, solder shorts between copper areas, and reduces affect of things like moisture. So, the soldermask is simply a film that goes on top of the top and bottom copper layers to protect it. Finally, the silkscreen layer is just engraved writing on the soldermask to help humans when assembling the components onto the board.

7.2 Traces and Vias

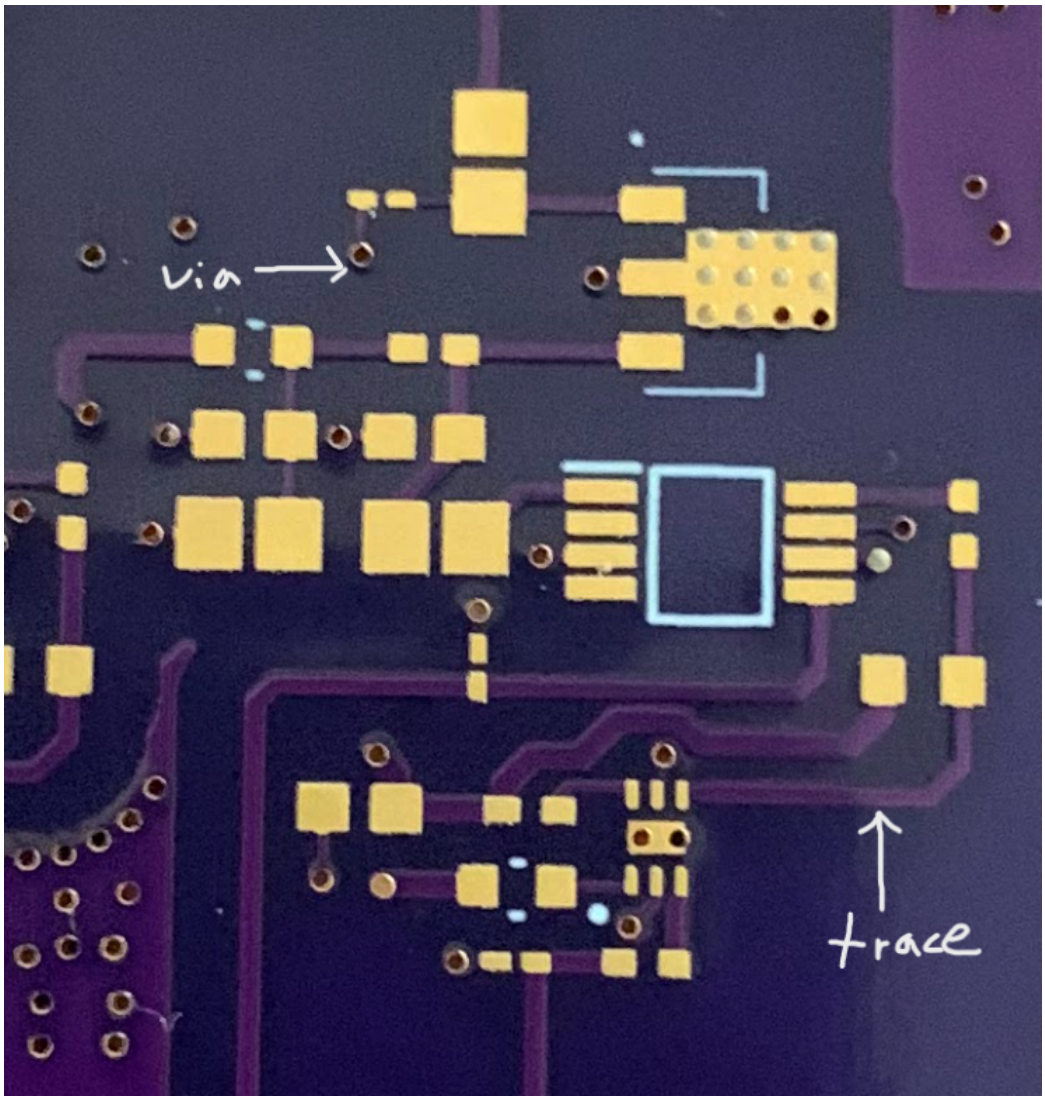

Figure 7.2: Traces and Vias from Mithril Board

In the last section, we mentioned copper layers being where copper is layed down where we want to transmit electricity from one place to another. This means we have to connect different components, power sources, grounds, etc. with strips of copper which we call traces. As can be seen in Figure 7.2, traces can vary in width, orientation, etc., and are basically drawn out by the designer based on where they want their electricity to go. Their width, orientation, etc. are important, and their affects can be found in Chapter 4. We also discussed that there can be multiple layers of copper within a board which is great for saving space, but how can we connect one layer of copper to another when there is an insulated dielectric between them? The answer is vias, which are copper plated holes drilled in the board that allow for connection of copper in different layers. As can be seen in Figure 7.2, the vias are very small holes which can vary in diameter and connect to a trace.

Like traces, the via diameter can affect its conductive properties in a similar fashion. By using traces and vias in tandem, we can build out our schematic in a small, consolidated board.

7.3 Components

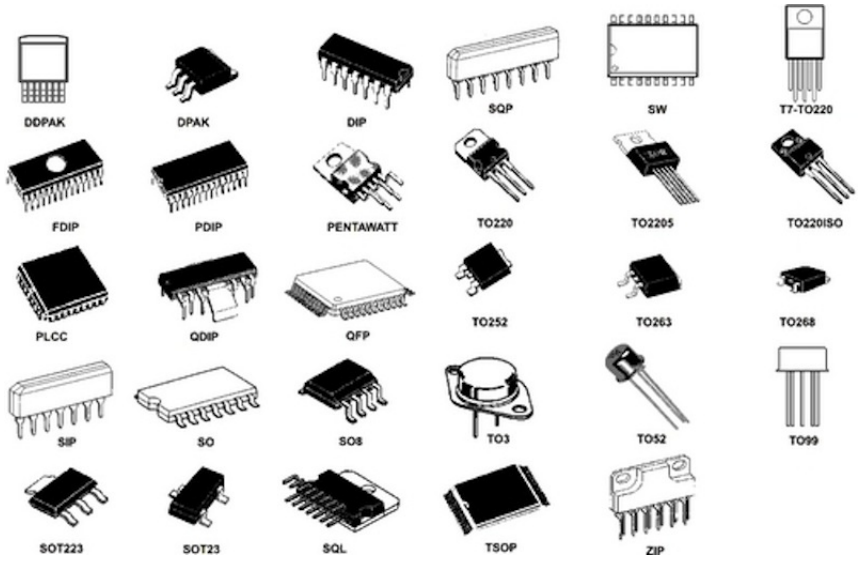

Figure 7.3: Component Package Types

Traces and vias are only useful when they are used with components that execute some task. Whether it be primitive parts like resistors and capacitors or integrated circuits, components are what make a PCB functional. Components usually come in standardized shapes with standardized pin placements called packages. Some common package types can be found in Figure 7.3. These components will eventually be soldered onto the PCB, so the designer must put solder pads for the component pins with correct spacing, pitch (width), and orientation.

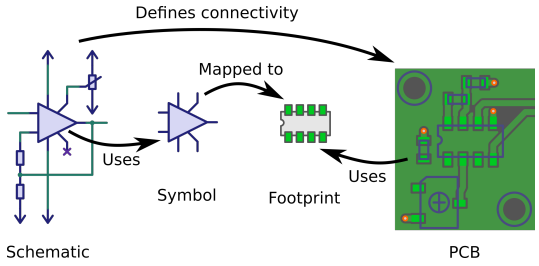

Figure 7.4: Footprint Schematic Symbol Relationship

Instead of the designer drawing all the solder pads with exactly the right dimensions by hand, CAD tools help in creating a ”footprint” or a digital drawing of the size and solder pads of a component. Footprints encapsulate all the physical aspects of a component as mentioned above, and the designer can map the schematic symbol pins of the same part to the footprint so they know

what connects where as can be seen in Figure 7.4. These footprints follow physical dimensions found in a component's datasheet, and most parts have a footprint out there that someone else made which you can essentially drag and drop into your design.

7.4 Printing and Assembling

Once you have created your PCB with layers, traces, vias, and components, its time to print the board, order the parts, and assemble the board. The main deliverables when printing and assembling a board are "Gerbers", a Bill of Materials(BOM), and drill files. Gerbers are ASCII standardized files that describe all the parts of your PCB (coordinates of parts, orientation of traces, traces on different layers, etc) and are what printers use to automate the printing process. CAD tools generate these files after your PCB is done, and you can simply send these to your manufacturer to get your board printed! The BOM describes all the components that are used in your design and makes it easy for you to source your parts. Drill files specify the locations of vias and chassis treads or any other holes you might have in your board, and again are generated by your CAD tool.

7.5 Allegro Overview

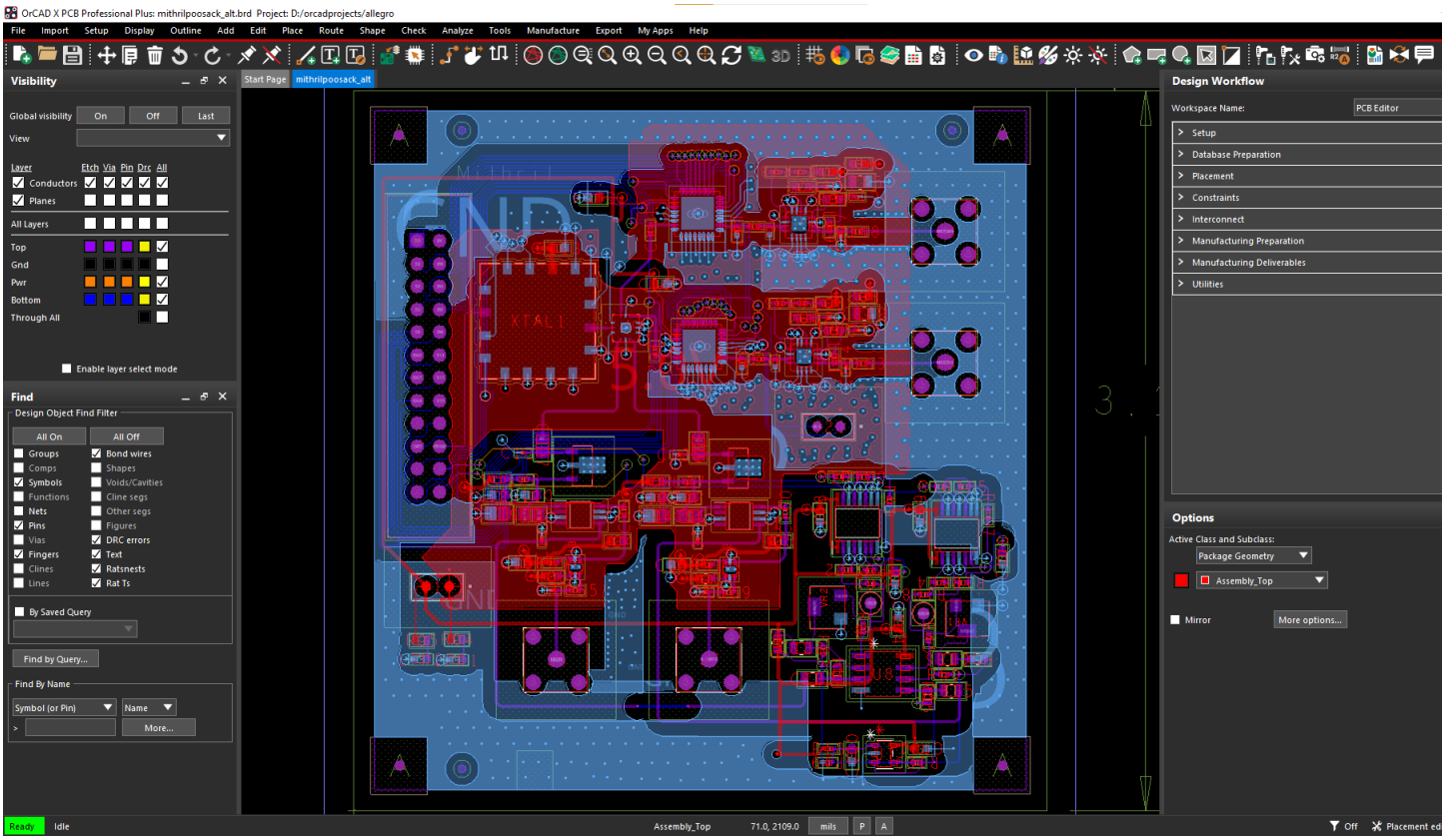

Figure 7.5: Allegro Layout

First we went over how to create a schematic in Capture CIS, now we will go over Allegro. Allegro is Cadence's counterpart to Capture which allows you to design a layout for the schematic that can be printed and assembled to create a functioning printed circuit board.

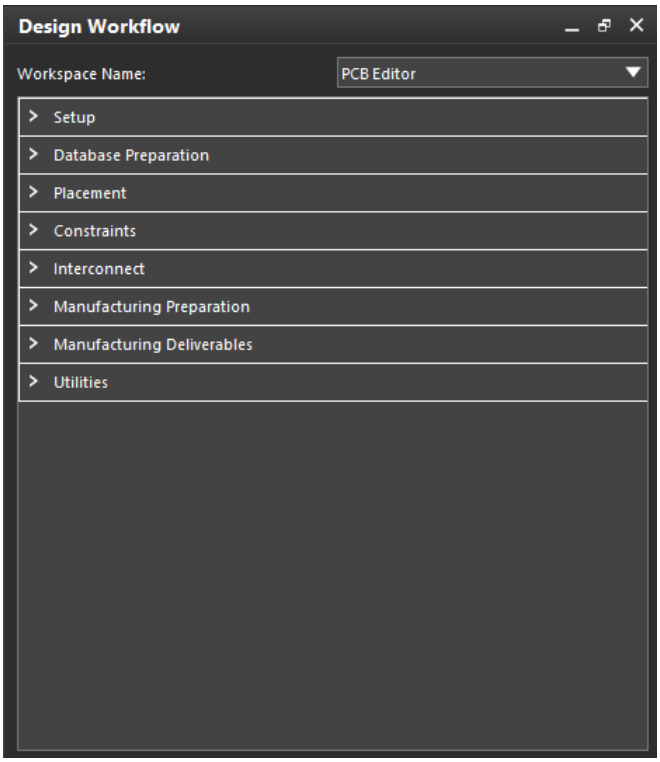

Figure 7.6: Design Workflow

The most important feature in Allegro is the design workflow. This pane shows all the steps you need to take in order to design a PCB in Allegro. We will go through all of these panes in the following sections, but make sure to have it open in your Allegro window to quickly go back and forth from different editing modes.

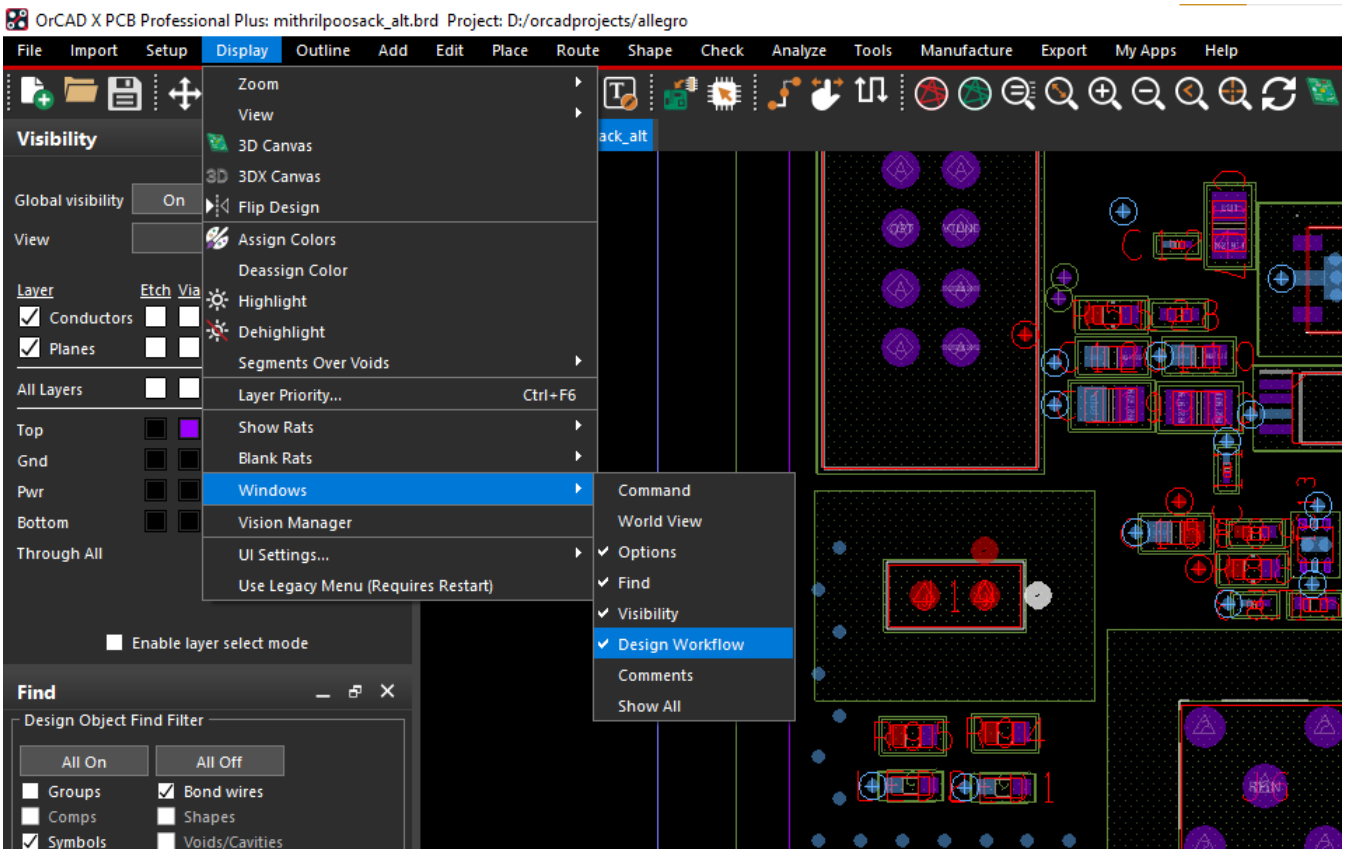

Figure 7.7: How to Display Design Workflow

If you do not see the design workflow pane in your design, on the top bar navigate to Display, Windows, and then select Design Workflow. You can then drag and drop the pane to put it wherever you want.

7.5.1 Design Setup

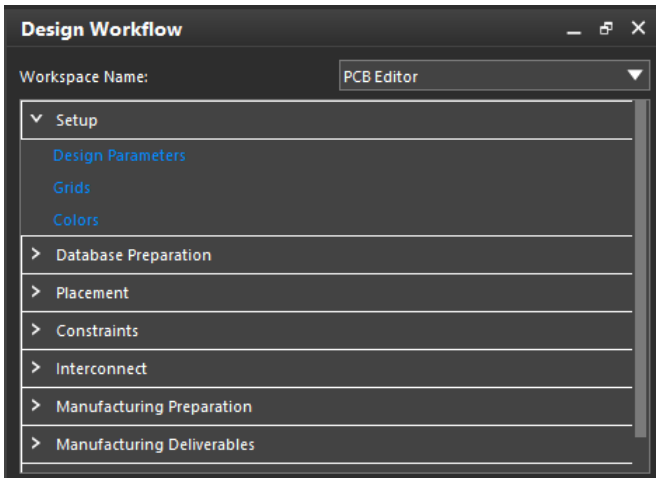

Figure 7.8: Design Setup

For Mithril, we followed the design workflow almost verbatim, and this is the workflow we will describe in this tutorial. The first step is setting up the design to fit our specifications. You can find the setup pane as shown in Figure 7.8.

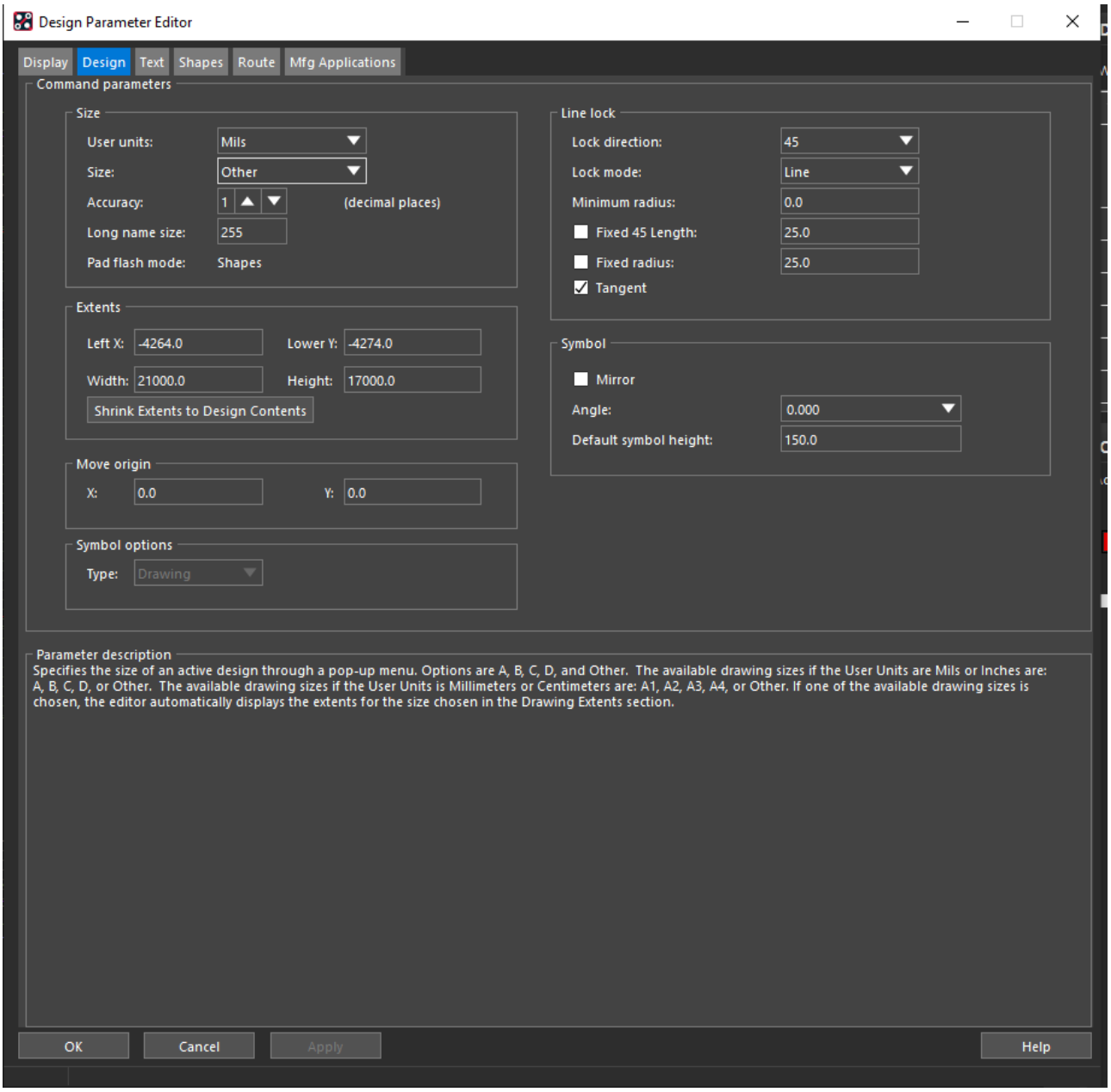

Figure 7.9: Design Tab

After clicking on ”Design Parameters”, you can navigate to the Design tab as seen in Figure 7.9. Honestly, this is the only tab that has stuff you want to change. One important thing to change is the user units, which is normally mils (tenth of inch) or millimeters. You can also switch grids on here, but I prefer them off to be honest.

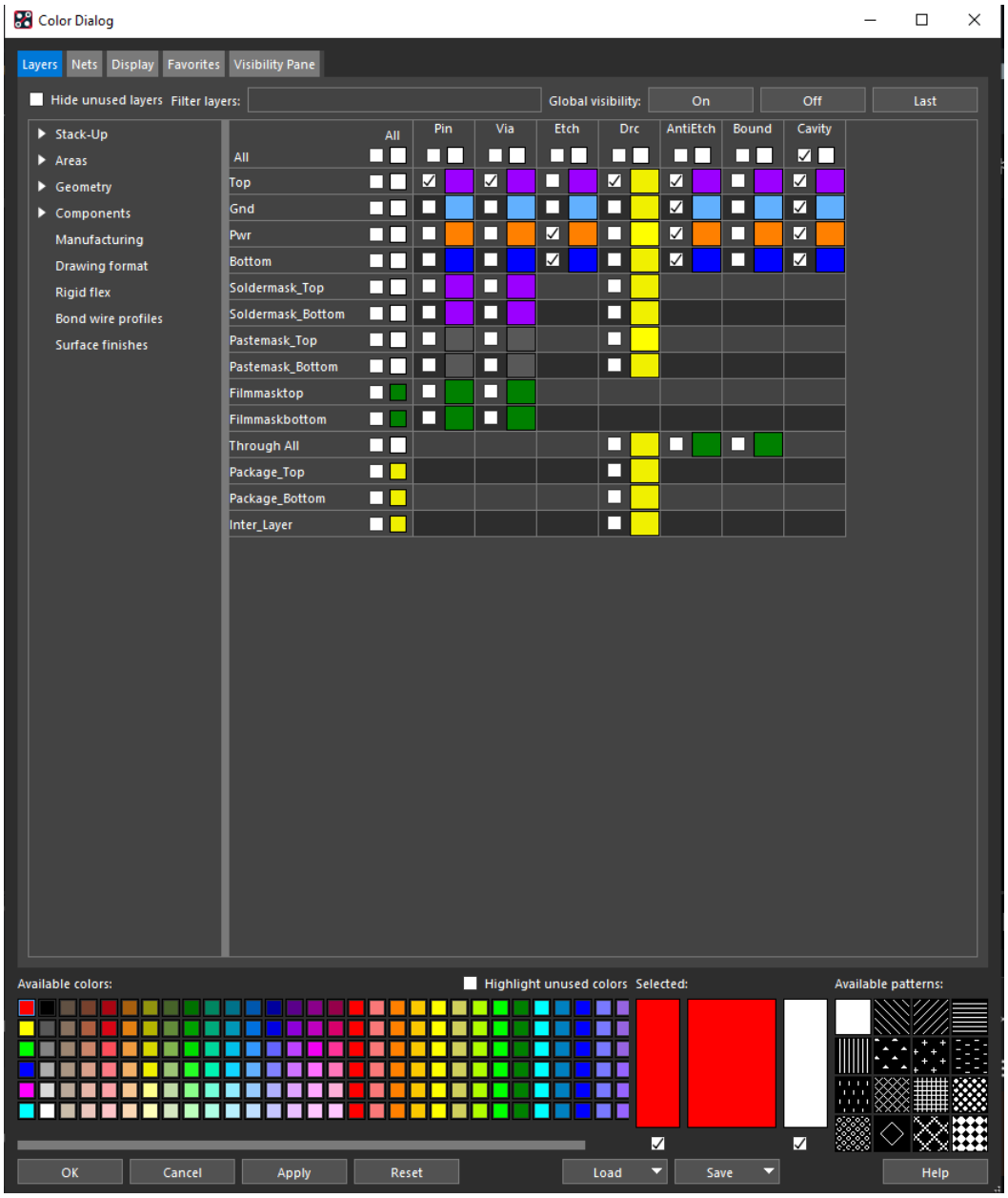

Figure 7.10: Layer Colors

The next, and quite frankly a very important tool, is the color tab. This is all really preference, but the defaults in Allegro are pretty bad and make it hard to see when designing. In the ”Layers” tab, there are several drop downs of types of objects that you can set colors for. You can see my Stack-Up section preferences in Figure 7.10, and to change colors you click a color on the bottom of the dialog, then click the little boxes next to a layer and type of object above it. The structure I used was to make everything in a layer the same color except DRC (Design Rule Check) which points out if you have an error. The first four rows are for my four layers, so I made sure to make them distinct colors.

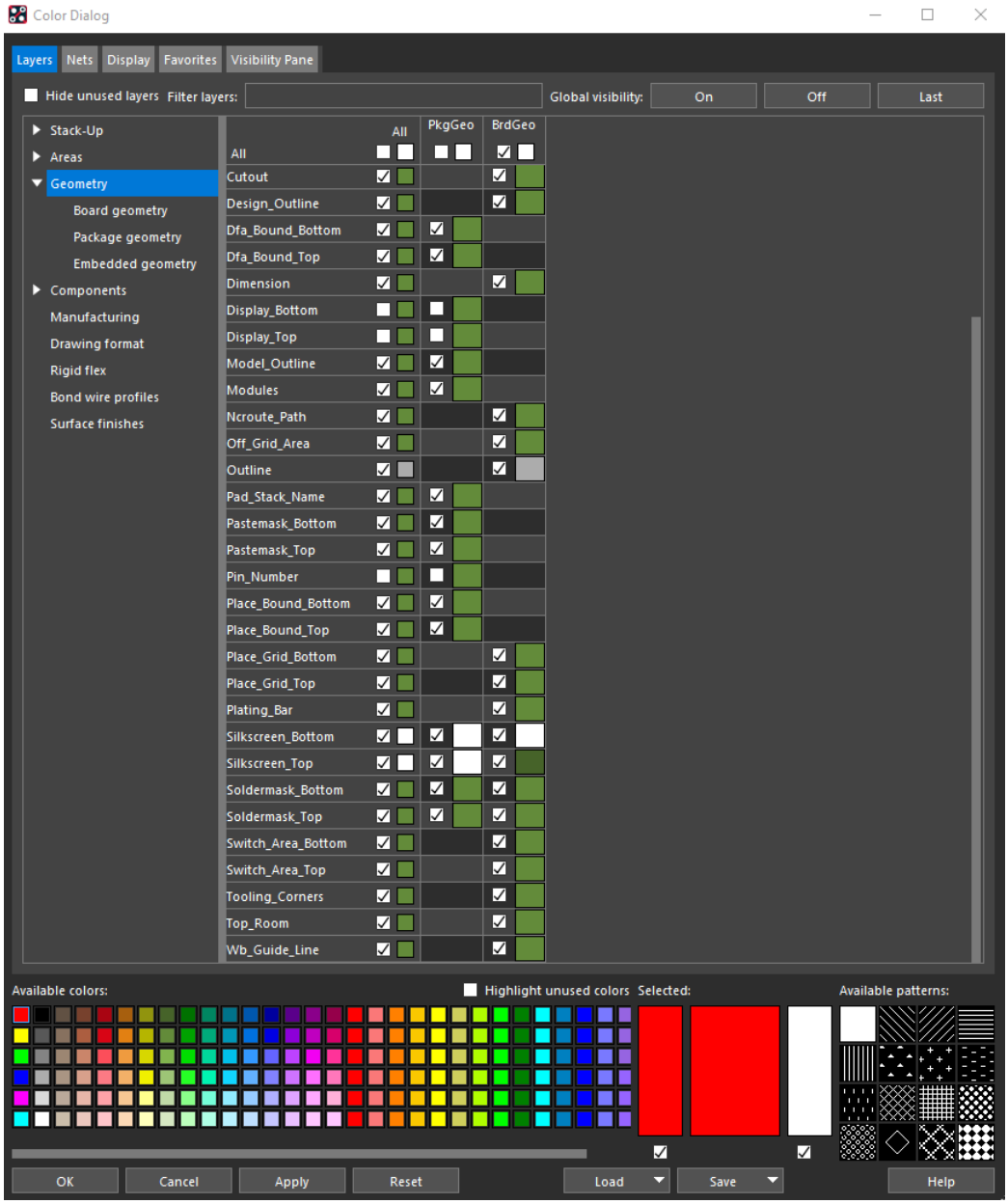

Figure 7.11: Geometry Colors

The layers will make it much easier to distinguish between layers, but there is a lot of bloat text that comes up on the screen with Allegro defaults. So, in the ”Geometry” tab of the colors dialog I switched off ”DisplayBottom”, ”DisplayTop”, and ”PinNumber”. I also changed the silkscreen text to white to make it easier to see. This will help with visibility in the long run, and you can see what the geometry dialog looks like in 7.11.

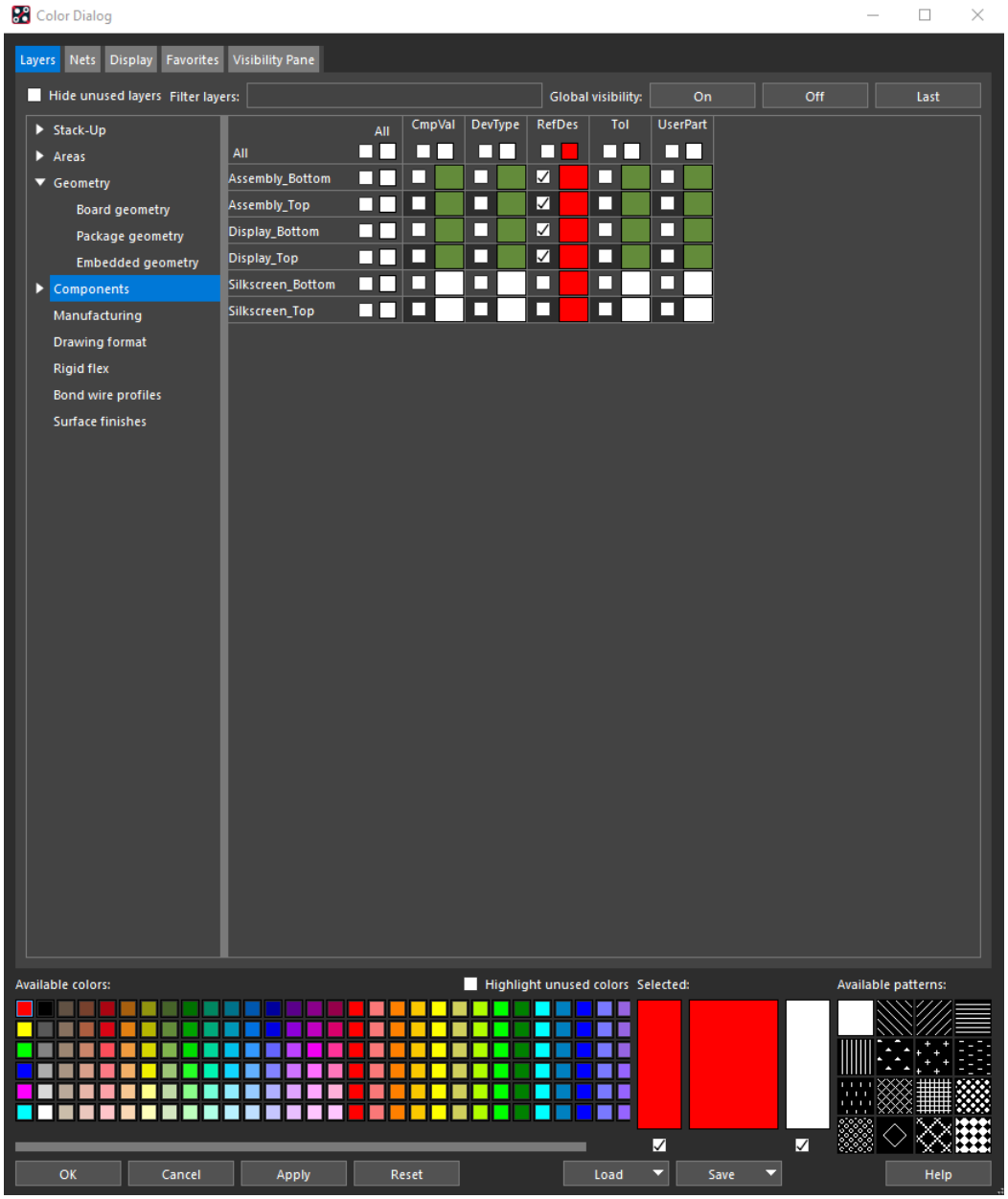

Figure 7.12: Component Colors

Again, to remove more bloat text navigate to the ”Components” tab and uncheck all columns except the ”RefDes” column. The RefDes column will put a text identifier next to every component that is the same as in the schematic, making it easy to identify what is what.

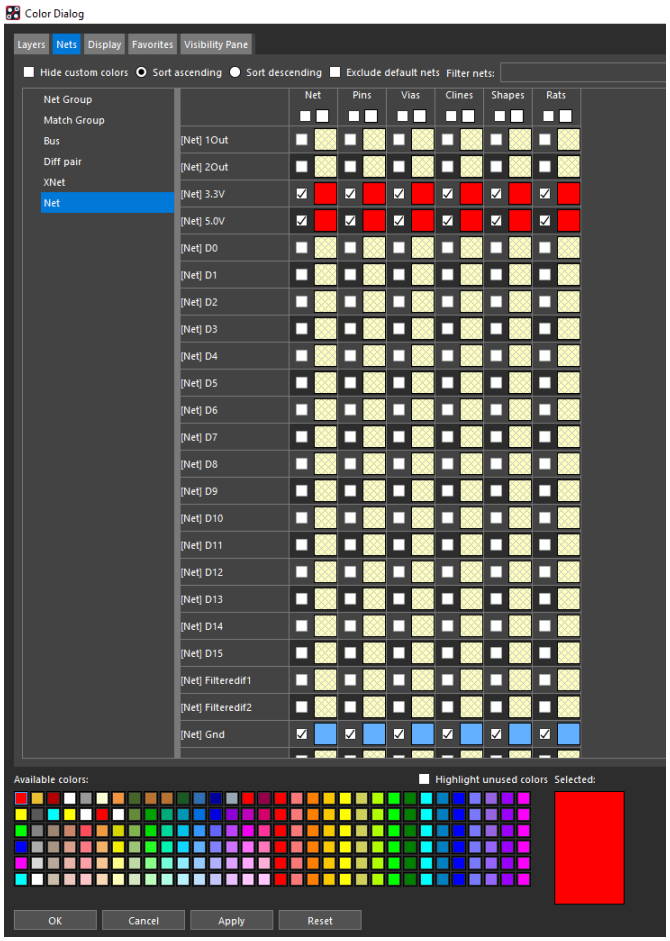

Figure 7.13: Net Colors

At the top of the colors dialog you can select the nets tab as seen in Figure 7.13. When we draw our traces, the traces by default will be colored what we selected in the stack-up

Database Preparation

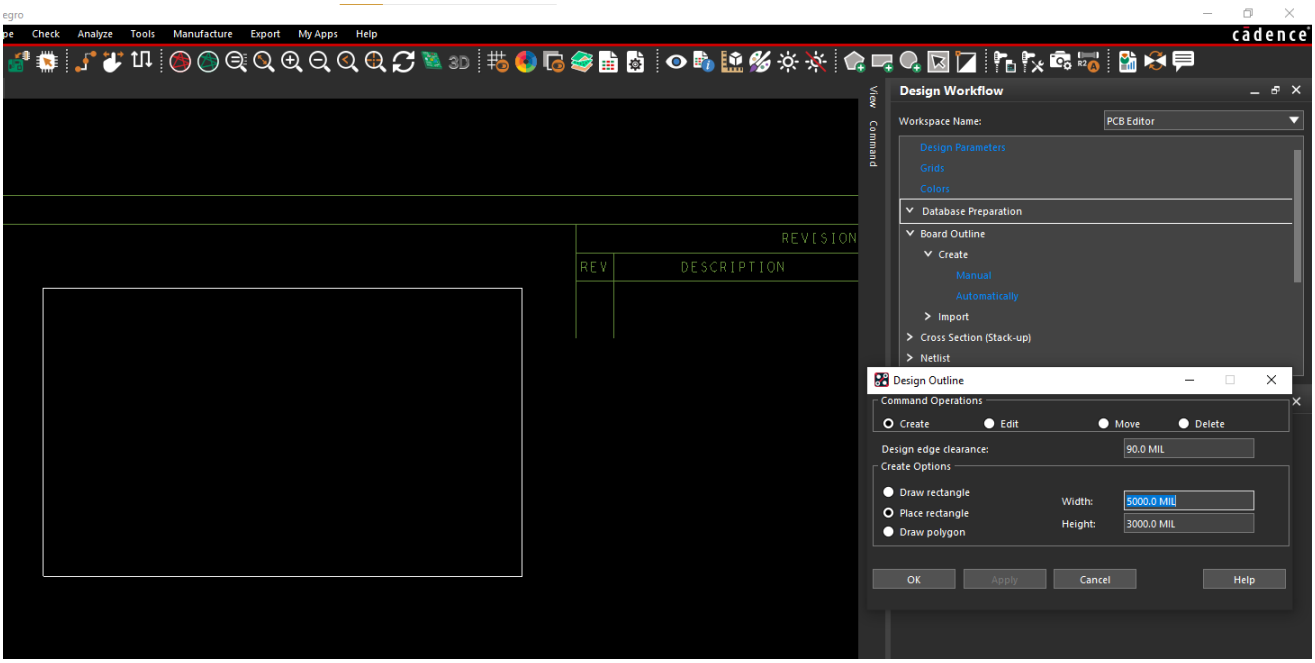

Figure 7.14: Board Outline

Now that our tool is setup, we can begin in our design. The first section in the Database Preparation is ”Board Outline” which can be seen in Figure 7.14. The board outline is for the designer to know the boundary of their future PCB. Pick the ”automatically” option under board outline to get the prompt in the figure. Click create in the prompt, and then click ”Place Rectangle”. Now you can enter the design edge clearance, which will dictate how far from the edge of the board you can place traces or components. Violating the clearance will pop up an error on your design for you to adjust accordingly. After entering your design clearance, you can enter the length and width of your desired board and a white line will appear that you can move around with your cursor. Then you can click for the final board and clearance outlines to be drawn.

Chapter 8

Digital Processing

Chapter 9

Networking

Chapter 10

Results

Chapter 11

Issues

Index