Designing Electronics that Work

Hunter Scott

This book is presented solely for educational purposes. The author and publisher are not offering it as professional services advice. While best efforts have been used in preparing this book, the author and publisher make no representations or warranties of any kind and assume no liabilities of any kind with respect to the accuracy or completeness of the contents and specifically disclaim any implied warranties of merchantability or fitness of use for a particular purpose. Neither the author nor the publisher shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused, or alleged to have been caused, directly or indirectly, by the information contained herein. Every company is different and the advice and strategies contained herein may not be suitable for your situation.

Digital version, v1.7

Contents

Preface

Successful electronics design is all about the details. You will find yourself accumulating a variety of techniques and tricks as you design and implement different products and devices. Some of these details may seem minor at the time, but their careful application can save a huge amount of time and money. This book is a collection of tips, techniques, and tricks that generally take a small amount of time and effort to implement, but have a disproportionate effect on the outcome of a design. The guidelines in this book are immediately applicable and almost all of them were learned the hard way: by screwing it up the first time.

This book was written to be relevant to the design of many different kinds of electronic devices, but does not cover all information for all classes of devices. If you're designing something that requires special certification, safety, or reliability constraints (like a medical device, spaceflight rated device, or safety critical device), use this book only as a reference and defer to the required standards. Perfectly following all of the guidelines in this book will not guarantee that your design will pass all certifications and work on the first try, but it will help you avoid many common and preventable problems.

This book assumes at least a basic understanding of electrical engineering. It's oriented towards product design and implementation, but is equally applicable to hobbyist projects, art installations, and other amateur electrical engineering projects. If you're designing your first PCB, reading this book may help you avoid mistakes you didn't know you were making. If you're designing your 1000th PCB, this book will act more as a checklist to help illuminate any area you forgot to think about. Some parts of this

book may seem like they are only applicable to the kind of work a professional electrical engineer would do at a company. And it's true that some of the more rigorous, process-based guidelines in this book aren't relevant for a hobbyist. But in those cases, the high level concepts *are* still applicable and useful. For example, it's probably unnecessary to write a traceable requirements document for a side project. But thinking about your design goals beforehand rather than just adding and removing features ad-hoc can help you actually finish your projects, prevent feature creep, and keep your cost down. So, before you dismiss a section that seems too "corporate", consider how the spirit of the idea can be helpful to you.

A note on mistakes

This whole book is designed to help you avoid mistakes. There are also dedicated sections on avoiding mistakes in particular areas. However, there is one critical technique to avoid mistakes that this book cannot do for you: you need to have other people review your design. They will catch things you miss either because you've been staring at the schematic for a week straight, or because they have experience (or have made mistakes!) that you don't. Get multiple people to look at the design, ideally from different engineering teams (electrical, industrial design, manufacturing, etc.). However, do not rely on your reviewers to catch all of your mistakes. Most people will only look at the high level system design, so don't expect them to be particularly perceptive to mistakes below that level. You must still own your design, take responsibility for it, and be intimately familiar with it.

Inevitably though, mistakes will happen. It's easy for novice and experienced designers alike to get frustrated at themselves when they discover they've made a silly mistake. This is normal. I struggled with this for a long time because it made me think I was a terrible engineer whenever I found a stupid mistake I made. I have talked to some of the most respected engineers in Silicon Valley who have designed influential products and have dozens of patents and deep domain expertise. I once asked one of them

how he got his designs right on the first try. His response? "Oh, you don't. Plan into the budget and schedule time for at least two PCB spins [revisions] because you'll always have to change something or find out you got a footprint wrong or something". One engineer told me that he specifically remembered the first time he got a design completely right on the first try. That gives you an idea of how often it happens, even for professionals. Spending too much time trying to get everything perfect can actually be bad, especially during design and prototyping, because it slows everything down, puts unnecessary pressure on people to be perfect, and can kill any creativity by getting bogged down with making a single iteration perfect. The good thing about silly mistakes is that they're usually easy to fix. I'll take an easy to fix silly mistake over a subtle bug that's hard to find any day.

Work on your design until you're 90% sure it will work, and then fabricate it. You will never go from design to shipping in a single iteration anyway, and even if you do get it perfect, the mechanical engineers will probably want you to change the dimensions of the board or something. The point is, it's ok to make mistakes. It can be really frustrating when the mistake you make is simple or avoidable, but when that happens, just write it down, figure out a fix, and move on.

DesigningElectronics.com

The website for this book, http://DesigningElectronics.com, contains a lot of useful resources. There you can find:

- A link to the latest version of this book
- A community for electronics designers
- Contact information for the author
- Links to recommended further reading
- Links to the tools and materials this book references

If you find a mistake in this book, or have a suggestion for an addition, please visit the website and reach out!

1

What to Build and How to Plan for it

1.1 Discovering Product Requirements

Every design is driven by both a set of requirements and a specification. If you're lucky, you'll be given both a set of product requirements and specifications that are reasonable and achievable. More often than not, however, you will need to create at least the requirements or the specification, or both. The *requirements* of a device are the list of things the device must be able to do. These are typically high level and don't include detailed technical implementation information. As an example, consider an underwater camera. A sample (abridged) set of product requirements might be:

1. Shoots 1080p video with audio
2. Syncs video with PC
3. Works up to 30 feet underwater
4. Battery life of at least one hour

Product requirements are typically a couple of pages long. A full product specification, on the other hand, may be one or

two pages to dozens of pages depending on the complexity of the device.

Leanna Rierson's book "Developing Safety-Critical Software: A Practical Guide for Aviation Software and DO-178C Compliance" has an excellent list of qualities that a good software requirement should have. A hardware-adapted version of her list is below:

- Atomic - each requirement should be a single requirement.
- Complete - each requirement contains all of the necessary information to define the desired system functionality. Each requirement should be able to stand alone without further amplification.
- Concise - each requirement should simply and clearly state what must be done and only what must be done. It should be easy to read and understand - even by nontechnical users of the system. In general, a requirement should not contain over 30-50 words.
- Consistent - requirements should not contradict or duplicate other requirements. Consistent requirements also use the same terminology throughout the specification. Requirements are not the place to practice creative writing.
- Correct - each requirement should be the right requirement for the system being defined. It should convey accurate information. This attribute is ensured by the requirement's validation effort.
- Implementation free - each requirement should state what is required without identifying how to implement it. In general, requirements should not specify design or implementation. That's for the product specification to define. However, there may be some exceptions, such as interface or derived system requirements.
- Necessary - each requirement should state an essential capability, characteristic, or quality factor. If the requirement is removed, a deficiency should exist.

- Traceable - each requirement should be uniquely identified and easily traceable through to lower-level requirements, design, and testing.
- Unambiguous - each requirement should only have one interpretation.
- Verifiable - it should be possible to confirm the implementation of each requirement. Therefore, requirements should be quantifiable and include tolerances, as appropriate. Each requirement should be written such that it can be verified by review, analysis, or test. Except for rare cases, if the behavior cannot be observed during verification, the requirement should be rewritten. For example, negative requirements are generally not verifiable and require a rewrite.
- Viable - each requirement should be able to be implemented, usable when implemented, and helpful to the overall system construction.

Requirements like resolution or battery life might be obvious, but don't forget about the more subtle requirements. If you're building a medical device, does it need to survive being sterilized in an autoclave? That's going to change your temperature requirements. Does it need to be bio-compatible? Will it be undergoing compression and decompression cycles in an aircraft? The list below contains some (but not all!) commonly missed requirements that may or may not be relevant to your design.

- Operating temperature
- Operating altitude
- Vibration
- Shock/impact resistance
- Physical size
- Finger/hand intrusion

- Solid particle protection (dust, tools, insects, etc.)
- Liquid ingress (dripping water to full submersion)
- Plasticity/flexibility
- Pressure
- Enclosure thermal/electrical conductance
- Security/encryption
- Shelf life for products that contain a battery

Depending on how you formulate them, some of these particular requirements might be better suited for inclusion in the product specifications instead. However you decide to write your product requirements, select them to allow maximum longevity of the product without over-engineering.

Here's an exercise that all hardware engineers should do at least once in their life. Take a visit to the nearest dump. The bigger the better. Get out of your car and look at how big the dump is. Think about the billions of pieces of trash in there. Consider that everything that isn't organic matter was once a manufactured product. Engineers designed it and produced it and sold it. Maybe people used it for a generation, maybe they used it for an hour. It doesn't matter anymore because now it's sitting in a huge pile of other products waiting for... nothing. As hardware engineers, we build physical things. That means at some point, everything you design and manufacture is going to end up in a dump. That doesn't mean you should think of engineering as a nihilistic exercise in futility. It *does* mean that you should think about the materials you design into your product, the reliability of the product, and the repairability of the product. Choose materials and processes that are as efficient and ethical as possible. Consider the full lifecycle of your product so that one day when it's sitting in a dump, you'll be able to sleep at night knowing it's not leaching mercury into someone's water supply. If a hazardous material must be used in your design, design for recyclability and safe disposal. For example,

sealed modules or cartridges containing the hazardous material are easier to recycle or remove and isolate than a reservoir that can spill or requires destructive disassembly. Your design should also be lead-free and RoHS compliant. Fifteen years ago, it could be challenging to meet RoHS requirements because a lot of parts still contained lead. Nowadays, it's possible to design a PCB that is lead-free without even realizing it.

It's easy for feature creep to first materialize in the product requirements. You begin to say things like, "We already have X, why not add Y?". Before long, you're left with a product that doesn't do any one thing really well, costs a lot of money, and no one will want to buy it. At the other extreme, being too minimalist will result in a product that is outdated before it even hits the shelf. So how do you pick product requirements? The shortest answer to this question is to talk to your users. Find the people that you are building your product for and literally take them to a coffee shop and just let them talk about what they want and how they will use it. You will need to do this a lot, with a lot of different people. A product may require dozens and dozens of interviews. However, the answers you get from people are still not the product requirements. There's a famous quote from Henry Ford that goes, "If I had asked what people wanted, they would have said a faster horse". The object of these meetings is not to build exactly what most people describe. Instead, it's to extract product requirements. The best way to do this is to ask "why" a lot. If someone says they want a faster horse, try to understand what the underlying reason is and then address that reason in your product requirements. The way you will actually achieve those requirements will be in the product specification that comes later. Talking to users to get useful product requirements is harder than most people think. I highly recommend the book "The Mom Test" by Rob Fitzpatrick as a guide for how to do this correctly. It's the best book I've ever read on the subject.

Some people talk about defining your users when writing product requirements. You don't "define" your users. You discover them. They may not be the people you thought they were. They may not even exist. The only way to find out is by talk-

ing to the people you think will be your users and following the trail until you get the elusive product-market fit. Don't worry so much about what your competitors are doing either. The most important thing is to make something people want.

Product requirements shouldn't include negative statements about what a product *shouldn't* do, but it can be valuable to think in those terms as you're writing the product requirements. It can help differentiate you from a competitor or improve the usability and experience of how the product is encountered and used. Doing this can also help you avoid scope and feature creep, since you will be explicitly deciding where the boundaries of the product are.

As the number of product requirements increases, achieving total coverage of those requirements gets harder and harder. Table 1.1 shows the number of requirements of several real projects (collected by Capers Jones for his book "The Economics of Software Quality"). Notice that the total number of lines of code (LOC) very quickly increases as the requirements grow and the number of those requirements that are actually implemented decreases.

LOC	Pages of Requirements	Requirements Completeness
1000	14	97%
10000	115	95%
100000	750	80%
1000000	6000	60%

Table 1.1: Requirements length vs. completeness

Most projects will not have more than about a dozen pages of requirements, but very large, multi-year, big budget projects may have incredibly extensive product requirements. The F-35 Joint Strike Fighter codebase is over 8.5 million lines long; think about how many pages of requirements that represents! The mismanagement lessons learned from the F-35 program are another topic entirely, but one way to try to keep your requirements completeness high is to minimize the number of requirements (which means avoiding feature creep) and to break large projects up into

subprojects.

Writing the requirements for a product or prototype is not something most engineers usually look forward to because you don't get to solder anything or chew on any interesting circuit design problems. But the requirements drive the entire project and if you get them wrong, it doesn't matter how genius and elegant your design is, no one will ever see it. People won't want to buy it, or worse, it may never even make it out of the factory. The reason this chapter is first in this book is because writing the requirements is the first thing you should do for a new design. The decisions you make here will drive all of the other decisions you make in picking parts, doing layout, assembly, and test.

One other important thing to note about product requirements is that while they can be allowed to change, this gets more and more expensive (in time and money) as the project progresses. If you need to make any changes, make them as early and as soon as possible.

The feedback loop for which requirements you use and which requirements you drop should again be driven by talking to users. It's extremely important to build beta hardware and get it in the hands of your users so you can learn the changes you need to make to your requirements. It will check assumptions you made about how the product will be used and how intuitive it is to use. But to really get honest feedback, you need to make your beta users pay you money for your device. If they have to give you their own money for it, they will be sure to let you know if they think it was worth it or if they're not satisfied. It raises the stakes. You may feel weird for making someone pay for an Arduino stuffed inside a 3D printed enclosure that looks bad, and indeed, some people will refuse to buy it. But if you can find people who will pay you for even a crappy solution to their problem, you not only will get honest feedback from your first users, you will also have shown that the problem you're trying to solve is bad enough that people are willing to try even an ugly solution.

This book is concerned with electronics design, not mechanical or industrial design. But of course, electronics design is not independent of these other disciplines, which will inevitably af-

fect and influence the electrical design. Be aware that you'll need to consider not just the *electrical* product requirements and specifications, but the other requirements and specifications from disciples that are outside of the scope of this book.

1.2 Writing Specifications

A product *specification* lists the technical requirements needed to achieve the product requirements. Continuing with the underwater camera example from before, the corresponding product specifications might be:

- Contains 1920x1080 RGB CCD capable of 60 FPS [1]
- Capable of capturing 32-bit audio [1]
- USB micro port [2]
- Enumerates as USB Mass Storage device [2]
- MicroSD card slot with support of up to 64 GB [2]
- File transfer over 802.11a/b/g/n [2]
- IPX8 compliant [3]
- Operates from 0C to 40C [3]
- Battery charging over USB [4]
- Replaceable battery [4]
- Battery life of one hour with camera on [4]

The bracketed numbers after each specification cite the product requirement that is being fulfilled by that specification. There will almost always be more specifications than product requirements because a high-level product requirement will need several specifications to be fully implemented. The product specification is what you will use to help select components, calculate the link and power budgets, and estimate the time and money required

for the project. In an ideal world, no project would ever go over budget or off schedule because the time and money required would be conservatively estimated from the specifications. If any of the specifications required too much time or money, the product requirements would be changed to put the specifications back below the threshold. This feedback loop is extremely important in making a product successful.

Because a well written product requirement is not specific to any particular implementation, there will likely be many unique product specifications that can fulfill a given set of product requirements. Good engineering starts with product specification design. Choose technologies and approaches that will keep the final product robust, low-cost, and elegant.

Finally, design your specifications to meet but not exceed the product requirements. There is no reason to design something "better" than the requirements if the requirements were written correctly. Designing above what is required only introduces liability and risk. If you feel that it's necessary to design past the requirements, change the requirements.

1.3 Writing the Test Plan

To verify that all of the requirements are actually met, each requirement needs to be tested. After writing the requirements and specifications, go through each item and design a test that can be used to prove the requirement has been met. For example, if your requirement says that the device needs to operate from 0C to 40C, you need to design a test that subjects the device to that temperature range using a thermal chamber. When designing your tests, think about whether or not any particular requirement may be affected by another requirement. Normally, each requirement is tested independently, one at a time. This is mostly because the equipment used for testing can only test one or two parameters at a time, but it's useful to consider whether or not two simultaneous tests could cause a device failure. Maybe your design passes your thermal test and vibration tests separately, but when the enclosure gets really cold, screws contract and can

work themselves out in vibration. It's usually not practical to test all parameters at all extremes at once, so use your judgment when choosing which, if any, tests to do simultaneously. The best thing to do is to get prototypes of your device into the real environment and watch for premature failures.

1.4 Accurate Schedule Prediction

The best manager I ever had was great for a lot of reasons, but one of the skills I admired most was his ability to accurately plan the project schedule. Engineers have a particularly difficult time estimating how long something will take because we are, by nature, optimists. We tend not to think in terms of how long it will actually take, but about how long we would like it to take, or how long it would take if nothing went wrong.

But that's not the only reason it's difficult to plan a design schedule. The process of engineering (especially something that's never been built before) is full of hidden traps and delays. You will encounter problems of different difficulties and it's hard to predict how many problems you'll hit and how quickly you'll be able to solve them. You don't know whether you're about to plow through a block of foam or granite[1]. A full discussion of schedule planning and prediction is a whole other book, but here are some basic tips to get you started.

One way to speed up hardware iteration time is to pay more money for faster turn times and faster shipping. It's almost always worth it to pay more money for a faster manufacturing run and overnight components. Think about how much paying your engineers costs each day and compare that to how much it costs to shave a day or two off of the PCB manufacturing process or paying for overnight shipping on parts.

The ability to perform skilled rework (either with a technician, or by yourself) can save you entire board spins and remove what would otherwise be at least a week of extra delay from your schedule. Now, you can't always make the changes you need with just a rework. But you may be surprised to see what

[1] This analogy is from Paul Graham's essay "Relentlessly Resourceful"

kind of advanced rework is possible. Adding or remove discrete parts, adding or removing ICs, and changing trace routing (even on internal layers) are all usually possible. In some cases, you can't perform a rework that will have 100% of the performance or functionality that you would get with a new PCB, but it's good enough to de-risk the fix so that you can be confident your new PCB will work before you've fabricated it.

Risk reduction is the name of the game. Always have a contingency plan. You don't always have time to start over or begin a new approach if your first plan doesn't work. You can overcome this problem by either building multiple versions of your design in parallel, or you can build contingency plans into the same PCB. Add footprints for parts that you can use if something doesn't work, add sub-circuits that you can rely on if you need to, and give yourself plenty of opportunities for easy rework in case it's necessary. Prove out pieces of a design before you put it all together. Use development boards or quick turn some PCBs of individual sections of your schematic to verify that everything will work the way you think before you spend all the time and money on the completely integrated system. The larger and more complex the design, the more important this is. Having small prototypes or breakout boards of individual chips before you build the main board can also help keep software on schedule since they can begin programming and developing on ICs much earlier.

If you can afford it, rapid prototyping tools like a circuit mill can add more speed to your iterations by allowing you to try things in the same day instead of a week. You'll still need to have parts on hand (and this is why it's useful to keep a reasonable parts library in stock), and you still won't have the same design flexibility that fabricating a PCB in a factory would give you, but it's great for de-risking things. A circuit mill can be especially useful for antenna development since no components are needed besides maybe a connector and some matching components.

Many projects and even entire companies have failed because of an unexpected production delay or a delay in getting a component. Source your components as early as possible and always have a backup supplier. Don't forget to take into account any

necessary long delays like certification or testing. During assembly itself, don't forget to take into account steps that take longer, like cure times for adhesives or composites. Otherwise your production may be on time, but your *rate* of production will be slower than expected and that can still result in delayed shipping.

1.5 How Manufacturing Works

Crowdfunded hardware products are notorious for being almost universally late. One of the leading reasons for this is an inaccurate or incomplete understanding of the manufacturing process. If you've never gone through the whole process of production for a significant quantity of a new product, it can be easy to underestimate. Every product will be a little different, but the product development lifecycle can usually boil down to the flowchart in figure 1.1.

The looks-like, works-like prototype is usually built with a 3D printed enclosure and electronics that are pretty close to what you think the final version will be. Once you select a contract manufacturer (CM) to actually build your design, you'll send your design files to their engineering team so they can begin tooling. Your CM will work closely with you and help with production and engineering reviews. Many products use an injection molded enclosure or have injection molded parts. The process of creating the tools for performing the injection mold is several steps itself, but that process is out of the scope of this book. The main thing to know is that tooling usually takes at least three iterations. Tools are often complex devices with moving parts and are made of very hard materials that take a long time to machine. Your CM will give you a good estimate of how long this process will take, but be aware that it can take longer depending on how experienced your mechanical engineers are, how good the engineers at the factory are, and how complex your product is. While you're working on this, you should also be designing the automated test equipment (ATE) that will be used in later phases. ATE equipment includes things like a bed of nails

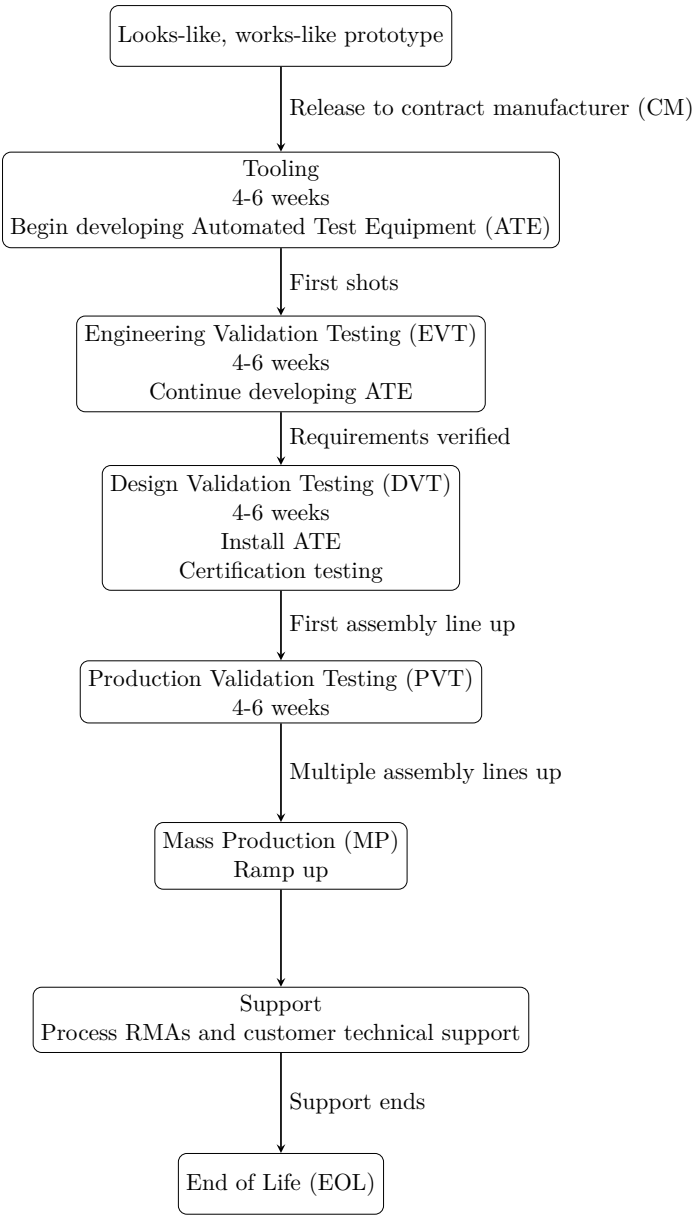

Figure 1.1: Product Development Lifecycle

test jig, programming stations, and test fixtures that quickly and automatically exercise all functions of your product to verify everything works before it goes out the door. More information about ATE is in the "Testing" chapter.

After your first shots on your tools ("first shots" are the first test injection molded parts that come out of your molding tool), you'll be able to assemble engineering models and enter the engineering validation test (EVT) phase. This will be the first version of your product made with injection molded parts. You'll likely need to iterate the PCB at least once during this phase. Ideally you will verify 100% coverage of product requirements during this phase, but it may be spread out into the next phase as well. The more problems you can discover in EVT rather than DVT, the cheaper it will be to fix them.

Once your requirements have been verified (or mostly verified), design validation testing (DVT) takes place. In this phase, you'll complete any compliance or regulatory testing. This is things like FCC, CE, RoHS, and UL testing. More details about this can be found in the "Testing" chapter. The units assembled during DVT will have molded parts that were shot on final hard tools. ATE should be done, and you should be able to produce units by hand with all final parts.

The next step is Production Validation Testing (PVT). At this point, your first assembly line has been set up and you are debugging mistakes in assembly. Units being produced are good enough to be sold to customers.

This phase should quickly become Mass Production (MP). Multiple assembly lines are brought up and the main focuses are keeping the yield up, keeping the lines running, and managing suppliers who may be bottlenecking your production.

Concurrently, since customers are now buying and using your product, you'll begin supporting them. Especially early on, this will feed back into your production line to fix issues like premature failures or other unforeseen usability issues. Your ability to make significant changes will be minimal unless you can spend a lot of time and money.

Eventually, after production has ceased and the decision has been made to no longer provide support, the product will be

considered end of life (EOL). Planning when EOL will happen is an important part of scheduling. Consider the schedule of your other product lines and how the economics of offering and supporting multiple products simultaneously will impact your business.

There's an important date to know every year to make sure your manufacturing doesn't get unexpectedly delayed: Chinese New Year. If you do any manufacturing in China, or if any of the parts or subsystems you need are made in China, or if you do any contracting in China, you will be affected by the Chinese New Year. Factories generally shut down for about 2 weeks in February, and it can take up to 2 weeks after that for factories to get back up to normal production volume. Part of the reason this ramp up takes so long is because many workers use this break as an opportunity to find a new job, and factories can see a roughly 25% reduction in their workforce. That means they need to hire and train new workers. The exact date of Chinese New Year changes every year, and not everyone takes exactly the same amount of time off. Talk to your contractors and factories to learn exactly when they'll be unavailable and make sure you build in that time to your schedule. Literally dozens of crowdfunded hardware projects have been delayed because they didn't know about or didn't correctly plan for Chinese New Year. If you need product for the first quarter, make sure you place the order so that it's completed by the end of January. Also keep in mind that China is not the only country that celebrates Chinese New Year. Vietnam, Thailand, and other southeastern Asian countries celebrate it to varying degrees. Of course, you can avoid Chinese New Year completely if you manufacture in a country that doesn't celebrate it, like the US, Mexico, or Europe. The other big holiday in China is National Day holiday week during the first week in October. Again, talk to your factory to understand exactly how this will impact your production schedule.

2

Selecting Components

2.1 How to buy parts

The goal when picking parts for your design is to ensure that each one meets all of your specifications. Maybe you find a great LCD screen that has the resolution and refresh rate that you need, but will it function in the environmental conditions required? In addition to component characteristics, good part selection takes into account a myriad of other factors, including complexity, robustness, and logistics.

Actually procuring the parts you have selected is an often overlooked but critical task. Always buy parts from an authorized distributor or the manufacturer directly. This keeps you from receiving counterfeit parts or parts that are damaged due to poor handling or incorrect storage. Before you buy from that questionable overseas distributor that isn't authorized by the manufacturer, ask yourself, "is saving an extra n cents per part really worth the possibility of getting bum parts?" Even worse than all of the parts failing, you may see intermittent failures if only *some* of the parts work. This can send you on a wild goose chase trying to debug your design or assembly process. It can also cause bad parts to fail once they're in the field, meaning higher than expected returns and unhappy customers.

Buying parts from strange sellers gets more tempting when all of your normal distributors are out of stock and you really

need parts fast. You can risk it, but be aware that it is definitely a risk and that you will need to perform extra quality assurance on those units. The best way to avoid this problem in the first place is to make sure that all parts you pick are in stock with at least one distributor, preferably two. Other companies will sometimes buy out all of the stock of a particular component from one or more distributors for a small manufacturing run. This can result in the unpleasant surprise of seeing a distributor have 2000 parts in stock one day and then zero in stock the next day. This is why it's critical to choose only parts that are for active design use. In other words, they are not marked "end-of-life". A part that is marked "end-of-life" is no longer being manufactured or supported. Some distributor websites will let you filter parts by their lifecycle status, and you should use that feature if it's available. You can also check the website of the original manufacturer, or just talk to one of their application or sales engineers. No parts should be marked as obsolete and all parts should be marked "active" all the way through the end of your expected production schedule. Manufacturers don't always put this information in the datasheet or on the website, so contact them to confirm if you need to.

When surface mount (SMT) components are made at their original factory, they are packaged into a long strip of tape. This tape is either a thick paper or thin plastic with small grooves cut into it for SMT components to sit in. To keep them from falling out, a thin, clear piece of plastic is stuck on top of them. A similar technique is used for through hole components, but the tape holds the leads in place and the body of the component sticks out sideways. In either case, these longs strips of tape are rolled up into a reel.

Figure 2.1: Piece of SMT cut tape.

Figure 2.2: A reel of SMT components.

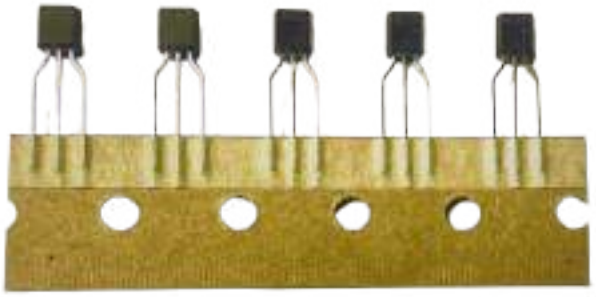

Figure 2.3: A piece of through hole cut tape.

The number of components that are in a reel depends on how big the component is, but a typical number is 3,000 or 10,000 (see the "Fabrication and Assembly" chapter for more information on how reels are used). If you're going to be populating parts with a pick and place machine, you'll need them on a reel. Some component distributors like Digi-Key will put custom numbers of parts on a reel for you so you don't need to buy 10,000 parts if your build requires less than that. If you need very small numbers of parts or are assembling by hand, you should order cut tape. Cut tape is exactly what it sounds like: small pieces of the tape are cut off from the reel with however many parts you need. Many distributors like Digi-Key and Mouser will let you order even single quantities of parts as cut tape. If you have loose cut tape that you need to use on a pick and place machine, there are companies that will re-reel parts for you, but some assembly companies will accept cut tape and re-reel it themselves.

Buy more than the exact number of parts you need. If you're hand assembling, you will drop or lose parts, especially tiny ones. Pick and place machines will also drop parts. Assemblers usually request that they be provided with an extra margin of parts for this purpose. If you have really expensive parts on your BOM, you can reduce the number of extras you provide and try to recover dropped parts. Recovering dropped parts is usually not worth the effort given their cost versus how much your time is worth. They would also have to be put back on tape somehow and even identifying which part is which would be painstaking. The exception is for physically large parts and very expensive parts. If your design uses a $5000 FPGA, you're not going to just forget about the ones that fall off your pick and place machine. Below are some guidelines for how many extra parts to buy.

Component size	Percent extra	Minimum extra
0201	50%	50
0402	25%	20
0603	15%	20
0805	10%	10
1206 or larger	10%	10

Table 2.1: How many extra discrete SMT components to buy

Component size	Percent extra	Minimum extra
0.6 mm or less	20%	20
0.6 mm - 1 mm	10%	10
1 mm - 1.6 mm	5%	5
1.6 mm - 2 mm	5%	2
greater than 2 mm	5%	1

Table 2.2: How many extra active components to buy

Component size	Percent extra	Minimum extra
Small devices	5%	2
Other devices	2%	1

Table 2.3: How many extra through hole components to buy

Component cost	Percent extra	Minimum extra
Less than $0.50	10%	5
0.50−2.00	5%	2
Greater than $2.00	2%	1

Table 2.4: How many extra mechanical fasteners to buy

Component manufacturers usually don't sell directly to most customers. Instead, they sell to an electronics distributor. The most popular distributors are Digi-Key and Mouser. Other reputable distributors include Avnet, Arrow, and Newark/Farnell. These are not the only reputable distributors, but these are the

ones that most electrical engineers are familiar with. Not all distributors charge the same amount for the same parts, so it's worth it to shop around for components if you're building a large quantity of anything.

There are several online tools that can help you do this. Octopart.com is a free website that allows you to import your bill of materials (BOM) and find the best price for each part from a large list of distributors. It's also great for cost engineering (see the "Cost Engineering" chapter for more information). Oemstrade.com is an alternative to Octopart that offers a similar service. Another good resource is findchips.com, which caters to a more professional market. Unlike Octopart, it's not free, but will give you more detailed and historical information about part availability and lifecycle to try to help you determine the supply chain risk of using a particular part. Siliconexpert.com is another competitor to findchips.com and also charges a monthly fee.

Some component manufacturers use more specialized distributors instead of Digi-Key or Mouser. The company RFMW is an example of this. If you go to rfmw.com, you'll see a list of the companies they represent and sell for. This list is also referred to as a "line card". Manufacturers that use these distributors don't want to spend the time or money on a large sales force, so they contract it all out to a specialized distributor. Specialized distributors can be really helpful because their sales engineers will help you with component selection and cost engineering. If you're planning on manufacturing a large number of your product, these sales engineers will also give you free development boards and can connect you to application engineers or other manufacturer tech support that you can't get any other way. They will also typically give you the best pricing on a part, even if that part is also available on Mouser or Digi-Key.

If you're in China, especially Shenzhen, there are a handful of Chinese language websites that you can use to find parts and suppliers, including local suppliers. If you can't read Chinese, you'll need to use Google Translate to navigate these websites. Some of the most popular ones are icgoo.net, www.ickey.cn, hqew.net, and szlcsc.com.

If you're looking for a hard-to-find datasheet or information about a part that's not readily available, try searching the forums at elecfans.com. It's a Chinese language website where users often post hard to find PDFs and schematics. You need to register an account to download things from the forum, and this can require some trial and error if you don't read Chinese. Again, Google Translate can help you figure it out.

Pay attention to your tolerances. If you are picking out a pull-up resistor, you probably care about the accuracy of your resistor a lot less than when you're picking out, say, a current sense resistor. Make sure you've gone through your design and specified the required tolerance so the right part gets ordered. If you pick a tolerance that is too tight, you'll spend more money than you need to. If you pick a tolerance that is too loose, your circuit might be unreliable or function incorrectly.

Try not to run parts at more than 90% of their specified voltage, temperature, and other ratings. This will help ensure performance since anything above the ratings in the datasheet has likely not been tested by the manufacturer. If you need to run a part above its nominal conditions or in a nonstandard way, be sure to talk to an application engineer about it and collect your own thorough test data before you trust the component to operate normally.

Knowing the failure mode of each part you select is also important. This depends both on the component itself and how it will be used. Will it fail open or closed? Will it get hot? Will it outgas anything? Don't choose a component with an unacceptable failure mode. This becomes especially important with capacitors, and will be covered in more detail in the capacitor section. For the components that you decide have an acceptable failure mode, use what you know about how they can and will fail to mitigate that failure. This might drive you to place the component in a particular area of your PCB or choose a certain enclosure design, for example.

If your product requirements call for a higher temperature range than normal components, look for automotive rated components. Many components are manufactured in both a standard operating temperature range and an extended operating temper-

ature range meant for automotive applications. These parts are tested extensively according to stringent automotive standards and are a good way of getting much more robust parts for not much more money. The level above automotive is mil-spec (suitable for military use), and parts rated for mil-spec are often significantly more expensive.

Every component manufacturer has resources available to help you use their parts successfully, including sales engineers and field application engineers. These people are a free way to parallelize your work. They will usually assist in part selection and give you a few suggestions of parts that meet your requirements. They know their product lines well, and they want to help make sure your design works! It's literally their job. So, take advantage of this and call them up if you have any questions about functionality, implementation, performance, or if you're planning on using a part in a non-standard way.

The component package is another important consideration. Passive components like resistors, capacitors, and inductors most commonly come in package sizes called 2512, 1206, 0805, 0603, 0402, 0201, and 01005. These numbers refer to the physical dimensions of the part in hundredths of an inch. For example, 0805 means the part is 0.08 inches by 0.05 inches. However, passive components are also available in that naming convention using metric rather than imperial units. This can get really confusing because several metric packages have the same names as some imperial packages but are vastly different sizes. The three you need to look out for are metric 0603 (which is really imperial 01005), metric 0805 (which is imperial 0302) and metric 1005 (which is imperial 0402). The most common problem this can cause is accidentally ordering the wrong part. Almost everyone in the industry uses the imperial naming convention when designing with or talking about passive parts. This book also uses that convention, so when you see a package size, it's in inches.

For most passive components, 0603 is a good balance between size and ease of assembly. They're large enough to be easily hand soldered, but small enough to not take up a lot of real estate on your PCB. Once you use packages smaller than 0402 (like 0201 and 01005), it's difficult to manually apply solder paste with a

stencil because the window that the solder paste must squeeze through gets so small. Those sizes also all but require the use of a microscope to solder by hand. Unless space is really at a premium, don't go smaller than 0402.

Packages like BGA (ball grid array) are great for size sensitive designs, but are difficult to rework and require an X-Ray to verify correct soldering during production. If size isn't a constraint and if an alternative is available, it's best to avoid BGA and similar packages. If you need to use a BGA package, consider using an underfill material if the product will be subjected to lots of vibration, shock, or moisture. Underfill is an epoxy material that is flooded underneath BGA components and then cured, resulting in a more solid mechanical anchoring to the PCB substrate than just solder balls. Loctite UF 3810 is a good choice because it's strong and has the added ability of being fairly easy to remove with heat if you need to do any rework.

BGA and similar components are good if you need to pack a lot of pins into a small area or need pins with extremely low stray inductance. The smaller the pitch (that's the distance between two adjacent pins) of a BGA part, the harder it is to route out the signals from the middle. This process is called escape routing or fanout. Very dense BGA chips require an HDI (high density interconnect) PCB to accomplish the escape routing. Things like blind and buried vias, vias in pad, a large number of layers, and extremely thin traces are the only way to get the signals out from under the chip. These processes are expensive, which is another reason why you should avoid BGA if you can. See the "Layout Design" chapter for more information on how to design with BGA if you need to.

If you're going to be hand assembling your design, you can make it easier on yourself if you choose IC packages with exposed pins. This will make it easy to solder with an iron and any solder shorts will be obvious. The downside is that exposed pins take up more space. ICs with pins underneath the chip (like QFN) take up less space, but require hot air soldering. If you've never used a hot air soldering iron, it's really not that difficult and is a totally viable method of hand assembly with a little practice.

Be cautious about trusting inherited hardware or compo-

nents. Maybe you have some leftover reels of parts from a similar project, or another engineer tells you about a part that has worked well before, or there's an entire module you want to reuse. The danger here is that the requirements may not be exactly the same. For example, if the operating temperature requirements have changed, a component or module that worked great before may now fail. There's certainly nothing wrong with using inherited hardware, and in fact it can save a lot of time and money, but don't assume it will still work. Take the time to ensure it meets all of your new requirements before you integrate it.

2.2 Passive Components

2.2.1 Capacitors

Capacitors are present in practically every circuit, but many designers mistakenly treat all capacitors as equal. Picking capacitors correctly can improve the performance of your design and prevent insidious problems that are hard to debug.

There are many different species of capacitors available. Multi-Layer Ceramic Capacitors (MLCC) are perhaps the most common and are often used for bypassing, decoupling, and filtering. These capacitors are often just referred to as ceramic capacitors and work at voltages up to about 100V and are available up to about 50 µF. Ceramic capacitors are available with a variety of different dielectric materials. These have names like X5R, C0G, and Y5V. These codes denote the capacitor's temperature coefficient and temperature coefficient tolerance. There are two classes of ceramic capacitor: Class I and Class II.

Class I capacitors are what all other capacitors wish they could be: they have a linear temperature coefficient, capacitance stays the same across voltage, they're low loss, have high Q (and thus are suitable for high frequency use) and have high stability and accuracy. Of course, there's a catch: Class I capacitors are only available in low capacitance values. You're not going to find a 100 µF Class I capacitor. In fact, you're going to have a hard time finding anything above 1 µF, and even 0.1 µF is going to be physically large. The most common Class I capacitors you'll

encounter are NP0/C0G. There are lots of other types of Class I capacitors available, but I have gone my entire career without ever using them.

If you need higher capacitance in a small volume, you need to use a higher dielectric constant material. And that means going to a Class II capacitor. That also means compromising on all of the wonderful properties of Class I capacitors and introducing new fun non-idealities, like a piezoelectric effect. But this is ok as long as you use Class II capacitors correctly. For most purposes, the only Class II capacitors you should use in your design are X5R and X7R. There are other dielectrics available like Y5R, but all of them have worse tolerance, worse performance over temperature, and cost about the same.

Class II ceramic capacitors must be derated for your design. A very common novice designer mistake is to determine that you need, say, a 10V and 10 µF capacitor and then go to your distributor of choice and pick out a capacitor rated for 10V and 10 µF. This can lead to trouble for a couple of reasons. First of all, it's likely that the voltage on that capacitor when it's in your circuit is going to exceed 10V some of the time. There will be transients and voltage spikes and dips that may only last milliseconds, but still affect the performance of the capacitor. The second problem, mentioned earlier, is that as you increase the voltage on a Class II ceramic capacitor, its effective capacitance goes down. A good capacitor datasheet will include a voltage versus capacitance curve (although not all datasheets will contain this curve). What this will tell you is that sometimes even a capacitor that says it is 10 µF and 10V will not actually give you 10 µF *at* 10V! It might be more like 4 µF. The plot below shows some example voltage versus capacitance curves. Don't use this plot to choose parts; use the charts in the datasheet of the components you're considering.

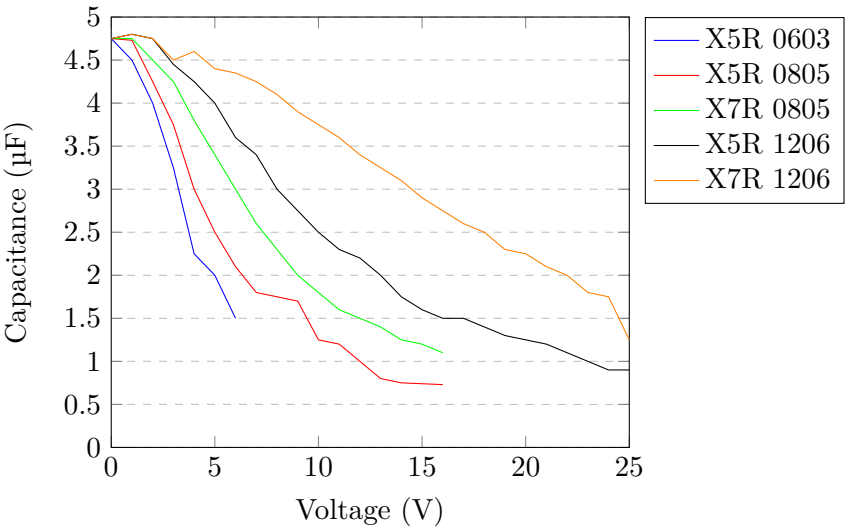

Figure 2.4: Example voltage versus capacitance

This voltage versus capacitance curve gets worse as the package size of the component gets smaller. In other words, an 0402 capacitor will start losing capacitance with increasing voltage faster than a 1206 capacitor will. So, if you're having trouble finding a part with a good enough capacitance at your voltage, try looking at a physically larger package.

This is why you should always pick a capacitor with a higher voltage rating than what you designed for. Table 2.5 shows the capacitor derating recommendations used by NASA[1]. Feel free to use parts that exceed these derating factors. More detailed derating information and modeling is available in the US military handbook Mil-Handbook-217F, "Reliability Prediction of Electronic Equipment".

[1]Taken from MIL-STD-975, "NASA Standard Parts List"

Type	Derating factor	Max. Ambient Temperature
Ceramic	60%	110°C
Tantalum	50%	70°C
Plastic film	60%	85°C

Table 2.5: Capacitor Derating Recommendations

Aluminum electrolytic capacitors (sometimes referred to as just electrolytic capacitors) are usually used in circuits where a large amount of capacitance is needed. Electrolytic caps range from about 0.1 µF to several farads. However, they're physically much bigger than other types of capacitors. Electrolytic caps are also available at higher voltages than other kinds of capacitors. So, if you need a 1 µF capacitor that is rated for 400V, you won't be able to find an MLCC that goes above about 100V, but you will be able to find electrolytic caps that can meet those requirements. Unlike some other capacitors, electrolytic caps are polarized. If you power them in the reverse polarity, the electrolyte inside the capacitor will heat up, vaporize, and possibly cause the capacitor to explode. A blown capacitor can spew electrolyte everywhere, which can cause shorts on other parts of your board. Be very careful about installing electrolytic capacitors the right way. Silkscreen on the PCB that indicates capacitor polarity can help avoid those kinds of mistakes. There's more information about working with polarized parts in the "Fabrication and Assembly" chapter.

Unlike other capacitor types, electrolytic capacitors have a shelf life. This shelf life is usually 2-3 years, assuming the capacitors have been stored correctly. Check with the manufacturer for the shelf life for specific part numbers. It's possible to "reform" electrolytic capacitors if they've been stored for more than 2-3 years to get them back to near their original performance. The US military has a handbook on how to do this called MIL-HDBK-1131.

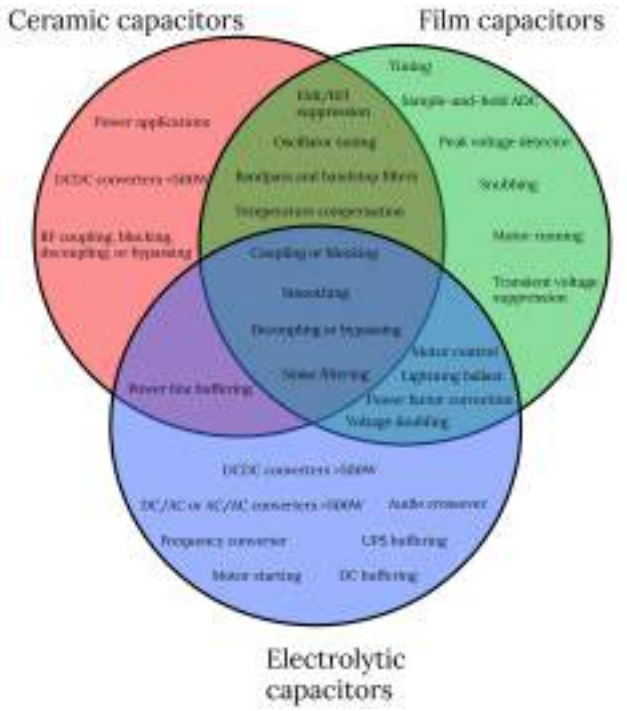

Figure 2.5: Applications of different capacitor types

Film capacitors are great for applications where you need low ESR (equivalent series resistance) and ESL (equivalent series inductance). They also have great temperature stability and aren't polarized. While they can handle higher current surges than electrolytic capacitors, they're also more expensive and physically bigger than electrolytic caps. You'll have a hard time finding a lot of film capacitors in an SMT package. Good applications for film capacitors include circuits that require a strong frequency stability or high Q, like resonant circuits. They're also great for applications that require handling high voltage surges, like snubbers. If you need a film capacitor with a higher temperature stability and are willing to pay more for it, look for polypropylene film capacitors rather than the standard polyester film caps.

Supercapacitors are a relatively recent entry to the capacitor market. The most common kind are electric double layer capaci-

tors (ELDCs). These caps have very low voltages (around 2.7V), but extremely high capacitances, up to hundreds of farads. ELDCs are polarized, but reverse biasing *symmetric* ELDCs doesn't cause them to fail catastrophically. It does cause them to degrade and should be avoided. ELDCs that are marked *asymmetric* cannot be safely reverse biased. While all ELDCs have a large capacitance, their low voltage means that the energy stored in them is not as huge as you might initially think. Many applications that use ELDCs could also use a small lithium battery, but use a supercapacitor instead because of the increased lifetime and possibly the higher current surge capability. Supercapacitors also have a very low self-discharge rate because they have such a low internal resistance. For this reason, supercapacitors are often seen in circuits that require a small battery backup to keep a clock running or volatile memory alive while the system is powered off.

Be very cautious with tantalum capacitors. While they do maintain their capacitance over voltage, tantalum capacitors can fail catastrophically if exposed to a voltage above their rating. This means it's necessary to derate them well below their advertised voltage. Tantalum capacitors are mostly used in applications where you need a large amount of capacitance in a small volume. If you must use tantalum, use polymer tantalum, not manganese dioxide tantalum. Polymer tantalum capacitors will still fail in an overvoltage condition, but they don't explode. Tantalum capacitors should also be sourced from a Tier One supplier with a documented, conflict-free supply chain. Tantalum is considered a conflict mineral and it's important to only use components that have ethically sourced materials.

An important curve in a capacitor datasheet is the capacitance versus frequency curve. There's no such thing as an ideal capacitor, which means as you start cranking up the frequency of a signal into a capacitor, it will start looking less like a capacitor and more like an inductor. The point at which a capacitor looks more like an inductor than a capacitor is its self-resonant frequency (SRF). It's important to make sure you're using all capacitors below their SRF. The plot in figure 2.6 shows an example of the frequency response of a couple of capacitors. The

SRF is the lowest point of each plot. All points to the left look capacitive and all points to the right look inductive. You can see how the size and rating of the capacitor changes the location of the SRF.

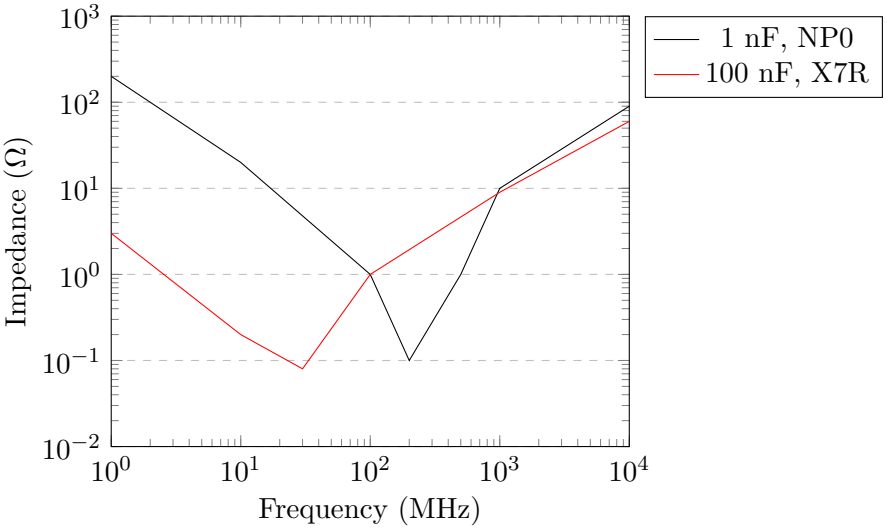

Figure 2.6: Example plots of capacitor self-resonant frequency

High-speed and RF signals require high Q (quality factor) capacitors. Capacitor manufacturers have specific product lines for high frequency use. If you use a regular, non-high Q capacitor, bad things will happen: your filters won't have the response you expect, your amplifiers will oscillate, and you'll see very high insertion loss through your capacitors. It is critical that you choose capacitors that are rated for the frequency that you're using. The GJM capacitor series by Murata (their high Q series) has a self-resonant frequency of 16 GHz. Johanson is another good manufacturer of high Q capacitors.

If you don't do any RF engineering, you may think that you'll never encounter this problem. But many wireless radio ICs (Bluetooth, Wi-Fi, NFC, etc.) require a small matching network or a balun right before the antenna that uses discrete inductors and capacitors. Bluetooth and Wi-Fi both operate at 2.4 GHz, so if

you don't use high Q components, your radio IC that's supposed to work out of the box may not work at all. Think carefully about where the high frequency signals in your design are and make sure that you call out the right part numbers in your BOM to prevent signal integrity issues.

The parasitic inductance (also known as ESL) of a capacitor is relatively independent of the capacitance and depends mostly on the package size. As you increase the size of the package, the inductance increases. Table 2.6 lists the approximate parasitic inductance of difference package sizes of ceramic and tantalum capacitors.

Capacitor size	Inductance (pH)	Type
0603	850	Ceramic
0805	1050	Ceramic
1206	1250	Ceramic
1210	1020	Ceramic
0805	1600	Tantalum
1206	2200	Tantalum
1210	2250	Tantalum
2312	2800	Tantalum

Table 2.6: Capacitor stray inductance

Since capacitors that are specifically rated to be high Q are designed to be used at high frequencies, you don't need to worry about package inductance. It's already taken into account. As a result, you'll be hard pressed to find high Q capacitors in packages bigger than 0603.

Make sure the equivalent series resistance (ESR) of your capacitors is low. This appears in the plot in figure 2.6 as the distance above zero on the y axis. A low ESR is important to prevent power and heating losses. It also improves the response time of capacitors to voltage transients. You'll sometimes see a call out in a datasheet or application note that low ESR capacitors must be used.

Like other passive components, capacitors are commonly produced with values in discrete intervals. Those intervals can be

seen below in table 2.7. Note that capacitors sold in these intervals are typically 10% tolerance. Rounding your capacitor values to one of those listed in table 2.7 (when you can safely do it) will make it a little easier to prototype and troubleshoot, since you'll probably have those values on hand. If you buy a capacitor kit, these are usually the values that it comes with.

10 pF	100 pF	1000 pF	.010 µF	.10 µF	1.0 µF	10 µF
12 pF	120 pF	1200 pF	.012 µF	.12 µF	1.2 µF	
15 pF	150 pF	1500 pF	.015 µF	.15 µF	1.5 µF	
18 pF	180 pF	1800 pF	.018 µF	.18 µF	1.8 µF	
22 pF	220 pF	2200 pF	.022 µF	.22 µF	2.2 µF	22 µF
27 pF	270 pF	2700 pF	.027 µF	.27 µF	2.7 µF	
33 pF	330 pF	3300 pF	.033 µF	.33 µF	3.3 µF	33 µF
39 pF	390 pF	3900 pF	.039 µF	.39 µF	3.9 µF	
47 pF	470 pF	4700 pF	.047 µF	.47 µF	4.7 µF	47 µF
56 pF	560 pF	5600 pF	.056 µF	.56 µF	5.6 µF	
68 pF	680 pF	6800 pF	.068 µF	.68 µF	6.8 µF	
82 pF	820 pF	8200 pF	.082 µF	.82 µF	8.2 µF	

Table 2.7: Standard capacitor values

Some capacitors are designed to be fail safe. Tantalum capacitors can fail short if they're exposed to overvoltage conditions. To combat this, some tantalum capacitors have built in protection against this and are called either fail open or fused tantalum capacitors. These capacitors have the advantage of failing as an open circuit instead of a short circuit and can prevent further damage in your circuit. The disadvantage is that the fuse adds extra series resistance (ESR), so keep this in mind if using a fused capacitor. MLCC capacitors aren't vulnerable to catastrophic failure in an overvoltage condition like tantalum capacitors are, but they are vulnerable to internal cracking if the PCB they are soldered to is flexed. As the name implies, MLCC capacitors have multiple layers of electrodes inside them that are stacked up on top of each other with a ceramic dielectric sandwiched between each layer. Ceramic is brittle, and if it cracks due to stress, adjacent electrodes can touch. To prevent these internal

cracks from causing a short, you can use open mode or floating electrode MLCCs. These work by separating the electrodes horizontally so that a crack will be less likely to short out both sides of the capacitor.

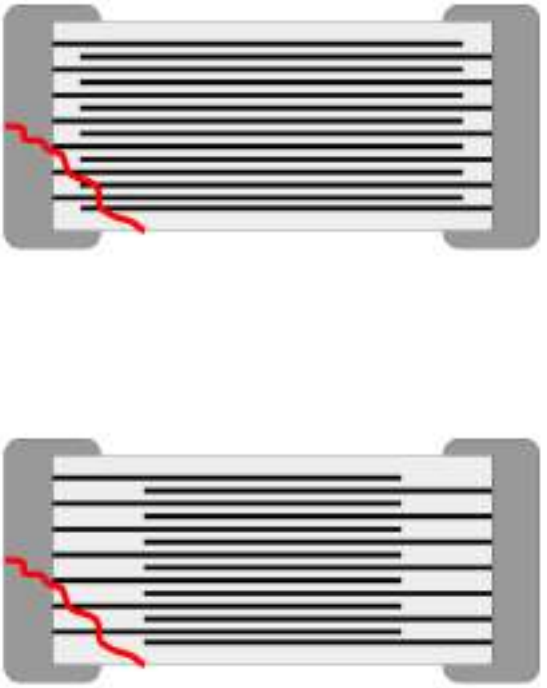

Figure 2.7: A crack in a typical MLCC (top) can cause adjacent electrodes to short together. An open mode MLCC (bottom) has a lower chance of causing a dead short when cracked because the electrodes are horizontally spaced.

For even more safety, you can use a flexible termination capacitor, which further reduces the chance of an internal short. These work by changing the way the terminals of the capacitor

are connected to the metal solder points so that stress isn't transferred to the interior of the capacitor in the first place. An open mode capacitor can withstand PCB flexing of up to about 2 mm, whereas a flexible termination capacitor can withstand flexing of up to about 5 mm. These numbers are just guidelines, so make sure to read the datasheet of any failsafe capacitor before you use it.

2.2.2 Resistors

Choose resistors that have the tolerance that your application requires. There's no need to use 1% tolerances for your pull-up resistors because the exact value there doesn't really matter. Since tolerances are specified as a percentage, the tolerance you need will change based on the value of the resistor you're selecting. For example, a very small value current sense resistor may be acceptable if it's a 0.5%, 1%, or 5% tolerance simply because the value of that resistor is so small that even 5% is low enough to not matter.

Resistors change their resistance based on what temperature they are. Your sensitive resistors shouldn't ever get hot enough for this to matter, but it's important to keep in mind. Another important thing to remember is that different kinds of resistors have different amounts of thermal noise. This is especially important when you're using a resistor as a sensor. The rule of thumb is that thick film resistors have higher thermal noise than thin film resistors, and physically larger resistors have higher thermal noise than physically smaller resistors.

If a resistor will be carrying a large amount of current or a high voltage, calculate the power that will be dissipated through it. NASA recommends derating resistors to 60% of their rated power[2]. Make sure that any heat released won't affect any other part of your circuit and is dissipated correctly. Resistors that can handle high power are physically large and sometimes require a heat sink. Plan accordingly. It's best to try to design your circuit such that you don't need to use high power resistors that

[2]Taken from MIL-STD-975, "NASA Standard Parts List"

are burning up power because it's wasteful and the resistors are expensive.

When picking resistors for a voltage divider or for pulling up or pulling down a signal, consider the current leakage through that resistor. Using a resistor value between 10k and 100k will keep your design from drawing unnecessary power. If you use resistor values that are too large, the current to your load will be too small. If you use values that are too small, the divider will consume more current than needed, burning it all up as heat. Instead, calculate the optimal resistor values that will meet your minimum current and voltage requirements.

Don't use a resistor divider to power a circuit directly. Instead, use a more efficient method that can actually regulate the output voltage like an LDO or a switched mode power supply. It's also not a great idea to use a voltage divider to convert logic levels. It's possible to use a voltage divider to convert a 5V signal to a 3.3V signal, for example, but there are a couple of reasons why you shouldn't do this. First of all, for any signals with fast edges or a high frequency, the stray inductance and capacitance of the resistors will wreak havoc on your signal integrity. A resistor divider won't work for a bidirectional signal, because it can only drop voltage. Your 5V signal can be seen by your 3.3V device, but your 3.3V signal may not be seen by your 5V device. However, for slow signals, like enable/disable lines, a voltage divider will work for converting higher voltage signals to lower voltage signals (again, only in one direction).

Zero ohm resistors are often used as jumpers, test points, or as a way to short circuit another series component. However, zero ohm resistors can sometimes have an actual value of as high as one ohm. If you need something very close to zero ohms, it's better to buy a low value, high tolerance part like a 0.001 ohm 1% resistor.

If you're doing RF work and looking for a 50 ohm resistor, you will find that there are startlingly few options available, and they are all very expensive. Instead, look for 49.9 ohm resistors. The difference of 0.1 ohms is negligible and many manufacturers make a 49.9 ohm resistor but not a 50 ohm resistor. This is because 49.9 appears in the E96 standard value series that manufacturers

use, but 50 does not. A 50 ohm resistor is technically a special value, and so manufacturers will charge much more for them.

As in the case of capacitors, resistors are available in standard values. The intervals can be seen below in table 2.8. Note that resistors sold in these intervals are typically 5% tolerance.

1.0	10	100	1.0K	10K	100K	1.0M
1.1	11	110	1.1K	11K	110K	1.1M
1.2	12	120	1.2K	12K	120K	1.2M
1.3	13	130	1.3K	13K	130K	1.3M
1.5	15	150	1.5K	15K	150K	1.5M
1.6	16	160	1.6K	16K	160K	1.6M
1.8	18	180	1.8K	18K	180K	1.8M
2.0	20	200	2.0K	20K	200K	2.0M
2.2	22	220	2.2K	22K	220K	2.2M
2.4	24	240	2.4K	24K	240K	2.4M
2.7	27	270	2.7K	27K	270K	2.7M
3.0	30	300	3.0K	30K	300K	3.0M
3.3	33	330	3.3K	33K	330K	3.3M
3.6	36	360	3.6K	36K	360K	3.6M
3.9	39	390	3.9K	39K	390K	3.9M
4.3	43	430	4.3K	43K	430K	4.3M
4.7	47	470	4.7K	47K	470K	4.7M
5.1	51	510	5.1K	51K	510K	5.1M
5.6	56	560	5.6K	56K	560K	5.6M
6.2	62	620	6.2K	62K	620K	6.2M
6.8	68	680	6.8K	68K	680K	6.8M
7.5	75	750	7.5K	75K	750K	7.5M
8.2	82	820	8.2K	82K	820K	8.2M
9.1	91	910	9.1K	91K	910K	9.1M

Table 2.8: Standard resistor values

2.2.3 Inductors

NASA recommends derating inductors to 50% of their rated voltage and 60% of their rated temperature[3]. Like capacitors, induc-

[3]Taken from MIL-STD-975, "NASA Standard Parts List"

tors should have a low ESR and must be operated below their SRF. Check the curves in the datasheet (if they're present). In figure 2.8, the SRF of the example inductor is the peak in that curve. At all frequencies lower than that, the inductor looks mostly like an inductor and at all frequencies higher than that, it looks like a capacitor.

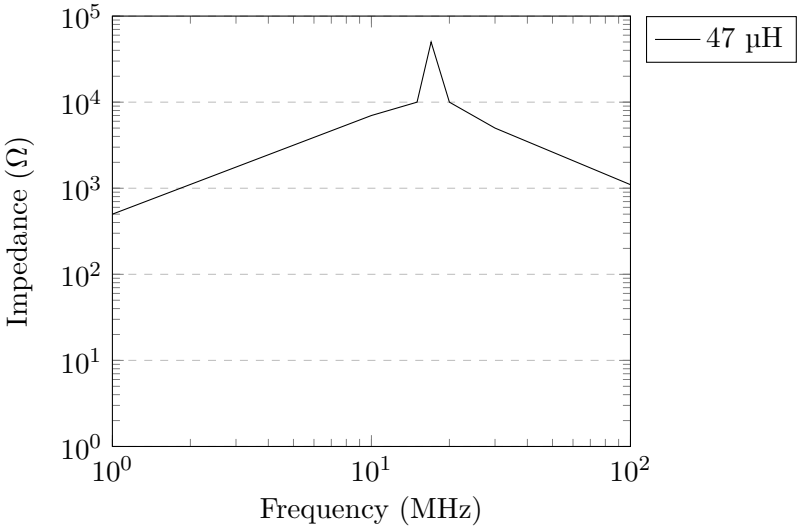

Figure 2.8: Example inductor self-resonant frequency

Inductors are often used in switched mode power supplies (SMPS). An inductor in an SMPS will see lots of time varying currents as the FET switches. This can cause electromagnetic interference (EMI) and electromagnetic compatibility (EMC) problems during compliance testing. Use a shielded inductor to help prevent interference with other components in your design and to help pass EMC testing. A shielded inductor incorporates material into the package that contains the fields produced during switching.

Like capacitors, inductors that will be used for high frequency applications must be high Q and rated for high frequency use. It's critical to choose high Q inductors for the parts of your circuit that are high frequency or nothing will have the response you

expect and your amplifiers will oscillate. Air core spring-type inductors usually have a higher Q than ferrite core inductors.

2.2.4 Ferrite beads

If an inductor has a very poor Q, it becomes a new component: the ferrite bead. Above a certain frequency, ferrite beads behave like an inductor that is bad at storing energy in its magnetic field and instead dissipates most of the energy as heat. This is useful when you want to filter out high frequency from a signal.

But what's the best way to choose a ferrite bead? There are some ferrite beads that are designed for specific applications, like USB. In that case, use the ferrite beads that are application specific. For most applications though, the rule of thumb is to pick a ferrite bead that will filter out everything past the 5th harmonic of the signal of interest. The reason for this is that if you filter out lower order harmonics, you will start to distort your signal. The 5th harmonic is a good balance that will cause a square wave to have minimal distortion but still remove higher frequency signals. For example, if your signal of interest is 1 MHz, use a ferrite bead that looks largely resistive starting at about 5 MHz.

When shopping for a ferrite bead on an electronics distributor's site, you'll see high level specifications like "120 ohms at 100 MHz". Use those specifications only as a very rough guide to what the frequency response of the particular bead looks like. The only way to correctly pick a bead is to look at the frequency response plot in the datasheet and see where resistance starts dominating inductance. If you use a ferrite bead in its inductive region, you'll get ringing as it couples with parallel capacitance in your circuit, potentially making the problem worse. The specifications visible in a distributor's search results can often mislead you into picking a part with an inductive region that ends at a higher frequency than you think.

The plot in figure 2.9 is an example of the impedance of a ferrite bead over frequency. This particular bead would be useful for suppressing noise in the range of about 180 MHz to 500 MHz. Below 180 MHz, the bead looks mostly inductive, and above

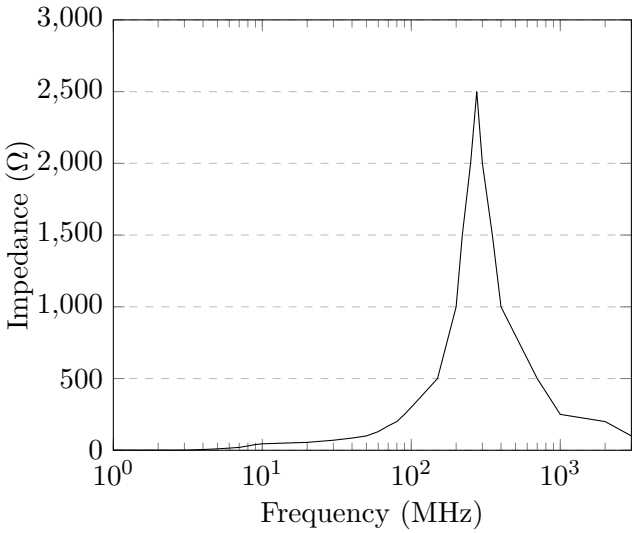

Figure 2.9: Example ferrite bead frequency response

500 MHz, it looks mostly capacitive. Within the band of 180 MHz to 500 MHz, it looks largely resistive, so it will dissipate energy at those frequencies as heat rather than storing them as an electric or magnetic field. That's exactly what we want. For this particular bead, since we want to filter signals past the fifth harmonic of the signal of interest, the fundamental signals that this bead could be used for range from 36 MHz to 100 MHz (that's $\frac{180MHz}{5}$ and $\frac{500MHz}{5}$).

Ferrite beads can be rated for power or for signals. This is essentially a measure of their DC resistance and power dissipation ability. This is more important in the case when you're filtering power since using a ferrite bead that is only rated for signals might fail or cause degraded performance. As with inductors and capacitors, the equivalent series resistance (ESR) needs to be sufficiently small. Ferrite beads are often used in series with power supply pins, so a high ESR can cause an undervoltage on the device being powered. When deciding on the maximum allowable ESR for a ferrite bead, use the maximum current that can flow through the bead, not the nominal current. If an IC

nominally draws 100 mA but can surge up to 200 mA, calculate the voltage drop across the ferrite bead with 200 mA of current, not 100 mA. Otherwise, your IC will be starved for voltage during current surges.

Ferrite beads have another "feature" that you need to watch out for if your device needs to operate at high temperatures. The maximum current through a ferrite bead drops off dramatically once the temperature begins to rise. Obviously, this will be different for every bead, so look at the current versus temperature curve to ensure that you won't exceed the maximum current of the bead at the maximum temperature of your design specification.

The amount of current through a ferrite bead will also affect the frequency response. The higher the current through the bead, the further up in frequency the impedance plot will shift and the lower the maximum impedance will be. So, a ferrite bead with 100 mA of DC current through it will have a much lower maximum impedance than the exact same bead with 200 mA of DC current running through it. Additionally, this maximum impedance point will be at a higher frequency when running 200 mA through the bead instead of 100 mA. Again, check the plots in the datasheet for exact values. Beware that a lot of plots in the datasheets will show impedance with zero DC bias current, a situation which will never happen in real life! In general, it's a good idea to derate the maximum current limit in a ferrite bead by a factor of 5. You should basically ignore the advertised maximum current rating and the resistance rating (e.g., 120 ohms @ 100 MHz) of ferrite beads and look at the plots in the datasheet to find the curves for the DC bias current you will be applying before choosing a part. Small changes in current can alter the performance of the bead by almost an order of magnitude in some cases.

This whole process may seem like it's hard or complicated to get right, but it's usually not. As long as you pick your bead correctly, any oscillation will most likely be damped by parasitic resistance. If you're unsure or suspect you may have problems with oscillation, run a quick simulation or take a measurement. Trial and error with a couple of different values is very common.

If you're using a ferrite bead in your design, be sure to read the section later in this book about schematic design and layout with ferrite beads to avoid other problems.

2.2.5 Connectors and cables

You may be surprised to find an entire section of this book dedicated to connectors and cables, but they're both components that can have as much of an impact on your design as any other component. Choosing the incorrect cable or connector, or placing either of them poorly can cause your design to fail emissions tests, conduct noise into your circuit, or even cause a safety issue.

Choose high quality cables and connectors. They are often one of the first things to fail. They are also expensive. Before you decide that you can get away with the cheapest Chinese cable you can find, order some and test them yourself. Cables, connectors, and electromechanical parts like switches and knobs should be qualified before you ship with them. What you can get away with will depend on your application. For example, if the cable will be plugged in once and then hidden away inside the enclosure, your qualification requirements will be different than a battery connector that will be plugged and unplugged hundreds or thousands of times by the user.

Make sure any connectors you use that will carry power are rated to carry that power. NASA recommends derating the advertised maximum connector current by 50%[4]. If you need a physically smaller connector, you can connect a single power net to multiple pins of a connector to reduce the current in any single wire.

Make sure edge mount connectors like end launch SMA connectors are selected to fit the right board thickness. Most PCBs are 62 mils thick by default, but you can find edge mount connectors that fit several different board thicknesses. If you want to use this style of connector, it's best to try to keep your PCB thickness to the standard 62 mils. If you use a stackup with a different thickness, it may be difficult to find a connector that

[4]Taken from MIL-STD-975, "NASA Standard Parts List"

you can use, since they're only manufactured for a few discrete sizes.

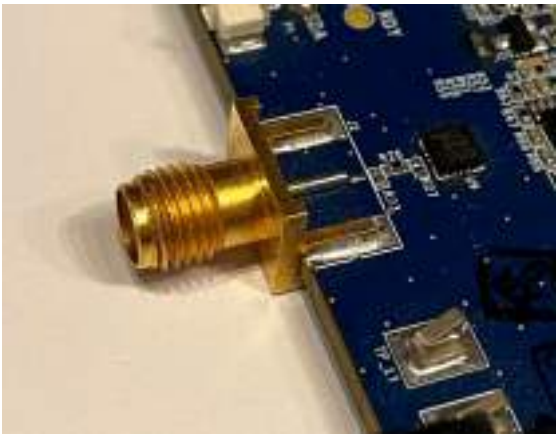

Figure 2.10: An edge mount SMA connector.

Pick your connectors to withstand the maximum temperature they will experience during reflow (if they will experience it), test, and normal use. Connectors should also be keyed so that it's not possible to plug anything into them the wrong way. It can be helpful to order some samples of connectors to test fit. It's sometimes possible to partially plug in even a keyed connector the wrong way and cause a short, arc, or otherwise damage your device. Playing with samples will help you learn things like that, which you wouldn't find in the datasheet. Connectors that have a physical clicking or snapping sensation when successfully mated should be preferred, since that will give you and your assemblers positive affirmation that the connectors are actually mated.

If you have a differential pair going over a cable, consider using twisted pair. Using twisted pair will reduce the electromagnetic emissions caused by the signals and will also reduce the susceptibility of those signals to external noise. Twisted pair also helps prevent crosstalk from other signals sharing the same cable. If you have multiple twisted pairs in the same cable assembly, use different twist rates on each pair. If two twisted pairs have the same twist rate and lie next to one another, the benefits

of twisted pair are somewhat degraded, because each conductor is spending the same amount of time next to its corresponding conductor in the other twisted pair, so some crosstalk can still occur. You can buy assembled cables that do this for you, the most common being ethernet.

Another way to reduce external noise and susceptibility in a cable is to use a shielded cable. It's best to use a pre-assembled shielded cable rather than building one. The shield needs to be grounded at both ends to act as an effective shield for an electric field, but it only needs to be grounded at one end to be an effective shield for a magnetic field. For more information about reducing EMI/EMC in cables, see the "EMI" section in the "Layout" chapter.

If you need an impedance-controlled cable, use coaxial cable (aka coax). There are lots of different kinds of coax with different losses, impedances, and dielectric materials. The most common coax cable to use for 50 ohm applications is RG-58/U, and the most common for 75 ohm applications is RG-59/U. Make sure you pick a cable type that can handle the frequency, power, and impedance of your application.

Certain types of coax use braided outer conductors, which can cause signal to leak out. This can be a problem if you have multiple coax cables bundled together. As an alternative, look for cables that use a foil over the braid, or that use a solid outer conductor. RG-223, rigid, and semi-rigid coax are all good options, but be aware that rigid and semi-rigid cables will be significantly less flexible and more expensive.

Not all connectors can handle all frequencies. For most RF applications, SMA connectors are fine. If you're doing work above about 18 GHz, you'll need more exotic (and unfortunately, more expensive) connectors. Avoid building your own RF cables if at all possible, especially for high frequency designs. It's easy to build a poor quality cable if you don't have experience and there is nothing more frustrating than debugging an insidious problem that ends up being a bad cable (trust me). Table 2.9 lists many different RF connectors and their maximum usable frequency. Use this table only as a reference: check the datasheet for the specific connector you use to get the most accurate maximum

frequency.

Connector name	Max frequency
Twinax	200 MHz
UHF	300 MHZ
Triax	300 MHz
MHV	500 MHz
Mini-BNC	1 GHz
F-Type	1 GHz
Mini-SMB	2 GHz
Mini-UHF	2.5 GHz
G-Type	3 GHz
L	3 GHz
SMZ	4 GHz
SMB	4 GHz
HN-Type	4 GHz
FAKRA	4 GHz
BNC	4-12 GHz
QMA	6 GHz
MMCX	6 GHz
MCX/OSX/PCX	6 GHz
HD-BNC	6 GHz
AMMC	6 GHz
AMC	6 GHz
AFI	6 GHz
OSMT	6 GHz
7/16	7.5 GHz
SC	8 GHz
C	8 GHz
SSMC	10 GHz
SMC	7-10 GHz
1.0/2.3	10 GHz
TNC	11 GHz
SC-Type	11 GHz
QN-Type	11 GHz
N-Type	11 GHz
C-Type	11 GHz

SSMA	12.4 GHz
SMA	18 GHz
APC-7 / 7 mm	18 GHz
Precision N	18 GHz
Precision TNC	18 GHz
OSP	22 GHz
GMS	26.5 GHz
OSSP	28 GHz
3.5 mm	34 GHz
SSMA	38 GHz
SMP	40 GHz
2.92 mm	40 GHz
GPO/OSMP/SMP	40 GHZ
K-Type	40 GHz
OS-50P	40 GHz
Precision (APC)	50 GHz
2.4 mm	50 GHz
V-Type	65 GHz
1.85 mm	65 GHz
SMPS	65 GHz
1.0 mm launch	110 GHz
0.9 mm	120 GHz
0.8 mm	145 GHz

Table 2.9: RF connectors and their maximum frequencies

When selecting RF connectors, prefer connectors that screw together rather than snap together. They're more reliable and you can use a torque wrench to ensure that they're firmly attached (but not over-tightened). When you're screwing in connectors, always rotate only the nut instead of twisting the mating connector or cable.

Check the price of both sides of the connector too. For example, an MCX female connector may only cost a couple dollars, but surface mount MCX male connectors are very hard to find and about ten times as expensive. That's because most applications use a female connector on both sides and a cable with a

male connector on each end.

Not all cables can handle all environments. For example, if you need an RF connector that will work outdoors, an N-type connector is a better choice than an SMA connector. Think about where the cable will be sitting (inside an enclosure, buried under the ground, underwater, etc.) and the environmental stresses it will undergo (heating from the sun, rain, ice, etc.). Using a cable not rated for the environment where it will be used can cause signal integrity problems or a catastrophic failure of the cable resulting in short or open circuits. Environmentally rated cables will often be thicker and less flexible than other cables.

Consider adding a strain relief feature to your enclosure if a cable will be external to the device. It will inevitably be yanked by the user and can accelerate wear around the crimps on connector pins or even cause wire to fatigue and break. Whether you need to add strain relief will depend on the physical width and length of the cable, its placement, the gauge of wire, cable assembly, etc.

All connectors and interconnects should make the connection with ground first and break connection with ground last. This is important for safety, especially when dealing with high voltages. Most connector manufacturers offer products that make ground first and break it last. The way they do this is by making the ground conductor longer than the other conductors so that it touches the mating connector first when inserting the connector and is the last conductor to be pulled out of the mating connector.

Another safety feature that should be considered is exposed vs. recessed pins and contacts. A high voltage connection should not have exposed pins that can be touched by the user or accidentally shorted against anything (necklaces, rings, keys, coins, etc.). Most connectors that are meant to be used at high voltages will take care of this for you, especially if they are UL or IEC certified.

High voltage and high current cables and connectors need to be safe. There are several standards and certifications that cables and connectors can be compliant with to assure safety. The difference between these certified parts and standard parts is mostly in the amount and type of insulation used. Do not pick

cables and connectors so that they fit in your enclosure. Instead, pick cables and connectors that are rated for the current and voltage you'll be putting through them and THEN design them into the enclosure.

Beware of connectors that have similar names but are different. For example, MCX and MMCX are different and will not mate together. MMCX is smaller, but looks otherwise identical to MCX. It's easy to order the wrong part or grab the wrong one if there's a pile of them mixed together. Another classic example is SMA. SMA connectors are ubiquitous in RF engineering, but they are not the same as RP-SMA. The RP stands for Reverse Polarity. Confusing SMA and RP-SMA is even more insidious because they will easily screw together and look connected, but aren't. The reverse polarity aspect of RP-SMA means that screwing it into an SMA connector will result in either two receptacles next to each other or two pins next to each other. In both cases, the connection looks like an open circuit. In RF applications, that will cause a large reflection, a large VSWR, and can destroy components. RP-SMA is most commonly used in Wi-Fi equipment like routers. So, if you unscrew the antenna from the back of your wireless router and screw it into your spectrum analyzer, the spectrum analyzer won't actually be connected to the antenna. It's a good idea to get a couple of RP-SMA to SMA adapters (you need both versions: hole to hole and pin to pin). They're cheap and can save you in a pinch.

One more set of connectors that are easy to get mixed up are U.FL connectors. U.FL is a trademarked name by Hirose, the manufacturer that invented it. Hirose has several size variants of U.FL including E.FL and W.FL. Because other companies legally can't call their connectors U.FL, they have to call it something different. Other names used include IPEX, IPAX, IPX, AMC, MHF and UMCC. All of these will mate with a U.FL connector, but there are lots of other tiny RF connectors that look very similar and do *not* mate with U.FL. They're all very small, look about the same, and you'll only realize they won't fit when you try to actually plug it in. To help avoid this problem, buy both mating sides of the connection from the same manufacturer. Connectors bought from one manufacturer will usually list mating connectors

in the datasheet, removing the risk of a connection that doesn't quite fit right.

Even if two connectors are supposed to be able to mate to each other, it's a good idea to buy both sides of the connector pair from the same manufacturer so they mate with the right tolerances. If you buy an MCX male connector from one company and an MCX female connector from a different company, they'll mate but the connection won't be a solid as if you bought both sides from the same manufacturer. The connection will still work, but since the tolerance stackup of a connector is designed with a particular mate in mind (namely the corresponding one made by the same manufacturer), you're more likely to have intermittent failures or weak connections. Using two manufacturers will likely result in a connection that doesn't meet all of the claimed specifications in either datasheet.

If you can avoid using a connector at all, do it. Connectors and cables add cost and can cause EMI and EMC problems. One way to do this is to use hot bar soldering. Lots of cheap consumer products do this to attach long ribbon cables to the PCB without requiring connectors. The stripped ends of the cable are laid down on top of pads on the PCB and a long, hot bar is laid across the cable and PCB pads so that it heats up the entire length of one end of the cable, soldering the whole cable at once. This is frequently used to save cost during large production runs. Another way to connect a large number of wires without connectors is to simply have an assembler solder each wire manually. This takes longer, but some production products do it.

Another less common way to reduce your connector use is to use card edge connectors. This simplifies the need for two connectors and a cable assembly down to a single connector that slides into exposed contacts on a mating PCB. Card edge connectors are commonly used in modular devices and equipment. They're fine for power, ground, and slow signals, but should not be used for anything that has fast edges, is high-speed, or RF unless you use a card edge connector specifically design for that. A card edge connector may also require some extra mechanical engineering or a specific PCB thickness to fit correctly.

Flat Flex Cables (FFC) can be very useful for low profile board to board connections. One trick that works well is to get an FFC that has conductors on only one side, and a receptacle that has contacts on both sides. This way you can position the receptacles on either the top or bottom of either PCB. It also lets you bend and crease the FFC without having to change the side of the PCB that the receptacle is on.

Crimped Connectors

Many people like to avoid crimped connectors because you have to buy the parts and a crimping tool and then take the time to build the cables (which, if done incorrectly, results in a fragile cable). They'll say it's much faster and easier to just solder wires without using any connectors. But in some cases, using crimped connectors can actually make things easier in the long run, especially in the prototype stage.

The advantage of crimped connectors over soldered connectors or even just soldering wires straight to pads is mechanical reliability. When you solder stranded core wire, the solder will wick up the wire underneath the insulation. This creates a weak point where the now rigid, solder containing part of the wire meets the wire without any solder and causes the strands to break under repeated strain or movement. This effect also occurs with solid core wire, but less so. This can be enormously frustrating, especially when the wire breaks underneath the insulation, making it difficult to find. A crimped connection gets around this problem by using no solder and only the pressure of the bent metal fingers of the crimped terminal to make contact with the wire. Correctly executed, a crimped connection will last much longer than a soldered connection and is more robust to bending, plugging and unplugging, and strain. So, while soldering a wire to a pad may save you a little time building your prototype now, using a crimped connector will help make your prototype more robust and make it easier to take apart, repair, and reassemble things without having to use a soldering iron every time.

Crimping pins are available with several different types of surface plating. The most common plating options you'll see are

tin, gold, and palladium nickel. Always use gold or palladium nickel plated contacts for the most reliable, long lasting connections. If you can't use those because of cost, you can use tin plated contacts, but you need to perform thermal and vibration testing on them first. Tin plated contacts can corrode, have a higher resistance, and can cause tin whiskers (more on that in the DFx section in the "Layout Design" chapter). Never mate a gold contact to a tin contact. This is known to cause corrosion and reliability problems. Mate gold to gold, tin to tin, palladium nickel to palladium nickel, but never gold to tin.

If you can spend the money, buying pre-made cables will save you time and guarantee reliability. If you decide to build your own cables, use crimping pins and housings that have both been sourced from a legitimate manufacturer. The risk with buying off-brand crimp connector parts from different sources is that they are not always compatible with each other. If you buy a housing from Molex but the crimp pins from a Chinese eBay supplier, there's a good chance they're not going to fit together very well, or even at all.

Building cables with crimped connectors requires a crimping tool. For every family of crimped connector, there is a tool made by the manufacturer that is designed to perfectly crimp the pins. Sometimes, the official tool is reasonably priced, and sometimes it can cost over $1000. You can also buy cheaper generic versions of crimping tools, some of which work well, some of which do not. A classic "gotcha" that connector manufacturers like to play is to make the connector housing and pins cheap but the crimping tool very expensive. This matters less in a big company and more for the hobbyist or budget constrained startup. Always check the price of the crimping tool as well as the connector parts if you're building your own cables.

When buying crimping tools, you should prefer to buy the official manufacturer tool over generic tools. The official tool will produce much more consistent and reliable crimps. If you can't afford the official tool new, try checking eBay for used ones. You can sometimes find them significantly cheaper there, but they're not always available. Another place to check is crimptools.com, which sells lots of used crimpers. Crimping tools typically only

work with a single connector family, so if you want to use two different connector types, you'll need two different crimping tools. If you want to save money by using a generic or knock-off crimping tool, make sure to carefully follow the crimping instructions provided by the connector manufacturer. These instructions will include detailed measurements and drawings showing what a perfectly crimped connection looks like, which will help you identify and avoid issues with a generic crimping tool. Another great resource is the Molex Quality Crimping Handbook[5]. This handbook contains detailed information on correct crimping technique, how to test and troubleshoot crimps, and lots of other very practical tips that apply to all connector families, not just Molex.

You can use the table below to compare the cost and efficacy of different crimping tools for common connectors. This is certainly not an exhaustive list, but is a good place to start. You'll notice that there are several generic crimpers that work for multiple types of connectors, but they don't always work equally well on all of them. If you search Amazon or eBay for crimpers, you'll likely find several offerings not listed here that look similar but are made by different Chinese companies. These brands come and go, and it's hard to know whether they'll work well or not. It may be worth it to take the risk and bet your $20 that you can get a good enough result if spending $500 on the official tool isn't an option. Also be aware that sometimes different wire gauges require different tools even if they're in the same connector family. So, before you buy a crimping tool, check to make sure it can crimp the wire gauge you want.

[5] You can find a link to this handbook on this book's website, DesigningElectronics.com

Connector	Pitch	Crimping tool	Cost (in 2021)	Tool Origin	Recommended?
Mini-PV, DuPont	2.54 mm	HT-95 / HT-0095	$1444	Manufacturer	Yes
		HT-213A	$300	Manufacturer	Yes
		HT-208A	$300	Manufacturer	Yes
		Engineer PA-09	$42	Generic	No
		Hozan P-707	$80	Generic	Yes
		YTH-202B	$25	Generic	Yes
		IWISS SN-01BM	$23	Generic	Yes
JST, PH	2 mm	WC-240	$490	Manufacturer	Yes
		Engineer PA-09	$42	Generic	Yes
		IWISS SN-01BM	$23	Generic	No
JST, XH	2.5 mm	WC-110	$490	Manufacturer	Yes
		Engineer PA-09	$42	Generic	No
		IWISS SN-01BM	$23	Generic	Yes
Picoblade	1.25 mm	Molex 63827-1400	$366	Manufacturer	Yes
		Molex 63827-1500	$366	Manufacturer	Yes
KK 254 / KK 100	2.54 mm	Molex 63811-8200	$363	Manufacturer	Yes
		IWISS SN-01BM	$23	Generic	No
SL	2.54 mm	Molex 63825-8800	$363	Manufacturer	Yes
		Molex 63811-8700	$363	Manufacturer	Yes
		Molex 64016-0201	$137	Manufacturer	Yes
		IWISS SN-2549	$23	Generic	Yes
KK 396 / KK .156	3.96 mm	Molex 63811-7500	$829	Manufacturer	Yes
		HT-225D	$20	Generic	No
		PA-21	$45	Generic	No

2.2.6 Circuit protection

Circuit protection isn't just important for passing certification tests. It's also an important part of building reliable hardware that can survive the harsh testing conditions and mistakes that may come up during the PCB bring up process.

Filters can be an important part of your electromagnetic compatibility (EMC) strategy. However, it's important to know that even if you choose the right filter, you can still render it useless if it is integrated into your design incorrectly. Be sure to consult the "Layout Design" chapter to ensure you get the performance out of your filters that you expect. It's also a good idea to only add electromagnetic interference (EMI) filters if you are confident that they are necessary (determined by either calculation or measurement) or if the engineer at your testing facility recommends one. Otherwise, you will just be adding unnecessary cost to your BOM. So, don't finish reading this section and start adding EMI filters everywhere unless you know you need to.

You have two choices when selecting an EMI filter: single pole or multi pole. The most common single pole filter for EMI suppression is a feedthrough capacitor. Feedthrough capacitors are most often used at transitions in the layout. If you have a critical, high-speed signal that is leaving a shielded enclosure, a feedthrough capacitor will pass the signal through the shielding and provide capacitance with very low inductance to ground. This ensures that high frequency noise can actually be bypassed to ground without seeing impedance from stray lead inductance. Feedthrough capacitors are typically bulkhead mounted parts that screw into a hole on the side of an enclosure or shield. Some connectors have built in feedthrough capacitors that make them easier to integrate.

Another form of feedthrough capacitor is a three terminal capacitor. These are usually surface mount devices that have, as the name implies, three terminals and either three or four pads. The first two terminals are for the signal input and output. The remaining terminal(s) are both connected to ground. Again, this ensures extremely low parasitic inductance on the capacitor so high frequency noise can be bypassed. Three terminal capacitors

are useful when crossing shielding boundaries on a PCB.

Multi pole filters can either be bought in a single package, or you can design them yourself. The five kinds of multipole filters you will encounter are below.

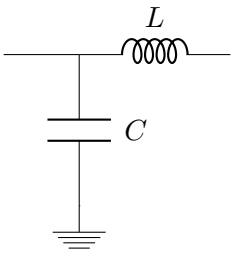

Figure 2.11: Shunt-series filter

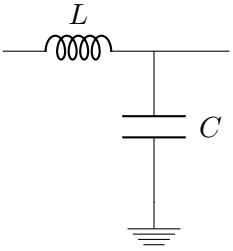

Figure 2.12: Series-shunt filter

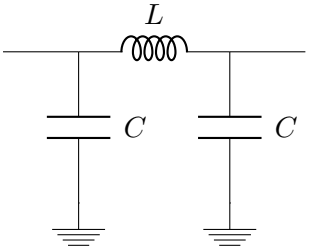

Figure 2.13: Pi filter

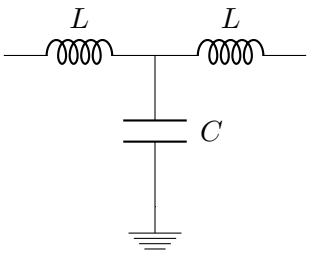

Figure 2.14: T filter

Notice that all of the topologies are low pass filters, since we're trying to filter out high frequency noise. To determine the topology you should use, first determine the impedance you need on the input and output of the filter. A shunt-series filter has a high impedance input and a low impedance output. A series-shunt filter has a low impedance input and a high impedance output. The rule of thumb to use is if the input source has an impedance of less than 100 ohms, use a series-shunt or T filter. If the input source has an impedance of greater than 100 ohms, use a shunt-series or Pi filter.

All EMI filters have an insertion loss. The reason you should care about the insertion loss at all is because it's essentially a benchmark to compare EMI filters against each other. You want high insertion loss at the frequencies you're trying to filter out and low insertion loss at the frequencies you want to keep. Like other passive components, you'll want to look at the plot of filter performance over frequency in the datasheet before selecting a part. Like ferrite beads, EMI filters should have at least a 5x frequency passband. In other words, if you're filtering a 5 MHz signal, your filter shouldn't have significant insertion loss until 25 MHz.

Electrostatic discharge (ESD) is something else you need to defend against. ESD occurs when any object that has been raised to a much higher potential than the PCB makes contact with the PCB. Your users will be a common source of ESD, as simply walking around on a carpet floor on a dry day can build up a massive potential that is equalized when they touch your

product. That means your device needs to be able to survive a very fast, very high voltage spike. A typical voltage transient might be 5 nanoseconds long and 10,000 volts. ESD diodes are specially designed diodes that have a very fast response time and can shunt that high voltage (and the resulting current) to ground. A lot of ICs have ESD protection already designed into them. When deciding where ESD diodes are necessary, think about where the device can be touched. Consider not only directly accessible conductive surfaces, but also conductive surfaces that are close enough to the exterior of the device that they can be arced to. A 10,000 V pulse can easily arc to a recessed signal pin on a connector, even if it's not possible to touch the pin directly. If you need to have ESD diodes on sensitive analog signal lines (for example, on a medical device that is making contact with a patient), use diodes with as low capacitance as possible. This will prevent excessive capacitive loading and it will help prevent noise from capacitively coupling into your sensitive signal. Polymer ESD diodes are a good option when low capacitance is required. Some standards require a certain amount of ESD protection (given as the maximum voltage spike that the device can tolerate from an ESD event). Even though your ICs may have protection already built in, you may need to add additional ESD diodes to cover the full range required. For example, many ICs only provide up to 5 kV of protection and some medical standards require 15 kV of protection. Low current traces can be protected against ESD events with a small series resistor, which simply restricts the flow of current that the discharge creates.

If you need to design to a particular standard, make sure you pay close attention to the exact ESD requirements, not just the maximum voltage you must sustain. There are different models of ESD discharge that are commonly used, like the Human Body Model, the Machine Model, and the Charged Device Model. These are all tested in different ways and they all have different output currents and peak times. Carefully read all of the ESD requirements of the standard you're following, and don't be afraid to get expert advice from an organization like Underwriter's Laboratories (UL) for help in understanding what applies to you. If you're not required to follow a particular standard but still need

ESD protection, reading through standards like IEC61000 can help you figure out what protection might be appropriate for your device.

SCR latch up is another danger to watch out for. A silicon controlled rectifier (SCR) is a semiconductor that is made by alternating P-N-P-N doped silicon. It has three terminals and doesn't start conducting until the third terminal reaches a voltage above a particular threshold. After that, the SCR will conduct even after that triggering voltage is removed. SCR latch up happens when you accidentally build an SCR in your IC and it gets triggered. Any place where the P-N-P-N structure is found, even if it's not designed as an SCR, will have this effect. The reason this doesn't happen all the time is that you typically need a very large triggering voltage to get it to turn on. But because that triggering voltage doesn't need to remain there after the SCR is conducting, a transient voltage spike can sometimes be enough to induce latch up. Latch up is most commonly seen when there's a large positive or negative voltage spike on the pin of a digital IC or when applying a supply voltage above the rated absolute maximum. One example where this might happen is if you have multiple power domains and each power supply takes a different amount of time to turn on. If the outputs aren't sequenced right, an unpowered IC might get a voltage on an input pin that causes latch up on that chip. There's more information in the "Schematic Design" chapter on how to prevent these kinds of sequencing errors. ESD events can also cause latch up. If you work in the aerospace industry, you will be familiar with single event upsets (SEUs) caused by high energy particles colliding with your electronics. In space, there is no protective magnetic field like there is on earth, so high energy cosmic radiation can collide with the sensitive electronics on a satellite or other spacecraft and cause latch ups. The reason latch up is worse in space is that there's no one there to reset the circuit. A total power cycle is the only way to clear a latch up. This is one of the reasons why spacecraft (and vehicles that operate at very high altitudes or are otherwise subjected to high radiation) often use radiation hardened (rad-hard) chips. Rad-hard chips are VERY expensive and are usually several generations behind the state

of the art in consumer electronics. This is simply because demand is not that high and engineers prefer chips that have been proven over time to be reliable. Using physically larger process sizes also helps reduce susceptibility to radiation. The way some designers deal with this problem without using rad-hard parts is by using distributed consumer grade parts designed in a failure tolerant architecture. Multiple chips running the same code in parallel are physically spread out over a spacecraft (to try to spread out the damage a burst of radiation will have) and vote on the correct output given the same inputs. If 3 out of 4 chips agree that something should happen, then it's likely that the 4th chip has been damaged in some way. A governor circuit can then trigger a hard reset of the suspected bad chip and hopefully fix the problem. If you're building something that will go into space and decide to go the cheaper consumer route (a lot of companies are doing this now), check out the MSP430 from TI as a microcontroller. Specifically, the MSP430s that use FRAM have been found to be highly resistant to radiation. They're not sold as rad-hard, so there's no guarantees, but that also means they're extremely cheap compared to rad-hard microcontrollers.

If there's a danger of an overcurrent event, use a fuse. There are two main kinds of fuses: slow blow and fast blow. A fast blow fuse is designed to blow (become open circuit) immediately after a certain current threshold is exceeded. A slow blow fuse has a time delay that will blow after the current threshold is exceeded for a certain amount of time. This way fast current transients can be allowed through, but a continuous overcurrent condition (like a dead short) will be prevented. To pick a fast blow fuse, use the equation below:

$$\text{Fuse current rating} = \frac{\text{nominal operation current}}{(\text{temperature derating} * 0.75)}$$

If you're operating the fuse at room temperature, you don't need to worry about the temperature derating factor (i.e., take it to be 1). As you increase the operating temperature of the fuse though, you need to start accounting for the fact that the

fuse will blow at a lower current. You can find the temperature derating factor in the datasheet of the fuse you use.

Picking a slow blow fuse isn't quite as simple as a fast blow fuse. Like ferrite beads, you'll need to look at the plots in the datasheet to figure out what will prevent damage to your device. In general, you want fuses to have a low DC resistance so they'll have a low voltage drop. No matter what fuse you pick, the only way to be absolutely sure that it will blow only when you want is to test it yourself in conditions as close as possible to the real application. Different sources will tell you to derate different amounts, and even different manufacturers have different methods and rules for rating their components. Choose parts based on the graphs in the datasheet and then verify that they will work for you by testing.

The problem with fuses is that once they blow, they need to be replaced[6]. Fuses are available in soldered packages that require desoldering to replace, or they can be used with fuse holders that make testing and replacing them much easier. The main deciding factors on whether you can use a fuse holder is the amount of space you have (since fuse holders take up extra height), the size of the fuse you need (fuse holders aren't available for every single fuse), and your budget (fuse holders are another BOM item that add cost). You can find both SMT and through hole fuse holders, and the SMT variants get surprisingly small. This lets you fit them in smaller enclosures, but of course it also means that it won't be able to handle as much current as a physically larger fuse. You should prefer a fuse holder if your design constraints allow it, because it makes repair much easier, even if it's not user accessible. However, you should consider the clearances and steps required to access the fuse for replacement during your enclosure design.

To avoid the problem of replacing blown fuses, you can use a PTC resettable fuse. This is a small passive device that sits in series with your power input. PTC stands for positive tem-

[6]Of course, blown fuses should be replaced AFTER the problem that caused the fuse to blow in the first place has been identified and fixed! Don't just replace the fuse and expect the problem to go away, and *definitely* don't short out the fuse to get it to quit blowing!

perature coefficient. As the name implies, as current increases through the device, the resistance also increases. That's why these components are referred to as resettable fuses: they act like an open during high currents, but look like a short once the current returns to normal levels. That means you don't need to replace them every time an overcurrent event happens. The downside to PTCs is that they can drop significant voltage and can have a high ESR. However, they are very low cost.

Fuses work well for very large voltages and currents, and PTCs fix the problem of having to replace a part whenever it fails, but there is an option that performs even better than a PTC. Several manufacturers make ICs that are specifically designed to prevent overcurrent and overvoltage conditions. They don't use a single name, but you can search for things like "over voltage clamp", "surge suppression IC", and "overcurrent protection controller" to find them. They all work in different ways, like a sense resistor and a FET or a modified crowbar circuit, for example. These chips will behave similarly to a PTC, but without the large voltage drop and sometimes with a faster response time. Some ICs will also give you digital outputs that you can use as an input to a microcontroller to detect faults and react to them. Other ICs have a programmable current limit. It's hard to recommend a single chip since there are so many and they all do different things, but take the time to shop around and see what's available. It can save you a lot of complexity in your schematic while also affording a great deal of robustness.

There are also protection ICs that will provide reverse bias protection. That way if someone manages to defeat any mechanical barriers to reverse biasing the device (for example, by forcing batteries in backwards or plugging in an unkeyed power cable the wrong way), your device won't be damaged. These ICs sometimes use an ideal diode circuit, which uses a FET to look like a diode with a forward voltage of 0V. The ideal diode circuit can be internal to the IC or external, but the application note will tell you exactly how to implement it. For a product that runs on user-installed batteries, reverse bias protection is essential. It can also be very useful on PCBs that are being tested or prototyped in the lab, since it's easy to get the test leads coming

from a power supply into your test board mixed up and reverse bias it.

Voltage spikes can result from ESD or lightning strikes, but they can also come from more mundane sources. Whenever an inductor is switched on or off, a voltage spike will occur. This is because of the relationship $V = L\frac{di}{dt}$. The inductor will create a voltage that opposes the change in current. This applies not just to discrete inductors in your circuit, but also to things that act like inductors. For example, if you're switching a motor on and off quickly or changing directions rapidly, its windings will act like an inductor and you'll see a voltage spike whenever the current through the motor changes. This spike is often clamped with a diode (referred to as a flyback diode when used for this purpose). The same effect can be seen when switching a transformer.

To prevent overvoltage conditions, you have several options. One of the most common methods is using Transient Voltage Suppressor (TVS) diodes. Any voltage above some device-specific threshold will be clamped down to a maximum of that threshold. TVS diodes have a very long lifetime. They can also clamp very quickly, as fast as 50 ps. Normal TVS diodes can protect from voltages as high as several hundred volts, but TVS diodes meant for ESD protection can be rated as high as 15 kV. TVS diodes are usually physically small and therefore cannot dissipate a large amount of energy. That means that any voltage spikes that last longer than a few microseconds can destroy the part. Characterize the kinds of voltage spikes you expect to see when choosing protection components. TVS diodes are available in SMT packages, meaning they can have a low lead inductance and thus, a fast response time.

Protection against lightning strikes falls under two categories: direct and indirect. The only way to survive a direct lightning strike is to divert the strike with a lightning rod and prevent it from getting to the PCB in the first place. Indirect lightning strikes are large surges that originally came from lightning striking another object. The methods used to suppress indirect lightning strikes are the same used for protecting against any extremely large voltage spikes. Chassis ground should be connected directly to earth ground. Connect all grounds together

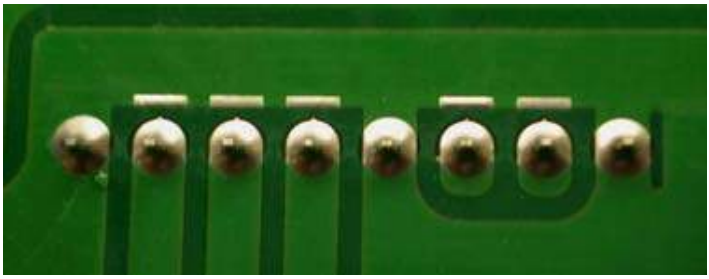

Figure 2.15: Spark gap on a connector

at one point with a 1M resistor and a 2kV 0.1 µF capacitor in parallel.

Spark gaps are often used to shunt very high voltages too. A spark gap can be implemented on a PCB by pulling back solder mask around a small area of ground plane. You can see an example of this in figure 2.15.

Spark gaps should not be relied upon as the only method of reducing voltage spikes. Their performance varies greatly based on the condition of the PCB, the humidity, and how ablated the spark gap is as a result of previous sparks. Spark gaps will also only reduce the voltage to a few hundred volts, which is still enough to damage a PCB. Where a spark gap is most useful is in preventing arcing across other protective components.

Another form of circuit protection is called galvanic isolation. This is mostly used as a safety measure (especially in medical equipment), but can also be used as an EMI reduction method by breaking ground loops. If one circuit is galvanically isolated from another, that means there is no DC connection between those two circuits. Galvanic isolation is typically accomplished using a transformer (since it has no continuous DC connection between the two coils) or an optoisolator, which uses an LED and phototransistor to pass signals across an air gap. The reason this adds extra safety is that it reduces the likelihood that a person touching the galvanically isolated side of the circuit will make a connection back to earth ground on the other side. This can prevent current from flowing through the person and electrocuting them. It's a good idea to use galvanic isolation when your circuit

is connected to a high voltage, especially if people will be near it or touch any part of it.

2.3 Active components

Datasheets have been known to fib about performance of some parts. After you gain some experience looking through distributor websites looking for particular parts, you'll get an idea of what's on the market. If you see a part that claims performance that seems too good to be true, it probably is. Datasheets love to use flashy claims on the first page to draw you in. Wideband amplifiers that claim 20 dB of gain will, upon closer inspection, actually only give you 20 dB of gain at exactly one input voltage, one input RF power, and one frequency. If you're planning on using a part at or near any of its limits, test it on an evaluation board first if possible. You can sometimes get these for free through sales engineers, or you may need to buy them. If they're too expensive, you can try fabricating it yourself, since evaluation board gerbers and manufacturing files are almost always released publicly or will be sent to you upon request. The idea is that you can verify that the datasheet is telling the truth and that it will perform the way you expect in your application.

2.3.1 Oscillators and crystals

When oscillators are used as an external reference for an IC, the datasheet and application notes often give oscillator parameters needed to guarantee nominal IC operation. They will sometimes also recommend specific part numbers. Follow these recommendations and use the specific part they list if possible. Sometimes the part numbers they use will be end-of-life or unavailable, so don't just blindly use the part they list. Make sure that it's still active and in stock with an authorized distributor.

Crystal oscillators typically need two external capacitors to function correctly, called shunt capacitors. The value of these capacitors should be taken from the datasheet or it can be calculated. A good reference for these calculations is the ST Oscillator Design Guide (available as a free PDF from ST Microelectronics).

That design guide contains a wealth of information about oscillators and crystals, so rather than recreate it here, I recommend giving it a look. In most circumstances, the best thing to do is use the values specified by the oscillator's datasheet. Always use NP0/C0G type capacitors. You can optionally decrease the capacitance by one or two picofarads depending on the physical size of the capacitors you're using to compensate for the extra capacitance you will get between the capacitor terminals and the ground plane of the PCB. As long as you're within a couple of picofarads of the correct value, the oscillator should behave just fine.

If you plan on using a crystal oscillator outside of its temperature range, you need to characterize its performance yourself. The datasheet will only guarantee performance within a certain temperature range, and outside of that, the crystal may experience something called an activity dip. This is the point where the crystal suddenly increases in resistance and can cause a frequency error of up to 20 ppm and cause your system to lose phase lock. These dips happen at very narrow temperature ranges, and they're almost never specified in the datasheet. If your design needs a highly accurate clock reference over a large temperature range, you should characterize the crystal you are planning to use and design around those points if necessary.

Crystals age over time and their resonant frequency changes slightly. Luckily, this is a very small amount, usually about 5 ppm during the first year and 10 to 20 ppm over the next ten years. Still, for very sensitive designs, it's important to be aware of this fact. This aging effect is not affected by temperature.

2.3.2 ADCs and DACs

There is a huge amount of information to consider when choosing an ADC or DAC, and there simply isn't room in this book for the discussion they deserve. Instead, check out TI's "Choose the Right Data Converter for Your Application" and the book "Data Conversion Handbook" by Analog Devices. They're both linked on the website for this book, DesigningElectronics.com.

2.3.3 Power supplies

Take the time to calculate your power budget. You can do this by determining the minimum and maximum current and power that each component will require and make sure your power supplies can provide at least 10% more than that. Make sure none of your components are dissipating more power than they are rated for, and that you've taken measures to deal with any heat that is produced by those parts.

When moving large amounts of power, it's more efficient to use a high voltage because the current will be lower, which will reduce your loss due to resistance in your traces, cables, or transmission lines. However, since most designs tend to run at a low voltage, you'll need to reduce the voltage back down again. A switched mode power supply (SMPS) is the best way to go from a low voltage to a high voltage and vice versa. There are a myriad of SMPS and DC-DC converter ICs that you can use. TI has a nice, free online tool called Webench[7] that will help you find a part that meets your requirements and run some simulations. SMPSs can be fickle if you don't use the right inductor, use a poor layout, don't have enough output capacitance, or run the part outside of its operating range. Make sure to use as many of the suggested parts in the application note as possible, and follow the layout guidelines. There's more information about that in the "Layout Design" chapter of this book.

Make sure you use DC-DC converters exactly as described in their datasheets and application notes. If you're using a DC-DC converter in a non-standard way, talk to the application engineers or simulate it using a model from the manufacturer and real (non-ideal) components. LTSpice is a good choice for simulating those kinds of things, and it's free. Keep in mind that DC-DC converters are less stable at small ($< 20\%$) loads. Don't use the synchronization feature on DC-DC converters unless you really need it. It's easy to cause increased noise or oscillation if the feature isn't used exactly right.

Run SMPSs at above 20 kHz so the switching noise is above the human hearing range. If the FET is switching *below* 20

[7] http://webench.ti.com

kHz, you will hear a soft humming or squealing noise coming from the inductor and capacitors. This is because some capacitors are slightly piezoelectric and some wire-wound inductors will physically move slightly as their magnetic field changes, creating sound. You can solve this problem by ensuring that any sound that is produced is above the human hearing range.

Sound shouldn't be the only reason you pick a particular switching frequency. More importantly, the switching frequency decides what value inductor you need, and the higher the inductance you need, the larger the physical inductor must be. Because there are so many different SMPS ICs, topologies, and applications, it's out of the scope of this book to discuss exactly how to choose a switching frequency for your particular design. The datasheet for your SMPS IC should have a section on switching frequency and usually some suggested inductors. The TI Webench tool will also let you sort SMPS ICs by the physical layout area required for operation, which includes the inductor size. It's common for chips to abstract away the switching frequency design decision for you and simply call out a single inductor value to use to get the performance described in the datasheet. In any case, you need to be aware of the size of the inductor required for a particular IC, especially if you have a space constrained design. Semiconductor companies can get tricky on you because they'll have big, glossy advertisements about how small their SMPS IC package is, but conveniently leave out that the size of the required inductor is just as big or bigger.

SMPSs can cause your power lines to have extra voltage ripple as a result of the inductor switching on and off. This ripple is characterized in the datasheet of the IC, but it's critical that you use the right capacitors to achieve datasheet performance. The number of capacitors, their type, where they're placed, and their value all determine the cleanliness of the power on the output of the SMPS. If ripple is important to you, check the datasheet of the part before you choose to use it, and make sure that it's acceptable even after derating it slightly.

If you use Low Dropout Regulators (LDOs), consider using parts rated for low noise. They usually don't have an impact on cost and can have significantly lower ripple on the output. They

can also save you a few components by reducing the number of output capacitors you may need.

Remember to compensate for the dropout voltage of your LDOs in your power budget. That means if you want 5V on the output, you need to provide *more* than 5V on the input. The lower the dropout of the part, the better, since less power will be lost as heat. LDOs operate most efficiently when their input voltage is *just* high enough to account for the drop, since any higher voltage will just be burned out as heat.

LDOs have a Power Supply Rejection Ratio (PSRR) that is given in the datasheet. This is the attenuation in dB of the noise on the input of the LDO to the output. It's not uncommon for this value to be 40 to 50 dB or even higher. You can use this to your advantage by using LDOs as a power supply filter. This can be helpful for removing ripple from an SMPS or any noise picked up by a long power cable. A common technique is to use an SMPS to do large voltage conversions more efficiently, and then put an LDO with a low dropout right before the ICs you're trying to power to help isolate them from the noise of the rest of the power nets.

Once nice feature that many power supply ICs contain is thermal shutdown. An internal temperature sensor will automatically disable the regulated output once the chip gets too hot. They will sometimes also have a "power good" pin, which you can use to drive an LED or check with a microcontroller to detect fault conditions and take the correct action. Look for both of these features when shopping for power supply chips.

2.3.4 Microcontrollers

Before you decide on an IC, especially a processor or other large, complex part, read the errata. It's a separate document that you need to download from the manufacturer website and it lists all of the know problems or mistakes in the silicon. Reading this document can save you days or weeks of troubleshooting, and in extreme cases, can cause you to pick a different part. The errata sheet will explain the known silicon-level problems, what their symptoms are, and what (if any) workarounds there are.

Estimate the amount of memory you will need. You need to do this for both RAM and nonvolatile storage. If you need more nonvolatile storage, you can use an external storage component. SPI EEPROMs are common and can store huge amounts of information (this can be useful for storing lookup tables to improve application performance). Talk to your software team about how much RAM and storage they will need to meet the design requirements. If your software team is already familiar with or already has a lot of code and boilerplate written for a particular processor family or architecture, you should probably give more weight to that processor family or architecture when selecting one.

Look for microcontrollers that already contain as many of your requirements as possible. If your requirements call for ethernet, try to find a microcontroller with built in ethernet instead of having to use an external ethernet IC. Other common busses that some microcontrollers include are USB, I2C, SPI, USART, and UART. Different chips have different numbers of each of those busses, so if you have a UART heavy application, try to find one that includes multiple UARTs. Bit-bang as few things as possible. A hardware implementation of a bus or protocol on your microcontroller will save you lots of development time, will generally have better performance, and is less likely to have bugs. Finding a microcontroller that supports specialized protocols like MIDI or CAN bus can save you a lot on your BOM, since you can eliminate an entire external IC and just use your microcontroller. You can even find microcontrollers with radio modules built into them.

The software development environment that is required for a microcontroller should be investigated before you make a decision on a part. If you decide to use the proprietary (and sometimes expensive) tools that the manufacturer provides, test them out first and take a look through the provided example code and libraries. You may be surprised at the poor quality of the code in the provided libraries, and it's likely that there will be bugs. If you can talk to anyone who has used a chip that you're considering using, they can be a great help in getting past the marketing veneer and telling you what it's really like to develop on the chip.

If you decide to use the manufacturer provided IDE, make

sure to budget for it. Sometimes they can be extremely expensive. There are some IDEs that cost money but have a free, feature reduced version. For example, IAR Workbench has a free version that only lets you compile up to 32 KB of object code. Not realizing this beforehand could cause you to exceed your budget or waste time optimizing code size if you can't afford the IDE. You also have the option of using a free, open-source tool chain. There are lots of tools and support for almost every microcontroller out there, but because of that, you'll probably need to take some time to get your development environment set up for your particular chip. One IDE that does a good job of integrating a lot of tools together for you is called PlatformIO[8]. It doesn't have support for every single IC, but it does support a lot, especially for IoT applications.

Before you go all in on a microcontroller, buy a development board for it. This will give you a good idea of how painful it is to actually get code running on the chip and get a feel for what development will be like. Development boards sold directly by the manufacturer are sometimes very expensive, but you can almost always find a cheaper third-party development board. Check sites like Tindie.com, eBay.com, or AliExpress.com.

It's always a good idea to derate microcontroller operating parameters so you're not running them at their maximum ratings. NASA recommends[9] derating integrated circuits as listed in table 2.10.

Stress parameter	Derating (Digital)	Derating (Linear)
Max supply/input voltage	90%	80%
Power dissipation	80%	75%
Max junction temperature	80%	75%
Max output current	80%	80%
Max Clock frequency	80%	80%

Table 2.10: Integrated Circuit Derating

[8] http://platformio.org
[9] Taken from MIL-STD-975, "NASA Standard Parts List"

2.3.5 RF

If you're picking out an RF part, pay attention to the insertion loss at the frequency you'll be using it. Insertion loss changes based on frequency, and the datasheet will usually advertise the minimum insertion loss across its entire frequency operating range. If you're not careful, your link budget will be wrong because you didn't account for the extra loss at your actual frequency of operation. As always, you need to check the plots in the datasheet and not go by the bullet points on the first page.

Gain is another frequency dependent parameter that is advertised in what can be a misleading way. This is especially true for wideband amplifiers. No matter what the datasheet says, an amplifier cannot be both perfectly wideband and perfectly high gain. There will be a curve showing gain over frequency, and *that* is the chart you should look at, not the gain in huge print on the first datasheet page or in the ad. The gain is usually lowest near the edges of the advertised frequency range. For example, if a part claims that it has 30 dB of gain and works from 1-2 GHz, you probably won't see 30 dB of gain at 2 GHz. Very high gain amplifiers will typically have a relatively small frequency range.

The higher the power that you're working with, the more important loss is in your parts. Consider a cable that you measure to have 1 dB of loss. At a low power, the difference of a single dB is a small amount of absolute power. But at a high power, a single dB can be watts of difference. This is why it's usually a good idea to put the high power/gain stage right before your antenna. Even low loss cables can cause you to drop significant power if they're too long or if you put them in the wrong place.

Don't count on being able to achieve the maximum power out that is advertised for amplifiers, especially if it's a lot of power. Give yourself at least a dB or two of headroom, even if you're going to be running the amplifier into hard saturation. As always, the best thing to do is to get an evaluation board of the amplifier you're interested in and driving it the way you'll be driving it in your product to see what performance you get. These development boards are often several hundred dollars, but you can usually get them for free from a sales engineer. You can

also check eBay.com and AliExpress.com for unofficial development boards for some parts, although these usually come with no documentation. Manufacturers will also usually give you the gerbers for the development board for free so that you can fabricate and assemble it yourself. The only downside with that approach is that you don't get a piece of paper with measured, verified performance data of your development board from the manufacturer.

If you want to amplify a signal up to any reasonably high power, you'll almost certainly need to use multiple amplifiers, or at least a single multi-stage amplifier. So, don't be surprised if you can't find a part that will do +30 dB of gain at +36 dBm out. Instead, maybe look for one amplifier that does +25 dB of gain at +26 dBm out and a second amplifier that does +10 dB of gain at +36 dBm out. There's a tradeoff between the maximum power out of an amplifier and the maximum gain it exhibits.

Prefer amplifiers that are unconditionally stable. Amplifiers have a stability factor, also known as a k-factor, that tells you if the amplifier will begin oscillating under poor impedance matching conditions. You can find amplifiers that are both conditionally and unconditionally stable. Unconditional stability means that no matter what impedance you put on the input or output of the amplifier, it won't oscillate. That doesn't mean it will never oscillate (there are other conditions you can subject it to that can cause oscillation), but using an unconditionally stable amplifier can save you a lot of troubleshooting. So then why would anyone ever want to use an amplifier that *wasn't* unconditionally stable? Well, it turns out, like in everything, there are tradeoffs in making an amplifier unconditionally stable. Specifically, the noise figure or P1dB may be worse in an unconditionally stable design versus a conditionally stable design. If these are tradeoffs you'd like to play with, then you can choose a conditionally stable amplifier. If you want to read more about this topic, there's a classic paper called "Avoiding RF Oscillation" by Les Besser with lots of good information. It's linked on this book's website, DesigningElectronics.com

P1dB, or the 1 dB compression point, is another important number to consider when selecting an amplifier. In an ideal am-

plifier, gain is linear. If you make the input signal a little bit bigger, the output will be proportionally bigger. But real amplifiers only behave that way up to a point, where the amplifier starts to go into compression. The input power where you no longer get a linear gain, but instead get a gain 1 dB less than you would expect an ideal part to have, is called the 1 dB compression point. This is an important number to look at, especially if you're going to be running an amplifier in compression or at its maximum output power.

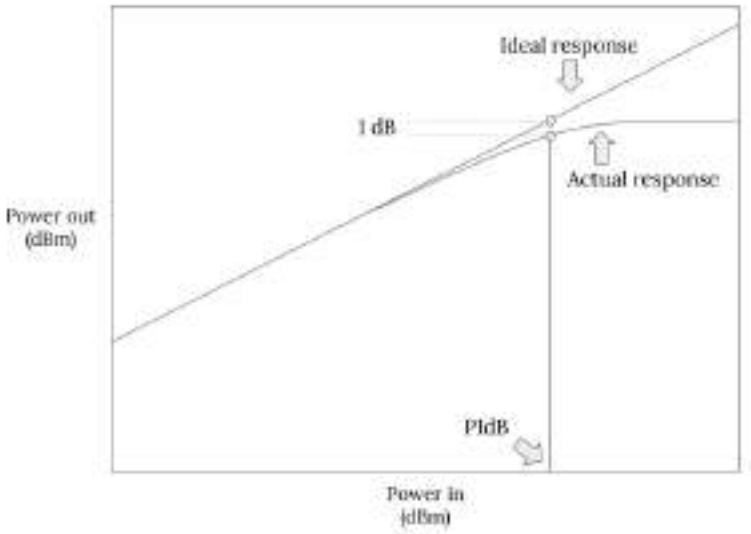

Figure 2.16: Explanation of P1dB

2.3.6 Transistors

MOSFETs should be derated according to the NASA recommendations below[10]:

[10]Taken from MIL-STD-975, "NASA Standard Parts List"

Stress parameter	Derating
Power	60%
Current	75%
Voltage	75%
Junction temperature	80%
Gate to source voltage	60%
Source to drain voltage	75%

Table 2.11: MOSFET Derating

MOSFETs do not instantly turn on and off. They have a gate capacitance that can be quite large in some cases. This capacitance adds a time delay before the MOSFET turns on. When picking a MOSFET, check that the gate capacitance is not so large that you won't be able to meet timing requirements. The equation for the time it takes for a MOSFET gate to charge to the gate voltage and turn on is[11]:

$$t = R_G * C_{iss} * \ln \frac{V_{GS} - V_{TH}}{V_{GS} - V_{gp}}$$

where R_G is the gate resistance, C_{iss} is the gate capacitance of the MOSFET as seen by the gate driving circuit at V_{DS}, V_{GS} is the final gate voltage, V_{TH} is the gate turn on threshold voltage, and V_{gp} is the gate plateau voltage. All of these values can be obtained from tables in the datasheet except V_{gp}. To get this number, you need to find the gate charge curve and read off V_{gp}. The gate charge curve will have V_{GS} on the Y axis and Q_g on the X axis. A typical curve is pictured in figure 2.17. In this example, V_{gp} is about 7V.

[11] The full derivation of the gate charge time equation above can be found in Vishay application note AN608A

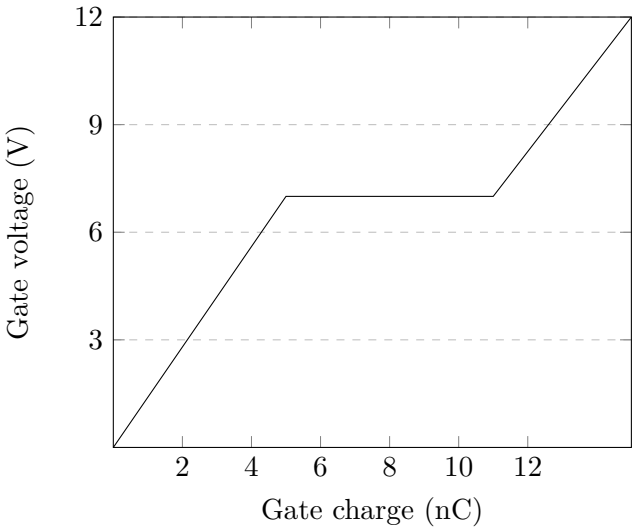

Figure 2.17: Example gate charge curve

Determine if you need a driver IC or circuit to control any switching semiconductors. For example, a microcontroller or processor GPIO pin will not be able to provide enough current or voltage to fully switch a large power transistor. GaN FETs require a specific power up sequence or else they will break, so you need special driving circuit for those devices. When choosing the driving method, pay attention to the switching times so that you can meet timing requirements. Having a fast MOSFET doesn't do you any good if you use a slow driver.

All diodes and FETs have a leakage current. This is because there is a nonzero drain to source resistance (R_{ds}). If you're designing low power or battery-operated electronics, a high R_{ds} will dissipate power unnecessarily. This can cause your measurement of consumed power to be higher than the value you calculated. In high power devices that are switching large currents, a high R_{ds} can cause your MOSFETs to overheat and fail prematurely or require significant thermal management. Avoid these problems in the first place by checking the drain to source resistance during component selection. You can calculate how much power (in watts) your MOSFET will dissipate as a result of R_{ds} by sim-

ply squaring I_{ds} and multiplying it by R_{ds}. It's typically pretty easy to find a MOSFET that has an R_{ds} of under an ohm, so that's a good place to start. It's also important to note that R_{ds} increases as temperature increases. So, if you're running your circuit in a hot environment or if you have insufficient heat sinking, your MOSFET will get hot, which will cause R_{ds} to increase, which will dissipate more energy as heat, which will make R_{ds} increase more, and you'll get stuck in thermal runaway until your MOSFET explodes. Make sure that you calculate the expected thermal dissipation due to R_{ds} across your entire temperature specification.

2.3.7 Diodes

Choose LEDs to be sufficiently bright. LEDs should have a maximum forward current of less than the maximum current source and sink rating on the LED driver or GPIO pin you're using to drive them. If they do not, they should be buffered. You can't run an ultra-bright, 10 watt LED off of a microcontroller pin.

When using a diode in series, make sure you compensate for the drop across the diode. If you need a really small forward voltage drop across a diode (usually the case when you use a diode in series), use a Schottky diode. The tradeoff is that they have a higher reverse bias leakage current, and this gets worse as the temperature of the Schottky diode increases.

Make sure the diode you pick can handle the current you need to put through it. You also need to specify the reverse voltage (the point at which the diode will begin to break down and start conducting in the "wrong" direction). Once you start approaching the rated reverse voltage, the diode will start to get leaky and small amounts of current will start to flow. You can use the I-V curve in the datasheet of the part you pick to figure out exactly how much current that is. If you don't give yourself enough headroom between the rated values and what you're actually subjecting the part to, you can end up with an inefficient design, since you'll be losing energy through diode leakage.

It's possible to use a Zener diode as a voltage regulator, but it's not recommended. You can also use it as part of a clamping

circuit to ensure the voltage at a particular point never exceeds the reverse voltage of the Zener diode.

2.3.8 Batteries

There are dozens of battery chemistries to choose from, and each one has its own advantages and limitations. Which chemistry you choose will depend on what fulfills your requirements, but 90% of designs can get by with one of the chemistries in this section. If you are in the 10% that is unable to use any of these chemistries, you either need a more exotic chemistry (which is out of the scope of this book) or more likely, you need to try to change your requirements or specifications.

Remember that you need to properly dispose of all battery types. If you just throw batteries away, they can poison the environment at best and explode or start a fire at worst.

If you need to measure detailed information about your battery performance, you can use a chip called a gas gauge. Gas gauges are all slightly different, but generally they have a Coulomb counter to measure charge accumulation into a battery, as well as current and voltage sensing. They can also sometimes provide digital outputs that report whether or not the battery voltage is sufficiently high, the charging state, and other useful pieces of information that you can use to manage battery charging or alert your user.

Note that in battery parlance, a primary battery means a battery that isn't rechargeable. A secondary battery means a battery that is rechargeable. A battery's capacity is measured in amp-hours (Ah) or milliamp hours (mAh). A battery with a rating of 1 Ah can discharge 1 amp for 1 hour or 0.5 amps for 2 hours, or 2 amps for 0.5 hours, etc.

Batteries also have a C rating, which is a dimensionless number used to describe the charge and discharge rate of a battery. When a battery's C rating is listed, it is usually referring to the maximum discharge rate. For example, a 1 Ah battery with a 1 C rating means that it can discharge no more than 1 amp in 1 hour before damage starts to occur. The same 1 Ah battery with a 10 C rating means that it can discharge up to 10 amps for

0.1 hours (which is 6 minutes). Just because you *can* discharge a battery at a huge C rate doesn't mean you should. Pulling a lot of current from a battery is going to make your battery get hot and will probably shorten the lifespan if you do it over and over. It's better to make sure you have a little room between the C rate you're using and the maximum C rating of your battery.

Never charge batteries at a C rating above what the datasheet recommends. For most batteries, this is 1 C. Doing so could damage the battery.

Lithium

When people talk about lithium batteries, they're usually referring to secondary (rechargeable) lithium-ion batteries. Within the group of lithium-ion batteries, the two most common chemistries you'll encounter in consumer electronics are lithium polymer (LiPo) and lithium iron phosphate (LiFePO4).

In lithium polymer batteries, the electrolyte is a polymer instead of a liquid. These batteries have a high energy density and a nominal cell voltage of 3.7V. Lithium polymer batteries with multiple cells must be balanced. That means the voltage on each cell needs to be the same so they are all discharged at the same rate. Many multi-cell LiPo batteries come with balancing circuitry already included in the pack. Depending on how the battery is constructed, you probably still need to use a balancing external charger. The failure mode you will see is that the pack will begin to get puffy, as gas escapes and builds up inside. If a LiPo is puffy, stop using it. Puffy LiPos can heat up, self-ignite, and then self-oxidize. Because the battery creates its own oxidizer when burning, it's impossible to put out with water. You just have to sit back and wait for the fire to burn itself out. Some battery packs have temperature probes built into them so the charger can detect any failure modes before they happen and stop charging. It's a good idea to use this feature if you can.

A lot of designs that use a rechargeable battery charge it while it's installed inside the device rather than have the user remove the battery and charge it externally. There are many cheap ICs designed explicitly for charging LiPo batteries, from one to many

cells. These chips do more than just apply a voltage and balance the cells. They also change the voltage and current curves going into the battery to optimize the lifetime of the battery and charge it as quickly and safely as possible.

The two most ubiquitous form factors of LiPo batteries are flat cells and 18650s. Flat cells are exactly what they sound like: thin, rectangular batteries. They come in all sizes, and cells of this kind are typically stacked on top of one another for multi-cell packs. 18650 batteries are an industry standard size cylindrical form factor. The name 18650 refers to the physical dimensions of the battery (18 mm diameter by 65 mm long [just ignore the ending zero]). There are other sizes of cylindrical cells available, but 18650 is by far the most common and popular. In addition to these two common form factors, LiPo batteries are also available as coin cells. There are many different sizes of coin cells, and you can buy both primary and secondary lithium coin cells.

Lithium Iron Phosphate (LiFePO4) is a recent chemistry that has the advantages of LiPo, but is much safer. When LiPo cells are shorted, they heat up and can cause a fire. LiFePO4 batteries won't catch on fire when punctured or crushed. The tradeoff is that they have slightly less energy density than LiPo batteries and can cost slightly more.

Lithium batteries should be stored at around 40% charge to maximize their lifetime. Most consumer products that use lithium batteries will be packaged and shipped at about 40%-70% charge. However, it's important to not allow the cells to discharge below 2.0V per cell or the battery may be permanently damaged.

A new lithium chemistry that is now commercially available is called Lithium Thionyl Chloride (Li-SOCL2). These batteries are primary (so, not rechargeable), but can have an extremely high energy density. A battery of this chemistry the size of a D cell alkaline battery can hold up to 20,000 mAh at 3.6V.

Lead Acid

Lead Acid batteries are almost never used in consumer electronics, but are sometimes used in robotics, automotive engineering,

or other contexts where you need a very large energy density, large current surge, and don't care much about the size or weight of the battery. Lead acid batteries are very heavy (since they contain mostly lead) and have a cell voltage of 2V. You usually see 6V, 12V, and 24V lead acid battery packs. Lead acid batteries are relatively safe. However, they do contain acid, and they also release hydrogen gas when they're charging, which can explode under the right circumstances. They're easy to recycle, provided you don't allow the lead to leach into the environment. To prolong their lifetime, lead acid batteries should be stored at 100% charge.

Alkaline

Alkaline batteries are primary batteries, meaning they're not rechargeable. This is the chemistry that AA, AAA, C, and D batteries use. Alkaline batteries are the cheapest way to implement battery power because they don't require recharging circuitry. However, they do require extra mechanical engineering, since your user needs a way to replace them when they die. A rechargeable battery can be charged just by exposing a small connector somewhere on the device. Removable battery covers can cause problems with dust or water ingress into the device.

Correctly stored, a primary alkaline battery can have a shelf life of 10 years. Because alkaline batteries are primary cells, you don't need to discharge them at all before you store them. There will be some self-discharge over time, but it will be fairly minimal. Like lithium batteries, alkaline batteries are also available as coin cells (although not all coin cells are alkaline, some use different chemistries).

3

Prototyping

You can't make it better until you make it work.
– excerpt from Atkin's Laws of Spacecraft Design

3.1 Hardware Prototyping Principles

Effective development is driven by risk reduction. One of the best ways to reduce risk is by prototyping and experimenting. Good prototyping skills can set you apart by increasing your development speed and by allowing you to pick the best solution not by guessing, but through results. Prototyping is all about being "close enough" and approximating more complete, fleshed out systems. You need to use your judgment to decide where the line is with respect to development speed and accuracy of results. Prototyping also requires a MacGyver-like intuition and inventiveness to use the materials that you have to prove out the core of an idea. A good prototype may look nothing like the final version, but will still prove or disprove your hypothesis.

When I advise companies, I always tell them that building something now and getting it out the door is better than waiting to build something perfect (unless you're talking about a safety critical or medical device). Just build it. Make a breadboard prototype, hack it together with an Arduino or Raspberry Pi or *something*. It's ok if it looks like a total mess of wires, it's more important to have something that works. Ideas are cheap.

Every prototype should have a purpose. Ask yourself what questions you're trying to answer from each prototype. Your first priority when prototyping should be to identify what your user wants or needs, what the purpose of the device is, and the high level constraints your design will have. Next, decide what a minimum viable product would look like and build that. The minimum viable product (MVP) is a prototype that has the bare minimum of functionality and features needed to be something that your users wants. Your MVP doesn't need to be highly robust, but it does need to work well enough for a small beta test with your users. You also need to make it safe. A quick, hacky prototype is no excuse for putting a device that could hurt someone into the hands of your first users. This will mean different things for different products, but you need to write your final specifications with safety in mind.

An MVP should test the core of your idea. You're allowed (and even encouraged) to cheat however you can. For example, if you can accomplish what you need by putting a Raspberry Pi with some sensors inside of a 3D printed enclosure, do it. Use parts from electronics hobby sites like Sparkfun.com and Adafruit.com. You're MVP is probably going to be bulky, look unprofessional, and may contain duct tape as a structurally significant part of the design. That's all ok, especially if it lets you quickly go from an idea to something you can test. If you're trying to eventually sell what you're building, try to sell your MVP. You'll have to make them by hand, and you will likely feel slightly ashamed at charging people money for it, but it's the best way to see how dire the problem is that your users are facing. If the problem is bad, users will take even a shabby version of a good solution over whatever they're currently doing to deal with the problem. The other thing that charging money will do is guarantee honest feedback from your first users. Since they have skin in the game, they won't be afraid to tell you if something about your product is bothering them.

If you need to make a lot of MVPs (on the order of hundreds), or if you need access to a special process or material that is difficult to get, you can find an existing product that does most of what you want, and then modify it for your special case. You

can do this by either buying and hacking products already on the shelf, or you can do it with the manufacturer. The way I've done this before is to go on Alibaba.com or Aliexpress.com and search for products close to what I wanted to make, or products that contained major parts of what I wanted to make. I reached out several of them and asked about the customization I wanted to make. Most of them are pretty amenable to doing custom work. I didn't even provide them with CAD or schematics, I simply explained in words what I wanted them to do and they drew up a CAD model and produced a single looks-like, works-like prototype for approval before making as many as I wanted. For a small kitchen timer type device with a custom enclosure, I was charged a flat fee of about $1000 for the work and the first single prototype.

If the look of your device is as important as the function, but you don't have the time, money, or skill to build something that both looks and works like you want, you can split them into two devices. These are commonly referred to as look-like and a works-like prototypes. A looks-like prototype can be made with materials like foam-core, 3D printing, or urethane casting. Foam-core and 3D printing are usually the cheapest and fastest ways to get close to the look you want. You may not be able to physically fit your electronics inside of this enclosure, but that's where the works-like prototype comes in. The idea is that you can show these two prototypes to investors or early users and demonstrate how the device works, while also letting them see the future vision.

Continue to iterate on your MVP until your users are happy and think you have a viable product. Don't wait for them to stop making feature suggestions, or for every one of your early users to agree on what to do next or that everything is perfect. Your users will never stop making feature suggestions, and you'll feel yourself continually coming up with new things to add too. On top of that, your users are not all going to agree with you or with each other. That's ok. The goal is to get to the point where you can stop making major changes to your product and start honing the design in an effort to get imperfect version one out the door and into more users' hands.

Once you are more sure of what you need to build, you can begin cost engineering. When designers start cost engineering before this point, they raise the likelihood of producing a product that people will feel is cheap. You also risk not meeting all of your product requirements and specifications.

After this, you can finally put more effort into industrial design. I have seen entire companies fail because they prioritized the industrial design over meeting the product requirements and building something that works. It sounds crazy, and it doesn't happen in an instant. It's a slippery slope, a boiling-the-frog scenario. You begin by getting an industrial designer to give you several options, and you eventually whittle it down to the one you want. The industrial designer assures you that what he or she has come up with is definitely manufacturable and should be no problem. You have trouble initially with some manufacturing problems but they seem solvable and before you know it, you're a couple months behind schedule, but you've already put so much time and money into this design (you went with the really prestigious industrial designer who wasn't cheap, didn't you?) and if you can only figure out a couple more issues, it'll all be fine. Meanwhile, you're burning through money and you miss the schedule to ship by Christmas, and before you know it, you're dead.

You can do another round of cost engineering after industrial design. A good designer will know how to cost engineer their design. If they don't have a LOT of experience doing manufacturing, do a round of DFM after industrial design too. Your factory can help you with this and they'll be better at it than your industrial designers simply because they know their machines and use them all day every day.

Depending on the kind of person you are, you may feel tempted to take great lengths to protect your intellectual property and keep others from stealing your idea. In the past, startups concerned about this have had early users sign NDAs or even refrained from letting anyone interact with the product outside of their company lab. It's true that it's easier to steal IP from a cobbled together prototype than from a security hardened final product. But this basically never happens. First of all, it's highly

unlikely that any of your first users will be motivated to try to rip you off. Second of all, at an early stage, not only will your product be underdeveloped, but the idea won't have even been fully proven yet. That's the whole reason you're building prototypes! I have known many early-stage startups that built hacky prototypes and thrust them into the world with zero regard for a competitor coming along and stealing the idea or technology, and none of them had any problems. The truth is, it's cheaper for a big company to wait for you to come up with a good solution and then just try to acquire you early, instead of stealing and replicating your project. This isn't to say that you should never consider security and IP protection, just that you shouldn't spend too much time on it when you're at this stage.

3.2 How to Develop Hardware Quickly

> *A good plan violently executed now is better than a perfect plan executed next week.*
>
> – General George S. Patton

I haven't gotten to interview Elon Musk yet. But Tim Dodd interviewed Elon Musk on September 28, 2019 at the SpaceX Starship Update event. Here are a few excerpts where Elon talks about how to make fast progress in hard tech.

Musk: "If a design is taking too long, the design is wrong. And therefore, the design must be modified to accelerate progress. One of the biggest mistakes made in advanced development is to stick to a design even when it is very complicated, and not strive to delete parts and processes. It's incredibly important."

To add some context to this, about a year before this interview, SpaceX switched the construction of their Starship from carbon fiber to stainless steel. This was after they had announced that it would be carbon fiber and had a huge press event discussing the design. They had also already bought and received a massive carbon fiber tooling part and were already building and testing huge carbon fiber pressure vessels. Despite publicly

committing to a design and spending tens of millions of dollars going down that path, Elon made the call to scrap all of it and start over with a totally different design. This was because the carbon fiber manufacturing process was taking too long.

Dodd: "So you're definitely not a sunk cost fallicist [sic]."
Musk: "No, definitely not."
Dodd: "You're like, 'This is clearly the new path forward, let's hop on it.'"
Musk: "Yeah. Is it in the future or not? If it's not in the future, who cares?"

Musk: "Everyone needs to broadly understand how the entire system works so you don't get subsystem optimization. The product errors reflect the organizational errors. [...] One department will design for the constraints given to them by another department without calling into question those constraints [...] You should actually take the approach that the constraints given to you are guaranteed in some degree to be wrong [...] You know they're not perfect. [...] Question your constraints. It does not matter if the person handing you those constraints won a Nobel prize. Even Einstein was wrong some of the time. Question your constraints. It's extremely important."

This concept that product errors reflect organizational errors is known as Conway's Law. Originally stated, it says "organizations which design systems [...] are constrained to produce designs which are copies of the communication structures of these organizations." As an example, remote teams tend to produce code that is more modular than teams where everyone sits in the same room. Elon suggests that the "seams" of an organization, the interfaces where different departments meet, can easily cause the organization to turn into a simple machine that almost blindly accepts and executes what each of the other departments say rather than thinking holistically and questioning *why* things are done a certain way and what drives the apparent constraints the department has declared.

Musk: "One of the biggest traps for smart engineers is optimizing a thing that shouldn't exist. When you're going through college, you have to answer the question the professor gives you. You don't get to say 'This is the wrong question'."

So how do you speed up hardware development? Building software is fast because compiling is fast and deployment is fast. Building hardware is slow because "compiling" hardware is slow and deploying hardware is slow. So, to speed up hardware development, we need to figure out how to speed up "compilation" and deployment.

Another way to say that you're speeding up "compilation" time is to say that you are reducing iteration time. You need to make a tradeoff here between the speed of development and the reliability of your system. Software developers do this too, but they can safely tradeoff reliability for speed more easily because they can deploy almost instantly, meaning fixing mistakes and improving reliability is cheap. If you want to speed up the iteration time of your hardware development, you will likely have to sacrifice some reliability. This is ok. Rather than deploying each iteration (like in software), you iterate on less reliable prototypes in your lab before deploying a single reliable design that incorporates all of your prototype iterations at once. The important thing is to not try to optimize for both reliability and speed at the same time.

But how do you speed up deployment? Well, this is really a question of logistics (and possibly manufacturing), and is out of the scope of this book. Appendix C has a list of logistics and shipping companies that can help you get your device into the hands of your customers as quickly as possible. Some of them can also handle the things inexperienced hardware startups might forget about, like returns, customs, packaging, and warehousing. If you're designing an electronic device, your core business is probably not shipping and logistics. So, find a partner company who has a lot of experience and can do it for you!

Progress only counts if you're moving in the right direction. It's critical that you understand *what* to build. Always remember that hardware serves the user, and the product team acts as

a proxy for your users. Software, hardware, and product are all interdependent on each other to deliver something that users actually want. Product informs software and hardware decisions based on their research and what they learn by talking to users. Hardware informs product and software about what capabilities are possible and what is feasible (or more often, not feasible). Likewise, software informs product and hardware about possible new features and what planned features are feasible. When all of these teams are working together correctly, the end product has the most valuable features possible, and they are things the users actually want. Engineers may think that the goal of moving quickly is only to reduce technical risk, but you also need to be incorporating as much product feedback as possible to reduce the product risk. For each iteration, ask yourself: what changes am I making that are a direct result of talking to users? There better be something!

When you're first trying to find a viable approach to a hardware problem, development often takes place serially. That is, the team starts down one path, and if it doesn't work, starts down a new path. Instead, try parallelizing. The key point is to have multiple people working on multiple approaches at the same time, and not to have the entire team working on a single approach. Bring up a low fidelity, proof-of-concept prototype for each approach before moving to the next revision of that approach or a new approach. For expensive, hard tech development, these low fidelity prototypes might just be simulations. You're trying to de-risk the path to fulfilling a requirement. Breadth first search is better than depth first search.

Don't get pigeon-holed into one design. If you're having an unusually difficult time getting something to work, eventually you need to put it aside, at least for a while. Find another approach that takes a separate technological path. Use different parts and techniques wherever possible. Sometimes you'll need to abandon the approach all together.

Do experiments. Spin cheap PCBs to test ideas. Measure things. Take the initiative to do small tests, build MVPs, and explore new products even before it's "official". If it doesn't work, kill it. Use your judgement and intuition to know when to end

an experiment and call it a failure.

Here are some practical ways to speed up hardware development. Note that most of these tips are aimed at early development and proving out ideas and basic functionality.

- Overnight everything. The cost of waiting an extra day or two is way more than the cost of overnight shipping. Never be waiting on any parts.
- Use the fastest turn time possible. The cost of waiting an extra day or two for fabrication or assembly is way more than the cost of expediting turn time.
- Cheat. Mock things up using "wrong" hardware (for example, a smartphone stuffed in a 3D printed enclosure running an app instead of a custom PCB with a microcontroller and screen). During prototyping, implementation doesn't matter as long as product requirements are met. Many engineers feel that the more difficult their work is, the more original it is, and the more clever it is, the better. They feel justified and proud of solving a hard problem. But one of the hardest lessons for an engineer to learn is that the real world does not care about how hard you worked to build something people want. It can feel unfair that a company with no interesting new technology and mediocre engineering can make way more money and be more successful than a different company with a much more interesting tech stack and harder problems. The good news is that you can use this to your advantage to increase development speed. Do not reinvent the wheel. People only care about the result. Do things that don't scale. Get help. Use application notes. Never implement anything yourself if you can find a chip that does it for you. If you can buy something that does or almost does what you want, buy it, don't build it. This isn't to say you should go into production with an expensive, hacky design, but for development and alpha testing with users, cheat as much as you can.
- Get a circuit mill. These can be expensive, with prices ranging from the $3200 Bantam Tools desktop PCB milling

machine to the $25,000 LPKF mill or the LPKF laser machines that are upwards of $50,000. These machines can also be expensive to run, since you need to buy lots of consumable end mills that wear out quickly because they're so small. There are other constraints when fabricating boards this way, like lack of solder mask and the ability to only mill two-layer boards. But in my experience, getting an LPKF milling machine dramatically sped up our hardware iteration time. We design a lot of RF stuff like antennas and amplifiers, and the LPKF works great. We order samples of substrates from Rogers for free, and can mill out and test designs within hours. We often mill multiple boards per day. Especially for RF development, where PCB manufacturers charge a lot to use exotic substrates and have long lead times for ordering the material they don't have in stock, milling boards can save you a lot of money and time in the long run. We use the LPKF ProConduct system to make vias, which consists of a silver epoxy that gets sucked through the via holes and baked out to make them conductive. Most of the designs we mill are exclusively surface mount and our smallest trace/space is 10 mils. We were skeptical at first, but if you have the money, getting a circuit mill can drastically change how fast you can iterate. In some cases (like RF development), it can actually end up paying for itself faster than you might think.

- Hire a technician. They'll let you parallelize work and are probably faster and more skilled at assembly and rework than you.

- Hire a layout designer. You can also contract it out, but contracting it out can actually cause delays in some cases. The reasoning here is similar to why you should hire a technician: a good, experienced layout designer is probably going to be better and faster at layout than you are, and they will let you parallelize work. You should expect to work closely with this person on a daily basis to guide and direct their layout. This is why contracting can be bad: it's hard

to closely collaborate with someone who is remote, especially if they're in another time zone. The other issue with a contractor is that they may have more than one job going at a time, which can slow them down.

- Use the right tools. High quality, purpose-built tools save you time and do a high quality job. That means good CAD tools, good lab equipment, and good components and materials. Invest in them even though they're more expensive. It'll save you time and money in the long run. Accuracy and precision matter. Good does not necessarily mean expensive. Good but cheap test equipment can be had on eBay for an order of magnitude less than brand new equipment.

- Use the simplest architecture possible, even if it means an expensive BOM. There will be fewer things that can go wrong. People don't doubt you can make electronics cheaper, they doubt that the product will work or that people will want it.

- Use evaluation and development boards liberally. Hook a bunch together to make prototypes. Design as few boards as possible. It will feel like cheating. That's ok.

- As a corollary, build multiple, separate modules when the time comes to start designing PCBs. Make things in chunks instead of a single huge integrated design. Have alternates for each module. They can be brought up in parallel, replaced if they break, and redesigns of subsystems are cheaper because you don't need to fab a huge board.

- Ignore form until you can't. Make the prototype as ugly as necessary to work. People who really have the problem you're trying to solve won't care about what it looks like. At the beginning, having a looks-like model only matters for investors. A works like model is what you need to figure out product market fit.

- Know when to ditch a design. Don't fall in love with any particular implementation.

- Pay for an assembly house to build your boards if assembling them yourself would take more than a day. Usually it's cheaper to pay for assembly than for you to do it. You can be bringing up other boards.

- Implement as few features as possible. What is the minimum viable design? Ten different designs with one feature each is better than one design with ten features. This is not to say that you shouldn't design your hardware to be adaptable, but don't get distracted designing and debugging features that are secondary to the primary problem your device is trying to solve. Don't build a Swiss army knife, build a steak knife.

- Redesign stuff that doesn't work well or that has a simpler implementation. Reuse circuits that have proven to be reliable and effective.

- Break up hard problems into multiple smaller problems. Break up systems into subsystems. Divide and conquer. Take it one part at a time.

- Keep a list of chips you like. You should know how well they actually work versus what the datasheet says. This is especially useful for chips that interact with firmware since you can reuse the hardware abstraction layer (HAL) and know that it works.

- If you're doing a complex but common task, use highly integrated chips. For example, an ethernet PHY can be implemented using an FPGA and some external buffers, but using a dedicated IC that does everything for you and can just be dropped in is better. If you're doing a simple task, don't use a highly integrated chip. For example, if you need to measure current, don't use a gas gauge chip that talks over SPI and I2C and measures 10 other things. It will take longer to integrate, can add unnecessary dependencies, and usually costs more on your BOM.

- Document. Write well. Don't use more words than you need to.
- Why do people do it that way? Don't do something just because that's how it's done. On the other hand, don't discount experience or proven components and methods just to be contrarian.
- Simple is better. Ask "do we need that?" for every feature. This does not necessarily apply to components on the PCB itself. Getting something working is more important than minimizing component count. You may not need that TVS diode, but you might as well put it in for the first prototype. You can always mark it as DNP later for free.
- Don't wait until you have all of the information you need to start analyzing, designing, or prototyping. You'll never have perfect information and it will always be a little bit incomplete. Guess or approximate.

Remember, these tips are only for the prototyping stage. They often sacrifice money and reliability for speed. These sacrifices are ok to make when you're trying to figure out what to build, but they're not ok when you're going to production.

Many software development teams use the Agile framework for development. Agile has several principles that motivate its processes. But since Agile was designed for software development, most of these principles are totally incompatible with hardware development. I came up with a list of fundamental principles that I believe should support an Agile-like development framework for hardware prototyping. Here they are.

- Modularity over integration - Don't try to put it all on one board too early
- User experience over final hardware - Cheat and use dev boards so the user can see what it will be like rather than wait till you're done designing custom hardware
- Working hardware over cost engineering - Spend NO effort on cost engineering until the hardware works

- Design reviews over rushed fabrication - Don't spend a week reviewing, but have someone else look at your design before fab. Catch silly mistakes. Don't just rush layout and immediately fab.
- Designing for adaptation over designing for manufacturing - Make rework easy
- Minimum viable features over extendibility - Avoid feature creep. Only put in what is needed. Everything else is just more that can go wrong.
- Speed over cost - Overnight everything. Use the fastest turn time possible. The money you save waiting a couple days is way less than the cost of delaying development by a couple days.
- Degraded performance now over perfect performance later - it's better to have something working now, even if the performance falls short of the specifications than it is to wait for weeks or months to achieve the full specifications. If the problem you're solving is really bad, people will use the degraded version, and software can be making progress while you continue to work to get performance up to full spec.
- Flexibility in software over flexibility in hardware - get as much done as you can in software rather than hardware.

Your development should be sustainable in terms of pace. People won't be able to work 16 hours a day for very long. You'll get high turnover and no one will have a healthy work-life balance. You also need to make sure you don't start to get sloppy. The pressure to move fast makes it tempting to accumulate large amounts of technical debt and start cutting corners in your design. Don't do it! It's only going to make progress slow down later.

It's impractical to get a design perfect before you send it for fabrication, but how long do you spend trying to find mistakes before you release it? A good rule of thumb is to get to the point

where you're about 90% confident that the design will work, and then release it. Past this point (for most designs), the extra time you'll spend ensuring that everything is 100% perfect will take longer than it would to just fix any small mistakes you missed. This won't work if your design review missed something major, so make sure your design reviews contain diverse engineers from multiple groups.

The 80-20 principal (also known as the Pareto principal) says that 80% of results come from 20% of effort. What 20% of engineering is going to make up 80% of the value of the final product? This can be a hard question to answer, but if you can, it will really help prioritize your work.

3.3 Modules & Eval Boards

Modules, evaluation boards, and breakout boards are a great way to get familiar with a chip or subsystem. Modules and evaluation boards usually include a chip and all of the required circuitry to drive it, while a breakout board usually refers to just a chip mounted on a PCB that makes it easier to solder to. Websites like Sparkfun.com and Adafruit.com sell dozens of modules and breakout boards for sensors, switching regulators, displays, and more. Having a box full of these modules can allow you to try new ideas or make last minute modifications very quickly. Your prototype won't be pretty, and it will take up much more space than the final version, but you can be relatively sure that each module will work on its own. This eliminates the question of whether or not you soldered something correctly or got the pinout right, or made any other implementation mistake. Modules and evaluation boards usually have schematics and layout available and are a great resource for implementing the device on your custom PCB since you have a known working design to start with.

The main thing to be careful with when using modules is to make sure you have them powered correctly and that all the logic levels between different modules are correct. It's very common for a design to use both 3.3V and 5V parts. There are modules

available that do bidirectional level conversion between 3.3V and 5V, and it's good to have a handful of those laying around to use between other modules that need level conversion.

All of these different functional blocks are usually connected with a nest of wire. One way to reduce the amount of soldering necessary while still allowing you to quickly change the connections between different boards are Wago lever nuts. Wago is the name of the company that makes them, and they're small terminal blocks that allow you to quickly connect and disconnect wire using a small lever actuator. A handful of these can be really useful in your toolbox. You can find a link to buy them on this book's website, DesigningElectronics.com.

3.4 Breadboards

Breadboards can be useful to some types of electronics prototyping, but not all. They have high stray capacitance and inductance, making them useless for any high frequency work. As Hobbs puts it in his book "Building Electro-Optical Systems", "... at least resist using white protoboards above 50 kHz, 100mA, 50V, or when a randomly sprinkled 0.1 ohms or 100 nH in series or 10 pF and 10 megaohm in shunt will screw you up."

One simple way breadboards defeat electronics newbies is by putting a break in the power rails on the edge of the board at the midpoint. To make it easier to use multiple voltages on a single breadboard, some breadboards will divide the power rails on the edge into 4 quadrants by disconnecting the rails at the middle of the board. Use a multimeter to check continuity between the very top power pins and the very bottom ones. If there's no connection, you'll need to add a small jumper to complete the connection. I usually leave these jumpers in if they're necessary so I don't forget to put them in each time I build a circuit. I've seen several instances where someone spends a long time trying to debug a breadboarded circuit only to discover that the bread-

Figure 3.1: Notice the jumper wires to extend the power rails along the entire length of the breadboard.

board they were using needed jumpers on the power rails.

Breadboards can also sometimes have problems making a good connection to the wires you stick in the holes. "You had one job!", you will scream at the breadboard after discovering that this has happened to you. If I have any problems getting a breadboarded circuit to work, the first thing I do is get out my multimeter and start doing continuity checks to make sure everything is DC connected as it should be. The other common mistake this will catch is being off by one hole when you plugged in a jumper wire.

3.5 Prototyping Platforms

The next step up from prototyping with modules is using purpose-built prototyping platforms. There are a myriad of prototyping platforms available with more being released all the time, so we'll only cover the most popular ones here.

3.5.1 Arduino

Arduino has done more for rapid electronics prototyping and electronics education than perhaps any other platform ever. A lot of more senior engineers will perhaps scoff at the idea of us-

ing Arduino for any serious prototyping, but don't let that stop you. It's not wise to use an Arduino in a real product (producing any significant quantity of such a product would very quickly become impractical), but it is excellent for trying things out quickly. There are so many libraries and so much software already written for the Arduino, it's amazing. Because Arduino is really just a bootloader, you can use the Arduino development environment on lots of different chips, not just the standard Atmel ATMega328p that comes on the Arduino Uno. You can also install the bootloader on more powerful chips with different architectures, like the STM32 family or the more power efficient MSP430 family. It's possible to deploy a product that runs Arduino code, but for a more reliable and feature complete codebase, you'll need to develop in an embedded C environment. You can often reuse parts of the Arduino libraries though, since they are written in C.

There are dozens of different sizes and shapes of Arduinos and Arduino compatible boards. Some contain only the bare minimum required to run code, and some are specifically designed for certain applications. You can find boards for applications like MIDI, automotive, wireless sensors, wearables, and many more.

The modules that are designed to work with Arduinos are called shields. They're easy to use because they plug straight into the Arduino, meaning they don't require soldering and they make it impossible to plug in incorrectly. Arduino shields don't have to be used with only Arduinos, so if you are looking for a prototyping module and find one you like that's actually an Arduino shield, you can easily use it with other prototyping platforms or development boards. You'll just need to wire it together yourself instead of simply plugging it in.

3.5.2 Raspberry Pi et al

The Raspberry Pi was one of the first extremely low-cost single board computers on the market. Since its release, it's come quite a long way. As of this writing, the latest Raspberry Pi has a quad core 64-bit ARM processor that runs at 1.5 GHz, dual band Wi-Fi, up to 8 GB of RAM, Bluetooth, Gigabit ethernet, and dual

4k video outputs, all for a starting price of $35. There are several different variants of the Raspberry Pi, and new models come out roughly every couple of years. The availability of the different models can vary, so make sure you can actually buy the model you want. Older models are discontinued, and even the latest model can be difficult to find since demand is so high.

The Raspberry Pi can run several different flavors of Linux. They also have 27 GPIO pins available on a 40-pin header, which includes SPI, I2C, and UART buses. Like Arduino, there is a massive community of developers with hundreds of libraries available for different applications and peripherals. There are also third-party hardware add-ons like Arduino shields, but they're called HATs (Hardware Attached on Top) in Raspberry Pi parlance.

While the Raspberry Pi was the first low-cost single board computer, there are now dozens of competitors at a similar price point. It's hard to enumerate them all, since new boards are released constantly, and old boards are discontinued with often little warning. Raspberry Pi remains the most popular by far, and competitors often find it difficult to match its price.

It's easy to accidentally damage or destroy development boards or single board computers. Many of them lack protection circuitry, since it adds to the BOM cost. Make sure you don't ever set a GPIO as a high output and then short it to ground. You should also never short a GPIO pin set high to another GPIO pin that is set low. In both of these cases, you can cause an GPIO overcurrent condition. GPIO pins can only source a certain amount of current before they're damaged. This maximum current can be found in the datasheet and you need to make sure you don't exceed it. Another one of the most common ways to damage one of these boards is to use the wrong logic level on the GPIO pins. If they use 3.3V logic and you apply a 5V signal (or vice versa), you'll destroy them.

3.5.3 RF Prototyping

RF prototyping is significantly more difficult than prototyping circuits at lower frequencies. You can't use a breadboard, be-

cause they have too much stray capacitance and inductance. The simplest way to prototype is to use screw together parts from a company like Mini-Circuits. You can buy amplifiers, attenuators, mixers, splitters, and many other components with SMA connectors. Then it's just a matter of screwing them together. While Mini-Circuits probably has the largest selection of screw together RF parts, other companies have higher quality or more specialized parts available.

Xmicrowave.com offers something closer to an RF rated breadboard. They produce dozens of small blocks with parts from different manufacturers and have devised a way to fit them all together without any soldering. The fact that they offer so many different parts from different manufacturers sets them apart from other companies that only offer screw-together modules for their own parts. Additionally, they have a free online simulator that you can use to prototype designs in software first and measure basic parameters before buying all the parts to build it.

Another useful tool for RF prototyping is a Software Defined Radio (SDR). SDRs allow you to digitally control a radio to transmit or receive over a large frequency range with a fairly large bandwidth. There are many SDRs on the market, but some of the most popular consumer grade SDRs include the USRP (Universal Serial Radio Peripheral), HackRF, LimeSDR, and BladeRF. SDRs can cost from several hundred to several thousand dollars, depending on the frequency range and bandwidth you need. One of the most popular SDRs for low bandwidth, low frequency applications is called the RTL-SDR. These can be picked up for about $30 and are great for simple receive tests or just experimenting. Be aware that RTL-SDRs are receive only, can't transmit, have an upper frequency range of usually about 1.75 GHz (depending on the exact model you get), and a bandwidth of about 2.5 MHz. Almost all SDRs can be controlled using a piece of open-source software called gnuradio. There are dozens of packages and plugins for gnuradio that allow you to do very complex things, and there's quite an active development community with lots of good documentation and tutorials online.

Prototyping with RF also requires some specialized test equipment to take measurements. See the "Building a Lab" chapter

for more information on what equipment you'll need and how to get it.

3.6 Lab Notebooks

> *For this you keep a lab notebook. Everything gets written down, formally, so that you know at all times where you are, where you've been, where you're going and where you want to get. In scientific work and electronics technology this is necessary because otherwise the problems get so complex you get lost in them and confused and forget what you know and what you don't know and have to give up.*
>
> – Robert M. Pirsig, *Zen and the Art of Motorcycle Maintenance: An Inquiry into Values*

A lab notebook is the cheapest but most important tool in your lab. A good lab notebook serves several functions. It is a good enough record of what you tried and designed that someone else with no prior knowledge could understand, replicate, and contribute to your work using only that notebook. It's not uncommon for a team to gain and lose people as time goes on, so it's very likely that your notebook will be used in this way at some point.

Another purpose your lab notebook will have is to prevent knowledge from being siloed. A term sometimes used to describe this is the "bus factor". The bus factor is the number of people on the team that would need to be hit by a bus to halt development progress due to lost knowledge. If a team only has one person, the bus factor is one. If the team has five people, ideally the bus factor will be five, because knowledge is spread out among everyone enough that there isn't a single point of intellectual failure. If the idea of large numbers of your team being hit by a bus seems unlikely to you, remember that the much less dramatic but all too common occurrence of engineers simply leaving the company will have the same effect.

Troubleshooting is another important use for a lab notebook. There's more information about this in the "Troubleshooting"

chapter, but a lab notebook can help keep you from accidentally doing the same work twice and can keep your ideas and hypotheses organized as you work through the science of troubleshooting.

The last thing that a notebook is valuable for is protecting you against claims of fraud or other engineering malpractice. If another company sues you, or if there's an investigation about the safety of a product, having good documentation about what you did and why during the development process can be valuable evidence in a courtroom, if it ever comes to that. Even in cases where the law isn't involved, if upper management is looking for a scapegoat or trying to place blame, you can use your notebook to prove that you acted ethically and in good faith.

Before the America Invents Act was passed, lab notebooks were used as legal documents to prove who came up with a particular idea and when exactly they invented it. Since the patent system is now first to file instead of first to invent, lab notebooks aren't really used this way in court anymore. While it's probably unnecessary to continue practices like having both the author and a witness sign every page in the notebook, it's still useful to number and date every page.

Some people prefer a digital notebook because it's easier to search for text and integrate images or other media. Personally, I prefer a physical notebook. It doesn't need to be charged, it won't get accidentally deleted, corrupted, or hacked, and it's less distracting (especially during meetings). You can still include images (and you should) by printing them out and taping or stapling them in the notebook. The inside cover of your notebook should have your name, phone number, address, volume number, and the dates covered. Some notebooks give you space at the beginning for a table of contents. If your notebook doesn't have that, leave enough space to make one yourself. As you add entries, add them to the table of contents. Write down the name of the experiment or investigation and the page number where it starts. Don't try to break up your notebooks by topic, just keep everything in chronological order. It'll be easier to find things later.

The Scientific Notebook Company makes great physical notebooks that I highly recommend. They also sell archival ink pens,

which are great to use for writing in a lab notebook. Archival ink won't fade or bleed, is waterproof, and won't smear or feather. Other good notebook manufacturers are Moleskin, Eureka, National, and Leuchtturm1917. Hybrid digital-physical solutions exist, like Rocketbook and smart pens. Popular purely digital solutions include an iPad Pro with a Pencil and Evernote. You can find links to all of these options and more on this book's website, DesigningElectronics.com.

Each entry should have a title, date, the hypothesis that you're testing or motivation for what you're doing, and a brief summary of any relevant background. The meat of the entry should have the materials you used, the conditions under which the test was performed, the equipment used, and setup of the experiment, the results, and an analysis and conclusion of the results. Include raw data, screenshots, plots, and any other relevant information. If your data is too big to fit, explain where to find it. If you really want to go the extra mile, you could even tape in a microSD card with the data on it.

Use your judgement to decide how much detail to include in your notebook. There's a tradeoff between keeping a good notebook and spending so much time on it that your progress on the actual product is very slow. It's probably a good idea to record the fact that you used high Q capacitors when tuning an antenna, for example, but probably not necessary to specify the mix of solder that you used. You do want to include tests that failed or didn't work as you expect. Negative results are often just as important as positive results. If data doesn't come out looking the way you want, or a plot turns out ugly, keep it in there anyway. It's the honest thing to do, and it will help you figure out whether or not you need to repeat experiments or conduct them in a different way to improve your results. For the same reasons, don't remove any pages from your notebook. If you need to cross something out, put a single line strike through it rather than scribbling it out. You might want read about the mistake you made later. Don't use scratch paper, use your notebook. If you're a perfectionist, you may be tempted to wait until you organize your thoughts somewhere else before you make an entry, or otherwise make something "good enough" to go into

the notebook. Don't do this. The lab notebook is not sacred. It is a place for elegant, beautiful, "eureka" moment breakthroughs as well as for wrong procedures, tests that didn't turn out right, and stupid mistakes. What matters is that everything is captured, not that it's a perfect picture of a genius at work. In the words of Ms. Frizzle, "Take chances, make mistakes, get messy."

Some people say that you shouldn't treat a lab notebook like a journal, but I disagree. A good notebook can almost act like a mind map, which will help you and others who read your notebook understand your train of thought and your motivations for your designs and experiments. It will help you stay focused, keep things organized, and see the bigger picture. Writing more like a journal will also make it much easier to get back into the same frame of mind after a long break. After not thinking about a particular problem for a few weeks, it's immensely helpful for me to read the "story" of where I was and what was doing. Again, see the "Troubleshooting" chapter for more information and tips about this technique.

It's a good idea to back up your lab notebook. If it's digital, this is easy. Just keep another copy of the files on a thumb drive, ideally geographically distant from where you work. If there's a fire, you shouldn't lose anything. Backing up a physical notebook is more work: you need to photograph or scan every page of your notebook and keep the pictures backed up in the same way as the digital notebook. It's harder to automate this with a physical notebook than a digital notebook, but you should still take the time to do it every once in a while.

The following two pages contain pictures of one of the best lab notebooks I have ever seen. I bought this notebook at a surplus sale, and it appears to be left over from a medical device company that went out of business. The first page shows all of the best practices talked about so far: a title, page number, date, description of the experiment, a drawing of the setup, and taped in data from the experiment. The second image shows an example of taping in pictures of the experimental results. Because this notebook was written in 2006, and because it was used for developing a medical device that had to be approved by the FDA, every page is signed and dated not just by the author, but by

a witness. As discussed before, this step isn't really necessary anymore. If you like the way this particular notebook is laid out, you can buy a blank one from the Scientific Notebook Company.

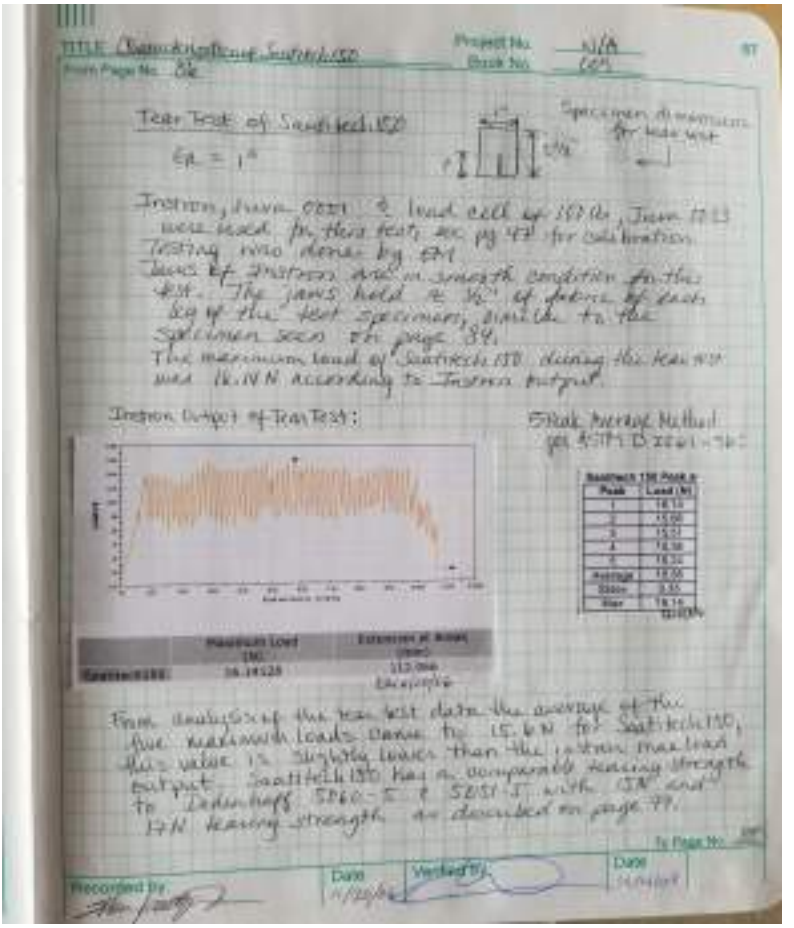

Figure 3.2: Example of a well written page in a lab notebook.

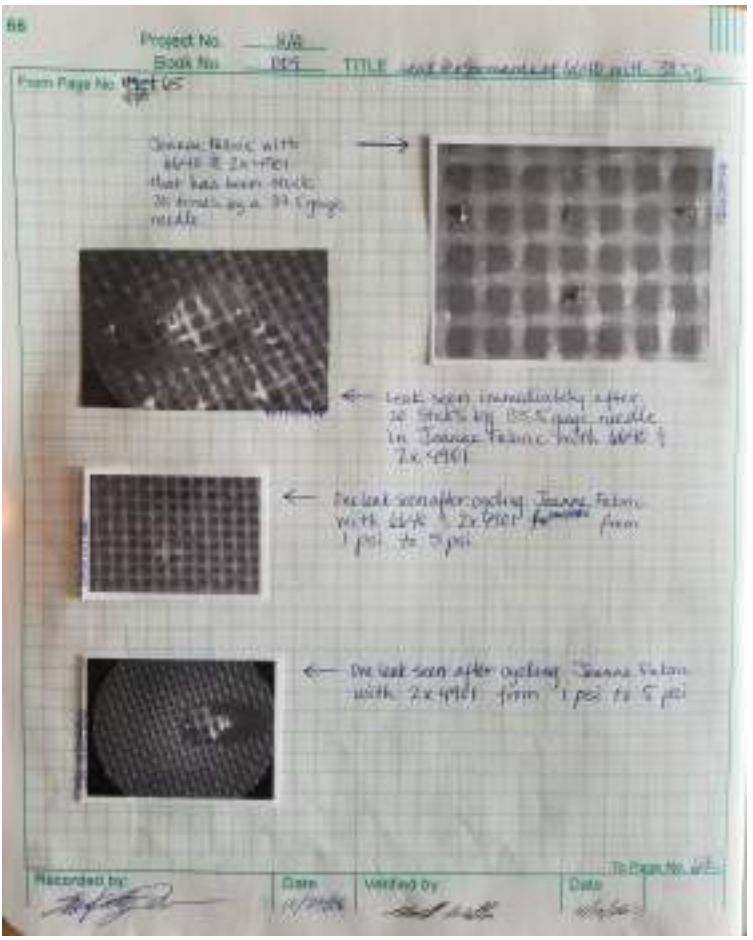

Figure 3.3: Example of a well written page in a lab notebook that includes pictures of the experiment.

4

Schematic Design

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one– and preferably only one –obvious way to do it.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.

– Excerpts from The Zen of Python by Tim Peters

One of the most important parts of creating a schematic is making it understandable. Its purpose is not just to tell EDA software which connections to make. It needs to be legible by humans too. Software engineers are encouraged to comment their code with explanations for why things are done a certain way and explain how things work. Schematics should likewise be annotated so that it's clear what's going on and why certain design decisions were made. This will make things easier for the other people who will be reading and reviewing your schematic, not to mention your future self when you forget why

you put that resistor there, or how you calculated that value.

Aside from the actual schematic, it's usually very useful to draw a high level block diagram of your circuit. If you use hierarchical blocks in your schematic, you can combine the two so that your high level schematic acts like a block diagram showing where signals go, and then opening the schematic that is represented by each hierarchical block shows you the details of how it's implemented.

4.1 Conventions

There is no one right way to draw a schematic, but certain rules will help keep things understandable and reduce the likelihood of mistakes. Maybe you are used to using different letters for component designators, or your EDA software doesn't behave in the way that's recommended here. That's fine, just be aware that the point of this section is to discuss ways to make things easier both during and after schematic capture.

Every component needs to have a unique designator, and most EDA software will automatically do this. Some CAD packages will assign a unique designator every time you put down a new component. Other CAD packages require you to explicitly run a designator assignment. Each class of component should have its own letter prefix. The standard prefixes are in table 4.1.

Just as every component in the schematic has exactly one unique designator, every designator must be mapped to exactly one component in the layout. There should be no components in the layout that are not in the schematic. Depending on the EDA software you use, you can usually check for this by running ERC (Electrical Rule Check). Some tools like Eagle will cause a huge fuss if the schematic and layout ever get out of sync. Other tools like Altium work on a different model and purposely let the schematic get ahead of the layout, and then have an update step in the workflow, where you specifically tell it to add the new parts and nets in the schematic to the layout.

A schematic can and should contain more than just the circuit diagram. It can also denote critical layout, shielding, and grounding requirements. Usually, schematic capture is done before layout, so anything that will be helpful during layout should be included in the schematic. Text boxes and comments work well for this, but the schematic itself can also describe graphically how to lay the circuit out. For example, when placing the capacitors for a particular IC near the chip itself in the schematic, smaller value capacitors can be placed next to the IC symbol and larger capacitor values can be placed further

Letter	Component
C	Capacitors
D	Diode, Zener, Schottky, LED
E	Antenna
F	Fuse
J	Connector
L	Inductor, choke
P	Mechanical features, fiducials, screw holes
Q	Transistor, FET, SCR, TRIAC
R	Resistor
S	Switch
TP	Test point
U	Integrated Circuit
X	Crystal

Table 4.1: Preferred component prefixes

away. This visually represents that capacitors should be placed the same way during layout. However, do not make the schematic look like the desired layout at the expense of schematic clarity.

If you have a large component with many pins, it may be impractical to draw a single schematic symbol for it. Instead, you can break it up into sub-symbols. To indicate this, append the designator with a letter. For example, a chip that has two schematic symbols would be called U3A and U3B. You can see an example of this in figure 4.1 and figure 4.2.

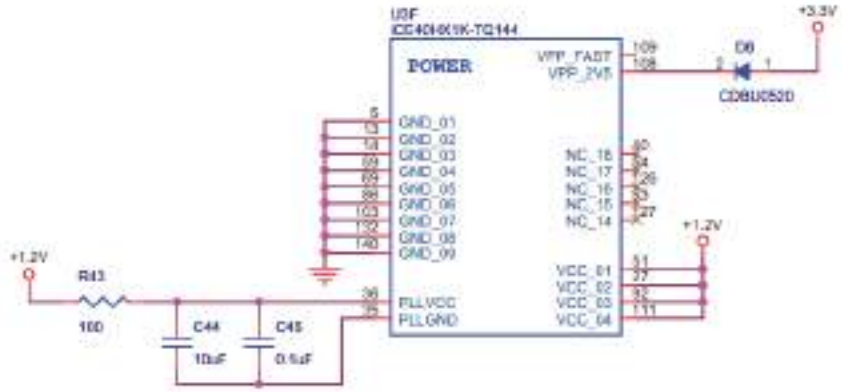

Figure 4.1: The FPGA in this schematic has been broken up into multiple blocks or subcomponents, and this one shows all of the power pins on the device, even though they're physically spread all over the chip.

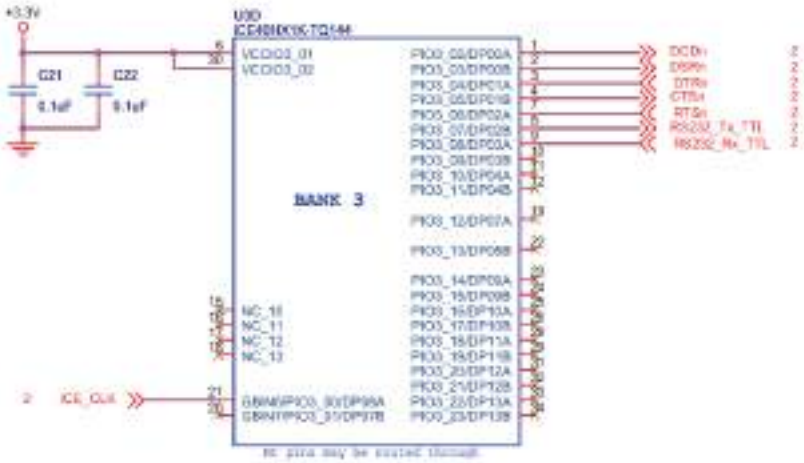

Figure 4.2: This other block from the same schematic shows a subcomponent containing a bank of IO.

One way to improve readability is to group all similar pins together in their own schematic sub-symbol: one with power and ground, one with analog, one with digital, etc. You also don't always need to draw

the schematic symbol with the same pin locations as the real chip. For example, if the real chip has communication pins spread all over, you can group them together on the schematic symbol to make the circuit easier to follow. Figure 4.1 and figure 4.2 show examples of grouping pins into a symbol by function.

Always place dots on all net junctions if they are meant to be connected. If you see two wires cross each other in the schematic and there is no dot at their intersection, those wires should not be electrically connected. You should try to avoid that case so that there are never two wires crossing without being connected, but that's not always possible or practical. It's better to use "T" junctions instead of "+" junctions so that it's easier to find where the EDA software mistakenly forgot a junction dot and thus didn't make an electrical connection.

Make sure to use a consistent grid when drawing your schematic symbols and when drawing your schematics themselves. If your symbol was drawn on a grid that uses different units or a different spacing, you can end up with parts being slightly offset or off-grid in your schematic. At best, it looks bad, and at worst, it can make a wire next to a pin look connected when it's actually not.

If a component is unusual or particular, you can make the manufacturer part number visible next to the schematic symbol. This will mostly be useful for when someone else is reading the schematic. They will be alerted to the fact that the part is unusual and will be able to copy and paste the part number into their browser to get the datasheet for it.

Footprint mistakes are all too common in the first iteration of a PCB. It's easy to mess up the mapping from signal to pad because there are several conversion steps in between that depend on different conventions. There's the pin numbers that are on the schematic symbol itself, then the pin numbers on the pads in the footprint, and finally the mapping between those. These should always be all the same. The pin number on the schematic symbol should be the same as the pad number in the footprint, so the mapping is always 1 to 1, 2 to 2, etc. The problems arise when you try to reuse a schematic symbol for a new footprint. This most commonly happens with connectors. You already have a schematic symbol drawn for a 20-pin connector, but you have multiple physical connectors that have 20 pins. *Usually*, connectors and ICs use the same pin numbering scheme, but not always! There's a lot more to say about avoiding mistakes with connectors that will be discussed later, but for now just know that keeping a one to one mapping between all pin numbers and pad numbers could save you a board spin or painful rework. Always manually check the mapping on

your connectors. The best way to avoid the schematic symbol reuse problem is to just not reuse schematic symbols. It doesn't take that long to draw a new one, and you'll know it's correct.

With the exception of two pin passive parts (like inductors, capacitors, and resistors), do not hide any pins or pin numbers in the schematic symbol. Your reviewers will assume the pinout is correct or forget to go into the library for the part and check. When the pins and numbers are visible, it's easy to pull up the datasheet and check against the pinout and application circuit. If you're using a polarized capacitor, denote which side is the anode and which is the cathode. The most common way of doing this is making the line on the cathode size curved ("C" for cathode is a good way to remember this), as seen in figure 4.3.

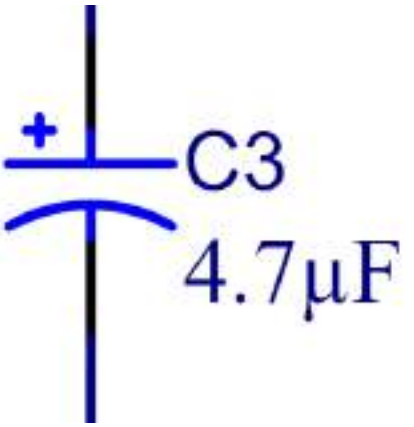

Figure 4.3: Polar capacitor symbol.

This works for visually denoting polarization, but if you get your footprint pin numbering mixed up, the part can still be populated wrong. Even if you are assembling by hand and pay attention to polarity, any future assembly jobs that are handled with a pick and place machine might get it wrong if the polarity is swapped in CAD. Some designers will set the pin names to "A" and "K" (anode and cathode) in both the footprint and the schematic symbol to guarantee that polarity is correct. This is also done with diodes, which also have only two pins and a polarity. The advantage of this approach is that you can re-use schematic symbols with less risk, since you're matching "A" to "A" and "K" to "K". This works even if different components use different conventions for where pin one is.

For neatness and completeness, put the name of the circuit, its version, and the project on each page. Schematic sheets typically have a small box in the lower right corner for this purpose. When you print schematic pages out and have them lying all over your desk, you will be thankful that you can tell which page goes where and if it's an old version or not.

Try to keep all schematic sheets the same size so it's easy to print them out on a normal printer. In the US, 8.5" x 11" is what most people are familiar with, but it's not always big enough to fit a schematic. The next size up you should use (in the US) is legal paper, which is 8.5" x 14", and after that is tabloid, which is 11" x 17". I like to use tabloid size when printing schematics out. Whatever size you use, try to be consistent and pick a size that allows you to adequately display the schematic without being too dense or too small to read. Remember, readability is paramount.

Don't feel pressure to keep the entire schematic on a single page. You or someone else will probably print out the schematic at some point, so make sure a single page isn't so crowded that it's difficult to read, or will be really tiny when printed out. Put subcircuits on their own schematic pages. Use hierarchical blocks to connect multiple schematic sheets together.

As mentioned before, text in a schematic is a great way to annotate and clarify. However, this text should not overlap any wires or symbols. Text that is a part specific detail should be placed as a visible component parameter, not as free text. That is, instead of dropping in a text box next to the component, go into the component properties and use the comment field, and make it visible next to the symbol. Not all EDA software supports this, but if the option is available it's easier to generate documentation if the software knows which note goes with which component. It's also useful if you need to move the part later, since the comment will move with it.

When drawing the schematic, it's typical practice to keep inputs on the left side, outputs on the right side, power on the top, and ground on the bottom. As usual, the exception to this is if there's another way to do it that results in a more legible and clearer schematic.

Power and ground symbols don't cost you anything in your schematic, so use them liberally. Don't draw large loops of wire to connect power and ground, just drop another symbol. Try to avoid rotating the power and ground symbols so they're not mistaken for another component that way.

Show the values of each component next to the schematic symbol. Use the suffix "R" to mean ohms when specifying resistance (using the

letter "O" can get confusing because it looks like a zero).

Call out test points on the schematic. It's also useful to call out information that may be helpful in debugging or troubleshooting faulty units. You can note what the expected waveform, voltage, current, or frequency should be at various points, especially test points. This is really helpful during assembly, testing, and debugging.

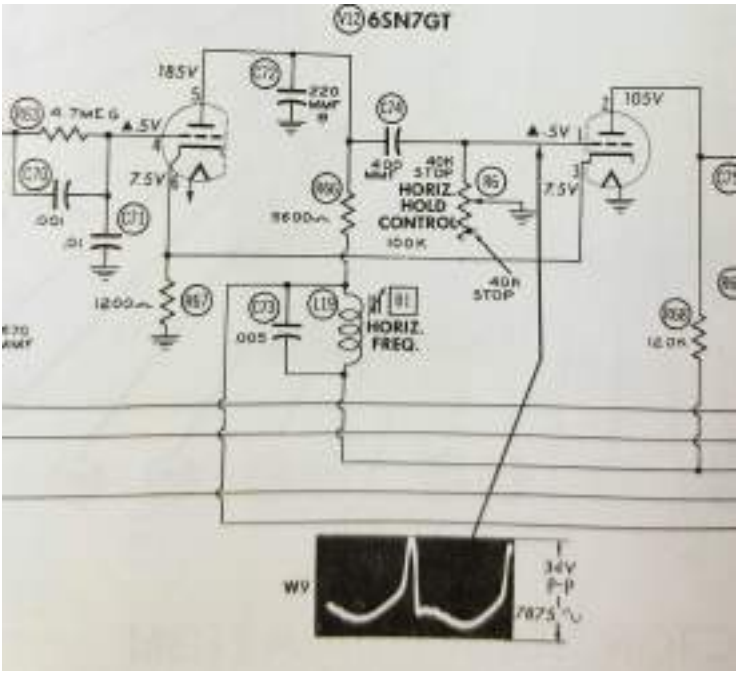

Figure 4.4: An example of an old television schematic from the 1950's that includes screenshots of the expected waveform at various points in the circuit.

4.2 Avoiding mistakes

Before you have other people review your design, the best way to catch mistakes is with your EDA software's ERC. This will automatically check for single ended nets, unused I/O, unterminated signals, and other errors. The goal should be to be have zero ERC errors and warnings. If you're getting warnings about unused pins, for example, add pin terminations or ERC ignore markers to them. If you do end up with some ERC warnings or errors, you need to be able to explain

every single one, why it's ok, and why it isn't fixed.

Check your logic levels. It's easy to connect a 3.3V GPIO to a 5V part without realizing it. If the 5V part doesn't have a logic threshold of at least 3.3V, you won't be able to control it with a 3.3V GPIO. You need to use a level shifter. You can either use a level shifter IC or you can make one using a buffer. Whichever solution you use, make sure that level shifting will work bidirectionally if necessary.

When you download datasheets, get them from the product page on the manufacturer's website. If you search for a part number on a search engine, the results may be out of date or pointing to an old version of the datasheet PDF. There are often multiple versions of the datasheet for a single part, so it's important that you have the most up-to-date copy.

If you're using interrupts on a microcontroller, check to make sure the pin you're using supports interrupts. Some microcontrollers only support interrupts on specific GPIO pins.

Don't forget pull-up resistors. Any open collector or open drain output should have pull-up or pull-down resistor to keep them in a known state at all times. Floating outputs can cause problems with other devices or interfaces. I2C buses need a pull-up resistor on both the SDA and SCL lines.

Include necessary protection circuity. If you're driving anything that is strongly piezoelectric, have protection circuitry for the driver so that any large mechanical impulses don't cause a voltage spike and destroy the driver. If you are designing an application that uses an H bridge or drives another large inductive load, make sure you use a flyback diode. An inductor that has a changing current through it will produce a large voltage to oppose any change in current. This means that when you switch current through a motor (which looks like an inductor), a large voltage will appear across the motor, which can destroy your driving circuit. To prevent this, use a flyback diode, which is just a diode placed such that current can only flow in the direction you want and can't destroy your driver.

Calculate a power budget. Leave yourself some wiggle room. Have you compensated for any voltage drops across diodes on your power supply or from LDOs? Do you know how much room you have above the maximum power consumption for your design? Every power supply should be able to provide enough current and power for the components they're powering. Think about efficiency too: you don't want to be burning a huge amount of power as heat in linear regulators. Large voltage conversions are better accomplished with a switching supply.

Check that every part will be running on its correct operating volt-

age. This is a really simple and seemingly obvious thing to check, but when you're dealing with dozens of chips that run at different voltages, it's surprisingly easy to mess up.

Calculate a link budget. Leave yourself some wiggle room. Every wireless link should work without SNR problems because you calculated the gains, losses, amplifications, and attenuation of each element in the chain. Estimate the channel loss and give yourself plenty of room. Don't forget that dB is a logarithmic scale, so it's easy to shrug off one or two dB. However, one or two dB when you're talking about multiple watts of power is significant. Converting things to watts can help give you a better intuitive idea about whether those couple of dB matter or not.

If your design is battery powered, make sure everything will keep working even when the battery is discharged to its minimum operating point. Where this point is exactly will be defined in your product specifications. LDOs are particularly sensitive to this. If you have an LDO with a 5V output, it may need at least 5.1V of input (or more). If you're running that LDO directly off your battery, that means your circuit will power off when the battery drops below 5V. You may need to add a boost converter to keep that voltage high enough even after the battery has discharged below the power threshold of the LDO.

Race conditions can happen in hardware just like in software. Look for any cases where the order of arrival of signals matter. Using pull-up and pull-down resistors can aid in preventing these cases. A technique called "back pressure" borrowed from FPGA development can be useful. As data moves along a pipeline, each stage waits for a "data ready" signal from the previous stage before trying to receive and process the data. This is often applicable to power supplies, some of which have "power good" pins that will tell you when the output is stable and ready for use. Using back pressure can help prevent cases where one stage or module in a design is activated before the previous stage has finished.

Polarized capacitors cannot be reverse biased. Design your circuit so that you know this won't ever happen. Reverse polarity protection for the entire design is a good idea, whether that's through electronic protection that can survive a cable that's plugged in backwards, or through mechanical means that make it impossible to plug a cable in backwards. If your device can act as a USB host, consider adding overcurrent protection to limit the current draw on the 5V rail. Misbehaving USB devices may try to draw more current than you specified, or may fail short.

One of the easiest things to mess up is connectors. It's all too easy to look at a datasheet drawing and confuse the top-down view with the

flipped-upside-down view and reverse all the connections. Check that all connectors and their mates are in the right direction, orientation, and gender. It's best to print out the layout at a 1:1 scale and lay the real connector on the piece of paper and make sure it's facing the right direction and will mate correctly. Don't just think about it in your head, actually do it.

Make sure it's ok to leave any unused inputs or outputs floating/unterminated. The datasheet will tell you if you need to connect unused pins to ground.

If you find a discrepancy between the design in a part's application note and the development board, prefer the development board design *as long as you've tested the development board and it performs as you want*. Many times, development boards are designed by junior application engineers. Despite this, the advantage of a development board is that they are usually characterized for performance, and you can request the measurements from the manufacturer to verify that you get the same results on your copy of the development board. If it's satisfactory, then you can exactly replicate the development board design. But be aware that they may not implement the exact set of features you need, and you may need to modify the design to work for your application. Even if a development board or application circuit appears to work well enough for your application, don't forget to make sure that it will pass EMI/EMC as well. Development boards and application notes often don't describe the optimal way to do things, especially when it comes to EMI/EMC, since they may not be FCC certified.

If you're using a potentiometer, keep in mind that as you turn the wiper it may bounce open momentarily. That means that your circuit needs to be able to survive the potentiometer looking like an open circuit for at least a couple microseconds. Push buttons and switches also have the property of bouncing around before settling on their position. Your software must also be able to survive this short period of bouncing, since if it polls the status of the switch or pot before it's finished bouncing around, it may read the wrong value. There are lots of examples of debouncing circuits that can deal with this problem, as well as debounce code for software that can handle it without any additional hardware.

Check that all of the parts on your BOM are in stock and orderable. You'll do this again after you complete your layout. You can lose valuable time if you get far into the design before realizing that one or more of the parts you were banking on using aren't available and you need to redesign without them.

High-speed and RF lines need to be terminated at the right impedance.

The right way to do this will depend on your application, but it's important not to forget.

4.3 Debugging

When it comes to debugging, one of the best things you can do is make your design easy to troubleshoot during the design phase. You're reading this book, so nothing will ever go wrong in any of your designs again, but *just in case*, make it easy on yourself to bring up and troubleshoot a board.

Add test points to important signals. Put ground pads next to those test points so it's easy to clip on an oscilloscope probe without stretching the ground wire across the entire board (important for not just practicality, but also signal integrity and accurate measurements). If you don't want to populate actual test points, you can just use the exposed edge of a nearby passive on that net or drop in a zero ohm resistor in series with the trace and use that. Be aware that high-speed, RF, or otherwise critical nets are a special case and often cannot have a test point hook added to them without compromising their signal integrity. Adding a soldered-in test point can add an electrical stub to the trace, which can cause reflections or cause other problems. Instead, you'll need to design a test point (for example, a directional coupler) that won't disturb your critical net.

Break out the clock and data lines of every bus you're using into a test connector. You don't have to populate the connector, but if you ever need to read, write, or observe anything on that bus, having a connector will make things much easier than having to solder wires to component pins. If you're using a high-speed bus, this probably won't be as simple as running another set of wires out to a connector since you might incur reflections or other issues that can prevent the bus from working correctly in the first place. In those special cases, follow the guidelines in the standard of the bus you're using.

If you have a SPI bus in your design (or any bus that has a chip select), make sure to put pull-up or pull-down resistors on the chip select pins of each slave device to ensure that they are default disabled (or un-selected). This will make debugging much easier, since you won't have more than one slave device vying for control over the bus. It's best to use external resistors to do this rather than relying on internal pull-ups or pull-downs on your microcontroller, so that you won't have bus contention even if you can't set the registers in your microcontroller to enable the pull-up or pull-down resistors. Most SPI devices are set up to require a pull-up resistor on the chip select line, since that pin is

usually active low.

If your design incorporates a UART, you can connect the TX line to an interrupt on the microcontroller so that when communication starts, the microcontroller will wake up if it's asleep. If you have a dedicated programming UART, you can use that interrupt to enter your firmware's debug mode when you start a serial terminal session.

All UARTs should have TX and RX connected correctly. This sounds simple, but is complicated by the fact that sometimes signals are labeled with *their* name, and sometimes they're named by what they *connect* to. Sometimes the signal labeled TX goes to the pin labeled RX, like you would expect. But sometimes the signal labeled TX goes to the pin labeled TX. Read the datasheet carefully and make sure you're connecting a transmitter to a receiver, not a transmitter to transmitter or a receiver to a receiver. Consider naming nets something more descriptive, like TX_DO for TX data out and RX_DI for RX data in.

Check your reset polarity. If something won't program, it might be because it's being held in reset. Check that you haven't put a pull-up or pull-down resistor on a reset line that pulls it the wrong way. Sometimes designers will put both a pull-up and pull-down resistor on a reset line so they can pick if they want the device to be default enabled or disabled, but forget to mark one as DNP. This can lead to unpredictable performance, so if you decide to give yourself the option of both a pull-up and pull-down resistor, make sure one is marked as DNP and the BOM reflects that. Similarly, make sure all enable pins are at the correct logic level and have a pull-up or pull-down resistor.

Your circuit should not be in an undefined state when chips are being programmed or held in reset or powered off. Everything should be under control at all times. This is really important for safety critical applications.

Measuring power consumption during bring up is always important. You can usually accomplish this by powering your board with a bench power supply and watching the current draw. If you want to measure current at specific points on your board, you can include a very small value current sense resistor in series with the power supply so it's easy to measure with a voltmeter, allowing you to calculate the current. You can also make the resistor zero ohms and not populate it, allowing you to connect an ammeter in series with the trace of interest. In either case, when you're done you can just put in a zero ohm resistor (for example, during production).

Always include at least one LED on the board. This will be really helpful during your first "hello world" test to see that the microcon-

troller got programmed correctly and has control of at least one GPIO. You can also use a single LED to great effect during embedded software development, since it can tell you when you've hit an interrupt or performed a task. You can also put an LED on each power supply to indicate when it's on and functional. Some power supply ICs have a dedicated "power good" pin that you can use. Another good place for an LED is on data busses, as a way to quickly visually verify that bits are moving (just make sure to set it up so that the LED doesn't interfere with the bus and that it's at the right voltage level). If you have extra GPIO on a microcontroller, stick an LED on it. You don't need to populate the LED, but they're useful for debugging and give you more flexibility for a rework if you decide you need to add another input or output (maybe another enable line or power good signal).

4.4 Ensuring performance

Read the application notes and datasheet. Follow what it says. Application notes are written for your benefit. Now, are all application notes the pinnacle of technical and pedagogical perfection? No. But you can probably learn something from every application note, and often it's information that doesn't appear anywhere else. The component designers have already spent time figuring out how to get the best performance out of their devices (because then they can put those really nice numbers on the front of the datasheet to convince you to use their part!). Follow the schematic and layout recommendations that they provide so you'll have the best chance of actually achieving the performance they advertise. At the same time, their recommendations are not the be-all end-all, and if you know what you're doing, you can make changes that will improve performance for your application.

Check the current limits on any load switches or power supplies. Use your power budget to figure out what the maximum current (and power) through any power supplies will be. Some load switches will further constrain current. Your power supply may be rated to provide enough power, but if your load switch has a lower limit, your device still won't work.

If you are driving multiple loads with a single clock, crystal oscillator, resonator, or RTC, use a clock buffer. If you don't, you may exceed the fanout capability of your output device. This can result in a reduction of the signal's amplitude to below the logic thresholds of your device. An example of where this may occur is when chaining together large numbers of I2C or SPI devices. Your data lines may also need to be buffered, since a large number of devices on the same

wire will present a large combined input capacitance that may make it difficult to drive.

Make sure that the power dissipated in all components is below their thermal and power rating. On parts like amplifiers, you will often need a heat sink when the power dissipated will be greater than the power that the package is rated for. Parts are often available in several different packages, so choose the one rated for the power you need. In general, the physically larger the part, the more heat it can dissipate.

Load switches work well for reducing power consumption by letting you switch out sections of your circuit when not using them. You may be tempted to save money by using a transistor instead of a load switch, but it may have a higher leakage current and it won't include ESD protection.

Make sure to consider the state of all devices at any given time using a flow chart or UML diagram. For example, if the device is in low power mode, it shouldn't depend on the functionality of a feature or state that is not available in low power mode. Devices like microcontrollers disable features in low power modes, so make sure you're not relying on any of those features when they're not available.

If your switched mode power supply has a soft start option, use it. Soft start will slow down the rise time of the output voltage as the power supply stabilizes. This results in less harmonic noise being generated at startup and doesn't stress the battery or SMPS input voltage source by drawing too much current too fast. It can also help reduce inductive flyback.

Adding an in-circuit test mode can be very helpful during PCB bring up and verification, as well as full production. The details of designing a test mode and preparing it for full production are outside of the scope of this book, but in general, any test modes should be able to exercise all device functionality and allow the board to be verified after it is assembled.

Don't use parts at their maximum ratings. That includes maximum voltage, the upper end of their tolerances, maximum current, speed, etc. This will prolong the life of your components, and frankly you're just asking for trouble if you use parts *barely* within their rated range.

All ICs, especially digital ICs, need multiple decoupling/bypass capacitors. This is because larger value capacitors have a low impedance at low frequencies, and smaller value capacitors have low impedance at higher frequencies. To maintain a low impedance to the power supply across a wide range of frequencies, we can put multiple capacitors of different values in parallel with one another. That way as one capacitor starts to increase impedance, the next smaller capacitor will start to

take over. So while in theory a 10 µF capacitor in parallel with a 1 µF capacitor would be the same thing as an 11 µF capacitor, it would actually behave completely differently in real life. Most datasheets and application notes will tell you the recommended bypass capacitors to use (how many, what value, and where to put them). Follow those recommendations to get the best performance.

Note that the chemistry and physical size of the capacitors also matters, so follow the package size, capacitance, and ESR recommendations of the datasheet and application notes. Physically smaller capacitors have less inductance. See the capacitor section in the chapter "Selecting Components" for more information. Other combinations of capacitors besides those recommended will probably work, but to minimize the risk of something going wrong, replicate the evaluation board or application note decoupling capacitor use exactly. That way you will know that you can expect the same performance as what the datasheet claims, since those values were measured using the configuration the manufacturer recommends.

If you use a ferrite bead to help reduce power supply noise, it should be placed in front of the bypass caps (that is, current should flow through the bead before the bypass caps). If the bypass capacitor goes before the ferrite bead, the IC will be unable to draw energy for current spikes from the capacitor because the inductive part of the bead will oppose any changes in current. You might as well not even have a capacitor there. That's why it's critical that the placement and order of ferrite beads is correct, so you don't do your design more harm than good.

Placing a ferrite bead on the output of a switching power supply is a good way to prevent noise created by the switching from getting to the rest of the circuit. Most SMPS datasheets will advise you on what bead to use and where to put it, but generally you want the bead on the output with one capacitor on either side of it.

Since ferrite beads are essentially lossy inductors, they can combine with nearby capacitors and form an oscillating LC tank circuit. This accidental oscillator will start spewing a frequency of $f = \frac{1}{2\pi\sqrt{LC}}$ into your circuit. Therefore, it's important to pick ferrite bead values that will not resonate with capacitors you designed in on purpose or on accident (stray/mutual capacitance). Obviously, every ferrite bead will resonate at least a little because it's impossible to eliminate all stray capacitance, but the goal is to keep this resonance from interfering with your signals of interest and to prevent it from causing EMI/EMC problems. If there's a lot of bypass capacitance next to your ferrite bead, it's a good idea to simulate that network in something like LTSpice to

see if oscillation will occur. Use the SPICE models of the actual parts you'll be using, if possible. It's also a good idea to have a kit of ferrite beads on hand during prototyping so you can measure the actual oscillation and try different bead values until you find one that works well. Parasitic capacitance may make it difficult to accurately simulate your circuit.

If you're having trouble eliminating ringing from a ferrite bead, you can try reducing the effective Q of your accidental oscillator by adding a resistor. The best way to do this without affecting any other performance characteristics is to add it as shown in figure 4.5.

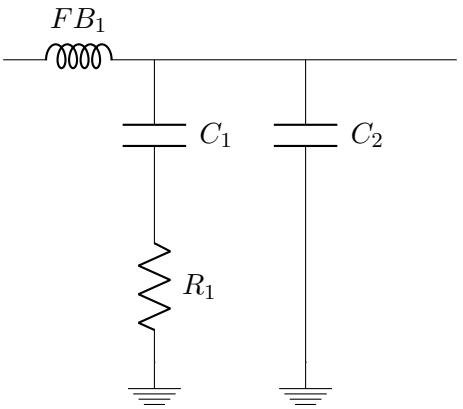

Figure 4.5: Adding a resistor to reduce the Q of the oscillator caused by the interaction of the ferrite bead and the two capacitors.

R_1 is behind a DC blocking capacitor, so there isn't a large voltage drop like there would be if it was in series with the ferrite bead, FB_1. Capacitor C_2 is a small value bypass capacitor (something in the nanofarad range) and C_1 is a much larger capacitor (something in the microfarad range). This reduces the Q of the resonator without significantly affecting the high frequency bypass performance of the capacitors. R_1 should be small, around one to three ohms. You can try simulating it, or using the foolproof method of testing: soldering in the parts and measuring it.

Especially in RF design, isolation between amplifiers is important to prevent unintended oscillation. One way to do this is to put a pi network between each amplifier stage that you can use to add some attenuation. Since oscillation is caused by some of the output getting back

into the input, attenuating the input can help reduce that unwanted output signal below the critical threshold that causes oscillation.

You should also be careful to not put too big of a capacitive load on your amplifier outputs. This applies to all amplifiers, not just RF parts. As the capacitance increases, the phase shift from the output to the input will increase. The closer this loop phase shift gets to 180°, the more likely it is that the amplifier will oscillate. A capacitive load isn't always the result of a physical capacitor: it can also be caused by driving a long cable or another part with a large input capacitance. The best way to avoid this problem is to check the maximum drive capacitance in the datasheet and stay below it. You might also find a graph of small signal overshoot versus capacitive load, and you'll want to make sure you choose a capacitive load that keeps the small signal overshoot low. Once small signal overshoot reaches 100%, congratulations, you've just built an oscillator. If you're designing your own amplifier, the maximum capacitive load you can drive will depend on your gain. It's possible to design an amplifier that can drive *any* output capacitance and not oscillate, and this is known as an unconditionally stable design. Input and output stages are especially important to design to be unconditionally stable.

5

Layout Design

5.1 General

There are a huge number of things to remember, consider, and check before sending a PCB out for fabrication. However, the most important step before sending out your PCB is to have it reviewed by other people. Try to involve people from all across the production process: other electrical engineers, production engineers, mechanical engineers, test engineers, assemblers, etc. It's best to give all of your reviewers a day or two to look things over before the formal design review meeting. Freeze the design after you distribute your schematics and layout to be reviewed, or else you may change something that could end up slipping through without being reviewed.

I like to tell everyone to not review together or share what they've found until the design review meeting. The more independent the reviews, the more likely you are to catch everything. For the same reason, it's dangerous to divide the design up into chunks and have each person look at a different piece. You want as much redundant coverage as possible. When I review designs, I like to print them out on a piece of paper, go to a quiet room by myself with a stack of datasheets for every part used in the design, and work through the whole thing again. It's still useful to use your CAD software to inspect layout features that may be difficult to see on a print out, but for me, looking at things with zero distraction helps me pay attention to detail.

Having engineers from different fields looking over your design will help you catch issues that you may have missed from your perspective, but will be apparent to people in other disciplines. Production engineers and assemblers in particular often have a tremendous amount

of practical knowledge from experience that can dramatically improve your DFM.

You own your design. Do not mistake a design review as a design committee. Take responsibility for your errors, but be the final word on what gets in the design and what stays out. You won't be working in a vacuum of course, and you should absolutely listen to and take the advice of other engineers (especially those in disciples that you don't have as much experience in), but you need to have a good reason for every decision you make. You are the voice of your product during development. Stand up for it. If you know you're right, don't give in. If someone requests a change that you know will be net bad (and you better have a good justification for that), refuse the change. Ten people can't all design one PCB. Ten people may work on it, but there needs to be one design lead who makes the final call.

Good layout is driven by good placement. Good routing should naturally fall out of your placement choices. Beginner designers will often place parts in a fairly haphazard way and spend the majority of layout time on routing traces. Instead, you should spend more time thinking about where to place parts, what rotation they should have, which side of the board they should go on, etc.

A good design is maintainable and repairable. Even though your pick and place machine can place parts with microscopic precision, think about how hard it will be to replace components by hand if something needs to be fixed. How good do your tolerances need to be? Can you fit a soldering iron down in there? How many screws do you have to remove? Do wires need to be desoldered to get at anything? If you assemble your engineering prototypes by hand, you will quickly discover any shortcomings in maintainability and repairability. Sometimes it's impractical to assemble your engineering prototypes by hand, or you'll need a technician to do it. Talk to your technician. Ask them what's frustrating to repair or rework so you can incorporate their feedback into your next revision. In general, be careful about placing components too close together or in such a way that they will interfere with tools that need to be used on the PCB.

There are also mechanical placement requirements to think about. Of course, there's the obvious examples like component height, connector overhang, and mounting and screw holes. There are also more subtle things, like how far the leads on the bottom of through hole parts stick out. One good way to detect and avoid these problems is to use a CAD package with 3D modeling support. Most electrical CAD programs support 3D component models and even allow you to export to a mechanical CAD program (like SolidWorks or Catia) so your

mechanical engineer or industrial designer can do a fit check before fabrication. Using this technique requires accurate and complete 3D models of all components, but many manufacturers provide 3D models of their components you can download for free.

One potential trip up for new designers concerns the units commonly used in PCB layout. When someone says "mils", they mean 0.001 inches (one thousandth of an inch). If you've ever worked in a machine shop, you may have heard this unit referred to as a "thou", short for thousandth. The mistake that some people make is that they think mil is short for millimeter. So in the world of electronics design, it's important to remember never to shorten the word millimeter to mil or you will really confuse people. The mechanical drawings in datasheets for the component are almost always in mils, millimeters, or inches. A good drawing should tell you what the units are, so make sure to pay attention.

If your design requires any traces have a particular impedance, tell your PCB fabricator to use impedance control on those traces. You'll first calculate the correct transmission line dimensions using a software tool like LineCalc (made by Keysight) or one of many free online impedance calculators[1]. However, the fabrication process will have some variance on the width, plating thickness, and spacing. Specifying impedance control will tell the manufacturer to take those variances into account and precisely hit a target impedance. Request a test coupon so you can verify that the impedance is correct.

Make sure you understand the manufacturing abilities of your PCB fabricator and design with those constraints in mind. Manufacturers are able to fabricate extremely small trace widths, complex via structures, and other exotic features. However, it will be very expensive. If you know what standard manufacturing specifications your PCB manufacturer offers, you can get your boards back at a lower cost and in less time. For example, say you design a trace that is 9 mils wide, but it would also be acceptable if it was 10 mils wide. If your manufacturer has a standard or low-cost tier that requires a minimum 10 mil trace size, your design will be more expensive than necessary because of the difference of a single mil, which isn't important for your design anyway.

PCB fabricators enforce design rules so that they can be confident that they're able to produce your board to meet their quality, yield, and time standards. It's really important to set up the design rules in your CAD software to mirror those of the fabricator you'll be using. These design rules usually appear in a table on the manufacturers

[1] You can find links to these calculators on the website for this book, DesigningElectronics.com

website. If your manufacturer has a DRC or design constraint file for the CAD package you're using, download it and use it. This will automatically import the minimum feature sizes and spacing requirements for the particular process you'll be using. Some manufacturers even have dedicated tools you can use to check your gerbers before you submit them. For example, Advanced Circuits has a website called FreeDFM.com that allows you to upload your gerbers and automatically look for and identify errors before you submit them for fabrication. While FreeDFM.com is only set up to check against Advanced Circuit's rules, those rules are fairly standard and it can still be useful for finding mistakes even if you don't ultimately fabricate with Advanced Circuits.

If you'll be designing a board with high-speed or RF requirements, look at the available stackups and materials of your board house before designing the impedance-controlled structures. Knowing what stackups are available can significantly impact your cost. Exotic or custom stackups are often much more expensive than the standard stackups a board house may use. They often also have much longer lead times. In many cases, adapting to the stackup that is available is not very difficult and may only change your layout slightly. But those small changes can save thousands of dollars and weeks of time.

Use the same PCB manufacturer that you will use for the final production PCB if possible. At least use a manufacturer capable of producing boards in the same class. Most PCBs are, by default, class II per the IPC standards body. Absolutely do not do a production run without fabricating a few prototypes first, even if you've proven the design with a different fabricator. Not all PCB manufacturers are created equal. Inspect the prototypes carefully. Pay attention to how close via holes are to the middle of the pad, look for any smeared copper or shorting, and check how evenly the solder mask is applied. If possible, verify that they are actually meeting the IPC class that they claim they are hitting. Some factories (especially in China) have been known to lie about being IPC certified or will show you an expired certification.

When laying out traces, use chamfered or mitered turns. The default trace routing style in EagleCAD is diagonal, so many novice boards have traces that run all over the board and look more like they're drawn freehand. Instead, using chamfered traces keeps things neater and helps enforce the good routing style of keeping intra-layer traces roughly parallel and inter-layer traces roughly perpendicular.

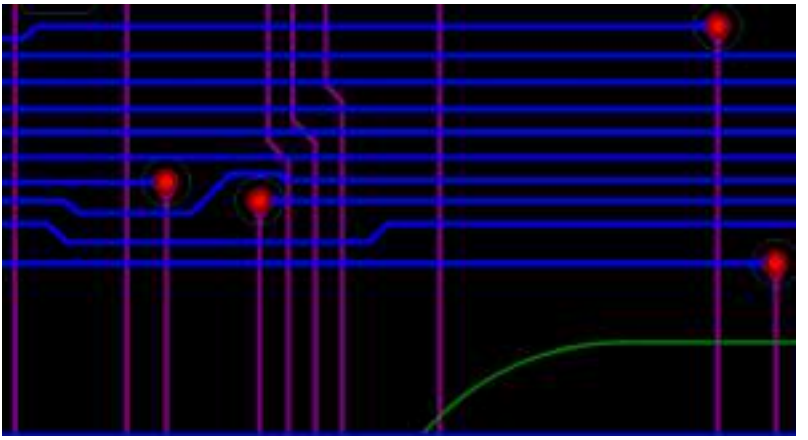

Figure 5.1: Traces on different layers are kept approximately perpendicular to each other. In this image, the purple traces are on one layer and the blue traces are on another layer.

If you ask some engineers why you should use chamfered turns, they will tell you probably one of two reasons. The first common explanation is that sharp angles will cause etching problems during manufacturing. This used to be true, but modern manufacturing techniques are good enough that this isn't really a concern anymore. The second reason you'll hear is that right angles cause reflections or signal integrity problems. But there are only certain conditions under which a right angled turn on a trace will cause signal integrity problems. Much has been written on this topic, but basically what happens is that a right angled turn adds just a little bit of extra capacitance to the trace. How much extra capacitance it adds depends on the width of the trace and the PCB substrate. In many cases, this extra capacitance is so small that it isn't noticeable. You can estimate the amount of extra capacitance due to a single 90 degree turn using this rule of thumb: 1.7 times the width of your 50 ohm transmission line in mils is the additional capacitance in femtofarads. For example, if your 50 ohm transmission line is 20 mils wide, then the extra capacitance from a 90 degree turn in that trace will be about $1.7 * 20 = 34$ femtofarads. Whether your design will care about an extra 34 fF depends on the rise time of your signals and their frequency. The graph in figure 5.2 shows the return loss of several example transmission line right angle turns over frequency. You can see that for narrow transmission lines, performance doesn't start to suffer until frequency is quite high. For wide transmission lines, this happens much sooner, but still is probably at a high enough frequency

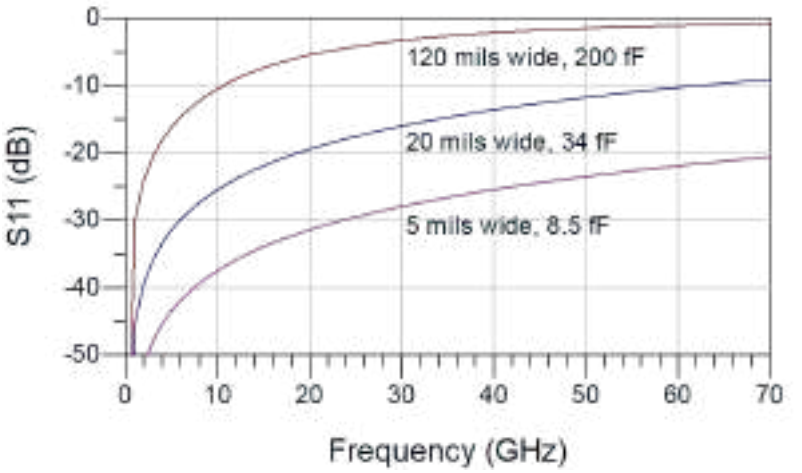

Figure 5.2: Simulation of the return losses of three different example transmission lines with a 90 degree bend.

that many designers won't need to worry about it.

Note that this rule of thumb was derived assuming the substrate dielectric constant is somewhere around 3.5 to 4, so it holds for substrates like FR-4 and RO4350B. If you use a substrate with a lower dielectric constant than that, the extra capacitance will be even less. If you use a substrate with a higher dielectric constant, the extra capacitance will be higher than the rule of thumb estimates. Another important thing to remember is that this is only the capacitance for a single 90 degree turn. If you meander a line back and forth with 90 degree turns, all of that capacitance will add up and can become a problem.

Even if your fundamental frequency is low, fast enough rise times can also be affected by 90 degree corners (since a signal with a fast rise time will contain high frequency components). The rule of thumb to use for that is if the width of your 50 ohm transmission line in mils is greater than 5 times the rise time in picoseconds, you should not use 90 degree trace turns. You should also avoid unnecessarily fast rise times when possible for EMI reasons. That's discussed further in the "EMC and EMI" section of this chapter.

So most of the time, right angle turns are fine. I happen to think chamfered turns look better and make routing easier, so that's why I use them. Like in the case of schematic capture, take care to perform layout in an aesthetic and organized manner. Keep things symmetric.

Do not do this at the expense of performance, however. Only after all of the performance requirements have been met should you make changes that visually clean things up. Check to make sure your changes don't subsequently affect performance.

Use a consistent grid unit during layout. That way you don't end up with any components offset from one grid, which will prevent annoying alignment problems during routing.

Drawing footprints for components can be time-consuming and error prone. Some component manufacturers provide footprints for you to download that you can trust as being correct. They will sometimes provide these files in a format designed to be read by a program called UltraLibrarian. UltraLibrarian is a free program that takes in a footprint in a single file format and allows you to export it into the format of whatever CAD program you're using. The user interface is a little dated and confusing, so be sure to read the readme file that comes with it.

Another place to find footprints that are already drawn is a website called SnapEDA. SnapEDA provides schematic symbols, footprints, and 3D models for millions of parts and will let you download those files in the native format of whatever CAD program you're using. If they don't have the footprint you need, you can pay them a small fee to draw it for you. You can find a link to SnapEDA and other resources on this book's website, DesigningElectronics.com.

Whenever you download a footprint that you didn't draw, it's a good idea to double check it and make sure it's correct. While a datasheet may have only a single recommended footprint, there can be multiple correct footprints for the same part. Some manufacturers will offer several different versions of the same footprint: small, medium, and large (sometimes called high density, medium density, and low density). The difference between these will be how much extra space each pad has around the pin itself. A smaller version of the footprint is good if space is at a premium and you can get away with using only a small amount of solder paste. A larger version of the footprint is good if you're going to be assembling the board by hand and need to get a soldering iron tip on the pads.

To improve silkscreen resolution, use two different layers of different color solder mask instead of standard silkscreen. You'll need to talk to your PCB fabricator about this, but solder mask is placed at much tighter tolerances and has sharper edges than normal silkscreen. This is the technique that Arduino PCBs use to achieve the fine visual detail that make their boards look great.

If your board has silkscreen labels, set the font size small enough

so that component designators fit next to their component, but not so small that the printer resolution is unable to clearly print it. Make sure all designators and text can be read in one of two directions. This will keep you from having to rotate the board constantly when trying to read the silkscreen. Keep silkscreen off of vias and pads so that it doesn't get cut off. Be sure to label the polarity of any polarized parts like electrolytic capacitors and diodes.

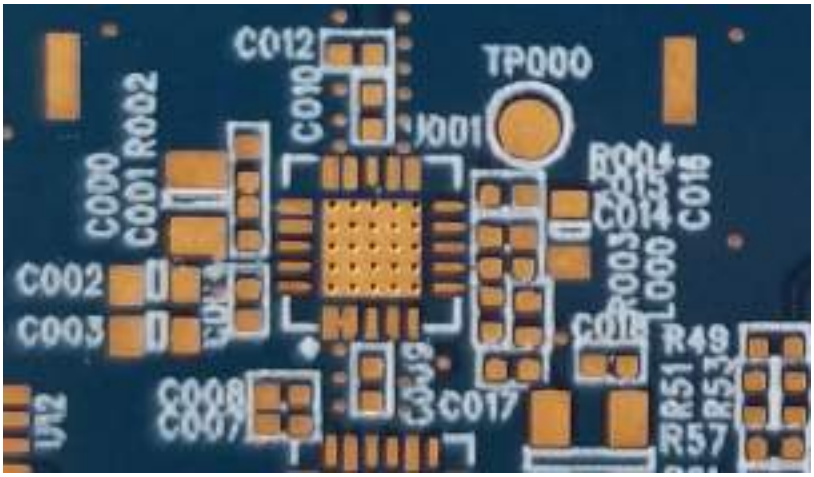

Figure 5.3: Keep silkscreen rotated in one of two directions so it's easy to read without spinning the board around too much.

Figure 5.4: You can use a small rectangle of silkscreen as a label.

Figure 5.5: Version number and other markers for this PCB revision. In this case, they were put on a copper layer instead of in silkscreen.

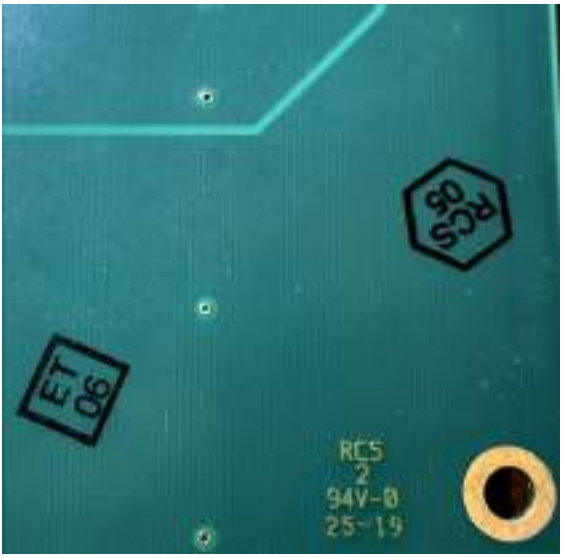

Figure 5.6: PCB fabricators will also usually put some markers for their own reference on your board. You can specify a side or location for them to put these if you want.

It's a good idea to put the PCB version and date in silkscreen so you don't confuse multiple iterations of the board. You can also put a small rectangle of silkscreen in a corner somewhere so you can write notes or label the PCB with a marker (e.g., a serial number).

If your board has any replaceable parts like fuses, mark the type and rating of the component in silkscreen next its location on the PCB so it's easy to know what replacement to use.

Most EDA packages have something called an autorouter. This may seem like a great way to save time by letting the computer do the often tedious job of layout for you. But do not be fooled: autorouters universally suck. They will make poor decisions about where to route things even if your placement is good, and they just don't know about all of the design considerations and requirements that you, the designer, know. I do not recommend using any autorouter.

Any traces that carry high current need to be wide enough to carry that current without dropping the voltage significantly. There are lots of trace width calculators available online. In some cases, a pin on an IC that needs to carry a lot of current may be smaller than the width of the trace needed to safely carry that amount of current. The best way to deal with this is to increase the width of the trace as quickly

as possible to "neck it up". If there are any layer changes along your high current paths, you'll need to use multiple vias at each transition to ensure low loss. Another thing to consider is the weight of copper to use. A trace made of 1-ounce copper will carry less current that the same trace made of 2-ounce copper. A more complete explanation of copper weight is in the "Stackup" section later in this chapter.

If your product will be mass-produced, think about how quality will be controlled. Inspection and test are the two most common ways to verify quality, and both of those techniques require thought during the design phase. Are there critical parts of the design that need to be visible to inspectors? Inspection of components usually happens along with assembly, so that subassemblies can be inspected before they are integrated into the rest of the design. Think about how your device can be systematically tested and inspected during assembly so that nothing needs to be unscrewed and the right parts are tested in the right order.

5.2 Performance

Don't share ground vias on capacitors. This is especially critical in RF and high-speed circuits. Having a single via to ground shared by multiple bypass capacitors adds unnecessary inductance and reduces the performance of the bypass capacitors. Remember: extra vias don't cost you anything! Use them liberally.

Like bypass capacitors, any resistors used for feedback should be placed as close to the IC as possible. The reason here is the same: at high frequencies, longer traces will add stray inductance and can change the performance of your circuit. Longer traces can also act as loop antennas and pick up noise or crosstalk from elsewhere on the PCB.

Resistors will actually change value when flexed. Bending a PCB, especially a thin substrate, will also slightly bend the resistors that are mounted on it. If the resistors are high tolerance, the simple act of flexing the PCB can throw the resistor value out of spec. If your design has high tolerance resistors, check to make sure that they won't get flexed when mounted, or during normal use.

If you're feeding an antenna, add a small pi network at the point closest to the antenna. This makes it easy to add a matching network if your antenna doesn't look like exactly 50 ohms. Pi networks are also easy to bypass, since you just need to add a single zero ohm resistor to the series component. When you're laying out your matching pi networks, make sure to keep the ground vias as close to the component pads as possible. Placing them further away can add stray inductance

that makes tuning more difficult. This is also why all of the components in the pi network should be placed as close together as possible.

Your designs should always include a ground plane. This will prevent a myriad of problems, and it significantly simplifies routing. Instead of drawing a trace back to your ground net, you can simply drop a via. Where and how big of a ground plane you use will depend on your design limitations. Ideally, a two-layer PCB will have a ground plane on the back side and everything else on the front. However, this isn't always feasible. In that event, at least keep ground planes under critical ICs and nets. Resist the temptation to flood fill a ground plane and then start running traces through it when you run out of room on the top layer. This will effectively create holes in the ground plane copper that can cause EMC problems.

If your design has parts on both the top and bottom of the board, check for instances of any large solder pads on the top layer that sit directly above any large solder pads on the bottom layer. This can cause heat transfer problems during reflow. Similar thermal problems can occur during the product's lifetime if both chips are trying to sink heat into each other. Instead, offset the footprints sightly.

Throughout this book and literature in general, you'll see people referring to some designs as high-speed. But what defines "high-speed"? Signals above 1 GHz certainly seem high-speed, and I've even heard people refer to a 3 MHz signal as high-speed. In reality, for signal integrity purposes, the rise time of the signal and the length of the trace it's traveling down are what matter. If the rise time of your signal is less than the time it takes for that signal to propagate to the end of the trace, you should consider it high-speed. To understand why this is, consider a square wave. To make a perfect square wave, you need infinite odd harmonics of sine waves. As you remove more and more of the high frequency odd harmonics, the straight edge of the square wave begins to droop, and the rise time gets lower and lower. So, while you may be routing a clock line that is only operating at 1 MHz, if it has extremely fast rise times, it contains very high frequency harmonics. For sine-wave-like waveforms, the rise time is roughly one third of the frequency. For these kinds of signals, you should consider the threshold of high-speed to be when the wavelength of the signal is more than one fifth of the length of the trace.

For most applications, an extremely fast rise time is not necessary. In fact, unnecessarily fast rise times are one of the major causes of EMI problems (see the "EMC and EMI" section at the end of this chapter for more information). To reduce the rise time, simply add a series resistor. Start with something like a 60 ohm resistor and check your

new rise time on an oscilloscope. The longer the trace, the higher the resistor value will need to be. It's a good idea to drop a series resistor footprint on signals you suspect may need to have their edges slowed down, and if you don't end up needing it, you can just populate a zero ohm resistor. The other way to move your signal out of the high-speed regime is to simply shorten the trace. The easiest way to do this is by changing placement of components, but this isn't always possible. It may be necessary to add more layers or even use HDI (high density interconnect) processes like blind and buried vias. See the next section on "Stackup" for more information.

If your signal has a wavelength shorter than a fifth of the trace length, you'll need to do impedance matching. Impedance is a property of a trace that is fully dependent on the physical properties of the PCB: the trace height and width, the PCB material, the number of layers, the distance between those layers and the trace, etc. There are calculators online that can help you plan out the dimensions you need to arrive at your correct final impedance. These typically work in two directions: you can type in the dimensions that you have and the calculator will tell you the resulting impedance, or you can give it your desired impedance and it will calculate the trace width required to get there. Some ECAD programs have built in tools to calculate impedance, or you can use simulation tools like Keysight ADS to model your PCB, including vias and transmission lines.

Do not simply split or fork high-speed signal traces. This includes clocks, data lines, and RF traces that are running above 10 MHz. Splitting these signals without special consideration can cause reflections that will degrade performance (or in the worst case, destroy the driving circuitry). Additionally, many clock and data drivers simply can't drive more than a single output. If you need to split a high-speed signal, use a buffer. If you need to split an RF signal, use an RF splitter. Make sure the component you choose is rated for the frequency at which you're using it.

Slow signals can be probed by simply touching an oscilloscope, multimeter, or logic analyzer probe to test points or component pads. However, RF signals cannot be reliably probed this way. Instead, you can add a U.FL connector or another similar small RF connector that is connected with a removable zero ohm resistor. As mentioned before, you can't just add a fork to a high-speed trace and expect it to still work the same way, so you need to make it easy to terminate the transmission line into a connector while disconnecting it from the rest of the circuit past the connector. A pair of zero ohm resistors as shown in figure 5.7 is a good reversible way to do this.

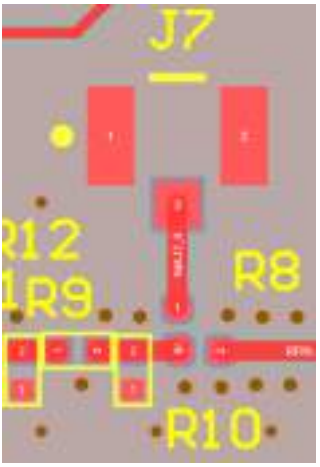

Figure 5.7: In this layout, two resistors are used to switch the direction of an RF signal into a U.FL connector (J7) and a coplanar waveguide. If the vertical resistor below the connector is installed, the signal is directed into the connector. If instead the horizontal resistor on the right side of the connector is installed, the connector is bypassed and the signal continues along the transmission line.

A directional coupler is another way to safely sample an RF signal. A directional couple can be a physical part that you solder in, or you can create a directional coupler in copper. By running a trace next to your transmission line of interest, you can sample a low amplitude version of whatever is on your transmission line.

Beware of crosstalk between traces that are physically close to each other. For sufficient isolation between two microstrip traces, make sure the distance between them is at least four times the thickness of the dielectric between the microstrips and their reference ground planes. To prevent crosstalk or coupling between two striplines, keep them apart by at least two times the thickness of the dielectric between the striplines and their reference ground planes. Making a little fence of ground vias between two traces is another way to keep them isolated. Make sure that the distance between the vias is no larger than $\lambda/20$, where λ is the wavelength of the signal you're trying to isolate.

Remember that when you're calculating the wavelength, you need to take into account the medium in which it's propagating. Real PCB substrates have a velocity factor. For example, in FR-4, electromag-

netic waves will propagate about 40% slower than in free space, which will make the effective wavelength shorter. This is especially important to keep in mind when considering whether a trace or feature is a significant fraction of wavelength. To calculate wavelength in a medium, use the following equation:

$$\lambda = \frac{300}{f\sqrt{\epsilon_r}} \text{ (in mm)}$$

Inductors and coils create a magnetic field that goes through the middle of the windings. That field can interfere with the field of other nearby inductors, so make sure to place inductors that are close to each other at right angles to each other. This will keep them from coupling to one another, since their fields will be perpendicular. Other good ways to reduce the coupling between inductors are to keep them far apart, and to use shielded inductors.

When placing components, try to clump parts together by function. For example, your analog circuitry should be confined to one area, your digital circuitry should be confined to a different area, your power supplies should be confined to a third area, etc. This is useful for a couple of reasons. It will help prevent crosstalk and noise from coupling into different subsystems. It will make troubleshooting, board bring up, and testing easier, since you can isolate, modify, and replace parts in a single area instead of all over the board. It also makes routing much cleaner. In fact, it's almost impossible to correctly lay out a PCB if you spread parts from all systems all over the board. When you finish the schematic and begin to place parts, the first thing you should do is arrange the parts into groups. After that you can take a first pass at placement and begin seeing where things have to fit on your board outline by dragging around your roughly placed groups.

If you have multiple radios on a single PCB, place them far apart, especially if they will be transmitting at the same time and if their antennas are on the same PCB. This will help prevent interference between the radios and keep the antennas from detuning.

If you decide to remove solder mask from parts of your PCB that contain ICs, try leaving a few mils of solder mask around the component footprint(s). This will help prevent solder paste from wicking off of the pads and onto the surrounding exposed copper. For example, in figure 5.8, solder mask has been pulled back from over top of a coplanar waveguide, but not all the way up to the pin of the chip.

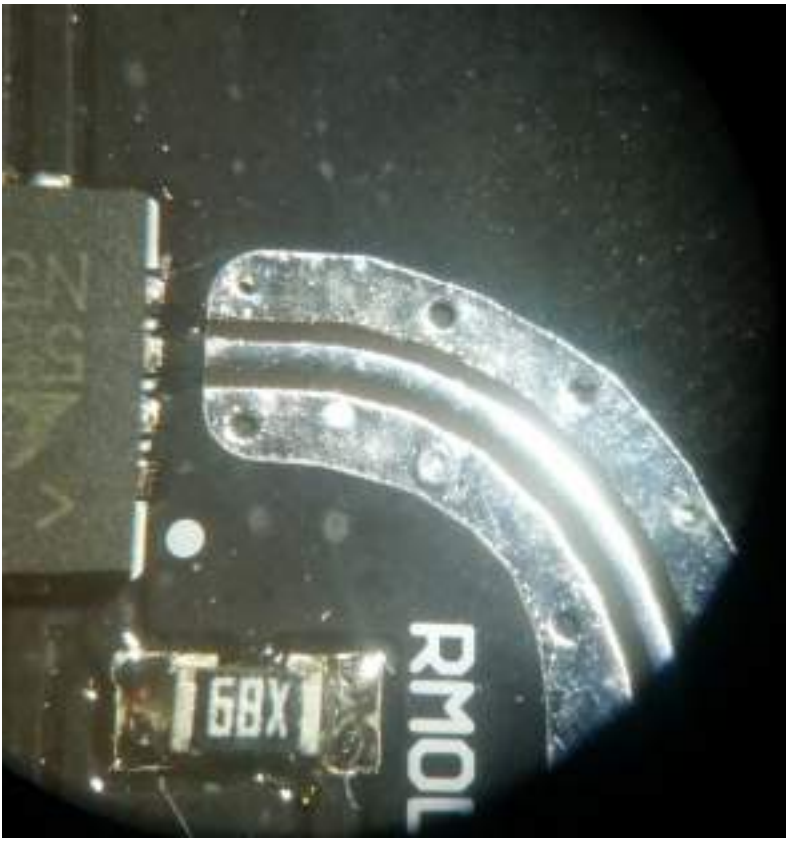

Figure 5.8: A small amount of solder mask prevents solder paste from the IC pad from wicking down the coplanar waveguide that it is feeding.

The only thing that should be routed under a switched mode power supply (SMPS) are traces used for that power supply. The possible exception to this rule is if there is a full ground plane between the SMPS and your non-SMPS traces. SMPSs create noise as a result of the switching FET. If the FET switching looks roughly like a square wave (and it often does), then it also generates higher frequency noise from those waveform edges. Running traces under the power supply can couple that switching noise into those traces. SMPSs should be placed away from sensitive parts of your design. Similarly, other sensitive circuits should not have anything routed underneath them unless they are separated by a solid ground plane.

Don't route digital and analog lines close together. They can couple noise and transients into each other, especially if the rise time of the digital signal is fast. If you have really sensitive analog signals you want to protect, you can separate your ground plane into two parts: an analog ground and a digital ground. If you do this, you need to do it correctly or you'll end up with terrible EMC problems. You need to connect your analog ground to your digital ground at a single point, and don't route any traces over top of the slits in your ground plane. You'll also want to keep the distance between your split ground planes at least 20 times the height of the signal traces above the ground planes on the layer beneath them. More information about this is in the "EMC and EMI" section later in this chapter. You'll most commonly encounter analog/digital coexistence issues when designing with ADCs. Be sure to follow the layout guidelines in the datasheet, especially for fast or high resolution ADCs.

It's usually better to route power to your analog circuitry with traces, rather than using a power plane that sits underneath everything. Sensitive analog lines may capacitively couple to the power plane, which is probably going to contain ripple. Instead, choose a stackup that puts a ground plane directly under your analog parts and then route individual traces to each chip for power.

Route the clock and data lines of any communication busses together. This gets more important as the data rate (and therefore the clock rate) increases. Eventually, if the clock and corresponding data lines are significantly different lengths, there can be phase offset problems and your clock and data edges won't align correctly at the receiver.

Test points are free, so use them! Place them on the input and output of any power supplies, data lines, clock lines, enable lines, etc. If a chip has a "power good" or other status output that you're not using, route it out to a test point anyway. This will help when debugging and also makes reworking easier later if you decide to use that signal for something. You can either use an exposed circle of copper as your test point, or you can use a test point that gets soldered on to make it easier to clip into. The Keystone 5018 and 5029 are great surface mount test points, and the Keystone 5001 is a great through hole test point.

If your PCB has components on only one side, consider placing test points on the back (the side without components). This makes building a test fixture easier, since you can flip the board upside down and use a flying lead tester, a bed of nails jig, or handheld probes without having to worry about components getting in the way. It's easiest to support the PCB along its edges, so make sure to leave clearance along the edges

of the board. Additionally, when placing test points, it's helpful to place a small ground pad next to the test point. This is really nice when you're probing signals on your test points because you typically need a ground connection. Having a ground connection close by reduces stray inductance and prevents your probe from acting like a loop antenna. Most oscilloscope probes have a small spring-looking attachment that is designed to be used exactly for this purpose. Instead of scraping off solder mask when you use it, you'll have an easy ground pad right there. When placing test points, make sure they're far enough apart that all test points can be accessed at the same time by a technician or test fixture. When deciding which signals to make available on a test point, think about your test plan (more information on that in chapter 8). Ensure that all of the signals you'll need to verify and test your design are easily accessible.

Request that the company that makes your PCBs include a test coupon, especially if your design has any impedance-controlled signals. A test coupon is a PCB that has alignment features, copper layers, and usually a transmission line or differential pair on it that allows you to verify that the stackup they used was correct and that the impedance was controlled well enough. If you just request a test coupon, the PCB fabricator will use a standard test coupon design that they have. You can also design your own test coupon if there are specific features or specifications that you want to verify. A test coupon is useful for single panel production runs as well as mass production, when you want to know that a panel has a manufacturing defect before you waste time and money populating and testing those boards.

Along with a test coupon, you can also add a test stack to the edge of your PCB panel. A test stack has a pyramid-type structure of the stackup that you can use for sectioning and analysis to be sure the fabricator hit the tolerances you expected and that the stackup is correct.

Another thing you can add to a PCB panel is a bad board marker. This is a small silkscreen square by each PCB in the panel that you can mark with a permanent marker to denote that a particular board is bad. This is useful when you populate a panel and test each board before separating individual PCBs from the panel. If any particular board fails the test, you can mark that it's bad so that when the boards are separated, the ones that failed the test can go back to be reworked.

ENIG plating should not be used on RF traces because it contains nickel. Nickel is much more lossy than copper, and because the nickel is on the outside of the trace, the skin effect will cause most of the RF signal to be concentrated in the nickel. The result is a transmission

line with more loss than you would otherwise expect. Figure 5.9 shows what an ENIG plating looks like.

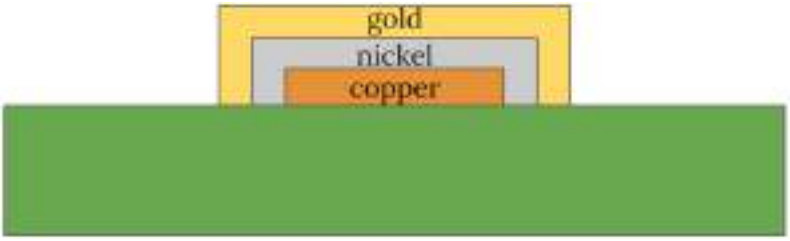

Figure 5.9: Diagram of ENIG

To give you an idea of the relative loss of ENIG at high frequencies, the tables below[2] shows some examples of the loss caused by other finishes besides ENIG.

	ENIG Insertion Loss (dB/inch)	
Frequency	Microstrip	Differential Pair
1 GHz	0.10	0.50
2 GHz	0.20	0.80
3 GHz	0.30	0.90
4 GHz	0.38	1.25
5 GHz	0.48	1.50
6 GHz	0.50	1.75
7 GHz	0.60	2.00
8 GHz	0.69	2.50
9 GHz	0.79	2.75
10 GHz	0.90	3.10

Figure 5.10: Insertion loss in ENIG versus frequency for both a microstrip and a differential pair.

[2]Taken from Wu, Xin, et al. "Surface finish effects on high-speed signal degradation." IEEE Transactions on Advanced Packaging 31.1 (2008): 182-189.

	Sn60Pb40 (HASL) insertion loss (dB/inch)
Frequency	Microstrip
1 GHz	0.10
2 GHz	0.18
3 GHz	0.28
4 GHz	0.30
5 GHz	0.40
6 GHz	0.50
7 GHz	0.59
8 GHz	0.65
9 GHz	0.73
10 GHz	0.89

Figure 5.11: Insertion loss in HASL versus frequency for a microstrip.

	Bare Copper insertion loss (dB/inch)
Frequency	Microstrip
1 GHz	0.09
2 GHz	0.18
3 GHz	0.22
4 GHz	0.30
5 GHz	0.40
6 GHz	0.45
7 GHz	0.55
8 GHz	0.60
9 GHz	0.70
10 GHz	0.80

Figure 5.12: Insertion loss in bare copper versus frequency for a microstrip.

	OSP insertion loss (dB/inch)	
Frequency	Microstrip	Differential Pair
1 GHz	0.10	0.25
2 GHz	0.18	0.35
3 GHz	0.28	0.45
4 GHz	0.30	0.50
5 GHz	0.40	0.60
6 GHz	0.50	0.75
7 GHz	0.59	0.80
8 GHz	0.65	1.00
9 GHz	0.73	1.20
10 GHz	0.85	1.25

Figure 5.13: Insertion loss in OSP versus frequency for a microstrip.

If your design requires adhering to a standard (especially a medical standard), you'll encounter something called creepage and clearance. Creepage distance is "as the snail crawls". In other words, if you were to put the tip of a pencil on one object and move it without lifting the tip to another object, that would be the creepage distance. The clearance distance is "as the crow flies". In other words, if you use a pair of calipers to measure the distance between two objects, that's the clearance distance.

The main reason that there are creepage and clearance requirements is to prevent arcing. Arcing can happen by several different mechanisms. First, dust can be electrostatically attracted to traces at high voltage. This dust can harbor moisture and cause either an arc or a weak short. Another way arcing happens is just through dielectric breakdown of the air or whatever material is separating two points with a high potential difference. If you follow the IPC or IEC standards for creepage and clearance, you won't have to worry about arcing.

5.3 Stackup

Stackup design is one of the most important parts of designing electronics, but it is very rarely discussed in school and lots of electrical engineers have a poor understanding of its nuances. This puts them

at a real disadvantage, since having insight into stackup design can save you a lot of time and money during fabrication, will improve your design performance, and make your layout much easier.

Just as important as the components that go on a PCB is the PCB itself. The combination of the PCB materials and physical dimensions of those materials, the traces, and other circuit features is called a stackup. The first place to start when specifying a stackup is the number of layers. The more layers you use, the easier the routing will be, but the more expensive it will be to fabricate. Use as few layers as you can reasonably get away with. High-speed and RF designs often need at least 4 layers so you can dedicate a single layer to an uninterrupted ground plane, which is important for signal integrity reasons. Using a layer as a power plane can also be convenient, since routing power lines becomes as simple as dropping a via to the power plane. The large surface area of a power plane that is adjacent to a ground plane can also create a little bit of helpful capacitance and help reduce ripple. Whatever number of layers you choose to use for a design, make sure to stick to an even number of layers. You can fabricate an odd number of layers, but to do that, your fabricator is just going to make an even number of layers but just put no copper on one of those layers. So you might as well route on it.

For most designs, the extent of stackup designing that you'll need to do is just to pick the number of layers you want, and you'll be fine using FR-4 as the substrate. Whatever board house you use will have a standard stackup that they use for this.

You may start to get into more complex stackup design if you're designing a board with high-speed or RF signals on it. The dielectric constant of FR-4 may be such that the dimensions of your transmission lines will be impractically large or impractically small. FR-4 is also quite lossy at high frequencies. Special substrates are necessary for high frequency designs (and can be quite expensive). For these designs, the most important properties to consider are the dielectric constant (also known as Dk or ϵ_r), and the loss tangent (also known as δ), sometimes called the dissipation factor (or Df). The dielectric constant (or permittivity) is roughly a measure of how a given material stores an electric field. The permittivity of air is one, and you can buy PCB substrates with a dielectric as low as two or as high as 10. The loss tangent of a material is how much of an electric field is dissipated as heat, so a lower loss tangent is better, especially at higher frequencies. Rogers, Megtron, and Isola are examples of companies that make a very wide range of substrates that can have a very small loss tangent and almost any Dk value between 2 and 10 that you want. Take a

look at these if you're trying to design an antenna, millimeter wave, or high-speed circuit.

If you're looking for a place to start, some popular RF substrates include FR408HR and 370HR from Isola, and 4350B and 4003C from Rogers[3]. Rogers offers free samples of their substrates, which are great for milling or etching on. Before you choose a substrate, check if your PCB fabricator has the material in stock. While lots of different options are out there, only a few are commonly kept in stock with most fabricators. If they don't have the material you want in stock, you'll have to wait for it to ship (which adds lead time). Alternatively, you can tell your fabricator beforehand that you'll be submitting an order using that material and they are typically happy to order it ahead of time. If you want specific guidance for what material to use, I highly suggest calling or emailing the substrate company you're interested in. Personally, I've had excellent support from Rogers. I was able to talk to an application engineer quickly and he even offered to review my stackup and make suggestions. Your PCB fabricator can also review your stackup and make suggestions to improve cost and manufacturability.

If the calculations for your transmission line show that the conductor must be impractically wide, picking a substrate with a higher dielectric constant typically allows you to reduce the width of the conductor, which may make the board easier to lay out. A higher dielectric constant can also be beneficial for antenna design, especially in reducing the size of the antenna (but note that a higher Dk also causes higher insertion loss, since more of the fields are contained within the substrate). Conversely, picking a lower dielectric constant allows you to increase the width of the conductor, which can be helpful if your initial calculation shows that your trace needs to be extremely thin. The other thing you can try changing is the thickness of the dielectric above and below your impedance-controlled trace. When designing impedance-controlled structures, make sure to use the dielectric constant in that datasheet that is marked as the design value, not the

[3]Rogers also has a substrate in this same 4000 series family called 4835, which costs about the same as 4350B and 4003C, but will oxidize slower. This only really matters for designs that must operate for many years, so you don't need to worry about this if you're just prototyping. Nevertheless, Rogers recommends 4835 for new designs. Rogers 4003C is very similar to 4350B and 4835, but isn't halogenated. This means that it doesn't have the same fire rating (4835 and 4350B are both UL 90 V-0), but it does have a slightly lower dielectric constant because halogen fire retardant molecules (in this case, an organobromide) are polar, which shifts the dielectric constant up.

process value. Many PCB substrate companies will provide two different Dk numbers: one that can be used during PCB production, and one that should be used by designers to calculate trace geometries. If you mix these up, you may end up with a PCB trace that doesn't have the impedance you want.

Both the dielectric constant and loss tangent are frequency dependent. Usually they're quite wide band, so if the datasheet only gives you the dielectric constant at 1 GHz, it will probably be about the same value at 2 GHz. Don't use the value that's in big font on the first page of the datasheet. Find the curve showing Dk or δ over frequency and read off the value from there.

When designing a custom stackup, layers can be made of two different types of materials: core and pre-preg. Core substrate is a dielectric material with copper plating on both sides. Pre-preg (which is short for pre-impregnated) is a dielectric material that is partly resin cured and does not have any copper plating. After your design is etched, the layers are stacked together in the order you specify (hence, a "stackup"). One stackup approach is to use pre-preg to separate sheets of core material, basically using the pre-preg sheets like glue.

But this is not the only way to construct at stackup. You can also put multiple sheets of pre-preg on top of each other. Since core material comes copper plated and pre-preg doesn't, you may think that you can only route on core layers. However, it's completely possible to route on pre-preg layers as well. That's because during construction, layers of copper foil can be laid down between sheets of pre-preg.

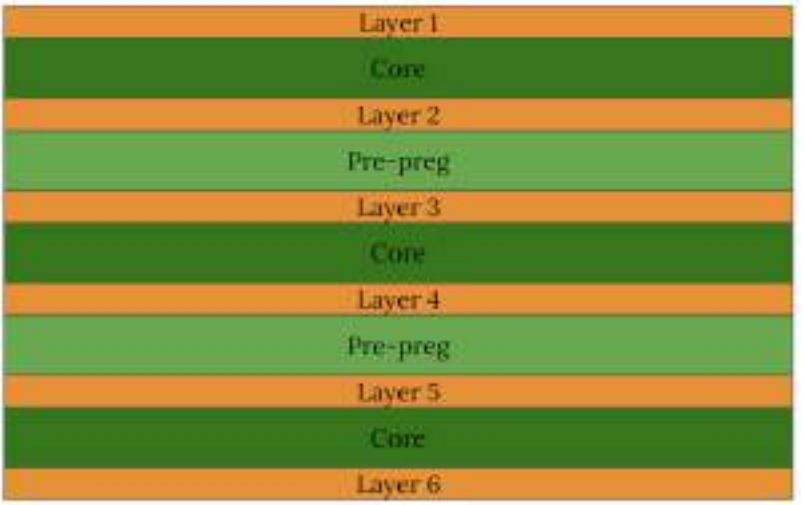

Figure 5.14: Example stackup of a six-layer board that alternates core and pre-preg starting with core on the outside.

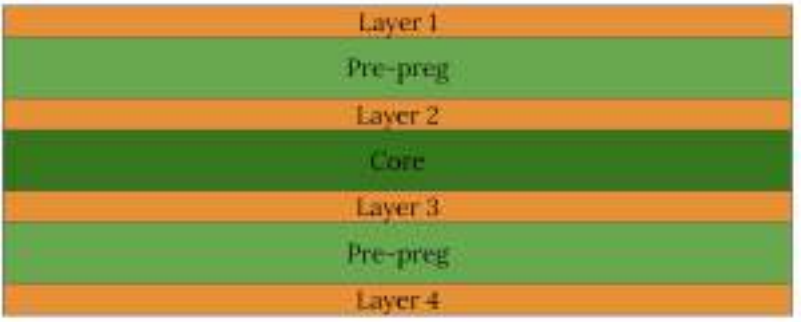

Figure 5.15: Very common 4-layer stackup with pre-preg on the outside.

Whatever material you choose for your design will have both a core and pre-preg version. For example, if you choose FR-4, your PCB manufacturer will use FR-4 core (which contains resin that is fully cured) and bond it to other FR-4 core layers with FR-4 prepreg (which contains resin that is partially cured). As long as the materials are compatible with each other, different combinations of material can be stacked to achieve your desired thickness. However, there aren't an

infinite number of thicknesses available, so you may need to combine several sheets of material to get to the thickness you want. Rogers 4000 series is a family of compatible materials used for RF designs. Core materials in this family are available in 8 mil, 12 mil, 16 mil, 20 mil, 32 mil, and 60 mil thicknesses, as well as several different panel sizes. The pre-preg in this family is available in 3 mil, 4 mil, and 5 mil thicknesses. There are a huge range of possible stackups and thicknesses you can achieve by mixing and matching those materials. And, since they're all in the same family, you know that they will all bond to each other.

When designing a stackup using materials of different thicknesses, make sure that the resulting PCB has a symmetrical cross section. Having an unbalanced or asymmetrical cross section can cause your board to warp during lamination.

If your design will be using a lot of current, you may want to change the copper thickness in your stackup. Copper thickness is expressed in sort of a strange way. Instead of mils, it's expressed in ounces, and this refers to the weight of copper that it would take to coat one square foot with the same thickness of copper that you want to use. For example, if you coated one square foot of space with a 1.4 mil thick layer of copper, you would have used exactly one ounce of copper. So, a PCB layer with a copper weight of one ounce means the copper is 1.4 mils thick. Here's some common weights and thicknesses. In general, weight $= \frac{thickness}{1.37}$ (where weight is in ounces and thickness is in mils).

Weight (oz.)	Thickness (mil)	Thickness (µm)
0.5	0.7	18
1	1.4	35
2	2.8	71
3	4.1	104
4	5.5	140

Table 5.1: Common weights of copper and their thickness equivalents.

Most designs use 1-ounce copper on the outside layers and 0.5-ounce copper on inner layers. You can increase copper thickness to as thick as you want, but it will start to get very expensive. High current designs may need to use something like 4-ounce copper. Besides cost, there is another issue that you need to contend with as copper weight increases: under etching. A very thin trace with very thick copper will have trouble etching uniformly, as seen in figure 5.16. You can use

table 5.2 as a guide for the minimum trace and space width for a given copper thickness, but your manufacturer will also advise you if they believe you'll encounter yield problems with your copper thickness and trace width combination.

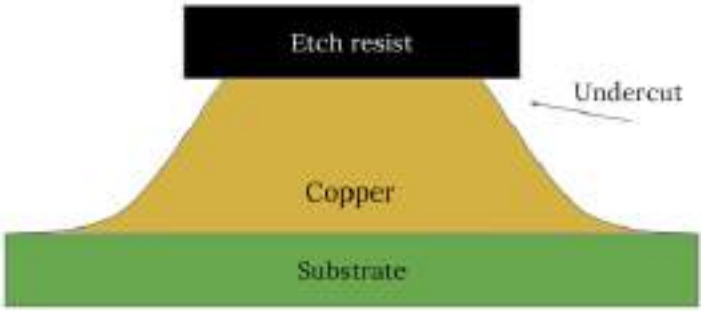

Figure 5.16: Under-etch and copper thickness vs. trace width

Copper thickness (oz)	Minimum trace/space width (mil)
6	25
5	20
4	16
3	10
2	8
1	6
0.5	3

Table 5.2: Copper thickness vs. minimum trace/space

When you send out your design, you need to specify which traces you need to be impedance-controlled. Because of manufacturing variability, the fabricator will make small adjustments to the dimensions of your stackup to meet the impedance you specified. So don't be alarmed if you get your board back and use calipers to measure your impedance-controlled trace and find that it's off by a couple mils. The impedance report that your fabricator should have provided with your PCBs will show the new dimensions and the measured actual impedance. Figure 5.17 shows an example stackup drawing sent to a fabricator. Notice how it calls out the intended impedance of traces on layers one and two, along with their reference layers.

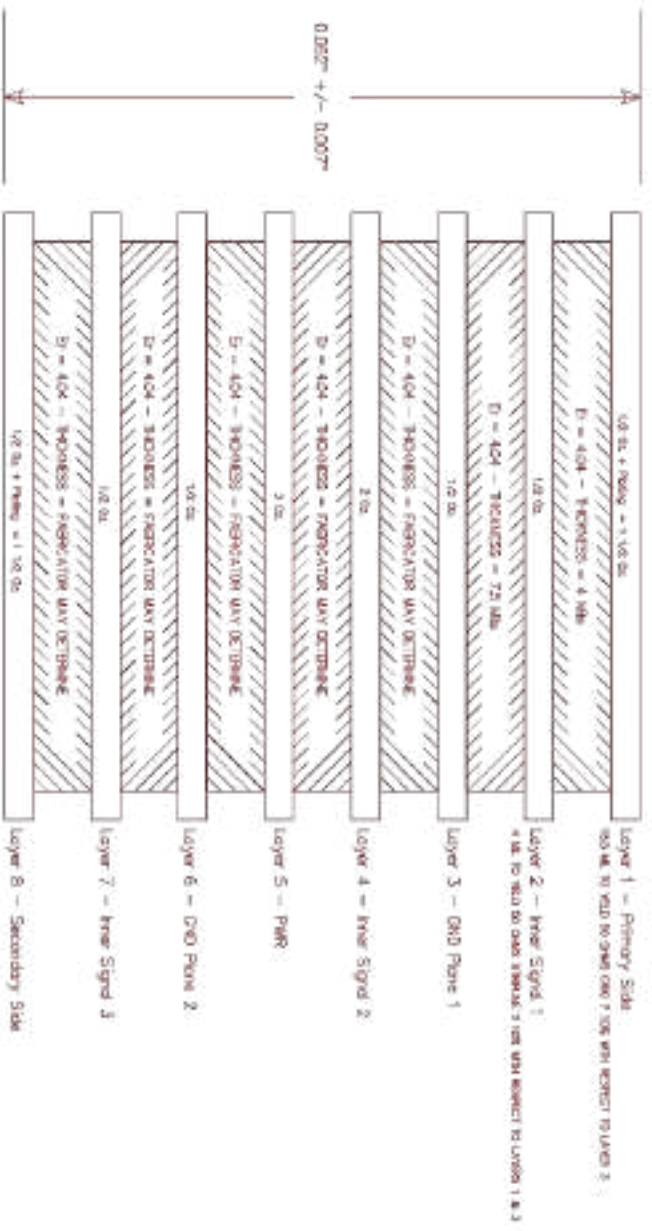

Figure 5.17: Example stackup of an eight-layer board with impedance-controlled traces on two layers.

Traces that are engineered to have a particular impedance are known as transmission lines. There are many different structures you can use to achieve this. When choosing a transmission line structure, there are three popular options: stripline, microstrip, and coplanar waveguide (CPW). The figures below show what they look like if you're unfamiliar with them.

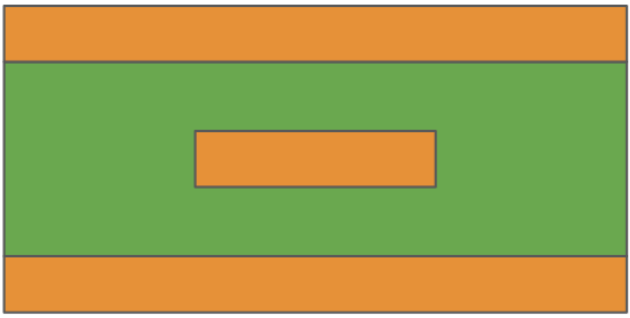

Figure 5.18: A stripline transmission line has a conductor sandwiched between two ground planes, one above and one below it.

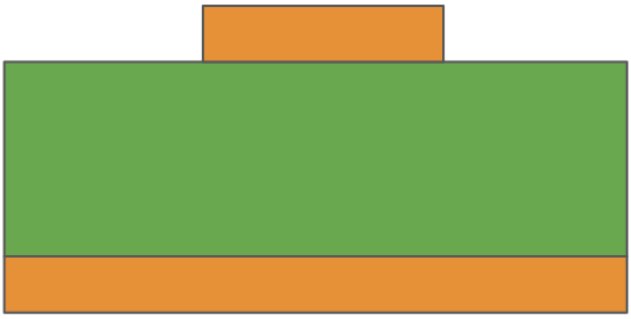

Figure 5.19: A microstrip transmission line has a conductor on the top layer and a ground plane under it.

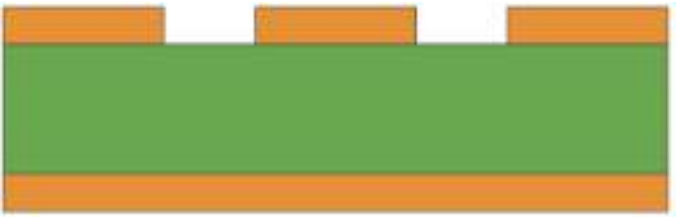

Figure 5.20: A coplanar waveguide is similar to microstrip, but has a ground plane on either side of the conductor as well as underneath it.

The impedance of each of these transmission lines depends on the exact dimensions and spacings of the copper and dielectric. There are plenty of online calculators you can use to figure out the spacings you need to get to a particular impedance. You can find links to some good ones on this book's website, DesigningElectronics.com

One advantage of using microstrip or coplanar waveguide transmission line structures instead of stripline is that the entire structure stays on the top layer, meaning you don't need to worry about via transitions (assuming all of your parts are on the same side of the PCB), and you have access to probe and modify the entire transmission line. Incorrect via transitions can make the performance of your transmission line suffer, and in the case of stripline, that's very hard to fix without respinning the board. Repairing microstrip or coplanar waveguide is also difficult to do without compromising performance, but it's much easier than having to excavate into the PCB to even get access to the signal in the first place. You can also probe microstrip and coplanar waveguide using an E field or H field probe, which isn't possible with stripline because it's shielded above and below by a ground plane. But because stripline is well shielded and therefore better isolated, you can route them more densely, and they're less likely to pick up noise or crosstalk. Given the same impedance, stripline is always going to be thinner than microstrip. But stripline also needs a very consistent dielectric constant across the entire substrate. This is because stripline is surrounded on all sides by the substrate material, and any change in the dielectric constant along the length of the trace will change the characteristic impedance of the transmission line and cause reflections and loss. Losses in microstrip tend to be less than losses in stripline because half of the fields in microstrip are in air rather than the PCB substrate, and air has a negligible dissipation factor compared to the substrate.

If you're prototyping RF designs using a circuit mill or etching your own PCBs, microstrip is great. You don't need to make any vias and you only need to etch or mill one side of your copper clad substrate. The precision spacing requirements that CPW has between the conductor and the top ground plane can be difficult to achieve with a mill or home etching, but microstrip doesn't have that problem.

The most cost-efficient way to choose a stackup that meets your technical requirements is to use a stackup that your PCB manufacturer already offers. Non-standard stackups can get very expensive. For most applications, one of the stackups that your PCB manufacturer offers will works just fine. They will typically offer a standard stackup for two-layer and four-layer (and sometimes six and eight layer) low frequency designs. They will also sometimes offer standard stackups for high frequency designs using materials like Isola FR408 instead of standard FR-4.

5.3.1 Vias

Another important thing your stackup should specify is the via structure. The simplest and cheapest designs contain only through hole vias, which are just holes that are drilled and plated after the layers of the PCB are pressed and bonded together. But there are other more exotic via structures possible: blind, buried, and microvias. Adding these kinds of vias to your design will add cost, but it will also allow you significantly more design freedom. Some high-performance designs actually can't be implemented without them. The best way to describe all of these different via types is with a picture, like the one in figure 5.21.

Blind vias start on the top or bottom layer, but don't go all the way through to the other side. Buried vias connect inner layers, but don't ever extend to the top or bottom layers. Microvias are very small vias and usually connect only one layer to an adjacent layer. It's possible to stack microvias, but this is usually not a good idea. Stacking microvias directly on top of each other can cause delamination and other reliability problems in your PCB. Instead, you can use staggered microvias, which has been shown to improve reliability.

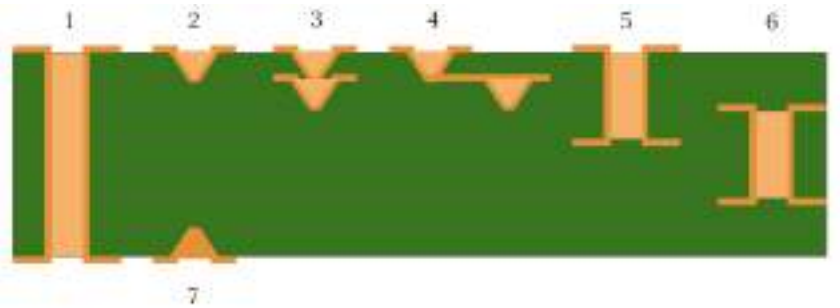

Figure 5.21: Different via types: 1 is a plated through hole via, 2 is a microvia, 3 is a stacked microvia, 4 is a staggered microvia, 5 is a blind via, 6 is a buried via, and 7 is a copper filled microvia.

Blind vias can substantially increase the cost of your fabrication and make it take much longer, depending on what layers they're on. Generally, it's best to keep your blind vias from overlapping on layers. For example, your PCB fabricator will be able to build a board with vias from 1-3 and 3-4 faster than a board with vias from 1-3 and 2-4. This is because the precision must be higher for overlapping blind via layers, since they have to be more carefully aligned. Depending on the design, it might also require another lamination cycle, which significantly increases the time and cost of the fabrication. A lamination cycle is the process of bonding all of the sheets of material together under high temperature and pressure to for a PCB. For more complex boards, the stackup is broken into sub-assemblies that are sequentially laminated together. You should discuss your planned via sets with your manufacturer to make sure that it's easy to manufacture and that it minimizes lamination cycles. Sometimes a change that is inconsequential to you can make a big difference to your manufacturer.

If you want to save some time and money while still having something like blind vias, see if you can get away with backdrilled vias. Backdrilled vias act like blind vias, but are manufactured in a different way. They start as through hole vias, are plated, but then a drill bit that is slightly wider than the original hole is used to drill out half of the via with plating. The result is a hole that goes all the way through the board, but only half of it is conductive. You might want something like this if you're transitioning a stripline to microstrip. If you use a through hole via for an RF transition like that, you'll end up with a stub on your transmission line from the extra length of the via, which can cause matching problems. Be aware that backdrilling is typically

done manually, by a skilled technician, so this approach will probably not scale for production and shouldn't be used on a huge number of vias.

When you place a single via between two unique layers, you have now added a drill step to the manufacturing process. This means that it's the same price to add a single via between two layers as it is to add 1000 vias between those same layers. So, if you're using an expensive blind via between layers one and two, only using a couple of them isn't going to save you any money. You will have already moved up to a more expensive pricing tier and longer lead time, so either try to eliminate it, or take advantage of it and make use of it.

The sizes of the holes in your vias are one of the main drivers of the cost to manufacture your PCB. Most manufacturer standard stackups will limit hole and via size to be above 6 mils. Holes below about 6 mils in diameter require a laser to drill, since mechanical drill bits that small can't reliably plunge without breaking. Laser vias are also sometimes used for blind or microvias. Be aware that adding laser vias to your design will also increase the cost.

Vias have a minimum aspect ratio of their depth to their width to be manufacturable. Regular drilled vias can have an aspect ratio of up to 10:1. That is, the hole can be drilled up to 10 times deeper than the via is wide (so a 10 mil hole can be drilled through up to 100 mils of substrate). Some fabricators can even go up to 20:1. For laser vias, the aspect ratio is much smaller at 2:1 or 1:1, depending on the fabricator. That means that if you want to laser drill a 6 mil via, you must do it on a substrate no thicker than 6 mils (for an aspect ratio of 1:1). Keep in mind that the maximum substrate thickness here refers to the substrate that the via must traverse, not necessarily the entire board thickness. So you might have a buried laser via that's going from layer 4 to layer 5 through a 4 mil prepreg, which is fine, even though your finished PCB is 62 mils.

In addition to the depth to width aspect ratio, vias must also have a minimum annular ring size. This size is specified as the distance between the edge of the intended drill hole and the edge of the copper pad that is going to be drilled. Drills don't have perfect accuracy, and if they don't hit exactly in the middle of the via pad, they could "breakout" and land partly outside of the intended via location. Making the annular ring larger allows for a less accurate drill hit to still be functional, and can be improved even further by adding a teardrop tapering along the side.

Figure 5.22: A center drill hit on the annular ring, a breakout, and a taper preventing a breakout.

The minimum annular ring size for a via will depend on the copper weight of the layers it's on. But for typical copper weights (0.5 to one ounce), a good rule of thumb is to use an annular ring of at least 6 mils. Talk to your fabricator to be sure.

Vias can cause assembly problems if you're not careful. If a via is on a component pad, solder can wick down into the hole during reflow, not leaving enough for the component to be firmly attached to the pad. There are two ways to deal with this. The cheapest way is to tent your vias. This is a process where solder mask is applied over top of vias on both sides of the board (but not in component pads). Wet solder mask has enough surface tension to cover the hole and prevent anything from getting into it. So the via in a component pad on one side has a thin membrane of solder mask on the other side, creating a well. The risk with tented vias is that there can be chemical entrapment inside the via, which over time can cause failure. Tenting vias is fine for prototyping purposes, but you may want to consider it more carefully before building a production design with them. The safest solution is to plug your vias. The holes can be filled with a non-conductive or conductive epoxy. They can also be filled with metal. When plugged vias are used on a pad, they are sometimes called active pads and the vias are plated over after being plugged. Another term you'll hear for this is "via-in-pad". Plugged vias ensure nothing can wick down into the via and that nothing can get trapped inside of it, but of course, this technique adds time and cost to your fabrication.

5.3.2 Stackup Example

Figure 5.23 shows an example of a real, slightly complex stackup. This report from the fabricator show a 10-layer board and uses two different substrates. Layers one and two use the Rogers substrate 4350B. This is a cheap, general purpose substrate that's good for RF applications because of its low loss tangent. The rest of the board is a substrate called 180A, which is basically the same thing as FR-4. The reason for these two materials is that layers one and two both have impedance-controlled transmission lines on them. They're both specified as 50 ohms, and one is a coplanar waveguide while the other is stripline. The ground plane references that they're using are noted, and the dimensions required to hit the 50 ohm goal are also listed.

There are three kinds of vias on this board: through hole, a blind via from layer one to two, and a blind via from layer one to three. Notice that this board is 66 mils thick, which is thicker than the standard 62 mil thick board. This is because while working with the PCB fabricator, it was determined that a 62 mil thick board would result in one of the lamination layers being too thin. This same discussion also revealed that the aspect ratio of some of the blind vias was too small, and the hole size needed to be expanded slightly. This PCB was also fabricated with a different, second PCB manufacturer who gave different design notes. They suggested expanding some of the vias so that they could be mechanically drilled instead of laser drilled, which saved significant cost on the fabrication. The message is clear: talk to your manufacturer early and often, and make sure it's a two-way conversation. They are experts at building boards and know what will work and what won't. They will save you significant time and money if you let them. It's also a good idea to talk to more than one fabricator, since there's almost always more than one way to make your design. This is a great learning opportunity and will usually save you time and money.

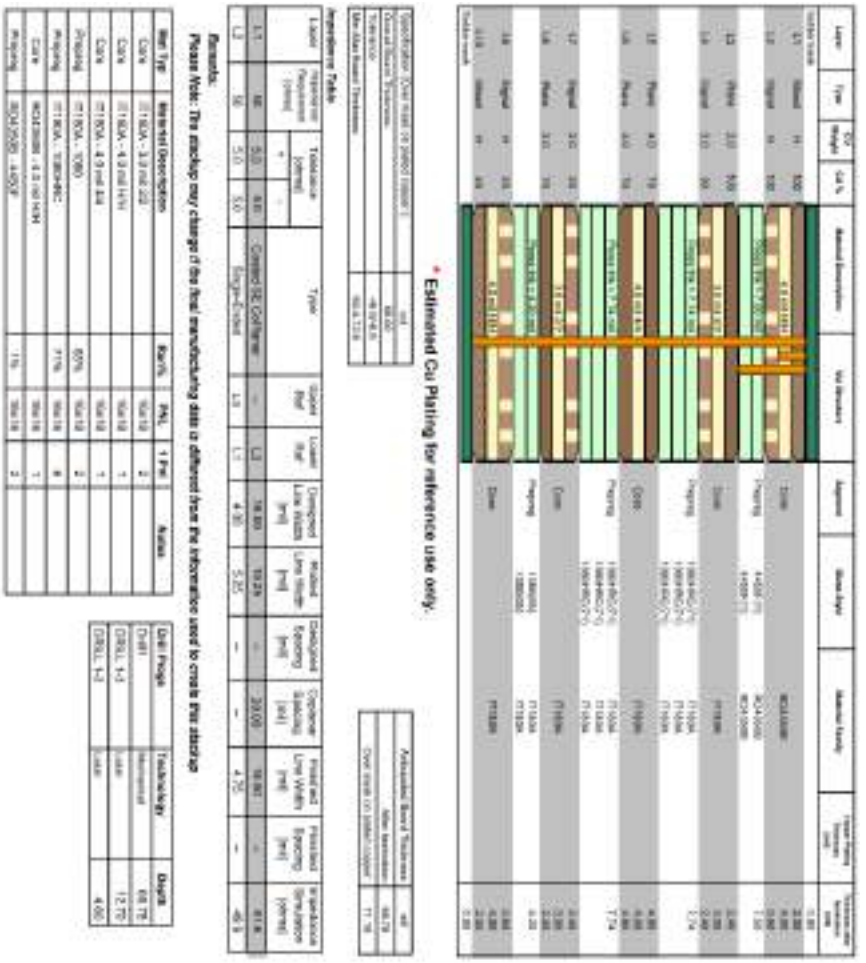

Figure 5.23: Example stackup of a 10-layer board with impedance-controlled traces on two layers.

5.4 DFx

There are acronyms that people like to use when referring to design considerations for particular areas, like Design for Manufacturing (DFM), Design for Test (DFT), Design for Inspection (DFI), Design for Variability (DFV), Design for Reliability (DFR), and Design for Cost (DFC). There are many more design considerations like these that may or may not apply to your design, but collectively these are known as DFx (Design for X or sometimes Design for Excellence). The layout stage of a PCB design is when a lot of these considerations actually get implemented. This section addresses particular things you should watch out for that relate to DFx.

One potential reliability issue that's often overlooked is tin whiskers. Tin whiskers are tiny filaments (0.5 to 10 microns thick, up to 1 cm long) that grow from tin coated surfaces that can cause shorts. Materials other than tin like zinc and cadmium can also grow whiskers. Tin whiskers can cause very significant problems. Eleven different NASA missions have had flight computer failures because of suspected tin whisker issues. Unsurprisingly, NASA has an entire website[4] dedicated to the study and prevention of tin whiskers. The scary part about them is that their formation is not well understood, and while there are methods to mitigate their formation, there is no known way to completely prevent them. The best ways to mitigate them are to use conformal coatings and avoid pure tin or zinc.

Practically, avoiding tin whiskers means you should use ENIG plating when possible (with the exception of RF traces, since ENIG adds extra loss at high frequencies). If that's not possible, don't use a plating that is pure tin or zinc. Tin whiskers can form if you don't use an alloy and can cause shorts later in the life of the product.

[4] https://nepp.nasa.gov/whisker/index.html

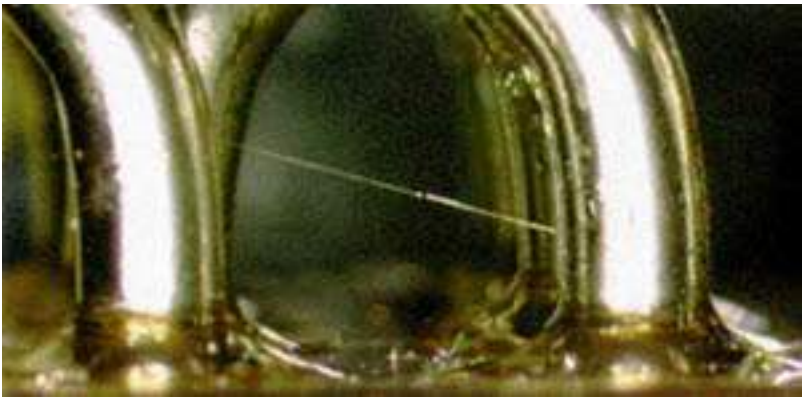

Figure 5.24: Tin whisker formation. Image courtesy of NASA Goddard Space Flight Center.

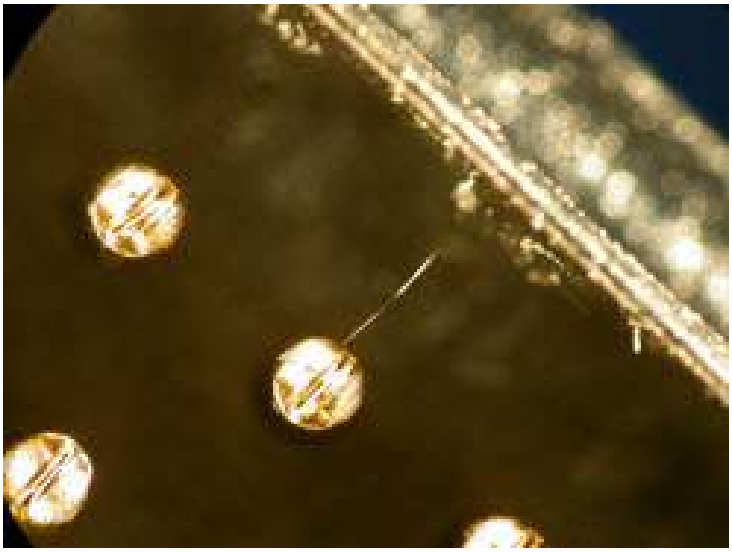

Figure 5.25: Tin whisker shorting a connector pin to the enclosure. Image courtesy of NASA Goddard Space Flight Center.

ENIG is just one of several finishes commonly available at most PCB fabricators. ENIG is popular because it has a flatter finish than tin/lead plating, which can reduce tombstoning and improve assembly yield. The downside of ENIG is that it's more expensive than other fin-

ishes and can suffer from a corrosion problem called "black pad". To address this last problem, you can add palladium plating, a process called ENEPIG (electroless nickel, electroless palladium immersion gold), or sometimes "universal finish". ENEPIG solves the black pad problem, but is even more expensive and sometimes doesn't solder quite as well as ENIG. On the opposite end of the spectrum is HASL (hot air solder leveling). HASL is very common, cheap, and often the default finishing option at PCB fabricators. It can be difficult to make perfectly flat, it contains lead, and isn't great for very fine pitch pins. However, HASL is usually the cheapest finish you can get. Lead-free HASL is also available for designs that need to be RoHS compliant. OSP (organic solderability preserve) is different from the other finish options we've talked about because it's not a metal. It's an organic coating (sometimes known by the trade name ENTEK) that is used to keep copper pads from oxidizing after etching. OSP remains flat, doesn't have lead, and is cheap. The downside is that it doesn't work well with plated through holes, has a short shelf life, and is sensitive to handling. Hard gold is another finish option, but you shouldn't use this unless you have a specific reason. It's expensive, doesn't go down very well, and can cause corrosion in rare cases. Hard gold is extremely durable and mostly finds use in card edge finger contacts or other contacts that experience a lot of mechanical wear over their lifetime.

If your design is going to be populated with a pick and place machine, you'll need to include fiducials. Fiducials are small visual markers on a PCB that the pick and place machine's camera can use to identify the orientation and coordinate plane of a PCB. There are usually two or three fiducials on a PCB, depending on the size and shape of the board or panel. In addition, you may want to add an extra fiducial next to any large chips (e.g., an FPGA or processor) to ensure an accurate placement and alignment of that chip. When panelizing a design, put fiducials on the panel itself as well. Talk to your production engineer when determining how many fiducials to place and where.

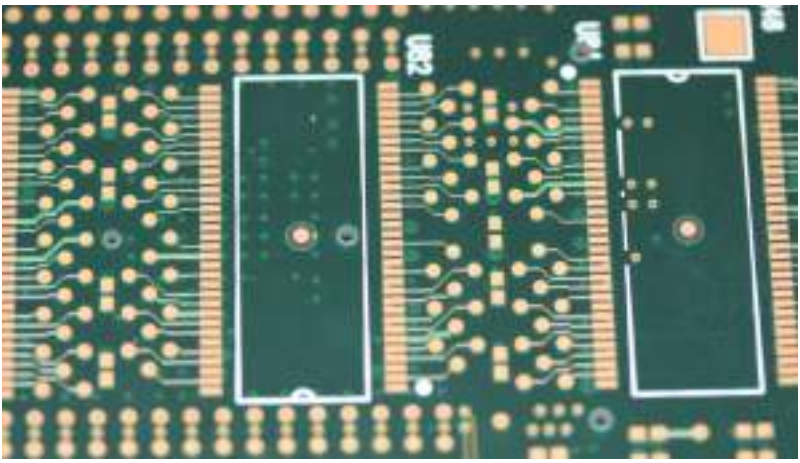

Figure 5.26: The middle of these two footprints have a fiducial to help ensure the pick and place machine lines up the part correctly. This is sometimes done on parts with a large number of pins like these.

You should place fiducials asymmetrically so that the pick and place machine can resolve any rotation of the board uniquely. The fiducials should also be at least 150 mils away from the edge of the board so that they are not obscured by the rails of the conveyor during assembly. Fiducials are typically round or square features on the top layer of copper. The copper layer is used rather than silkscreen because the method used by PCB fabricators to place copper features is much more accurate than the method used to place silkscreen. An example fiducial might be a 35 mil diameter circular pad with a 100 mil diameter solder mask relief.

If any parts need to be accessed by a person, tool, or test fixture (for example, a potentiometer, connector, jumper, button, or test point), make sure there's enough clearance around the component for the tool, fixture, or finger to access it. If someone's finger is going to be used, make sure it's safe to put a finger there. That means no nearby high voltage or hot parts that you could accidentally touch. You should be able to get a soldering iron and tweezers to each component. If a part has a screw on it (like a TO-220 package), make sure there's enough room to get a screwdriver to the screw without it hitting anything.

If there's space for it, all notes, component designators, and pin markers in silkscreen should still be visible after all components have been populated. If your CAD tool supports 3D component models,

you can use that to assist in determining whether any silkscreen will be occluded. Your PCB fabricator will typically list the minimum font weight and height for silkscreen. Because silkscreen is laid down in a much poorer resolution than copper features, make sure you pay attention to these limits, or none of your silkscreen will be legible.

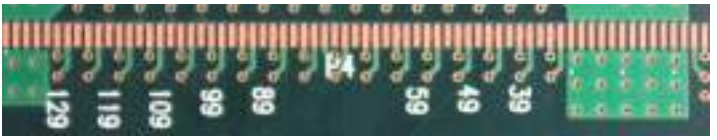

Figure 5.27: The silkscreen in the middle of this image is over the top of a via, making it difficult to read.

Most PCB fabricators will remove all silkscreen that overlaps pads during their DRC process, but you should check it yourself before you send it out. Put a silkscreen dot next to pin one of every chip. This makes it easier to trace out signals, replace parts, and check that the part was populated in the right orientation. You can also add small silkscreen tick marks next to every 5th or 10th pin on large connectors and ICs to make it easier to count pins during debugging and bring up. Another option is to mark the pin numbers at each corner of the chip.

Two pin discrete parts (resistors, inductors, capacitors) that are packed closely together can often be difficult to differentiate during manual assembly. To help tell which pad goes with which footprint, you can either use a rectangular silkscreen outline around the footprint (like in figure 5.28), or put two small silkscreen lines connecting the two pads (like in figure 5.29). If your footprint is big enough, you can even connect the two pads with a small silkscreen symbol of the component that gets populated there, as in figure 5.30.

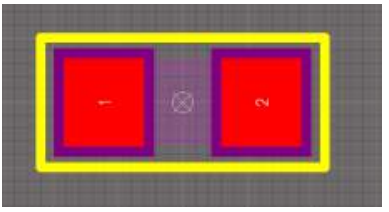

Figure 5.28: Rectangular silkscreen outline (in yellow) around a capacitor footprint to help differentiate it from surrounding footprints.

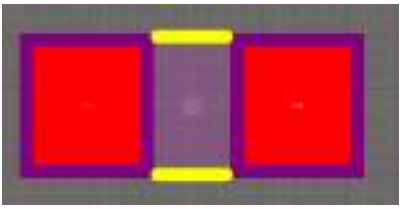

Figure 5.29: Alternative silkscreen method to help differentiate a part from surrounding footprints.

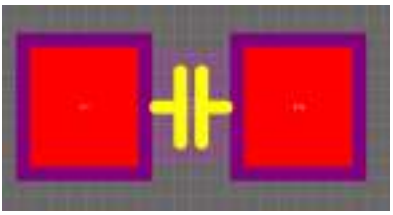

Figure 5.30: A small silkscreen component symbol can both differentiate it from surrounding footprints and tell you what kind of part belongs in this footprint.

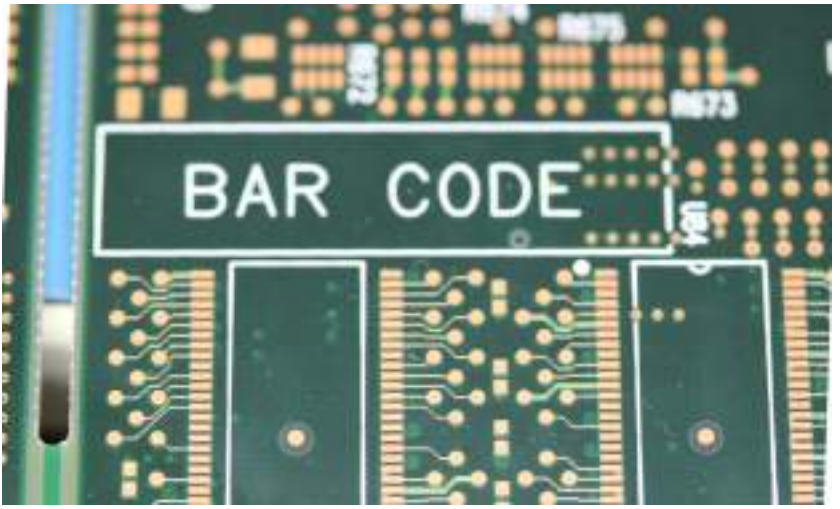

Figure 5.31: You can use a silkscreen outline to show assemblers where to put identifiers like a barcode to track individual units.

It's not practical to add test pads to every single net, but you can use vias as test pads as long as they're not tented. Consider leaving vias untented for the first prototype of your design so it's easy to use needle probes to connect to any trace with a via. You can even solder wires or add discrete components directly to the exposed vias if rework is necessary. If you need to probe a net with a test fixture for production, you'll need to use a real test point. A via is too small and fragile, and your production board should use tented vias to prevent accidental shorting.

Test pads should all be on the same side of the PCB if possible. That makes the test fixture much easier to build. Additionally, if all of your components are on the same side of the board, you can put all of your test points on the *back* of the board, making it very easy to build a bed of nails test fixture without having to worry about hitting any components. To make sure your test fixture pins can hit your test pads, make your test points at least 1 mm in diameter. If your test points are on the same side as your components, give the test points a 5 mm or greater clearance around any large components or connectors so that your test fixture pins don't collide with them.

If at all possible, align all of your test pads to the same grid size. A lot of people like to use a spacing of 100 mils. This allows you to use 100 mil header pins, breadboards, perf boards, and many readily available connectors to assist you in building programming and test

jigs.

Differently shaped annular rings around vias can be used to signify connection to different planes. For example, an untented octagon rather than a circle can be used to demark a power plane. This makes identification easier during debugging and testing.

If you're hand assembling a PCB, you can make it easier to assemble by lengthening the pads slightly so they're easier to access with a soldering iron. This only applies to footprints that you will be soldering with an iron, so packages like BGA and QFN that don't have externally accessible pads and require reflow should use the recommended footprint dimensions that are in the datasheet. Some chips have externally accessible pins, but a ground or thermal pad underneath. If you need to solder a chip like that without reflowing it, you can put a large plated through hole in the middle of the ground/thermal pad. That way you can solder the chip's external pins on the top side, then flip the board over and put your soldering iron in the hole to make a good connection to the plated hole and ground plane.

You need to make sure components and connectors aren't going to hit each other, the enclosure, or any other mechanical features. One way to do this is to draw a border around each part in silkscreen so that when you're placing parts, you know how much clearance to leave around them. A better method, if your CAD package supports it, is to use 3D models to ensure nothing collides. Many component manufacturers provide free 3D models of their products that you can import so you don't need to painstakingly model every component you use.

Be aware that many manufacturers consider a hole and a via two different things. They usually require a larger clearance around a hole than a via. If you accidentally use too small of a clearance around a hole, the manufacturer will usually edit your design to add extra space if it looks like it won't affect the design. But it's better if you catch it before they do.

All copper needs to be pulled back by at least 1 mm from the edge of the PCB, or from any edge that will be cut or milled out. PCB fabricators have different requirements for this, but 1 mm should be enough to meet the requirements of most fabricators. This is also important if you will be separating boards yourself with a bandsaw or Dremel. The reasons are the same: if there is copper on multiple layers along the line where a cut will be made, the copper can get smeared down along the edge during the cut and short to other layers.

Use manufacturer provided DRC files if possible, but be aware that they may not always be complete. Take the time to check your DRC against the rules listed on the manufacturer's website.

Use thermal reliefs on parts that are connected to large copper planes or polygons. This is especially important on discrete components like resistors, capacitors, and inductors to prevent tombstoning (as seen in figure 5.32) during reflow. It also makes it easier to populate and remove components when hand soldering (either with an iron or hot air). High-speed and RF parts need excellent, low inductance paths to ground. Thermal isolation will add extra inductance in the path to ground, so don't use this technique on high-speed and RF parts. On components that are safe for thermal isolation, ensure that there are at least three struts to the main polygon. Using thermal isolation where possible makes soldering and desoldering much easier, since it reduces the effective thermal mass of the component.

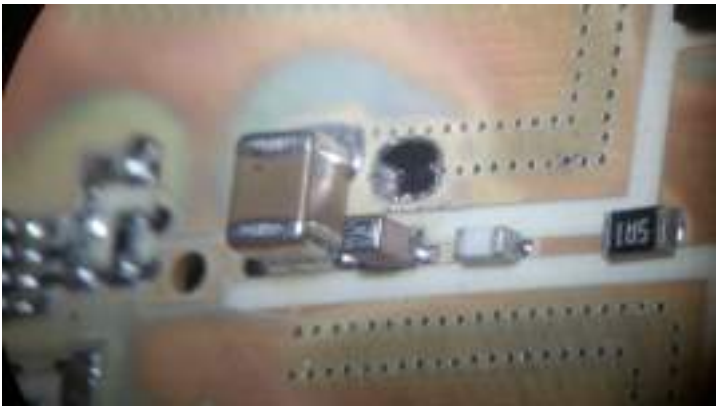

Figure 5.32: Use thermal reliefs to prevent the tombstoning that can be seen here. Notice that the middle capacitor is only soldered down on one end. That's because the thinner copper heated up faster and the surface tension of the solder was enough to pull the entire component upright.

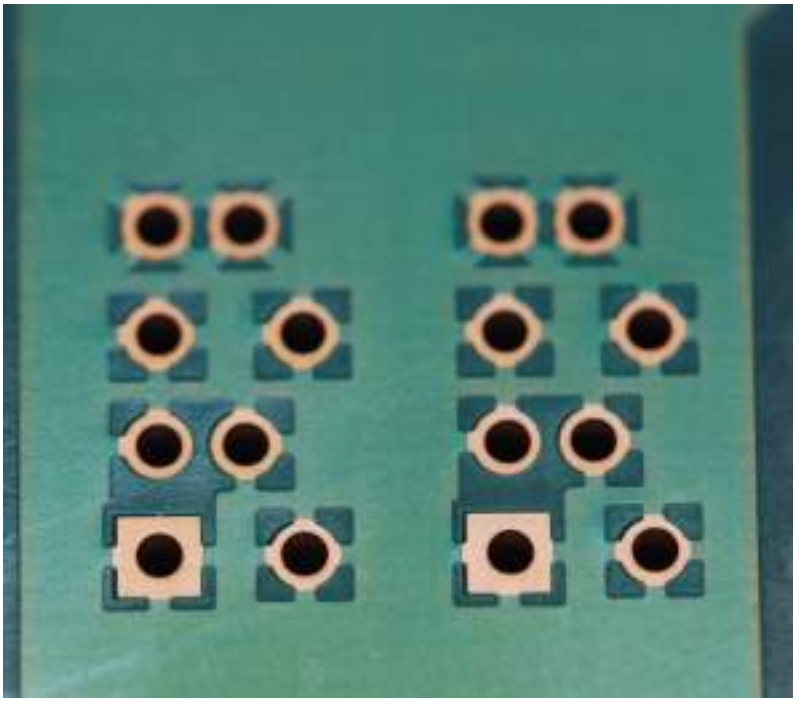

Figure 5.33: Examples of vias with thermal reliefs.

Pay attention to the placement of your parts that have a large thermal mass. During soldering, if a component absorbs a lot of heat and it's right next to much smaller components, the smaller components may take longer to heat up as the heat is drawn into the larger component. This is especially important to consider if you're assembling by hand without a reflow oven, since it will be very difficult to heat up a large area all at once. Thermal reliefs can be useful here.

Some components that generate a lot of heat will recommend that the footprint contain a grid of vias to the ground plane. This helps draw away heat from the chip and spread it out over the rest of the PCB. You will also see this recommendation on many RF parts, as a very low inductance path to ground is critical for correct operation. Pack in as many vias as you can safety fit.

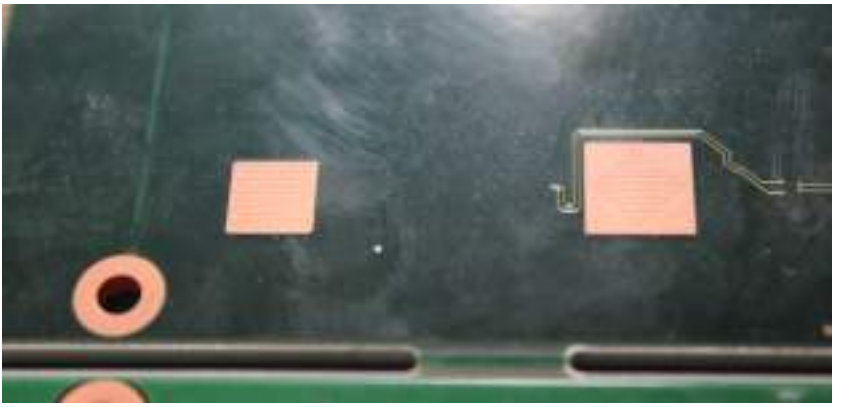

Figure 5.34: Vias can be seen here (plated over) on the center pad of two components.

Make sure all component footprints have their coordinate origin located in the middle of the part. If you export your pick and place instruction file from your CAD program and the origins of footprints are offset from the middle of the part, the pick and place may incorrectly place your part with the same offset.

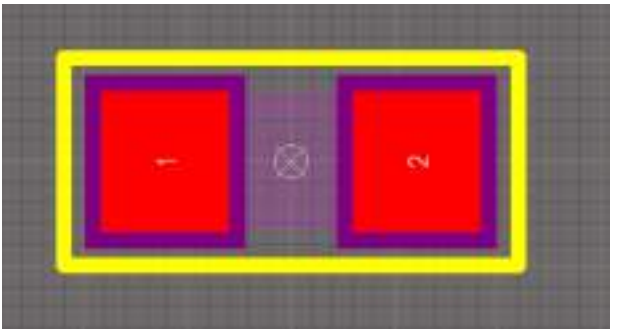

Figure 5.35: Component with origin in the middle of the part.

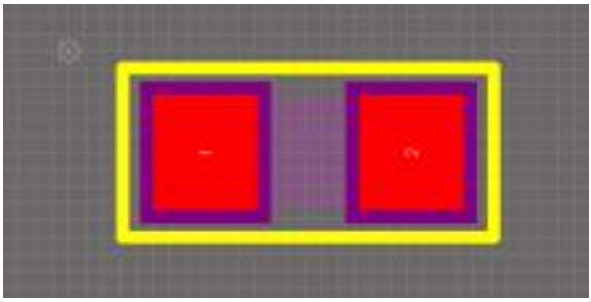

Figure 5.36: Component with origin offset to the top left.

Isolation between parts and traces is very important when manufacturing a PCB without solder mask. If you're using a circuit mill, pads and traces tend to peel more easily, especially as you add heat to solder. If you have to add more heat to overcome more thermal mass, you'll have a higher risk of peeling up an adjacent trace or damaging the board. Physical spacing and isolation will help prevent this. Another thing that makes a board with no solder mask more difficult to deal with is the fact that it's much easier to accidentally short pins, traces, and planes together. This is especially true when using a soldering iron instead of solder paste and hot air. The further you can get your traces apart from each other after they leave the package pins, the lower the chance you'll have of getting a solder blob that shorts out 3 or 4 pins at once. Using flux is highly recommended when assembling these so called "barebones" boards.

Sometimes you'll see PCBs that use a technique called copper thieving. There is a lot of confusion around what copper thieving even is and why it is done. Sometimes you'll hear that it was used to keep a thermal equilibrium across the PCB. If there's lots of copper in some areas and none in others, the side with more copper will take longer to heat up, causing uneven expansion of the PCB. This can cause warping of the PCB during reflow. Copper thieving addresses this by adding electrically unconnected squares of copper to roughly even out the distribution of copper on the board. It can also help the manufacturer by increasing the longevity of their etching solution, since less total copper needs to be removed. The second reason you'll hear that copper thieving is needed is to ensure high quality vias. If the outer layers of your PCB have different amounts of copper in different places, the current for electroplating won't be evenly distributed, which means some areas of copper will be thicker than others. The concern is that some vias won't get plated enough and can fail during reflow. Copper thieving

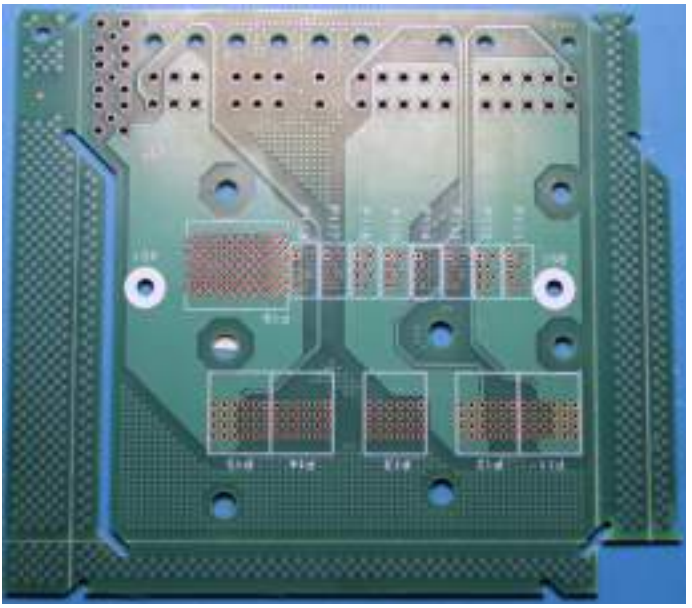

Figure 5.37: The grid like copper pattern here is an example of copper thieving.

"steals" some of the plating current to help ensure a more even distribution. Realistically, modern PCB manufacturing is good enough that neither of these problems should be a concern. If you see copper thieving on a modern PCB, it's more likely that it was done out of superstition than necessity.

5.5 Mechanical

If you want your design to be compatible with standard assembly equipment, the smallest outer dimension of your PCB should be at least two inches. Typical assembly line rails don't adjust to any smaller than about two inches. You have two options if your design is less than two inches on a side. You can either add additional copies of your design to the panel until it is at least two inches on a side, or you can add processing edges that will be scrapped after assembly. This technique also applies to PCBs that have an unusual shape and can't be easily slid down the two rails of the assembly line (for example, a circular PCB).

When you panelize a design, use V-grooving to make it easier to separate boards. V-grooving is a process that some PCB fabricators offer where they cut a V shaped groove on the top and bottom of the PCB outline so that the individual PCBs stay in the panel, but you can just use your hands to snap apart the PCBs into individual boards. This has the advantage of requiring very little force and leaving a relatively clean edge without using a band saw. You may want to perform basic testing after the PCBs in the panel are assembled but still panelized. Once testing is complete, you can either route out the boards with a mill, or if the panel is V-grooved, you can just break them off by hand. This will let you avoid the vibration and dust created by the milling process. However, if you plan on using V-grooving on a design that will go in an enclosure, be aware that V-grooving may leave a few extra mils on all sides of the board. V-grooving machines cannot stop and turn in the middle of the board like a mill can, so you have to make all V-groove lines go all the way through the board, edge to edge. V-grooving also can't be used on very thin boards (usually a thickness of greater than 3 mm is needed). Because the tool that cuts the groove is V-shaped, you need to keep tall components away from the cut line. Your PCB manufacturer can give you exact numbers based on their machines, but an example guideline might be to keep components taller than 25 mm at least 5 mm away from the cutting line.

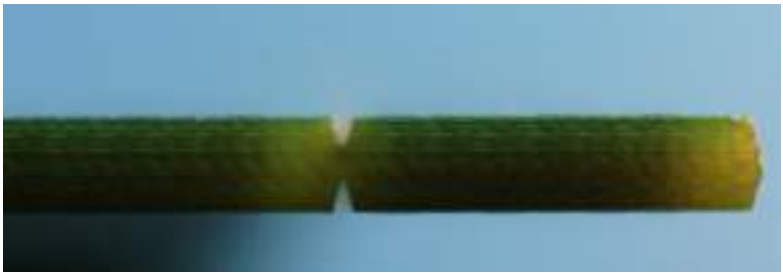

Figure 5.38: Edge-on view of a V-grooved PCB.

Figure 5.39: If your PCB is not a rectangle, you can use a technique like this to fit it in a rectangular panel while making it easy to depanelize by making only straight-line breaks. The thin white line to the left is a v-groove, and a routed slot to the right of it will create a contoured board outline when the v-groove is broken.

The crudest way to separate boards is to use a hacksaw or bandsaw. You'll need to use this technique if you're only building a couple of boards and don't have any of the tools needed to do depaneling and/or don't want to pay for V-grooving. You'll also need to use this technique if you're doing "stealth panelization", which is when you put multiple designs on a single board for a fabricator that doesn't allow panelization. Some quick turn places don't like it when you try to include multiple designs on a single PCB, but as long as you don't use tabs or V-grooving, you can usually sneak it by them. In that case, you'll need to cut it apart using a hacksaw or bandsaw. Since these methods are both fairly crude (especially the hacksaw), you need to allow a channel at least 5 mm wide that contains no copper on any layer through which you'll saw. If you have a board with multiple layers and flood fills on any two of those layers, sawing through those layers can smear copper from one layer to another along the cutting edge and can cause shorts.

If you're building production boards, you'll want to fit as many boards as possible on each panel to get the highest yield. There is ded-

icated software to help do this planning, but you should work with your production engineer before trying to run any large number of boards through the assembly process. It's not as simple as just cramming as many boards on there as possible, since adding tabs and perforations to separate the boards later can cause problems during assembly if they're not placed well. Boards can sag during solder paste application or vibrate and flex in the pick and place machine. In general, a well designed PCB panel has about 70% usage. The rest ends up as scrap from spacing and clearances between individual PCBs and the edge of the panel. Board houses will request 100 mils between the edges of each individual PCB in the panel, as well as a 0.5 inch edge around a two-layer panel and one inch around a multilayer panel.

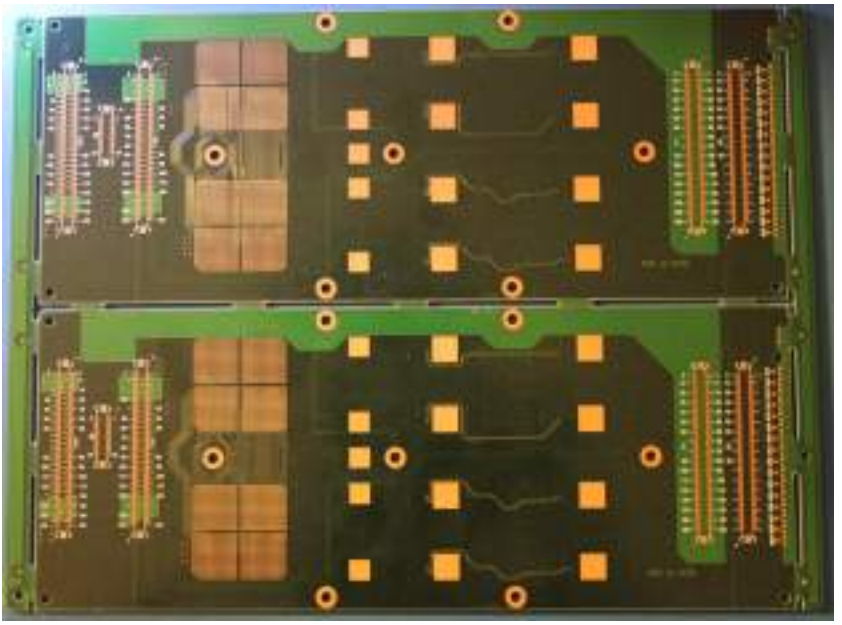

Figure 5.40: A PCB panel, ready for assembly. This panel has two PCBs on it and is supported on the left and right side by the conveyor.

A standard panel size is 18 inches by 24 inches. You can ask the PCB fabricator what panel size they prefer to save time and money in having to change the assembly line setup. If you're running multiple unique panels, make them all the same size so the conveyor width doesn't need to be changed.

There are a couple of options for separating individual boards from a panel. Solid tabs are mechanically robust, but they require being routed out with a mill or laser. Perforated tabs are solid tabs with small holes in them to create a perforation. These are sometimes also called breakaway tabs or mouse bites. You can use a pair of strong angle cutters to cut along the perforation, or you can break it with your hands. To remove a perforated tab, use a pair of pliers to grip the edge. Bend upward until you hear a crack and then stop. Now bend it back downward and remove the edge. This will prevent the PCB from delaminating.

You need to keep all components and copper at least 100 mils from the perforation holes. Special care needs to be taken with SMT MLCC capacitors. Because these capacitors are very inelastic, the force of bending and snapping off the tabs can actually cause microscopic cracks through the middle of the capacitor and cause it to fail immediately or much later, when moisture has seeped into the middle of the capacitor. For that reason, MLCC capacitors should be place about twice as far away from the perforation as everything else (about 200 mils). Typically, a perforated tab will have 5 holes in it. You can use fewer if you need to, but use at least 3 holes. The fewer the holes, the closer together they should be to maintain board strength. IPC-7351 has detailed information on spacing and clearance for perforated tabs. Because you want smooth edges on your boards once they're depanelized, you need to make sure the perforation holes are along what will be the final edge of the board. Don't place perforation holes in the middle of the tab or on the outer edge of the tab or else there will be a little nub of PCB material sticking out after you've separated each board.

If your design is going to be wave soldered rather than reflowed, you need to make sure there aren't any large holes or gaps in the panel. If there are, solder from the solder wave can lap up over the edge and spill onto the top side of the board. Any hole or space that is larger than 600 square mils needs to have what's called a "knockout" in it. That's a section of PCB that has no components or traces on it and will simply become scrap after depanelization. Note that the 600 square mil rule is NOT required for designs that will be reflowed, however if there are any parts of the panel that are not well supported or anchored to the panel edge, you can use this technique to prevent flexing and vibration during assembly.

Your panel will also need tooling holes. These are holes that will be used for test jigs (e.g., a bed of nails) to align the board to the fixture. Tooling holes are usually 125 mils in diameter and are unplated. You need at least three tooling holes, and they should be about 200 mils

from the edge of the panel.

Think about how your PCB will be supported within its enclosure. Screw bosses and standoffs are popular methods for locating and fixing a PCB within an enclosure. Also think about how your device will be used. For example, a bathroom scale will have hundreds of pounds exerted on it. Will your PCB bend when that happens? In complex cases like this, you can use finite element analysis to model the result of arbitrary forces on your enclosure and PCB. You should work closely with your mechanical engineering team on these issues *before* the design is finalized.

If your enclosure needs to be grounded, the mounting holes on your PCB need to be plated and connected to ground. They also need to have an annular ring around the hole with exposed copper so the bottom of the screw head will make solid contact with the ground plane. All traces and components should be at least 100 mils from the edge of the annular ring. Otherwise, if the screw is overtightened and begins to bite into the PCB, it can short or open traces.

Screw holes need to be accessible to a screwdriver without having to disassemble anything. This applies more to the mechanical design of how the PCB is attached to the enclosure, but make sure you don't "paint yourself in a corner" and end up with a design that is impossible to assemble. The best way to avoid this is to make sure that no fastener heads are covered up by anything else. If that's not possible, then think through the order in which you will assemble and disassemble the device when choosing the fastener locations.

Check the height of all of your components versus the height of the enclosure to make sure everything will fit. When you're drawing or importing the 3D models for each component, make sure that you use the worst case tolerance. For example, if your enclosure height is 6 mm off the surface of the PCB and a part's datasheet says that it is 5 mm +/- 0.5 mm tall, model it as 5.5 mm tall, not 5 mm tall. The same thing holds for the other direction. If you need a part to be at least a certain height, model it as the worst case tolerance in the other direction (so 4.5 mm in the example above). Tolerances add up and you want to make sure you can handle the worst case.

Consider making the outline of the PCB asymmetrical. This helps assemblers because it means there's only one way for the PCB to go into the enclosure. For example, you can cut off one corner of a square board to make the orientation clear. However, the mechanical design must also reflect this. If you have a square shaped groove that the PCB sits in, it doesn't matter if the PCB has had a corner removed to make it asymmetric because it will still fit in that square groove in any

orientation. The groove also needs a corner removed.

If you're hand assembling a board, connectors need to have enough clearance around them so that they can be soldered without melting the plastic connector housing. They also need to have enough clearance for the mating connector, including the wires that stick out of the back of the mating connector (if it's a cable). The first centimeter behind a connector on a cable can be surprisingly rigid, and there needs to be room for those wires to bend around on their way to their destination. Route cables cleanly through your design and plan where they will go. Take into account their thickness, the number of wires, and their rigidity (stranded vs. solid core). Cable bundles can be created using either zip ties, a cable sheath, or lacing, an older technique that uses string to keep wire harnesses neatly bundled. You can also build cable channels directly into your enclosure that will help guide wires and reduce their chances of being pinched by something during final assembly. This is also the best technique for assemblers, since it's clear exactly where the wires need to go. You can use tape or adhesive to keep the bundles from falling out of their channels.

If you have two connectors next to each other, make sure there is enough room for both of the mating cables to be plugged in at once. Some cables have a very wide lip around their connector, and this can vary quite a bit even within the same connector type. The plastic shell around the connector can easily take up double the space of the connector itself both horizontally and vertically. Think about which cables you're going to use and make sure that there is plenty of space between adjacent connectors.

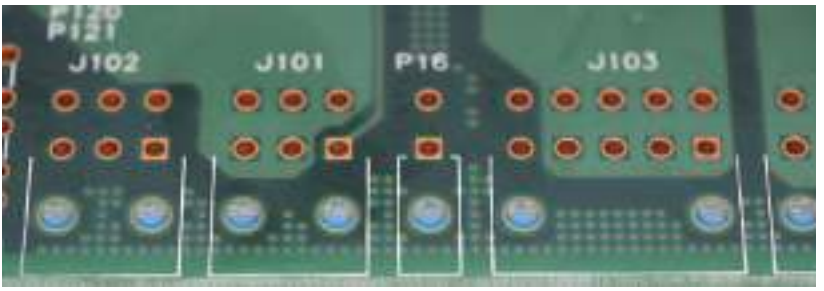

Figure 5.41: Use silkscreen outlines of the connector dimensions to help ensure you leave enough space between adjacent connectors.

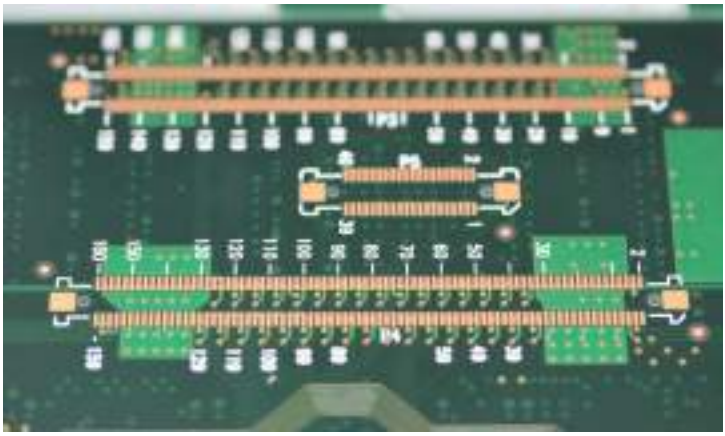

Figure 5.42: You can also use the silkscreen connector outline technique for non-edge connectors to check for clearance and to illustrate the correct orientation of the connector for assemblers. Pin numbers have been included here to assist in debugging and bring up as well as to help prevent connector orientation mistakes.

The same goes for switches and dials. For example, two potentiometers may have enough clearance next to each other to turn when there's no knob on them, but depending on the size of the knob, it can easily become impossible to have both on at once without them rubbing against each other. Plan the finished knob locations first, and then figure out where the potentiometers need to go afterwards, not the other way around.

Ribbon cables and Flat Flex Connectors (FFC) need to have enough clearance around them to access the latching mechanism or tabs used to insert and remove the cable. Avoid having to use a tool to connect or remove the cable if possible.

No matter what kind of cables you use, think about the order in which you'll assemble the device and how that will affect cable placement. For example, if you have an FFC underneath an overhang that must be assembled first, it might be impossible to plug the cable in. The length of the cables needs to be taken into account too. Think about how much slack you'll need to make the connection when the device is pulled apart, but also think about where that extra slack will go once the enclosure is buttoned up.

If there are lots of cables inside your device, and especially if they

are FFC or thin ribbon cables, consider rounding the outline of your board to prevent pinching or other accidental damage to the cables during assembly. Everything may look fine when it's apart, but tightening the final screws on the outside can apply pressure and cut or damage cables that have been poorly placed.

5.6 Thermal considerations

Nothing is perfectly efficient, which means your designs are going to generate heat. Since this heat is usually generated inside of IC packages, the first step in thermal management for electronics is usually to try to efficiently conduct the heat out of that package and into something that can either move the heat to another place or dissipate it. Thermal vias are often employed for this. Chips that get hot almost always have a thermal pad underneath the chip or a large exposed thermal tab that you can couple to your thermal management system. A via that touches one of these thermal pads to conduct heat from one layer of a PCB to another is known as a thermal via. You'll need a lot of them all right next to each other, and the datasheet of the part will usually show the recommended number and placement of the vias. For better thermal conductivity (and easier assembly) thermal vias can be conductively filled. Chips that call out the use of thermal vias are designed to use them: it's better to move or dissipate heat from the thermal vias than to put a heat sink on top of the IC package. Thermal vias can be used with normal PCB substrates like FR-4, but in extreme cases you may need to use a metal core PCB (MCPCB). This is an expensive technique and should only be used if other thermal management methods fail and you have no other options. MCPCBs are sometimes seen on designs with high power LEDs. If your design incorporates high-speed or RF parts, a stackup with a metal core may not be possible.

You have a lot of options to move heat around and dissipate it. The two most obvious solutions to remove heat are heat sinks and fans. They can be used together (a "fansink") for an even higher thermal handling ability. But a heatsink or fan is only effective if it's thermally coupled to the source of the heat. This is typically accomplished with a thermal interface material, which can reduce the thermal resistance between two objects. One common thermal interface material is thermal paste. This is usually a white, greasy paste that is evenly distributed in a thin layer between a heat source and a heat sink. There's also something called thermal epoxy, which is a permanent, two-part adhesive that has a high thermal conductivity. Thermal paste is easy to

remove if you ever need to replace or repair parts, while thermal epoxy is much harder to remove and will leave hardened residue stuck to both surfaces.

Another thermal interface material is the graphite pad. These are thin sheets of thermally conductive material that you can use instead of thermal epoxy or thermal paste. Graphite pads can be cut with scissors to size, usually have one or both sides coated in a weak adhesive, and are available in several different thicknesses. One thing to watch out for with graphite pads is that they can be weakly electrically conductive. If you use graphite pads normally, this probably won't affect you, but I once tried to slide some thermistors down between a heatsink and a PCB that had a graphite sheet between them, and ran into problems when small pieces of the graphite sheet caused my thermistor values to be wrong.

Sometimes you need to move heat before you can dump it out to a heat sink. The best way to do this is with a heat pipe. Heat pipes are thin, flat copper tubes that are sealed at both ends. Inside is usually a little water. They're great at moving heat from a small surface (like the top of an IC) into a larger heat sink. Heat pipes can be fairly low-cost and are great for applications where there's not enough room for a heat sink directly on top of the heat source.

It's very important that you make sure there isn't any air or empty space between a heat source and the rest of the thermal system. Air is a great insulator, which means you'll have a large thermal resistance in your thermal system and all of your carefully designed heat dissipation methods won't be doing any good.

One more technique for dissipating heat is to use a thermally conductive resin or potting material. If you're planning on potting your design (see the "Fabrication and Assembly" chapter for more information about potting), choosing a thermally conductive material can let you treat your entire enclosure and all of the potting material as a heat sink. Heat will be absorbed into the potting material, as well as efficiently conducted between the heat source and the enclosure that the potting material is making contact with.

To determine exactly what thermal management materials and techniques are necessary for your particular design, you'll need to perform a full analysis of your thermals. A great resource that walks you through how to do this is the TI Application Note AN-2020: "Thermal Design by Insight, Not Hindsight". Rather than reproducing the contents of that application note here, I recommend you read it and use it as a resource to help you calculate each piece of your thermal circuit and determine what you need and what you don't.

5.7 Avoiding mistakes

When importing or exporting between the myriad of CAD formats, check the result carefully. It's all too common for imports to work incorrectly and subtle mistakes can creep in. Missing solder mask, keep-outs, polygons, and silkscreen are some of the more common import problems. Check everything.

You can use other DRC checkers besides the one that comes with your CAD program. For example, Mentor Graphics offers a program specifically for importing your design and running design checks on it. It can perform more in-depth checks and even do simulations on your design to find mistakes that your CAD program's DRC might miss. You can find a link to several DRC checkers on this book's website, DesigningElectronics.com.

If your PCB fabricator provides a DRC file, use it. If they don't, you'll need to create your own using the design rules listed on the PCB fabricator's website. Using the existing or suggested rules will help keep your cost down. Once you're producing boards in the thousands, or even in the hundreds, custom design rules won't have as big of an impact on your board cost.

When you run your DRC check, it should pass with no errors. If there are errors or warnings that you don't fix, you should have a good reason for every instance and it should be documented.

Make sure that your silkscreen font size and weight are above the minimums recommended by the manufacturer. Going too small or skinny will result in unreadable silkscreen text.

Before you send out your PCB for fabrication, print it out on a piece of paper at a 1:1 scale. If you have the housing for the PCB already done or mocked up, you can do a fit check and make sure the holes all line up. If you have any of the parts in your BOM on hand, you can line them up with the printed-out footprints and make sure they're correct. You can also use this paper model to check connector orientation and pinouts to make sure everything will mate correctly.

After you've exported the gerber files, open them in a gerber viewer like gerbv or ViewMate[5]. Some CAD software may not automatically repour polygons and flood fills when you expect it to, and polygon pour errors can cause shorts. Check for shorts or left-over thermal isolation, especially on inner layers, since those are the hardest to rework. Additionally, someone else who did not draw the layout should review the completed layout and gerbers.

[5]Both of these tools can be found on the website for this book, DesigningElectronics.com.

Make sure your DRC is set up to run all checks in batch mode. DRC rules can be checked continuously as you route traces (known as "online" mode) or only when you click the DRC check button (known as "batch" mode). In some CAD software (like Altium), it's possible to accidentally check only a subset of the DRC rules, even after running both batch and online DRC. In Altium, you can see all DRC rules under DRC->Rules to Check. Make sure that all of the rules you want to check are selected to be checked in either online or batch mode.

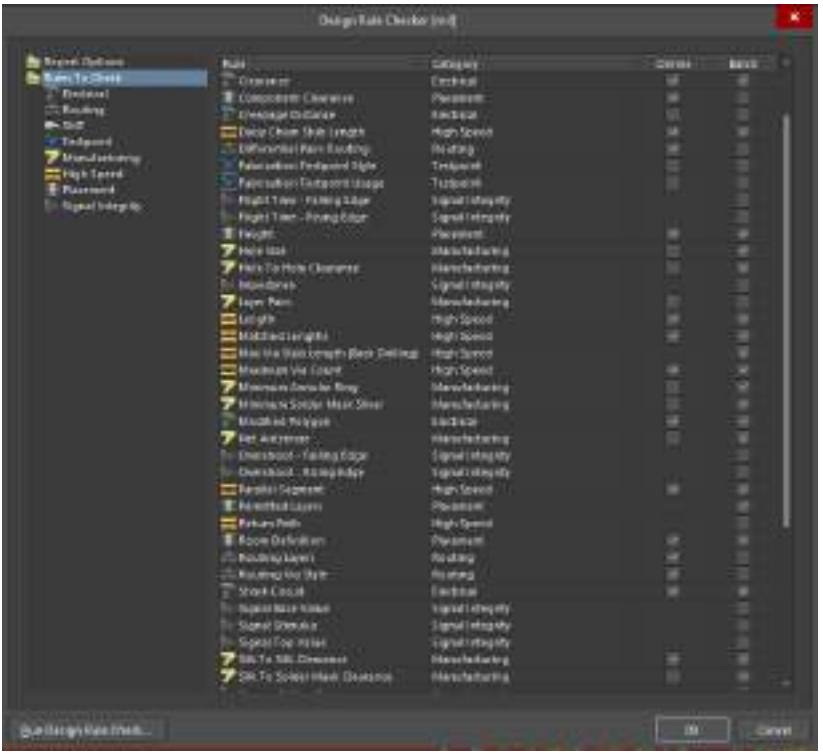

Figure 5.43: Notice how there are some DRC rules here that will not be checked in either batch or online mode.

Even if all DRC rules pass, you need to make sure your DRC and ERC rules and criteria are correct in the first place. If the manufacturer has a DRC file that can be used, use it. But don't assume that it will perfectly cover everything. Sometimes the provided DRC rule files don't include all of the rules or are for a different manufacturing process (e.g., two layers versus four layers).

Remember to verify every pin and dimension of any footprints you didn't draw. Digi-Key, the part's manufacturer, services like SnapEDA, or random Github repositories are all potential sources of part libraries that you can use to save time. However, you absolutely need to check every dimension, the silkscreen, the aperture, the pin assignments, the schematic symbol, and paste mask against the recommendations of the datasheet. If a datasheet doesn't specify a particular parameter (like the paste mask, for example), use the value taken from the relevant JEDEC standard.

It is exceptionally easy to make a pin assignment mistake on small 3 pin MOSFETs and BJTs. There are many variants of the same package with different pin mappings and very subtle naming differences in the part that are easy to miss, often only a single letter in a 14 character manufacturer part number. Give extra scrutiny to these components. One way to avoid mistakes on those parts is to name the pins and pads "B", "E", and "C" (Base, Emitter, Collector) for a BJT or "G", "S", and "D" (Gate, Source, Drain) for MOSFETs. Using this naming convention instead of numbers will let you re-use the same schematic symbol for multiple footprints while avoiding pin to pad mapping errors.

Don't place component designators under the component if you can help it. You'll want the designators to be visible after assembly to aid in bring up, debugging, and rework. Additionally, keep the pin one marker out from underneath the component for the same reason. Some people like to use a square with one corner cut off to outline where the chip sits in the footprint and designate where pin one is. You can do that, but it's better if you also use a silkscreen dot next to pin one so that it's still visible after assembly.

Don't put untented vias too close to the pad of an IC, especially if those untented vias are underneath the IC. Solder can bridge the pad and the untented via, which causes a short that is completely invisible without removing the chip. I have seen it happen even on TQFP parts. There will be no externally visible shorts on the legs themselves, but solder can wick underneath the chip and cause an invisible short there. The best way to deal with this problem is to move the vias further away from the chip pads, out from underneath the chip if possible, or to just tent the vias.

5.8 EMC and EMI

In the United States, one job of the Federal Communications Commission (FCC) is to enforce regulations designed to keep electronic devices from interfering with each other. The FCC does this by limiting the

fields that electronic products are allowed to emit, and by requiring that devices must remain operational even if there are interfering fields present from other devices. Together, these two requirements are commonly referred to as electromagnetic compatibility (EMC). Being able to accept interference from other devices is called electromagnetic immunity, but when people use the acronym EMI, they're often referring to electromagnetic interference. The specific laws regarding these requirements are often referred to as "FCC part 15" (really Title 47 CFR Part 15). These are likely the laws that you'll need to follow, at least in the United States, but check with a lawyer or compliance lab to be sure.

If your device intentionally radiates, the FCC and other regulators around the world verify that your product meets field emission requirements by testing it in a lab. Even if your product doesn't intentionally radiate, you are still required to meet unintentional radiation limits. Some big retailers may require that you have a third-party verify that your product meets FCC requirements before stocking it. Exactly what is required depends on the country that you're selling in, so make sure to consult a compliance lab to understand what's required of you.

To ensure that your device will comply with EMC standards, you need to make sure you don't accidentally design antennas into your layout. Long traces and loops can easily radiate. The fields that are emitted from a current carrying loop are proportional to the current, the square of the loop radius, and the square of the frequency. And don't forget that ground/return currents are just as important as currents flowing into components.

Don't put slots in your PCB. Slots on the board forces currents to move around the hole, which creates large current loops. This also holds for slots in copper planes, not just holes in the board. Don't route any signals on your ground plane. Some application notes or datasheets may direct you to use a smaller, local ground plane around an IC to try to improve isolation. This is most commonly seen in ADCs. It's important to do this correctly to prevent severe EMC problems. Do not punch a hole in your ground plane to inset another split ground plane. Instead, connect the two ground planes at a single point, underneath the ADC where your digital and analog signals meet. You'll also want to keep the distance between your split ground planes at least 20 times the height of the signal traces above the ground planes on the layer beneath them. Figure 5.44 shows a cross sectional view of this and figure 5.45 shows the top-down view.

Figure 5.44: Side view of a PCB showing minimum separation required for a split ground plane (not to scale).

Figure 5.45: Top-down view of a PCB showing split planes connected at a single point and minimum separation.

Some designers believe that you should basically never split a ground plane.[6] You can almost always accomplish what you want by moving sub-circuits far apart rather than putting them on a split a ground plane. This is why it's so important to think about placement first, and to clump sub-circuits together.

The exception to splitting ground planes is when you need galvanic isolation. This is often done for safety reasons. To maintain galvanic isolation, you can't run any traces over the gap between the two ground planes, and this will prevent any EMC problems. The only things that are allowed to cross the gap are optical signals (like in an optoisolator) or fields (like in a transformer).

In general, the area of the loop is more important than the length of the trace. A long and skinny trace that goes out but then returns directly underneath itself will have a long trace length, but a small loop area. A shorter trace that goes out into a large circle before

[6]There's a joke among EMC consultants that goes like this: "What do you call an engineer who splits a ground plane? A customer."

coming back may have a shorter radius, but a larger loop area. The best strategy is to try to keep traces as short as you can, route them straight to their destination (unless you need to meander for length matching purposes), and ensure there is a return current path directly underneath or very close to the outgoing trace.

A common misconception is that current flows along the path of least resistance. That's only true at DC and very low frequencies. Once frequency increases, current flows along the path of least inductance. This is because at high frequencies, impedance on your PCB is dominated by inductance. That means that a trace carrying any signal above about 1 MHz will have its return current flowing directly underneath it. That's why it's so important to use a ground plane under an RF trace and why a cut in the ground plane can make things worse: the return current now has to make a larger loop by moving around the edge of the cut instead of flowing right underneath the RF trace. Every single current must eventually return to its source. Remember, even if the frequency of your signal is low, fast *edges* on that signal will contain lots of high frequency content. So even a 10 kHz clock signal with sharp, square edges will contain high frequencies way above 10 kHz and can easily cause you to fail emissions testing if you accidentally create a big loop for the return current.

If a signal has high frequency content (that is, it has either fast edges or a high fundamental frequency), don't route it under components or traces that are used for I/O that leaves the PCB. If a signal with high frequency content capacitively or inductively couples to an input or output signal that will leave the board, EMC problems can occur because of how long the path back to ground can be for the return current.

Route differential signals together and length match them. If you want them to behave like differential signals and not just two single ended signals, their spacing and width must be precise. Use an online calculator or a tool like LineCalc from Keysight ADS[7] to figure out the exact dimensions required. Routing a differential pair incorrectly can lead to both signal integrity problems and radiated emissions problems.

DDR, PCI Express, SATA, and USB are all examples of buses that that have high-speed differential pairs. If you need to transition a differential pair to another layer, you need to make sure that there is a good current return path near that transition. Figure 5.46 shows the right way to do this.

[7]Links to these tools can be found on this book's website, DesigningElectronics.com.

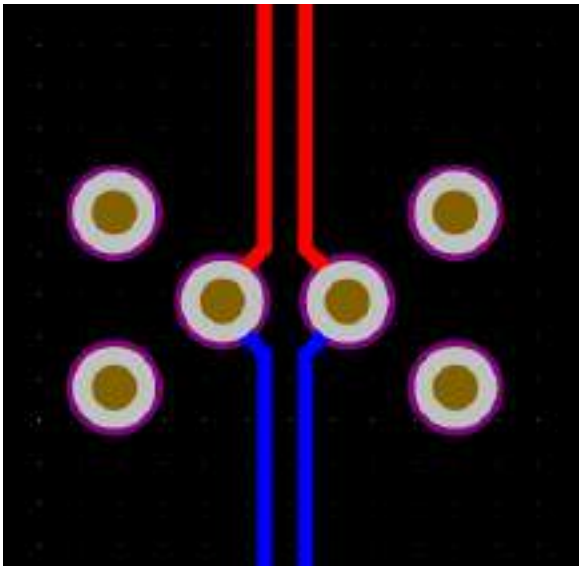

Figure 5.46: Differential pair via transition with ground vias near the signal vias to provide a low impedance path for return currents.

If you're routing very high-speed signals (greater than about 5 Gigatransfers per second) and your trace runs in a straight line for more than four inches, you need to be concerned about something called the fiber weave effect. This is caused by one trace in a differential pair running directly on top of the fiberglass weave in your PCB substrate and the other trace running directly on top of the space between weaves. This results in two different dielectric constants and can cause signal integrity problems. Figure 5.47 shows what a route vulnerable to the fiber weave effect looks like.

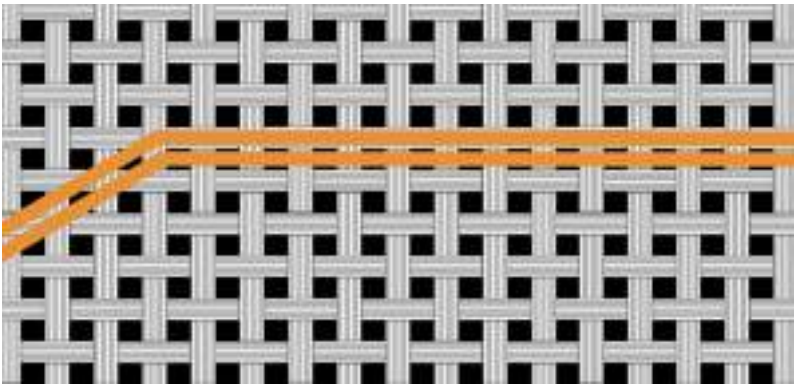

Figure 5.47: The fiber weave effect is caused by different dielectric constants under each trace.

To combat this effect, you can use a zig-zagging pattern to ensure that both traces experience roughly the same dielectric constant throughout their path. Figure 5.48 shows how this works.

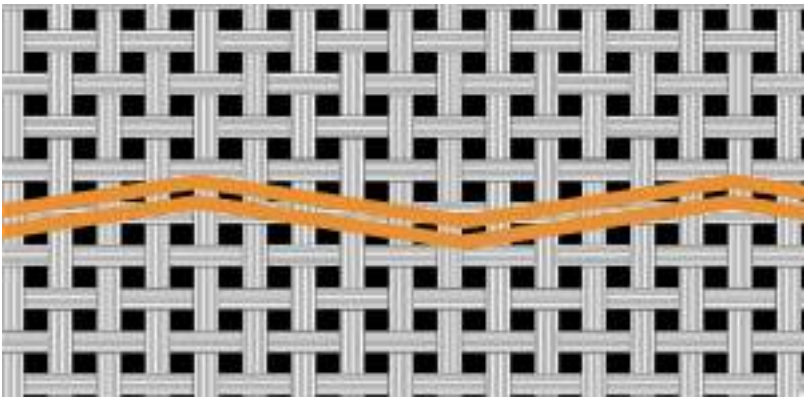

Figure 5.48: Routing method to combat the fiber weave effect.

Physical isolation can be helpful in preventing noise and EMI from one component or subsystem from contaminating other components or subsystems. Switching power supplies can be a large source of noise as a result of the fast switching of the FET. Keep switching power supplies isolated from one another and far away from antennas, RF signals, and any sensitive analog components or high-speed digital components.

Another important way to keep your signals clean is to place filters

for incoming and outgoing signals as close to the edge of the board as possible. If a "dirty" input signal makes it to the middle of the PCB before it gets filtered, that noise can couple into other signals and components before it's filtered out. Similarly, if an output signal is filtered near the middle of the board instead of the edge, noise that is in the bandwidth of the filter can couple back into the signal from other components and signals on the PCB before it leaves. In both cases, you're defeating the purpose of the filters. Note that this also applies to cables, not just signals on a PCB. If a cable with a filtered signal is run next to the cable with the unfiltered signal, the filter won't do much good.

Other sensitive signals need to be physically isolated too. Differential pairs and traces that are length or impedance-controlled should not have any other signals running parallel to them for long distances. This can result in an accidental coupling effect or can throw off the otherwise carefully controlled structure of your traces. If a trace is impedance-controlled (using a structure like stripline, microstrip, or coplanar waveguide), there is a mandatory separation distance on all sides of the traces that must be adhered to if you expect to get the same performance that you calculated. There are lots of online calculators you can use, or you can use a tool like LineCalc that comes with Keysight ADS[8]. Whatever you use, these calculators will specify a separation distance between every feature. If you're not using a particular structure but are still concerned about accidental coupling, use a separation distance of between 3 and 5 times the width of your trace. After that distance, the field strength is low enough that coupling isn't usually a problem.

You should try to keep all traces as short as you can, but high-speed or fast edged signals are especially important to keep short. The longer those traces are, the more likely it is that they will couple into other traces.

Correctly using bypass capacitors involves more than just part selection. Bypass capacitors also need a low inductance path to power, ground, and the IC they are servicing. To accomplish this, place bypass capacitors physically near the IC. Smaller value capacitors should be closest, followed by larger values. If there's room, use multiple vias to connect the capacitors to power and ground. For high-speed applications where performance is critical, you can use via-in-pad so there is zero distance between the capacitor terminals and their path to ground. If you don't use via-in-pad technology, then you'll still want

[8]Links to these calculators can be found on this book's website, DesigningElectronics.com.

to keep the vias as close to the pad as you can. The longer the path from the capacitor terminal to the ground plane or power plane, the more stray inductance there will be and the worse your capacitor will be at bypassing.

Connector placement matters. Any connectors represent an opportunity for energy to radiate out into the world because most connectors and cables happen to act like good antennas for at least one frequency. One common way this can happen is if you have connectors on opposite ends of a PCB. Those two connectors can easily act as a dipole antenna and cause EMI problems. Instead, all connectors should be placed on the same side of the PCB. Sometimes this is impossible for product design reasons, but it will significantly mitigate the risk of EMI problems. If you have multiple signals going to a single offboard device, run them all through the same connector and cable. Splitting groups of signals up and running them through two connectors and cables before recombining them has a similar effect as putting connectors on opposite sides of the PCB. It can create common mode voltages that turn the cables into antennas. Other techniques that can help reduce cable emission problems include common mode filters, ferrite chokes, and shielded cables.

It's also important to keep any signals that are high-speed, RF, or have fast edges away from other connectors. High frequency signals generate a lot of harmonics and can more easily induce a voltage in connectors and cables, causing them to act like antennas. If you need a high-speed or RF trace to leave the PCB, use a controlled structure like stripline, microstrip, or coplanar waveguide and keep source/generating circuitry away from the connector. Controlled impedance structures will give you more contained, predictable EM fields. This will make it easier to shield and mitigate any EMI problems that result.

Stackup matters a lot in preventing EMI and EMC problems. The name of the game is containing fields, and your stackup can go a long way in preventing stray EM fields from coupling into other signals or feeding accidental antennas. It is very important to have at least one complete ground plane. Keep fast clocks, signals with fast edges, and other high-speed signals sandwiched between ground planes or power and ground planes on inner layers. RF traces on the top layer using coplanar waveguide or microstrip can be really helpful when trying to debug or bring up a PCB (since you can easily add a pigtail, cut a trace, or add filtering and matching parts), but those structures will also radiate more than a buried structure like stripline will. All of these structures should be kept away from the edge of the PCB. Fields can more easily escape into the world when high-speed signals are run along

the edge of a PCB. A good rule of thumb is to leave about 400 mils between the edge of the board and the high-speed trace for any sections of the trace that run parallel to the edge.

Recall that we have been treating signals with fast rise times similarly to RF and high-speed signals because a fast rising or falling edge must have high frequency components to it. Again, think about the Fourier decomposition of a square wave. An ideal square wave has infinitely fast edges, and requires infinitely many odd harmonics to create. As you reduce the number of odd harmonics, the edges will take on a smaller and smaller slope. In electronics design, this is actually a good thing (to a point). Digital ICs will often have a timing diagram in their datasheet that will specify the minimum edge needed to function properly. You should try to reduce your clock and data line edges to be just above this minimum. The idea is that a slower edge will result in less high frequency content in that signal, which in turn will reduce the likelihood of EMI problems. Higher frequency signals more easily couple into components and traces than low frequency signals because a shorter wavelength means there are more things that are a significant fraction of the wavelength, and will therefore act like an antenna. The best way to reduce the speed of edges is to use either series resistors or ferrite beads. The time it takes a signal to traverse a trace should be at least 5 times as long as the rise time of that signal.

When routing power, utilize a power plane. That way instead of routing traces all over the place (inviting the opportunity for coupling and radiation of high frequency noise into other traces and components), you can just drop a via straight down to your power plane. If you have more than one voltage in your design, use multiple planes on the same layer. Circuits using the same voltage should be clumped together anyway, and keeping all power planes on the same layer prevents power planes from overlapping and capacitively coupling noise into one another. Keep a healthy clearance of 100 mils between plane edges. This helps minimize coupling along the edges of the planes. If your application has high voltage, a larger clearance may be necessary to prevent arcing.

Another reason to use a power and ground plane is that simplifying your routing can also improve performance. Rather than routing a pad to a via through a trace and then connecting it to ground, having a power and ground plane allows you to remove the extra inductance of that trace and only incur the stray inductance of a via. This can be really helpful for the performance of, for example, power supplies. Bypass capacitors are more effective over a larger frequency range when using this technique.

It's important to understand that the energy of any electrical signal is contained in its fields. That means that if you have a power plane above a ground plane, the energy will be stored in the fields within the dielectric between them. Once you think about it this way, it makes sense why it would be a bad idea to insert another signal layer between your power plane and ground plane, since it will be running right through the fields of your power plane and the two will crosstalk.

If you use more than one ground plane in your design, they need to be connected together often. The vias that are spread all around your board to do this are often called stitching vias. In sensitive areas, like around RF traces or between parts and signals that shouldn't couple, you can add what are called shielding vias, or a via fence. This is just a group of vias that ensures a low inductance path to ground for any return currents and prevents ground planes on multiple layers from accumulating a significant common mode voltage. If you've got any kind of high-speed, RF, or fast edges on your board, adding shielding vias can only help you. They'll help ensure that everything is at the same potential, and that there is always a low impedance path to ground for your return currents.

One place you don't want to use lots of signal vias is high-speed and RF traces. Vias in a transmission line cause reflections, which increases the loss and decreases the power through the trace. Vias can be specially designed into a transmission line to minimize this loss, but avoid them if at all possible.

If you're particularly worried about a certain area on the PCB radiating, or if you have some very sensitive circuitry, you can use a can-type shield. Can shields are little metal lids that sit on top of the PCB and keep electric and magnetic fields from getting in or out. You can either solder the shield perimeter down directly to the board, or you can solder down clips that the shield presses into. The advantage of using clips is that it's much easier to rework, debug, and measure the circuit being shielded. Using clips is slightly more expensive because you need to buy both the clips and the shield, but it's still highly recommended that you use clips. Even if you have the design finalized and totally perfect, using shields that are easily removed will make repairs easier when units fail in the field and are returned. Shields can either be solid or they can have holes in them. As long as the holes are very small compared to the wavelength of the signals you're trying to block, the shield will still work fine and you can get the advantage of a little more air circulation for heat dissipation. For the shield to be effective at a particular frequency, the size of the holes must be 5 to 10 times smaller than the wavelength. Shields can also be shapes other

than just a square or rectangle. Most shield manufacturers can custom make shields of whatever dimensions or shapes you need. If you're on a budget, look for pre-made shields and design your sensitive circuits to fit within those areas.

Emissions are only half of the equation. Your device also needs to be immune to interference caused by other devices. In addition to the techniques discussed above, you can also use firmware to assist. Things like watchdog timers, memory canaries, checksums, EDAC, and status signals from peripherals can help you detect errors and deal with them appropriately.

6

Cost engineering

Cost engineering is the design phase where you try to reduce not only the cost of components, but assembly and repair costs too. Do not start cost engineering too soon. First design to meet your product specifications, and only then should you begin to look for replacement parts that still meet all specifications. Cost engineering is where products go from long lasting to cheap crap if you're not careful. Cost engineering has failed if it causes any of the specifications to no longer be met, or if it significantly degrades the user experience. Two things may have identical functionality, but if one is light weight and has a plastic enclosure, and the other is hefty and has a brushed metal enclosure, the user experience will be different. One will connote a feeling of luxury and importance, while the other will come across as inferior and cheap. Think about the impression you want to make on your users. There is not one right answer here either. It might be okay if something feels cheap if it's designed to be disposable, for example.

Cost engineering also does not happen in a vacuum. Your mechanical engineers and industrial designers will be major contributors to this effort. Talk to them to see how your design is constraining their work and see if there are any changes you can implement that can reduce cost for another member of the engineering team.

The best way to reduce the cost of your design is to remove things. The most obvious example of this is to take entire components out of the design, but this isn't always possible. Instead, think about the other things you can reduce: the number of bits in ADCs or DACs, the amount of memory in your EEPROM, the resolution of your display, etc.

Cost engineering by looking for replacement ICs can get dangerous. For something like a power supply IC, the process of qualifying and test-

ing the new part should be fairly straightforward. But for something like an EEPROM or other peripheral, new firmware will probably need to be written, which will significantly impact your schedule. If you decide to switch microcontrollers or processors, a huge amount of work will need to be done to integrate the new part. Any cost engineering of this kind should be done as soon as possible at the beginning of development. The importance of a low BOM cost should be expressed in the requirements so that only one set of firmware is ever written.

How you approach component cost engineering depends on your production size and your market. For a very large production volume, saving one cent on one component can result in a lot of money saved in total. For a small production volume, many cost saving strategies are not worth the time to source a new part. So before you go looking for a capacitor that costs $0.005 instead of $0.01, calculate what you'll actually be saving.

As with everything, the more you buy, the cheaper it is. You can see some of the price breaks for volume on Mouser or Digi-Key, but they usually only go up to about 10,000 pieces. Past that, you should talk to the manufacturer or distributor sales reps. They will usually be able to beat the prices at Digi-Key and Mouser at both large and small quantities, so it's worth it to contact them and get quotes.

6.1 Cost Reduction in your Schematic

Don't be afraid to put down footprints for parts even if you're not sure you'll use them. You can always just leave them unpopulated. If you need a DC connection through the unpopulated footprint (like if you remove a series capacitor, for example), you can simply populate a zero ohm resistor of the same size. Maintaining a DC connection through an unpopulated IC footprint is trickier, and requires that you design in that option. But having the flexibility to try different circuits or add and remove parts can let you experiment while keeping you from having to fabricate another PCB iteration. Zero ohm resistors can also be used to jump over traces on the top layer, effectively giving you another layer on the PCB for free. For RF signals, there are purpose made RF jumpers that can be used to hop over perpendicular transmission lines without significant loss or reflection.

While it may not always be useful to try to save fractions of a cent on components in a small production run, both large and small production runs can benefit from reducing the number of unique parts. For example, if you have a pull-up resistor on one GPIO pin that is 10k, and another pull-up resistor on a different GPIO pin that is 15k,

you probably don't need two separate values. Using two 10k resistors instead of one 10k and one 15k will reduce the number of feeders used on the pick and place machine during assembly. If you're manually assembling the boards, it will be faster too. In both cases, a lower unique part count will result in lower assembly costs. There is an exception to this rule: high tolerance parts. High tolerance parts are expensive, and you should only use them where they are needed. Do not accidentally change out a part that needs to be high tolerance with a standard tolerance part in an attempt to reduce unique part count. Similarly, do not change all values of a single part to be high tolerance if only one of them actually needs to be high tolerance.

One way to lower the total part count is to use resistor and capacitor arrays. These can only be used if you have several resistors or capacitors of the same value that are physically close to each other. The advantage is that you again save assembly cost because it's faster to populate a single part than it is to populate 4 parts. Keep in mind though, that using an array may increase your *unique* part count by one if not all resistors of that value can use an array. For example, if you have five 10k resistors on a PCB and four of them are close enough to use an array, you have now increased your *unique* part count by one, since you need a single 10k resistor and a 10k array resistor. One way to prevent this is to use the array even in places where you don't need to use every resistor in the array. Resistor arrays cost more than individual resistors though, so you'll need to do a little cost analysis to determine whether it's worth it to use an array at all.

You may be tempted to replace a load switch IC with a FET. Given that load switches are designed to have a very low on resistance and that they typically have ESD protection built in already, it's usually not worth it to replace a load switch IC with a FET from a cost perspective. Load switches are already very inexpensive, and usually cost less than buying a FET and ESD diode.

A lot of modern microcontrollers and processors have an internal oscillator. If your application can accept the reduced accuracy and lower clock rate of an internal oscillator, you can eliminate the need for an external crystal oscillator.

Use as few connectors in your design as possible. The cost of connectors can add up quickly. Instead, consider using soldered wires. This process can be done quickly in production using something called hot bar soldering. A hot bar (sometimes called a thermode) is laid across the wires on the pads and all the solder melts at the same time. It's fast and suitable for mass manufacturing. Hot bar soldering can be used on wires or on flex cables. Another option is to use a card

style connection, where the PCB has exposed contacts along one edge that plug into a receptacle on another board. This reduces the parts requirement from two connectors and a cable to just a single connector. However, connections that require a high signal integrity (high-speed or RF signals) will have to use a properly rated cable and connector.

One great place to save money on connectors is the programming connector. During prototyping and development, you will be using the programming port all the time. But during production assembly, you'll use it probably once to flash the firmware. Rather than paying for a connector that gets used once, you can use a component-less programming header. A product called TagConnect is a popular example of this. A special cable with pogo pins and plastic clips latches onto a specially design footprint on the PCB, makes good contact, programs your board, and is then removed. This has the advantage of not adding the cost of a connector to your BOM as well as making it easy to access the programming pins again later during repair or maintenance. You can also integrate a TagConnect programmer (or just some pogo pins that accomplish the same thing) into your bed of nails test jig that you can use to both program and verify functionality of your product during production. See the "Testing" chapter for more information on how to do that.

Discrete components of the same values will often cost different amounts depending on the package that the come in. Usually 0603 size discretes will be a good tradeoff of cost versus ease of assembly and rework. You can also check to see if any parts you're using are cheaper when using a different package.

If you need a higher level of ESD protection, stacking two ESD diodes in series is sometimes cheaper than using a single ESD diode with a higher rating.

6.2 Cost Reduction in your Layout

The easier a PCB is to manufacture, the cheaper it will be. This sounds obvious, but what makes a PCB easy to manufacture? The simplest answer to this is to use some standardized design choices that the PCB manufacturer stipulates, like a minimum trace width, minimum space between copper, and a minimum via size. A common set of requirements for this cheapest tier of manufacturing might be a 6 mil minimum trace width, a 6 mil minimum space between copper, and a 10 mil minimum via hole size. These requirements will be different for every manufacturer, so look them up on the website of whoever you decide to use. As before, this is an area where making a trace one

mil wider might not make any difference to you, but can impact your manufacturing cost greatly.

Once you start adding features like HDI, blind vias, buried vias, micro vias, exceptionally tiny trace widths, thicker copper pours, and plugged vias, cost will start to quickly mount. It usually doesn't matter how *many* of these features you add, the cost increase will be the same. For example, adding a single blind via versus adding 100 blind vias won't really increase your cost (assuming they're all between the same layers). The increase comes when you go from zero blind vias to any number greater than zero.

Following manufacturer recommended design rules can also speed up manufacturing time. A standard design that uses tolerances and techniques that the manufacturer can quickly execute can usually be fabricated in a couple of days, or even as quickly as a single day in some cases. More complex designs can take weeks.

In general, you should try to design PCBs with all parts on a single side. This will save you time and money in assembly, since only one size needs to be reflowed. If you must use two sides, try to use only surface mount components. If you use a mix of surface mount and through hole, your board will either need to be both reflowed and wave soldered, or it will require some manual assembly. Having parts on only a single side also makes building test fixtures much easier. Using only through hole or only surface mount parts will save your assembler time since they'll only need to run through a single process. Even a small number of parts on the bottom side of a board still requires that the boards go through a whole second reflow.

Wave soldering is sometimes done at a higher temperature than reflow soldering, so make sure your surface mount parts are rated for that higher temperature. Mixing surface mount and through hole parts on a single board is very possible, but be aware that it'll be more expensive.

The PCB substrate you use can have a big impact on cost. This is most relevant for RF designs where the PCB material must have a particular relative permittivity and loss tangent. One option for reducing substrate cost in a design is to use modules. Instead of manufacturing the entire PCB on the expensive substrate, you can use it for only the parts of the circuit that need it and treat that as a module that plugs in to the rest of the electronics. You can accomplish this by fabricating a separate smaller board on the expensive substrate and either using a connector to send signals back to your main board, or, if you only need to route low speed signals to and from your smaller board, make it a castellated module that is soldered on top of your main board. This

may not end up being cheaper for small designs, but for large designs it can ensure you are only paying for expensive substrate in the places that you absolutely need it.

If you need only a subset of your layers to be on a particular substrate, you can design a stackup that uses more than one core material type. In the "Stackup" section of the "Layout Design" chapter, we saw an example board that used Rogers 4350B on the top three RF layers and a standard FR-4 type material for the rest of the layers. Depending on how many boards you're making and what substrate you're using, this can be cheaper than making the entire board from a more exotic material.

Another trick to save money on PCB costs is to use a single PCB design for multiple product versions. This only works when multiple versions are mostly identical except for a few auxiliary features, but it can be very powerful. Let's say you have two versions of a product: one with wireless connectivity, and one without. Rather than design and fabricate two PCBs, design the more complete version (the one with a wireless IC) and then simply mark those parts as DNP (Do Not Populate) on the BOM for the simpler version. Some chips with different functionality have identical footprints and pinouts, which can also be used this way. If you want to sell a version of your product at a lower price with less memory, you can often find memory in the same family with the same pinout and footprint. All that's needed is to change one feeder on the assembly line and to flash a different firmware version.

A classic trick in prototyping PCBs is to take advantage of panelization. In quick turn prototypes, fabs often panelize without telling you. Since people are only ordering small quantities of boards, they stick them all together on the same big panel and fab them. That's why they restrict you to using certain materials and dimensions: everyone is sharing the same board.

Not every distributor sells the same parts for the same price. There are several different services you can use to compare component costs from different distributors. Personally, I like to use Octopart.com. Octopart allows you to upload your BOM and will automatically search through a list of dozens of distributors to see who has the part in stock, if they have enough for the number of boards you want to build, and what the cost per part is at that quantity. This is a great way to predict what the manufactured cost of your design will be at production volumes, even before you actually want to make that many. Links for Octopart and other similar sites are available on this book's website, DesigningElectronics.com.

Another tool you can use to predict component, PCB fabrication, and assembly production costs at volume without waiting days for a quote is CircuitHub.com. CircuitHub is a turnkey PCB manufacturer that lets you upload your design files and get an instant quote for anywhere from a single board to 10,000 copies. Even if you don't order from CircuitHub, it's quite instructive to upload your design and play around with the sliders to understand how quantity affects cost in your design.

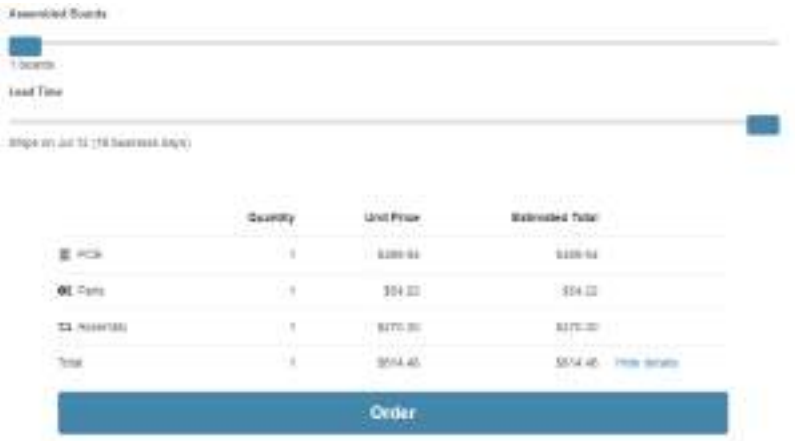

Figure 6.1: Circuithub quote for manufacturing a single board.

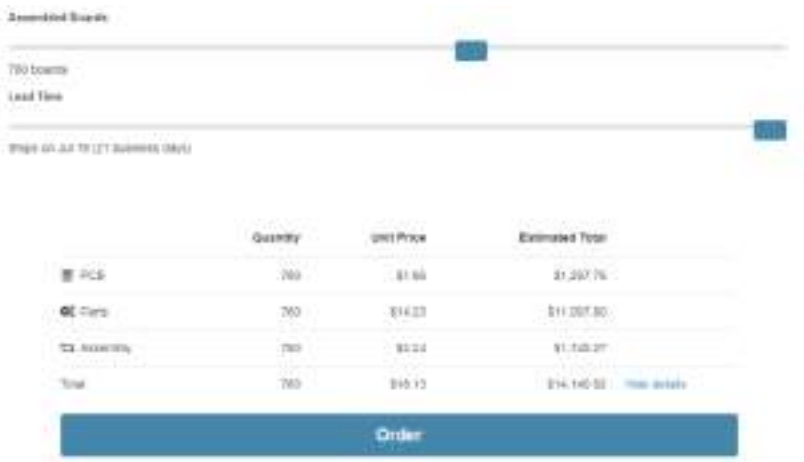

Figure 6.2: Circuithub quote for manufacturing several hundred of the same design.

6.3 Cost Reduction in your Assembly

A BOM contains the list of parts that are used to manufacture your device, but the BOM is a subset of the Cost of Goods Sold (COGS). The one aspect of the COGS that is directly affected by a device's electrical design is any manual assembly that needs to be done. The biggest item here will be hand soldering. Other manual steps may include applying glue or adhesive, screwing in fasteners, and building and installing wire harnesses. The fewer steps, the cheaper it will be.

There are also strategies for reducing the cost of the enclosure itself, but those are outside the scope of this book since your mechanical engineer will likely be handling that. However, it's important that you work with your mechanical engineer on this task because changes to an enclosure to reduce manufacturing cost will very likely impact your design, and may affect it in ways that your mechanical engineer is unaware (for instance, the effect of enclosure material on shielding properties).

Use as many industry standard measurements and dimensions as you can. If you design a PCB panel that's 1 cm bigger than the rail size at the factory, they're going to charge you a lot more because they have to retool their line for you. If you pick a solder mask that's a different color than the one they already have in their machine, they're going to charge you extra. Talk to your factory and ask them how you

can reduce their cost of labor and materials.

A method called chip-on-board can be used to reduce the cost of the ICs themselves. Normally you buy a chip in a package from the manufacturer. The manufacturer puts their die in a lead frame that contains the external pins and then molds a black plastic epoxy shell around it, resulting the familiar look of a chip. However, you can sometimes cut a better deal with manufacturers by buying the die by itself. To actually use this die in your design, you'll need to epoxy the die to the PCB, then use an automatic wire bonding machine to place tiny wire bonds between the pads on the die and the PCB. Then the entire thing is covered up with glob of protective epoxy. If you've ever taken apart a calculator and noticed the black blob in the middle of the PCB, you've seen a chip on board in the wild. This process only makes financial sense if you're doing a very large production volume, but can lend itself to a significant reduction in BOM cost.

7

Fabrication and Assembly

7.1 Preparing for Fabrication

One of the easiest ways to get unexpectedly behind schedule is underestimating the time between sending your design files to your PCB fabricator/assembler and when they actually start work. If the factory is missing files, can't open them because they're in the wrong format, or finds any issues that they think will impact the fabrication, they will place a hold on the order and work with you to resolve it. Because of the speed at which modern PCB fabricators move, a single hold order, even if it's resolved within a few hours, can move your ship date back by a day or more.

To avoid holds, run DRC and ERC checks before you generate your output files. All PCB fabricators will have a page on their website listing the DRC requirements for fabricating with them, so you can make sure you're checking against their specific design constraints. Some manufacturers will even provide a downloadable DRC file for common CAD tools so you can simply import it and run it. Other manufacturers have an online tool that allows you to upload your design files and automatically check for errors.

There are a lot of files a manufacturer needs to fabricate a PCB. A list of the most important files to send for a two-layer PCB is in table 7.1 (note that a PCB with more than two layers will have extra files not listed here).

File extension	Description
.gto	Top layer silkscreen
.gtl	Top layer copper
.gtp	Top layer solder paste
.gts	Top layer solder mask
.gbo	Bottom layer silkscreen
.gbl	Bottom layer copper
.gbp	Bottom layer solder paste
.gbs	Bottom layer solder mask
.gko	Keepout
.drl	Drill file

Table 7.1: Files needed to fabricate a PCB

The file format that every PCB fabricator uses is called the Gerber. This name is left over from the company that invented it in the 80's, but it's ubiquitous in electronics design. The current version is called RS-274X, so make sure your CAD program is generating that format instead of the obsolete RS-274-D (by default, all modern tools should be using the correct version). Since Gerber files are just text files that contain graphics information about where features are on your board, the file extension your CAD program generates may not be exactly the same as what's in the table above. Your CAD program may also generate many more output files than those listed here. For example, if you have more than two layers, you'll need another file for each layer. Your CAD program may also generate an outline file, mechanical layers, netlist, or other information about the design. Don't forget the drill file! It's really easy to forget to include it, since some CAD program export the drill file separately.

Even if the PCB fabricator doesn't ask for it, you can send them supporting files to help them better understand the design. One example of this is a ReadMe file. It's good practice to include a readme.txt file that lists the files you submitted, describes what they are, lists the stackup you're fabricating on, and any other design specific information that they might need to know. Sometimes a hold will be placed on a design if the fabricator sees something unusual or contrary to what most designs do. You can try to head this off by calling out what might look like a mistake but actually isn't. An alternative to sending a ReadMe file with specific instructions is to put those instructions as their own layer in your Gerbers. There's less of a chance that your fabricator will miss the information because they're going to have to

look at the Gerbers to fabricate your design, and they'll see your notes for sure.

If you're using a custom stackup, be sure to include a diagram that shows the material, substrate thickness, copper thickness, and dielectric constant for each layer. If you have any impedance-controlled lines, you should have a list of the nets that need to be impedance-controlled, their desired width, impedance, and which layer they are referenced to.

There's lots of information that you can call out in fabrication notes, and the more specific you are, the closer your board will be to what you expect. Here's an example set of fabrication notes for a design that you can use as a template for fabrication notes on your own design. Not all of these fields are necessary, and many fabricators already follow some of the call outs listed, but this gives you an example of a pretty complete and detailed set of fabrication notes.

Fabrication notes:

- All dimensions are in mil
- Linear and angular tolerances are 0.01", 0.5°
- Interpret in accordance with DOD-STD-100
- Number of layers = 8
- Fabricate per IPC-ML-6012B, Class 2
- Inspect per IPC-600 latest revision
- Material
  - Type: 370HR
  - Raw Cu weight outer layers (see stackup)
  - Raw Cu weight inner signal layers (see stackup)
  - Raw Cu weight inner plane layers (see stackup)
    - ☐ High Tg[1]

[1] Check this box if you want the fabricator to know that you need a material with a high glass transition temperature (Tg). If your fabricator doesn't have the exact material you specified in stock, they may use a different material with a very similar dielectric constant and loss tangent for convenience and speed. However, it may not have the same Tg, and calling out that you need a high Tg can keep them from accidentally doing this.

☑ RoHS[2]
☑ Halogen Free[3]

- Processing
  - All vias and plated through holes to be located within 75um of true position
  - Annular ring of 125um nominal, 100um minimum
  - Copper thieving not allowed
  - Remove all unused pads on artwork on all inner layers
  - Final conductor and pad widths to be within 25um of artwork originals
  - Registration +/- 50um
  - Etch date code and UL recognized vendor mark on secondary side[4]
  - Fabricator shall test and stamp passed circuits on the secondary side
  - Optimize circuit arrangement for best panel yield
  - Boards must be routed and free of burrs
- Impedance
  - Use requirements in stackup. If none, then stackup is total thickness and layer structure only
  - Non-controlled traces will be within 20% of specified aperture value in artwork
  - Fabricator must inform, beforehand, of deviations required to achieve impedance
  - Transmission line impedance measurement report required

[2]RoHS, or Reduction of Hazardous Substances, is a European directive that requires your design to not contain any of 10 hazardous substances, including lead. RoHS is often used interchangeably to mean "lead-free" even though there are nine other things you also must avoid to be RoHS compliant. This applies to both components and solder.

[3]Another environmental requirement. Materials that contain halogens like fluorine, chlorine, and bromine are often used in cable insulation and in some components. Burning these materials releases dangerous gases that are bad for the environment and can make fires in enclosed spaces even more dangerous for people trying to escape.

[4]You can specify the location for marks the fabricator puts on your board for spacing or aesthetic reasons.

- Cu Plating
  - Unless otherwise specified, hole dimensions are for finished holes[5]
  - Unless otherwise specified, hole barrel thickness to be 0.001" min[6]
- Finish plating: finish all exposed pads with
  - ☑ Electroless nickel/immersion gold (ENIG)
  - ☐ Immersion silver
  - ☐ Tin/lead (HASL)
  - ☐ Lead-free HASL
  - ☐ Hard gold over nickel (30um/80um)[7]
  - ☐ Organic solderability preserve (OSP)[8]
- Solder mask
  - Solder mask per IPC-SM-840, Type A, Class 2[9]
  - LPI blue solder mask over bare copper both sides
  - Thickness 25um +/- 10 um
  - Solder mask registration to be within +/- 0.003" of relevant layer[10]
  - No mask to appear on pads, except where specified by design (e.g. solder mask defined pads)
- Silkscreen
  - Use white non-conductive ink
  - No silkscreen to appear on pads
- Warp and twist not to exceed 0.010 inch/inch

[5]This specifies that a hole that's called out as, for example, 10 mils should be drilled slightly larger than 10 mils so that when it's plated with copper, the diameter shrinks to be a finished size of 10 mils.

[6]This is specifying the minimum plating thickness of the inside of a via.

[7]This means 30 microns of gold over 80 microns of nickel.

[8]OSP is a very thin coating that goes over the entire board to prevent oxidation and ensure that the pads are easy to solder to and aren't weakened by any compounds that may form on the pads between plating and assembly.

[9]This is the IPC standard for solder mask application

[10]This refers to the alignment of the solder mask layer with respect to the copper layer it is covering.

- Milling
  - All board dimensions to be held within 0.005"
  - If no dimensions present, use board outline to feed directly into milling processor
- Via special features
  - ☑ Via in pad
  - ☐ Plugged vias
  - ☐ Plugged nonconductive vias plated over flat and flush
  - ☑ Via in pad plugged conductive plated over flat and flush
  - ☐ Tented vias primary side
  - ☑ Tented vias secondary side
  - ☐ Blind vias
  - ☐ Buried vias
  - ☐ None
- Assembly
  - ☐ Leaded assembly
  - ☑ Lead-free assembly
  - ☑ Clean assembly
  - ☐ No-clean assembly
- All 8 mil vias to be plugged and plated over flat and flush
- All 5 mil vias to be Cu filled and plated over flat and flush
- All 9.84 mil hole to be tented on secondary side

It's a good idea to take one last look at your generated Gerber files before you send them out to make sure that everything exported the way you think it did. Gerbv and ViewMate are both good programs to use for this. Gerbv is free and open-source, and there is a free version of ViewMate. Using software like this is also good for discovering missing files, since you'll notice missing information when you open all the gerbers at once and see them stacked on top of each other. You can find links to gerbv and ViewMate on this book's website, DesigningElectronics.com.

A properly prepared BOM contains all the important information needed to purchase the right components from the right supplier, and

can speed the procurement process considerably. A BOM can have lots of different fields, but the most important ones are the **reference designator**, the **quantity**, and the **manufacturer part number**. Other useful fields are:

- Distributor part number
- Unit cost
- Subtotal cost
- Description
- Footprint
- Link to part on distributor website

There are an overwhelming number of PCB fabricators and assemblers. They vary in cost, speed, and quality. A good resource for getting simultaneous quotes from lots of different manufacturers at the same time is pcbshopper.com. This service lets you put in the basic design requirements of your board and returns the cost and lead time for fabricating it from 22 different PCB manufacturers. It will also give you a quote for assembling your board from 7 different assemblers. While this can be a great way to find the cheapest manufacturer for your design, it's up to you to vet the quality and capability of whoever you choose.

7.2 Buying parts

You have two options for buying parts: a distributor or the manufacturer. For the majority of your parts, and for smaller volume builds, you'll be likely sourcing all of your parts from distributors. Common, reputable distributors are Digi-Key, Mouser, Newark/Farnell, and Arrow. For RF parts, RFMW and Richardson RFPD are good places to go.

One interesting and unadvertised feature of the Mouser website is that the search bar understands Digi-Key part numbers. This is useful when trying to find parts that may be out of stock on Digi-Key, or if you're sourcing parts for a BOM on Mouser and only have Digi-Key part numbers. Note that Digi-Key does not understand Mouser part numbers.

Many distributor websites have a BOM import tool. These typically let you upload a BOM as an XLS or CSV and automatically search and add all of your parts to your cart at once. They'll also

usually alert you to any parts that are out of stock or have insufficient stock, and sometimes even suggest replacement parts. It's a good exercise to try to add all of your parts to your cart before you're ready to order them, just to make sure you will be able to actually buy all of your parts. The service mentioned before for checking multiple distributors at once, Octopart.com, also has a BOM upload tool and will search multiple distributors at once. It can tell you who has how much stock of what part and if you're in danger using unavailable parts.

When you're exporting your BOM to upload to a distributor website, make sure to use the original manufacturer part numbers rather than distributor part numbers to find parts. This will make your BOM agnostic to which distributor you buy parts from, and will keep you from having to update part numbers if the distributor changes anything. At a minimum, BOM upload tools require you to include columns for the manufacturer part number and quantity.

When ordering parts, some distributors allow you to specify a custom field for each part, sometimes called the "comment", which they will print on the bag of parts they deliver. I have found that it's very helpful to use this for the reference designator for each component and/or the name of the project. This helps a lot with organization, especially if you have multiple projects going at the same time. If your PCB has silkscreen labels for each reference designator, it's also much faster to hand assemble your board.

If you go to buy a part and notice that the price has suddenly increased to an incredible level, it's likely that the distributor either has a small number of those parts left and isn't planning on buying more, or the part has become end of life. In either case, this price spike is to discourage you from using that part. You should follow this suggestion and find a replacement, because it's very likely that there is some kind of supply chain issue and you won't be able to reliably get that part.

Some distributors offer to put parts on a reel for you, even if you buy less than a full reel's worth of parts (Digi-Key calls this service their Digi-Reel). This is great if you're using a pick and place machine, but can be annoying if you're just assembling by hand. Reels take up a lot of space and make it very difficult to store a small number of parts if you have any left over. A small antistatic bag with a label on it is much easier to deal with than a six-inch diameter reel with a short piece of cut tape that's always falling off. Only get small quantities of parts reeled if you actually need them on a reel. It'll also be cheaper, since Digi-Key charges a $7.00 fee per Digi-Reel.

Make sure you order parts far enough ahead of your build that

there's no bottleneck waiting for parts to arrive. If you're using a pick and place machine or an assembler, you'll need to get parts before you plan to start assembly. Using a pick and place machine requires setting up the feeders with the parts and programming the machine. The earlier you get parts to your production team, the more likely you are to stay on schedule.

Some component manufacturers and turnkey PCB manufacturers will program parts for you before or after production. For example, Microchip has a service that allows you to buy microcontrollers in tape and reel pre-programmed from the factory with your firmware image (https://www.microchipdirect.com). CircuitHub.com allows you to upload your firmware image when you're ordering your boards and have it programmed during manufacturing. This can be a great resource if you're building a large number of boards since you don't need to build a programming jig or worry about the logistics of flashing every single board. The downside of using a preprogrammed chip is that you need to have a completely finished firmware image to give to the manufacturer. If you ever need to update anything, you're going to be connecting to every single board and flashing them manually, which kind of defeats the whole purpose of having the manufacturer handle it. Depending on your design and if it's capable of connecting to any networks, you may be able to have the manufacturer upload a version of your firmware that does automatic updates by checking for a new firmware image at a hardcoded address or can perform over the air updates. Again, if anything goes wrong with the automatic update, you have to flash every board one by one.

7.3 Finding and working with CMs

If you're producing a product in any sort of serious quantity, you'll need to engage a contract manufacturer (CM). Choosing a CM is a big deal. If you choose poorly, your product can easily end up delayed or poorly made. Switching CMs is a long and expensive process. The best way to find a CM is to get a warm introduction from someone who is happy with their CM. Talk to your friends, people you know in the industry, people you know at other companies, your investors, your investor's other portfolio companies, etc. You can also cold contact CMs by finding them on sites like globalsources.com, hktdc.com, makersrow.com, or even alibaba.com.

Don't assume that it will be cheaper to manufacture in China. It used to almost always be cheaper to manufacture in China, but rising wages there have changed the economics of manufacturing. If

you're in the US, choosing a CM that's also in the US will make travel and communication with the factory much easier, and may reduce the cost of shipping and logistics for the finished product (since it doesn't require shipping across the Pacific Ocean). The intellectual property laws in the US are both much stronger and actually enforced, so if you're worried about IP theft, manufacturing in the US is a good way to mitigate that risk. Mexico can be a good middle ground between the US and China, since it can be cost effective, but it still fairly easy to travel to. The other advantage that manufacturing in North America has is that the time zone differences between you and your factory will be smaller, and you will have fewer language and cultural barriers with your factory.

Whichever CM you pick needs to be closely aligned to what you're trying to make. Obviously you wouldn't go to a textile factory to produce a phone, but you also probably don't want to go to a factory that only makes TV remote controls. Find a CM that is already making something similar to your product. They'll have experience with similar components and requirements and that will ultimately save you both a lot of time and money.

One of the first questions your CM will ask you is how many units you want them to make. There are many different sizes of factories and engaging the wrong size factory will result in either waste or stress. Do your best to come up with a realistic estimate of the number of units you'll want produced for up to two years of production. The larger your error on this number, the more difficult it will be to keep your factory happy. This is especially true if you overestimate the number of units you want, since the CM may hire more people or buy expensive equipment in anticipation of handling a larger order volume, only to be left footing the bill.

Be responsive to your CM. This is harder if you're based in the US and they're based in China, since they're starting up production for the day just as you're getting ready for bed. It is in both of your best interests to keep production moving at all times. It can be extremely beneficial to have someone at the factory during the beginning stages of production to help fix the inevitable issues that will arise. The CM can fix problems with their equipment, but if your devices start to fail tests at a high rate, they will have no idea how to fix it and will have to hold until you can get someone into the factory to debug it. If you can't have someone in the factory, have your phone on you at all times, read your emails and respond to them as quickly as possible, and be ready to go to the factory at a moment's notice. Nothing is more frustrating than trying to work with someone else and having them be unresponsive.

If you have to make design changes, make them as minimal as possible. Ideally, you won't have to make any changes after production has started. If you do have to make changes, using the contingency features on your PCB that you designed in can make those changes much less expensive, since hopefully all you need to do is change what parts are populated where. This isn't always possible, and if a significant change is necessary, it may end up being quite expensive and will delay production. As you may expect, the further along you are in production, the more expensive changes are.

Learn from your CM. They have probably been doing this longer than you have, and they can offer suggestions to improve your DFM. They will have a lot of intuition about the manufacturing process that can help keep your product on schedule and free of problems. Respect the people on their team and treat them as your partners.

Once you start talking about money with your CM, they'll probably want either net 30, net 60, or net 90 terms. This is standard lingo that just means you need to pay in full either 30, 60, or 90 days after delivery. There are other financing options available that will be specific for your situation and what your CM is comfortable with, but the important thing to know is that you need to talk about payment and figure out a solution that works for everyone.

According to Fictiv's 2018 hardware review survey, 54% of developers consider themselves very or extremely knowledgeable about manufacturing processes. But only 27% of manufacturers consider their clients to be very or extremely knowledgeable about manufacturing. People submit things to manufacturers all the time that are unmanufacturable. You need people with manufacturing experience to review your design. If you don't have anyone on your team who has that experience, you can contract someone, or even better, you can work with your CM. Don't assume you know more about manufacturing than your CM. If they ask you to change something, it's because they want to give your product the best shot possible at being manufactured correctly on the first try. The most common place you and your CM will butt heads about this is on industrial design. Big companies like Apple famously go to extreme lengths and cost to never sacrifice any industrial design requirements. You are not Apple (probably). If you decide to act like Apple when you are not Apple, your product will never launch. You will run into preventable manufacturing problems because you don't have the time, money, or expertise to not compromise on design.

The other big problem that contributes to manufacturing errors and delays is miscommunication. You are especially vulnerable to miscommunication if you and your CM are several time zones apart, you

don't share a first language, or if you're from two different cultures. It's imperative that you communicate clearly and frequently, probably more clearly and frequently than you think is necessary.

It's important that your fabricator use a standard and proven process to inspect and control the quality of your boards. There are three main standards that your fabricator probably uses. The first is DOD-STD-100, which is a US military standard on engineering drawing practices. This standard describes how to draw mechanical and engineering drawings for all kinds of purposes, not just PCB design. The concepts in this standard are so widely known and followed that no one should have to refer to the standard to figure out what your drawing is trying to show, but it does define some conventions and best practices for clear engineering drawings, which can be helpful if you don't have much experience drawing them. Explicitly calling out this standard removes any ambiguity for the person interpreting the drawings.

The second main standard your fabricator will use is IPC-6012, which is the standard for "Qualification and Performance Specification for Rigid Printed Boards". If you're building a flex circuit, then IPC-6013 is the standard for that. These standards specify the level of inspection and test done on a board, and break it up into four different levels from least to greatest performance. Class 1 is the lowest level of performance and has poor inspection and test. However, it's also the cheapest to have done. Boards that get inspected to this level are things like the electronics in a fast-food meal toy. It's almost completely inconsequential if the board doesn't work, and they're being manufactured as cheaply as possible. Class 2 is the normal level of inspection and test, and most boards are manufactured at this level. If you don't specify an IPC class, this is the class your fabricator will use. Class 3 is for high reliability applications like safety or life critical devices, where a failure may put lives in danger. Military and aerospace devices are also usually manufactured as Class 3. There is a fourth IPC class called Class 3/A, which is even more stringent and is specifically for space and military avionics. As you might expect, the higher class you specify, the most expensive it will be. However, the difference between a Class 2 board and a Class 3 board is not as great as you might think. It's not uncommon that a Class 2 board may happen to meet most or even all of the requirements of Class 3.

The final standard that your fabricator will use is IPC-600, which is "Acceptability of Printed Boards". This standard sort of goes hand in hand with IPC-6012 and pertains specifically to inspection of a finished board. It helps your fabricator know when to accept or reject a board because of manufacturing defects.

Certifications can be one way to ascertain the quality and abilities of a PCB manufacturer or assembler. Some common certifications can be found in table 7.2.

Name	Meaning
ISO 9001	Comprehensive quality management certification. Good manufacturers will have this. There are multiple versions, one from 2008 and an updated version from 2015.
ISO 13485	Quality management system for the design and manufacture of medical devices. This is a standard, and manufacturers can be certified as being compliant to this standard. FDA may require this certification if you're making a medical device.
ITAR	This isn't really a certification, it just means that the company has been registered with the State Department's Directorate of Defense Trade Controls. This is necessary if they will be exporting physical things or information related to defense that is covered under ITAR. Dealing with a product that is regulated under ITAR has a lot of other requirements, but that's out of the scope of this book.
AS9100C/D	This is the same thing as ISO 9001, but adds extra requirements for aerospace applications. The last letter in the name is the revision. The current revision as of 2021 is revision D.
UL PCB	There are two versions of this certification: full recognition and flame only. Flame only just ensures that the materials and processes used result in a board that meets certain fire ratings. Full recognition also checks tolerances and bounds on the final product (things like solder mask thickness, copper thickness, etc.). Good manufacturers will have at least one of these certifications.
Dodd Frank Conflict Metals Statement	Not a real certification, but important. Gold, tin, tantalum, and tungsten are conflict minerals, meaning that buying them from a disreputable source can result in you indirectly financing conflict in the Democratic Republic of the Congo, where they're mined. Having this statement just means that these materials are procured through State Department approved channels that do not support this conflict.

REACH	Not a standard, but a company can say they are compliant. REACH is a European directive that bans certain hazardous chemicals from any part of a product (including the electronics, paint, enclosure, etc.).
RoHS	Same thing as REACH, but only applies to the electrical part of a product (the components, solder, and PCB).
MIL-PRF-31032	Military standard for making PCBs. If someone says a board is MIL-SPEC, they probably mean it meets these standards. Also not a certification, just something you can be compliant with.
MIL-PRF-55110	Old version of the military PCB specification. This has now been superseded by MIL-PRF-31032.
IPC	A huge body of standards relating to PCB production and assembly. Good assemblers (the individual people, not the company) are IPC certified. Since IPC also makes standards, it's also possible to be IPC compliant but not IPC certified.
JEDEC	Another large set of standards that guide PCB production and assembly. Good manufacturers should be compliant.

Table 7.2: PCB certifications and what they mean

Beware that just having these certifications doesn't automatically mean your boards will be perfect. Some less scrupulous companies let their certifications expire but still claim to have them, and of course it's possible that the company was once able to pass these certifications but no longer can (even if the certification hasn't expired). You should also make sure the certifications they claim to have actually exist. Some Chinese fabricators have been known to put certificates and plaques up on their wall that look real, but aren't from any official standards organization and are meaningless.

7.4 PCB fabrication

You have two options for PCB fabrication: do it yourself or have a factory do it. Given the cost and abilities of modern PCB fabrication companies, you should almost always have a factory do it. Etching a PCB is an educational exercise that is good to do once, but you can typically only have copper on a single side and small features (less than about 10 mils) aren't possible. Any through holes have to be drilled

manually and won't be plated unless you stuff a wire through the hole and solder it to both sides, or use very tiny rivets. Etching requires caustic chemicals and should really only be used if you need a simple PCB within a few hours. It can also be easy to accidentally under-etch or over-etch your board, resulting in the defects seen in figure 7.2 and figure 7.3.

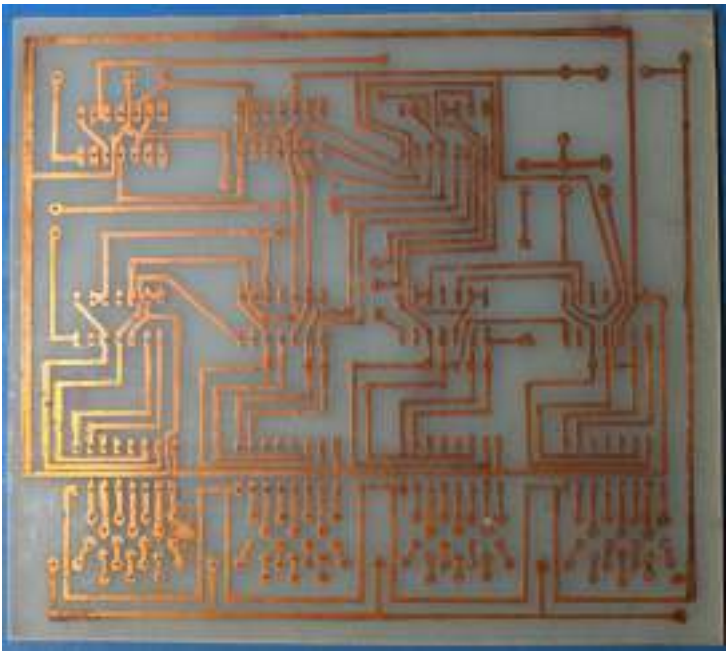

Figure 7.1: An example of a homemade etched PCB.

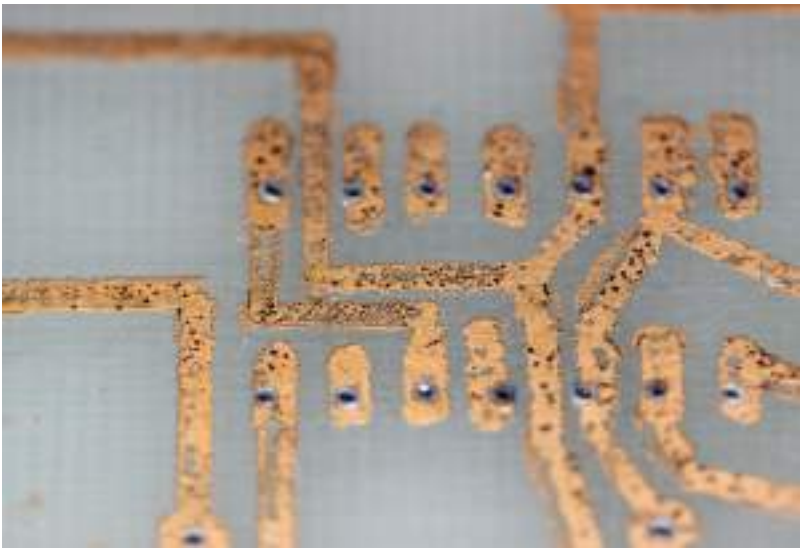

Figure 7.2: This copper trace has been either poorly coated in photoresist, left in the etch bath too long, or both. This trace will not be able to carry the current that a trace of this width is normally rated to carry. It's also at risk of causing an open circuit.

Figure 7.3: Extra copper is visible here because too much photoresist was used and some contaminated this area that was meant to be removed. Alternatively, this effect can be caused by too little or incomplete etching acid coverage. This defect risks causing a short.

Choosing which company should manufacture your PCB and assemble it is an important decision. All fabricators are not created equal, and who you chose may change depending on the time, budget, and sensitivity of your design. Table 7.3 lists a few fabricators that cost less, but won't do any exotic or highly sensitive manufacturing. In other words, if you need a simple two- or four-layer board *without* blind vias, particular copper weights, very small trace widths, or other special requirements, the companies in table 7.3 may be a good option. If you need some or all of those special requirements, and you need it fast, table 7.4 lists some fabricators that are probably a better fit. Note that the companies in both of these tables are only a small fraction of the companies out there, and the way they're sorted here is based on my personal experience with them as a designer in the Northern California area. The quality of the builds that these companies produce may vary, and a board house that one person had a terrible experience with may have been just fine for someone else. I'm not guaranteeing anything about them. Use these reviews as a starting point, not as an absolute ranking.

Name	Notes
OSHPark	Very popular with hobbyists, good quality. Offers two- and four-layer services, as well as flex. Normal lead time is nine to 12 days, fastest lead time is four to five days. Their four-layer service is on FR408, which is good enough for some RF applications. US based.
Seeed Studio Fusion	Offers one-, two-, four-, and six-layer boards on FR-4, as well as flex. Allows for a little more customization than OSHPark. Lead time is three to four days. China based.
ITead	Offers two- and four-layer FR-4, along with flex. Four to six day lead time. China based.
Advanced Circuits	Their "Barebones" service limits designs to two layers with no silkscreen or solder mask and a maximum size, but has a 24 hour turn time. They also offer a "33 each" service that charges $99 for three copies of a two-layer board with a three day turn. Their "66 each" service charges $264 for four copies of a four-layer board with a five day turn. All of these services are on FR-4. US based.
JLCPCB	One-, two-, four-, and six-layer boards on FR4. A little more customization is possible, kind of like Seeed Studio Fusion. Three day turn time. China based.
PCBway	Lots of customization possible. Their cheapest service is $5 for 10 copies of a two-layer board with a 24 hour turn time. They also offer up to 14 layer boards as well as flex. China based.
PCB.ng	Offers both PCB fabrication and assembly, and you must buy both together. Starts at $6 per board for fabrication and assembly, with around a minimum $36 order. Turn time for their fast and more expensive service is five days, and is 12 days for their slower and cheaper service. US based.

Table 7.3: Low-Cost PCB Fabricators

Name	Notes
Royal Circuits	Lots of exotic materials in stock. Lead times and prices can sometimes be high, but are usually pretty fast to quote. US based.
EuroCircuits	Popular European fabricator with specific price tiers for standard, controlled impedance, RF, and flex boards.
Sierra Circuits	Decent pricing and lead times. US based.
Gorilla Circuits	PCB fabrication as well as assembly. Very high quality. US based.

Table 7.4: High-performance PCB fabricators

Many hobbyists use the manufacturers in table 7.3 for side projects, but they can also be used for quick and simple prototypes or proofs of concept. The best way to find good companies that aren't in these tables is to ask people you know who have fabricated PCBs before. Who did they use? What was their experience like?

Assemblers can be a separate company that receives PCBs after they have been fabricated, or they can be integrated with the PCB manufacturer as a turnkey solution. You can consign parts to assemblers (meaning you ship them components you want them to use), or they can procure them all for you. Note that many assemblers charge about a 10% markup on materials and parts they buy, so you can consign parts if you're trying to save money. You can also sometimes negotiate this markup down. Assembly can be as much of an art as a science, so experience is what you should be looking for when choosing an assembler.

Another option you have to manufacture circuit boards is to mill them yourself. PCB mills work by milling away copper from a piece of copper clad board to create the footprints and traces on your PCB. The nice part about this is that you can design a board and have it in your hand in a matter of hours rather than days or weeks. However, this is not without tradeoffs. Milled boards do not have solder mask, do not have plated vias or holes (unless you apply a specific process to plate your holes), and can only mill features down to about 10 mils. This means that any chip with a pitch smaller than 10 mils (which you will mostly encounter with SMT ICs) cannot be implemented on a mill. It's best used for through hole parts, large pitch SMT parts, and structures like antennas.

There are two main companies that make circuit mills: Bantam Tools and LPKF. The Bantam Tools Desktop PCB Milling Machine (previously known as the Othermill) is a lower cost option that starts at about $3200. As the name implies, it's a small machine and can only mill material that is 4.5 inches by 5.5 inches.

The second option is one of many machines made by a company called LPKF (colloquially, people often refer to the machines as "an LPKF", even though that's really the name of the company, not the machine). LPKF makes machines that do both mechanical milling and laser milling. These mills are much more capable than the Bantam Tools machine, but they are also over an order of magnitude more expensive. LPKF mills can create PCBs with spacing as small as 2 mils. There are also options for additional equipment that you can use to make conductive vias, plated vias, multilayer boards, and even add solder mask. The mills without lasers cost between $20k and $30k.

Milling boards yourself involves a large initial purchase as well as expensive consumables (endmills and substrate), but they can let you iterate a design in hours instead of weeks. This is really powerful if you're trying to move quickly.

7.5 How Assembly Works

Now that we've talked about *where* to get your boards assembled, let's talk about how it actually works. There are several ways to approach PCB assembly depending on the resources available. We'll look at three different production levels and discuss the best way to do assembly given the resources you have.

First, for a large company doing a large production run of a single product, there will be component engineers, process engineers, and production engineers all working together with the external factory(ies) and internal production lines to tune the production precisely for that product. In this situation, you'll build custom fixtures to aid in assembly and work to optimize the yield of your assembly line. Assembly of the electronics will be followed by a functional test and then packaging and shipping. At this scale, there are entire companies that are capable of handling each of those steps for you. Because very large production runs are so specialized, we'll focus mostly on smaller production sizes.

Small startups will usually not have the resources to do manufacturing and assembly in house, so they'll contract it out to a factory or turnkey service. Several employees will work with the factory to ensure assembly is done correctly. You'll start by selecting a manufacturing partner. Get quotes from multiple companies that you trust. The best

way to start identifying good companies is to look at factories that people you trust have recommended. Talk to other hardware startups (cold email them if necessary) and get an idea of who will do a good job. Make sure they're at least ISO 9001 certified (and their certification is current), but you may also require other certifications depending on what you're manufacturing. If you're doing a one-off prototype run, find a company that knows what they're doing and has a lot of experience. If you're doing entire product production runs rather than just PCB assembly, it's more important to establish a relationship with the factory owner. As a small startup, you may not initially be bringing in the same level of revenue as a larger established company, so the factory is making a bet on you and hoping that you'll grow and become a long term partner with them.

If you're a hobbyist or an early-stage startup, using a factory for assembly might cost too much money. However, with the right techniques, you can assemble quality PCBs at home in low volumes. The next section includes some assembly techniques you can use if you're soldering together boards yourself.

7.6 Soldering techniques

This section should be prefaced by saying that the best way to learn soldering is to watch videos of good soldering techniques[11], and to try it yourself. Soldering is a very dynamic process, and it's hard to convey what things should look like as the solder gets to the right temperature and how it flows around component leads without using video. This book assumes that the reader is capable of at least basic through hole soldering. Having good soldering skills vastly increases the amount and kinds of rework that you can perform, which can save you time, money, and revisions. If you don't know the basics yet, definitely take the time to learn them and practice. People often get intimidated by surface mount or hot air soldering, but it's really not very difficult. If you don't want to fabricate your own PCB to practice SMT and hot air soldering, there are links to some good practice kits on DesigningElectronics.com.

Soldering with an iron is all about having the right soldering tip, temperature, and using the right solder and flux. If you're having trouble getting a solder joint to flow correctly, either increase the size of the tip you're using, increase the temperature of the iron, or try adding a little flux. One mistake that people often make while soldering

[11] You can find some of these videos on this book's website, DesigningElectronics.com.

is applying too much pressure to the iron. Pressing down hard doesn't make the heat transfer any faster and will just damage your soldering tip.

If your soldering iron has an adjustable temperature, make sure you're managing the temperature correctly. If it's too cold, the solder won't melt well. If it's too hot, you can damage parts or cause traces to peel up from the board.

Always apply solder to the solder *joint* and not the tip of the soldering iron. You can tin the tip of the iron with a little solder before you start to remove any oxides and improve heat transfer, but when you're actually soldering a component, use the iron to heat the solder joint and apply solder to that joint. Use the right size solder tip for the job. Larger tips will heat components faster, but are also harder to get into tight places. You'll know when a solder joint is successful when it looks shiny and the solder "flows" around the joint. If the solder looks dull or beads up, you probably have a cold solder joint, which isn't electrically or mechanically robust. Don't use too much solder either. You want a nice fillet or chamfer on the pins of your component. Again, beading up means you used too much. All you need is enough to cover the pad and pin in a thin layer.

Flux is a wonderful tool that can make soldering significantly easier. Flux is a sticky tree sap-based material (synthetic is also available) that lowers the melting point of solder and increases the surface tension. If you're having a hard time getting solder to flow, try adding a little flux and it will usually start behaving. Don't mind the puff of smoke, that's normal. In fact, pretty much all of the smoke you see when soldering is actually flux, not lead. You can buy flux as a liquid or as a thicker paste. Pure liquid flux can get messy and spread out very thin, but flux paste can be moved around and targeted to a specific area. Flux is usually applied with a syringe or a Q-Tip. It's also available in pens. Flux pens kind of look like a felt tip pen, except instead of ink, the tip is soaked in flux. A little bit of flux goes a long way, so using flux pens is a convenient way of dispensing the right amount of flux and having a portable supply of flux with you that doesn't get messy.

Some flux (like rosin flux) is marketed as "no-clean", meaning you don't need to clean the PCB after you use the flux. This does NOT mean that the flux will leave no residue: every flux leaves residue. No-clean flux is designed to leave residue that isn't harmful to the longevity of your PCB. If a flux is not marked as no-clean, you need to clean your PCB before it goes out the door because the residue it leaves behind could weaken components or cause premature failures. There are lots of ways to clean a PCB, including 99% isopropyl alcohol and an industrial

solvent called Ensolv (n-propyl bromide). When you buy flux, make sure you are buying flux for electronics. Plumbing flux is not the same thing but the packaging can look similar.

If you're having trouble getting the solder to not ball up on you as you heat it with your soldering iron, make sure your tip is in good condition. Use brass turnings instead of a wet sponge to clean the tip. The brass will do a better job at scraping off the oxide layer and will put less thermal shock on the tip. You can also buy tip cleaner, which is a little tray of material that you can use to coat your soldering iron tip before you turn it off. Coating the tip in solder before powering it off will help keep oxide from building up on the tip surface. This oxide is chiefly responsible for the areas of the tip that the solder doesn't seem to stick to.

You will need to specify whether your design is leaded or lead-free. Almost all products are now RoHS compliant, which means that they don't contain any lead or other environmentally harmful chemicals. You need to be careful if you use any components that are leaded, because they are designed to only survive a leaded manufacturing process. Lead-free assembly has to use higher temperatures during reflow because of the higher melting point of the lead-free alloy. However, it's pretty easy nowadays to design an entire board using only lead-free components without even realizing it. Still, you should check to make sure all of your parts can survive the temperatures of your assembly process. The only other reason you should consider using a leaded process is that it's a little easier to rework. Because lead has a lower melting point, repairing and reworking boards is easier and you're less likely to damage the PCB substrate itself (e.g., go beyond its glass transition point). When doing assembly by hand, use leaded solder paste. If your design is otherwise lead-free (which is should be), then you can easily use a lead-free assembly process when you move to larger scale production in a factory.

You need to put some thought into which solder paste to use, especially if you're doing production assembly. There's an entire field for understanding how solder paste (and other materials) flow and behave, called rheology. So if you hear someone referring to the "rheology" of the solder paste, that's what they're talking about. If you're assembling and reflowing by hand, using expired solder paste is probably ok. It's not going to behave as well as new stuff, and will be more clumpy and harder to work with. If you're using a solder paste stencil, new, unexpired solder paste will come off much nicer and leave clean edges on the solder pads. Messy edges can cause too much or too little solder paste to go down, which can cause opens or shorts. Make sure you

keep your solder paste stored at the right temperature. There is solder paste available that is temperature stable and doesn't need to be refrigerated, but most paste needs to be kept cold and dry. If it's not stored correctly, the flux that holds the tiny balls of solder in suspension will evaporate.

Run tests with solder paste before you do a full production. Make pastes prove themselves before you use them. Your CM and/or production team will probably have some specific solder paste brands and mixes that they know are reliable and work well with their line. The most common leaded solder to use is 63% tin and 37% lead. This is known as a eutectic mix because this exact amalgam will all melt at the same temperature. Other proportions of these two metals will result in a solder that is never completely liquid and always at least a little gummy. A popular lead-free alloy is 96.5% tin and 3.5% silver. You can find low temperature solder pastes (also lead-free) that contain primarily bismuth and have melting points as low as 140C. Some sensitive components can't survive the high reflow temperatures of lead-free solder or even leaded solder. A low temperature alloy commonly available is 57% bismuth, 42% tin, and 1% silver.

A good, well-known brand of solder to get is Kester, and they have lots of different alloys available. Another good company, especially for low temperature and thermally stable solder paste, is ChipQuik.

Solder paste is available in small jars or in syringes. The version that comes in a jar is good for hand stenciling. The version that comes in a syringe is good for manually applying a dot of solder paste to each pad. If you end up using a syringe, use the right size tip. Too large of a tip will make it impossible to control the amount of paste coming out you'll end up using way too much. Too small of a tip will take forever to populate a PCB.

When soldering surface mount parts with an iron, you can use a raking technique with QFP type packages to quickly solder every pin. Put down a thin layer of flux across the pins on one side of the package. Then place a blob of solder on one side and drag it across the pins with the iron. You should see the flux smoke and the solder grab onto each pin perfectly. If you have any shorts, remove them with solder braid. For parts with an exposed thermal tab (like a TO-220), use a large chisel tip to get enough heat for the solder to flow. For really large parts or parts that are very well grounded to a large ground plane (and thus have a huge thermal mass), use both hot air and a chisel tip to heat the area. This can damage the board or nearby parts, and it can also damage your soldering iron tips (or reduce their lifetime) so be very careful when using this technique. Heat sink any sensitive parts

that you're not trying to work on.

Another way to quickly solder lots of surface mount parts is to reflow them. There are several methods for this, but they all have the same first steps. You need to prepare the solder pads by applying solder paste. You can do this by hand using a syringe and squeezing out just enough paste to cover the surface of the pad. You can either use your thumb to depress the plunger in the syringe, or you can use an air powered approach. You can buy a regulator designed to dispense glue or solder paste for about $80, which includes a foot pedal you can step on to dispense a small amount of solder paste. You'll also need an air compressor that can do at least 100 PSI. The advantage of this setup is that it's much easier to control the amount and speed of the solder paste you're putting down, and it's much easier on your thumb.

A faster way to apply solder paste to an entire board is to use a stencil. Getting a solder stencil fabricated can be a cheap way to improve your DIY assembly process. These are especially useful when you have a board with a large number of surface mount components on it. Solder stencils can be made of either steel or Kapton and are designed to be lined up with the solder pads on the PCB. Then, a squeegee is used to spread solder paste across the entire stencil. In a process similar to silkscreening, the stencil is removed and solder is left only on the solder pads of the board. From there, you can use a pair of tweezers to carefully place each part onto the paste covered pads. Stencils can be used in a paste printer that automatically applies the paste, or you can do it by hand with careful alignment and some practice.

To improve the reflow of surface mount parts with ground pads underneath, it's a good idea to use a technique called "windowing", also known as grid printing. Rather than applying solder paste to the entire ground pad, it's applied in small squares. This makes it easier for volatiles like flux vapor to escape from underneath the chip. It also helps prevent the chip from lifting off of any pads along its edge as a result of too much solder beading together under the chip during reflow.

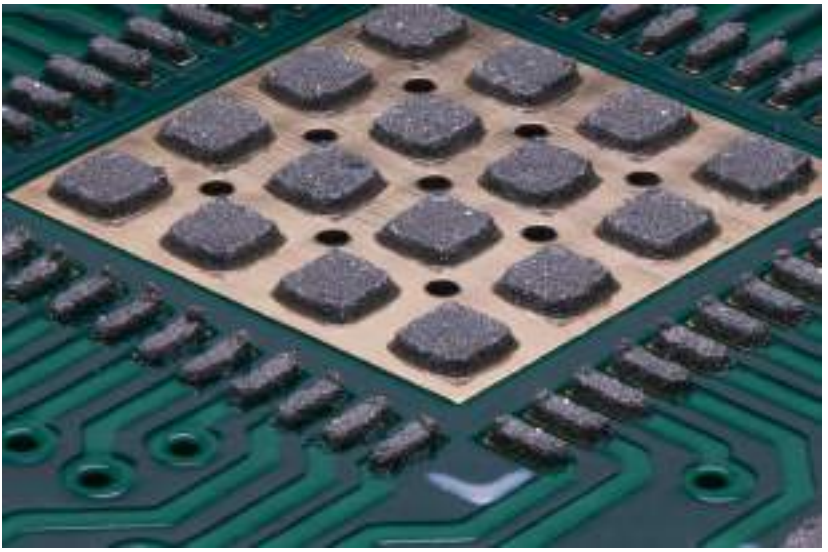

Figure 7.4: Grid printing on a QFN center pad. Image courtesy of Greg Davill.

Once the solder paste is applied, components are placed. Align the parts as best you can, but it doesn't need to be perfect. The next step, reflow, can be accomplished several ways. You can use a hot air rework station to heat a single component or an entire area of a board. Make sure the air speed isn't so high that it blows the parts away. Hold the hot air iron directly above the part you want to heat up and adjust the temperature at the board by lifting the iron higher or lower. You'll see the flux evaporate, the paste start to spread out slightly, and then it will suddenly melt, pull together, and turn shiny. The part will probably move around a little as the surface tension of the molten solder pulls it into perfect alignment. If you have a ground or thermal pad under the part that also needs to melt, you will probably need to wait another couple seconds for that to begin to flow too. You will probably see some flux start to leak out from under the chip and then the part will be "floating" on molten solder. You can confirm this by very gently tapping on any side of the chip with tweezers and seeing if it springs back into place. Once the part springs back from any perturbation on all sides, you can remove heat (by lifting directly up, so you don't blow the part out of alignment) and let the part cool. There shouldn't be any solder paste left and all joints should look fairly shiny. If there's any pins that look like they missed some solder, you can apply a little

directly to them with a syringe and heat the chip up again. There may be some larger balls of solder either on the component leads or on the PCB itself if you used too much solder paste. If they're on the PCB, you should be able to just knock them off or pick them off with tweezers, since they will likely be held on only by flux residue. If there are beads or shorts on component leads, you can either use solder wick to remove them, or you can heat the part again and use an X-Acto knife to push the molten bead off of the pin and onto to PCB to for easy removal after cooling. It requires a fine touch, but removing excess solder this way has less of a chance of leaving too *little* solder left, since surface tension will automatically distribute the solder correctly. Performing all of this under a microscope will make things significantly easier, and I highly recommend a good binocular microscope with a ring light.

In addition to a hot air reflow iron, you can also use hot surfaces or an oven. Hobbyists routinely reflow boards using a skillet or hot plate. Simply place the board with the components sitting on solder paste into the skillet or hot plate, turn it on, watch solder reflow. You need to be more careful with temperature here because it's possible to get skillets and hot plates above the glass transition temperature of FR-4, meaning you will cook the bottom of your PCB. The JEDEC standard limits soldering temperatures at 260C (for lead-free solder), but your parts will probably survive a little above that. The skillet and hot plate methods work best when assembling an entire board at once. It's not a great idea to heat the entire board up if you're just trying to work on a single part.

The method that professional assemblers use when building large quantities of boards is a reflow oven. There are two main types of reflow oven: infrared and vapor phase. Infrared reflow ovens use radiation as the primary method of heat transfer. The PCB runs through several different heating zones. First, the preheat zone slowly ramps up the temperature to prevent thermal shock and to keep parts from thermally expanding too quickly. Next, the soak phase increases temperature a little bit more and is designed to get everything up to the same temperature. Then the reflow phase quickly ramps up the temperature to just above the melting point of the solder paste and holds it just long enough for all of the components to actually get soldered. Finally, the cool down phase eases the board back down to room temperature. The times and temperatures that the industry uses for this process are described in JEDEC J-STD-020. Some reflow ovens do this whole process under a nitrogen atmosphere to improve the reflow of lead-free solder. IR ovens can have a tendency to heat unevenly and take up a lot of space, since many heating zones are needed.

Vapor phase reflow addresses these problems and is the latest and greatest reflow method. Vapor phase reflow uses convection instead of radiation as the heat transfer method. Perflouropolyether (aka PFPE, trade name Galden) is poured into the soldering chamber with the PCBs sitting on a grate above the liquid. The chamber is closed and as heating begins, the PFPE vaporizes and fills the chamber. Heat is transferred through the vapor into the board very evenly as the PFPE condenses on the PCB. The same temperature curve that is used in IR reflow ovens is followed, but a vapor deposition oven only needs a single chamber, so it takes up less space and overcomes the uneven heating problems of IR reflow.

While industrial reflow ovens are gigantic and cost tens of thousands of dollars, there are versions available for startups and hobbyists. They're almost all IR reflow[12], but they work just fine. There's a model from China that some people like called the T962. It's made by dozens of manufacturers, fits on a desk, and usually costs around $200-$300. You can find it on eBay or AliExpress. There is some replacement, open-source firmware that was developed by a hobbyist which supposedly improves the performance[13]. If you want to make your own version, there are lots of toaster-oven-to-reflow-oven conversion kits available online. They all basically do the same thing, which is run a PID loop to hit a particular temperature curve by looking at a temperature probe inside the oven and switching the oven heater relay on and off. Finally, the simplest version of this kind of reflow oven is an unmodified toaster oven. I have successfully reflowed many boards in a completely stock toaster oven that I bought simply because it was the cheapest one in the store. It's rare for toaster ovens to even be able to get above 260C, and hitting an exact temperature curve isn't necessary for one-off prototypes. However, you may see better results with more feature complete reflow ovens, especially as you reflow more and more boards. And it goes without saying that you should never put food in a toaster oven again after it's been converted into a piece of electronics assembly equipment.

7.7 Tips for Assembly

If you're not doing the assembly yourself, you will likely have a trained assembler doing it for you. Maybe a pick and place machine is doing

[12] As of early 2021, there is a company working on a small vapor phase oven, called the Vapor Phase One. It costs about $3000 and is slated to be available sometime in mid-2021.

[13] https://github.com/UnifiedEngineering/T-962-improvements

most of the work, but there's a good chance a person is involved along the way somehow. You should treat this person with the utmost respect and listen to their suggestions. They likely have way more experience than you in assembling electronics and they're intimately familiar with how the process works. If they suggest changing the assembly procedure, consider it carefully. There's a very good chance that they will be able to help improve assembly speed and accuracy. The other reason you should be very nice to assemblers is that you will find yourself relying on them to get you out of jams. If you screwed up your design and need it to be reworked, they'll do it. If you need your job rushed, they'll be the ones working overtime.

For assembly of the whole product (also known as kit assembly), make an assembly document or process. For very simple designs, this is probably trivial. But it can be surprisingly easy to paint yourself into a corner, so to speak. Assemble the first unit yourself (sometimes called the engineering model), and use it to figure out the order that things should be assembled. If there are subassemblies, they need to be tested before being integrated in the final assembly. Make sure there aren't any "impossible steps", e.g. screws that can't be tightened because of insufficient tool clearance. Think about possible assembly mistakes. Look for parts that can be rotated incorrectly or forced to fit in the wrong place. Once you've created a step-by-step instruction manual for assembling the device (with clear pictures of each step!), you can pass it off to your assemblers on your factory floor. Be sure to update that document if certain steps need to be clearer or if there are any problems or changes to the product. An easy way to format an assembly document is as a PowerPoint presentation. That way assemblers can have each step full screen on their monitor and can press a single button to move to the next step without scrolling around. Be sure to include notes and callouts. It should be clear, have plenty of diagrams, drawings, and pictures, and it should have quality checks in it so that the assembler can be sure each step was completed correctly. It should include tolerances, torques, maximum and minimum acceptable values, whether or not any equipment needs to be calibrated first (like a network analyzer), and of course the obvious stuff like the part numbers to use. Specify the color, gauge, and core type (solid or stranded) of any wires that are soldered in as part of the assembly process. If you use any cable ties or cable lacing, specify where and what size. Wire termination also needs to be specified. Does it need heat shrink? Does it need to be crimped or soldered or something else?

Travel sheets are a common way to help track inventory and manage quality during assembly and repair. A travel sheet (sometimes called

a traveler) is a document that travels around with the PCB during the production or repair process. It's a check list of everything that needs to be done to the board, when it happened, and who did it. You can accomplish this digitally with a spreadsheet, serial number, and bar code scanner, but having a physical piece of paper that people have to sign is much harder to forget about and easier to keep up to date. Travel sheets help ensure that procedures are being followed correctly. If something fails unexpectedly in the field, you can use its travel sheet to determine whether or not any steps were missed or performed incorrectly.

Inevitably, something during assembly or prototyping will require tape. If you are new to electronics, you may assume that electrical tape is the obvious choice here. Do not use electrical tape. Electrical tape is the worst. It leaves a sticky black residue, doesn't stick well to begin with, and is messy, imprecise, and doesn't work well at high temperatures. Instead, you should switch to polyimide tape. Sometimes referred to as Kapton (the 3M trade name for polyimide tape), it is a wonderous thing. It doesn't leave residue, it's electrically insulating, and can survive high temperatures (even direct application of a soldering iron). Every electronics lab should have at least one roll around. Its most common use is to tape wires in place and to provide a layer of insulation between conductive surfaces.

Similar to Kapton tape are so-called circuit dots or circuit tape. These look like small stickers that come on a sheet and can be peeled off and used to fix wires into place. This most common application of these are when a PCB has been reworked and has a "blue wire fix". To keep the repair clean and to keep the wire from being hung up on another part of the assembly, circuit tape can be used to stake the wire down at certain points on the PCB. Circuit tape does not leave residue but has a stickier adhesive than Kapton. It's not as tolerant to high temperatures as Kapton, but can be easier to work with in tiny pieces. This adhesive also has a cure time of about 72 hours, meaning it will increase in strength after it's applied. They generally come in two forms: small circles and a butterfly shape. The butterfly shape is specifically designed to hold down wires along its length. Circuitmedic.com sells these products and they will even send you a free sample sheet.

Another common tape used during assembly is copper tape. Copper tape can be used to tune antennas or as a shield. It's easy to solder and typically has a conductive adhesive, but be aware that the adhesive is not nearly as conductive as the copper itself. It's useful to have around because it can let you easily test shielding effectiveness during EMI testing and can even be used to replace ripped traces or pads.

Figure 7.5: Copper tape was used here to change the response of this antenna. Simple trial and error with a network analyzer dictated where the tape should go and how much should be used.

Using the right tools makes all the difference. When you're assembling anything with RF connectors (even when testing and experimenting in the lab), use a torque wrench designed for the connectors you're using. Finger tightening can result in insufficient torque and higher signal loss. Using the wrong torque wrench can result in overtightening the connector and breaking it. Different connectors need different torques. Unless you're working with very high frequency signals, you'll likely just need an SMA torque wrench, since that's the most common screw type RF connector typically encountered. You may also want an N-type wrench, since that's what most test equipment uses.

If your device will be subjected to particularly rough conditions during use, you can consider using a conformal coating or potting on the PCB. Conformal coating is a thin, clear liquid polymer film that completely covers a PCB. This prevents dust, moisture, and other environmental contaminants from affecting the PCB and components. It can also help prevent arcing between high voltage components that are close together and can prevent tin whiskers from forming. The advantage of conformal coating is that it's still possible to repair and rework the board and see all silkscreen and components after the coating is applied. You need to make sure that the coating you choose can survive the entire temperature range of your specifications.

Acrylic conformal coating is easy to remove using a mild solvent like acetone. Urethane and epoxy coatings are typically very hard, meaning they are more prone to cracking than other coatings. They're

Conformal Coating	Temperature Range (C)	Condensation Rating	Humidity Rating	Tin Whisker Mitigation	Difficulty of Rework
Acrylic	-65 to 125	Good	Excellent	Good	Low
Epoxy	-40 to 125	Poor	Poor	Excellent	High
Parylene	-65 to 200	Excellent	Excellent	Excellent	Very high
Silicone	-70 to 200	Excellent	Poor	Good	High
Urethane	-65 to 125	Poor	Good	Excellent	High

Table 7.5: Comparing different types of conformal coatings.

also quite hard to remove and can outgas, making them unsuitable for aerospace applications. Silicone can be removed with abrasion. It can also outgas, especially RTV (room temperature vulcanized rubber). If your silicone isn't fully cured, the materials that get outgassed can corrode your PCB. Parylene is great at protecting your board, but it has to be deposited in a vapor deposition chamber under vacuum. For these same reasons, it's almost impossible to remove. If you do manage to remove it, you need to re-coat the area with a new conformal coating material because Parylene doesn't stick to itself very well.

There are a few special cases where a conformal coating cannot be used. For example, a barometer IC has a small hole in the top to sense ambient pressure. If conformal coating is applied over this IC, it won't work. Care needs to be taken to ensure that all parts are able to be conformally coated. In general, if a part is not marked as no-clean, it can be safely coated. To maintain maximum repairability, it's not recommended to conformally coat a PCB unless you really need to. Try using a sealed enclosure instead. Your repair technicians will thank you.

An alternative to conformal coating is potting. Potting material can be either a rubbery or rigid resin material that fills the enclosure holding the PCB or is molded around the PCB. Potting material can be useful in frustrating reverse engineering efforts because it's difficult to remove to view components and traces and access signals. However, this also means that it's very difficult to repair or rework the PCB after it's potted. Some potting material can be removed with a solvent like acetone or by heating and plucking the material off with pliers. You need to make sure there aren't any air bubbles in the potting material if the device will be exposed to a wide range of pressures. Air pockets can expand and pop, damaging the board. Outgassing is another possibility that will need to be evaluated when dealing with products used in aerospace applications.

Some potting materials, adhesives, epoxies, and soft gaskets have a curing time associated with them. This can have a big impact on your assembly time if you have to wait for the material to fully cure before moving to the next step. If you get impatient, your material might not get cured all the way and can fail prematurely. One way to deal with this problem is to use light curable materials (LCM). These materials are nice because they cure very quickly (by shining UV light on them) and can immediately be inspected. It also removes the need to correctly mix multi-part epoxies. Of course, you need to be able to get light to whatever you're trying to cure. That may mean you need to make the channel that will contain the light curable

material transparent. You're also going to need to experiment with different LCMs. Find a material that works well with the materials that you're bonding together and that cures in the right way (e.g. soft or hard). Carefully follow the instructions and recommendations of the manufacturer. They will suggest certain curing lamps and dispensers to use. You also need to make sure your curing lamp meets the minimum brightness requirement to completely cure the LCM. You can do this by using a radiometer to measure your lamp. This would be an important step in an assembly process. LCMs can introduce other issues, like if your dispenser isn't opaque to light, your LCM will cure inside of your dispenser and clog it up. LCMs can be great, but you need to make sure your process for applying and curing them works well and that you follow all of the manufacturer recommended instructions.

Check whether any parts in your BOM are marked as "no-clean" in the datasheet. Normally after assembly, the PCB is cleaned with a solvent like isopropyl alcohol or sometimes n-propyl bromide. Some components have sensitive materials or structures that won't survive a cleaning process. If boards are being hand cleaned, you can just avoid the sensitive components. If they're being batch or automatically cleaned, you'll need to skip the cleaning step to prevent any damage.

Store components in humidity and ESD controlled environments before use. You can use something called a dry box, which is exactly what it sounds like: a purpose-built box that protects components from moisture. All components have a floor life, which is the amount of time they are rated to spend on the factory floor before they will have absorbed too much moisture and will cause assembly problems. Once the parts come out of their sealed bags, the timer starts. By placing components in a dry box, you can pause that timer. If you exceed the floor time, or if parts are exposed to higher humidity, they need to be baked out before assembly. Baking out parts involves putting them in an oven to slowly evaporate out any moisture that has migrated into the IC package. To prevent melting the plastic tape and reels or other packaging, baking is typically done at low temperatures for a long time. Baking can take anywhere from 3 hours to 79 days depending on the component and how it's packaged. IPC/JEDEC standard J-STD-033D contains tables you can use to figure out the temperature and time required to safety bake out your components. Parts that aren't stored properly can fail in strange ways. For example, too much moisture in an IC package can result in a phenomenon called "popcorning", where the water trapped inside the chip expands and causes voids.

All of that said, for small scale manufacturing or prototyping, you don't need to worry about getting a dry box or baking out parts. These

tools and techniques are really only important when you're dealing with a large number of parts and need high assembly yield. The easiest way to keep parts in a humidity and ESD controlled environment is to not open the packages they come in until just before assembly. Cut only a small slit in the top of the packages so you can store unused parts back in the same bag with the original desiccant packet and humidity card. Alternatively, you can buy zip lock ESD bags for very cheap and keep parts sealed in there. You should also keep PCBs stored this way. You can also save old zip lock ESD bags once you've used all the original parts to store other parts in later.

Testing is an extremely important part of the product manufacturing process. In fact, this book has an entire chapter dedicated to it! Your PCB manufacturer should do a 100% electrical test before shipping your boards. That means they will verify that the electrical connections on the PCB are exactly the same as those in the CAD. After you assemble the board, you'll need to perform functional tests to make sure all the components are working and that all solder joints are good. Note that if your design depends on high tolerance ceramic capacitors, you need to wait at least 48 hours after assembly before testing. For example, if you have a high-performance filter or sensitive analog front end that relies on ceramic capacitors being very close to their rated value, you will only see the expected tolerance after about 48 hours following reflow. This is because ceramic capacitors undergo an aging process that can affect their capacitance. Barium titanate molecules that make up the dielectric material of the capacitors will rearrange themselves during heating, and this process will continue for years after the initial heating. However, the largest change in capacitance as a result of this occurs within the first 24 to 48 hours. If you are just using ceramic capacitors for bypassing your IC power pins, this waiting period isn't necessary.

The bill of materials that you submit to your assembler needs to contain ALL the parts necessary for assembly, not just the parts that get soldered on the board. For example, if you need thermal compound, heat sinks, or jumpers, those need to be included.

You should always try to design boards that only have parts on one side. This makes assembly much easier, since the board only has to go through the reflow process once. If there are parts on both sides, glue or epoxy will be needed on one side to keep the parts from falling off once the solder melts again when the parts are upside down. If you need parts on both sides, you can use something like Loctite 348 to keep the parts attached to the board. In a production environment, this glue can be dispensed in a similar way to solder paste and at the

same time. The heat from the reflow oven will cure the glue. When the board goes through again, but upside down, the solder will melt, but the glue won't. The solder won't drip off because it will be held in place with surface tension.

Inspection is a critical part of assembly. If you're assembling boards yourself, carefully inspect the fabricated boards before you begin soldering them together. There's nothing worse than spending a lot of time soldering a board together only to discover that there was a fabrication error and the PCB should have been rejected. You should also inspect the board again after you've finished populating it. Use a microscope for this. If you're using an external assembler or turnkey service, it's their responsibility to do the PCB inspection before they begin assembly. Whenever I get a board in (assembled or not), I always take a picture of it before I do anything else. For bare boards, a flat bed scanner is a great way to capture the board in high resolution. That way I can always go back and see if an issue was present when the boards arrived, or if it was incurred during bring up and test. If you see a problem, tell your fabricator! They can help diagnose the issue. Reputable manufacturers will offer to refabricate the design for free if they made any mistakes. Having pictures of the boards before anything was done to them is a great way to compare manufacturers. You can check their tolerances, finishes, and quality to inform future decisions on which fabricator to use.

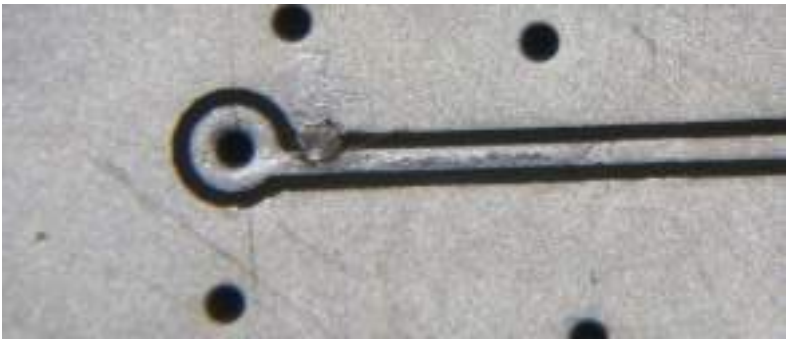

Figure 7.6: A thorough visual inspection under magnification will catch fabrication errors like this short circuit.

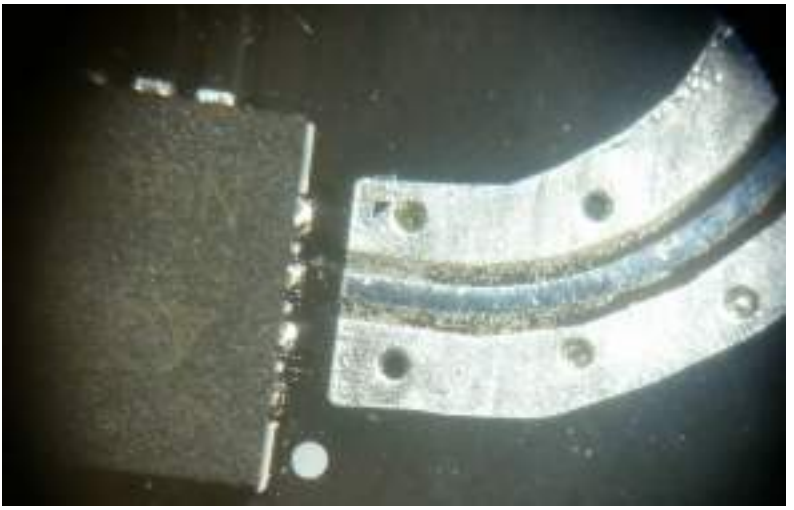

Figure 7.7: This is an example of an assembly defect. Small amounts of solder paste are sitting on a coplanar waveguide. This can easily cause a short (especially at high frequency) and is a result of poor PCB cleaning after reflow and possibly bad solder paste printing.

If you're assembling products en masse, you can do what's called a time study to figure out where your assembly inefficiencies are. Observe your assemblers as they work and identify the parts that are slowing them down, parts that can be automated, and other preparations that can be done prior to assembly that will speed things up. Make sure all of their equipment is sufficient for the job, is being used correctly, and is functioning properly. Properly trained assembly and repair technicians should know when they need new equipment and how to correctly perform rework. Having assemblers trained by an organization like IPC is a good way to make sure they know what they're doing, but lots of good experience is what's most important. If you're a startup or individual without the money for expensive training or certifications, you can sometimes find IPC documents online or on eBay. You won't be certified, but those documents will show you how to perform reliable rework. As you can probably tell, assembly is very hands on, even for people who aren't doing the actual assembly. Reliable assembly processes require attention to detail and clear, continuous communication with the assemblers.

To help eliminate static discharge problems during assembly, you

can use something called an alpha ionizer. Alpha ionizers attach to your compressed air source and ionize the air coming out using the small amount of radioactive polonium inside of them. The radioactive material itself doesn't come out of the end, and it's completely safe to use. Alpha ionizers are handy because you can simply spray your board with ionized air to remove any debris or static buildup. Using an air compressor *without* an alpha ionizer to blow debris off of your boards is actually bad, since compressed air is especially vulnerable to ESD problems. As the air rubs against insulating surfaces inside the compressor and against other air molecules when it exits the nozzle, it can build up a charge, which can collect on your PCB and cause ESD damage. Alpha ionizers prevent this by creating a roughly equal number of positively and negatively charged particles in the air that leaves the nozzle, which keeps charge from piling up and creating an ESD problem.

7.8 System Integration

System integration is harder than you think. After all of your individual boards are assembled, tested, and working, you may be under the impression that system integration is as simple as connecting them together. Even if you've managed to test all of your boards together, the task of stuffing those guts into an enclosure and having them all still work can be surprisingly difficult. Make sure you leave plenty of time in your schedule it. The trick to a smooth system integration is planning and incremental testing.

Planning starts all the way back in your requirements and specification documents, when you design the interfaces and connections between different parts of the final product. Don't forget to think about wire harnesses and cable management. You'll need to work closely with your mechanical engineer(s) to design how all of the parts come together, where they go in the enclosure, and how they're installed.

Once you have all of your parts ready for integration, make sure that they're tested individually first. Then gradually add subassemblies one by one, and finally test the entire system together. Working piecemeal will allow you to catch problems early and avoid having to take apart your entire system to figure out what's going wrong. It can also prevent you from accidentally damaging another part of your system if one subassembly fails in a catastrophic way. It's a good idea to test your subsystems on bench power supplies at first, so you can make sure your current consumption is what you expect and everything is working properly.

8

Testing

NASA's Jet Propulsion Lab has a saying: test like you fly and fly like you test. That means you should design your tests to mimic the actual environment where the device will be used, and you should only use your devices in a way that's consistent with how you tested them. If your device will be used in cold temperatures, test to those actual temperatures (plus a safety margin). If your device was never designed or tested for use in cold temperatures, then don't expect it to work under those conditions! But keep in mind that if you start to see a lot of units fail under conditions outside of what you tested, you may have made an incorrect assumption about your users, the environment that your product is used in, and how people are using it. These are all product questions, and changing product requirements is a big deal. But if your units are passing internal tests and failing in the field, consider that the problem may be that you are not testing your product the way it is actually used.

You need to prove that you can actually achieve all of your product requirements. To do this, testing is absolutely critical. But testing in and of itself isn't enough: testing must be complete and relevant. The first question that testing answers is "can it?". The second question it answers is "for how long?". Both parts of the puzzle are needed for the user to understand the reliability of the device. The answers to these questions matter to the manufacturer because they'll tell you how often to expect repairs to be necessary. They also tell you what the expected lifetime of the product is, which is important for customer support and end-of-life purposes.

All components have an expected lifetime. This lifetime is described by something called the bathtub curve, seen in figure 8.1. Here's how it works. At the beginning of the bathtub curve, parts with man-

ufacturing problems will fail early during acceptance testing. The goal is to design your tests so that all components with manufacturing defects fail before the product goes out the door. In the next part of the bathtub curve, parts continue to function for their normal lifetime. Finally, parts start to fail at the end of their expected life. Figuring out what this bathtub curve looks like for your product is important because it will drive your test plan and your warranty plan. If you don't test enough, you may miss some of the early failures, which will lead to devices failing early in the field. If you test for too long, you'll be wasting time. If your warranty covers failures well into the expected end of life of the parts, it's going to be really hard to make money, since you'll be replacing products that have reached the natural end of their life instead of only products that failed unexpectedly early.

Figure 8.1: The bathtub curve

8.1 Regulatory testing

8.1.1 Approval marks

In addition to the tests you design to validate your own product, there are tests used by the government or industry to independently ensure that your product was designed to be safe and reliable. If you look closely at almost all electronic devices (usually on the bottom), you'll find several logos that indicate which of these tests the device has passed.

The CE mark indicates that the device is "Conformité Européenne", or conforms to European safety and environmental standards. This mark is required if you want to sell your device in Europe. To get the CE mark, you need to identify the relevant directives, identify the harmonized standards, figure out which requirements apply to your prod-

uct, decide whether an independent conformity assessment is needed, actually run the tests, and keep correct documentation of all of this. While it's possible to do all of this yourself, you shouldn't. Talk to a testing lab. They are experts and will tell you what applies and what doesn't. There are lots of directives to look through, some of the most commonly required being RoHS (which limits hazardous substances you can put in a product) and WEEE (an electronics recycling directive). If you want to get a quick idea of what directive may apply to you, you can look up products similar to what you're building and see what directives applied to them. Searching "DOC" or "declaration of conformity" will usually return a short summary of requirements and tests it had to pass.

Another mark you'll commonly see is the UL mark from Underwriters Laboratories. Many people are surprised to learn that UL is a private company, and there is no legal requirement to get UL approval. So why do people pay for it? UL is one of the oldest testing labs in the US and performs consulting services to help you determine what regulatory testing is required, and will actually perform those tests. This includes safety testing, which is what they are most known for. While getting UL approval isn't legally required, it is often practically required since many retailers will not carry a product unless it has been safety tested, and UL is the most well-known safety tester.

8.1.2 Electromagnetic compatibility

Electronics must not only be safe for people to use, they must also be safe to use around other electronics. Electromagnetic compatibility is a general term that usually refers to electromagnetic emissions and electromagnetic immunity. Devices must not produce unintended radiation above a certain level because other victim devices may malfunction. The complete details of international electromagnetic emissions laws are out of the scope of this book, but make sure you know which laws apply to your product and how to comply with them.

All electronics devices that are sold as products must undergo regulatory emissions and immunity testing. In the United States, the Federal Communications Commission (FCC) specifies these tests and approves devices. Navigating the set of laws that govern all of this can be complicated, but it's also not your job! There are plenty of testing labs that can help you figure out which tests you need to pass. It's a good idea to reach out to several testing labs to get a quote. This ensures that you are getting a fair price, but it is also a free way to get multiple experts to judge what your testing requirements actually are. That way if one of them makes a mistake and has added or removed a

test that another lab said you needed, you can ask them about it and help ensure your product is being tested correctly.

You can schedule a meeting with UL to talk to an engineer who will answer any questions you have and tell you exactly which regulations are relevant to your product (not just electromagnetic compatibility rules, but other safety and environmental regulations too). It's best to start asking questions as early as possible. Some people think you need to wait until your product is ready to test to engage these people, but that is almost always too late. The situation you really want to avoid is to have a product that is 90% done, only to discover that your specifications for EMC were wrong and you have to do an expensive and time-consuming redesign late in the design cycle. Some EMC labs report that around 50% of tested products fail EMC requirements on the first try. This can be avoided if you engage with experts early. Talking to experts early can also help if you're trying to distribute your product in multiple countries. An EMC or UL consultant can help you navigate your unique set of regulation specifications.

Similarly, instead of waiting until your product is in production to run any of the actual EMC/EMI tests, you should take advantage of pre-scan tests that most labs offer. Pre-scan tests will save you money both because the tests themselves are less expensive than the final regulatory tests, and because catching EMC/EMI problems early in the design will be much cheaper to fix than problems caught during production. Pre-scan tests collect all the same data that the FCC requires for approval, it's just not submitted to the government. An example rate for a test like this is about $200 an hour, which includes full use of an antenna chamber and a technician's time to help you set up and run the tests. Once the tests represent your product in its shippable form, you can actually submit the data from your pre-scan to the FCC, along with a lot of paperwork, and use that as your approval data. To give you an example of the cost of getting full FCC approval, a company like UL will charge around $15,00-$20,000. You can end up paying more than this depending on the number of countries you need certification for and how complex your product is.

It's important to run EMC/EMI tests under realistic conditions, as the product will actually be used. Use your real enclosure and your real cables. Run all of your electronics in their worst case mode. For example, if you have a transmitter that has an adjustable power output, run it at the maximum power out. Turn everything on so the device is using the maximum current possible. The easiest way to do this is to write a separate version of your firmware for EMC testing.

When running immunity tests, you need a way to determine if your

device survived or not. The same firmware you use for EMC testing can also include code to exercise the functionality of each part of your device. For example, you can display a test pattern on any screens, blink LEDs, read and write to memory, check sensor outputs, etc.

Sometimes it can be useful to look up the FCC submission of another product. The easiest way to do this is to use the website http://fccid.io. Simply type in the name of the product, the company, or the FCC ID (found somewhere on the outside or perhaps in the battery compartment of the device) and the entire submission will come up. Companies are allowed to ask the FCC to withhold data about their testing until the product is actually released, so you may not be able to see everything immediately. Once the product is on the market though, it's required that all of that data be public.

8.1.3 Medical devices

One of the biggest regulatory burdens comes when trying to build a medical device. The amount of red tape you need to navigate to get a medical device on the market is daunting, as it should be. Medical device regulatory approval is out of the scope of this book. To determine what applies to you and what your design requirements need to be, talk to a field specific consultant. For example, UL offers a service where you sit down with several regulatory experts, describe your device, and determine exactly what you need to do to get it approved and on the shelf. The cost of this service is usually something like $600-$1000, depending on how long the meeting takes. And of course, UL will also perform those tests for you and submit the right data and documentation on your behalf.

8.1.4 Getting standards

If you need the full text of a standard, you will quickly find that buying it from the standards organization is extremely expensive. Luckily, there are several tricks for getting them cheaper. If you need a European standard, you can buy it much cheaper from the Estonian Center for Standardization.[1]. You can also Google for the name of the standard and add "filetype:pdf" to show only PDFs. This will often return drafts or slight variations of the standard you're interested in. You can usually download draft versions of standards for free once you sign up on the standards organization website, although they often remove drafts after a while, so you can't always find what you need this way.

[1] https://www.evs.ee/shop

Finally, you can search eBay for hard copies of standards, which are often significantly cheaper than buying a digital version from the standards organization.

8.2 PCB functionality tests

The first line of defense in preventing defective products from getting shipped to customers are the tests the factory does to determine your PCBs were manufactured correctly. Before assembly, they'll often perform a 100% electrical test using something like a flying probe. This is a machine that quickly moves probes to each of the pads on your board and runs continuity checks to make sure nothing is accidentally shorted or open circuit. After assembly, manufacturers usually put your board into an automated optical inspection machine (AOI) to do visual inspection of the components. This machine will automatically recognize defects like tombstones, misaligned parts, or missing components. If your design has BGA components, it will go through an automated X-Ray machine that will perform similar visual detection of solder shorts and other issues that aren't otherwise visible.

8.2.1 Board bring up

The process of testing a newly assembled PCB for the first time is often referred to as "bring up". You'll make sure there aren't any major mistakes on the board and exercise all of its functions. Start by visually inspecting the board, looking for parts that have been rotated, missing parts, missing solder, shorts, etc. Next, use a multimeter to check continuity between your power planes and ground. A dead short there can damage components[2].

If you designed your board as separable subcircuits like we talked about earlier, you can bring up one subcircuit at a time. This will make it much easier to identify where any potential problems lie, and can prevent a single misbehaving circuit from killing your entire board. So if you have any jumpers or zero ohm resistors or connectors that connect subcircuits together, remove them all and add them back one at a time as you verify each subcircuit.

Next, set up a current limit on your power supply. Give yourself a little room above the calculated current consumption and power it

[2]Note that some RF components may make it look like there's a DC short between power and ground during a continuity check, even if there's not. If you have a development board of the suspected RF component, you can perform the same continuity check and verify that it's expected.

on. That way if anything goes wrong, you'll minimize the damage since the power supply will go into constant current mode and start dropping voltage. If you see more current drawn than you're expecting, don't run the board for too long. If that extra current is being consumed because of a short or other problem and is turning into heat, you might damage a component. A thermal camera is a great way to figure out if any particular part is consuming more current than it should be. You usually only need to power the board on for a second or two to see which chip is getting way hotter than any others. Traditionally, thermal cameras have been really expensive, but you can now buy a thermal camera that plugs into your smartphone for a few hundred dollars. If you can't afford a thermal camera or can't get one in time, you can lay a layer of clear plastic wrap over your PCB and power it on. The hot chip will slightly melt the plastic wrap and you'll be able to instantly tell where the problem is.

After you've verified that the current consumption looks correct, nothing smells burned, and nothing is getting too hot, use a multimeter to check the voltage on each power plane and voltage regulator. The next step is to try programming any programmable ICs. You should also do a fit check of any enclosures or mechanical components. From here on, it's mostly a software exercise to test each component. Try reading and writing to any memory, talking over SPI, I2C, or any other buses, and capturing inputs and writing outputs.

These methods work for bringing up your first boards, but testing like this isn't scalable when you have hundreds or thousands of board to test. Once you've got all of the bugs in your design fixed and can reliably bring them up by hand, it's time to automate the process and run your test plan.

8.2.2 Test fixtures

The infrastructure that allows you to run your test plan is often fairly extensive. This includes the software and hardware needed to exercise 100% of the functionality of your product. This test infrastructure has been referred to as "the product that lets you test your product"[3]. Don't underestimate how much time and engineering it will take to design and build a high quality test jig and the accompanying firmware. A test jig usually performs tasks like programming ICs, running in-circuit tests (ICTs), boundary scanning (like JTAG), verifying memory (by writing and then reading each bit to check that the write worked), measuring voltages, currents, and even pressing buttons or checking LEDs.

[3]Paraphrased from Andrew "bunnie" Huang.

The most common way of testing a PCB in production is with a jig called a bed of nails. It looks kind of like a panini press, and the PCB sits inside of it while the lid is shut. Special pins called pogo pins are used to make contact with test points. Pogo pins come in several varieties, but they all have spring loaded tips that can press into the test points when the bed of nails jig is closed. You can find pogo pins with a spear head, crown head, needle head, or cupped head. The company Merifix sells a lot of parts and kits that are useful for building a bed of nails fixture. Other common parts you may find useful for fixture building are fixture clips, pogo pin probes, and toggle clamps[4].

Just as important as the tests themselves is the human side of testing. As you're designing your test jigs and writing your test plans (which is covered in the next section), think about how the people executing the tests will run them. Your test jigs need to be reliable, updatable, intuitive, and require as little human interaction as possible. The details of how to come up with a good user experience (UX) are out of the scope of this book (it's a whole other field!), but as an example, consider symbols instead of text in your interface. Different people from different parts of the world may be using your jig, depending on where you're manufacturing. An ideal test jig only requires two human interactions: putting the board into the jig and pressing "start", and removing the board when the test is done. Automation should be as complete as possible.

Once you have a great testing plan and testing jig, you need to track which boards have been tested, their status, and you need to verify that all the tests are actually being run correctly. Less scrupulous factories may sometimes try to cheat by passing devices that weren't tested or that failed in order to artificially boost production yield. This is relatively rare, but mistakes or poor record keeping can also result in devices getting shipped that shouldn't have been. Andrew "bunnie" Huang suggests only generating a serial number after all tests pass, as a last step. This way the serial number itself acts as a unique proof that the device can be shipped.[5] You can also use a spreadsheet to keep track of individual units or prototypes and their status. For example, you can track the ID/serial number, status, location, and problem history. This works best for prototypes or a small number of units, and can be used for subassemblies or entire assemblies.

There are also companies that will do test tracking and logging

[4]Links to suppliers for all of these parts can be found on this book's website, DesigningElectronics.com.

[5]Read his excellent article on factory testing here:
https://www.bunniestudios.com/blog/?p=5450

for you. Assembly.com offers a service that will monitor production lines in China and give you a picture of every device off the line with proof that it was tested according to your test plan and that it passed. Assembly also offers several other services that are really valuable for manufacturing in China, especially if you've never done it before.

The ability to debug and update your test jig during production is very important. Inevitably, something will go wrong or break and you'll need to figure out why and how to fix it. Time is very expensive when production is halted, so having your own complete duplicate test jig specifically for development and debugging is really important, especially if you're not in the same country as the production line.

8.3 Designing Tests

Designing good tests is difficult. The good news is that you don't have to start from scratch. There are a myriad of test specifications out there. IEC, FDA, JEDEC, NIST, and other standards organizations lay out the exact test procedures for testing many types of products. Some of those standards include:

- MIL-STD-810
- MIL-HDBK-310
- SAE J1211
- IPC-SM-785
- Telcordia GR3108
- IEC 60721-3

Even if your product doesn't fit into exactly one box, you can pick and choose tests from different standards to start out with. The goal is to completely test your device, end to end, in relevant, accurate conditions. While you technically only need to test up the limits that you guarantee to the customer, it's a good idea to test beyond those limits to points where the product may be misused. Customers will judge how well built a product is based on how realistic the test specifications are and how far past the guaranteed operating points that the product will survive. The Mars rovers routinely survive years past their guaranteed or goal lifetime and are seen as some of the best engineered devices in the world. On the other hand, if your test conditions only cover part of what you know the device will likely experience, your product will be perceived as shoddy. For example, imagine you design a Bluetooth

earpiece and specify that it survive a drop test of two feet. Even if your test passes and it can survive that drop distance, it's not a realistic test condition. People are going to drop the earpiece from head height as they're putting it on and taking it off, so even though you can correctly say that the device passes your drop test, consumers will still consider it poorly made because you didn't design and test to a realistic specification.

How do you decide which tests to choose for your product? How do you design new tests? The purpose of testing is to ensure that your product meets the specifications you designed it to meet. In some cases, those specifications are the same or very close to the same specifications that other products use. For example, it's unlikely that you're designing your device to work from -200 C to + 200 C. A more common temperature range is something like -20 C to +85 C, which many specifications use. Because of that, there are proven test procedures for these specifications. Even if existing standards don't have test procedures for the exact specification you're using, you can still extend existing test procedures. But it's important to not rely blindly on existing test procedures. You need to consider the real circumstances under which your product will be used. Think about how things can fail given the environment and manner in which your product will be used. Consider how reliable your users will expect it to be. Is correct operation a matter of life and death? What will the user be satisfied with? What will they be disappointed with? You can't wait until you're designing your test procedures to make these decisions. You need to think about roughly what your test plan will look like when you're writing your product requirements and specifications. Once you've answered these questions, then you can start designing detailed test procedures that will ensure you actually meet those requirements.

A well designed test has a clearly defined success criteria and failure criteria. Each test will start with the materials and equipment needed to run the test and describe how to set them up. Next, the test itself is described, including how to run the device under test, what to record, and how long to run for. Each test should directly verify at least one requirement. The test should have a defined ending, at which point it can be immediately determined whether or not the test was passed. The passing criteria should be data driven, not subjective. Anyone should be able to run the same test and get the same result.

While not possible in all designs, it can be valuable to test devices to their failure point. This will not only inform future design iterations, it will also help you understand what the signs and symptoms of your product failures are. When you get a device sent back for repair or need

to understand why devices are failing in the field, having a complete set of point of failure test results that have been analyzed to their root cause will be enormously helpful in identifying the problem, fixing it, and preventing it from happening again.

Make sure the final version of the product you ship out the door has been fully tested. Last minute changes and tweaks are all too common and it's easy to let those small changes make it into the final product without being fully tested because of time or budget constraints. Ideally you can perform complete end to end testing with the version that ships, but this is often not practical. Instead, use your best judgment to identify what new risks there are as a result of these last minute changes. If you remove an extra screw on the enclosure, you probably don't need to redo any electrical testing, but you will likely want to redo vibration, drop, or ingress protection tests.

One measurement that may come up during reliability testing is Mean Time Between Failure (MTBF) or Mean Time To Failure (MTTF). As the names imply, this is the average time that it takes for a device to fail. Getting an accurate value for MTBF and MTTF can take a long time because you need to run a lot of tests to get enough data for a statistically significant result. MTBF can give you valuable insight into how many warranty claims you can expect to get and when you will start seeing them.

Don't forget that your product needs to survive not only in its normal operating environment, but also its shipping environment. Your product packaging and the shipping logistics will play heavily into what the shipping environment requires in additional product requirements and testing. When most people think of intense shipping environments, they think of military or aerospace applications. For example, some military supplies are dropped in by parachute and need to survive the impact. Equipment that is used on the space station or on a satellite needs to survive a high vibration environment on the rocket ride up. But the much more common cause of product failures due to the shipping environment is air freight.

A lot of cargo that rides on planes isn't in a temperature or pressure controlled area. This means that it can experience rapid changes in temperature (which can cause moisture to condense on the device), rapid changes in pressure (which can *also* cause moisture to condense), and vibration, all at the same time. Cargo planes are typically pressurized to 8,000-10,000 feet and will get to this pressure at about 1,000 feet a minute after takeoff. The temperature inside the cargo bay can vary a lot, but it's typically between 49F and 79F. If you will be shipping your product via air cargo, these are requirements you may want

to design for. Surviving the shipping process can also be a task that you offload to the packaging of the product.

Here's a list of the most common tests used to qualify a design and the parameters you'll need to choose [6]:

- Temperature - maximum, minimum, and ramp/dwell times
- Humidity - condensing or noncondensing
- Corrosion - salt, hydrogen sulfide, chlorine, nitrogen dioxide, etc.
- Atmosphere - low pressure, high pressure
- Power cycling - number of cycles
- Supply voltage - maximum, minimum
- Supply current - acceptable range, device modes
- EMI/EMC
- ESD - voltage
- Mechanical strain - static, cycling, torsion
- Mechanical shock - G's, waveform, number of events
- Drop test - height, surface to drop on
- Random vibration - PSD, time, kurtosis
- Harmonic vibration - G's, frequency
- Ingress protection - dust, water, IP level

These are all *potential* tests you can run. As discussed before, choose which tests you need based on how and where you product is actually used. If you're designing an MP3 player, you probably don't need to do cycles in a hydrogen sulfide atmosphere. Then again, maybe you're building a radio for use in mines and hydrogen sulfide might come in contact with your device. The best way to decide what parameters to test your product to is to think about the environmental and stress conditions that the device will undergo. You can calculate these conditions, measure them, or run tests to figure out what the conditions will be. Think about the average conditions as well as the variability of those conditions and the worst cases.

[6] Tulkoff, C. (2021). Test Plan Development using Physics of Failure: The DfR Solutions Approach. Retrieved from https://www.dfrsolutions.com/hubfs/Resources/services/Test-Plan-Development-using-PoF.pdf

Another thing you'll need to decide is the order in which these tests occur and if any of them need to be performed simultaneously. For example, if your device is going to be on a crab fishing boat in the North Atlantic, a temperature test combined with a salt corrosion test might be a good idea, since any salt fog on the outside of the device may freeze. If your device will be touching something with a large temperature differential, that can cause condensation to form on your device and possible puddle around it. A great resource on combined environmental testing is NASA Lessons Learned #643.[7]

Highly Accelerated Life Testing (HALT), or sometimes called Accelerated Stress Testing (AST) is a way to simulate the aging of a device to determine when and how it will fail. The way this is done is by stressing the device to higher than its normally rated specifications, with the assumption that stressing it in this way will cause the same failures in the same way as a device that is run at its nominal specifications for a much longer time. No one has time to sit around and run a device for 5,000 hours. Instead, the temperature and voltages on a device are both increased and the device is run for something like 48 hours. It may seem unbelievable that doing this will cause the same failures as running the device normally for a very long time, but the research seems to show that it is mostly the case. There are always exceptions, but HALT is generally considered by the industry to be a valid method of testing the life span of your device without actually waiting for the entire life span.

ESD testing is typically performed on one of three models: the human body model, the machine model, or the charged device model. The human body model represents the ESD caused by charged up humans touching your device. The machine model represents ESD caused by conveyors or other moving machinery parts. The charged device model represents what happens your device gets charged up and is then put in contact with a material of a much lower voltage. Some people are very concerned about touching a part and killing it with ESD. At times, I've been quite cavalier in not using ESD protection when soldering or handling components and I have *never* killed a part due to ESD. I always wondered about this, and it turns out there was a study done that showed only 0.10% of documented ESD damage was caused by a human discharging into a component. 99.9% of the cases had damage due to the charged device model[8]. This doesn't mean you should not use any ESD protection when handling parts, and this study

[7] https://llis.nasa.gov/lesson/643

[8] Pierce, Roger J., "The Most Common Causes of ESD Damage". Evaluation Engineering, November 2002.

doesn't speak for every environment, but what you should take away is that one of the biggest risks of incurring ESD damage is actually from your components getting charged up and then discharging on another object. Make sure all of your lab equipment is well grounded, including your soldering iron and your work surface. ESD mats are one way to make sure your work surface is ESD safe.

Humidity testing can be broken up into two primary types of tests: condensing and non-condensing. A condensing humidity test means that the device under test will be at a lower temperature than the dew point inside the test chamber. Once this happens, the moisture in the air will condense into droplets on the device. It's possible that your device under test will rarely or never have condensation on it because it will never get cooler than the dew point of its environment. This can happen if your device gets hot after its powered on. In that case, condensing humidity tests may not be necessary. But before you write your device off as immune to condensation, think about the entire temperature range your device will be subjected to and whether turning the device on and off with the right timing could cause it to be at the right temperature to cause condensation, even if only for a few minutes. You should also think about whether or not condensation can drip onto your device. Maybe your device runs warm enough that it will always be above the dew point, but a cable or antenna above it doesn't get warm and allow condensation to form.

What you're looking for in a humidity test is signs of water ingress into areas where it shouldn't be (which can cause shorts), and electrochemical migration. Copper and other metals can form dendrites that can cause shorts. Dendrites form when metal ions get dissolved into water (like condensation) and start re-depositing as crystals. These crystals grow between areas with a potential difference. When they touch, they begin conducting and cause a short. This ion migration happens very quickly in the presence of water, and that's why it's important to run humidity tests to try to identify and prevent it. High voltages can also catalyze this reaction and cause faster than normal growth. There are some subtleties to the different mechanisms and types of electrochemical migration, but you may also hear these kinds of effects referred to as tin whiskers or tarnish creeping.

There are several different standards that describe humidity tests. An example MIL-STD temperature-humidity test curve can be seen in figure 8.2 and a summary of several other standards is in table 8.2.

Accelerated Stress Test	Test Standard	Standard Life	Long Life	Automotive
Autoclave	JESD22-A102	Optional test 48 hours	Optional test 96 hours	96 hours
Data Retention Bake	JESD22-A117	504 hours at 150C	1008 hours at 150C	1008 hours at 150C
Electrostatic Discharge Charged Device Model	JESD22-C101	+/- 500V (or classification)	+/- 500V (or classification)	+/- 750V for corner pins, +/- 500V for all other pins (or classification)
Electrostatic Discharge Human Body Model	ANSI/ESDA/JEDEC JS-001-2012	+/- 2000V (or classification)	+/- 2000V (or classification)	+/- 2000V (or classification)
Electrostatic Discharge Machine Model	JESD22-A115	Optional test +/- 200V (or classification)	Optional test +/- 200V (or classification)	+/- 200V (or classification)
Highly Accelerated Stress Testing	JESD22-A110	48 hours at 130C 132 hours at 110C	96 hours at 130C 264 hours at 110C	96 hours at 130C 264 hours at 110C
High Temperature Operating Life Test	JESD22-A108	5 year equivalent use time	10 year equivalent use time	1008 hours at 125C or 406 hours at 150C
High Temperature Storage Life	JESD22-A103	Optional test 504 hours at 150C or 240 hours at 175C	Optional test 1008 hours at 150C or 504 hours at 175C	1008 hours at 150C or 504 hours at 175C
Latch up	JESD78	+/- 100 mA (or classification)	+/- 100 mA (or classification)	+/- 100 mA
Temperature Cycling	JESD22-A104	200 cycles from -65C to 150C, or equivalent per JESD94	500 cycles from -65C to 150C or equivalent per JESD94	500 cycles from -65C to 150C or 1000 cycles from -50C to 150C

Table 8.1: Common accelerated life tests

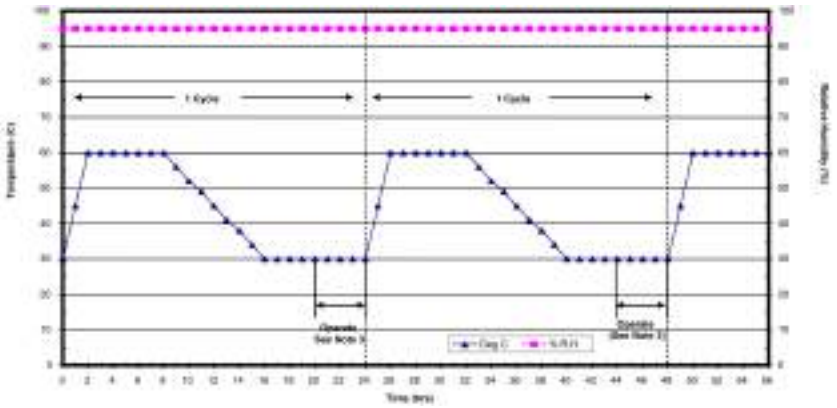

Figure 8.2: Example humidity testing curve from MIL-STD-810g. See standard for details and additional test curves.

Standard	Temperature	Humidity	Duration
IEC 60068-2-3	40 C	93%	500 hours
IEC 61189-5	40 C	93%	72 hours
IPC TM-650, 2.6.3.5	35 C	88%	96 hours
IPC TM-650, 2.6.14.1	40 C	93%	500 hours
IPC TM-650, 2.6.14.1 (alt)	65 C	88%	500 hours
IPC TM-650, 2.6.14.1 (alt)	85 C	85%	500 hours
IPC 476A	65 C	85%	500 hours
GR-78-CORE	35 C	85%	96 hours
GR-78-CORE (alt)	65 C	85%	500 hours
ISO 9455-17	40 C	93%	168 hours

Table 8.2: Humidity testing standards

Conformal coatings on their own are not necessarily enough to protect against condensation forming on a PCB. Just because you're using a conformal coating doesn't mean you get to skip humidity testing. Failures in humidity testing will often happen intermittently, so you need to monitor your test continuously. If you simply test functionality after the test is completed, it may pass and you'll miss the temporary failures that happened during the test.

It's recommended that you run at least 3 test articles through humidity testing, place them in different orientations, and keep the product continuously powered for the entire duration of the test.

A related test that a lot of consumer products advertise nowadays

is IP certification. Here, IP stands for Ingress Protection. An IP rating has two numbers: the first represents the degree of protection against solid particles and the second represents the degree of protection against liquid. Table 8.3 below describes these ratings in detail. The full test standard is IEC 60529.

8.4 Avoiding Testing Problems

You can't expect to be able to successfully test or debug anything unless you're sure of your equipment and its use. This is the problem with cheap Chinese test equipment: you can't trust it. Buy or borrow high quality equipment as much as you can. If you can only get your hands on cheap tools, you should at least try to understand their error. If you have two multimeters, you can check them against each other. If you have a suspicious bench power supply (most labs I've seen have at least one of these lying around), measure the output on a scope to see how bad the ripple is and how accurate the output is.

Make sure that none of your equipment has silently failed. Things like probes, cables, and test equipment inputs can easily be damaged without you realizing it. This is especially true if the equipment is being shared by multiple people. Cables will often not fail outright, but degrade, leading to frustrating hours of debugging or testing only to discover that the problem was a bad cable. If you have any evidence that a cable is damaged, throw it away. Do not put it back on the rack for someone else to repeat your ordeal! Occasionally, test the S21 of your RF test cables and make sure that the loss is what you expect. Moving RF test cables around can weaken them, and in some cases the coax outer conductor braid can shear off from the connector, rendering the cable totally useless. If an RF cable is acting funny, either label it with the S21 or throw it away. RF adapters can also cause problems with measurements, so make sure to check them occasionally too.

When testing amplifiers, the outputs need to be terminated into a load. Usually this is 50 ohms. If the amplitude of your output is N watts, you need at least an N watt terminator. Even though terminators are just resistors, keeping a 10 watt signal on a 2 watt resistor for even a few seconds is enough to blow up the terminator. It will probably fail as an open, which will create a huge mismatch, huge VSWR, and very possibly blow the amplifier. It's a good idea to measure your terminators every once in a while, just in case someone has accidentally killed one and didn't realize it. Even if a bad terminator doesn't blow up your amplifier, it can still throw off all of your measurements.

Make sure you know your equipment. Not only will this help you get

Solid particle ingress number	Degree of protection	Liquid ingress number	Degree of protection
0	None	0	None
1	Any large surface of the body, such as the back of a hand, but no protection against deliberate contact with a body part	1	Vertically drip water equivalent to 1 mm rainfall per minute for 10 minutes
2	Fingers or similar objects	2	Tilt DUT 15 degrees and vertically drip water equivalent to 3 mm rainfall per minute for 10 minutes
3	Tools, thick wires, etc.	3	Spray water at an angle up to 60 degrees at 0.7 liters per minute for 5 minutes
4	Most wires, slender screws, large ants etc.	4	Splash water on the DUT at 10 liters per minute for 5 minutes
5	Ingress of dust is not entirely prevented, but it must not enter in sufficient quantity to interfere with the satisfactory operation of the equipment.	5	Use a 6.3 mm nozzle at a distance of 3 m from any direction to spray 12.5 liters per minute for 3 minutes
6	No ingress of dust; complete protection against contact (dust-tight). A vacuum must be applied. Test duration of up to eight hours based on airflow.	6	Use a 12.5 mm nozzle at a distance of 3 m from any direction to spray 100 liters per minute 3 minutes
		7	Immerse DUT in 1m of water for 30 minutes
		8	Immerse DUT in water depth specified by manufacturer for time specified by manufacturer

Table 8.3: Descriptions of tests for different IP ratings.

the most out of its features, it will prevent you from making mistakes that will aggravate your testing. Pay attention to the warnings about the maximum input conditions. Make sure the tool you're using is actually capable of measuring the thing you're interested in. Check the settings. If someone was using it before you, who knows what they may have changed.

Make sure the test equipment you are using for your test is calibrated. For most equipment, this just means it has been serviced and calibrated by the manufacturer recently. For a VNA, this means performing calibration before you begin the test. VNA calibration needs to be repeated every so often, possibly multiple times per day, depending on the changes in temperature in the room and the manufacturer recommendations.

Make sure that your test setup is actually capable of accurately measuring the quantity you're interested in. Are connectors and cables rated for the frequency you're using? Is your probe and its connection to your circuit going to introduce signal integrity problems? Some people mistakenly treat test equipment (especially expensive equipment) as a magical device that will always just show you the signal you want to see in perfect clarity. But the faster or smaller the signal of interest, the more thought you need to put into how you're going to correctly probe it.

9

Building a Lab

Effective testing and development requires a lab. Before you start buying equipment, you need to decide what capabilities you need. To do that, you need to make a decision about the tradeoff you want to have of time vs. money. The more expensive equipment you get, the faster you can prototype and debug things (to a point). While a lab with all of the latest and greatest equipment may end up costing hundreds of thousands of dollars, you can build a lab that will get you 80% of the performance for much, much less. Even if you're doing RF work that traditionally requires expensive equipment, there are ways you can get by while spending a fraction of what a well funded lab would spend.

All of the equipment in this section has links on this book's companion website, DesigningElectronics.com. The links are kept up to date, so you'll know you're getting the right thing at the best price I know of. It's also updated with any new, better, or cheaper versions of the tools I use in my lab every day. None of this equipment is listed here because the manufacturer paid me: this is all equipment I use personally in the lean, fast paced lab where I work. I found these tools usually after trying several alternatives and settling on what's listed here.

9.1 Lab furniture

A lab needs excellent lighting. Get ring lights for your microscopes and very bright overhead lights for the lab bench. When you're dealing with parts that are hundredths of an inch long, you don't want any shadows. You also want the light to be as white as possible to make discolorations of parts easier to see.

Get a lab bench that has shelves above it for test equipment. You can build a lab bench just using Ikea furniture (I like the Gerton table as a starting point), or you can buy one purpose-built. Most test equipment is very long and deep, so putting it directly on your work surface won't leave you very much room for doing anything. A shelf keeps your primary work surface clear and makes it easy to drop down probes from the test equipment to the device under test. This can really improve cable management, since instead of cables sprawled out all over the table, they drop straight down from above and don't get tangled up. You also want to get as deep of a shelf as you can so you can push your test equipment back to a comfortable distance and not have them hanging halfway off the edge.

The work surface itself needs to be ESD safe and heat safe. You can buy ESD soldering mats made of a high temperature resistant material. ESD mats usually have a connector on them for attaching an ESD wrist band. For this wrist band to be effective, you need to ground the ESD mat to earth ground. Other ways to manage static include anti-static floor mats and even anti-static carpet.

Simple ergonomics are important. Get a good chair and make sure it's at the right height. Especially if you're working under a microscope a lot, you need to make sure you're sitting up straight and not slouching. You may need to put some books under your microscope if it doesn't adjust high enough.

9.2 Soldering equipment

Your lab will need at least one microscope per soldering station. The term to search for when shopping for one is "inspection microscope". These are designed to have a much longer focal length than a normal microscope and a lower magnification. The objectives shouldn't be more than 10x and a zoom level beyond about 5x isn't very useful. You also need your microscope to be binocular (two eye pieces). Binocular vision gives you a sense of depth, which is immensely helpful when trying to soldering anything under magnification. The cheapest option for an electronics lab microscope is about $200 from Amazon. It has a built in light, which works well, and the magnification range is good. A step up from that is an articulated arm microscope. Amscope is a popular brand. These have better optics, larger zoom ranges, and give you more room to work. They also have really handy mounts that let you easily swing or slide the microscope out of the way when you're not using it. This sounds like a minor convenience, but it's actually extremely useful and lets you get more use out of your bench space. The

next step up from a binocular articulated arm microscope is something like a Mantis microscope. These use special optics that let you look into something more like a screen than two eye pieces. If you're going to spend all day, every day under a microscope, it's more comfortable on your eyes. Some models also have the ability to look at different angles while the PCB is sitting flat. So you can have a board sitting on your desk and turn a knob to be able to see sideways into the edge of a chip (to inspect the BGA balls, for example). These microscopes are much more expensive than an Amscope, however. A Mantis goes for about $1400, where a Amscope is about $600.

Make sure you adjust the focus on each objective independently before you adjust the height of the microscope, so each eye is in focus. You may not notice at first if one eye is slightly out of focus, but it can give you a headache after a day of working.

A fume extractor or fan is good for keeping solder smoke away from your face. A common misconception is that the smoke given off while soldering has lead in it, but this isn't true. Soldering irons can't reach high enough temperatures to vaporize lead. Instead, the smoke is almost all flux. It's not poisonous, but like any smoke, it's not great to breathe it in all the time. It's also just annoying to have smoke in your eyes and face. A simple fan works well, but high end systems use a vacuum with a flexible tube that you can position next to your work piece to suck the fumes away and avoids blowing fumes into your neighbor.

Every lab should have a vise to hold PCBs and other small work pieces. Panavise is usually the brand that most people go with, since their products are specifically designed to hold a PCB. You can also use a "third hand". The most commonly available third hand has two alligator clips attached to flexible arms and a magnifying glass attached to a third flexible arm. You can use the alligator clips to hold and position wires and PCBs while you work on them. This is really useful for positing things in weird ways that a normal vise won't allow. Unfortunately, most third hands cost about $5 and are really terrible. The base is too small, so it tips over easily, the alligator clips break off of the arms easily, and the magnifying glass is basically useless.

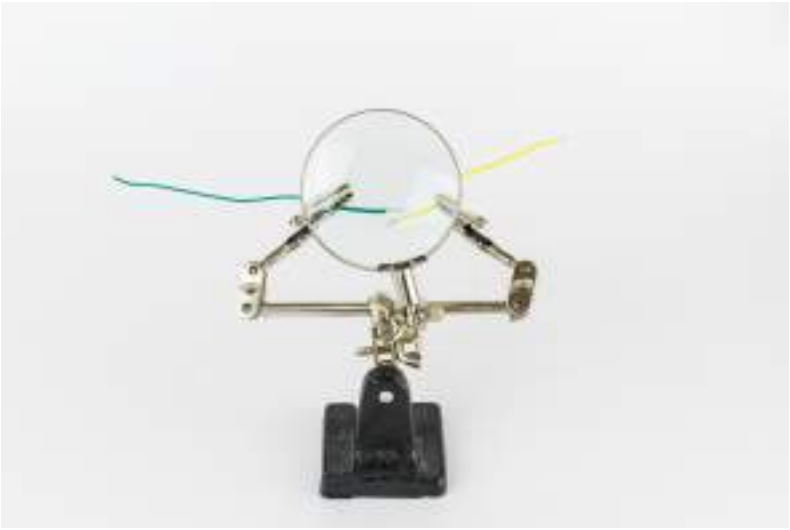

Figure 9.1: Terrible third hand

Luckily, you have a few options for a better third hand. You can find 3D printable designs on Thingiverse[1] that have as many as eight arms, including some with built in mini-grabbers for easy probing. One design in particular that I like is called PCB Workstation with Articulated Arms by giufini. You can print his models for free, or pay him for the already printed parts.

Another option for a third hand is building one using flexible segmented coolant hose. There are several tutorials available online on how to do this, but once you get the gist of how the construction works, it's easy to customize it for your own purposes.

[1] http://thingiverse.com

Figure 9.2: A section of flexible coolant hose

If you're applying solder paste with a syringe, I highly recommend using a pneumatic solder paste dispenser. It's much easier on your hand and thumb and allows you to apply paste more precisely. To do this, you'll need a dispenser regulator. These are often marketed as solder paste dispensers, glue dispensers, or liquid dispensers. It's all the same thing, and has a pressure gauge, regulator knob, an air hose input on the back, and at least one air hose output on the front. You'll need an air compressor to run it, and possibly some hose fittings. Try to get a quiet air compressor (I like the Ultra Quiet series from California Air Tools), or at least keep the compressor in a sound proof closet somewhere rather than the lab. Typically you'll set the air compressor pressure to 100 PSI or so and run that to the back of the solder paste dispenser. You'll remove the plunger from your solder paste syringe and clip on a sealed connection to the output air hose from the dispenser. Make sure you have the right syringe size for the connector to your regulator. Typical sizes are 5cc and 10cc. Then use the regulator knob on the dispenser to adjust the speed at which paste comes out of your syringe. Many dispensers come with a foot pedal to control when paste comes out. Some dispensers also have DIP switches on the back to change the mode of operation from continuous flow (pressure is applied to the syringe as long as the foot pedal is depressed) to discrete flow (pressure is applied for a fixed time interval once the foot pedal is depressed). You'll probably want to use continuous flow. A list of recommended dispensers, parts, and fittings can be found on this book's website, DesigningElectronics.com.

9.2.1 Choosing a Soldering Iron

Buying a soldering iron is like buying a mattress or tires. It's worth it to spend more money because you're going to be using it constantly. If you can spend the money, Metcal and JBC are the two best manufacturers. They work a little differently than other soldering irons. The tip itself has the heating element in it. This means that the temperature regulation at the tip is much better than a normal iron, but it also means that new tips are more expensive. Both Metcal and JBC use induction heating, so you don't need to wait for the tips to get hot. They reach their set temperatures within seconds of turning on. Metcal irons don't have a temperature control on the soldering station, but use the tip itself to control temperature. You order particular tips that reach particular temperatures. Both Metcal and JBC offer tweezer soldering irons as well. These are great for quickly adding or removing SMT capacitors and inductors. Metcal and JBC irons can cost around $1000 new. If you can't afford that, you can usually pick up used stations on eBay for much less.

The tier below Metcal and JBC is Weller and Hakko. Many websites and YouTube videos will recommend a Weller or Hakko as a first soldering iron. These irons are fine and work well. Temperature control isn't quite as good, and you have to wait for them to heat up, and they use traditional tips that are very cheap. People have gone their entire careers soldering on only a Weller iron. Hakko has really upped their game in the last several years and offers quality irons that some say approach Metcal or JBC in quality. Don't let the Fisher-Price aesthetics fool you. Hakko also offers irons that use induction heating.

The lowest tier of soldering iron is the "Radio Shack Special". These are irons that plug directly into the wall, have no temperature control, and really don't work well. You can often find an old, low end Metcal iron on eBay for only a little more than these irons, and you will be much happier with a Metcal.

Hot air irons are a different market than regular soldering irons. The high end soldering iron companies also make hot air rework stations, but the majority of the hot air stations you will find online are Chinese. There are maybe 3 or 4 different designs and probably 20 companies that make them. They often look exactly the same, but have a different name on them. The good news is that these stations are fairly cheap, but they can also be hit or miss in terms of reliability. I have had stations that have lasted many years, and others that lasted only months. I've had most success with the Hakko and Quick brands. Metcal makes an expensive one that I bought in an attempt to have one would actually last, but it's very complicated to use, and I don't

recommend it. See DesigningElectronics.com for an up-to-date list of hot air irons and where to buy them. Like regular soldering irons, hot air irons have different tips available. I get the most use out of small, circular nozzles.

If you're going to be doing a lot of hot air work, a board preheater can be really helpful. A board preheater is a device that sits under your PCB and heats up the entire board, but not enough to melt the solder. If your PCB is large or has a large thermal mass, it can be difficult to get a hot air iron to heat up a small area enough for the solder to melt. A preheater raises the temperature of the entire board enough to make this easier.

9.3 Organization

Small petri dishes are excellent for holding screws and parts that you're taking off or putting onto an assembly. I like them because I can put some parts in a dish, label the dish with a sharpie on the lid, and then stack them near my microscope. When I need a part or am going back to work on something, it's really easy to find. Try to get glass petri dishes if possible, because it's easy to accidentally melt plastic dishes. Get half size dishes instead of the normal full size.

Large, gallon sized anti-static bags are great for holding old project parts. When I'm done with a project (or if I'm putting it on ice for a while), I put all the parts, PCBs, and notes in a gallon sized anti-static bag, put a piece of tape with the name of the project on it on the outside, and stick it in a filing cabinet. Small anti-static plastic bags are useful for storing individually populated PCBs. You can buy all sizes of anti-static bags very cheaply, but I can usually keep my small sized stock full by just saving extra bags that come when I order parts.

As you accumulate parts, you're going to need a place to store them and index them. The solution most people use is component drawers. These work pretty well, but can take up a lot of space if you have a lot of unique parts. Get component drawers that are clear so you can see what's inside them. They should also click shut, if possible, so you know they're closed and they stay closed on their own. It's also nice if the back of the drawer is completely enclosed, so if you flip the draws upside down or lay them flat, parts won't spill between drawers. You will appreciate this if you ever have to move your component drawers between buildings. Another way to organize parts is in reels. This works best for components that you have hundreds or thousands of, but you can also buy cut tape mounted in reels with smaller quantities. Reels can be easier to store, since you can just use a bookshelf or

something similar. An alternative to parts drawers is part notebooks. You can buy 3 ring binders with plastic sleeves in them that fit cut tape sizes. For loose SMT parts, a company called AideTek makes some really cool folding boxes with tons of little snapping compartments. You can find links to all of these things on DesigningElectronics.com

Some people will be very insistent that everything you use to store a component be anti-static. Static typically isn't a problem because there isn't enough capacitance between the component leads and another object with a significantly lower potential (aka, ground). This may not be the case for parts on a PCB though, because they will have a significantly higher capacitance to ground. If you're worried about your components, or want to play it safe, just keep components in an anti-static bag if your component drawers or other storage system is not anti-static.

Figure 9.3: Binder for holding cut tape

If you use a binder for holding cut tape like the one in figure 9.3, you will quickly run into the problem of needing to insert a new piece of cut tape into the middle of a page that's already full. To make moving the cut tape easier, don't use stickers to label each entry. In figure 9.3, each row of components has a label right above it. These labels were made by cutting the Digi-Key label that they came with into a rectangle that only contains the Digi-Key part number and the part value and description. These two rows of text happen to be right above one another, so you can fold the cut out label in half long-ways (without removing the sticker backing) and slide it in the sleeve. That

way the component values are printed right above the parts, and the back of the sleeve has the exact Digi-Key part number, in case you want to order more. Making your own binders this way is vastly superior to having a box somewhere filled with bags of parts that you have to dig through when you want to find a particular part. It really doesn't take that long to fill up a couple of these binders and it will save you time, space, and money.

If you don't want to make your own parts books, or if you have too many gaps in your parts collection, you can buy pre-made books of resistors, capacitors, and inductors. Sometimes companies will even give you these for free. I like to have books with 0402 and 0603 sizes of each. I also like to keep a book of 0402 high Q inductors and capacitors for RF work.

If you want to be really organized, there is open-source inventory management software you can use to keep track of where parts are and how many you have. It can be really difficult to keep inventories updated, especially when you're hacking around on something or building a quick prototype. However, for large scale operations (like a company), at some point inventory management becomes necessary.

Electronic components aren't the only things that come in pre-organized sets. It's really nice to have a couple fastener sets with imperial and metric nuts, bolts, washers, and spacers. Having both metal and Nylon fasteners available will give you even more flexibility. Heat shrink tubing of different diameters is also available in small sets and is equally useful.

Electronics assembly often involves the use of solvents like acetone, isopropyl alcohol, and other very flammable substances. Get a flammables cabinet. They're not very expensive and they go a long way in keeping you safe. Of course, for them to be effective, they have to actually be used. Small amounts of isopropyl in a dispenser on the lab bench is fine, but keep the gallon refill jug in the flammables cabinet. If you are working in a commercial space, it may be against the law or a code violation if you don't.

9.4 Test equipment

The biggest test equipment companies are Keysight (formerly known as Agilent, formerly known as HP), Rohde and Schwarz, and Tektronix. They've been around a long time, make high quality equipment, but are also the most expensive. They are far from the *only* test equipment companies though. There are plenty of others of varying quality that specialize in different things. Other names you can look up are Keithley,

Fluke, BK Precision, GW Instek, Siglent, Rigol, and Anritsu. I'll call out certain models that I like in this section, but remember that all of these companies make competing equipment, and it's worth it shop around between them. If you work for a company or startup, contact the sales engineers at these companies and they'll usually let you try out the equipment for a week or two before you buy it.

Test equipment is notorious for being very expensive. To save money, you can ask for quotes on used or demo equipment. This is usually equipment that has been loaned to different companies to try out or to borrow while the units they bought were being repaired or calibrated. There is often limited demo equipment available, so you may not be able to get the exact models you want. Another way to save money is to buy the previous model. New versions of test equipment don't come out every year, but when the transition happens, you can sometimes get the previous version for less than the new version. Other times, the manufacturer will purposely price the new version slightly lower than the old version, to incentivize people to buy the new version. Finally, you can try negotiating the price itself. This works best if you have multiple companies fighting for your business. A nice technique to use is to make sure one company visits your lab and sees their competitors equipment being evaluated on your bench. Sales engineers have some flexibility in the discounts they can offer, and it's always worth asking. But don't negotiate so hard that you burn your relationship with your sales engineers. They are fantastic assets. The representative for my company has gone above and beyond multiple times for us, including letting us borrow very expensive equipment for free and spending time helping us figure out our specific test and measurement problems.

When looking for used equipment online, make sure to research the instrument before you buy it. Different options may be enabled or disabled and features may be missing that you were expecting. The best way to do this research is to search for the datasheet or manual of the instrument. These are often old scanned documents that you have to dig through to find the information you want. Pay careful attention to what options and variants of the instrument are available and check the details of the listing of the one you're looking to buy to verify it has everything you need. Unfortunately, there isn't one central location for documentation of all old test equipment. One exception to this is the Tek wiki[2], which is a great resource for all old Tektronix gear.

Test equipment is often overbuilt and lasts a very long time. We run equipment in my lab that is older than I am and it does whatever we

[2] http://w140.com/tekwiki/

ask of it every day. Because of how robust many instruments are, they change hands throughout their lives and subsequently drop in price. eBay is one of the best places to find old equipment. Keysight even runs their own official eBay store to sell used units. Besides eBay, you can ask test equipment rental companies about their old units, check out ham radio swap meets, or watch for auctions from old military labs, university surplus, and bankrupt tech companies.

9.4.1 Test Leads

Every labs needs test leads. Use Pomona as often as you can afford. They're more expensive, but they're by far the best test leads out there. You will need banana jack to banana jack, banana jack to mini-grabber, mini-grabber to mini-grabber, mini-grabber to BNC, and banana jack to BNC cables. Banana jacks plug into your multimeter, power supplies, and low frequency signal generators. BNC jacks plug into oscilloscopes, frequency counters, and some signal generators. Mini-grabbers are used to attach to your circuit. These connectors only work for DC and low frequency signals.

For RF signals, you'll need coax cables (probably SMA, depending on the frequency you're using), terminators, directional couplers, and other application specific adapters. If you're using a VNA, you should consider getting some instrumentation quality cables for it. These are rated for high frequency, have very low loss, and maintain a constant phase despite being moved around. Instrumentation cables can cost thousands of dollars apiece. Yet again, eBay is a great place to find used cables for much cheaper. Expensive cables will often come with a sheet of measurements taken on those exact cables, so you know what the loss and S11 is. It's a good idea to take these measurements yourself on cheaper cables that don't come with measurements. Often times having lossy cables doesn't really matter, as long as you know what the loss is and can account for it.

9.4.2 Power Supplies

If you need a really cheap power supply, search for "DPS adjustable power supply module" on eBay. These are small benchtop power supply "front ends" that cost about $40. You need to supply a fixed voltage and current, but the DPS turns that into an adjustable power supply with a color LCD screen, current limiting, and power limiting. It's not going to have high accuracy, but it's really not a bad little power supply. There are several models available with different max voltage and current limits. Search terms like DP20V2A or DPS3003 to find

them. Another nice Chinese gadget to have on your desk is something known only on eBay as a "component tester". This is a little LCD screen that runs on a 9V battery and measures various parameters of transistors, MOSFETs, diodes, capacitors, inductors, and resistors. You can find them for about $10. Again, no guarantees on accuracy, but it's really handy to be able to quickly figure out what loose components you've got laying around are.

Nicer bench top power supplies come from companies like Keithley, BK Precision, and Keysight. These start in the mid-hundreds of dollars and end, as usual, in the thousands. They are very accurate and most come with extensive programming and automation abilities. This is great for when you want to set up an automated testing rig and need to control your benchtop supplies in software. To save money, look for supplies with multiple independent channels. Instead of buying two power supplies with one output each, you can buy a single supply with two outputs.

9.4.3 Multimeters

Multimeters are used to measure voltage, current, continuity, resistance, and sometimes capacitance and diode forward voltage. They come in either handheld or benchtop versions. A multimeter should be the first piece of test equipment that you buy. There are a plethora of cheap Chinese multimeters that you can buy online, but they're typically inaccurate and unreliable. Buy a Fluke instead. The Fluke 115 is a great first multimeter. It's about $200, is accurate (and remains calibrated for a long time) and very rugged and reliable.

Keysight makes great benchtop multimeters. While they're not very portable and must be plugged into the wall, they offer much higher precision than a handheld multimeter can. They are also capable of taking measurements programmatically and remotely. Some models have the ability to measure in picoamp resolution, which is really useful when characterizing low power systems.

When buying multimeters, you'll often see the display specified in half digits, as in "5 1/2-digit display". This refers to digits that don't have a full range of zero to nine. For example, a 5 1/2-digit resolution would mean that the display has five digits that can range from zero to nine, and one digit that ranges from zero to one. The "1/2" means the digit has a maximum value of one, and two possible values (zero and one). If you had a "4/5" digit, that would mean the digit could display a maximum value of four, with five possible values (zero, one, two, three, and four). Pretty much all multimeters will only use 1/2 digits to express range. Remember that display digits are not the same

thing as resolution! The thing you really want to look for when buying a multimeter is resolution and full-scale voltage range, not necessarily how many display digits there are.

9.4.4 Oscilloscopes

After the multimeter, the first big test equipment purchase a lot of people make is an oscilloscope. There are lots of things to consider when buying an oscilloscope, but bandwidth is one of the most important. As a rule of thumb, buy a 'scope that has a bandwidth at least three times higher than the highest frequency signal you want to measure. This may seem confusing, especially if you have a tight budget. Why can't a 1 GHz 'scope see a 1 GHz signal? Well, it can, but when you're measuring a 1 GHz signal, you actually need to be able to see frequencies above that. If you can see up to the third harmonic of the signal of interest, you will have a much better view of the true shape of the wave, especially the edges.

You can find oscilloscopes in all price ranges, from under $100 to tens of thousands of dollars. Rigol is a popular entry level brand, with their 1000 series models as low as $260. This will get you two channels and a bandwidth of 50 MHz. At one point, someone discovered that it was possible to hack the firmware on the 50 MHz version of this oscilloscope and unlock the bandwidth to be 100 MHz. This involves running potentially sketchy code on your oscilloscope and may not even be possible depending on the version of firmware your oscilloscope is running, so look into this at your own risk. Rigol makes higher end 'scopes too, with much lower prices than comparable models from Keysight or Rhode and Schwarz. I have not tested these higher end models, so I can't vouch for them, but it may be worth a look if you have no other option.

Tektronix has some low end 'scopes that start at about $800, but has received most acclaim for its MSO series. The MSO/DPO2000B starts at about $1400, has 4 analog channels, 16 digital channels (for logic analyzer functionality), and up to a 200 MHz bandwidth. It's a great general purpose lab oscilloscope. It has a big brother, the MDO3000, which has up to a 1 GHz bandwidth, as well as spectrum analyzer and signal generator functionality. The MDO3000 starts at about $3600 and goes up to about $15,000 for the highest performance version.

eBay is rife with old 'scopes made by HP, Agilent, Tektronix, and Rhode and Schwarz. Don't be afraid of an analog oscilloscope. They may be giant, heavy, and lack the fancy interfaces of digital oscilloscopes, but if they have the specifications you need, they'll work great.

If you go the eBay route, make sure to have the device calibrated before you use it. The original manufacturer *may* still offer this service for the undoubtedly discontinued model you bought, but other test equipment distributors or rental companies can calibrate it if the manufacturer won't.

An oscilloscope is only as good as its probes. If you have a 'scope with a 10 GHz bandwidth but crappy probes, you're not going to see anywhere near 10 GHz of bandwidth. Unfortunately, high-performance probes can cost up to ten thousand dollars. Don't forget to factor this into your budget when buying an oscilloscope!

9.4.5 Logic analyzer

While many people's first big lab equipment purchase is an oscilloscope, it should probably be a logic analyzer instead. Most of the time when I want to see a time domain signal, it's from a communication bus like I2C or SPI. A logic analyzer works much better than an oscilloscope for that. You can see the clock and data lines at the same time, and you can also decode the bus traffic directly. Some oscilloscopes can do this too, using multiple channels and potentially a software decode option, but you can usually get a logic analyzer for much cheaper.

For most applications, you can get by with a Saleae logic analyzer. This is a little USB device with either eight or 16 channels that runs on free software from your laptop. The software is really fantastic, and you can download it to play around with it before you decide to buy the hardware. The downside of a Saleae is that it's not great at advanced triggering and it's not as fast as more expensive analyzers. The probes are only ok quality, and it doesn't support any differential buses. However, bus decoding works great, and it can even decode low speed USB. The cheapest model starts at $400.

More expensive logic analyzers have many more channels available, can capture faster events, and can do more complex analysis of the signals. Fewer manufacturers are making purpose-built logic analyzers and are instead incorporating that functionality into oscilloscopes.

9.4.6 Spectrum analyzer

If you're going to be working with RF, you're going to need a spectrum analyzer. New spectrum analyzers from Keysight or Rhode and Schwarz usually start at about $20,000 and can run up to over $100,000. Price is most strongly correlated to maximum frequency.

You can get used analog spectrum analyzers for much cheaper than you might expect. The Tektronix 492 has a massive range of 10 kHz to

21 GHz. You can find these on eBay for about $1000-$2000. If you buy one, make sure it has options 1, 2, and 3. However, the interface on the Tek 492 is a little vintage and doesn't include all of the buttons and functions you would expect from a modern spectrum analyzer. One of my favorite models is the HP 8595E. This is a great little instrument that goes up to 6.5 GHz and can be had for about $2000. The HP 8596E is in the same family and goes up to 12.8 GHz for $3000-$4000.

Handheld spectrum analyzers are nice for working in the field or carrying around the lab. The FieldFox from Keysight (and previous versions from Agilent) are high-performance instruments that run off of a battery and can even act as a two port VNA to measure S parameters. The RF Explorer[3] is a low-cost, low bandwidth, portable analyzer that starts at only a couple hundred dollars. Different models cover different bands, but the highest frequency model can reach about 6 GHz. It can also plug into your computer and be used with more full featured software.

USB spectrum analyzers are also available, often priced at a few thousand dollars. Software Defined Radios (SDRs) can also be used as a USB spectrum analyzer, and are available at lots of different performance levels and price points. The cheapest example is the RTL-SDR, which costs about $30 and can measure up to 1.75 GHz. It has a lot of software support, but is limited to a fairly small bandwidth. The other advantage of SDRs is that they can demodulate the signal that they are measuring. This is a feature usually relegated to very expensive modern spectrum analyzers with equally expensive options enabled. However, purpose-built spectrum analyzers will typically have more sensitivity, resolution, and a better noise figure than an SDR.

9.4.7 Vector Network Analyzer

VNAs are the most expensive piece of test equipment in most labs. New VNAs made by companies like Keysight, Rhode and Schwarz, and Anritsu can easily hit $50,000 to $100,000 on the low end. They also have a plethora of options and upgrades available, which a sales engineer can help you navigate. Most VNAs are two port, meaning they can measure four S parameters. Four port VNAs are also available, and modular VNAs can provide up to 32 ports to work with.

If you're on any sort of budget, look for old HP, Agilent, Keysight, Rhode and Schwarz, or Anritsu VNAs on eBay. They can be had for surprisingly cheap ($1,000 to $10,000). Note that some old VNAs come in two parts: the display and the test set. Before you get excited that

[3] rfexplorer.com

you found a VNA for $500, make sure that it includes both the display and the test set, or it has the test set built in. Even if you work with very high frequencies, some older VNA models can work up to 110 GHz. My favorite used VNA is the HP 8753D. They're available up to 6 GHz and cost between $3,000 and $5,000.

If you have only hundreds of dollars and need a VNA, you're going to have to sacrifice performance. You can buy USB VNAs (although these can quickly get into the price range of a used VNA from eBay), or you can use a handheld VNA. Both of these options are more affordable, but won't have the same frequency range and will have a reduced set of features. A really cool recent entry into the market is the NanoVNA[4]. This is a handheld, 2 port VNA available in several frequency ranges: 50 kHz to 900 MHz, 10 kHz to 1.5 GHz, and 50 kHz to 3 GHz. The most amazing part is the price: $75-$125. These have become really popular with hobbyists recently and if you work with anything in those frequency ranges, it's hard to see a reason not to buy one.

Unlike a multimeter or spectrum analyzer, a VNA needs to be calibrated every time it's turned on. This is done with a calibration kit that contains a short, an open, and a load. Calibration kits can be shockingly expensive. Buy them used if possible. The reason they're so expensive is that they work over a very large frequency range. There are two types of calibration kits: electronic or e-cal kits, and mechanical calibration kits. Mechanical calibration kits contain a short, open, and load in a connector form factor. One by one, these are screwed onto the ports of the VNA for calibration. An e-cal kit saves time by allowing you connect each port of the VNA to the e-cal kit, and then plugging in the e-cal kit to the VNA with a USB cable. The VNA controls the e-cal device through USB and switches between a short, open, and load automatically, rather than requiring you to manually attach and remove each successive calibration connector. Manual calibration kits are usually cheaper than e-cal kits. You may be able to build your own calibration kit by modifying some connectors, but it's unlikely that your 50 ohm resistor looks like 50 ohms from zero to 26 GHz. What you can get away with all depends on the accuracy you need and your frequency range. For most applications, it's not worth it to skimp on the calibration kit when you spent thousands of dollars on the VNA itself. You want your measurements to be accurate.

Since VNAs are sensitive to phase, you also need to calibrate out any test cables or pigtails you're using. To do this, you can either perform a calibration with the cables connected, or you can use what's called a port extension. On old VNAs, you need to manually tell it the

[4] nanovna.com

length of the cables you're using and it will compensate for the added phase shift. New VNAs can do this automatically.

9.5 Small Hand Tools

A microdrill (sometimes called a micromotor) will allow you to perform extremely precise and clean rework that would otherwise be impossible. A microdrill looks and sounds like a dental drill, and it basically is. It's kind of like a smaller and more precise Dremel tool. It has a hand piece and different burr tips as well as speed control and sometimes a foot pedal to turn it on and off. You can use a microdrill to excavate small areas of the PCB to perform rework on inner layers. A microdrill is much faster than an X-Acto knife for cutting traces, but you need to be careful not to cut too far. Good microdrills are several hundred dollars. You can find cheaper models on eBay, but they don't have a lot of torque.

A good pair of tweezers is absolutely necessary. Jeweler or surgical tweezers are best. Titanium tweezers are strong and flexible, but they can be quite expensive. Avoid tweezers that are magnetic or can be magnetized since it can be very frustrating to try to release parts that keep sticking to your tweezers. Having a couple pairs of cheap $5 tweezers around is good for tasks that risk damaging your more expensive pairs. Get as fine of a tip as you can. A good pair of tweezers should be sharp and the tips should make full contact when squeezed shut. Both curved and straight tip tweezers are good to have around. Some very large parts may be too wide for normal tweezers to grasp, so some larger pairs can be helpful for that.

Tweezers are not the only medical tool that find use in an electronics lab. A good pair of hemostats can be great for reaching difficult places in an assembly and can also be used in lieu of a pair of tweezers for large components. Dental picks can be used to scrape away solder mask, flux, or other debris on the PCB. A scalpel is a good upgrade from an X-Acto knife, but make sure you get a #10 blade it with. Just be careful because it will be substantially sharper than an X-Acto blade. Try to get surgical grade tools if you can, because the cheap stuff just isn't worth it.

Diagonal cutters, sometimes called angle cutters or flush cutters, are constantly used. The most common task is to cut the excess length from soldered through hole components, but they can also be used to cut almost anything else. A word of warning though: do not attempt to cut thick wire with small angle cutters. Small angle cutters are not designed to bite through larger gauge wire and will cause dents on the

blade that ruin the tool. Use wire cutters to cut anything thicker than about 20 gauge.

Wire strippers come in two flavors: manual and automatic. Have at least one pair of each. Automatic wire strippers are very fast and make a perfect strip each time. Manual wire strippers work with any gauge wire (including those that your automatic stripper will not) but there's often a danger of biting down too deep and damaging the wire under the sheath. You can also buy thermal wire strippers, which use heat to melt the insulation around the wire and pull it off.

Needle probes are one of my favorite tools. They're very cheap and I use them constantly. A good pair will be all metal and conductive. You can use a mini-grabber cable to grab on to the top of the probe and use a pair of needle probes to probe very hard to reach places on a PCB with a multimeter. Besides being able to reach tight places, needle probes can also be pressed down into a trace or flux covered solder joint to make good contact that a normal multimeter probe might be too blunt to make.

A thermal imager can very quickly help locate dead shorts or thermal management problems. Until several years ago, the only thermal imagers available cost thousands of dollars and were very fragile. Today, smartphone thermal imagers like the FLIR One or Seek Thermal let anyone have access to this technology for only a couple hundred bucks. The resolution and framerate isn't as good as the expensive versions, but it's usually sufficient for simple thermal analysis of a PCB. Another cheap way to get a good thermal camera is to, again, check eBay. You can sometimes find FLIR cameras with a firmware hack that unlocks software restricted features only available in more expensive models (but have the same hardware as the less expensive models). Search "FLIR upgrade" to see if any are available. You can also perform this hack yourself using instructions and files available on various forums. But beware, if anything goes wrong during your hack attempt, or if the hacked one you buy breaks, your warranty is extremely void and you'll need to buy another camera.

You're going to need a full screwdriver set. IFixit and other companies sell a kit with a small screwdriver and 64 different bits that will cover almost any application you encounter. Another neat screwdriver is what's called a screw holding screwdriver. A company called Klein Tools makes these, and they have a special bit that can hold the screw to the tip of the screwdriver, even if it's not made of metal or magnetic.

10

Troubleshooting

> *In essence, when machines work properly, it's because they have no choice.*
>
> – How to Diagnose and Fix Everything Electronic

Strong troubleshooting skills are among the most useful things a designer can cultivate. Nothing ever works exactly as planned on the first try. The key to making rapid forward progress is the ability to quickly assess the problems you encounter and fix them correctly. This means not just treating the "symptoms", but the root cause of the problem. Fixing the symptoms may buy you a little time (good enough for a hackathon demo or something similar), but in the end it will only make things worse. In the case of safety critical devices, correctly diagnosing and fixing the root cause of the problem is even more important because human lives are on the line. Troubleshooting can also be one of the most frustrating aspects of engineering because it almost always takes longer than you wish it did and you can feel exasperated when it seems like nothing you're trying is working. A joke among computer scientists is that the stages of debugging go like this:

- That can't happen.
- That doesn't happen on my machine.
- That shouldn't happen.
- Oh, I see.
- How did that ever work?

10.1 How Electronics Fail

Electronics failures can be grouped into two categories: a design problem and not a design problem. Design problems are simple: you designed something, but not what you thought. Somewhere there's a mistake in your understanding of how the system should work and you either left something out or added something wrong in. The second kind of failure isn't your fault (directly). These failures are a result of manufacturing problems or environmental factors. A list of causes of this second kind of failure is below[1].

1. Chemical / Contaminant
    (a) Moisture Penetration
    (b) Electro-Chemical Migration
    (c) Pitting / Crevice Corrosion
    (d) Free ionics not bound or cross linked
2. Electrical
    (a) Electro-Migration
    (b) Conductive Filament Formation
    (c) Thermal Degradation
3. Mechanical
    (a) Fatigue
    (b) Creep
    (c) Wear
4. Over Stressed Issues
    (a) Supplier compliance to material specs
    (b) Field loading
    (c) Product durability

Electrical Overstress (EOS) is an overvoltage or overcurrent event that typically lasts longer than an ESD event. EOS can cause failures

[1] M. McMeen, J. Tynes, M. Bixenman and D. Lober, "Electronic assembly warranties challenge the industry to improve risk mitigation test methods," *2016 Pan Pacific Microelectronics Symposium (Pan Pacific)*, Big Island, HI, USA, 2016, pp. 1-13, doi: 10.1109/PanPacific.2016.7428406.

in two ways: thermal damage and dielectric breakdown. An overvoltage condition will break down the dielectric inside a component and conduct thought it. This is sometimes called avalanche breakdown or shoot through. Thermal damage results from overcurrent conditions. A large current through even a small resistance will create heat, and this heat can be enough to damage components. One common example of this is bond wires in an IC melting and creating an open circuit.

Understanding the root causes of why electronics fail is important in helping you troubleshoot problems to trace them down to the root cause. How close to the root cause you are able to get may depend on the equipment and facilities available to you. For example, discovering that a microvia has caused delamination in an inner layer of a circuit board requires sectioning the board, preparing the section, and looking at it under a microscope. Dedicated failure analysis labs do this kind of thing all the time, but you may not have the tools or skills to do this. In that case, you won't be troubleshooting down to the root cause, you'll be troubleshooting to determine what the problematic component is. In this example, you would take apart your system and eventually isolate the issue to some unknown problem on an inner layer of a particular PCB. You could then either replace the board and hope the same issues doesn't arise on the replacement, or you could send off the board to a failure analysis lab and have them figure out the root cause.

10.2 Tips for Troubleshooting

If at any point you find yourself wondering if a component came from the manufacturer defective, you're almost certainly wrong. This basically never happens. More likely, something's not hooked up right, or you killed the part.

One of the first things you should look for when troubleshooting a problem is accidental short and open circuits. A thermal camera is a great way to quickly do this. When you apply power, you'll be able to clearly see short circuits caused by a component since they'll quickly get hotter than everything else. If you have an open circuit somewhere and a part isn't getting power at all, you'll also be able to see this on a thermal camera because that part will remain totally cold while all the parts around it begin to heat up as soon as you apply power. Because modern thermal cameras are so sensitive, even a change of a few degrees that would be difficult to perceive with your finger is very obvious.

Your nose is a very important tool when troubleshooting electronics. There is a very particular smell that damaged parts emit, and

with experience, you will be able to identify it almost instantly. Even when a board is inside of an assembly or enclosure, putting your nose against it and smelling can tell you if something failed inside. Once you suspect a problem, you can use other techniques like the thermal camera to figure out exactly which component is at fault.

When you first start to debug a problem, check for the "obvious" stuff first. Is it plugged in and turned on? Is the power polarity right? Is it configured correctly? How about your test equipment? Did any wires come unsoldered? These things are all quick to check and quick to fix, so check them first before you start tearing the whole device apart looking for subtle bugs.

If you have problems getting an IC to program or respond to input, make sure the part isn't being held in reset. This can happen either because of a design mistake or because of a short or open on the PCB due to an assembly error. Measure the reset pin of the chip in question. If the chip runs on an external oscillator or clock, measure it with a 'scope to make sure it's within the right threshold voltages and doesn't have undershoot or overshoot. If communication busses aren't working, you should first use a logic analyzer to verify that each pin has the right signal on it and that the frames and packet structures are correct. You can also use your 'scope to make sure the noise and signal integrity of your buses is acceptable.

If you're having problems with EMI, EMC, or oscillation of some kind, a field probe is a good way to locate the offending component or area on your PCB. There are two kinds of probes: electric field (E-field) and magnetic field (H-field) probes. You can buy probes ranging in cost from a few hundred dollars to thousands. Depending on the frequency of the things you're trying to measure, you may or may not need to spend a lot of money on probes. An H-field probe can be easily built with a loop of rigid co-ax cable for a few dollars, but its frequency response may not be very good at higher frequencies. Some probe kits come with small amplifiers you can use to try to compensate for this. If you want to make precise measurements of signals above 1 GHz, you'll need a pre-built and characterized probe set that is rated for the frequency you're interested in.

In general, you should isolate each problem on its own. Don't try to investigate multiple problems at the same time. Change only one variable. Once you start experiencing the problem, start removing things until the problem goes away. Then, once the problem has stopped, start adding parts back in until the problem reappears. It's very important that you take a second to think about your experimental setup and whether you are actually changing only one thing at a time.

If the thing you're debugging has both hardware and software, try to first isolate the problem to one or the other. This is easier said than done, and it's possible that the problem is in both, or that there is more than one problem. But think about what tests you can design that will tell you whether you should focus on hardware or software debugging first. If you have a device that is working correctly and one that is not working correctly, you can try inducing a failure in the functional device to replicate the issues you're seeing on the faulty device. You can also try swapping hardware or firmware between the two to narrow down the source of the problem.

If a problem mysteriously solves itself, be very careful. Do not assume that the problem is fixed and will never come back. Take the time to continue troubleshooting. It's possible that a change in the environment or usage of the product or test made it go away. Try to make it come back. Knowing how to reliably replicate the problem is important so you can understand what conditions are necessary for the problem to occur, and so that you can test whether or not you fixed the root cause or just a symptom.

Ask for help. Especially after debugging something for a long time, it's really beneficial to have a fresh set of eyes look at the problem and try to find things that you've overlooked. Ask people with more experience than you, and people with expertise in different fields than you. Sometimes they can spot problem that is obvious to them but invisible to you.

You must document your troubleshooting process. You will have no hope of keeping track of what you've tried and what to try next unless you record not just your results, but your data and experimental setup. Collect as much data as you can before, during, and after failures. This is important not just for future you, but for when you ask for help from other people or so other people can replicate your results. More information about keeping a good lab notebook, as well as an example, can be found at the end of the "Prototyping" chapter. There's also more information about documentation as it pertains to debugging later in this chapter under the "Narrative Based Troubleshooting" section.

Know the system you're debugging. If you designed all of it, this is easy. It's harder if you only designed part of it, or even none of it. Do your homework upfront and read the available documentation, talk to the designer(s) if you can, and understand how the device was designed and what design choices were made. Likewise, know the tools you're using to debug. Know their limitations. A good resource is the application engineers at the company that made any test equipment you're using. They can help you make sure you're set up correctly to

accurately take the measurements you intend.

Consider the effects that adding instrumentation may have on how the system runs. Logic analyzer probes add a bunch of capacitance, ammeters add resistance, etc. You can try to account for this by making sure you can replicate the problem after instrumentation is added. Again, when in doubt, ask the application engineers at your test equipment manufacturer and they should be able to help.

Quit thinking and look. This is a wonderful piece of advice from the book Debugging by David Agans. Looking is hard and you're not going to want to do it. You have to set things up, solder wires, replace parts. It's a whole ordeal. You will sigh when you consider the work you need to do just to get results that may or may not be helpful. This will cause you to want to sit at your desk and think hard about what the problem could be, or talk to people about what the problem could be, or think up new, simpler experiments that you could run instead. On their own, there's nothing wrong with any of that. But do not let it slow you down from actually putting in the work and trying things. That's the only way to get new information.

10.3 Medical Model for Troubleshooting

The first troubleshooting model we'll look at is the process that is used in medicine. Doctors make observations of a patient, perform tests, and try to conclude what the underlying disorder(s) are to plan a treatment. Their troubleshooting processes need to be adapted slightly for electrical engineering use because the mechanisms of the problems and disorders of living things are quite different from a circuit. Still, the concepts are valid. The human body is the most complex machine we know about. Imagine trying to debug a system that complex, and consider the fact that we don't even fully understand how all the parts work! The troubleshooting techniques in this section are good enough for that machine, so it's safe to say they're good enough for whatever we design.

Medicine makes the important distinction between the signs, symptoms, and etiology of a disorder. Signs are observations that anyone can make. Symptoms are experiences that the patient has which differ from their norm. Etiology is the underlying cause of signs and symptoms. So for a product, we would say that the signs are things we can observe on the PCB, like a burned-out component or a cut trace. Symptoms would be behaviors of the circuit that are unexpected or not normal. And the etiology would be the root cause of the problems (like plugging in a battery backwards or exceeding the temperature rating

of the board).

To determine whether or not a suspected mechanism is causing a particular problem across multiple units, you can use something called Koch's postulates. These are a set of statements that pathologists use to help determine if a particular pathogen is causing a particular set of signs and symptoms. Here is an adapted version of Koch's postulates that can be used to help determine if a failure mode is due to a certain cause when there are multiple instances across multiple units.

- The suspected cause or causes of the problem should appear to be present in all cases known to be defective and not in units that are not defective. Note that this requires full failure analysis of every faulty unit because sometimes signs may present themselves differently in different units, even if they had the same failure cause. For example, an IC can fail catastrophically and be clearly burned, or it can fail internally and look fine from the outside. When both of these signs can occur due to the same underlying cause, you may be misled into thinking that you are dealing with two different problems instead of one. Failure analysis will prevent this.
- When fixed, the signs and symptoms of the problem should go away. If the problem starts happening again, the signs and symptoms should come back. If this happens, you likely haven't found the true root cause and have been treating symptoms. If different signs and symptoms come back, you may be looking at a new problem with a new underlying cause.
- If you noticed signs or symptoms before a significant failure, or if the signs or symptoms get worse as the suspected underlying cause gets more severe, the suspected cause is more likely to be correct.
- The failure modes inferred from your observations should be consistent with known failure modes of those components.
- The signs and symptoms of the failure should be reproducible if you purposefully create the suspected underlying cause in a known working unit.

Anyone familiar with the original Koch postulates will know that they have been very heavily adapted here, but the underlying logic for troubleshooting is valuable and works outside of just medicine. Some of the postulates above may seem obvious or elementary, but they're important to remember. It's very easy to get "lost in the woods" when troubleshooting and start walking in circles or forgetting the basics.

What happens when you have multiple problems stacked on top of each other? Or what if a sign or symptom is just a fluke, a one time thing? You may be tempted to do intensive observations and try to correlate behaviors with signs or see if there's a statistical link between behaviors. It's not suggested that you do this. The teaching in the medical world is to instead perform experiments. This is because it can take a long time to gather enough observations for a meaningful analysis, and you still won't be able to say with absolute certainty if you have correctly identified the cause. Performing experiments usually means correcting what you think is the underlying cause. It can also mean changing or perturbing the device to get a better understanding of the "rules" of the symptoms and signs. When you begin to understand how symptoms work (whether certain symptoms always appear together or never appear together, under what circumstances, etc.), the medical community calls it a syndrome. A syndrome can be caused not just directly by an underlying cause, but also by a series of causes, each creating the next. You may have heard of "asking five why's" when troubleshooting, and that's what it's trying to address. If you incorrectly identify an underlying cause that is really a sign or symptom with a deeper underlying cause, the problem will eventually reoccur. A correct underlying cause is not a sign or symptom itself, and can always explain all other signs and symptoms. Keep in mind that some underlying causes may have signs and symptoms that are subject to a promoter. For example, if some of your devices start failing whenever the PCB is installed into the enclosure, the underlying cause is not that the PCB is being installed into an enclosure. Let's say after some more analysis, you eventually determine that the PCB is flexed slightly when it is screwed into the enclosure and that flex is creating an open circuit on a bad solder joint. This is an example of a problem that already existed, but wasn't visible until something increased the expression of the problem (in this case, screwing it into an enclosure). Flexing the PCB is not the underlying cause that explains the syndrome here, the bad solder joint is.

A very common problem in medicine is trying to determine whether or not there is any relationship between a suspected cause and observed effect. This can be difficult enough for a complex electronic device, but you and I will never design anything as complex as the human body. Studies attempting to find links between behavior or environment and health are published every day, but perhaps the most famous of these was a series of studies linking smoking to lung cancer. Sir Austin Bradford Hill was a statistician who proposed a set of criteria for establishing a link between a cause and effect. I have adapted them slightly for

electrical engineering.

- A small association does not mean that there is not a causal effect, though the larger the association, the more likely that it is causal.
- Consistent findings observed by different persons in different places with different samples strengthens the likelihood of an effect.
- Causation is likely if there is a very specific sign or symptom at a specific location and has no other likely explanation. The more specific an association between a factor and an effect is, the bigger the probability of a causal relationship.
- The effect has to occur after the cause (and if there is an expected delay between the cause and expected effect, then the effect must occur after that delay).
- Greater exposure to the underlying cause should generally lead to greater incidence of the effect. However, in some cases, the mere presence of the underlying cause can trigger the effect. In other cases, an inverse proportion is observed: greater exposure leads to lower incidence.
- A plausible mechanism between cause and effect is helpful (but Hill noted that knowledge of the mechanism is limited by current knowledge).
- Occasionally it is possible to appeal to experimental evidence.
- The effect of analogous similar factors may be considered.

Doctors use medical history, physical examination, and diagnostic tests to help learn more about the disorder they are attempting to treat. They also frequently consult with specialists to take advantage of experience they might not have themselves. These same techniques can be used when troubleshooting electronics. If the failure happened in the field, get as much information about the usage history and conditions of use as possible, ideally from the users directly. Carefully examine both the hardware and behavior of the device to identify signs and symptoms. Perform tests to help identify the root cause. And don't be afraid of reach out to people who specialize in the suspected problem area. If you don't know anyone personally who you can reach out to, contact an application engineer.

Another method used by doctors to identify a disorder is something called differential diagnosis. This is basically a fancy way of saying "process of elimination". You collect a list of signs and symptoms, list

all possible causes for those signs and symptoms, and start trying to rule things out. In medicine, this list is prioritized by urgency or danger to the patient. If you manage to rule out everything on your list, that means you either made a mistake and forgot a possible explanation, or it means that this is a never-before-seen problem. Search the internet, ask on forums and talk to other engineers to try to discover other instances or similar instances of the problem you're seeing.

10.4 Narrative Based Troubleshooting

Narrative based troubleshooting is not a precise troubleshooting methodology, but more of a general practice that can make troubleshooting easier. The idea is that you treat your lab notebook like a story instead of a spreadsheet. Write in it like a journal. For example: "Today I picked up from yesterday and replaced C12. I thought it might have heat damage because it looked pretty burned out. I didn't have another 12 µF capacitor, so I replaced it with a 10 µF capacitor. Still didn't work. Maybe that filter in front of U4 has too low of a roll off. I'm going to replace R34 with a 4.7k resistor. Ok, that helped a little." Etc. You will find yourself making a lot of changes during troubleshooting and it's very easy to forget what you already tried. Pretty soon you end up going around in circles and lose track of the state of your device, which can make troubleshooting way harder than it has to be. Writing everything down in a lab notebook is critical. The advantage of writing it as a story instead of just test results is that it's easier (at least for me) to follow what's going on, where you've been, and where you want to go. This is especially useful if you take a prolonged break or go on vacation during the troubleshooting process and come back later. Just reading through what you were thinking can help illuminate areas you missed and areas you already covered. It's also much easier for another person to take over if they need to, since all of your reasoning is written out instead of just results.

10.5 Scientific Troubleshooting

Scientific troubleshooting is using the scientific method to discover a root cause. There are actually several ways to perform what we call the scientific method. They are all similar and mostly have differences in the philosophy that underlies them. This book is not a study of epistemology, so we will constrain ourselves to the following definition:

- 1. Use your experience: Consider the problem and try to make sense of it. Gather data and look for previous explanations. If this is a new problem to you, then move to step 2.
- 2. Form a hypothesis: When nothing else is known, come up with an explanation.
- 3. Deduce predictions from the hypothesis: if you assume your hypothesis is true, what consequences follow?
- 4. Test (or experiment): Look for evidence that conflicts with these predictions in order to disprove your hypothesis.

A common mistake people make when using the scientific process is to switch steps 2 and 3. Rather than trying to disprove their hypothesis, they look for evidence that it's true and overlook or ignore evidence to the contrary. You see this most often when you really want a particular hypothesis to be true. You might think that only conspiracy theorists are vulnerable to this mistake, but don't dismiss it too quickly. Proving your hypothesis wrong may mean you have to spend a lot of money to fix the issue, or may set your schedule back more than you want, or indicate that your development path is wrong, or even just mean that you have to humble yourself and admit you were wrong. When your project and/or reputation are on the line, it becomes easier than you might think to try to prove yourself right and ignore contrary evidence without even realizing it.

Another related mistake is a logical fallacy called "affirming the consequent". This happens when you take a true statement, like "if the fuse is blown, the device won't power on" and assume that the converse is also true: "if the device won't power on, it's because the fuse is blown". The converse isn't necessarily true. In our example, it's true that the fuse must be intact for the device to power on, but there are lots of things besides a blown fuse that could prevent it from turning on (maybe a dead power supply, maybe it's not plugged in, maybe it's been smashed with a hammer). Affirming the consequent is an especially important thing to watch out for when debugging. If someone debugging a device hasn't even seen a particular failure mode, but the symptoms are similar, they may incorrectly assume they know what's causing the problem. This can be exacerbated if they don't have a deep understanding of how the device works. Affirming the consequent can also accidentally happen if you are otherwise mentally pigeon-holed into a single subsystem or discipline. Maybe you're only looking for software problems when the issue is in hardware, or you're thinking through the electrical system when the issue is caused by the

mechanical system. This is another good reason why you should talk to people on other teams and get advice from other experts.

While many people consider the scientific method as some sort of oracle that produces absolute truth, the reality is that the scientific method is a *process* that can only inch us closer to the truth. That means that you can never absolutely verify your hypothesis. There's a famous Einstein quote that encapsulates this: "No amount of experimentation can ever prove me right; a single experiment can prove me wrong."

11

Appendix A: Rules of Thumb

- 50 ohm Microstrip: 1 pF/cm and 3 nH/cm
- Bandwidth = 0.35 / (rise time in ns)
- Bessel filtered signal rise time = 0.35 / 3 dB bandwidth
- You can safely terminate up to 17 dBm (50 mW) into an 0402 50 ohm resistor.
- A through hole component lead has about 4 nH of parasitic inductance. A surface mount component has about 0.4 nH of parasitic inductance.
- If you need a low pass filter with the smallest possible overshoot in the response in the time domain, use a Bessel filter (or other phase linear filter).
- If you need a low pass filter with the fastest possible roll off in the frequency domain, use a Cauer filter (or another elliptic filter).
- Every 3 dB increase of a signal doubles the power.
- **Power** in decibels is $10 * \log_{10} \frac{P_1}{P_2}$
- **Voltage** in decibels is $20 * \log_{10} \frac{P_1}{P_2}$
- The dielectric constant of regular FR-4 is usually between 4.3 and 4.7.
- Resonant frequency of an LC tank circuit is $f = \frac{1}{2*\pi*\sqrt{LC}}$

- On FR-4 with a dielectric constant of 4.2, signals travel 6 inches per nanosecond, or 165 ps/inch
- The time it takes a signal to traverse a trace should be at least 5 times as long as the rise time of that signal.
- Characteristic impedance = $Z_0 = \sqrt{\frac{L}{C}}$
- Lower ϵ_r = higher impedance
- Wider trace = lower impedance
- As distance between the trace and plane goes up, impedance goes up
- Thickness of PCB to use as a USB connector: 1.63 mm
- Don't worry about trace corners unless the width of the trace in mils is greater than 5 times your rise time. Alternatively, if you assume a rise time of 10ps, don't worry about your corners unless your 50 ohm trace width is wider than 50 mils.[1]
- The mismatch between 50 ohm and 75 ohm connections is only about 0.177 dB. So if you find a part that is rated for 75 ohms instead of 50, it might be ok to use.
- 1 millimeter = 1000 µm
- 1 inch = 1000 mils
- 1 mil = 25.4 µm
- 1 millimeter = 39.37 mils
- A line half as long can handle twice the speed

- To be able to treat something as a lumped element, all parts of it must be smaller than 1/10th of a wavelength at the highest frequency you'll be using.
- If you are $\frac{2d^2}{\lambda}$ or farther from an antenna, you are in the far-field.
- If your input source has an impedance of less than 100 ohms, use a series-shunt or T filter. If the input source has an impedance greater than 100 ohms, use a shunt-series or Pi filter.

[1] E. Bogatin, "When to worry about trace corners: Rule of Thumb #24," EDN, 04-Mar-2020. [Online]. Available: https://www.edn.com/when-to-worry-about-trace-corners-rule-of-thumb-24/. [Accessed: 03-Mar-2021].

12

Appendix B: How to Give a Demo

A bad design with a good presentation is doomed eventually.
A good design with a bad presentation is doomed immediately.
– Excerpt from Atkin's Laws of Spacecraft Design

An effective demo of your product or technology can be a matter of life and death for your company. A successful demo works when it's supposed to, demonstrates your core technology, and effectively communicates your message. The first step to crafting a great demo is to determine the purpose of the demo. Are you trying to get funding? Impress someone? Sell a product to customers? Think about the message you're trying to communicate. Make sure what you're demonstrating conveys and supports your message. Make the demo specific to your audience. Don't talk about your product in terms of what *it* can do, talk about it in terms of what it can accomplish for the user. There's a big difference between "our product has 32 gigs of storage" and "our product has enough storage that you don't need to ever worry about running out of space".

Think about the technical level of your audience. Almost no one will admit that they don't understand something during your demo. You can even ask people, "Do you understand?" and they will lie to your face and say yes, just so they don't feel the slight sense of shame that comes from admitting you need something explained more simply. If people don't understand what's going on or why your demo matters, they'll leave without getting the point you're trying to make and you won't even realize it. Give background and context for your demonstration.

If you know who you're demoing for ahead of time, you can take time to learn what they care about, their technical level, and the exact problems they face that you'll be addressing. You learn this information directly by asking questions, and indirectly by researching the people, organization, and market of the audience. But sometimes you'll need to demo to an audience you know nothing about, like at a convention. In that case, before you demo, you need to take a minute or two to ask some probing questions to get a critical understanding of your audience. Ask questions that get them talking about the problems they face. As they do this, start thinking about whether they're describing symptoms or a real root problem. Remember, you're not selling them your product, you're selling the outcome that your product will provide. Adjust your demo in real time to their particular needs.

Clarity and relevance aren't the only things you're fighting for in a demo. You are also in a battle for attention, whether you realize it or not. It's you versus the phone in your audience members' pocket. The second they get bored or uninterested, they will pull out their phone and you're done. There's very little chance of getting them back. Even if you do get them back, they're going to have missed a lot of what you said and did. Be more interesting and compelling than their phone.

Giving a good demo requires good storytelling. Good storytellers know that facts don't inspire people, stories do. This isn't to say your story should be untrue, but it should keep the statistics and numbers to a minimum. You can convey the same information that you would in a list of facts by using a story instead, and it will be much more memorable and have a larger impact.

Just like any story, a demo should have a beginning, middle, and end. Choose and script your demo so it has this flow. Who are the heroes? Who is the villain? What is the conflict? The best stories usually follow a similar formula, called The Hero's Journey. There are lots of versions of this formula, but one of the simplest is the one that Pixar uses as a framework for their stories:

1. Once there was a _____. (The hero)
2. Every day he _____. (The hero's ordinary world before conflict)
3. Until one day _____. (Conflict)
4. Because of that _____.
5. Because of that _____.
6. Until finally _____. (The climax of the story, good triumphs over evil)

7. Ever since then _____. (Moral of the story)

You can use this to help organize the story you're trying to tell. And don't just tell the story of what's happening in the demo. Tell the backstory. Share your struggles. If they don't know the story of the product and don't understand the conflict and difficulties you have overcome in getting the demo to work, they won't be as impressed as they could be. But more importantly, sharing your problems and making yourself vulnerable to your audience connects you to them. You want them to feel the pain of trying over and over and failing, and you want them to feel the triumph of finally succeeding. Make it personal. People won't remember exactly what you said anyway. They'll remember how you made them feel. Connect with your audience before you try to communicate the facts about your product.

When you tell your story, be passionate. Don't just explain what the product does, explain why the things it enables makes your heart sing. What is your higher mission? What does it all mean?

To really wow people, you need to be unexpected. Have a grand finale or climax of your demo. Keep their expectations low at first so you can have the big reveal at the end. Figure out what your audience cares about and play to that. Start with the "easy" part of the demo, then show more and more functionality, ending with the most impressive thing you have. Make sure the audience understands what's going on behind the scenes and why the demo they're seeing is so amazing. Use analogies to explain what's happening, not numbers and jargon. This is especially important if your product is technically complex. It's really easy, especially as an engineer, to want to dive deep into the technology and implementation. It's fun to talk about and it makes you feel smart. But you will lose your audience and it will be almost impossible to get them back. People don't really care about the *how*, they care about the *why*.

To effectively communicate your story, speak slowly, be concise, and use words efficiently. To do this, you need to practice a lot. More than you think. You can become concise by writing out what you want to say and continually trimming it and rewording it and consolidating it. After you know what you're going to say, practice saying it. You can't read what you want to say verbatim, and you can't use notes. You have to memorize it and practice until you sound clear, confident, and natural.

Here are some practical tips for giving a demo that will help things run smoothly. Never rely on anything at the venue. Don't depend on the local Wi-Fi network to be working and have internet. If you have to use Bluetooth, pair the devices before the demo. Make sure

all batteries are charged. If you're using a projector, test your exact computer or product with the exact projector that you'll use for the demo. The goal in preparing for a demo is to have total control over everything you will be presenting with. Do a full dress rehearsal. If you can't perform the demo in the exact room with the exact equipment you'll use, get as close as you can. The demo should be perfectly reproducible. After the rehearsal, don't change anything. Even after all that preparation, have a backup. You can either fall back to a simpler demo or have a video of the demo you made earlier. You don't want to spend 10 minutes fiddling around trying to get things to work. You'll immediately lose the interest of your audience and any momentum you had. Give yourself about 15 seconds to troubleshoot your demo, then fall back to your backup.

Demos often involve prototype, unfinished hardware. A nice enclosure goes a long way in making a technology look more real and plausible. You can use a 3D printed enclosure or a urethane cast. Both of these can be fabricated quickly and fairly cheaply. They'll also look much better than a breadboard and pile of wires.

A demo does not need to have perfect, flawless tech. The demo does need to go smoothly and *look* perfect. There's a famous story about the first iPhone presentation that Steven Jobs gave. It's now iconic (you can find it on YouTube) and shows off the features of a device that blew the competition out of the water. But what no one in the audience knew was that the demo was held together with duct tape and they cut it very close to the deadline to get everything working. The GSM radio in the prototype phone Steve was using performed so poorly that they had a miniature cell phone tower behind the stage to make sure it got service. This still wasn't enough to give it full bars of service, so they hard coded the cell service bars to always show full, even if they weren't. There were actually three prototypes on the podium that Steve discreetly switched out at various points when the previous ones failed and had to be rebooted. Of course, they were able to actually deliver on everything the demo promised. This is the important distinction between showmanship and fraud: you have to be able to deliver. In general, Apple is a great example of how to design a demo. Watch some keynotes to get inspiration.

Make your demo relevant. Play to your audience and show how you can solve the problems that *they* have. Do not just go down a laundry list of features and show every single one. Tailor it to the problem you're solving for this particular audience. Show them how much better you are than the alternative, which can even mean demoing your product alongside your competitor's product. Here's an example

of that. There's a company called Weebly that makes a website builder. For their demo, they rebuilt the websites of the investors they were pitching in 5 minutes while they watched. This was powerful because these people had spent a lot of time and money on the alternative (paying someone a lot of money and waiting several months), so they had a direct comparison to the "better way". You can use comparison and contrast to demonstrate the gulf between your product and the competition. This is a great way to give context to what your demo means and why it's relevant.

If you're pitching investors with your demo, the goal should to get another meeting, not to get a check written right then. You're trying to buy yourself the chance for a much longer meeting with a full pitch. It's a good sign if investors (or whoever your audience is) start acting like they're on your team and make suggestions to help you. They'll go from asking hard questions to trying to answer those hard questions for you. And speaking of hard questions, think of all the questions you don't want people to ask about your product or demo and figure out the answers ahead of time. A perfectly performing demo can be wrecked by a few questions that appear to rattle you and make your audience doubt what they've just seen. Remember that saying "I don't know" or "we don't have that all the way figured out yet" is a perfectly valid and acceptable answer to some questions! No one has everything figured out about a new product, and pretending you can see into the future is often worse than just being honest.

Finally, if you're giving demos to try to raise money, show the demo before you need the money. This starts to get into fundraising advice, which is out of the scope of this book, but I'll just leave you with this oft quoted line in Silicon Valley: *If you want advice, ask for money. If you want money, ask for advice.*

13

Appendix C: Companies that can help

None of the companies in this appendix paid me or even requested me to include them here. Either I have used these companies, or friends of mine have used them. I'm not endorsing any of them, since I have no formal relationship with any of them and can't keep track of whether they're still good or not. But this should at least give you some leads for help in these areas.

13.1 Tools

GrabCAD is a free online repository of thousands of 3D models, including models of electrical components. http://grabcad.com

Memfault provides an SDK for embedded software that handles firmware release management, remote programming and development, error monitoring, and telemetry. http://memfault.com

Octopart lets you search for electronic components with different distributors and manages your BOM to optimize cost and ensure part availability. http://octopart.com

SnapEDA is a massive source of verified PCB footprints, schematic symbols, and 3D models. http://snapeda.com

Upverter is a free, completely in-browser PCB CAD tool with as many features as most desktop-based EDA tools. http://upverter.com

13.2 PCB Fabrication and Assembly

Advanced Circuits is a PCB manufacturer that offers a "bare-bones" service with reduced features but a one day turn time. http://4pcb.com

Assembly helps companies design, build, and deliver products in China. http://assembly.com

CircuitHub is a turnkey PCB fabricator and assembler. Their quoting process is entirely online and instant, and you can use a slider to see how your cost will change when making one board versus 10,000. http://circuithub.com

FusionPCB is Seeed Studio's PCB fabrication service. They're based in Japan and have longer lead times, but their prices are very low. http://www.seeedstudio.com/fusion_pcb.html

Gorilla Circuits is a PCB manufacturer and one of the best assemblers I have ever worked with. http://gorillacircuits.com

MacroFab is turnkey PCB manufacturer and assembler based in the US. http://macrofab.com

OSHPark offers two-layer, four-layer, and flex PCBs at extremely low costs, but slightly longer lead times. http://oshpark.com

OSHStencils makes low-cost solder paste stencils to aid in assembly. http://oshstencils.com

PCB:NG is PCB assembler with low pricing and a really nice web interface. http://pcbng.com

Tempo Automation is an advanced turnkey PCB fabricator and assembler with online tools and monitoring that other assemblers don't

offer. http://tempoautomation.com

Small Batch Assembly is a PCB assembler that trades off design flexibility for very low cost. http://smallbatchassembly.com

13.3 Contract Manufacturers

Aoyagi is a medium sized OEM with one factory in Shenzhen. They specialize in IoT and electronic toy products. They offer a quick turnaround, attention to detail, and flexible terms.
http://www.aoyagihk.com.hk

Approach Industries is a medium sized electronics manufacturer based in Shenzhen. They offer turnkey electronics assembly from the PCB level to injection molding and product assembly.
http://www.approach.com.hk

Evermuch is a Hong Kong and Shenzhen based manufacturer that is smaller, but nimble. They have a MFi license, which means they're authorized to manufacture Apple accessories.
http://www.evermuch.com.hk

Ryder Industries is a medium sized accessories manufacturer that makes a lot of gaming accessories and small IoT type products. http://www.ryderems.com

Shin Tech Group is a large manufacturer with six factories in Dongguan. They make a wide range of electronics products, small and large.

Tankya is a Shenzhen based cable and accessory maker. They can handle the entire design, tooling, and manufacturing process and are MFi licensed. http://tankya.com

Worthington Assembly is a US based electronics CM in Massachusetts. http://worthingtonassembly.com

13.4 Part Fabrication

Addifab specializes in small quantity injection molding for about the same cost as 3D printing. http://addifab.com

E-Make is great with painted CNC plastic and aluminum parts with an emphasis on cosmetic finishes. http://e-make.co

Fictiv provides 3D printing and CNC services with an instant online quoting system. http://fictiv.com

Focus Manufacturing is a US based metal fabricator and CNC shop. http://focusmanufacturing.com

Gener8 offers design and engineering services for small quantity production, up to 10,000 pieces. They also have experience with medical devices. http://gener8.net

Hanking is another Shenzhen based manufacturer with medium cost, great quality, and can meet medical grade requirements. http://hanking.cc

Junyi Metal and Plastic is a Shenzhen based manufacturer that can do CNC, injection molding, casting, and other fabrication techniques. They're cheap, fast, good quality, and easy to work with. http://www.junyimould.com

MicroConnex specializes in flex PCB and high-density designs. http://microconnex.com

Protolabs is a large rapid prototyping company that can make almost anything out of almost any material, in quantities up to 10,000 pieces. http://protolabs.com

ProtoTech is a high quality CNC shop based in Europe. http://prototech.no

Rubber Captain makes custom rubber and plastic parts in China, but is a US based company. http://rubbercaptain.com

Sculpteo offers inexpensive SLS 3D printing services. http://sculpteo.com

Shapeways can 3D print in a wide variety of materials, but lead times tend to be long. http://shapeways.com

Specialty Coating Systems can do conformal coating with perylene to waterproof your electronics. http://scscoatings.com

Star Rapid does rapid, small run prototype production. They tend to be more expensive than other options. http://www.starrapid.com

StrongD Model Technology are CNC experts, ship quickly, and work with mainly aluminum and plastics. Located in Shenzhen, China. http://www.cncmachinedprototypes.com

Trio Labs can 3D print objects as big as one square meter. http://trio-labs.com

Voodoo Manufacturing can 3D print parts in quantities from one to 10,000. Use them to get enclosures and parts in large volumes without paying for expensive injection molding tooling. http://voodoomfg.com

Xometry does rapid manufacturing in the form of CNC, 3D printing, sheet metal fabrication, injection molding, and urethane casting. http://xometry.com

13.5 Materials

McMaster-Carr carries nearly every mechanical thing or raw material you could possibly think of. They also have one of the best designed websites and user experiences I've ever seen. http://mcmaster.com

Tap Plastics has a huge variety of, well, plastics, as well as lots of molding materials. Great for putting together prototype enclosures. http://tapplastics.com

13.6 Paperwork

Cognition IP is a law office dedicated to IP protection and patents. http://www.cognitionip.com

Drip Capital can help you finance your imports and exports. http://dripcapital.com

Enzyme offers a quality management system and other tools to manage your FDA submission process. http://enzyme.com

Flexport is a freight forwarder and customs broker. They can help you get your product from your factory's loading dock to your customers front door. http://flexport.com

Flowspace can warehouse your product, deliver it, and perform other logistics tasks like repackaging and labeling. http://flow.space

Lumi lets you design, source, and order packaging of all kinds. http://lumi.com

Schox Patent Attorneys are one of the most respected law firms in Silicon Valley. Their lawyers are very technical (many have engineering degrees) and write great patents. http://www.schox.com

13.7 Shipping and Logistics

ImportGenius allows you to search ocean freight shipping manifests and find the CM of products you admire or of products that are

similar to yours. Starts at $99 per month. http://importgenius.com

Pakible lets you create custom packing materials like boxes, bags, and mailers. http://pakible.com

ShipBob is a shipping fulfillment company that lets you offer 2 day shipping. http://shipbob.com

Sourcify organizes your supply chain in one place and can help you find an overseas factory to source your product. http://sourcify.com

13.8 Design Services

APROE does great work building early-stage and mechanically complex prototypes. http://aproe.com

Model Solutions is a South Korean firm that creates very high quality, expensive looks-like models that are suitable for photo shoots. http://model-solution.com

Radicand specializes in product design, mechanical and electrical engineering, and embedded systems. They can help you make your product from scratch and are used to working with startups. http://radicand.com

13.9 Chip Fabrication

3D Printed Microtec specializes in 3D printed microstructures and IC packaging. http://3dprintedmicrotec.com

Bridg is an IC fabricator that caters to research groups and startups. http://gobridg.com

SiFive allows you to make custom SoCs based on RISC-V. http://sifive.com

Skywater can help you design and fabricate ASICS. http://skywatertechnology.com

13.10 Component Distributors

Digi-Key is one of the largest electronics distributors and is often the first stop for finding and buying electronics components. http://digikey.com

Mouser is another very large electronics distributor that carries some things that Digi-Key does not, like parts from Coilcraft and some test equipment. http://mouser.com

RFMW is a distributor that specializes in RF components. http://rfmw.com

Richardson RFPD is a distributor that also specializes in RF components. They have some overlap with RFMW, but they also represent manufacturers that RFMW does not. http://richardsonrfpd.com

13.11 Fulfillment

DCL is a major fulfillment center for several large hardware companies. http://dclcorp.com

DotCom is a very large fulfillment center that requires 500-1000 orders per day, but specializes in "white glove" service, which is high end packaging and material. http://dotcomdist.com

Floship is a B2C fulfillment company out of Hong Kong that especially likes working with crowdfunding projects. They can work with small quantities and are especially good for fulfillment in Asia and Australia. http://floship.com

Fulfillrite is a fulfillment company great for US east coast distribution. http://fulfillrite.com/

Newgistics is a major and well-respected fulfillment center, but you need to be shipping at least 5,000 units per month before they will want to engage. http://newgistics.com

RushOrder provides customer service, order taking, and fulfillment around the world, and has locations around the world. http://rushorder.com

Shipwire acts like a software layer on top of lots of fulfillment centers. They have an online tool for estimating shipping costs and are pretty easy to work with. The down side is that they're more expensive and it's harder for them to fix problems within the fulfillment center because they're one layer removed. http://shipwire.com

SimpleGlobal is a smaller fulfillment company that provides great service to young, growing companies. http://simpleglobal.com

About the Author

Hunter Scott is currently the director of hardware engineering at Reach Labs, a startup working on long range wireless power in Silicon Valley. He graduated from Georgia Tech with a degree in computer engineering. He cofounded two hardware startups, both of which went through Y Combinator. He has been designing electronics for over 10 years for a variety of applications, including art installations, medical devices, and communications products. His work has been featured in publications such as NPR, CNN, The Guardian, and The Chicago Tribune. He has spoken at conferences such as DEFCON, Altium Live, and Hackaday Supercon. His talks, articles, and projects can be seen at http://hscott.net. You can email him at hunter@designingelectronics.com.