EARLY ACCESS

the hardware hacker

ADVENTURES IN MAKING AND BREAKING HARDWARE

ANDREW "BUNNIE" HUANG

no starch press

the hardware hacker

Andrew “bunnie” Huang

Early Access edition, 9/23/16

ISBN-10: 1-59327-758-X
ISBN-13: 978-1-59327-758-1

Publisher: William Pollock
Production Editor: Alison Law
Developmental Editor: Jennifer Griffith-Delgado
Interior Design: Beth Middleworth
Copyeditor: Rachel Monaghan

contents

The chapters in **red** are included in this Early Access PDF.

part 1
adventures in manufacturing

I first set foot in China in November 2006. I had no idea what I was walking into. When I told my mother I was going to visit Shenzhen, she exclaimed, “Why are you going there? It’s just a fishing village!” She wasn’t wrong: Shenzhen was just a town of 300,000 back in 1980, but it had exploded into a mega-city of 10 million in 30 years. Between my first visit and the time I wrote this book, Shenzhen gained an estimated 4 million people—more than the population of Los Angeles.

In a way, my understanding of manufacturing over the years has mirrored Shenzhen’s growth. Before going to China, I had never mass-produced anything. I didn’t know anything about supply chains. I had no idea what “operations and logistics” meant. To me, it sounded like something out of a math or programming textbook.

Still, Steve Tomlin, my boss at the time, charged me with figuring out how to build a supply chain suitable for our hardware startup, Chumby. Sending a novice into China was a big risk, but my lack of preconceived notions was more of an asset than a liability. Back then, venture capitalists shunned hardware, and China was only for established companies looking to build hundreds of thousands of units of a given product. My first set of tours in China certainly supported that notion, as I primarily toured mega-factories serving the *Fortune* 500.

Chumby was lucky to be taken under the wing of PCH International as its first startup customer. At PCH, I was mentored by some of the finest engineers and supply chain specialists. I was also fortunate to be allowed to share my experiences on my blog, as Chumby was one of the world's first open hardware startups.

Although meeting the minimum order volumes of our conventional manufacturing partners was a constant struggle, I kept noticing small things that didn't square with conventional wisdom. Somehow, local Chinese companies were able to remix technology into boutique products. The so-called Shanzhai integrated cell phones into all kinds of whimsical forms, from cigarette lighters to ornamental golden Buddha statuettes. The niche nature of these products meant they had to be economical to produce in smaller volumes. I also noticed that somehow factories were able to rapidly produce bespoke adapter circuits and testing apparatuses of surprisingly high quality in single-unit volumes. I felt there was more to the ecosystem—a story that was being told over and over again—but few had the time to listen, and those who did heard only the parts they wanted to hear.

The financial crisis of 2008 changed everything. The consumer electronics market was crushed, and factories that were once too busy printing money were now swimming in excess capacity. I made friends at several medium-sized factories in the area. I started to inquire about how, exactly, these factories were able to so nimbly produce their internal test equipment, and how Shanzhai were able to prototype and build such bespoke phones.

The bosses and engineers were initially reticent, not because they wanted to hide potential competitive advantages from me, but because they were ashamed of their practices. Foreign clients were full of corporate process, documentation, and quality procedures, but they also paid dearly for such overhead. Local companies were much more informal and pragmatic. So what if a bin is labeled "scrap"? If the bits inside are suitable for a job, then use them!

I wanted in. As an engineer, tinkerer, and hacker, I cared a lot about the cost to produce a few units, and a couple of minor assembly defects was nothing compared to the design issues I had to debug. I eventually managed to coax a factory into letting me build a part using its low-quality but ultra-cheap assembly process.

The trick was to guarantee that I would pay for all the product, including defective units. Most customers refuse to pay for imperfect goods, forcing the factory to eat the cost of any part that isn't exactly to specification. Thus, factories strongly dissuade customers from using cheaper but low-quality processes.

Of course, my promise to pay for defective product meant there was no incentive for the factory to do a good job. It could have, in theory, just handed me a box of scrap parts and I'd still have had to pay for it. But in reality, nobody had such ill intentions; as long as everyone simply tried their best, they got it right about 80 percent of the time. Since small-volume production costs are dominated by setup and assembly, my bottom line was still better despite throwing away 20 percent of my parts, and I got parts in just a couple of days instead of a couple of weeks.

Having options to trade cost, schedule, and quality against each other changes everything. Since then, I've made it a point to discover more alternative production methods and continue shortening the path between ideas and products, with ever more options along the cost-schedule-quality spectrum.

After Chumby, I decided to remain unemployed, partly to give myself time for discovery. For example, every January, instead of going to the frenzied Consumer Electronics Show (CES) in Las Vegas, I rented a cheap apartment in Shenzhen and engaged in the “monastic study of manufacturing”; for the price of one night in Las Vegas, I lived in Shenzhen for a month. I deliberately picked neighborhoods with no English speakers, and forced myself to learn the language and customs to survive. (Although I’m ethnically Chinese, my parents prioritized accent-free fluency in English over learning Chinese.) I wandered the streets at night and observed the back alleys, trying to make sense of all the strange and wonderful things I saw going on during the daytime. Business continues in Shenzhen until the wee hours of the morning, but at a much slower pace. At night, I could make out lone agents acting out their interests and intentions.

If there’s one thing those studies taught me, it’s that I have a lot more to learn. The Pearl River Delta ecosystem is incomprehensibly vast. As with the Grand Canyon, simply hiking one trail from rim to base doesn’t mean you’ve seen it all. I have, however, picked up enough knowledge to build a full-custom laptop, and to develop a new process for peel-and-stick electronic circuits.

In this part of the book, you’ll follow my journey as I learned the Shenzhen ecosystem over the years, via a remix of blog posts that I wrote along the way. Some of the essays are reflections on particular aspects of Chinese culture; others are case studies of specific manufacturing practices. I conclude with a chapter called “The Factory Floor,” a set of summary recommendations for anyone considering outsourced manufacturing. If you’re in a hurry, you can skip all the background and go directly there.

However, hindsight is 20/20. Once you've walked a path, it's easy to point out the shortcuts and hazards along the way; it's even easier to forget all of the wrong turns and bad assumptions. There's no one-size-fits-all method for approaching China, and my hope is that by reading these stories, you can create your own (perhaps different) conclusions that better serve your unique needs.

1. made in china

Before my first visit to China, I was convinced that Akihabara in Tokyo was the go-to place for the latest electronics, knick-knacks, and components. That changed in January 2007, when I first set eyes on the SEG Electronics Market in Shenzhen. SEG is eight floors of all the components a hardware addict could ever want, and only later did I learn that it's just the tip of the Hua Qiang electronics district iceberg.

As the lead hardware engineer at Chumby at the time, I was in China with then-CEO Steve Tomlin to figure out how to make chumbys (an open source, Wi-Fi-enabled content delivery device) cheaply and on time. With prices like those at SEG, we were definitely in the right country to make at least the first part of that mission a success.

赛格电子市场
SEG ELECTRONICS MARKET
www.segem.com.cn
3/F 电脑及配件、品牌专卖
市场服务中心
市场维修中心
COMPUTER&COMPUTER ACCESSORIES
BRAN EXCLUSIVE SHOP
4/F 品牌专卖店 Bran Exclusive Shop
3/F 电脑及配件、品牌专卖 Computer&Computer Accessories Bran Exclusive Shop
2/F 电子元器件 Components
1/F 电子元器件 Components
8/F 电脑及配件 Computer&Computer Accessories
7/F 电脑及配件 Computer&Computer Accessories
6/F 电脑及配件 Computer&Computer Accessories
5/F 电脑及配件 Computer&Computer Accessories
天

Shenzhen's SEG Electronics Market, the new electronics Mecca. Akihabara, eat your heart out!

THE ULTIMATE ELECTRONIC COMPONENT FLEA MARKET

When I first stepped into the SEG building, I was assaulted by a whirlwind of electronic components: tapes and reels of resistors and capacitors, ICs of every type, inductors, relays, pogo pin test points, voltmeters, and trays of memory chips. As a total newcomer to manufacturing in volume, I was blown away by everything I saw at SEG.

All of those parts were crammed into tiny six-by-three-foot booths, each with a storekeeper poking away at a laptop. Some storekeepers played *Go*, and some counted parts. Some booths were true mom-and-pop shops, with mothers tending to babies and kids playing in the aisles.

Some family-run component shops

Other booths were professional setups with uniformed staff, and these worked like a bar—complete with stools—for electronic components.

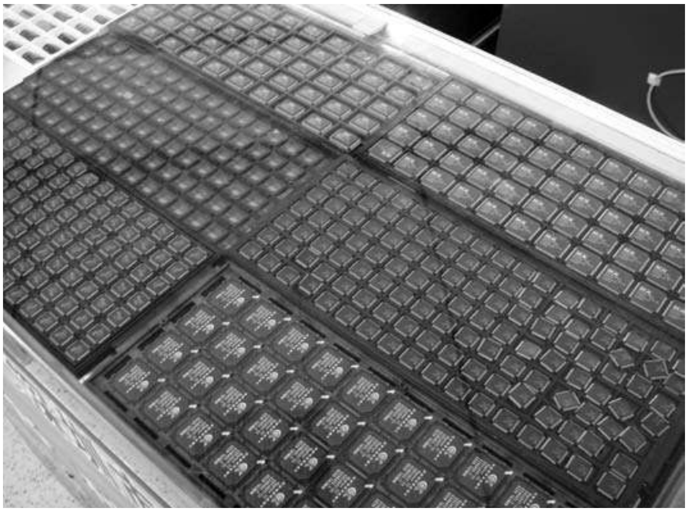
A swanky professional parts seller

No one at SEG said, "Oh, you can get 10 of these LEDs or a couple of these relays," like you might hear in Akihabara. No, no. These booths specialize, and if you see a component you like, you can usually buy several tubes, trays, or reels of it; you can get enough to go into production the next day.

Looking around the market, I saw a woman sorting stacks of 1GB mini-SD cards like poker chips. A man was putting sticks of 1GB Kingston memory into retail packages, and next to him, a girl was counting resistors.

The bottom-left corner of this display was packed with all kinds of SD cards.

Another booth had stacks of power supplies, varistors, batteries, and ROM programmers, and yet another had chips of every variety: Atmel, Intel, Broadcom, Samsung, Yamaha, Sony, AMD, Fujitsu, and more. Some chips were clearly ripped out of used equipment and remarked, some of them in brand new, laser-marked OEM packaging.



The sheer quantity of chips for sale at a single booth at SEG was incredible.

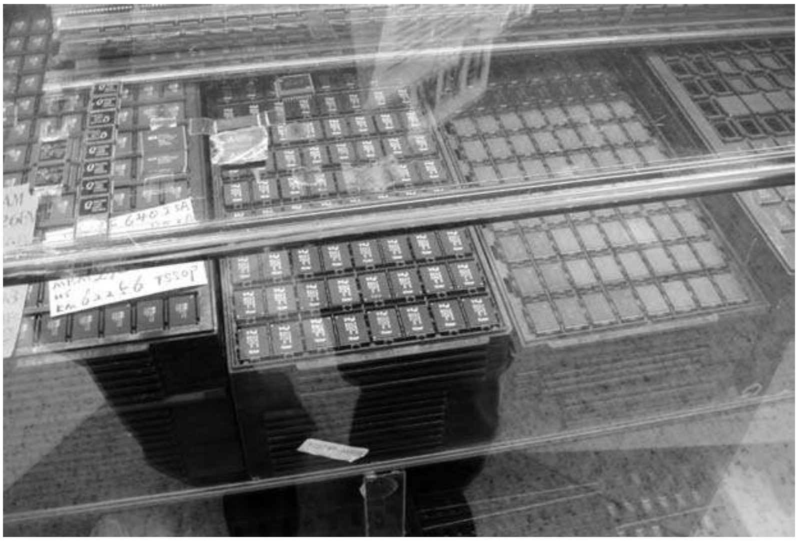
I saw chips that I could never buy in the US, reels of rare ceramic capacitors that I could only dream about at night. My senses tingled; my head spun. I couldn't suppress a smile of anticipation as I walked around the next corner to see shops stacked floor to ceiling with probably 100 million resistors and capacitors.

Reels and reels of components, in every shop window

Sony CCD and CMOS camera elements! I couldn't buy those in the US if I pulled teeth out of the sales reps. (Some sellers even have the datasheets behind the counter; always ask.) Next, I spotted a stack of Micrel regulator chips, followed by a Blackfin DSP chip for sale. Nearby, a lady counted 256Mb DRAM chips—trays of 108 components, stacked 20 high, in perhaps 10 rows.

The equivalent of Digi-Key's entire stock of DRAM chips sat right in front of me!

And across from her were a half-dozen more little shops packed with chips just like hers. At one shop, a man stood proudly over a tray of 4Gb NAND flash chips. All of this was available for a little haggling, a bit of cash, and a hasty goodbye.



A close look at a tray of 4Gb flash chips

And that was just the first two floors of SEG. There are six more floors of computer components, systems, laptops, motherboards, digital cameras, security cameras, thumb drives, mice, video cameras, high-end graphics cards, flat-panel displays, shredders, lamps, projectors—you name it. On weekends, "booth babes" dressed in outrageous Acer-branded glittery body suits loiter around, trying to pull you in to buy their wares. This market has all the energy of a year-round CES (Consumer Electronics Show) meets Computex, except instead of just showing off the latest technology, the point is getting you into these booths to buy that hardware. Trade shows always feel like a bit of a strip tease, with your breath making ghostly rings on the glass as you hover over the unobtainable wares underneath.

But SEG is no strip tease. It's the orgy of consumer and industrial electronic purchasing, where you can get your grubby paws on every piece of equipment for enough *kuai*[*] out of your wallet. Between the smell, the bustle, and the hustle, SEG is the ultimate electronic component flea market. It's as if Digi-Key went mad and let monkeys into its Minnesota warehouse, and the resulting chaos spilled into a flea market in China.

Of course, a lot of the parts I marveled at in 2007 are antiques now. For example, 4Gb flash chips are trash, and 1GB flash disks are old news. At the time, however, those things were a big deal, and SEG is still the best place to get the latest tech in bulk.

THE NEXT TECHNOLOGICAL REVOLUTION

Three blocks down the street from SEG lay the Shenzhen Bookstore.[†] The first and most visible rack was a foreign book section, packed with classics like Stanford University professor

**Colloquial word for yuan, the base counting unit for the renminbi (RMB), the currency in China.*

†This bookstore has closed since the visit I describe here.

Thomas Lee's *The Design of CMOS Radio-Frequency Integrated Circuits* and several titles by UCLA professor Behzad Razavi. I picked up Lee's book, and it cost 68 *kuai*, or $8.50 USD. Holy cow! Jin Au Kong's book on Maxwell's Equations? $5. Jin Au Kong *taught* me Maxwell's Equations at MIT.

I went on a spree, packing my bag with six or seven titles, probably around $700 worth of books if I'd bought them in America. At the checkout counter, I bought them for less than $35, complete with the supplemental CDs. That's equivalent to buying an economy class ticket to Hong Kong!

In China, knowledge is cheap. Components are cheap. The knowledge in the books at the Shenzhen Bookstore was the Real Deal, the parts to use that knowledge are down the street at SEG, and within an hour's drive north are probably 200 factories that can take any electronics idea and pump it out by the literal boatload. These are no backward factories, either. I saw with my own eyes name-brand, 1,550 nanometer, single-mode, long-haul, fiber-optic transceivers being built and tested there. Shenzhen is fertile ground, and you need to see it to understand it.

Shenzhen has the pregnant feel of the swapfests in Silicon Valley back in the '80s, when all the big companies were just being founded and starting up, except magnified by 25 years of progress in Moore's Law and the speed of information flow via the Internet. In this city of 12 million people, most are involved in tech or manufacturing, there's plenty of foreign influence, many are learning English, and all of them are willing to work hard.

There has to be a Jobs and Wozniak there somewhere, quietly building the next revolution. But I'm a part of Shenzhen, too, and I still tremble in my boots with terror and excitement at the thought of being part of that revolution. This is my story, starting with that eye-opening trip to Shenzhen for Chumby.

TOURING FACTORIES WITH CHUMBY

In September 2006, Chumby was just a team of about a half-dozen people, and we had just given away about 200 early prototype chumby devices at FOO Camp, a conference put on by Tim O'Reilly. The devices were well received by the FOO Camp attendees, so I got the go-ahead to build the Asian supply chain.

Steve and I went to China to visit potential factories in November, but before we left, we had a trusted vendor in the US give their best price for the job as a baseline for negotiations with the Chinese manufacturers. Then, we called up a lot of friends with experience in China and lined up about six factory tours. We hit quite a variety of places, from specialty factories as small as 500 people to mega-factories with over 40,000 people.

There's no substitute for going to China to tour a factory. Pictures can only tell the story framed by the photographer, and you can't get a sense of a facility's scale and quality without seeing it firsthand. In general, factories welcome you to take a tour, and I wouldn't work with one that didn't allow me to visit. However, most factories do appreciate a week's notice, although as your relationship with them progresses, things should become more open and transparent.

Speaking of openness, Chumby's open source nature helped the factory selection process a lot. First, we had no fears about people stealing our design (we were giving it away already), so we'd eliminated the friction of NDAs (non-disclosure agreements) when sharing critical information, like the bill of materials. I think this gave us a better reception with factories in China; they seemed more willing to open up to us because we were willing to open up to them. Second, there was no question in any factory's mind that this was a competitive situation. Anybody could and would quote and bid on our job (in fact, we received a few unsolicited quotations that were quite competitive), so it saved a round of huffing and puffing.

After reviewing several manufacturing options, Steve and I eventually decided to work with a company called PCH China Solutions. PCH itself owns only a few facilities, but it has a comprehensive network of trusted and validated vendors, primarily in China but also Europe and the United States. Not surprisingly, the factories that PCH subcontracts to were some of the best facilities we visited in China. PCH is actually headquartered out of Ireland, and thus most of their staff engineers are Irish, so there was also no language barrier. (PCH engineers are also hardworking, resourceful, and well trained—and, as a bonus, they always seem to know the best place to find a pint, no matter where they are. I had no idea China had so many Guinness taps!)

There's a lot to take in when you tour even one factory, let alone a half-dozen, and it's easy to get overwhelmed and lost in the vagaries of electronics manufacturing. But there were some key details I found most fascinating during my factory tours for Chumby, and in working with PCH to bring the chumby to life.

Scale in Shenzhen

One stunning thing about working in China is the sheer scale of the place. I haven't been to an auto plant in Michigan, or to the Boeing plant in Seattle, but I get the sense that Shenzhen gives both a run for their money in terms of scale. In 2007, Shenzhen had 9 million people, and while in general, China has more males than females, Shenzhen seems to have all the women (the ratio of women to men is about 7:1). Once you see the gender ratio of a major factory, you'll understand why.

To give you an idea of the scale of a Shenzhen factory, the New Balance factory there employed 40,000 people and had the capacity to produce over a million shoes a month. I estimate that from raw fabric to finished shoe, the process took about 50 minutes, and every perfectly stitched bundle of plastic and leather was sewn by hand on an industrial sewing machine. The stations are designed so that each stage in the process takes a worker about 30 seconds.

Of course, the New Balance factory is dwarfed by Foxconn, the factory where iPods and iPhones are made.

You know you're big when you have your own exit off the freeway.

Foxconn is a huge facility, apparently with over 250,000 employees, and it has its own special free trade status. The entire facility is walled off, and I've heard you need to have your passport and clear customs to get into the facility. That's just short of the nuclear-powered robotic dogs from the nation-corporation franchulates of Neal Stephenson's *Snow Crash*.

Feeding the Factory

There's an old Chinese saying *min yi shi wei tian*. A literal translation would be "people consider food divine," or "for people, food is next to heaven." You can also look at it as a piece of governing advice: "the government's mandate [synonymous with heaven] is only as robust as the food on people's plates." Or, you can interpret it as an excuse to procrastinate: "let's eat first [since it is as important as heaven]."

Whichever way you cut it, I think the saying still holds in China. One important metric for gauging how well a factory treats its employees is how good the food is, as it's common for factory workers to be housed, fed, and cared for on site.

The food is actually quite good at some factories. For example, when eating with the workers at the factory that manufactured chumby circuit boards, I was served a mix of steamed fish, broiled pork, egg rolls, clean fried vegetables, and some pickled-vegetable-and-meat combo. Rice, soup, and apples were also provided in "help yourself" quantities.

A meal from the factory that made the chumby circuit boards

Every facility I visited also had separate utensils and plates for guests. At one factory, my food was served on a Styrofoam plate with disposable chopsticks, while a factory worker I ate with was served food on a steel plate with steel chopsticks. I hadn't passed the factory's physical examination, so they gave me disposable eating tools to prevent me from contaminating the factory with potential foreign diseases.

Going back to scale, some factory food operations are impressively large. I heard that Foxconn's workers consume 3,000 pigs a day. From pigs to iPhones, it all happens right here in Shenzhen!

A truckload of pigs, exiting the highway toward Foxconn

Dedication to Quality

After I started working with PCH on actually manufacturing the chumby, I ran into a situation sometime around June 2007 that showed me just how dedicated the factory workers in Shenzhen are to getting their jobs right.

I had updated the chumby motherboard to include an electret microphone, with an integral pre-amp field-effect transistor (FET). The microphone needed to be inserted in the correct orientation with respect to the circuit, so the FET would receive a proper bias current.

The first samples I got back from PCH's factory had the microphone in backward, and I called the factory to tell them to reverse its polarity. I was going to visit the factory the next week, and I wanted to see corrected samples. When I arrived and tested the microphone, I found to my dismay that the microphones were *still* not working.

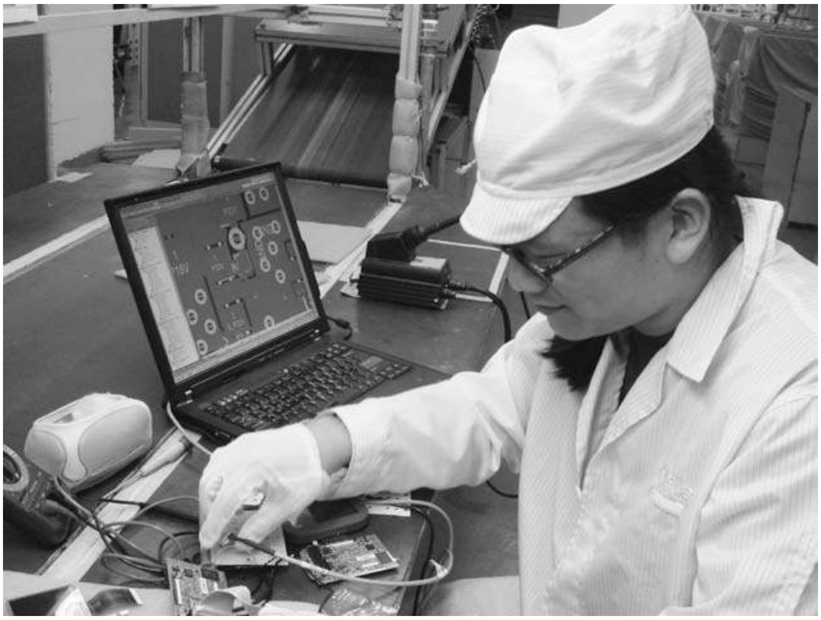
How could that be? There are only two ways to connect a microphone.

It turns out there were two operators on the line assembling the microphone. One soldered the red and black wires to the microphone. The next soldered these red and black wires to the circuit board. The operators were told to reverse the order, and both of them dutifully complied—giving me a microphone that was still soldered in backward, but with the color of the wires swapped. (This is actually a pretty typical story for problems in China.)

The factory was scheduled to manufacture a first pilot run of 450 circuit boards the next day. Everything had to go perfectly for Chumby's production timeline to stay on schedule. We had soldering stencils rebuilt (we were debugging a yield issue with the QFN packaged audio CODEC as well) and ready by around noon, and by around 6 PM, I had the first boards in my hands to test. I ran the final factory test, and the device failed again—at the microphone. This was not a happy moment for anybody in the factory, as the factory is liable for any manufacturing defects.

I donned my smock and marched onto the line to start debugging the problem.

For the rest of the night, I remained in the factory, and so did every manager and tech involved in manufacturing the chumby. The pressure was enormous: right next to us was a line churning out 450 potentially defective circuit boards, and I was unwilling to pull the plug because I still didn't know what the root cause was, and we had to stay on schedule.



I was debugging circuits at 3 am on the day of the final factory test for chumby.

I literally had a panel of factory workers standing by the entire night to bring me anything I needed: soldering irons, test equipment, more boards, X-ray machines, microscopes. Remarkably, not a single person hesitated; not a single person complained; not a single person lost focus on the problem. People canceled dinner plans with friends without batting an eyelash. Anyone who wasn't needed in a particular moment was busy overseeing other aspects of the project. I hadn't seen blind dedication like that since I worked with the autonomous underwater robotics team at MIT.

And this went on until 3 AM.

Embarrassingly, the problem wasn't PCH's fault in the end. The problem was the new firmware release I received earlier that day from the team in the US. It had a bug that disabled the microphone due to a hack that was accidentally checked into the build tree.

Even more impressively, when PCH found out, nobody was angry, and nobody complained. (Well, the saleswoman gave me a hard time, but I deserved it; she had been kind enough to accompany me on the production line all night long and be my translator, since my Mandarin wasn't up to snuff.) They were simply relieved that it wasn't their fault.

We all parted ways and I came back into the factory the next day at 11 AM after a good night's sleep. I saw Christy, the factory's project manager for manufacturing the chumby boards. I asked her when she came in to work, and she told me she always has to report by 8 AM. I started to feel really bad; Christy stayed up late because of our bug, and she came in early while I slept in. I asked her why she stayed up so late even though she knew she had to report to work at 8 AM. She could have gone home, and we could have continued the next day.

She just smiled and said, "It's my job to make sure this gets done, and I want to do a good job."

Building Technology Without Using It

Here's another interesting story. On our way out of the factory floor one day, Xiao Li (the quality assurance manager at the factory where we made chumby) asked me, "What does a chumby do?" I didn't speak Chinese very well, and she didn't speak English very well either, so I decided to start with a few basic questions.

I asked her if she knew what the World Wide Web was. She said no.

I asked her if she knew what the Internet was. She said no.

I was stunned, and I didn't know what to say. How do you describe the color blue to the blind?

Xiao Li was an expert in building and testing computers. On some projects, she probably built PCs and booted Windows XP a hundred thousand times over and over again.

(God knows I heard that darn startup sound a zillion times during the microphone incident, as there was a bank of final test stations for ASUS motherboards right next to me.) But she didn't know what the Internet was.

I had assumed that if you touched a computer, you were also blessed by the bounties of the Internet. All at once, I felt like a spoiled snob and a pig for forgetting that Xiao Li probably couldn't afford a computer, much less broadband Internet access. Given the opportunity, she was certainly smart enough to learn it all, but she was too busy making money that she probably sent back home to her family.

In the end, the best I could do was to tell Xiao Li that the chumby was a device for playing games.

Skilled Workers

Shenzhen workers may not know a lot about everything they make, but on top of their dedication, they are highly skilled. I once watched a guy working at the same factory that sewed the chumby bags, and I swear, he could sew cosmetics cases together at a rate of 5 seconds per bag. And he wasn't even 100 percent focused on his task; he was listening to his iPod while he sewed.

And apparently, he wasn't their fastest employee! They had someone about twice as fast, and he'd been with the company for about seven years. I went to watch the faster worker, but he had already gone to lunch because he'd finished everything; there were two enormous bins of finished cosmetics cases next to his workstation.

On a similar note, I was amazed to learn how rubberized tags (the ones you see all over clothes) are made in China. I always thought they were pressed by a machine, but I was wrong. All those words, colors, and letters are drawn by hand. Someone just places a logo stencil over the blank tag, paints over the stencil with amazing precision, and moves on to the next tag in their queue. When there are multiple colors, there's a person for each color, to keep the process quick.

I asked PCH if they had any mechanized factories for stuff like that. They told me the facilities exist, but the minimum order quantity is enormous (hundreds of thousands, sometimes millions) because of the extraordinarily low cost of the product and the relatively high cost of tooling for the automated process. This is consistent with what I've heard about McDonald's Happy Meal toys. They're usually held together with screws because it's cheaper to pay someone to screw together a toy over the whole production run than it is to make a steel injection-molding tool with the tolerances necessary for snapping the toys together.*

There was a similar trade-off inside the chumby hardware. There were four connectors on the internal chumby electronics. Using the US-based vendors that I could source, one connector had a best price of about $1 USD, and the other three had a best price of about $0.40 each. PCH's very talented sourcing expert (her reputation was feared and respected by every vendor) managed to find me connectors that cost $0.10 and $0.06, respectively, saving almost a full $2 in cost. There's one catch: the connectors lacked the sacrificial plastic pick-and-place pad that would enable them to be machine-assembled.

The solution? A person, of course.

*. *Due to high wage inflation since this particular visit, this is probably no longer true.*

This man hand-placed the cheaper connectors on every chumby, for about a nickel per unit. Thanks to him, chumbys were $2 cheaper, which freed up more money for us consumers to spend at Starbucks.

The Need for Craftspeople

I'd like to introduce you to a man I know simply as Master Chao. I met the Master during the chumby manufacturing process, and I'm pretty sure that in your lifetime, you have used or seen something that he created.

When I went to the sample room for the factory where Master Chao worked, I was shocked at how many items on their shelf I had purchased, used, or seen in a store in the US myself. Top-tier consumer brands manufacture their stuff in this factory, and to the best of my knowledge, the factory had just one master pattern maker at the time: Master Chao. He's had a hand in creating cosmetic bags for Braun, accessory cases for Microsoft, and the medical braces for major brands sold in drugstores, among many other products.

Master Chao is the person in the foreground; in the background is Joe Perrott, chumby's excellent project engineer from PCH China Solutions.

The Master is a craftsman in the traditional sense. It used to be that the finest furniture was designed and built only with the intuition and skill of a master craftsman. Now, we all go to Ikea and get CAD-designed, supply-chain-managed, picture-book-assembly furniture kits—and despite all that it doesn't look too shabby. As a result, the word *craft* has been relegated to describe some scrapbook or needlepoint kit you buy at Michaels and put together on a slow weekend. We've forgotten that in an age before machines, "craft" was the only way anything of any quality was built.

It turns out, however, that traditional craft still matters, because CAD tools haven't brought about the ability to simulate our mistakes before we build them.

The creation of a *flat pattern* for textile goods is a good example of a process that requires a craftsman. A flat pattern is the set of 2D shapes used to guide the cutting of fabrics. These shapes are cut, folded, and sewn into a complex 3D shape. Mapping the projection of an arbitrary 3D shape onto a 2D surface with minimal waste area between the pieces is hard enough. The fact that the material stretches and distorts, sometimes in different directions, and that sewing requires ample tolerances for good yields, makes pattern creation a difficult problem to automate.

The chumby cases added another level of complexity, because they involved sewing a piece of leather onto a soft plastic frame. In that situation, as you sew the leather on, the frame distorts slightly and stretches the leather out, creating a sewing bias dependent upon the direction and rate of sewing. This force is captured in the seams and contributes to the final shape of the case. I challenge someone to make a computer simulation tool that can accurately capture those forces and predict how a product like that will look when sewn together.

Yet, somehow, Master Chao's proficiency in the art of pattern making enabled him to very quickly, and in very few iterations, create and tweak a pattern that compensated for all of those forces. His results, all obtained with cardboard, scissors, and pencils, were astoundingly clever and insightful. Be grateful for his old-world skills; they've likely played a role in the production of something you've used or benefited from.

There's not a single computer in Master Chao's office, yet the products I saw here wrapped around a wide array of high-tech devices.

Automation for Electronics Assembly

Before my work at Chumby, I thought almost everything was made by a machine. Of course, the tours of the textile factories corrected my impression very quickly; yet, high-tech stuff like electronics assembly does still tend to be heavily automated, even in China. The only exceptions I saw during my factory tours were, ironically, the lowest-cost products, such as toys. These shops were still dominated by lines of workers, stuffing and dip-soldering circuit boards by hand.

One interesting dichotomy related to automation is the bimodal distribution of products that use *chip-on-board (CoB)* technology. CoB assembly directly bonds a silicon die to a PCB. Finished CoB assemblies have the distinctive "glob of epoxy" look to them, as opposed to the finished plastic-package look. High-end, dense electronics assemblies often employ CoB technologies. I've done a couple of CoB designs for some 10 Gb optical transceivers in my time, and they were not cheap.

At the same time, however, almost all toys use CoB technology, to eliminate the cost of the IC package! It's a testament to toy factories' tenacity about cost reduction that they would buy an automated wire bonder and stick it next to lines molding doll heads and sewing up stuffed animals because having an in-house wire bonder saves a nickel.

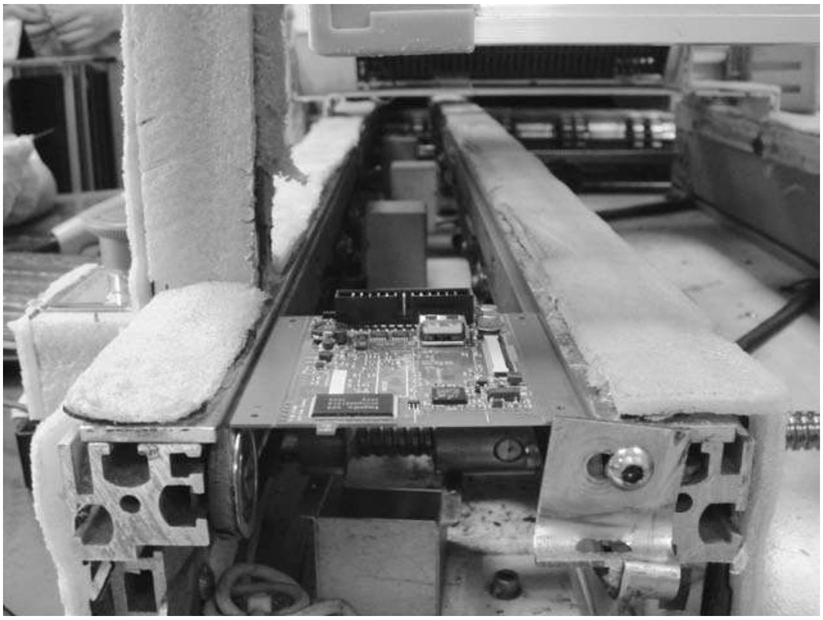
A typical wire bonder bonds a wire as thin as a human hair to a site on a silicon chip not much larger than the wire diameter, and it does this several times a second. Wire bonders are very fast, precise pieces of equipment. The bonding happens so quickly that the board seems to swivel smoothly around, but in fact, it stops 16 times as it spins around, and at each stop, a wire is bonded between the chip and the board.

Immediately before bonding, however, the chip is glued very carefully to the board by hand, and immediately after bonding, the chip is encapsulated by a human operator dispensing epoxy very carefully by hand. That means wire bonder is the only automated piece of equipment on assembly lines for simple toys. Seeing that process gave me a new appreciation for what goes into those talking Barney dolls that sell for $10 at Target.

The chumby manufacturing process used a bit of automation, too, courtesy of a chip shooter. Chip shooters (as well as pick-and-place machines) place surface-mount components on PCBs so the components can be soldered.

The chumby PCB assembly factory in China had dozens of lines filled with tried-and-true Fuji chip shooters.

It's absolutely mesmerizing to see a chip shooter in action. The chip shooters at the chumby PCB assembly factory were capable of placing 10,000–20,000 components per hour, per machine. This means that each machine could put down 3–6 components per second. The robotic assemblies move faster than the eye can see, and it all turns into an awe-inspiring blur. The chip shooter I saw at the chumby factory worked something like a Gatling gun: the chip gun itself was fixed, and the board danced around beneath the gun. The chip shooter actually "looked at" each component and rotated it to the correct orientation before putting it down on the board.


This is the end of the line for a chumby core board assembly!

The factory we used for chumby's PCB assembly also produced name-brand PC motherboards, and seemed to have no problem pushing out well over 10,000 such complex assemblies each day. But even though processes like component placement can be automated, there are some things a machine just can't do.

Precision, Injection Molding, and Patience

In the course of engineering the chumby, I also had to learn about injection molding, because the circuit board had to go inside a case of some kind. For an electronics guy with little mechanical background, this was no small hill to climb. The concept seems simple: you make a cavity out of steel, push molten plastic into it at high pressure, let it cool, and voilà—a finished part comes out, just like the Play-Doh molds from elementary school.

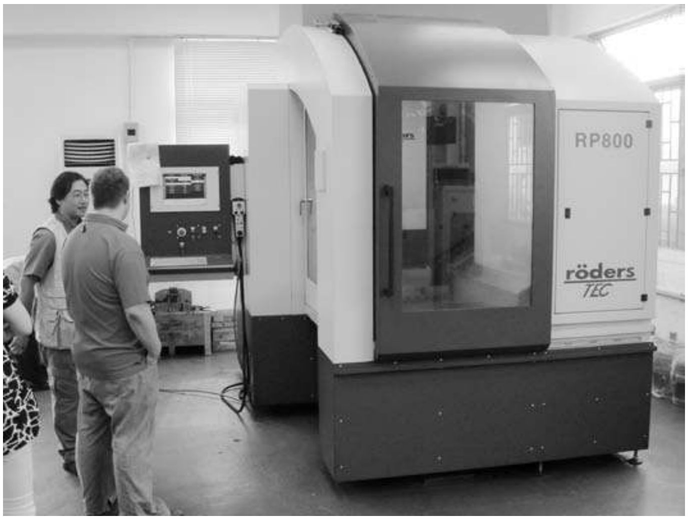
Oh, if only the process were that simple.

Sure, plastic flows, but it's not particularly runny. It moves slowly, and it cools as it flows. The color of the plastic is impacted by the temperature changes, and when using an improperly designed mold, you can even see flow lines and knit lines in the final product. There's also a whole assortment of issues with how the finished part is pulled from the mold, how the mold is made and finished, where the gates and runners are for getting the plastic inside the mold, and so on.

Fortunately, PCH had experts in China who knew all about this, and I got to learn mostly by watching.

If I were to summarize injection molding with a single adjective, it would be *precision*. When done right, the molds are precise to better than hair-thin tolerances, yet they are made out of hard steel. Achieving this level of precision out of such a durable material is no mean feat, and it's impressive to see a machine cut a mold out of raw steel.

The machine that cut the molds for the chumby case had a moving stage that rapidly pushed around a block of steel probably weighing several hundred pounds; it milled away at the metal in quite a hurry!



The mold-cutting machine used in manufacturing chumbys. Compare it to the people standing next to it for scale.

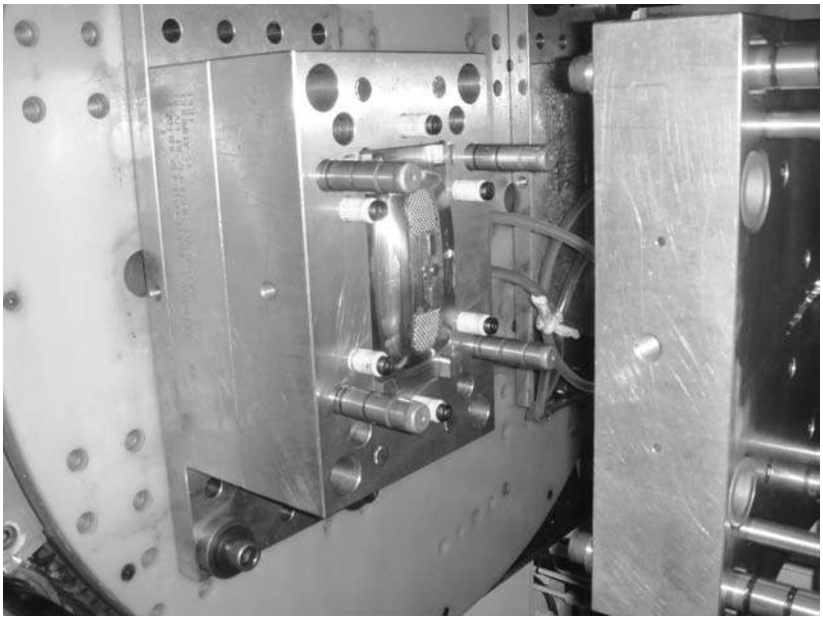
But machining is only the roughest step in mold making. After the rough shape is cut out, the mold is put into an *electrical discharge machine (EDM)*, where a burst of electrons knocks microscopic chunks off the steel surface. This is a terrifically tedious process: I've watched many EDMs do their job, and it's like watching paint dry. EDMs are, however, wicked precise, and they yield spectacular, repeatable results.

From a project management standpoint, the phenomenally long lead times of production-quality injection-molded plastics was the biggest eye opener for me. All told, the chumby mold transformed from a block of raw steel into a first-shot tool in four to six weeks, and I had to go to China and see the tooling shop do its work before I was convinced there wasn't some gross amount of schedule padding.

Even more harrowing from the risk management standpoint was the lack of good simulation tools to predict how plastics will flow through a mold. If we saw visible blemishes like flow lines and knit lines, we had to wait four to six weeks to see if the new mold was better. Ouch!

Fortunately, the toolmakers Chumby used in China anticipated these issues, and they made the tools to err on the side of excess steel, because removing material to fix a problem is much easier than adding material. It's like the old carpenter's saying: measure twice, cut once, and if you have to cut wrong, cut long.

The mold that was used to create chumby's back bezel was extra complex, because it involved a process called *overmolding*. If you happen to own a chumby classic, look at the back side. There's a rubbery TPE surrounding the hard ABS bezel. Many people assumed this was a glued-on rubber band. In fact, the TPE is molded in place on the back piece. This requires a two-shot mold.



The final mold for chumby's back bezel, inside an injection-molding machine

There were actually two molds, and one side of the mold spun around so that the alternating material systems could be molded at the right points in the process.

A lot of hard work goes into the humble plastic parts you see every day, and that's all part of creating quality products. But at the same time, there's also a very real need to meet the expectation of cheap prices.

The Challenge of Quality

Clearly, with the expectation of low cost of China-made goods comes a great challenge in quality management. Look at the media coverage on topics like lead paint in toys, industrial chemicals in food, and other items made in China, and you can see some of the bad decisions made to keep prices down.

When considering cases like that, I think it's important to apply Hanlon's Razor. To paraphrase, "Never attribute to malice that which can be adequately explained by ignorance."

The Brits also have a nice, pithy version of the aphorism: “Cock-up before conspiracy.”

Some manufacturers are indeed out there to make a buck at any cost, but I think the majority of mistakes are made out of ignorance. Most of the rank-and-file in factories don’t know what their product is ultimately used for, and under intense pressure to reduce costs, they make those bad decisions. Factories also have to deal with products that are woefully underspecified, as well as customers who overwhelm them with all kinds of frivolous requirements—and most customers don’t follow up in either case. In the end, the factories play a game of “ship and find out,” and if the customer doesn’t notice a missing spec, then the spec must not have been important. It’s not a great game, and it means that customers need to be ever vigilant about audits and keeping the quality standard up.

THE DISCONNECT BETWEEN AMERICA AND CHINA

One fundamental problem behind this game is that many Chinese do not understand or appreciate basic things that we take for granted in America, and vice versa. Many Chinese factory workers are well educated, but they didn’t grow up in a “gadget culture” like we have in the US, so you can’t assume anything about their abilities to subjectively interpret specifications for a product.

For example, you can tell a US engineer, “I’d like a button on that panel,” and you’ll probably get something pretty close to what you expect in terms of look and feel, since you and the engineer share common experiences and expectations for a button on a panel. If you did the same in China, you’d probably get something that looks a little awkward and has a clunky feel, but is darn cheap and really easy to build and test. While the latter properties are desirable for practical reasons, American gadget connoisseurs just won’t buy something that’s aesthetically awkward or feels clunky.

Yet, ultimately, it's those consumers who want—nay, demand—low-priced goods, and that need drives the decision to manufacture in China. The trouble is that aside from the label on the product that says "Made in China" or "Made in the USA," consumers really don't care about the manufacturing process. What markup would you pay for a gadget that said "Made in the USA" on it? The cost premium for US labor is 10x what it is in China. Think about it: can the average US factory worker be 10x more productive than the average Chinese factory worker? It's a hard multiplier to play against.

I'm not saying there's no value in domestic vendors: it would be a lot less effort and less risk for me to get stuff made in the US. In fact, most early prototypes are made in the US because of the enormous value that the domestic vendors can add. However, the pricing just doesn't work out for a mass-market product. Nobody would buy it because its price wouldn't justify its feature set. One could even accuse me of being lazy if I were to just stick with a domestic vendor and pass the higher cost on to the customers.

BEING INVOLVED IN THE MANUFACTURING PROCESS

In the end, manufacturing in China is the best way to keep costs down, and to maintain quality, there is no substitute for going to China and getting directly involved. Almost every factory will "clean up" the day you come to visit, but with a sharp eye and the right questions, you can see through any quick veneers put in place.

When I evaluated factories for Chumby, I always visited the quality control (QC) room. I expected to see rows of well-maintained and well-worn binders with design documentation and QC standards, as well as *golden samples*, which are pre-production samples of a product. I'd demand to see the contents of a random binder and the golden sample associated with it, and verify that the employees knew what was going on in the binder. (Some factories do fill product binders with random data.) I also considered hard investments in equipment a good sign: the best manufacturers I visited all had a couple of rooms with sophisticated equipment for thermal, mechanical, and electrical limit testing, and of course, operators were in the room actually using the equipment. (I could definitely imagine a Chinese manufacturer buying a room of equipment just for show.)

But I suspect that toy manufacturers and food manufacturers don't fly technicians like me out to factories in China to oversee things on a regular basis. Contrast that with Apple, which regularly sends a cadre of engineers to work intense two-week (or longer) shifts in the factories (usually Foxconn, affectionately nicknamed "Mordor" by some at Apple). As a result, I bumped into many Apple engineers at the expat bars in Shenzhen.

Western-style management and quality control based in China is one of the important services that PCH China Solutions offered us at Chumby. If we had a problem with a vendor, PCH sent someone to the factory right away to see what was going on—no phone tag, no FedEx filibuster. And factory owners in China tend to be very responsive when you show up at their doorstep.

Thus, Chumby's approach to the quality conundrum was holistic. We started by having an engineer (me) at the factory almost on day one to survey the situation. It's important to learn what the factory can and cannot do. I looked at what was being built on the line, and the techniques used. Then, when it came time to engineer the product, I tried to use the processes and techniques that were most comfortable for the factory. When I had to do something new (and any good, innovative product will need to), I picked my battles and focused on them, because anything new would be a multiweek challenge to get right. This strategy applies to even the smallest details: if the factory shrink-wraps goods in plastic, and you want to wrap your product in paper, then plan to focus heavily on developing the paper-wrapping process, because it's quite possible that none of the line workers at your factory of choice have even seen a paper-wrapped product before.

Of course, when developing a new process for chumby, I preferred to be in the factory, and I still do. There's nothing like standing on the line and showing the workers who will be building your device how it should be made. For example, I personally trained the chumby assembly-line workers on how to attach a piece of copper tape to the LCD assembly to form a proper EMI shield.

It's difficult to describe the intricacies of how to fold tape across a complex piece of sheet metal to ensure it makes good electrical contact to the grounding surfaces without risking a short-circuit to other components. Subtleties like the fact that the adhesive on one side is a poor insulator also require a basic understanding of physics that line workers simply don't have. Worse yet, explaining these concepts requires technical words that your translator might not even know.

In my case, even a good 3D drawing or photograph of the finished assembly couldn't have gotten the whole concept across, because the stiffness of the tape required a particular motion to fold without tearing. Describing the process remotely, approving samples via photographs, and ultimately approving a unit delivered via FedEx might have taken a couple of weeks, but standing in front of a group of workers and demonstrating the process firsthand took only a few minutes. And despite the language barrier, I could tell from their facial expressions and body language whether they understood the importance of a particular step. Given those cues, I immediately reviewed processes that were ambiguous or difficult to master.

Typically, when you can demonstrate a process at this level of detail and intimacy, the workers will get it right within hours, instead of weeks. This is part of the reason I spent so much time in China during the development of chumby's manufacturing process.

Everyone was involved in the chumby quality process. This photo shows CEO Steve Tomlin (far left) and Artistic Director Susan Kare (middle) at the sewing factory, working out the details of logo silk-screening.

HOME-GROWN REMOTE TESTING

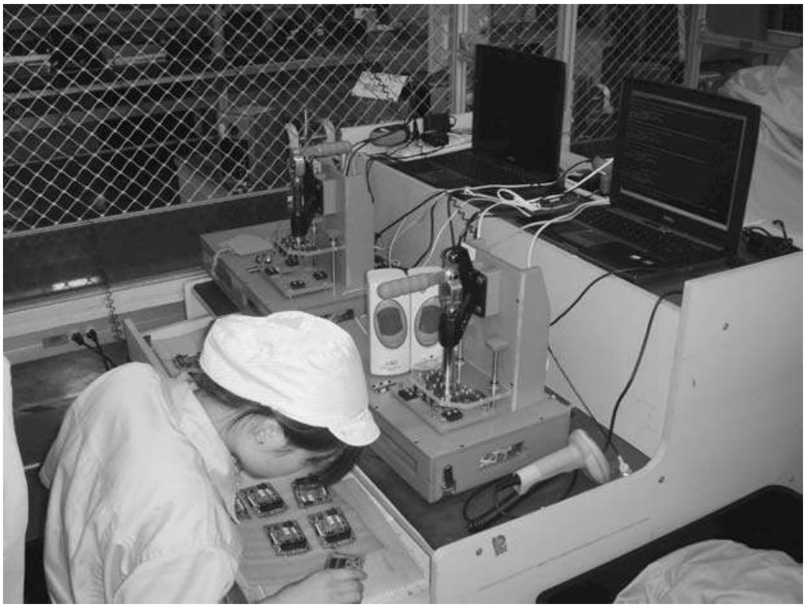
However, it wasn't always possible for Chumby to send someone to China. I, for one, preferred not to live in China, so at Chumby, we relied a lot on PCH to watch the quality and make sure things went well, and they did a superb job.

Often, working long distance meant that new processes took weeks to phase in if I wasn't there to tweak and approve on the spot, because every single tweak involved sending something almost round-trip through FedEx. After going through that process a few times, I learned to allocate two weeks per tweak, as opposed to the few hours it took when I was on the factory floor.

Those sets of two weeks added up fast.

Given the difficulty of overseeing operations in China from the US, remote electronic monitoring of the products' test results was essential. For chumby, I developed a set of testers that programmed, personalized, booted, verified, and measured every device off the assembly line. All data from the testing process was recorded to a log, and at the end of the day, the log was transferred to a server in the US.

This data let me debug a plethora of problems on the floor. I could tell if an operator at a particular tester was having trouble with their barcode scanner. I also knew immediately if there was a yield problem that day, or if the throughput was slower than expected. It was very powerful to have this home-grown audit capability in place, because the factory knew I was watching them. In fact, having such a capability in place can make relationships with the factory run better: the factory eats the cost of yield problems (at least initially), so they appreciate it when the design engineer can offer expedient advice and help before any problems get out of hand.



A pair of chumby test stations in the factory in China. There's quite a story about the trouble we went through getting those laptops into China.

FURTHER FACTORY TESTING

Once you've finished setting up the testing process, it can run autonomously at the factory. For example, at chumby's PCB factory, the first pass of final inspection was done manually—one person went over every circuit board, and then with the help of a cardboard template, another operator ensured that no components were missing. The units then went on to automated testing.

Periodically, both PCH and the factory also performed Restriction of Hazardous Substances (RoHS) testing on chumby units to ensure that there was no contamination with a specified set of potentially harmful chemicals, including lead. RoHS is a hazardous chemical safety standard required in Europe, but ironically not in the US. Factories do this test routinely on all products, even those only shipping to the US, because latent contamination on the line could prevent other products manufactured on the same line from shipping to Europe.

Even after all that testing, back in the US, chumby continued to sample units for QC purposes. To this end, we ordered devices regularly, characterized them, and dissected them to ensure that all the operating procedures were being followed.

MISTAKES STILL HAPPEN

Despite such safeguards, some mistakes will be made on any product. Every product goes through a phase where bugs that weren't caught by internal QA get pounded out. You have to rely on a top-notch customer service and support team, and you have to plan on being very agile and innovative during this phase to solve the problems and prevent them from ever happening again.

When I was at Chumby, if I heard about a unit in the wild with hardware problems, I actually called the customer who reported it. I wanted to know what went wrong so I could fix the problem and make sure it never happened again, to anyone!

My biggest hope with chumby, however, was to avoid what happened to Microsoft and the Xbox 360's "red ring of death," where consoles would experience a major hardware failure, stop working, and just display a red light around the power button, causing huge frustration for players. This problem only exhibited itself after the Xbox 360 had been out for years, after millions of units had been shipped. Situations like the red ring of death are a product engineer's worst nightmare.

So you see, getting the chumby (or any product) to the point where it can ship to consumers is just the beginning. The real challenge starts after.

If you ever find yourself at this point in the manufacturing process, I wish you luck!

CLOSING THOUGHTS

The stories told here share some of my adventures—and failures—learning how to build products in volume. The next two chapters are more reflective and less narrative. The next chapter takes us on a virtual tour of three factories to see what we can learn from them, and Chapter 3 attempts to summarize all the lessons I've learned about manufacturing so far.

2. inside three very different factories

It’s hard to understand how a computer works without opening it and looking around inside. Likewise, it’s hard to understand how products are made without going into a factory and touring the line. Although we often think of manufacturing as the necessary but boring step after innovation, in reality, the two are tightly coupled. An inventor thinks about a product once; a factory thinks about the same product day in and day out, sometimes for years on end.

The importance of factories as an innovation node is only growing in today’s connected global economy. The reality is that there is no “Apple factory” or “Nike factory.” Rather, there is a series of facilities that are domain experts in processes (such as PCB fabrication or zipper manufacturing) that are curated by the familiar brands. Thus, it's not uncommon to see two competitors' products running side-by-side down similar lines in a single facility. This concentration of domain-specific expertise means that the best place to learn how to make an aspect of your product better is often the same place that makes a similar aspect in everybody else's products.

Some of the greatest insights I've had into improving a product have come from observing technicians at work on a line and seeing the clever optimization tricks they've developed after doing the same thing over and over for so long.

This chapter takes you on a tour of three factories that make everyday things: PCBs (in particular, the ones used in the Arduino), USB memory sticks, and zippers. By peeling back the curtain, you'll get some insight into the design trade-offs behind the products, and how they can be made better. In the PCB factory, I discovered the secret of how they print a high-resolution map of Italy on the back of every Arduino; in the USB memory stick factory, I witnessed a strange marriage of high- and low-tech manufacturing techniques; and in the zipper factory, I found out how even the humblest of products can bear valuable lessons for product designers.

WHERE ARDUINOS ARE BORN

It was July 2012, and it had been about six months since my previous startup, Chumby, ceased operations. I had decided to take a year off to figure things out and cross a few items off the bucket list, one of which was a trip to Italy. My girlfriend had the bright idea of reaching out to the Arduino team to see if I could visit their factory in Scarmagno (this was years before the Arduino/Genuino split) as part of our itinerary. Members of Officine Arduino (particularly managing director Davide Gomba) kindly took time out of their busy schedules to show me around their factory. They patiently waited as I expressed my inner shutterbug and general love for all things hardware, and I definitely came away with a lot of great photos.

A small town in northern Italy, Scarmagno is about an hour and a half west of Milan by car, near the Olivetti factories on the outskirts of Torino. The town handles all the circuit board fabrication, board stuffing, and distribution for officially branded Arduinos. I was really excited to see the factories, and the highlight of my tour was seeing System Elettronica, the PCB factory that made the Arduino PCBs.

One charming aspect of System Elettronica is that the owner painted the factory green, white, and red to match the colors of the Italian flag. On the factory floor, I saw some of that spirit in the red and green posts that ran the length of the facility.

A wide view of the factory floor at System Elettronica, as of August 2012

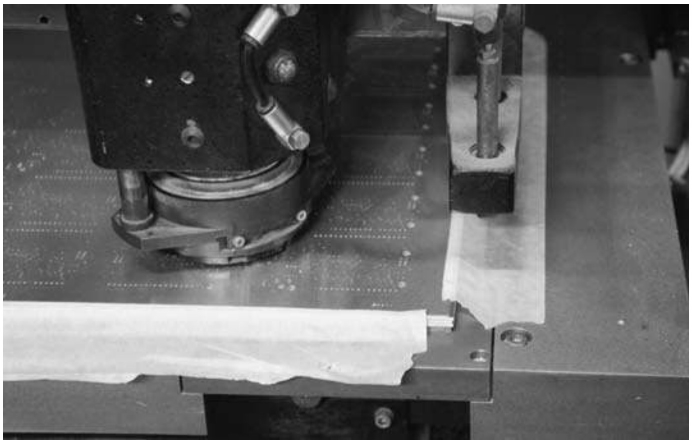
But I soon stopped paying much attention to the décor, as that factory floor was also where I got to follow a fresh batch of Arduino Leonardos through the entire manufacturing process. Here’s how those boards were made.

Starting with a Sheet of Copper

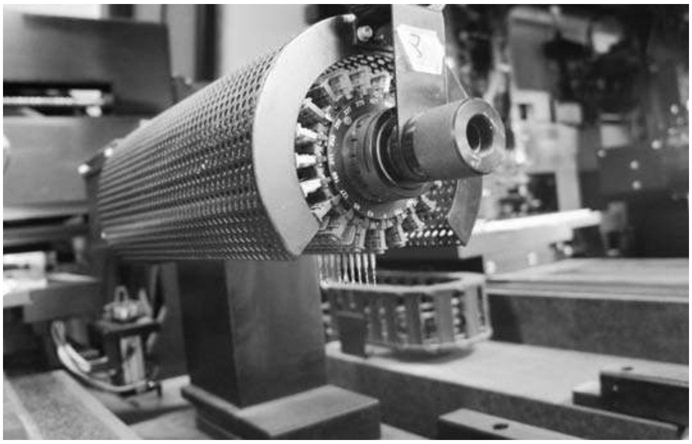
Arduino Leonardo boards start as huge sheets of virgin copper-clad FR-4, a material made of fiberglass and epoxy that most PCBs use for a substrate, an insulating and structural layer between the copper layers. The sheets were 1.6mm thick (the most common thickness for a PCB, which corresponds to 1/16 inch), probably a meter wide, and about a meter and a half long.

A stack of copper sheets waiting to become Arduino boards

The first step in processing PCBs is to drill all the holes—pads, vias (the small holes that connect different layers of the PCB), mounting holes, plated slots, and so forth. When a PCB is manufactured, the holes are drilled before *patterning*, the stage where a masking chemical is photographically defined on the sheet everywhere the final boards need to have copper, including locations of traces, solder pads, and so on. Some of the drilled holes are used to align the masks that pattern the traces later in the process. Drilling is also a dirty and messy process that could damage circuit patterns if they were in place beforehand.


The CNC drilling head used to drill the Arduino boards

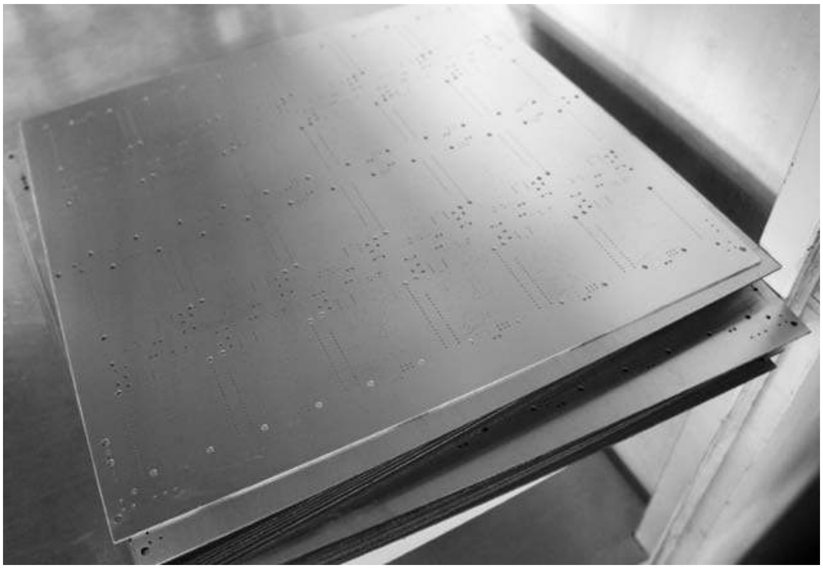
The blank copper panels were stacked three high, and a CNC drill took a single pass for all three, allowing it to drill three substrates at a time.


The drill rack used by the CNC drilling machine.
If you've ever had to create NC-drill files, this is that "drill rack."

Every hole in the Arduino board was mechanically drilled, including vias. The same is true of any PCB with through-holes, which is why the via count is such an important parameter in calculating the cost of a PCB.

Note that the particular drill I saw at System Elettronica was relatively small. I've seen massive drill decks in China that gang (mechanically attach) four or six drill heads together in a truck-sized machine, processing dozens of panels at the same time as opposed to the three panels this drill could handle. The reasoning behind this approach is that the precise, robotic positioning assembly is the expensive part of a drilling machine. The drill itself is cheap—just a spinning motor to drive the bit. So, one way to increase throughput is to gang several drills together on one large assembly and move them in concert. Each individual drill still goes through its own stack of panels, but for the price of one X-Y positioner, you get four to six times the throughput as the drill I saw on my trip to Italy. Those bigger machines drill so fast and hard that the ground shakes with every via drilled, even from several meters away.

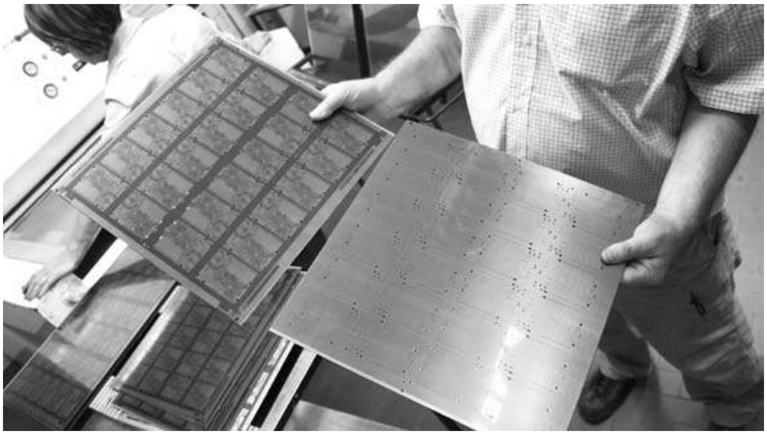
Once the panels are drilled, cleaned, and deburred, they are ready for the next step in the manufacturing process.



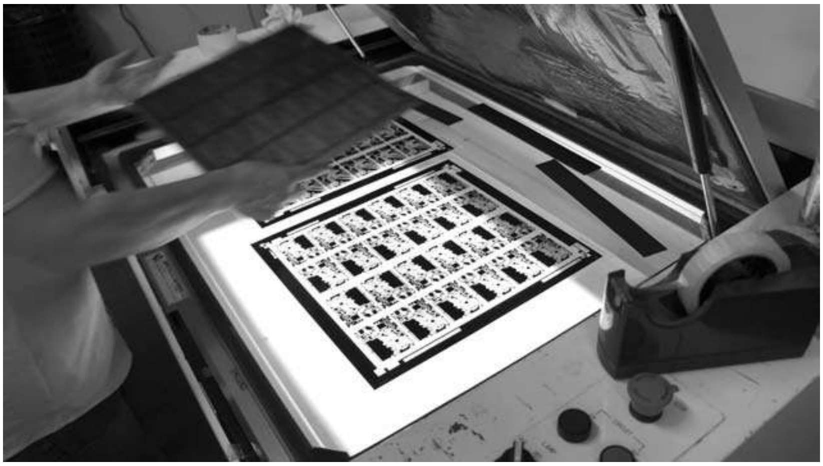
A stack of finished, drilled panels of Arduino Leonardo boards

Applying the PCB Pattern to the Copper

The next step is to apply a *photoresist*, a light-sensitive chemical, to the panel and expose a pattern. At System Elettronica, this process used a light box and a high-contrast film. I've also seen direct laser imaging—in the form of a raster-scanning laser—used to apply a pattern to a PCB. Direct laser scanners are more common in quick-turn prototype houses, and film imaging is more common in mass-production houses.

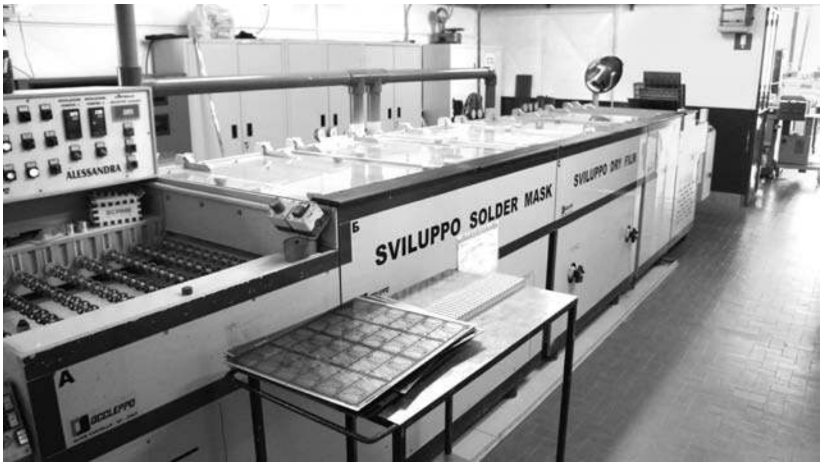


Before and after: the right panel shows photoresist prior to exposure, and the left panel after.

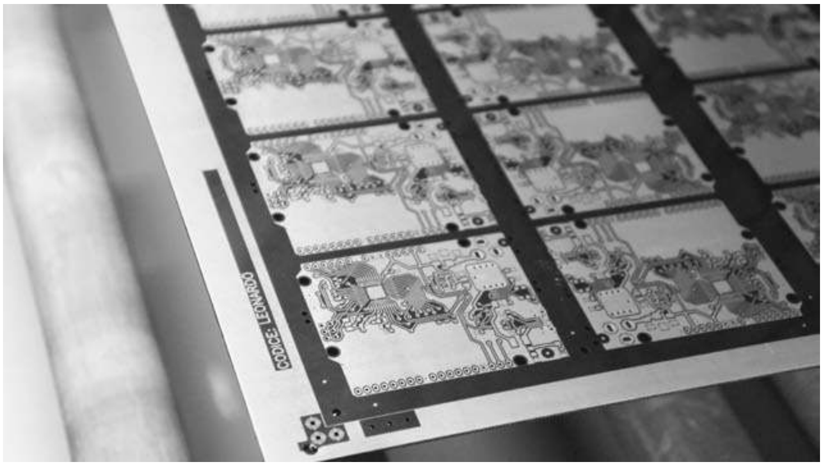


A PCB being mounted into a light box that will expose its unprocessed backside film

After the pattern is applied, each panel of boards is sent into a machine to be developed. In this case, the same machine is used to develop both the photoresist and the soldermask.



The machine that develops the photoresist



This photo of a panel with developed photoresist is one of my favorite photos from the System Elettronica factory. Also, something about "Codice: Leonardo" just sounds cool.

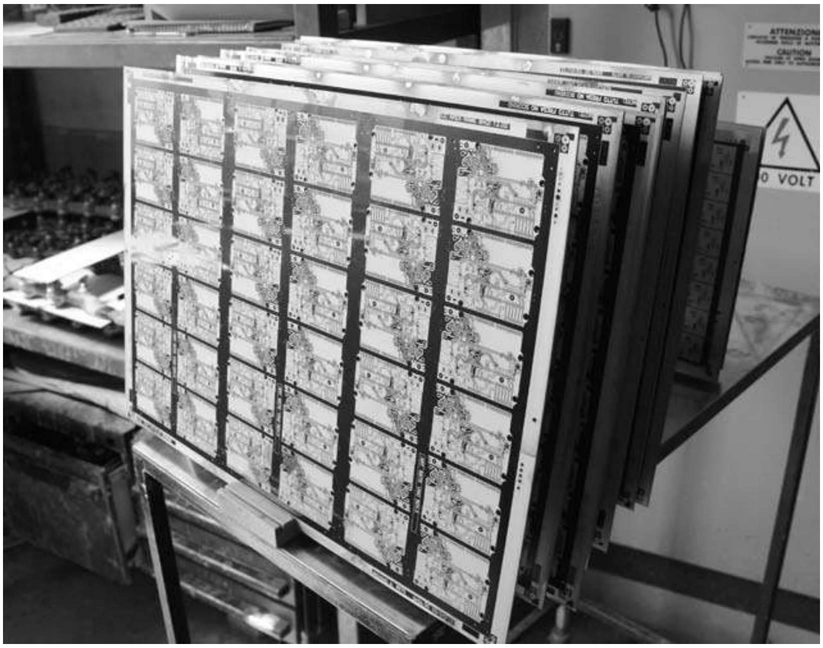
Etching the PCBs

After photo processing and development, the panels go through a series of chemical baths that etch and plate the copper.

The panels are swished gently back and forth in a chemical bath to expedite the etching process. The movement also circulates used etchant away from the panels, ensuring a more uniform etch rate regardless of the amount of copper to be removed. Moving the panels through these chemical baths was fully automated at Scarmagno. Automation is necessary because the panels must be treated with a series of caustic chemical baths with minimal exposure to oxygen. Oxygen can spoil a panel in a matter of seconds, so the transfer between the baths needs to be fast, and the amount of time a panel spends in a bath must be consistent. The baths also contain chemicals harmful to humans, so it's much safer for a robot to do this work.

A machine that moves panels around in etchant

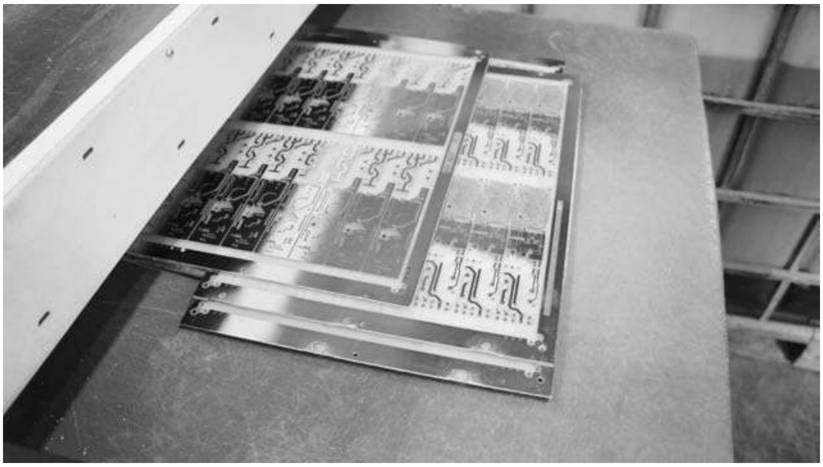
Once the panels are processed in this series of solutions, a dull, white plating (which I'm guessing is nickel or tin) develops on all the non-resist-covered surfaces of the panel, including the previously unplated through-hole vias and pads.



Panels of Arduino Leonardo boards after going through a series of chemical baths

At this point, the resist and unplated copper are stripped off, leaving just the raw FR-4 and the plated copper. The final step of processing produces a bright copper finish.

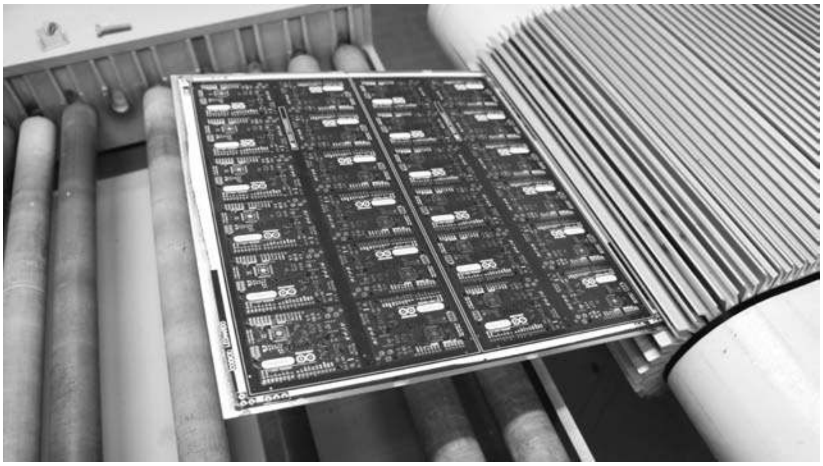
A panel etched of unwanted copper



PCB panels with bright, shiny copper. This photo doesn't show an Arduino panel, as those weren't going through the machine when I photographed it.

Applying Soldermask and Silkscreen

Once the copper is polished, the panels are ready for the *soldermask* (a protective, lacquer-like layer that insulates the copper traces below and prevents solder bridging above) and *silkscreen* (the ink used to label components, draw logos, and so on). These are applied in a process very similar to that of the trace patterns, using a photomask and developer/stripper machine.



A panel of Arduino boards with both soldermask and silkscreen developed

In the case of Arduinos, the silkscreen is actually a second layer of soldermask. A very specific formulation of dry-film white soldermask was procured for the Arduino team to create a sharp, good-looking layer that resolved the intricate artwork you see on Arduino boards—particularly the map of Italy on the backside. Other techniques I've seen for producing silkscreen layers include high-resolution inkjet printing, which is better suited for quick-turn board houses, and of course, the namesake squeegee-and-paint silkscreen process.

Testing and Finishing the Boards

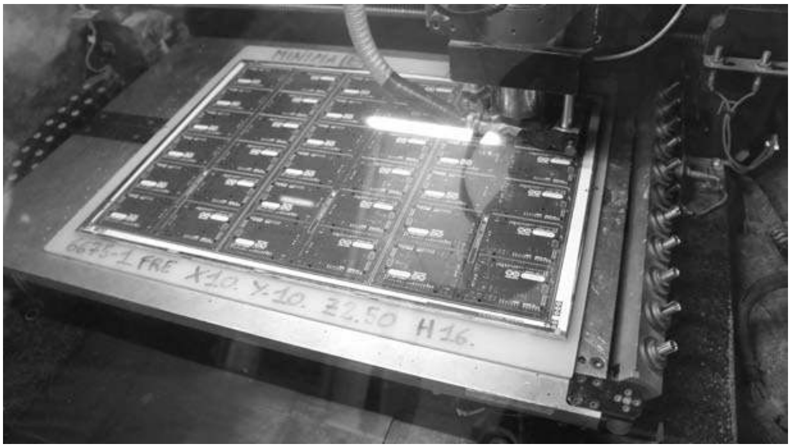
After all that chemical processing, the panels receive a protective plating of solder from a hot-air solder leveling machine.

With the solder plating in place, every board is 100 percent tested. Every trace has its continuity and resistance measured with a pair of flying probes. The process I saw is called *flying head testing* (also referred to as *flying probe testing*) and in that sort of setup, several pairs of arms with needlelike probes test continuity between pairs of traces in a swift tapping motion. Considering all the traces on an Arduino Leonardo, that's a lot of probing! Fortunately the robot's arms move like a blur, as it can probe hundreds of points per minute.

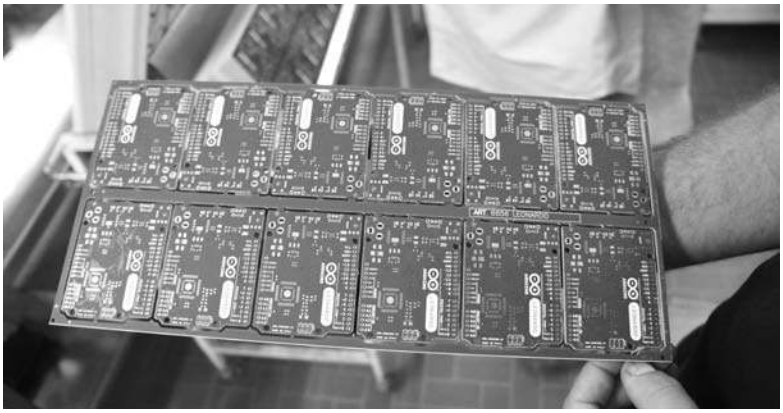
NOTE *An alternative to flying head testing is to use clamshell testers, where a set of pogo pins is put into a fixture that can test the entire board with a single mechanical operation. However, clamshell fixtures are very labor-intensive to assemble and maintain, and require physical rewiring every time the Gerber files describing the PCB images are updated. So, in lower volumes, flying probe testing is more cost-effective and flexible than clamshell testing.*

A stack of near-finished PCB panels, ready for a final step of routing out the individual boards

This particular facility only created the panels; a different factory actually populated the components. In situations like that, before the panels can be sent to the next factory, the individual PCBs need to be routed so they'll fit inside *surface mount technology (SMT)* machines to have the components placed. The panels are once again stacked up and batch-processed through a machine that uses a router bit to cut and release the boards. After that, the boards are finally ready to ship to the SMT facility.



Several Arduino panels, stacked for routing



Smaller 2×6 panels make SMT processing more efficient.

A veritable stack of about 25,000 bare Arduino PCBs, ready to leave the PCB factory. From there, they were stuffed, shipped, and sold to makers around the world!

I'm glad I made the side trip to visit the Arduino PCB factory. I've visited several PCB factories, and every one has a different character and its own set of tricks to improve yield, as well as unique limitations that designers need to compensate for. It was also interesting to see the little trick about using an extra layer of soldermask instead of silkscreen for achieving high cosmetic quality. While the resolution of a silkscreen is limited by the mesh of the silk barrier to hold the paint, soldermask is limited by the quality of the optics and chemical developing, giving over an order of magnitude improvement in resolution and ultimately a higher perceived quality. Normally the lower quality of silkscreen is acceptable because end users don't see the circuit boards inside computers, but for Arduino, the end product *is* the circuit board.

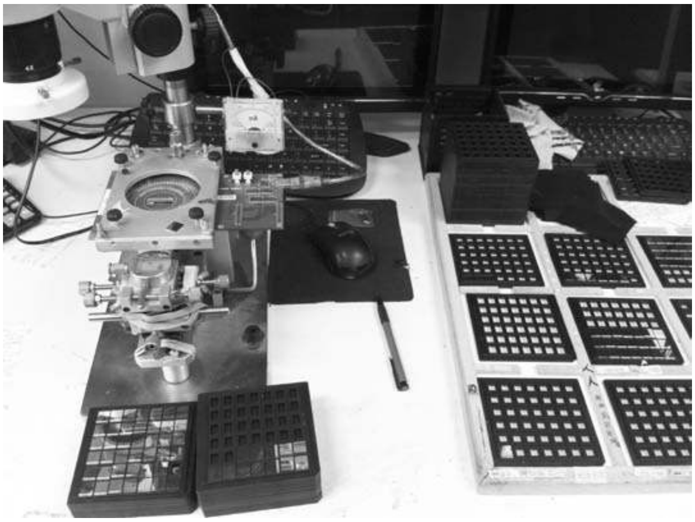
WHERE USB MEMORY STICKS ARE BORN

Several months after my tour of the Arduino factory, I had the good fortune of being a keynote speaker at Linux Conference Australia (LCA) 2013. In my talk, "Linux in the Flesh: Adventures Embedding Linux in Hardware," I discussed how Linux is in all kinds of devices we see every day. This story isn't about Linux, but it does connect me and, tangentially, LCA to a factory.

One of the tchotchkes I received from the LCA organizers was a little USB memory stick with Tux the penguin, the Linux mascot, on the outside. When I saw the device, I thought it was a neat coincidence that about a week before the conference, I was in a factory that manufactured USB memory sticks exactly like it. I saw the USB stick board assembly process from start to finish, and it surprisingly involved a lot less automation than the Arduino manufacturing process.

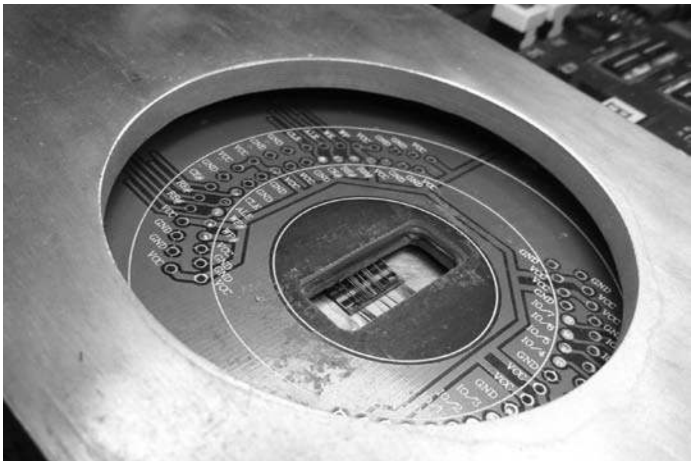
The Beginning of a USB Stick

USB sticks start life as bare flash memory chips. Prior to being mounted on PCBs, these chips are screened for memory capacity and functionality.

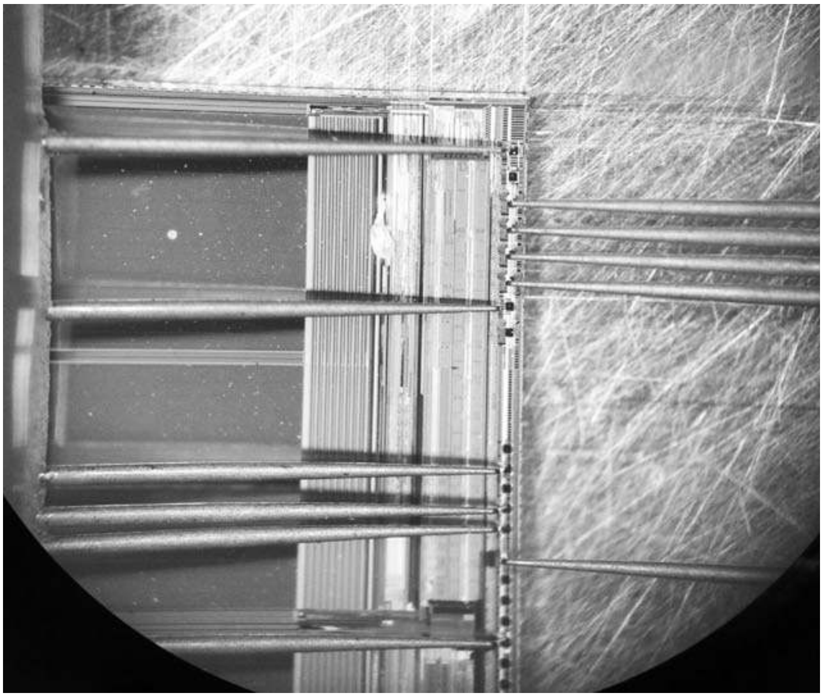


A workstation where flash memory chips are screened.
The metal rectangle on the left with the circular cutaway is the probe card.

At a workstation in this factory, stacks of bare-die flash chips awaited testing and binning with a *probe card*, which has tiny, very accurately positioned pins used to touch down on pads only a little bit wider than a human hair on a silicon wafer's surface. (I love how the worker at this particular station used rubber bands to hold an analog current meter to the probe card.)



The probe card, up close

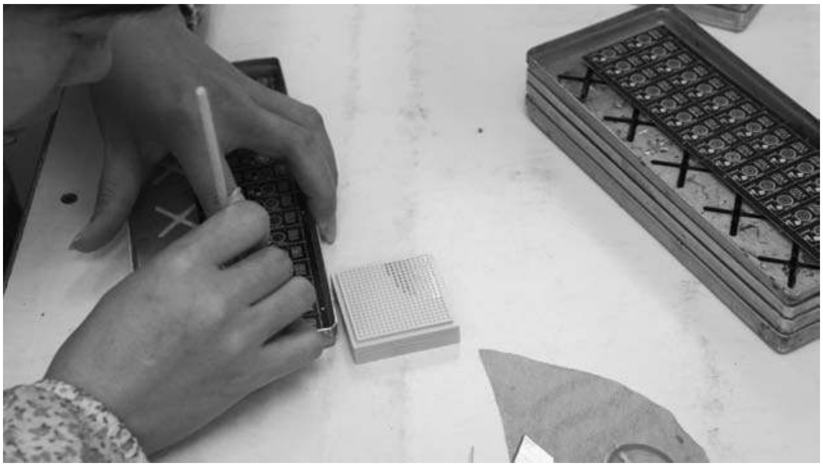


Looking through the microscope on the microprobing station. Notice the needles touching the square pads at the edge of the flash chip's surface. Each pad is perhaps 100 microns on a side—a human hair is about 70 microns in diameter.

Interestingly, the chips I saw were absolutely not tested in a clean-room environment. Workers handled chips with tweezers and hand suction vises, and mounted the probe cards into their jigs by hand.

Hand-Placing Chips on a PCB

Once the chips were screened for functionality, they were placed *by hand* onto the USB stick PCBs. This is not an unusual practice; every value-oriented wire-bonding facility I've visited relies on the manual placement of bare die.



A controller IC being placed on a panel of USB stick PCBs. The tiny bare dies are on the right, sitting in a waffle pack.

A zoomed-out view of the die-placing workstation

The lady I watched placing the bare die was using a chopstick-like tool made of hand-cut bamboo. I still haven't figured out exactly how the process works, but my best guess is that the bamboo sticks have just the right surface energy to adhere to the silicon die, such that silicon sticks to the tip of the bamboo rod. A dot of glue is preapplied to the bare boards, so when the operator touches the die down onto the glue, the surface tension of the glue pulls the die off of the bamboo stick.

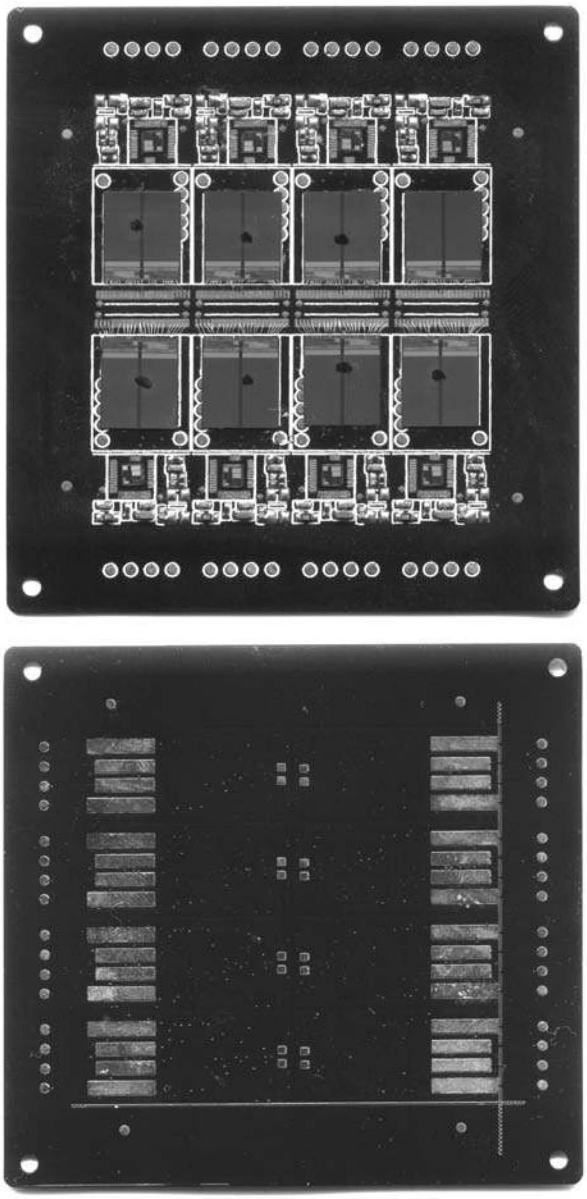
It's trippy to think that the chips inside my USB stick were handled using modified chopsticks.

Bonding the Chips to the PCB

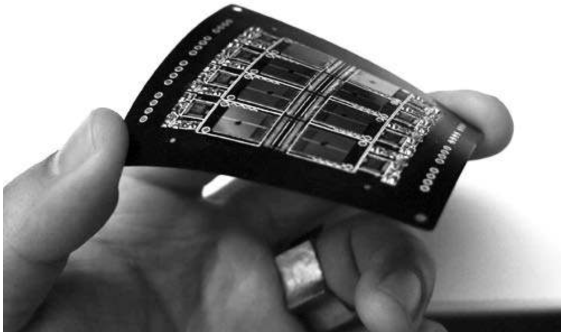
Once the chips were placed on the PCB, they were *wire bonded* to the board with an automated bonding machine, which uses computer-assisted image recognition to find the location of the bond pads (this is part of the reason the factories can get away with manual die placement). Wire bonding is the process that connects an integrated circuit to its packaging, and the automated bonding machine connected wires to the IC at an insane speed, rotating the circuit board all the while. As I watched this process, the operator had to pull off and replace a misbonded wire by hand, and then refeed the wire into the machine. Given that these wires are thinner than a strand of hair, and that the bonding pads on the packaging and the IC are microscopic, that was no mean feat of manual dexterity.

A Close-up Look at the USB Stick Boards

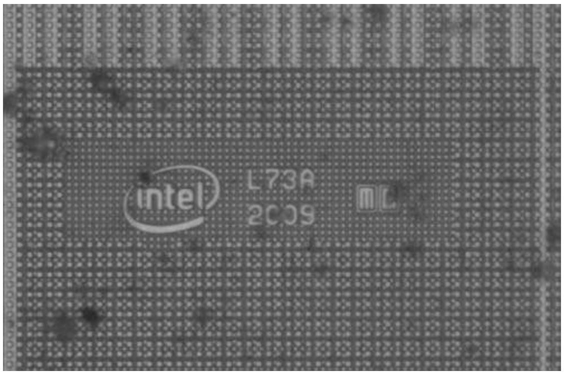
Just as the Arduino factory used panels containing multiple Leonardo boards, the USB memory stick factory used panels of eight USB sticks each. Each stick in the panel consisted of a flash memory chip and a controller IC that handled the bridging between USB and raw flash, a nontrivial task that includes managing bad block maps and error correction, among other things. The controller was probably an 8051-class CPU running at a few dozen MHz.



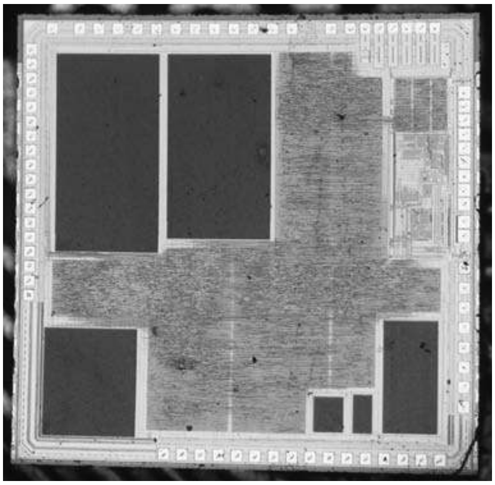
The partially bonded but fully die-mounted PCB that the factory owner gave me as a memento from my visit. Some of the wire bonds were crushed in transit.



Interestingly, the entire USB stick assembly is flexible prior to encapsulation.



The die marking from the flash chip. Apparently, it's made by Intel.



A die shot of the controller chip that went inside the USB sticks

Once the panels were bonded and tested, they were overmolded with epoxy and then cut into individual pieces, ready for sale.

But that's enough about electronics manufacturing; next, I want to show you a different kind of factory floor.

A TALE OF TWO ZIPPERS

My friend Chris "Akiba" Wang has a similar background to mine, except in his younger years he was way hipper: he was a dancer for acts like LL Cool J and Run DMC in the '90s. After going through a phase working for big semiconductor companies, he eventually quit and followed his passion to design and manufacture his own hardware projects. An expert in short-range, low-power wireless networking (he's authored a book on Bluetooth Low Energy and sells an Arduino + 802.15.4 variant called the "Freakduino"), he now consults for organizations like the United Nations and Keio University, runs FreakLabs, and collaborates with various dance acts, such as the Wrecking Crew, to provide unique and compelling lighting solutions for stage shows.

I had the good fortune of introducing Akiba to the greater Shenzhen area on a trip with MIT Media Lab students in 2013—the same trip where we toured the USB memory stick factory. Since then, he's been exploring deeper and deeper into the area. As his work spans the disciplines of performance art, wearables, and electronics, his network of factories is quite different from mine, so I always relish the opportunity to learn more about his world.

In January 2015, Akiba took me to visit his friend's zipper factory. I was very excited for the tour: no matter how humble the product, I always learn something new by visiting its factory. This factory was very different from both the Arduino and the USB stick facilities. There were far fewer employees, and it was a highly automated, vertically integrated manufacturer. To give you an idea of what that means, this facility turned metal ingots, sawdust, and rice into zipper parts.

Approximately 1 ton of ingots, composed of 93 percent zinc and 7 percent aluminum alloy

Compressed sawdust pellets, used to fuel the ingot smelter

Rice, used to feed the workers

Finished zipper puller and slider assemblies

Let's look at one side of how that process actually works.

A Fully Automated Process

Between the three input materials and the output product was a fully automated die-casting line to create the zipper pullers and sliders, a set of tumblers and vibrating pots (or, as I like to call them, "vibrapots") to release and polish the zippers, and a set of machines to deburr and join each puller to its slider. I think I counted fewer than a dozen employees in the facility, and I'm guessing their capacity well exceeds a million zippers a month.

I was mesmerized by the vibrapots* that put the zippers together. There were two vibrapots: one with pullers and one with sliders. Both sliders and pullers were deposited onto a moving rail, and as I watched these miracles at work, it looked as if the sliders and pullers were lining themselves up in the right orientation by magic. Each fell into its rail, and at the end of the line, they were pressed together into a familiar zipper form, all in a single, fully automated machine.

When I put my hand in the pot, I found there was no stirrer to cause the motion; I just felt a strong vibration. I relaxed my hand, and found it started to move along with all the other items in the pot. The entire pot was vibrating in a biased fashion, such that the items inside tended to move in a circular motion. This pushed the pullers and sliders onto the set of rails, which were shaped to take advantage of asymmetries in the objects to allow only the pieces that jumped on the rail in the correct orientation to continue to the next stage.

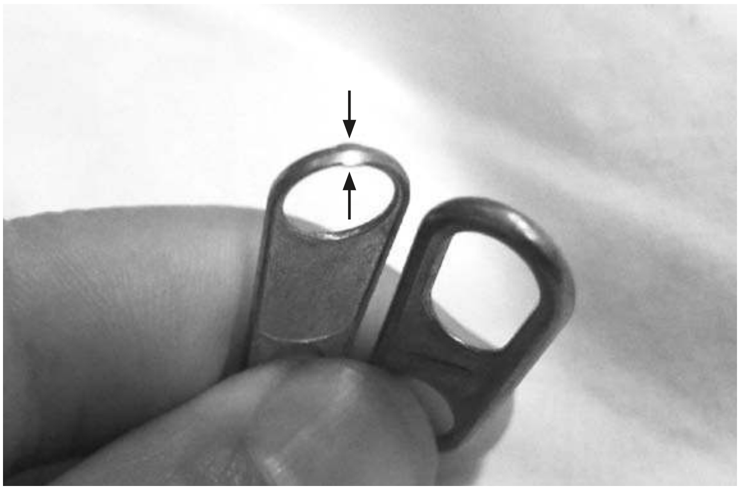
A Semiautomated Process

Despite the high level of automation in this factory, many of the workers I saw were performing one operation. They fed the pullers for a different kind of zipper into a device connected to another vibrapot containing sliders, while the device put the sliders and pullers together.

Of course, I asked, "Why do some zippers have fully automated assembly processes, whereas others are semiautomatic?"

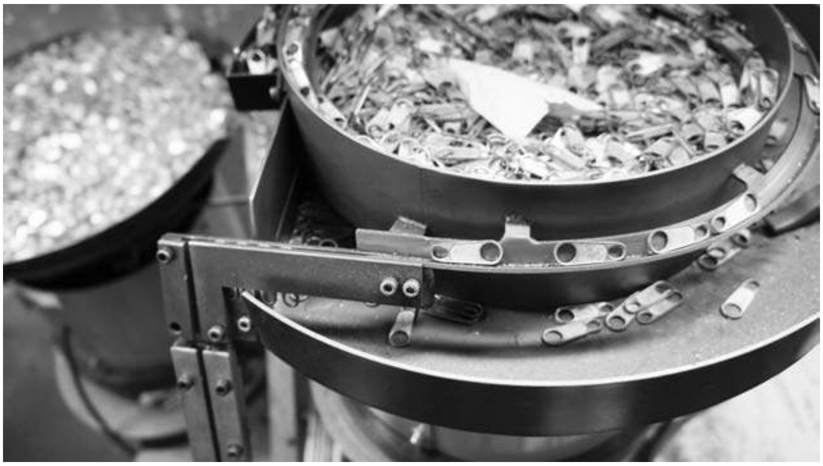
The answer, it turns out, is very subtle, and it boils down to shape.

*. *I honestly don't what they're called, so yes, I'm going to keep calling them that.*



Note the difference in these two pullers, indicated by the arrows.

One tiny tab, barely visible, was the difference between full automation and needing a human to join millions of sliders and pullers together. To understand why, let's review one critical step in the vibrapot operation. A worker kindly paused the vibrapot responsible for sorting the pullers into the correct orientation for the fully automatic process, so I could take a photo of the key step.



Pullers coming through the vibrapot

When the pullers came around the rail, their orientation was random: some faced right, some left. But the joining operation must only insert the slider into the smaller of the two holes. That tiny tab allowed gravity to cause all the pullers to hang in the same direction as they fell into a rail toward the left.

The semiautomated zipper design doesn't have this tab; as a result, the design is too symmetric for a vibrapot to align the puller. I asked the factory owner if adding the tiny tab would save this labor, and he said absolutely.

At this point, it seemed blindingly obvious to me that all zippers should have this tiny tab, but the zipper's designer wouldn't have it. Even though such a tab is very small, consumers can feel the subtle bumps, and some perceive it as a defect in the design. As a result, the designer insisted upon a perfectly smooth tab, which accordingly had no feature to easily and reliably allow for automatic orientation.

The Irony of Scarcity and Demand

I'd like to imagine that most people, after watching a person join pullers to sliders for a couple of minutes, would be quite content to suffer a tiny bump on the tip of their zipper to save another human the fate of manually aligning pullers into sliders for eight hours a day. Alternatively, I suppose an engineer could spend countless hours trying to design a more complex method for aligning the pullers and sliders, but there are two problems with that:

- The zipper's customer probably wouldn't pay for that effort.
- It's probably net cheaper to pay unskilled labor to manually perform the sorting.

This zipper factory owner had already automated everything else in the facility, so I figure they've thought long and hard about this problem, too. My guess is that robots are expensive to build and maintain; people are self-replicating and largely self-maintaining. Remember that third input to the factory—rice? Any robot's spare parts have to be cheaper than rice for the robot to earn a place on this factory's floor.

In reality, however, it's too much effort to explain this concept to end customers; and in fact, quite the opposite happens in the market. Putting the smooth zippers together involves extra labor, so the zippers cost more; therefore, they tend to end up in high-end products. This further enforces the notion that really smooth zippers with no tiny tab on them must be the result of quality control and attention to detail.

My world is full of small frustrations like this. For example, most customers perceive plastics with a mirror finish to be of a higher quality than those with a satin finish. There is no functional difference between the two plastics' structural performance, but making something with a mirror finish takes a lot more effort. The injection-molding tools must be painstakingly and meticulously polished, and at every step in the factory, workers must wear white gloves. Mountains of plastic are scrapped for hairline defects, and extra films of plastic are placed over mirror surfaces to protect them during shipping.

For all that effort, for all that waste, what's the first thing users do? They put their dirty fingerprints all over the mirror finish. Within a minute of a product coming out of the box, all that effort is undone. Or worse yet, the user leaves the protective film on, resulting in a net worse cosmetic effect than a satin finish.

Contrast this to satin finished plastic. Satin finishes don't require protective films, are easier for workers and users to handle, last longer, and have much better yields. In the user's hands, they hide small scratches, fingerprints, and bits of dust. Arguably, the satin finish offers a better long-term customer experience than the mirror finish.

But that mirror finish sure does look pretty in photographs and showroom displays!

3. the factory floor

The previous two chapters were filled with stories of my personal experiences learning, making mistakes, and growing with the manufacturing ecosystem in the greater Shenzhen area. In January 2013, after I'd learned the ropes, the MIT Media Lab asked me to start mentoring graduate students on supply chain and manufacturing, and I took them on a tour of Shenzhen (the same tour where I met Akiba and visited the USB memory stick factory). This chapter is an attempt to distill everything I taught over a course of weeks into a couple dozen pages.

The challenges and trade-offs in low-volume manufacturing are different from those of well-funded corporate exercises that prototype at the scale of thousands of units. I learned this over time, but not everyone has six years to bumble through all the newbie mistakes. If you're already in a fast-moving tech startup, you probably don't have the luxury of doing any exploration at all. The lessons in this chapter are applicable to anyone looking to bootstrap a hardware product from an initial prototype to moderate volumes (perhaps hundreds of thousands of units). Treat this summary as a general guideline, not a detailed roadmap. The devil is always in the details, and one fun part of making new, innovative hardware products is there's no end of novel and interesting challenges to be solved.

HOW TO MAKE A BILL OF MATERIALS

Most makers trying to scale up their output quickly realize the only practical path forward is to outsource production. If only outsourcing were as easy as schematic + cash = product!

Whether you work with the assembly shop down the street or send your work to China, a clear and complete *bill of materials (BOM)* is the first step to outsourcing production. Every single assumption you make about your circuit board, down to the color of the soldermask, has to be spelled out unambiguously for a third party to faithfully reproduce your design. Missing or incomplete documentation is the leading cause of production delays, defects, and cost overruns.

A Simple BOM for a Bicycle Safety Light

For a case study, suppose you ran a successful Kickstarter campaign for a bicycle safety light. It contains a circuit that uses a 555 timer to flash a small array of LEDs. After a great marketing campaign, several hundred orders need to be filled in a few months' time.

At first, your BOM for the bicycle light might look like this:

Quantity	Comment	Designator
1	0.1uF	C1
1	10uF	C2
3	white LED	D1, D2, D3
1	2N3904	Q1
1	100	R1
2	20k	R2, R4
1	1k	R3
1	555 timer	U1

A Very Basic Bicycle Safety Light BOM

This BOM, along with a schematic, is likely sufficient for any graduate of a US electrical engineering program to reproduce the prototype, but it's far from adequate for a manufacturing cost quotation. This version of the BOM addresses only electronics. A complete BOM for an LED flasher also needs to include the PCB, battery, plastic case pieces, lens, screws, any labeling (like a serial number), a manual, and packaging (plastic bag plus cardboard box, for example). It may also need a master carton to ship multiple LED flashers together, as a single boxed LED flasher is too small to ship on its own. Although cardboard boxes are cheap, they aren't free, and if they aren't ordered on time, inventory will sit on the dock until a master carton is delivered for final pack-out prior to shipment.

The following key information is also missing:

- Approved manufacturer for each component
- Tolerance, material composition, and voltage specification for passive components
- Package type information for all parts
- Extended part numbers specific to each manufacturer

Let's look at each of the missing items in more detail.

Approved Manufacturers

A proper factory will require you to supply an *approved vendor list (AVL)* specifying the allowed manufacturer(s) for every part on a PCB. A manufacturer is not a distributor but rather the company that actually makes a part. A capacitor, for example, could be made by TDK, Murata, Taiyo Yuden, AVX, Panasonic, Samsung, and so on. I'm still surprised at how many BOMs I've reviewed list DigiKey, Mouser, Avnet, or some other distributor as the manufacturer for a part.

It may seem silly to trifle over who makes a capacitor, but there are definitely situations where the maker of a component matters—even for the humble capacitor. For example, blindly substituting the filter capacitors on a switching regulator, even if the substitute has the same rated capacitance and voltage, can lead to unstable operation and even boards catching fire.

Of course, some parts in a design can be truly insensitive to the manufacturer, in which case I would mark "any/open" on the BOM for the AVL. (This is particularly true for parts like pull-up resistors.) This invites the factory to suggest their preferred supplier on your behalf.

Tolerance, Composition, and Voltage Specification

For passive components marked "any/open," you should always specify the following key parameters to ensure the right part is purchased:

- For resistors, specify at minimum the tolerance and wattage. A 1 kΩ, 1 percent tolerance, 1/4 W carbon resistor is a very different beast from a 1 kΩ, 5 percent tolerance, 1 W wire-wound resistor!
- For capacitors, specify at minimum the tolerance, voltage rating, and dielectric type. For special applications, also specify certain parameters such as ESR or ripple current tolerance. A 10 μF, electrolytic, 10 percent tolerance

capacitor rated for 50V has vastly different performance at high frequencies compared to a 10 µF, ceramic, 20 percent tolerance capacitor rated for 16V.

Inductors are sufficiently specialized that I don't recommend ever labeling them as "any/open" in your BOM. For power inductors, the basic parameters to specify are core composition, DC resistance, saturation, temperature rise, and current, but unlike resistors and capacitors, inductors have no standard for casing. Furthermore, important parameters such as shielding and potting, which can have material impacts on a circuit's performance, are often implicit in a part number; hence, it's best to fully specify the inductor. The same goes for RF inductors.

Electronic Component Form Factor

Always fully specify the *form factor*, or package type, of a component. Poorly specified or underspecified package parameters can lead to assembly errors. Beyond basic parameters like the Electronic Industries Alliance (EIA) or JEDEC Solid State Technology Association package code (that is, 0402, 0805, TSSOP, and so on), consider the following package information as you create your BOM:

Surface mount packages The height of a component can vary, particularly for packages larger than 1206 or for inductors. Pay attention to whether the board is slotting into a tight case.

Through-hole packages Always specify lead pitch and component height.

For ICs in general, try to also specify the common name that corresponds to the package, not just the manufacturer's internal code. For example, a Texas Instruments "DW" type package code corresponds to an SOIC package. This consistency check helps guard against errors.

Extended Part Numbers

Designers often think about components in abbreviated part numbers. A great example of this is the 7404. The venerable 7404 is a hex inverter, and has been in service for decades. Because of its ubiquity, *7404* can be used as a generic term for an inverter among design engineers.

When going to production, however, you must specify information like the package type, manufacturer, and logic family. A complete part number for a particular hex inverter might be **74**VHCT**04**AMTC, which specifies an inverter made by Fairchild Semiconductor, from the VHCT series, in a TSSOP package, shipped in tubes. The extra characters are very important, because small variations can cause big problems, such as quoting and ordering the wrong packaged device and being stuck with a reel of unusable parts or subtle reliability problems.

For example, on a robotics controller I designed (codenamed *Kovan*), I encountered a problem due to a mistaken substitution of *VHC* in the part number for a component in the *VHCT* logic family. Using the VHC part switched the input thresholds of the inverter from TTL to CMOS logic-compatible, and some units had an asymmetric response to input signals as a result. Fortunately, I caught this problem before production ramped. The correct part was used on all other units, and I avoided a whole lot of potential rework—or worse, returns from upset customers. Luckily, the only cost of the mistake was reworking the few prototypes I was validating before production.

Here's another example of how missing a few characters in a part number can cost thousands of dollars. A fully specified part number for the LM3670 switching regulator might be LM3670MFX-3.3/NOPB. If */NOPB* is omitted, the part number is still valid and orderable—but that version uses leaded solder. This could be disastrous for products exporting to a region that requires RoHS compliance (meaning lead-free, among other things), like the European Union.

The *X* in the part number is another, more subtle issue. Part numbers with an *X* come in reels of 3,000 pieces, and those lacking an *X* come in reels of 1,000 pieces. While many factories will question an */NOPB* omission since they typically assemble RoHS documentation as they purchase parts, they rarely flag the reel quantity as an issue.

But *you* should care about the reel quantity. If you plan to build only 1,000 products, including the *X* in the part number means you'll have 2,000 extra LM3670s. And yes, you're on the hook to pay for the excess, since your BOM specified that part number. There are many valid reasons for ordering excess parts, so factories will rarely question a decision like that.

On the other hand, parts ordered in lots of 1,000 units are a bit more expensive per unit than those ordered in lots of 3,000. So, if you leave out the *X* as your volume increases, you'll end up paying more for the part than you have to. Either way, the factory will quote your BOM exactly as specified, and if your quantity specifiers are incorrect, you could be leaving money on the table—or worse, losing money.

The bottom line? Every digit and character counts, and lack of attention to detail can cost real money!

The Bicycle Safety Light BOM Revisited

With those four points in mind, consider how a proper, fully specified BOM for the bicycle safety light example might look.

Qty	Value	Package	Designator	AVL1	AVL1 P/N	MOQ	Lead time
1	0.1uF, ceramic, 25V, 10%, X5R	0402	C1	Taiyo Yuden	TMK105BJ104KV-F…	10000	8 wks
1	10uF, ceramic, 16V, 10%, X5R	1206	C2	TDK	C3216X5R1C106K(085AB)	2000	12 wks
3	white LED, water clear lens	T-1 ¾	D1, D2, D3	Lumex	SSL-LX5093UWC/G	3000	12 wks
1	2N3904	SOT-223	Q1	ON Semiconductor	PZT3904T1GOS	1000	6 wks
1	100 ohm, 1/2W, 5%	2010	R1	Panasonic	ERJ-12SF100U	5000	8 wks
2	20k, 1/16W, 1%	0402	R2, R4	any/open		10000	8 wks
1	1k, 1/16W, 5%	0402	R3	any/open		10000	8 wks
1	NE555D	SOIC-8	U1	TI	NE555D	1000	4 weeks
1	PCB, FR4, 1.6mm +/- 10%, green soldermask, HASL, white silkscreen, 5cm x 8cm		PCB	TBD	FLASHYLIGHT_GERBERS_V1.ZIP	1000	4 weeks
1	Plastic ABS, bottom case, satin finish, lead free, black			TBD	FLASHYLIGHT_BOT_V1.STEP	1000	16 wks / 4 wks
1	Plastic ABS, top case, satin finish, lead free, black			TBD	FLASHYLIGHT_TOP_V1.STEP	1000	16 wks / 4 wks
1	Plastic polycarbonate, lens, mirror finish, lead free, clear			TBD	FLASHYLIGHT_LENS_V1.STEP	1000	16 wks / 4 wks
4	Screw, M2x4, pan head philips, self-tapping 5mm			any/open		4000	stock
1	Battery Snap, 9V,15CM red and black 26 AWG wires (5mm Leads)			Kaweei	CBS-150	5000	1 week
1	Instruction manual, A4 sheet, black and white, two sides printed			any/open	flashylite_manual_v2.ai	1000	3 weeks
1	10cm x 12cm PE plastic bag, clear			any/open		1000	1 week
1	Bar code label, serial number and date code, CODE39 5mm x 15mm			any/open	barcode_sample_v1.pdf	1000	1 week
1	Cardboard box, 6cm x 6cm x 10cm, natural color, 50lb stock			any/open	see included box sample	1000	1 week
0.02	Master carton, 60cm x 40cm x 20cm			any/open		100	1 week

The Improved Bicycle Safety Light BOM

There's big difference between a BOM that any engineer could use to produce a prototype, like the first one I showed for the bicycle safety light, and a BOM like this, which any factory could use to mass-produce a product. Notice the MOQ (minimum order quantity) and Lead Time columns in particular. These columns are irrelevant when you're building low-volume prototypes, as you'd typically buy parts from distributors that have few MOQ restrictions and maintain stock for next-day deliveries. When scaling into production, however, you save a lot of money by cutting the distributor overhead and buying through wholesale channels. In wholesale channels, MOQs and lead times matter.

The good news is that the factory will fill in the MOQ and lead time as part of the quotation process. But you'll find it helpful to track these parameters from the beginning. If the MOQ of a particular component is very high, the factory may have to buy massive numbers of excess parts, which increases the effective price of the project. If the lead time of a part is very long, you may want to consider redesigning for a part with a shorter lead time. Using parts with shorter lead times not only saves time, but also improves cash flow: no one wants to tie up cash on long-lead components four months in advance of sales revenue.

This BOM also includes several nonelectronic items—like the box, a bar code label, and so on—which wouldn't be on the engineering prototype's BOM. These miscellaneous bits are easy to forget, but a missing user manual in an initial BOM is often not discovered until the final sample is opened for approval, leading to a last-minute scramble to get the manual into the final product. Many products have been delayed simply because a user manual or box art wasn't completed and approved in time, and it sucks to have a hundred thousand dollars' worth of inventory idling in a warehouse for want of a slip of paper.

Beyond a proper BOM, providing the factory with golden samples of your product along with your CAD files is another best practice. These working prototypes enable the factory to make smarter decisions about any ambiguities in your submitted BOM. Hand-soldering one more unit just for the factory may seem annoying, but in my opinion, a few hours of soldering beats a week of trading emails with the factory.

NOTE *When you're building a business model, parts and packaging still aren't the only costs to consider. Even this detailed BOM doesn't list factory margin, labor for assembly, pack-out, shipping, duties, and so on. I discuss these "soft costs" in "Picking (and Maintaining) a Partner" on page 105.*

Planning for and Coping with Change

Of course, even if your design is perfect and your BOM is ideal, your design may still have to change if vendors *end-of-life (EOL)*, or stop making, components you selected. And let's face it: there's always a chance your design assumptions won't survive contact with real consumers, too.

Before crossing the threshold into production, formalize the process for changing a design with the factory. It's best practice to use written, formal *engineering change orders (ECO)* to update the factory on any changes after the initial quotation. At minimum, an ECO template should include:

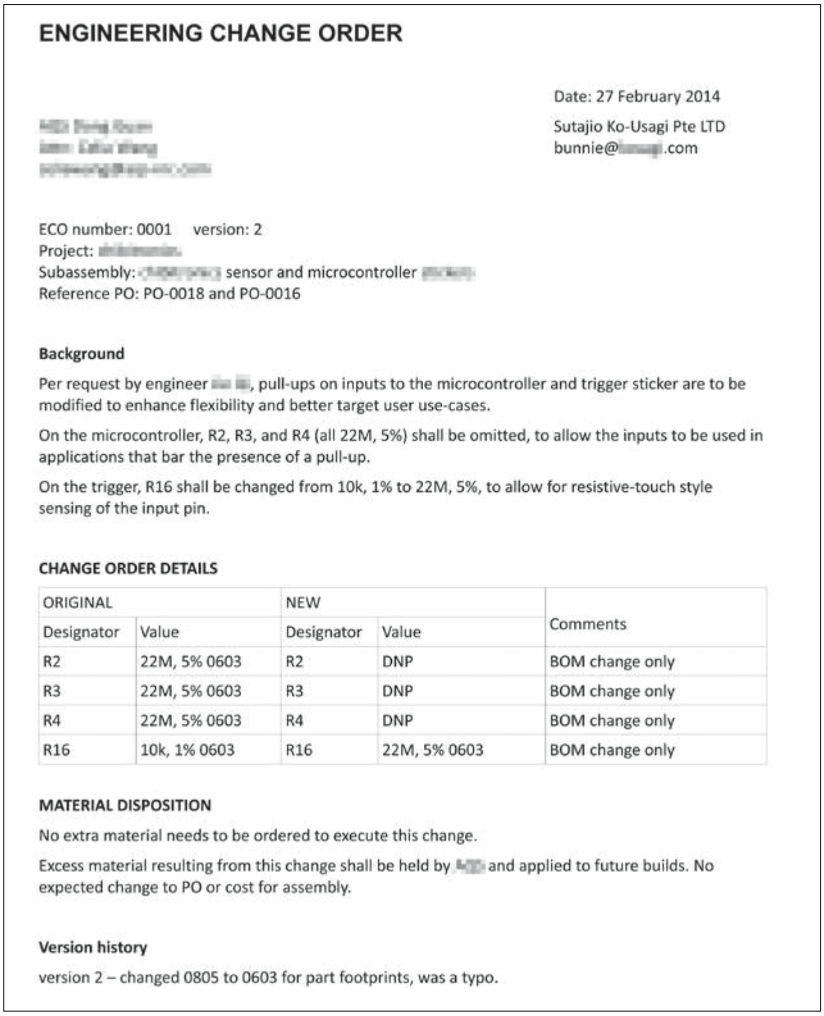
- The details of each changed part, and a brief explanation of why the change is needed
- A unique revision number for conveniently referencing the change down the road
- A method to record the factory's receipt of the ECO paperwork

Be thorough with ECOs, rather than relying on casual emails, or the buyers at your factory may buy the wrong part. Worse yet, the factory might *install* the wrong part, and entire lots of your product will need to be scrapped or reworked. Even after troubleshooting a problem with the factory engineers, I still write up a formal ECO and submit it to the production staff to formalize the findings. I hate paperwork as much as the next engineer, but in production, one small mistake can cost tens of thousands of dollars, and that thought keeps me disciplined on ECOs.

On the next page is an actual ECO I issued that ended up saving me time and money.

Note the date on this ECO: February 27, 2014. This ECO was issued right before the Chinese New Year, when the factories go on holiday for a couple of weeks. There is significant turnover of unskilled labor inside factories after the holidays, and thus there's a lot of opportunity for work orders to get lost and forgotten. Worried that the ECO would be missed, I consulted with the managers after the factory resumed production to ensure the ECO wasn't forgotten. They assured me it was applied, but I still felt a vague paranoia, so I asked for photos of the circuit board to confirm. Sure enough, the first production batch was missing the change in my ECO.

Thanks to the detailed ECO, the factory readily admitted its error, repaired the entire production run, and paid for the reworking. But if I'd sent the change order in a quick email without referencing specific batches or work orders, there could have been sufficient ambiguity for the factory to get out of the rework charges. The factory could have argued that it thought I meant to apply the change to a future production run, or it could simply deny receiving a confirmed order, as emails are a fairly casual form of communication. Either way, a few minutes of documentation saved days of negotiation and hundreds of dollars in rework fees.



ENGINEERING CHANGE ORDER

Date: 27 February 2014
Sutajio Ko-Usagi Pte LTD
bunnie@[redacted].com

[redacted]

ECO number: 0001 version: 2
Project: [redacted]
Subassembly: [redacted] sensor and microcontroller [redacted]
Reference PO: PO-0018 and PO-0016

Background

Per request by engineer [redacted], pull-ups on inputs to the microcontroller and trigger sticker are to be modified to enhance flexibility and better target user use-cases.

On the microcontroller, R2, R3, and R4 (all 22M, 5%) shall be omitted, to allow the inputs to be used in applications that bar the presence of a pull-up.

On the trigger, R16 shall be changed from 10k, 1% to 22M, 5%, to allow for resistive-touch style sensing of the input pin.

CHANGE ORDER DETAILS

ORIGINAL		NEW		Comments
Designator	Value	Designator	Value	
R2	22M, 5% 0603	R2	DNP	BOM change only
R3	22M, 5% 0603	R3	DNP	BOM change only
R4	22M, 5% 0603	R4	DNP	BOM change only
R16	10k, 1% 0603	R16	22M, 5% 0603	BOM change only

MATERIAL DISPOSITION

No extra material needs to be ordered to execute this change.

Excess material resulting from this change shall be held by [redacted] and applied to future builds. No expected change to PO or cost for assembly.

Version history

version 2 – changed 0805 to 0603 for part footprints, was a typo.

Caption TKK

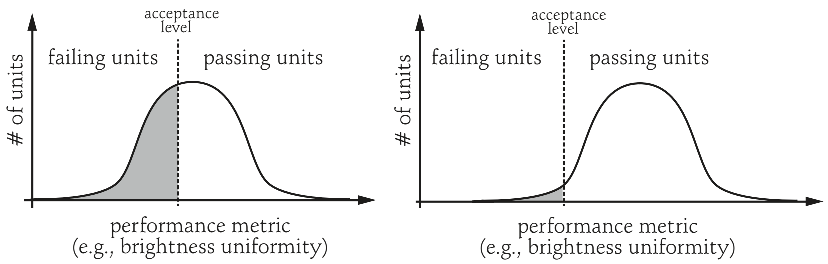
PROCESS OPTIMIZATION: ON DESIGN FOR MANUFACTURING

While you're designing your final product and putting together a BOM, considering *yield*, the number of good units that come out of the manufacturing process, is also important. Yield is a boring subject for many engineers, but for entrepreneurs, success or failure will be determined in part by whether they achieve a reasonable yield. Fortunately, you can help your yield by designing with it in mind.

Why DFM?

Unlike software, every copy of a physical good has slight imperfections. Sometimes the imperfections cancel out; sometimes, they gang up and degrade performance. As production volume ramps, a fraction of the product always ends up nonsalable. In a robust design, the failing fraction may be so small that functional tests can be simplified, leading to further cost reductions. In contrast, designs sensitive to component tolerances require extensive testing, and will suffer heavy yield losses. Reworking defective units incurs extra labor and parts charges, ultimately eroding profits.

Thus, redesigning to improve robustness in the face of normal manufacturing tolerances is a major challenge of moving from the engineering bench to mass production. This process is called *design for manufacturing (DFM)*.



Left, before DFM, almost half the units are not meeting the acceptance level and failing. Right, after DFM, the acceptance level is the same but the average performance is improved, leading to most units passing.

To understand the importance of DFM, consider these graphs. Each depicts a *bell curve*, which is an assumed statistical distribution of a particular parameter. The x-axis is a parameter of interest, and the y-axis is the number of items produced that hit the given parameter. For example, in a plot of the brightness of thousands of LEDs, the x-axis would be brightness, and the y-axis would be the number of LEDs that reach a given brightness. The position of the bell curve relative to the pass/fail criteria determines the net production yield.

On the right-hand curve, most LEDs are bright enough, and most of the production inventory is shippable. On the left-hand curve, maybe 40 percent of the LEDs pass. Given that most hardware companies operate with about a 30–50 percent gross margin, scrapping 40 percent of the material would mean the end of the business. In such a situation, the only viable options are to either spend the time and effort to rework the LEDs until they pass, or to lower the performance requirement. The product wouldn't be as high quality as hoped, but at least the business could keep operating.

Tolerances to Consider

The goal of DFM is to ensure that your product always passes muster, and you're never faced with the unsavory choice of reducing margins, lowering quality standards, or going out of business. But there are some component aspects to think about when applying DFM.

ELECTRONIC TOLERANCES

Passive component tolerances are the most obvious tolerances to design for. If a resistor's true value can be +/–5 percent of its labeled value, be sure the rest of your circuit can cope with the edge cases.

Active component datasheet parameters—like current gain (hFE) for bipolar transistors, threshold voltage (V_t) for field effect transistors (FETs), and forward bias voltage (V_f) for LEDs—can also vary widely. Always read the datasheet, and watch for parameters with a great disparity between their minimum and maximum values, a difference often referred to as a *min-max spread*. For example, the min-max on hFE for Fairchild's 2N3904 ranges from 40 to 300, and the V_f on a superbright LED from Kingbright is between 2 and 2.5V.

Nominal operating voltage aside, a component's maximum voltage rating is particularly important for capacitors and input networks. I try to use capacitors rated for twice the nominal voltage; for example, where possible, I use 10V capacitors for 5V rails and 6.3V capacitors for 3.3V rails. To understand why, consider ceramic capacitor dielectrics, which have reduced capacitance with increasing voltage. In designs operating near a ceramic capacitor's maximum voltage, that component's operating capacitance will be at the negative end of its tolerance range. Also, *input networks* (any part of the circuit that a user can plug something into) are subject to punishing electrostatic discharge and other transient abuses, so pay special attention to the ratings of capacitors there to achieve your desired reliability.

Finally, after you have a good sense of the components you'll use, pay close attention to trace widths and layer stack variations when designing your PCB. These will impact systems that require matched impedance or deal with high currents.

MECHANICAL TOLERANCES

Electronic tolerances aren't the end of your worries, though; mechanical tolerances are important, too. Neither PCBs nor cases will come out exactly the right size, so design your case with some wiggle room. If your case design has zero tolerance for the PCB dimensions, half the time the factory will force PCBs into cases, when either the PCB is cut a little large or the case comes out a little small. This can cause unintentional mechanical damage to the circuitry or the case.

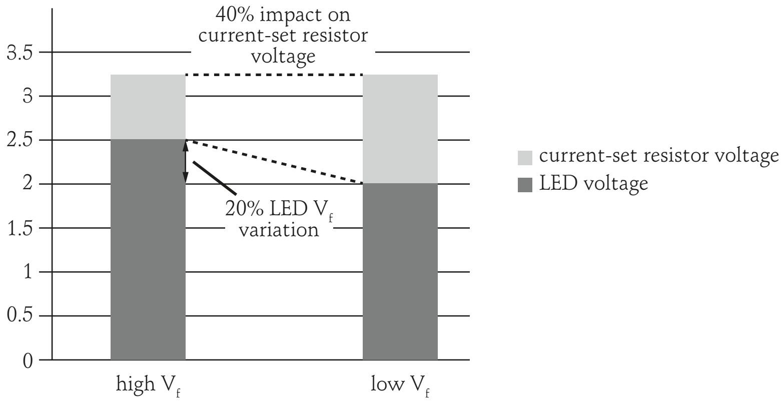
And don't forget about cosmetic blemishes! Any manufactured product is subject to small blemishes, such as dust trapped in plastics, small scratches, sink marks, and abrasions. It's important to work out the acceptance criteria for such defects with the factory ahead of time. For example, you might tell the factory that a unit can be considered "good" if it has no more than two dot blemishes larger than 0.2mm, no scratch longer than 0.3mm, and so on. Most factories will have a particular system they've adopted to describe and enforce these standards. If you discuss these parameters in advance, the factory can craft the manufacturing process to avoid such defects, as opposed to the more expensive alternative of building units and throwing away those that don't meet criteria imposed late in the game.

Of course, avoiding defects isn't free. To keep your product cheaper, avoid high-gloss finishes and consider using matte or textured finishes that naturally hide blemishes.

Following DFM Helps Your Bottom Line

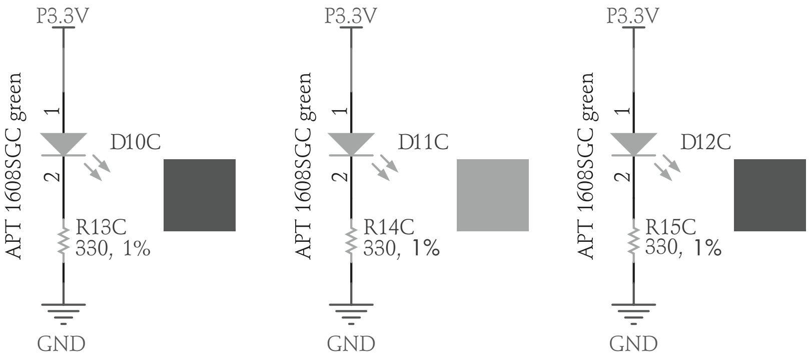
To imagine DFM in a real-world scenario, return to the bicycle safety flasher case study from "How to Make a Bill of Materials" on page 72. Say the prototype design calls for an array of three LEDs in parallel, each with its own resistor to set the current. The *forward bias voltage*, or V_f, of an LED at a given brightness can vary by perhaps 20 percent between devices; in this case, that swing is from 2.0 to 2.5V.

A design that limits the current to the LEDs with resistors, called *resistive current limiting*, will amplify this variation. This happens because an efficient circuit would drop a minority of the voltage across the current-limiting resistor, leaving the parameter that sets the current (the voltage drop across the resistor) more sensitive to the variation in $\mathrm{V_f}$. Since the brightness of an LED is not proportional to the voltage but rather the current flowing through it, setting the LED brightness with resistive current limiting can cause jarring inconsistencies in LED brightness.



Comparing high V_f and low V_f corners

In this example, a 20 percent LED V_f variation (from 2.0V to 2.5V, per the LED manufacturer's specification) leads to a 40 percent change in the voltage across a current-set resistor for a fixed 3.3V supply. This will cause a 40 percent change in the current flowing through the LED. As brightness is directly proportional to current, the change manifests as up to a 40 percent variation in perceived brightness between individual LEDs. A design like that may work well most the time; the problem would only be pronounced when a high V_f unit is observed next to a low V_f unit.



Setting current for individual LEDs using resistors can lead to dramatic variations in brightness.

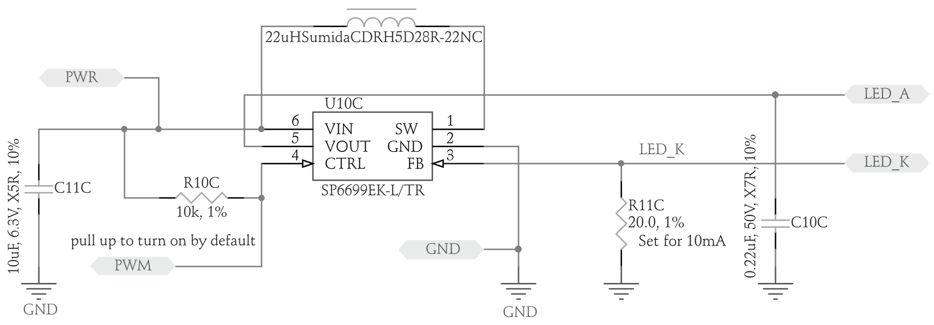
The one or two units prepared on the lab bench during development may have looked great, but in production a meaningful fraction may have such serious brightness uniformity issues that units must be rejected. As most large hardware businesses have to survive on lean margins, losing even 10 percent of finished goods to defects is a terrible outcome.

One stop-gap option is to rework the failed units. A factory can identify an LED that is too dim or too bright in an array, and replace it with one that matches its cohorts better. But that rework would drive up costs and result in an unexpected and unpleasant invoice at the 11th hour of a manufacturing program. Naive designers may be inclined to blame the factory for poor quality and argue over who should bear the cost, but it's better to proactively avoid these kinds of problems by subjecting every design to a DFM check, and using a small pilot run to sanity-check yield before punching out a whole bunch of units.

The cost of yield fallout quantifies how much money to spend on extra circuitry to compensate for normal component variability. For example, a product with a $10 *cost of goods sold (COGS)* that yields 80 percent good units has an effective cost per salable unit of $12.50, as calculated with this formula:

$$\text{Effective cost} = \text{COGS} \times \text{total units built} / \text{yielded units}$$

Increasing the COGS by $2.50 to improve yield to 100 percent would allow you to break even. But using the same formula, spending $1 extra dollar in COGS to improve yield to 99 percent would actually improve the bottom line by $1.38.



A circuit to set the current on three LEDs, created by applying DFM

In the case of the bicycle safety light, that dollar could be spent on a current-feedback boost regulator IC like the SP6699EK-L/TR, allowing the LEDs to be stacked in series instead of parallel. The design would be far more complicated and expensive than using individual resistors, but it would guarantee each LED has a consistent, identical current flowing through it by driving all three LEDs in a series circuit with a fixed-current feedback loop. That would virtually eliminate brightness variation. While the cost of the boost regulator is much greater than the penny spent on three current-limiting LEDs, the improvement in manufacturing yield more than pays for the extra component costs. In fact, this trick is standard practice for applications that require good uniformity of brightness out of LEDs, such as in the backlights of LCD panels. A typical mobile phone backlight uses about a dozen LEDs, but, thanks to circuits like this, you never see light or dark splotches despite the large variations in V_f between the constituent LEDs.

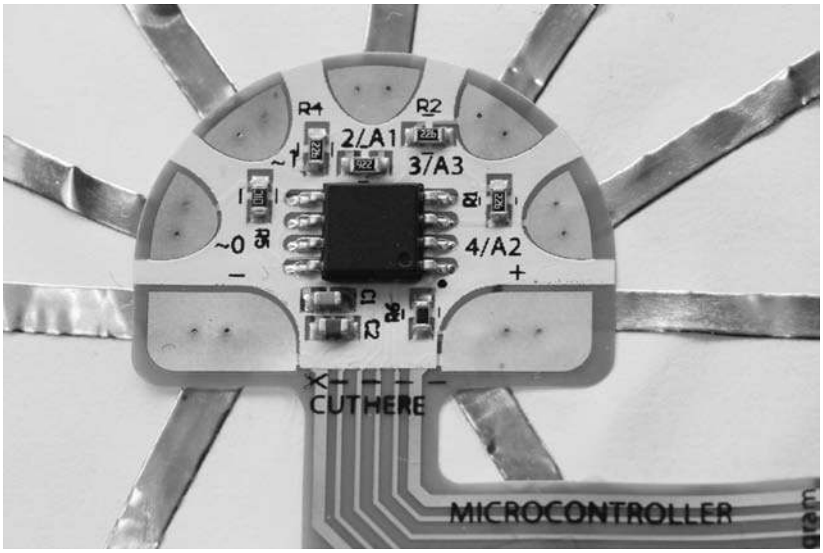
The Product Behind Your Product

Alongside dealing with tolerances, another often-neglected design responsibility is the test program. A factory can only detect the problems it is instructed to look for. Therefore, every feature of a product must be tested, no matter how trivial. For example, on a chumby device, every user-facing feature had an explicit factory test, including the LCD, touchscreen, audio, microphone, all the expansion ports (USB, audio), battery, buttons, knobs, and so on. I made sure that even the simplest buttons were tested. While it's tempting to skip testing such simple components, I guarantee that anything not tested will lead to returns.

I like to call the factory tester "the product behind your product." That's because in some cases, the factory tester is more complicated and more difficult to engineer than the product you're trying to sell. This is particularly true of simple products.

A REAL-WORLD TEST PROGRAM

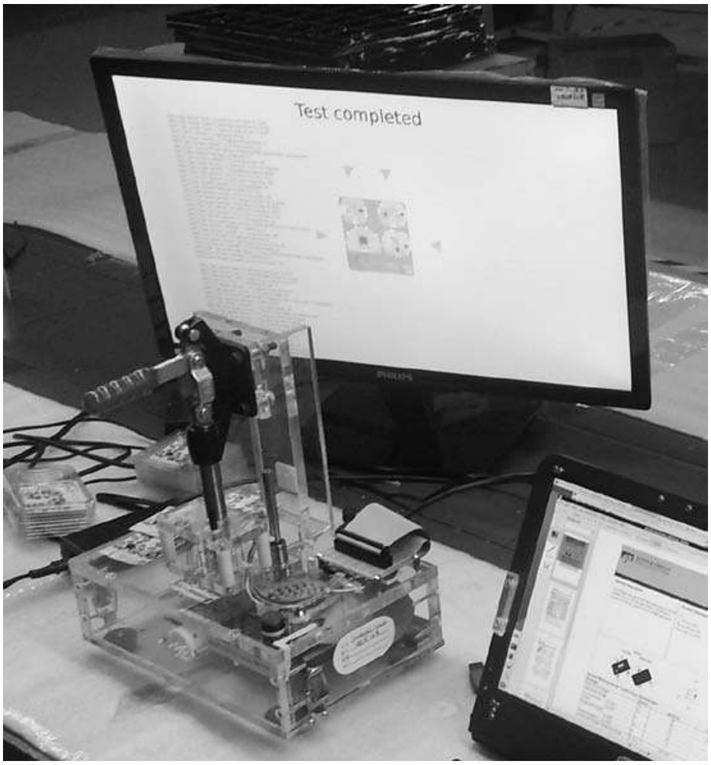
As a case study, consider this microcontroller sticker from Chibitronics, a project I discuss at length in Chapter 8.



A microcontroller circuit—on a sticker

This circuit is very simple: it consists of just an 8-bit AVR microcontroller and a handful of resistors and capacitors.

(It's also the same product referred to in the ECO example on page 82.) My collaborator and I sketched in Adobe Illustrator for about two days before we derived the final shape for this product. Then we spent about a day in Altium designing the circuit, and about a week coding in the Arduino IDE to create its firmware. In all, the development process took about two weeks. For production, the microcontroller is paired with a set of sensors that can process sound, light, and touch, and as a result, the test program runs on all four at the same time.



The testing machine for the Chibitronics microcontroller sticker

The test rig pictured consists of a 32-bit ARM computer running Linux with a graphical UI rendered on an HDMI monitor. Behind this is an FPGA, some adapter electronics to create analog waveforms for testing, and a mechanical pogo-pin assembly for touching down on the sticker. Breaking down the design process for this rig into its component parts, we spent:

- Several days designing in Altium
- A week programming in the Xilinx ISE for the FPGA
- A couple of weeks hacking on Linux drivers
- A couple of solid months hacking in C++, to create the Qt integration framework
- A couple of days in SolidWorks, to create the mechanical apparatus to hold the whole thing together

Altogether, creating the tester for the microcontroller sticker took over two months, compared to the two weeks to create the product itself.

Why go through all this effort? Because time is money, and defects and returns are expensive to process. The tester can process one board in under 30 seconds; and in those 30 seconds, the tester has to program two microcontrollers; test sensors for light, sound, and touch; and confirm operation at both 5V and 3V. A manual test for all these operations could take several minutes of skilled labor, and wouldn't be as reliable. Thanks to this tester, we processed zero returns due to defective material. Also, the graphical UI on the tester makes it very easy for the factory to determine exactly which point in the circuit is failing, facilitating fast rework of any imperfect material.

GUIDELINES FOR CREATING A TEST PROGRAM

As a rule of thumb, for every product you make, you're actually making two related products: one for the end user, and a test for the factory. In many ways, the test for the factory has to be as user-friendly and foolproof as the product itself; after all, tests are not run by electrical engineers. But the related testing product will be much quicker and faster to build if adequate testing features are designed into the consumer product.

And no, don't outsource the test program to the factory, even if the factory offers that service. The factory often won't understand your design intent, so their test programs will either be inefficient or test for the wrong behavior. Factories also have an incentive to pass as much material as possible, as quickly as possible, so their test programs tend to be primitive and inadequate.

Here are some guidelines to follow when designing your own program:

Strive for 100 percent feature coverage.

Don't overlook simple or secondary features like status LEDs or an internal voltage sensor. When creating the test list, I take an "outside/inside" approach. First, look at the product from the outside: list every way a consumer can interact with it. Does your test program address every interaction surface, even if only superficially? Is every LED lit, every button pressed, every sensor stimulated, and every memory device touched? Has every bullet point in your marketing material been confirmed? Promising "world-class" RF sensitivity is different from simply advertising the presence of a radio. Then, think about the inside: from the schematic, look at every port and consider key internal nodes to monitor. If the product has a microcontroller, review which drivers are loaded to cross-check the test list, and make sure no components are forgotten.

Minimize incremental setup effort.

Optimize the amount of time required to set up the test for each unit. This is often done through jigs that employ pogo pins or prealigned connector arrays. A test that requires an operator to manually probe a dozen test points with a multimeter or insert a dozen connectors is time-consuming and error-prone. Most factories in China can help design the jig for a nominal cost, but jig design is easier and more effective if the design itself already includes adequate test points.

Automate test execution in a linear execution flow.
An ideal test runs with a single button press, and produces a pass or fail result. In practice, there are always stop points that require operator intervention, but try not to require too much. For example, don't require an operator to key in or select an SSID from a list during each Wi-Fi connectivity test. Instead, fix the test target's SSID and hardcode that value into a test script so the connection cycle is automatic.

Use icons and colors to communicate with operators, not text.
Not every operator is guaranteed to be literate in a given language.

Employ audit logs.
Record test results correlated to device serial numbers by incorporating a barcode scanner into the test rig. Alternatively, have the device print a coupon with a unique, timestamped code or a locally stored audit log to prove which units passed a test. Logs will help you figure out what went wrong when a consumer returns a failed product, and they let you quickly check that all products were tested. After an eight-hour shift of testing, an operator may make mistakes, such as accidentally putting a defective unit into the "good" bin. Being able to check that every shipped product was subjected to and passed the full test can help you identify and isolate such problems.

Provide an easy update mechanism.

Like any program, test programs have bugs. Tests also need to evolve as your product is patched and upgraded. Have a mechanism to update and fix test programs without visiting the factory in person. Many of my test fixtures can "phone home" via a VPN, and I can SSH into the jig itself to fix bugs. Even my simplest jig employs a Linux laptop (or equivalent) at its core. This is in part because Linux is easier to update and maintain than a bespoke microcontroller that requires a special adapter for firmware updates.

These guidelines are easy to implement if your product is designed with testability in mind. Most of the products I design run Linux, and I leverage the processor inside the product itself to run most tests and help manage the test user interface. For products that lack user interaction surfaces, an Android phone or a laptop connected via Wi-Fi or serial can be used to render the test user interface.

Testing Versus Validation

Production tests are meant to check for assembly errors, not parametric variations or design issues. If a test is screening out devices because of normal parametric component variations, either buy better components or redo your design.

For consumer-grade products, you don't need to run a five-minute comprehensive RAM test on every unit. In theory, your product should be designed well enough that if it's all soldered together correctly, the RAM will do its job. A quick test to check that there are no stuck or open address pins is often enough. Name-brand chip vendors typically have very low defectivity, so you're not validating the silicon; rather, you're validating the solder joints and connectors, and checking for missing or swapped components. (But if you buy clone chips or off-brand, remarked, or partially tested devices to cut costs, I recommend making a mini-validation program for those components.)

VALIDATING A SWITCH

To illustrate the difference between production testing and validation, let's look at how both might work for a switch.

A production test for a switch may simply ask the operator to hit the switch a few times and verify that the feel is right, and that electrical contact is made through a simple digital indicator. A validation test, on the other hand, may involve selecting a few devices at random, measuring the switch contact resistance with a multimeter that is accurate to five significant digits (also called a *five-digit multimeter*), subjecting the devices to elevated humidity and temperature for a couple of days, and then putting the devices into an automated jig that cycles the switches 10,000 times. Finally, you might remeasure the switch contact resistance with a five-digit multimeter and note any degradation in close-state contact resistance.

Clearly, this level of validation can't be performed on every device manufactured. Rather, the validation program evaluates the switch's performance over the expected lifetime of the product. The production test, on the other hand, just makes sure the switch is put together right.

NOTE *It's good practice to rerun validation tests on a couple of randomly sampled units out of every several thousand units produced. There are formulas and tables you can use to compute how much sampling you need to achieve a certain level of quality; just search online for "manufacturing validation test table."*

But how much testing is enough? You can derive one threshold for testing through a cost argument. Every additional test run incurs test equipment costs, engineering costs, and the variable cost of the test time. As a result, testing is subject to diminishing returns: at some point, it's cheaper just to take a product return than to test more. Naturally, the testing bar is much higher for medical or industrial-grade equipment, as the liability associated with faulty equipment is also much higher. Likewise, a novelty product meant to be given away may need much less testing.

ON DESIGNING YOUR TEST JIG

As a final thought, always apply solid engineering to your test jig design. When I worked on the chumby 8, there was a problem where a 50-pin flat flex cable adapter was exhibiting random cold-solder-joint failures. I asked the factory to build a test to validate the adapters. Their solution was to hang LEDs from every pin of the adapter, apply a test voltage to one side of the cable, and look for LEDs that didn't light on the other side. The cold solder joints weren't simply open or closed; some just had high resistance. Enough current would flow to light an LED, yet there was also enough resistance to cause a fault in the design.

The factory proposed buying 50 multimeters and attaching them to every pin to check the resistance manually, which would have been expensive and error-prone. It's not reasonable to expect an operator to look at 50 displays hundreds of times a day and be able to reliably find the out-of-spec numbers. Instead, I chose to daisy-chain the connections across the adapter and use a single multimeter to check the net resistance of the daisy chain. By putting the connections in series, I could check all 50 connections with a single numeric measurement, as opposed to the subjective observation of an LED's brightness.

As this case illustrates, there are good and bad ways to implement even a test as simple as checking for cold solder joints on a cable adapter. Ever more complicated components require ever more subtle tests, and there's real value in using engineering skills to craft efficient yet foolproof tests.

FINDING BALANCE IN INDUSTRIAL DESIGN

Even if your product passes all validation tests with flying colors, it still may not be successful if consumers don't want it. Remember: sex sells. To within a factor of two or so, the performance of a CPU or amount of RAM in a box is less important to a typical consumer than how the device looks. Apple devices command a hefty premium in part because of their slick industrial design, and many product designers aim to emulate the success of Sir Jonathan Ive, Apple's chief design officer, in their own products.

There are many schools of thought in *industrial design*, the process of designing how a product will look before actually making it. One school invokes the monastic designer, who creates a beautiful, pure concept, and the production engineers, who spoil the design's purity when they tweak it for functionality. Another school invokes the pragmatist designer, who works closely with production engineers to hammer out gritty compromises to produce an inexpensive and high-yielding design.

In my experience, neither extreme is compelling. The monastic approach often results in an unmanufacturable product that is either late to market or expensive to produce. The pragmatist approach often results in a product that looks and feels so cheap that consumers have trouble assigning it a significant value. The real trick is understanding how to strike a balance between the two, and it begins by getting into the factory and understanding how things are done. Here's a couple of examples of what I've learned about how different factory processes affect that balance, from Chumby and Arduino.

chumby One's Trim and Finish

Trim and finish are difficult, making them points of distinction in a product's appearance. When I worked at Chumby, we wanted the final product to have a minimalist, honest finish. (*Honest finishes* feature the natural properties of the material systems in play, and eschew the use of paints and decals.) Minimalist designs are very hard to manufacture because with fewer features, even tiny blemishes stand out. Honest finishes can be difficult, too, as all the burs, gates, sinks, knits, scoring, and flow lines that are a fact of life in manufacturing are laid naked before the consumer. As a result, this school of design requires well-made manufacturing tools that are constantly checked and maintained throughout production.

If you don't have pockets deep enough to invest in new equipment and capabilities on behalf of your factory (that is, if you're not a *Fortune* 500 company), the first step is to learn the vocabulary available. A *design vocabulary* is defined by the capabilities of the factory or factories producing the goods, like what materials you can obtain, what finish is possible, what tolerances are achievable, and what fastening technology exists. These are all heavily dependent upon the processes available to your factory.

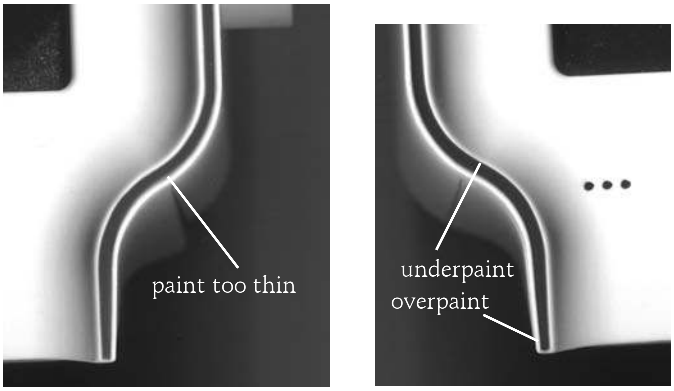
Therefore, I find that visiting a factory in person early in the design process results in a better design. After a factory visit, you'll discard some design vocabulary, but you'll discover some new vocabulary as well. The engineers who work in the factory day in and day out develop process innovations that can open up novel design possibilities that you won't discover unless you visit.

The chumby One is a concrete example of the impact manufacturing processes can have on design outcome. In the original concept art, a blue highlight was added around the front edge to resemble a speech balloon, like those used in captioning comics. The idea was that the chumby would caption your world with snippets from the Internet.

A finished chumby One unit

But applying a blue trim across a raised surface was very hard. The first factory used paint, because the front edge wasn't flat enough to make silk screening an option. *Pad printing* (also known as *tampo printing*, a process in which ink is transferred from a silicone pad to an object) can handle curved surfaces, but the alignment of the ridge on the chumby One wasn't good enough, and the tiniest ink bleed over the edge looked terrible from the side. Decals and stickers likewise couldn't achieve the alignment we wanted. In the end, a small channel was carved to contain the paint, and the factory created the highlight with a stencil and spray paint.

The yield was terrible. In some lots, over 40 percent of the chumby One cases were thrown away due to painting errors. Fortunately, plastic is cheap, so throwing away every other case after painting had a net cost impact of about $0.35.



Two chumby One units with bad paint jobs

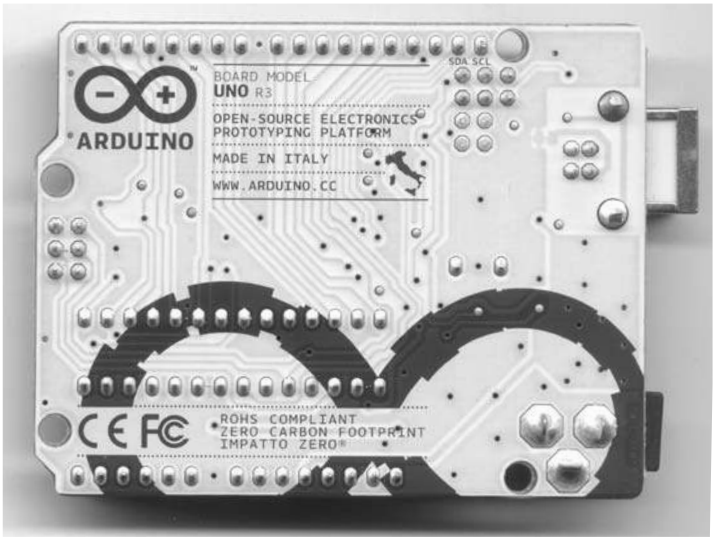
Midway through production, we started producing chumby One units in a second-source facility. The second factory had different plastic molding equipment, and unlike the first factory, this facility could do *double-shot molds*. Double-shot molds involve twice the number of tools of a single-shot injection mold, but they can injection-mold two different colors, or even two different materials, into the same mold. At the new factory, we tried a double-shot process for the thin blue strip instead of painting.

A perfect chumby One ridge, from the double-injection mold process

The results were stunning. Every unit came off the line with a crisp blue line, and no paint meant a more honest, clean finish. But the cost per case jumped to $0.94 apiece with the more expensive process, despite the 100 percent yield. It would have been cheaper to throw away more than half of the painted cases, but even the best painted cases could not compare to the quality of the finish delivered by the double-shot tool.

The Arduino Uno's Silkscreen Art

Another great example of how tweaking a factory process can improve a product's appearance is the Arduino motherboard. The wonderfully detailed artwork on the back side, sporting an outline of Italy and very fine lettering, isn't silkscreen. The factory that makes these boards actually puts on two layers of soldermask: one blue, and one white.



The underside of an Arduino Uno R3

When Arduino boards are manufactured, soldermask is applied through the photolithographic process I described in "Where Arduinos Are Born" on page 44. This process results in artwork with much better resolution, consistency, and alignment than a silkscreen. And since an Arduino's look is the circuit board, this art gives the product a distinctive, high-quality appearance that is difficult to copy using conventional processing methods.

Thus, the process capability of a factory (whether it's painting versus double-shot molding, or double soldermasking versus silkscreening) can have a real effect on a product's perceived quality, without a huge impact on cost. The factory, however, may not appreciate the full potential of its processes, and until a designer interacts with the facility directly, your product can't harness that potential, either.

Unfortunately, many designers don't visit a factory until something has gone wrong. At that point, the tools are cut, and even if you discover a cool process that could solve all your problems, it's often too late.

My Design Process

Design is an intensely personal activity, and as a result, every designer will develop their own process. If you need a framework for developing your own, however, this is the general process I might use to develop a product on a tight, startup budget:

1. Start with a sketchbook. Decide on the soul and identity of the design, and pick a material system and vocabulary that suits your concept. But don't fall in love with it, because it may have to change.
2. Break the design down by material system, and identify a factory capable of producing each material system.
3. Visit the facility, and note what is actually running down the production lines. Don't assume anything based on the one-off units from the sample room. Practice makes perfect, and from the operators to the engineers, factory workers execute procedures they do daily much better than they would an arcane capability they don't use often.

4. Reevaluate your design based on a new understanding of what's possible at the factory, and iterate. Go back to step 1 if small tweaks aren't enough. This is the stage when it's easiest to make compromises without sacrificing the purity of your design.
5. Rough out the details of your design. Pick parting lines where pieces of the case snap together, sliding surfaces, finishes, fastening systems, and so on based on what the factory can do best.
6. Pass a revised drawing to the factory, and work with them to finalize details such as draft angles, fastening surfaces, internal ribbing, and so on.
7. Validate the design using a 3D print and extensive 3D model checks.
8. Identify features prone to tolerance errors, and trim the initial manufacturing tool so that the tolerance favors modifications that will help you minimize costly changes to the tool. For example, consider injection molding, where a steel tool is the negative of the plastic it's molding. Removing steel from a tool (adding plastic) is easier than adding steel (removing plastic), so target the initial test shot to use more steel on critical dimensions, as opposed to too little. A button is one mechanism that benefits from tuning like this: predicting exactly how a button will feel from CAD or 3D prints is hard, and getting the tactile feel perfect usually requires a little trimming of the tool.

Of course, this process isn't a set of hard rules to follow. You may need to add or repeat steps based on your experience with your factory, but if you choose a good factory, this should be a good starting point.

PICKING (AND MAINTAINING) A PARTNER

Just like the wands from *Harry Potter*, a good factory chooses you as much as you choose it, so forget the term *vendor* and replace it with *partner*. If you're doing it right, you aren't simply instructing the factory; there should be a frank dialogue about the trade-offs involved, and how the manufacturing process can be improved. That's the only way to get the best product possible.

A healthy relationship with a factory can also lead to better payment terms, which improves your cash flow. In some cases, factory credit can directly replace raising venture capital, taking loans, or getting Kickstarter funding. As a result, I treat good factories with the same respect as investors and partners in a business. For an idea of what that means, here are some tips on how to choose and work with your factory.

Tips for Forming a Relationship with a Factory

First, pick the right sized factory for your product. If you work with a factory that's too big, you risk getting lost in bureaucracy and pushed out of the production line by bigger customers at critical times. Work with a factory too small, and it won't be able to provide the services you need. As a rule, I pick the biggest facility where I can get direct access to the *lao ban* (factory boss) on a regular basis, because if you can't talk to the boss, you're nobody. It's a good sign if the lao ban is there on the first meeting to give you a tour and asks astute questions about your business over lunch.

Second, follow the adage, "Sunlight is the best disinfectant." If a factory won't quote with an open BOM, where the cost of every component, process, and margin is explicitly disclosed, I won't work with them. Cost reduction discussions cannot function without transparency because there are too many places to bury costs otherwise. Likewise, if cost discussions turn into a game of Whack-a-Mole, where reduced costs on one line item are inexplicably popping up in another, run away.

This final tip applies primarily to startups. In your early stages, everyone knows your cash supplies are finite. Even if you've just closed a big round of financing, swaggering into a factory with money bags is not a sustainable approach. Smart factories know your cash supplies are limited, and if the greatest value you propose to bring to the factory is piles of money, your value is limited; in the best case, it won't really pay out until years down the road when the product is shipping in high volumes. As a result, it's helpful to try to deliver value to the factory in nonmonetary ways.

As silly as it sounds, being a pleasant and constructive person goes a long way in currying the favor of your facility. Manufacturing is a high-stress, low-margin business, and everyone in the facility has to deal with difficult problems all day. I find I get better service—even better than customers with deeper pockets—if I treat my factories as I would treat a friendly acquaintance, and not as slave labor or a mere subcontractor. Mistakes happen, and being able to turn a bad situation into a learning experience will benefit you the day when you make a stupid (and perhaps expensive) mistake.

Tips on Quotations

Openness aside, know that if a quote seems too good to be true, it often is. When negotiating prices with a factory, step back and check if the quote makes sense. Factories that lose money on a deal will stop at nothing to make it back, and many manufacturing horror stories have roots in unhealthy cost structures. A factory's first prerogative is survival, even if that means mixing defective units into lots to boost margin, or assigning novice engineers to a flagging project to better monetize their seasoned engineers on more profitable customers.

As you evaluate a quote, make sure it includes the following:

- The price of each part
- The excess material for the job due to *minimum order quantities (MOQs)*
- Labor costs
- The factory's overhead cost
- *Nonrecurring engineering (NRE)* fees

Let's look at a few of these items in detail.

KEEPING AN EYE ON EXCESS

Excess is the result of what I call the "hot dogs and buns" problem. Hot dogs come in packs of 10, but buns come in packs of 8. Unless you buy 40 servings, you'll have left over buns or hot dogs.

Likewise, many components only come in 3,000-piece reels. A 10,000-piece build requires 4 reels for a total of 12,000 pieces, leaving 2,000 pieces of excess. Factories can buy parts in cut tape or partial reels, but the cost per part of cut tape is much higher, as the risk of excess material is shifted onto the distributor.

Excess isn't all bad, though: it can be folded into future runs of a product. As long as your product sustains a decent production rate, excess component inventory should turn into cash on a regular basis. At some point, however, production will end or pause, and the bill for the excess will arrive, putting a crimp on cash flow. If a quote lacks an excess column, the factory may charge you for the full reel but keep the excess for their own purposes; this is where many of the gray-market goods in Shenzhen come from. They may also just send an unexpected invoice for it down the road, which often arrives at the worst possible time—revenue from the product has already ceased, but bills keep coming in. Either way, it's best to know up front the complete cradle-to-grave business model.

FIGURING OUT LABOR COSTS

Labor costs are devilishly tricky to estimate, but the good news is that for high-tech assemblies, labor is typically a small fraction of total cost. The labor cost of assembling a straightforward board with 200 parts on it in small volumes in China may be about $2 or $3, while the cost of assembling in the US is closer to $20 or $30. Even if labor prices double overnight in China and halve in the US, China may still be competitive.

This is in contrast to the lower-value goods moving out of China (such as textiles), where the base value of the raw material is already low so labor costs are a significant portion of the final product cost. I usually don't argue much over labor costs, since the end result of scrimping on labor is often lowered quality, and pushing too hard on labor costs can force the factory to reduce the workers' quality of life by trimming benefits.

THE FACTORY'S OVERHEAD

Negotiating factory margin is also a bit of an art, and there are no hard and fast rules. I'll give guidance here, but there are always exceptions to the rule, and every factory can cut you a special deal depending on the circumstances. Ultimately, it's important to look at the big picture when reviewing a factory's quote and use some common sense.

What constitutes a fair margin for a factory depends on how much value it adds to your product, and the volume of production. The definition of "margin" also varies depending on the facility. Some facilities include scrap, handling overhead, and even research and development expenses into the margin, while others may break those out on separate lines.

In general, margin ranges between single-digit to low double-digit percentages depending upon volume, value add, and project complexity. For very low-quantity production lots (less than 1,000 pieces), you may also be charged a per-lot *line fee*. This fee partially defrays the cost of setting up an assembly line only to tear it down after a couple of hours. A line's throughput may be very fast, producing hundreds to thousands of units a day, but it also takes days to set up.

NONRECURRING ENGINEERING COSTS

NRE costs are one-time fees required to set up a production run, such a stencils, SMT programming, jigs, and test equipment. Note that reusing test equipment between customers is considered bad practice; if a multimeter is required as part of a production test, don't be surprised if a bill for a multimeter is tacked onto the NRE. Customers have drastically varying standards around the maintenance and use of test equipment, so good factories don't take chances with it.

Miscellaneous Advice

Who you can talk to and how open the factory is about costs are certainly key concerns, but with experience, you'll learn a lot more about dealing with factories that doesn't fall into any particular category. To close, here are a few more important points to keep in mind when selecting a factory.

SCRAP AND YIELD

Ideally, you'd pay a factory only for good, delivered items, and the factory would bear the burden of defective units. This gives the factory an incentive to maintain a high production quality, because every percent of defectiveness eats away at its margin. But if your design has a flaw or is too hard to build, and defectiveness is high, the factory may start shipping lower-quality units as a desperate measure to meet production and margin targets. It may also start selling defective goods on the gray market to recover cost, leading to brand reputation problems down the road.

To avoid situations like that, reach an understanding with the factory ahead of time on how to handle scrap units or exceptional yield loss. This may include, for example, a dedicated "scrap" line item inside the quotation to handle defectiveness explicitly.

ORDER MORE UNITS THAN THE PROVEN DEMAND

Despite everyone's best efforts, mistakes will happen, customers will receive bad devices, and you'll want extra working units for returns and exchanges. Ordering 1,000 pieces to fulfill a 1,000-piece Kickstarter campaign means if customers want to return or exchange units that were broken in shipping, all you can do is issue refunds. It's just not practical to fire up the factory to make a dozen replacement units.

As a general rule, I order a few percent excess beyond the number of units I need to deliver to customers, to have stock on hand to handle returns and exchanges. Units that don't get used up by the returns process can be turned into demo loaners or business development giveaways to drum up the next set of orders!

SHIPPING COSTS MONEY

Keep an eye on shipping costs. These fees aren't typically built into a factory's quotation, but they impact your bottom line, even more so for low-volume products. Shipping FedEx is a great way to save time, but it's also very expensive. Courier fees can easily wash out the profit on a small project, so manage those costs.

NOTE *Couriers offer discounts to frequent shippers, but you have to call in to negotiate the special rates.*

FACTOR IN IMPORT DUTIES

Components imported to China without an import license are levied a roughly 20 percent compulsory duty on their value. The general rule for China is dutiable on import, duty free on export. If something is accidentally shipped across the border to Hong Kong, expect to pay a duty to get it back into China, too.

Get a customs broker to work angles for saving money; for example, some brokers can get goods taxed by their weight and not their value, which for microelectronics is typically a good deal. I haven't figured out all the customs rules, as they seem to be a moving target. Every month it seems there's a new rule, fine, exceptional fee, or tariff to deal with. There are also plenty of shady ways to get goods into China, but I sleep better at night knowing I do my best to comply with every rule.

Quotations don't include duties because factories assume by default that you will have an import license. Import licenses enable the duty-free import of goods. But import licenses cost a few thousand bucks, take weeks to process, and have no room for flexibility, as they are tied to an exact BOM for the product. Small engineering change orders can invalidate an import license. I've known customs officers to count the number of decoupling caps on a PCB, and if it doesn't match the count in the license, a fine is levied and the license is invalidated. Even deviations in the material used to line a decorative box can invalidate a license. In short, this import license scheme favors high-volume products, and punishes low-volume producers, so tread lightly.

CLOSING THOUGHTS

Going to China for manufacturing clearly isn't for everyone. Particularly if you're based in the US, the overhead of courier fees, travel, duties, and late-night conference calls adds up rapidly. As a rule of thumb, a small US-based company is often better off assembling PCBs in the US for volumes less than 1,000 units, and you won't start seeing clear advantages until volumes of perhaps 5,000 to 10,000 units.

That math shifts in China's favor as processes like injection molding and chassis assembly come into play, due to the expertise Chinese factories have in these labor-intensive processes. The break-even point can also be much lower if you live in or near China, as courier fees, travel, and time zone impact are all a small fraction of what they'd be from the US. This compounds with the fact that locals are more effective at leveraging the component ecosystem in China, leading to further cost reductions compared to a design produced using only US parts.

On the other hand, physically large assemblies or systems built using lots of dutiable components may be cheaper to build domestically, as they save on shipping costs and tariffs. In the end, keep an open mind and try to consider all the possible secondary costs and benefits of domestic versus foreign manufacturing before deciding where to park production.

part 2
thinking differently: intellectual property in china

China has a reputation for lax enforcement of intellectual property (IP) laws, and that leads to problems like fake and copycat products. This part of the book takes a nuanced look at China's IP ecosystem, and finds a novel way to reward innovation that serves as an alternative to traditional Western IP practices.

First, consider this question: what, exactly, constitutes a fake? It seems relatively straightforward to answer; anything that's not an original must be a fake. The situation becomes muddied, however, when you consider the possibility that some contract manufacturers produce fakes by running a *ghost shift*, an after-hours production run not reported to the product's brand owner. These items are produced on the same equipment, by the same people, and with the same procedures as the original product, but they're sold directly to customers at a much higher margin to the manufacturer.

In fact, the spectrum of fakes runs an entire gamut of possibilities. Used and damaged goods get upcycled; production rejects with minor flaws are refurbished and sold as originals; original products get relabeled to advertise a higher capability or capacity (for example, memory cards with 4GB actual capacity are sold as 8GB), and so on. Chapter 4 relates several encounters I've had with fake goods in China, and dives into the issues and incentives enabling the rise of such fakes.

Cloning and copying are also common practices in China. A nebulous and sometimes shadowy group of rogue innovators known as *Shanzhai* create products that attempt to mimic the features and function of an original product, often with assistance from the original's blueprints. But the clones are heavily modified to save cost or include unique features. Often, the most offensive aspect of the practice is the use of the original product's brands and trade dress on the clones. Aside from trademark violations, a look inside the products reveals an incredible amount of original engineering and innovation.

Dismissing the Shanzhai as mere thieves and copycats overlooks the fact that they can achieve what few Western companies can: they can build complete mobile phones, and on a shoestring budget to boot. Chapter 5 takes a deep dive into a prime example of Shanzhai engineering, a feature phone designed for emerging markets that costs under $10. The phone is a tour de force of cost reduction and a fresh look at ways of building to address markets that are untouchable with Western engineering practices.

One of the most insightful lean engineering practices enabling the creation of complex systems on a shoestring budget is the Shanzhai method for sharing IP. I'll explore this by comparing and contrasting the Western notion of Open Source with the Shanzhai method, which I refer to as *gongkai*. In Western law, Open Source is a proper noun, referring specifically to an IP sharing system governed by an explicit license to share. This license is granted by the copyright holder, often with significant commercial restrictions. Open Source advocates vigorously defend this notion and are quick to disavow any IP that doesn't explicitly use an approved license.

In gongkai, if you can obtain a copy of the blueprints, you can use them as you please; it doesn't matter who made them. Yet people still share their ideas, because the blueprints act as an advertisement. Blueprints often refer explicitly to certain chips, or have a phone number for the firm that drew them. The creators hope circulating their blueprints will bring business to their factory when people order parts or subassemblies referenced within, or when people call their firm to improve or customize the design. In other cases, blueprints are traded. For example, there are bulletin board exchanges where before you download a blueprint, you must contribute one of your own.

In short, the gongkai IP ecosystem is an evolution of ad-driven business, but applied to hardware. Just as Google provides high-quality search, email, and mapping services for free in exchange for showing ads, Shanzhai innovators share ideas to land follow-up orders in their factories.

Here lies a key distinction between most Western innovators and their counterparts in Shenzhen: everyone who is anyone in Shenzhen owns or has close ties to a factory. The fastest path to material wealth is selling more product. Arguing over who has rights to abstract ideas is a waste of effort best left for *baijiu*-fueled discussions after dinner.* On the opposite end of the spectrum are Western patent trolls so removed from factories that they probably don't even have a soldering

* *Baijiu is a type of strong Chinese alcohol.*

iron, yet they invest millions of dollars into litigation and collecting royalties on ideas they didn't invent.

Neither system is perfect, but the gongkai method is uniquely adapted to the fast pace of technology. In a world where chips get faster and cheaper every couple of years, a 20-year patent lifetime is an eternity. Spending a decade to bring a product to market simply is not an option; the best factories in China can turn a napkin sketch into a prototype in days, and bring it to scale production in weeks. Long patent terms may be appropriate for markets like pharmaceuticals, but in fast-moving markets, investing months and tens of thousands of dollars in lawyer fees to negotiate a license or just apply for a patent can lead to missed opportunities.

Perhaps a discussion on reforming the Western patent system is long overdue. The gongkai ecosystem is living proof that granting 20-year monopolies on ideas as trivial as "slide to unlock" for a smartphone may not be the One True Path to incentivize innovation. I look forward to starting the conversation with this whirlwind tour of the good, the bad, and the ugly of the Chinese IP.

4. gongkai innovation

If the term *intellectual property* sounds like an oxymoron to you, you're not alone. If I give you an apple and say, "This is your apple," what that means is pretty clear. You can do what you want with that apple: you can eat it, sell it, or even use the seeds to plant an apple tree and make more apples, which you can then sell or use to feed your family. But if I hand you a phone and say, "This Apple iPhone is yours," you own the collection of atoms in your hand, but you have extremely limited rights to the software, patents, and trademarks—the intellectual property—associated with that phone. Unlike with the fruit, you can't take what's inside your iPhone and use that knowledge as a seed to make more iPhones.

Intellectual property works very differently in China, though. There, you could (and people do) use a phone as the seed for your own original works. Two experiences I had in China opened my eyes to that fact that there isn't one true path for dealing with intellectual property.

I BROKE MY PHONE'S SCREEN, AND IT WAS AWESOME

My first story begins, as many of my adventures do, with stepping out of a taxi at the Futian border checkpoint going into China. It was May 2014, and I was heading to Shenzhen to hammer out production plans for the Novena open hardware laptop, which I'll talk more about in Chapter 7. As I stepped out of the taxi, my hand caught on my backpack, sending my phone tumbling toward the concrete sidewalk. As the phone smashed into the ground, I heard the dry "thud" of a shattering touchscreen.

There is no better place in the world to break your phone's screen than the border crossing into Shenzhen. Within an hour, I had a new screen installed by skilled hands in Hua Qiang Bei, for just $25—including parts and labor.

I originally planned to replace the screen myself. The phone still worked, so I hastily visited iFixit for details on how to replace the screen, and then booked it to Hua Qiang Bei to purchase replacement parts and tools. The stall I visited quoted me about $120 USD for a new screen, but then the shop owner grabbed my phone out of my hands and launched a built-in self-test program by punching *#0*# into the dialer UI.

She confirmed that there were no bad pixels on my OLED display and that the digitizer was still functional, just cracked. She then offered to buy my broken OLED and digitizer module, but only if her shop could replace my screen. I said that would be fine as long as I could watch to make sure they didn't swap out any other parts.

Of course, they had no problem with that. In 20 minutes, they took my phone apart, removed the broken module, stripped the adhesive from the phone body, replaced the adhesive, fitted the phone with a "new" (presumably refurbished) module, and put it all back together. The process involved a hair dryer (used as a heat gun), copious amounts of contact cleaner (used to soften the adhesive), and a very long thumbnail (in lieu of a spudger/guitar pick). Unfortunately, I couldn't take pictures of the process because the device I would have used to do so was in pieces in front of me.

This is the power of recycling and repair. Instead of paying $120 for a screen and throwing away a functional piece of electronics, I just paid the cost to replace the broken glass. I had assumed that the glass on the digitizer is inseparable from the OLED, but apparently those clever folks in Hua Qiang Bei found an efficient way to recycle those parts. After all, the bulk of the module's cost is in the OLED display. The touchscreen sensor electronics, which are also grafted onto the module, were undamaged by the fall. Why waste perfectly good parts?

And so my phone had a broken screen for all of an hour, and it was fixed for less than the cost of shipping spare parts to Singapore (my country of residence). Experiences like this get me thinking: why aren't there services like this in every country? What makes Shenzhen so unique that you can go from a broken screen to a fixed phone in half an hour for much less than the cost of a monthly phone bill? A multitude of factors contribute to this phenomenon, most of which can be traced to a group of people called the *shanzhai*.

SHANZHAI AS ENTREPRENEURS

The shanzhai of China originally became famous as the producers of knockoffs of products like the iPhone, so they've historically been dismissed by the popular press as simply "copycat barons." But I think they may have something in common with teams like Hewlett and Packard or Jobs and Wozniak, back when they were working out of garages.

Who Are the Shanzhai?

To understand why I think this, it helps to understand the cultural context of the word *shanzhai*. Shanzhai (山寨) comes from the Chinese words *mountain fortress*, but the literal translation is a bit misleading. The English term *fortress* connotes a large fortified structure or stronghold, perhaps conjuring imagery of castle turrets and moats. On the other hand, its denotation states that it is simply a fortified place, and this is closer to the original Chinese meaning, which refers to something like a cave or guerrilla-style hideout.

In its contemporary context, *shanzhai* is a historical allusion to the people that lived in such hideouts, like Song Jiang and his 108 bandits, a group of outlaws who lived in the 12th century. A friend of mine described Song Jiang as a sort of Robin Hood meets Che Guevara. He was a rebel and a soldier of fortune, yet selfless and kind to those in need. The tale is still popular today; my father recognized it instantly when I asked him about it.

Modern shanzhai innovators are rebellious, individualistic, underground, and self-empowered—just like Song Jiang. They're rebellious in the sense that they are celebrated for their copycat products. They're individualistic in the sense that they have a visceral dislike for the large companies. (Many shanzhai are former employees of large companies, both American and Asian, who departed because they were frustrated by the inefficiency of their employers.) They're underground in the sense that once a shanzhai "goes legit" and does business directly through traditional retail channels, they no longer belong to the fraternity of the shanzhai. They're self-empowered in the sense that they're universally tiny operations, bootstrapped on minimal capital, and their attitude is, "If you can do it, then I can as well."

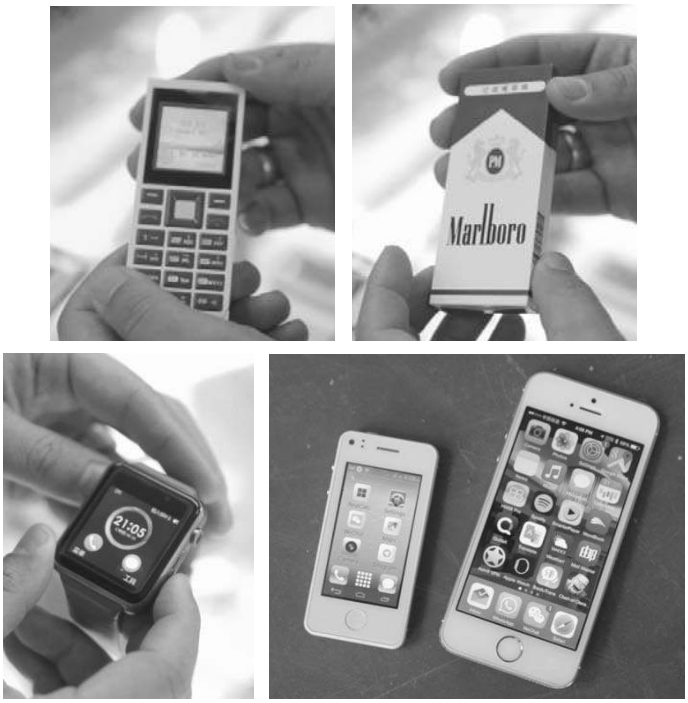
An estimated 300 shanzhai organizations were operating in Shenzhen in 2009. Shanzai shops range from just a couple of folks to a few hundred employees. Some specialize in processes like tooling, PCB design, PCB assembly, or cell phone skinning, while others have broader capabilities.

Since the shanzai are small, they have to be efficient to maximize output. One shop of under 250 employees can churn out over 200,000 mobile phones per month with a high mix of products, sometimes producing runs as short as a few hundred units. Collectively, shanzhai in the Shenzhen area produced an estimated 20 million phones per month in 2009. That's an economy approaching a billion dollars a month. Most of those phones sell into third-world and emerging markets like India, Africa, Russia, and southeast Asia.

More Than Copycats

Significantly, the shanzhai's product portfolio includes more than just copycat phones. They innovate and riff on designs to make original products as well. These original phones integrate wacky features like 7.1 stereo sound, dual SIM cards, a functional cigarette holder, a high-zoom lens, and a built-in UV LED for counterfeit money detection.

The shanzhai do to hardware what the Web did to mashup compilations. Mobile phones that are also toy Ferraris and watch-phone combos (complete with camera!) are good examples: they don't copy any single idea, but rather mix IP from multiple sources to create a new heterogeneous composition, such that the original source material is still distinctly recognizable in the final product. Also, like many web mashups, the result might seem nonsensical to a mass market (like the Ferrari phone), but is extremely relevant to a select long-tail market. In a way, some shanzhai products are just ahead of their time; the watch-phones I saw, for example, predated smartwatches by several years.



Top left and right: front and back of a phone made to look like a pack of cigarettes. Bottom left: an Android-based smart watch, which, unlike the Apple Watch, includes a phone in the watch. Bottom right: a Shanzhai-produced "baby iPhone," shown next to an Apple iPhone 6 for scale.

Community-Enforced IP Rules

The shanzhai also employ a concept called the *open BOM*: when one shanzhai builds something new, they share the bill of materials and other design documents with the others. If the product is based on an existing product, any improvements they make are also shared. These rules are policed by word of mouth within the community to the extent that if someone is found cheating, they are ostracized by the shanzhai ecosystem.

This system is viewed very positively in China. For example, I once heard a local say it was great that the shanzhai could not only clone an iPhone, but also improve upon the original by giving the clone a user-replaceable battery. US law would call this activity illegal and infringing, but given the fecundity of mashup culture on the web, I can't help but wonder if hardware mashup isn't a bad thing. There's definitely a perception in the United States that if it's strange and it happens in China, it must be bad. This bias casts a long shadow over objective evaluation of a cultural phenomenon that could eventually be very relevant to the United States.

In a sense, the shanzhai are brethren of the classic Western notion of hacker-entrepreneurs, but with a distinctly Chinese twist. My personal favorite shanzhai story is about a chap who owns a three-story house that I envy extraordinarily. His bedroom is on top, the middle floor is a complete SMT manufacturing line, and the bottom floor is a retail outlet for the products produced a floor above and designed in his bedroom. Talk about a vertically integrated supply chain! Owning infrastructure like that would certainly disrupt the way I innovate. I could save on production costs, reduce my prototyping time, and turn inventory around aggressively, thereby reducing inventory capital requirements. And if my store were in a high-traffic urban location, I could also cut out the 20–50 percent minimum retail margin typically required by US retailers.

I have a theory that when the amount of knowledge and the scale of the markets in Shenzhen reach critical mass, the Chinese will stop being simply workers or copiers. They'll take control of their destinies, and ultimately, become innovation leaders. These stories about the shanzhai and their mashups are just the tip of an iceberg with the potential to change the way business is done—perhaps not in the United States, but certainly in that massive, untapped market often referred to as "the rest of the world."

THE $12 PHONE

Mashup cell phones demonstrate the shanzhai's innovation and willingness to experiment. But despite all the bells and whistles, those phones are quite affordable. One question you might ask, then, is how cheaply can you make a phone?

A short jaunt to the northeast corner of the Hua Qiang electronics district brings you to the Mingtong Digital Mall. It's a four-story maze packed with tiny shops hawking all manner of quirky phones with features useful in economies that lack the infrastructure of consistent electricity or cable networks. For instance, some phones can run for a month thanks to comically oversized batteries. Others have analog TV tuners, integral hand-crank chargers, and multiple user profiles, enabling a family or small village to share a single phone.

During a visit to Hua Qiang in 2013, I paid $12 for a complete phone, featuring quad-band GSM, Bluetooth, MP3 playback, an OLED display, and a keypad for the UI. It's nothing compared to a smartphone, but it's useful if you're going out and worried about your primary phone getting wet or stolen. And for a couple billion people, it may be the only phone they can afford.

Keep in mind this is the contract-free price. In countries that allow carriers to lock phones, such as the United States, phones are often given away or sold to buyers at a fraction of their cost in exchange for a subscription contract often worth several times the phone's value. The fact that I paid $12 over the counter for a contract-free, nonpromotional, unlocked, new-in-box phone with a charger, protective silicone sleeve, and cable means that the phone's production cost has to be somewhere below the retail price of $12. Otherwise, the phone's maker would be losing money. Rumors placed its cost below $10.

My simple but functional $12 phone

This is a really amazing price point. That's about the price of a large Domino's cheese pizza, or a decent glass of wine in an urban US restaurant. It's even cheap compared to an Arduino Uno. Admittedly, the comparison is a little unfair, but humor me and take a look at the specs for both, shown here:

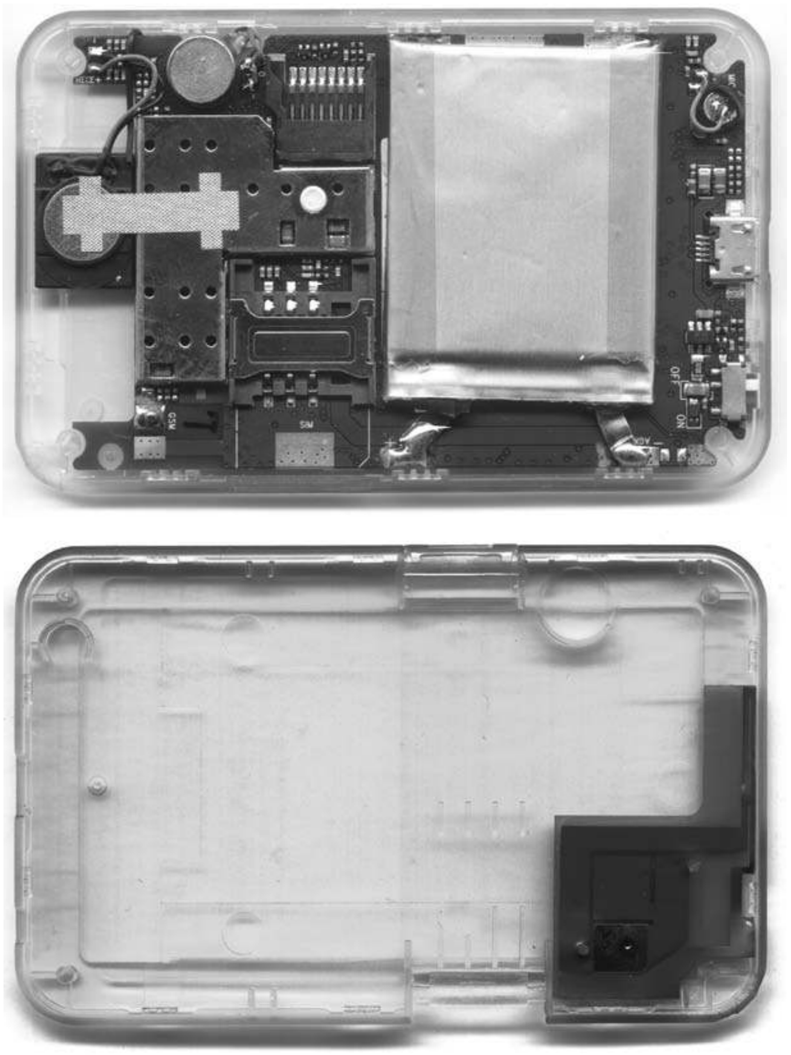
Comparing the $12 Phone with an Arduino

Spec	**This phone**	**Arduino Uno**
Price	$12	$29
CPU speed	260 MHz, 32-bit	16 MHz, 8-bit
RAM	8MiB	2.5kiB
Interfaces	USB, microSD, SIM	USB
Wireless	Quadband GSM, Bluetooth	—
Power	Li-Poly battery, includes adapter	External, no adapter
Display	Two-color OLED	—

How is it possible that this phone has better specs than an Arduino and costs less than half the price? I don't have the answers, but I'm trying to learn them. Tearing down the phone yielded a few hints.

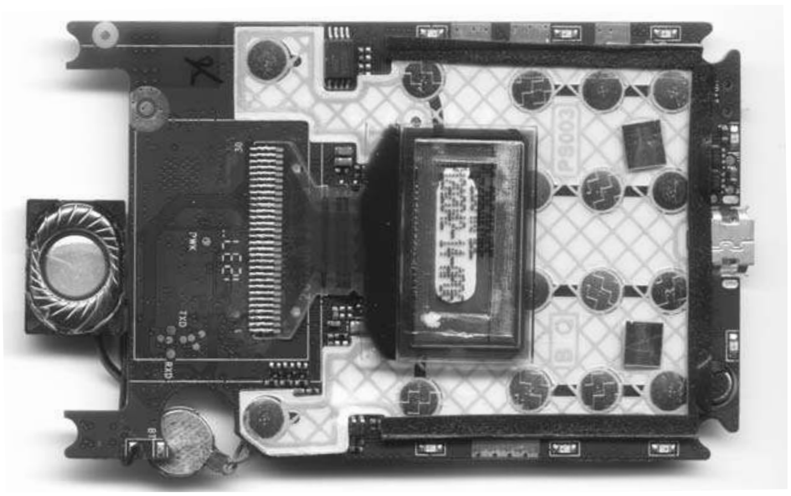
Inside the $12 Phone

First, there are no screws in this phone. The whole case snaps together.

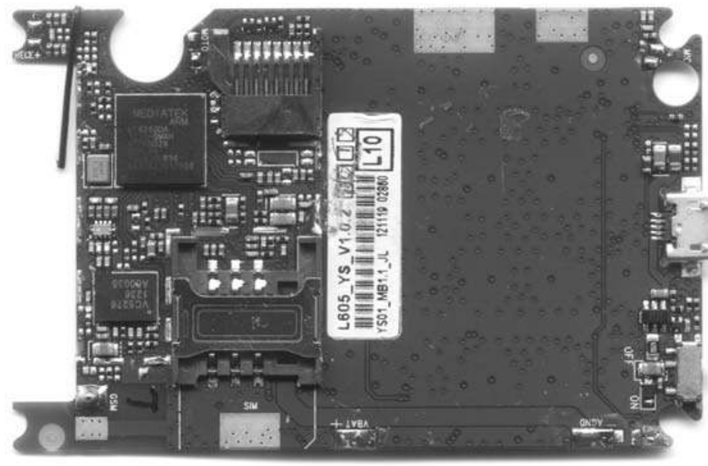


The back of the phone, after the cover is removed

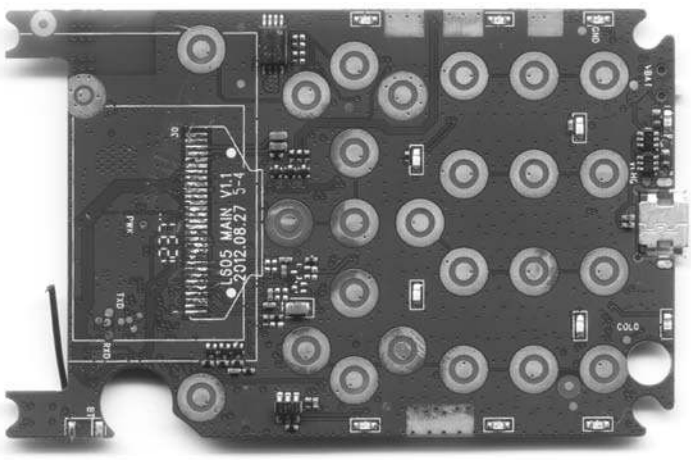
There are (almost) no connectors on the inside. For shipping and storage, you get to flip a switch to hard-disconnect the battery. As best as I can tell, the battery also has no secondary protection circuit. Still, the phone features accoutrements such as a backlit keypad and decorative lights around the edge.



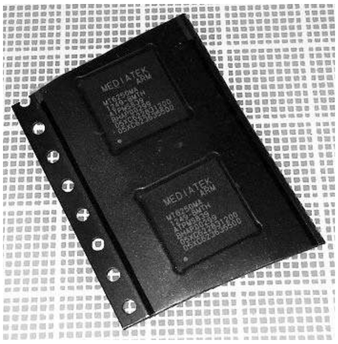
Everything from the display to the battery is soldered directly to the board.



There are little decorative LEDs all over this PCB.


The Bluetooth antenna is the small length of wire on the upper right.

The electronics consist of just two major ICs: the Mediatek MT6250DA and a Vanchip VC5276. The MT6250 is rumored to sell in volume for under $2. I was able to anecdotally confirm the price by buying a couple of pieces on cut tape from a retail broker for about $2.10 each.* That beats the best price I've ever been able to get on an ATMega of the types used in an Arduino. With price competition like this, Western firms are suing to protect ground: Vanchip got into a bit of a legal tussle with RF Micro, and Mediatek has been subject to a few lawsuits of its own.


Two Mediatek MT6250 ICs

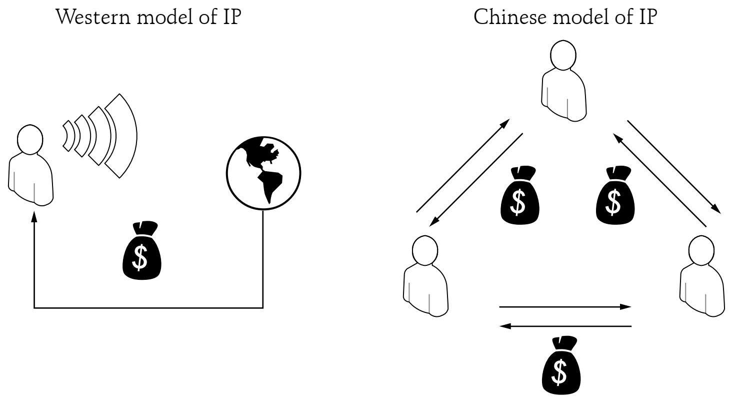
* *No, I will not broker these chips for you.*

Of course, you can't just call up Mediatek and buy these chips. It's extremely difficult to engage with them "going through the front door" to do a design. However, if you know a bit of Chinese and the right websites, you can download schematics, board layouts, and software utilities for something similar to this phone, possibly with some different parts . . . for "free." Free is in quotes because you could obtain the source code, but not the unambiguous legal right to use it, as the source code was distributed without the explicit legal consent of the copyright holders. But anyone unconcerned or unfamiliar with such legal frameworks could build versions of this phone, with minimal cash investment. It feels like open source, but it's not: it's a different kind of open ecosystem.

Introducing Gongkai

Welcome to the Galapagos of Chinese "open" source. I call it *gongkai* (公开), which is the Chinese transliteration of the English *open*, as applied to *open source*. There's a literal translation for *open source* into Chinese (*kaiyuan*), but the only similarity between gongkai practices and Western open source practices is that both allow you to download source code; the legal and cultural frameworks that enable such sharing couldn't be more different. It's like convergent evolution, where two species may exhibit similar traits, but the genes and ancestry are totally different.

Gongkai refers to the fact that copyrighted documents, sometimes labeled "confidential" and "proprietary," are made known to the public and shared overtly, but not necessarily according to the letter of the law. This copying isn't a one-way flow of value, as it would be in the case of copied movies or music. Rather, these documents are the knowledge base someone would need to build a phone using the copyright owner's chips, and sharing the documents promotes sales of their chips. There is ultimately a quid pro quo between the copyright holders and the copiers.



Comparing IP models

This gray relationship between companies and entrepreneurs is just one manifestation of a much broader cultural gap between the East and the West. The West has a "broadcast" view of IP and ownership: good ideas and innovation are credited to a clearly specified set of authors or inventors, and society pays them a royalty for their initiative and good works. China has a "network" view of IP and ownership: one attains the far-reaching sight necessary to create good ideas and innovations by standing on the shoulders of others, and people trade these ideas as favors. In a system with such a loose attitude toward IP, sharing with the network is necessary, as tomorrow your friend could be standing on your shoulders, and you'll be looking to them for favors.

In the West, however, rule of law enables IP to be amassed over a long period of time, creating impenetrable monopoly positions. That's good for the guys on top but tough for upstarts, causing a situation like the modern Western cell phone market. Companies like Apple and Google build amazing phones of outstanding quality, and startups can only hope to build an "appcessory" for their ecosystem.

I've reviewed business plans for over 100 hardware start-ups, and the foundations for most are overpriced chipsets built with antiquated process technologies. I'm no exception to this rule; the Novena uses a Freescale (now NXP after an acquisition) i.MX6 processor, which was neither the cheapest nor the fastest chip on the market when I designed the laptop. But it's a chip with two crucial qualities: anyone can freely download almost complete documentation for it, and anyone can buy it on DigiKey.

Scarce documentation and scarce supply for cutting-edge technology forces Western hardware entrepreneurs to look primarily at Arduino, Beaglebone, and Raspberry Pi as starting points for their good ideas. Chinese entrepreneurs, on the other hand, churn out new phones at an almost alarming pace.

Every object pictured here is a phone.

Phone models change on a seasonal basis. Entrepreneurs experiment all the time, integrating wacky features into phones, such as cigarette lighters, extra-large battery packs (to charge a second phone), huge buttons (for the visually impaired), call-home buttons only (to give to children for emergencies), watch form factors, and so on. This works because small teams of engineers can obtain complete design packages for working phones—including the case, board, and firmware—allowing them to fork the design and focus only on changing the pieces they really care about.

As a hardware engineer, I want that.

I want to be able to fork existing cell phone designs. I saw the $12 phone, and I, too, wanted to use a 364 MHz 32-bit microcontroller with megabytes of integrated RAM and dozens of peripherals that costs $3 in single quantities. The Arduino Uno's ATMega microcontroller, a 16 MHz 8-bit microcontroller with a few kilobytes of RAM and a smattering of peripherals, pales in comparison, yet costs twice as much at $6.

From Gongkai to Open Source

So, I decided to take my study of the phone one step further from a teardown, and attempt to make my own version—in the style of the shanzhai, but interpreted through Western eyes. That's how Sean "xobs" Cross and I started a project we dubbed *Fernvale*. Sean has been my adventure partner on dozens of projects since we first met at Chumby, where I recognized his talent as a firmware engineer when he showed me how he ported Quake to chumby in his spare time. Sean has always marched to the beat of his own drum. Born in Germany to American parents, he studied cognitive science in college, and prior to working at Chumby, he spent six months wandering New Zealand and Australia, searching for adventure and work. At Chumby, he was easy to spot, thanks to his ponytail and kilt (actually, a "Utilikilt").

After Chumby went out of business, Sean and I found ourselves washed up on the shores of Singapore, where I started a boutique hardware consulting firm called Sutajio Ko-Usagi, which is "bunniestudios" translated to Japanese and then romanized into English characters. Sean's virtuoso coding abilities have been an excellent complement to my hardware design skills, and since then, we've completed several significant open source projects.

We figured at first we should at least try to go "through the front door" and inquire directly with the chipmakers about what it might take to get a proper Western-licensed embedded development kit (EDK) for the chips used in these shanzhai phones. Our inquiries were met with a cold shoulder. I was told the volumes for our little experiment were too small, or we'd have to enter minimum purchase agreements backed by a prohibitive cash deposit in the hundreds of thousands of dollars.

Even for people who jump through such hoops, these EDKs don't include all the reference material the Chinese get to play with. The datasheets are incomplete, and you're forced to use the companies' proprietary OS ports. It feels like a case of the nice guys finishing last. Could we find a way to get ahead, yet still play nice?

Engineers Have Rights, Too

Thus, Fernvale had two halves: the technical task of reverse engineering and re-engineering the phone, and the legal task of creating a general methodology for absorbing Gongkai IP into the Western ecosystem. I'll recount the technical task in Chapter 9, which falls into the "Reverse Engineering" part of the book, and focus on the legal task for the remainder of this chapter.

After some research into the legal frameworks and challenges, I believed I'd found a path to repatriate some of the IP from Gongkai into proper open source. I must, however, give a disclaimer: I'm not a lawyer. I'll tell you my beliefs, but don't construe them as legal advice.*

* *I've often wondered why the "I am not a lawyer" disclaimer is necessary. It was explained to me that even the appearance of dispensing legal advice without the disclaimer can make me guilty of practicing law without a proper license. I could also be held accountable for bad decisions made by people who construe the opinions as legal advice.*

My basic idea with Fernvale was to exercise the right to reverse-engineer in a careful, educated fashion to increase the likelihood that, if push came to shove, the courts would agree with my actions. But I also feel that shying away from reverse engineering simply because it's controversial is a slippery slope: you must exercise your rights to have them. If women didn't vote and black people sat in the back of the bus because they were afraid of controversy, the United States would still be segregated and without universal suffrage. Although reverse engineering is a trivial issue compared to racial equality and universal suffrage, the precedent is clear: in order to have rights, you must be bold enough to stand up and assert them.

DEALING WITH PATENTS AND OTHER LAWS

Open source has two broad categories of IP issues to deal with: patents and copyrights. Patents present complex issues, and it seems the most practical approach is to essentially punt on the issue. For instance, nobody, as far as I know, checks their Linux commits for patent infringement before upstreaming them, and in fact, many corporations have similar policies at the engineering level.

Why? Determining which patents apply and if a product infringes take a huge amount of resources. Even after expending those resources, you can't be 100 percent sure. Further, becoming very familiar with the body of patents amplifies the possibility that any infringement is willful, thus tripling damages. Finally, it's not even clear where the liability for infringement lies, particularly in an open source context.

Thus, Sean and I did our best not to infringe with Fernvale, but we couldn't be 100 percent sure that no one would allege infringement. However, we did apply a license to our work that includes a "poison pill" clause for patent holders who might attempt to litigate. Poison pills make the entire body of open source work unavailable to any party who files a lawsuit alleging infringement of any part against any entity.*

For copyrights, the issue is also extremely complex. The Coders' Rights Project from the Electronic Frontier Foundation (EFF) has a Reverse Engineering FAQ (*https://www.eff.org/issues/coders/reverse-engineering-faq/*) that's a good read if you really want to dig into the issues. To sum it up, courts have found that reverse engineering to understand the ideas embedded in code and to achieve interoperability is fair use. As a result, anyone likely has the right to study the Gongkai-style IP, understand it, produce a new work, and apply a Western-style Open IP license to it.

However, before I could attack the copyright issues for Fernvale, I had to make sure we wouldn't bump into other laws that could impede our fair use rights. First, there's the DMCA. The DMCA makes circumventing any encryption designed to enforce a copyright basically illegal, with only a few poorly tested exemptions allowed. Since none of the files or binaries Sean and I downloaded were encrypted or had access controlled by any technological measure, we didn't have to do any circumvention. No circumvention, no DMCA problem.

All the files we obtained came from searches linking to public servers, so there would be no Computer Fraud and Abuse Act (CFAA) problems. None of the devices we used in the work came with shrink-wraps, click-throughs, or other end-user license agreements (EULAs), terms of use, or other agreements that could waive our rights.

DEALING WITH COPYRIGHTS

With the DMCA, CFAA, and EULA concerns set aside, we were finally able to address the core issue: what to do about copyrights.

* *Specifically, Apache 2.0, section 3 reads: Grant of Patent License. [...] If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.*

The cornerstone of our methodology hinged on decisions rendered on several occasions by courts stating that facts are not copyrightable. For example, Justice O'Connor wrote in *Feist Publications, Inc. v. Rural Telephone Service Co., Inc* (449 U.S. 340, 345, 349 (1991) that.*

> Common sense tells us that 100 uncopyrightable facts do not magically change their status when gathered together in one place. . . . The key to resolving the tension lies in understanding why facts are not copyrightable: The sine qua non of copyright is originality.

And:

> Notwithstanding a valid copyright, a subsequent compiler remains free to use the facts contained in another's publication to aid in preparing a competing work, so long as the competing work does not feature the same selection and arrangement.

Based on this opinion, anyone has the right to extract facts from proprietary documentation, and carefully re-express those facts in their own selection and arrangement. Just as the facts that "John Doe's phone number is 555-1212" and "John Doe's address is 10 Main St." are not copyrightable, facts such as "The interrupt controller's base address in 0xA0060000" and "Bit 1 controls status reporting of the LCD" aren't copyrightable, either. Sean and I extracted such facts from datasheets and re-expressed them in our own header files where, as the legal owners of newly created expressive speech, we applied a proper open source license of our choice.

MAKING A PROGRAMMING LANGUAGE

But the situation was further complicated by hardware blocks we had absolutely no documentation for. In some cases, we couldn't even learn what a block's registers meant or how the blocks functioned from a datasheet. For these blocks, we isolated and extracted the code responsible for initializing their state. We then reduced this code into a list of address and

* *See also Sony Computer Entertainment, Inc. v. Connectix Corp., 203 F. 3d 596, 606 (9th Cir. 2000) and Sega Enterprises Ltd. v. Accolade, Inc., 977 F.2d 1510, 1522-23 (9th Cir. 1992).*

data pairs, and expressed it in a custom scripting language we called *scriptic*. We invented our own language to avoid subconscious plagiarism—it's too easy to read one piece of code and, from memory, code something almost exactly the same. By transforming the code into a new language, we're forced to consider the facts presented and express them in an original arrangement.

Scriptic is basically a set of assembler macros, and the syntax is very simple. Here is an example of a scriptic script:

```
#include "scriptic.h"
#include "fernvale-pll.h"

sc_new "set_plls", 1, 0, 0

  sc_write16 0, 0, PLL_CTRL_CON2
  sc_write16 0, 0, PLL_CTRL_CON3
  sc_write16 0, 0, PLL_CTRL_CON0
  sc_usleep 1

  sc_write16 1, 1, PLL_CTRL_UPLL_CON0
  sc_write16 0x1840, 0, PLL_CTRL_EPLL_CON0
  sc_write16 0x100, 0x100, PLL_CTRL_EPLL_CON1
  sc_write16 1, 0, PLL_CTRL_MDDS_CON0
  sc_write16 1, 1, PLL_CTRL_MPLL_CON0
  sc_usleep 1

  sc_write16 1, 0, PLL_CTRL_EDDS_CON0
  sc_write16 1, 1, PLL_CTRL_EPLL_CON0
  sc_usleep 1

  sc_write16 0x4000, 0x4000, PLL_CTRL_CLK_CONDB
  sc_usleep 1

  sc_write32 0x8048, 0, PLL_CTRL_CLK_CONDC
  /* Run the SPI clock at 104 MHz */
  sc_write32 0xd002, 0, PLL_CTRL_CLK_CONDH
  sc_write32 0xb6a0, 0, PLL_CTRL_CLK_CONDC
  sc_end
```

This script initializes the Phase Locked Loop (PLL, a circuit for generating clock waveforms) on the target chip for

Fernvale, the Mediatek MT6260. To contrast, here are the first few lines of the code snippet from which that scriptic code was derived:

```
// enable HW mode TOPSM control and clock CG of PLL control

*PLL_PLL_CON2 = 0x0000; // 0xA0170048, bit 12, 10 and 8 set to 0
                        // to enable TOPSM control
                        // bit 4, 2 and 0 set to 0 to enable
                        // clock CG of PLL control
*PLL_PLL_CON3 = 0x0000; // 0xA017004C, bit 12 set to 0 to enable
                        // TOPSM control

// enable delay control
*PLL_PLLTD_CON0= 0x0000; // 0x A0170700, bit 0 set to 0 to
                         // enable delay control

//wait for 3us for TOPSM and delay (HW) control signal stable
for(i = 0 ; i < loop_1us*3 ; i++);

//enable and reset UPLL
reg_val = *PLL_UPLL_CON0;
reg_val |= 0x0001;
*PLL_UPLL_CON0  = reg_val; // 0xA0170140, bit 0 set to 1 to
                           // enable UPLL and
                           // generate reset of UPLL
```

The original code actually goes on for pages and pages, and even this snippet is surrounded by conditional statements, which we culled as they were irrelevant to initializing the PLL correctly.

Knowledge of our rights, a pool of documentation to extract facts from, and scriptic were tools in our armory. With them, Sean and I derived sufficient functionality for our Fernvale project to eventually boot a small, BSD-licensed, real-time operating system (RTOS) known as NuttX running on our own custom hardware. I'll go more into the gory details of how we did that in Chapter 9.

KNOW AND EXERCISE YOUR RIGHTS

Rights atrophy and get squeezed out by competing interests if they aren't vigorously exercised. Sean and I did Fernvale because we think it's imperative to exercise our fair use rights to reverse-engineer and create interoperable, open source solutions. For decades, engineers have sat on the sidelines and seen ever more expansive patent and copyright laws shrink their latitude to learn freely and to innovate. I'm sad that the formative tinkering I did as a child is no longer a legal option for the next generation of engineers.

The rise of the shanzhai and their amazing capabilities is a wake-up call. I see it as evidence that a permissive IP environment spurs innovation, especially at the grassroots level. If more engineers learn their fair use rights and exercise them vigorously and deliberately, perhaps this can catalyze a larger and much-needed reform of the patent and copyright system. Our Fernvale project is hopefully just a signpost pointing the way for much bigger efforts to bridge the gap between the gongkai and open source communities.

Being able to cherry-pick the positive aspects of gongkai into the Western IP ecosystem is an important tool. Rule of law has its place, and an overly permissive system has its own problems. The next chapter explores some of the negative consequences of an overly permissive IP ecosystem: fake and counterfeit goods.

5. fake goods

The gongkai system fosters an amazing amount of innovation in China, and the shanzhai can make interesting original products, like the cell phones I showed you in Chapter 4. That said, China does produce plenty of fake electronic goods, and they aren't all knockoff iPhones. Clever counterfeiters can produce fake integrated circuits, including microSD cards and even FPGAs.

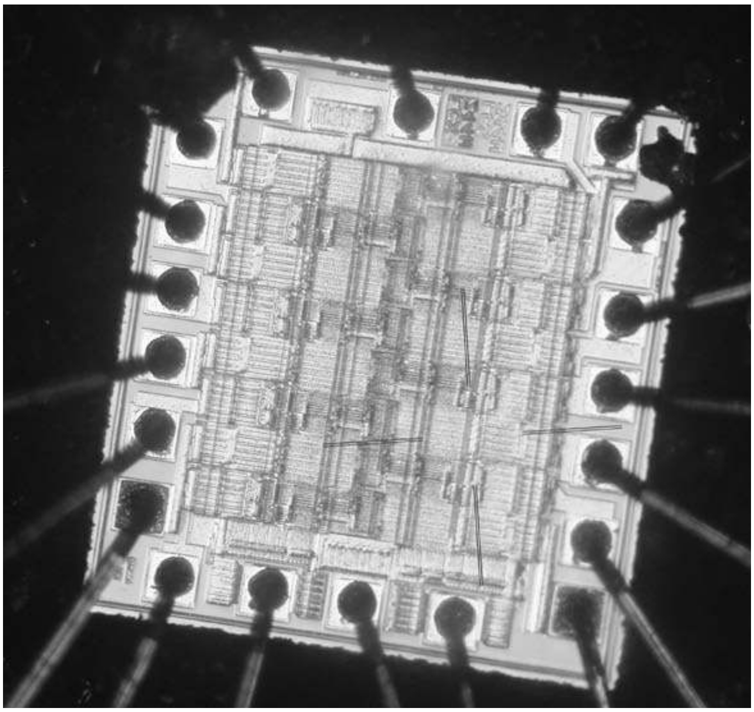
WELL-EXECUTED COUNTERFEIT CHIPS

For instance, in 2007 (while I was still working with Chumby) I encountered some counterfeit chips so well executed that I couldn't be certain they were fake without investigating.

Two suspicious chip specimens from an Asian source

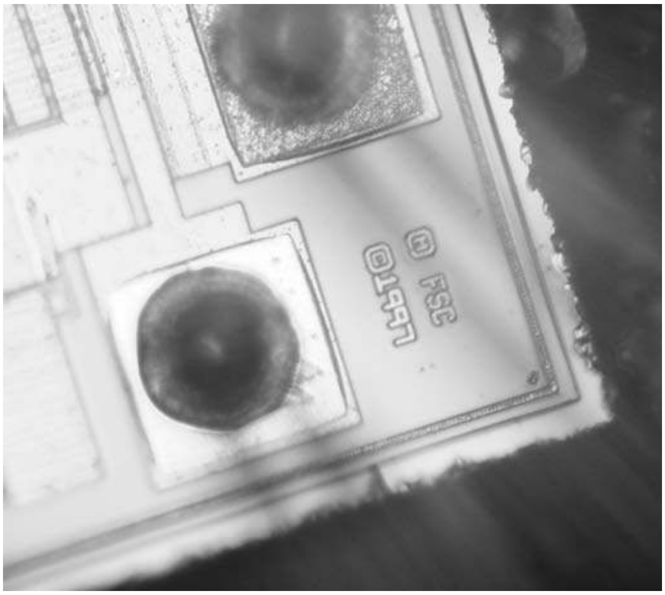
The chips claimed to be ST19CF68s, a chip made by STMicroelectronics and described on its datasheet as a "CMOS MCU Based Safeguard Smartcard I/O with Modular Arithmetic Processor." ST19CF68 chips are normally sold prepackaged in *smartcard* (for example, the chip on the front of a credit card) or *diced wafer* (a silicon wafer that's been diced into individual chips, but with no other package around it) format, but curiously, these were SOIC-20 packaged devices. To find out the reason for the odd package choice, I dissolved the black epoxy packaging off the top of one chip to decapsulate it so I could inspect the silicon on the inside using a microscope.

The die inside the package was much too small and simple for a complex microcontroller unit (MCU) matching the description of the ST19CF68. The pattern of gold-colored rectangles tiled across the chip was too coarse; I could make out individual transistors at low zoom with an optical microscope. The size of these features is referred to as the chip's *process geometry*. The process geometry of a smartcard would typically trail a cutting-edge CPU by at most three or four generations, making transistors very difficult to resolve even at the highest levels of zoom.



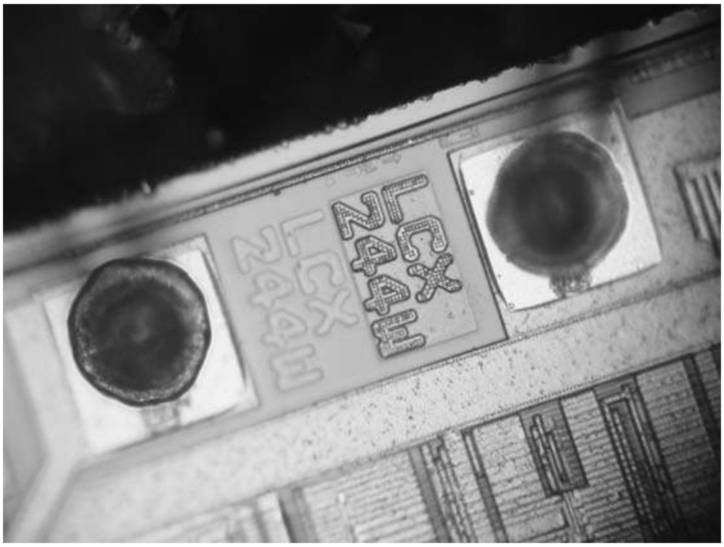
The silicon inside the fake ST19CF68

Along with the unexpectedly coarse process geometry, why did this part have 20 bondable pads and 20 pins, when according to the datasheet, it should have only 8 pads? Zooming in a bit on the die revealed some interesting details.



The chip manufacturer and copyright date

The chip wasn't made by STMicroelectronics after all! The label on the silicon said *FSC*, indicating it was made by Fairchild Semiconductor. Of course, then I had to check the part label on the silicon, too.



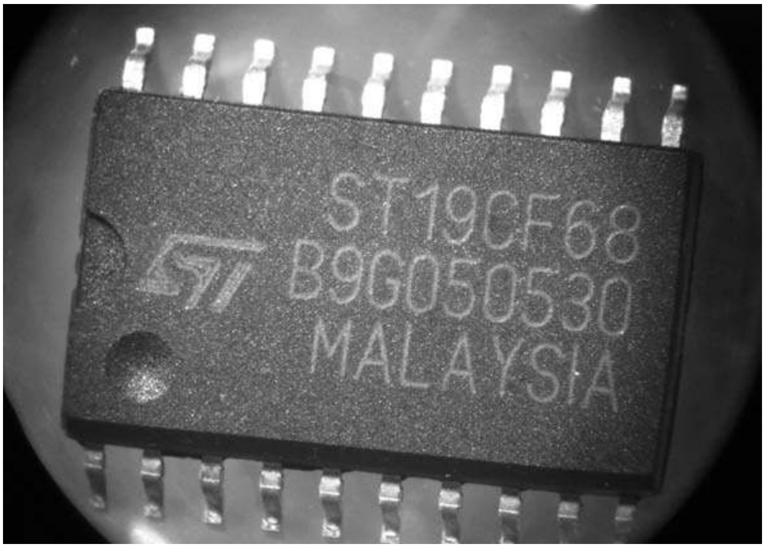
Discovering the true part number

The die within that chip turned out to be a Fairchild 74LCX244, which is a "Low Voltage Buffer/Line Driver with 5V Tolerant Inputs and Outputs." The 74LCX244 is a much cheaper piece of silicon than the ST19CF68 the package supposedly contained.

Of course, the mismatched pin count was suspicious, but manufacturers have been known to put chips in larger packages, especially during early runs of the chip before it has been size-optimized. The thing that really got me was the convincing quality of the package and the markings.

Normally, remarked or fake chips look cheesier than this one. The original chips are sanded down or painted over to remove the previous markings, and the new marking is typically applied with silkscreened paint.

But these chips showed no evidence of remarking at all. The markings are of first-run quality: someone acquired unmarked blanks of the 74LCX244 chip and programmed a production laser engraver to put high-quality fake markings on an otherwise virgin package. They even got the proportions of the *ST* logo exactly right.



A close-up of the outside of the fake ST19CF68

The quality difference between a remarked chip and first-run marking is like the quality difference between spray paint used to hide a scratch on a car and the car's original, factory-fresh paint job. This chip definitely had the "new car" look.

This discovery left me with a lot of unanswered questions. How did someone acquire unmarked Fairchild silicon? Was the person an insider, or did Fairchild sloppily throw away unmarked reject chips without grinding them up or clipping off leads so they couldn't be picked out of a dumpster and resold? The laser-marking machine used to make those markings wasn't a cheap desktop engraver, either; it had to be a high-power raster engraver, and the artwork was spot-on.

I still find it hard to believe those fake chips were made and sold, but maybe I shouldn't. I've seen brazen remarking of dual inline memory modules (DIMMs, the memory used in personal computers) in SEG Electronics Market, and many counterfeiters at the market openly display their arsenal of professional-quality thermal transfer label printers and hologram sticker blanks.

If fakes of this quality become more common, they could present a problem for the supply chain. Clearly, whoever made the counterfeit ST19CF68 can fake just about any chip, and the fakes are gradually appearing on the US market. Resellers, especially distributors that specialize in buying excess manufacturer inventory, implicitly trust the markings on a chip.

I don't think chipmakers will put anticounterfeiting measures on chip markings, but the quality of these fakes definitely made me wary when I discovered them, and it still does. Not all fakes get spotted before they're used, and fake components pose problems in any project where they appear.

COUNTERFEIT CHIPS IN US MILITARY HARDWARE

Counterfeit chips can be particularly problematic when they find their way into military projects. The US military has a unique problem: it's one of the biggest and wealthiest buyers of really old parts, because military designs have shelf lives of decades. Like anything else, the older a part is, the harder it is to find, and sometimes contractors are sold fakes. For example, a 2011 Senate hearing report revealed that some parts used in the P-8 Poseidon (a plane the US Navy commissioned from Boeing) were, as an article from the Defense Tech website put it, "badly refurbished," causing a key system to fail.

The US government attempted to reduce fakes in its supply chain with Amendment 1092 to the National Defense Authorization Act for Fiscal Year 2012. The amendment is a well-intentioned but misguided provision outlining measures designed to reduce the prevalence of counterfeit chips in the US military supply chain.

Even before Amendment 1092 was put on the table, the Defense Authorization Act drew flack for a provision that authorizes the US military to detain US citizens indefinitely without trial. It also rather ironically requires an assessment of the US federal debt owed China as a potential "national security risk" (section 1225 of HR1540).

Under the anticounterfeit amendment, first-time offenders can receive a $5 million fine and 20-year prison sentence for individuals, or a $15 million fine for corporations—a penalty comparable to that of trafficking cocaine.* While the amendment explicitly defines *counterfeit* to include refurbished parts represented as new, the wording is regrettably vague on whether you must be willfully trafficking such goods to also be liable for such a stiff penalty.

** See Sec 2320 (b) at https://www.govtrack.us/congress/bills/112/hr1540/text.*

If you took a dirty but legitimately minted coin and washed it so that it looked mint condition, nobody would accuse you of counterfeiting. Yet this amendment puts a 20-year, $5 million penalty not only on the act of counterfeiting chips destined for military use, but also potentially on the unwitting distribution of refurbished chips that you putatively bought as new. Unfortunately, in many cases an electronic part can be used for years with no sign of external wear.

The amendment also has a provision to create an "inspection program":

> (b) Inspection of Imported Electronic Parts —
>
> (1) … the Secretary of Homeland Security shall establish a program of enhanced inspection by U.S. Customs and Border patrol of electronic parts imported from any country that has been determined by the Secretary of Defense to have been a significant source of counterfeit electronic parts …

Inspecting fruits and vegetables as they enter the country for pests and other problems makes sense, but requiring customs officers to become experts in detecting fake electronic components seems misguided. Burdening vendors with detecting fakes when there are such high penalties for failure is also misguided, given how easy it is for forgers to create high-quality counterfeits.

Types of Counterfeit Parts

To better understand the magnitude of the chip counterfeiting problem, let's look at how fakes are made. The fake chips I've seen fall into the following broad categories.

EXTERNAL MIMICRY

The most trivial counterfeit chips are simply empty plastic packages with authentic-looking top marks, or remarked parts that share only physical traits with the authentic parts. For example, a simple transistor-transistor logic (TTL) chip might be placed inside the same package, with identical markings, as an expensive microcontroller.

I consider external mimicry trivial because fakes produced this way are easy to detect in a factory test. At worst, you're sold a mixture of mostly authentic parts with a few counterfeits blended in, so that testing just one part out of a tube or reel isn't good enough to catch the issue. But most products employ 100 percent testing at the system level, so typically the problem is discovered before anything leaves the factory.

REFURBISHED PARTS

Counterfeits don't technically have to be fake at all, though. Refurbished parts are authentic chips that are desoldered from e-waste and reprocessed to look new. They're very difficult to spot since the chip is in fact authentic, and a skilled refurbisher can produce stunningly new-looking chips that you'd only be able to verify were used through isotopic or elemental analysis.

This category also includes parts that are "new" in the sense that they've never been soldered onto a board, but have been stored improperly, perhaps in a humid environment. Such chips should be scrapped, but are sometimes stuck in a fresh foil pack with a more recent date code, and sold as new.

REBINNED PARTS

Counterfeiters sometimes remark authentic parts that have never been used (and so can be classified as new) as a better version of an otherwise identical part. A classic example is grinding and remarking CPUs with a higher speed grade, or more trivially, marking parts that contain lead as RoHS-compliant.

But rebinning can get more sophisticated. Vendors may reverse-engineer and reprogram the fuse codes inside the remarked chip so that the chip's electronic records actually match the faked markings on top. Vendors have also been known to hack flash drive firmware so that a host operating system will perceive a small memory as much larger. Such hacks even go so far as to “loop” memory so that writes beyond the device capacity appear to succeed, thus requiring a time-consuming full readback and comparison of the written data to detect the issue.

GHOST-SHIFT PARTS

Some fakes are created on the exact same fabrication facility as authentic parts; they’re run very late at night by rogue employees without the manufacturer’s authorization and never logged on the books. These unlogged production runs are called *ghost shifts*. It’s like an employee in a mint striking extra coins after-hours. Ghost-shift parts are often assigned a lot code identical to a legitimate run, but certain testing steps are skipped.

Ghost shifts often use marginal material left over from the genuine product that would normally be disposed of but was intercepted on the way to the grinder. As a result, the markings and characteristics of the material often look absolutely authentic. These fakes can be extremely hard to detect.

FACTORY SCRAP

Factory rejects and prototype runs can be recovered from the scrap heap for a small bribe, given authentic markings, and resold as new. To avoid detection, workers often replace the salvaged scrap with physically identical dummy packages, thus foiling attempts to audit the scrap trail. This practice of replacing salvageable scrap with dummy fakes helps drive the market for the trivial “external mimicry” fakes. The existence of an industry that supplies low-quality fakes to dodge audits that would otherwise prohibit high-quality fakes gives you an idea of how sophisticated and mature the counterfeiting industry has become.

SECOND-SOURCING GONE BAD

Second-sourcing is a standard industry practice where competitors create pin-compatible replacements for popular products to drive price competition and strengthen the supply chain against events like natural disasters. The practice goes bad when inferior parts are remarked with the logos of premium brands.

High-value but functionally simple discrete analog chips such as power regulators are particularly vulnerable to this problem. Premium US-branded power regulators sometimes fetch a price 10 times higher than drop-in Asian-branded substitutes. However, the Asian-branded parts are notorious for spotty quality, cut corners, and poor parametric performance. Clearly, there is ample opportunity for counterfeiters to make a lot of money by buying unmarked chips from the second-source fab and remarking them with authentic-looking top marks of premium US brands. In some cases, there are no inexpensive or fast tests to detect these fakes, short of decapsulating the chip and comparing mask patterns and cross-sections, as I did for the ST19CF68.

Fakes and US Military Designs

The variety of counterfeiting methods available, combined with the fact that many commodity parts have production cycles of only a few years, presents a big problem for institutions like the US military, where design lifetimes are often measured in decades. It's like asking someone to build a NeXTcube* motherboard today using only certifiably new parts, with no secondhand or refurbished parts allowed. I don't think it's possible.

The impossibility of this situation may sometimes make military contractors complicit in the consumption of counterfeit

* *Remember that one? The NeXTcube was a computer released in 1990 by Steve Jobs's company, NeXT.*

parts to bad effect. In the P-8 Poseidon case, people were quick to point fingers at China, but a poor refurbishing job is probably detectable with a simple visual inspection. Maybe part of the problem is that a subcontractor was lax in checking incoming stock—or perhaps looking the other way. If those parts were the last of their kind in the world, what else could be done?

My guess is that the stocks of any distributor in the secondhand electronics business are already flooded with undetected counterfeits. Remember, only the bad fakes are ever caught, and chip packaging was not designed with anticounterfeiting measures in mind. While all gray-market parts are suspect, that's not necessarily a bad thing.

Gray markets play an essential role in the electronics ecosystem; using them is a calculated, but sometimes unavoidable, risk. In fact, many traders in the gray market are very upfront about their goods being recycled. Many even post signs on their stalls advertising this fact. However, these signs are written in Chinese. In that case, whose fault is it—the seller for selling recycled goods, or the buyer for not being able to read the sign?

Anticounterfeit Measures

The counterfeit chip situation is a mess, but some simple measures could fix it.

PHYSICAL IDENTIFIERS

Embedding anticounterfeit measures in chips approved for military use is one option. For chips larger than 1 cm wide, a unique 2D barcode could be laser-engraved by equipment relatively common in chip packaging facilities. Despite a tiny footprint, the codes would be backed with a guarantee of 100 percent uniqueness. Such techniques are effective in biotech, where systems like Matrix 2D track disposable sample tubes in biology labs.

Another potential solution is to mix a UV dye into the component's epoxy that changes fluorescence properties upon exposure to *reflow* temperatures—a consistent set of well-defined temperatures at which solder melts. This makes it impossible to recondition the chip to a "new" state after it's been soldered down the first time. If the dye is distributed through the entire package body, it will be impossible to remove with surface grinding alone.

CHANGING HOW E-WASTE IS HANDLED

Managing e-waste more effectively would also alleviate the counterfeit problem. E-waste is harvested in bulk for used parts. Crudely desoldered MSM-series chips—the brains of many Android smartphones, made by Qualcomm and marketed under the brand name of Snapdragon—are purchasable by the pound, at around 10 cents per chip. Counterfeiters clean up the chips, *reball* (that is, add new solder balls, for ball-grid array packages) and sometimes remark them, put them into tapes and reels, and sell them as brand new, commanding a markup 10 times the original purchase price. A single batch of refurbished chips can net thousands of dollars, making the practice a compelling source of income for skilled workers who would otherwise earn $200 per month in a factory doing exactly the same thing. (Factories are typically authorized to recover chips off of defective boards or consumer returns that can't be repaired.)

If the US stopped shipping e-waste overseas for disposal, or at least ground up the parts before shipping them, then the supply for refurbished chip markets would decrease. Processing e-waste domestically would also create more jobs, a resource as valuable as gold.

On the other hand, I think component-level recycling is quite good for the environment and the human ecosystem in the long term. Most electronic parts will function perfectly for years beyond a consumer's trash bin, and emerging economies create technology-hungry markets that can't afford new parts purchased on the primary market.

KEEPING A RESERVE OF AUTHENTIC PARTS

A final option to ensure trustworthiness for critical military hardware could be to establish a strategic reserve of parts. A production run of military planes is limited to perhaps hundreds of units, a small volume compared to consumer electronics production runs. I imagine the lifetime demand of a part, including replacements, is limited to tens of thousands of units. Physically, then, a parts reserve isn't unmanageable: 10,000 chips will fit inside a large shoebox.

Financially, I estimate purchasing a reserve of raw replacement components for critical avionics systems would add only a fraction of a percent to the cost of an airplane. This could even lead to long-term savings, as manufacturers can achieve greater scale efficiency if they run one large batch all at once.

Obviously, anticounterfeit measures would be incredibly useful in civilian projects, too. I have sympathy for anyone who has to deal with counterfeit parts, as I myself have been burned on several occasions. Here's a tale of a particularly annoying issue I ran into during my work on the chumby One.

FAKE MICROSD CARDS

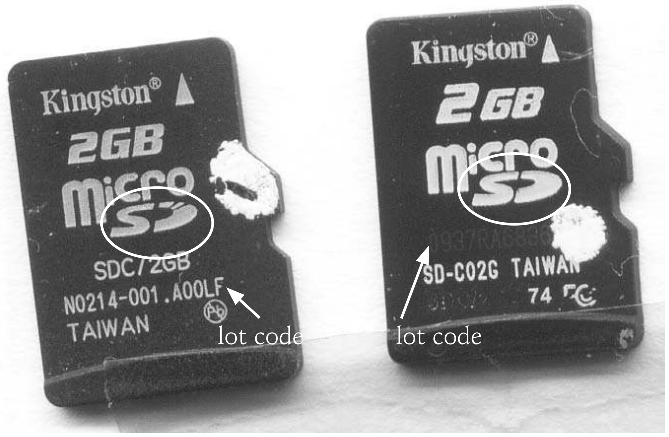
In December 2009, in the middle of the chumby One's production run, I set out on a forensic investigation to find the truth behind some irregular Kingston memory cards. The factory called to tell me that SMT yield dropped dramatically on one lot of chumby Ones, so I drove over to see what I could do to fix the problem. After poking and prodding at some chumby Ones, I realized that all failing units had Kingston microSD cards from a particular lot code. I had the factory pull the entire lot of microSD cards from the line and rework the units that had these cards loaded. After swapping the cards, yield returned to normal.

The story should have ended there. In this situation, I'd usually get a return merchandise authorization (RMA) from the manufacturer for the defective parts, exchange the lot for parts that work, and move on. But I had a couple of problems.

First, Kingston wouldn't take the cards back because we programmed them. Second, there were a lot of defective cards (about 1,000 all together, and chumby was already deeply backordered) and memory cards aren't cheap. This type of memory card cost around $4 or $5 at the time, leaving a few thousand dollars in scrap if we couldn't get them exchanged. Chumby couldn't afford to sneeze at a few kilobucks, so I kicked into forensics mode.

Visible Differences

Irregular external markings were the first suspicious feature I noticed about the defective Kingston cards.



An irregular microSD card (left) and a normal card (right). The arrows show suspicious differences.

The strangest physical difference was that the lot code on the irregular card was silkscreened with the same stencil as the main logo. Silkscreening a lot code isn't unusual, but typically, the manufacturer won't use the same stencil for the lot code and the logo. There should be some variance in the coloration, font, or alignment of the lot code from the rest of the text. The entire batch of irregular cards also had the same lot code (N0214-001.A00LF). Typically, the lot code changes at least every couple hundred cards. Contrast the irregular card to the normal card, which is laser-marked. The normal cards' lot codes varied with every tray of 96 units.

The second strange feature was subtler and perhaps not damning: an irregularity in the microSD logo. Brand-name vendors like Kingston are very picky about the accuracy of their logos: SanDisk cards have a broken *D*, but Kingston cards sold in the US almost universally use a solid *D*.

Investigating the Cards

Oddities in the external markings were just the start. When I read the electronic card ID data on the two cards (by checking */sys* entries in Linux), this is what I found in the irregular card:

```
cid:41343253443247422000000960400049
csd:002600325b5a83a9e6bbff8016800095
date:00/2000
fwrev:0x0
hwrev:0x2
manfid:0x000041
name:SD2GB
oemid:0x3432
scr:0225000000000000
serial:0x00000960
```

And this is what I found in the normal card:

```
cid:02544d5341303247049c62cae60099dd
csd:002e00325b5aa3a9ffffff800a80003b
date:09/2009
fwrev:0x4
hwrev:0x0
manfid:0x000002
```

```
name:SA02G
oemid:0x544d
scr:0225800001000000
serial:0x9c62cae6
```

First, notice the `date` code on the irregular card. Dates are counted as the offset from 00/2000 in the CID field, so a value of 00/2000 means the manufacturer didn't bother to assign a date. Furthermore, in the year 2000, 2GB microSD cards didn't even exist. Also, the serial number on the defective card is very low: in decimal, 0x960 is 2,400. Other cards in the irregular batch had similarly low serial numbers, in the hundreds or thousands.

For a popular product like a microSD card, the chance of getting the very first units out of a factory is pretty remote. For example, the serial number of the normal card is 0x9C62CAE6 in hexadecimal, or 2,623,720,166 in decimal, which is much more feasible. Very low serial numbers, like very low MAC ID addresses, are hallmarks of a ghost shift.

Finally, the manufacturer's ID on the irregular card is 0x41 (capital *A* in ASCII), which I didn't recognize.* The original equipment manufacturer identification (OEMID) number was 0x3432—an ASCII 42, which is one more than the hex value for the manufacturer ID. Manufacturer IDs are usually the ASCII character given by the hexadecimal value, not the hexadecimal values themselves. Confusing hex and ASCII is a possible sign that someone who didn't appreciate the meaning of the fields was running a ghost shift making these cards.

Were the MicroSD Cards Authentic?

Armed with this evidence, Chumby confronted the Kingston distributor in China and Kingston's US sales representative. We asked whether the cards were authentic, and if so, why the serialization codes were irregular. After some time, Kingston swore the cards were authentic, not fakes, but it did reverse

* *JEDEC Publication N. 106AA lists all SD card manufacturer ID codes, and 0x41 wasn't on there.*

its position on exchanging the cards. The company took back the programmed cards and gave us new ones, no further questions asked.

However, Kingston never said why the card ID numbers were irregular. I know Chumby was a small fry compared to the Nokias of the world, but companies should still answer basic questions about quality control, even for small fries. I was once accidentally shipped an old version of a Quintic part, and once I could prove the issue, I received world-class customer service from Quintic. The company gave me a thorough explanation, and immediately paid for a full exchange of the parts. That was exemplary service, and I commend and strongly recommend Quintic for it. Kingston, on the other hand, did not set an example to follow.

I'd normally have disqualified Kingston as a vendor, but I was persistent. It was disconcerting that a high-profile, established brand would stand behind such irregular components. Who could say SanDisk or Samsung wouldn't do the same? Price erosion at the time hit flash vendors hard, and as a small fry, I could have been taken advantage of by any of those companies as a sink for marginal material to improve their bottom line. Given the relatively high cost of microSD cards, I needed *incoming quality control (IQC)* guidelines for inspections to follow to accept or reject shipments from memory vendors based on set quality standards. To develop those guidelines, I continued digging for the truth behind those cards.

Further Forensic Investigation

First, I collected a lot of sample microSD cards. I wanted to collect both regular *and* irregular cards in the wild, so I went to the Hua Qian Bei district and wandered around the gray markets there. I bought 10 memory cards from small vendors, at prices from 30 to 50 RMB ($4.40–$7.30 USD).

Shopping for irregular cards was interesting. In talking to a couple dozen vendors, I learned that Kingston, as a brand, was weak in China for microSD cards. SanDisk did a lot more marketing, so SanDisk cards were much easier to find on the open market, and the quality of gray-market SanDisk cards was fairly consistent.

Small vendors were also entirely brazen about selling well-crafted fakes. They had bare cards sitting loose in trays in the display case. (page 11 in Chapter 1 has photos showing what an SD card vendor's stall looks like.) Once I agreed on a price and committed to buying a card, the vendor tossed a loose card into a "real" Kingston retail package, miraculously pulled out a certificate—complete with hologram, serial numbers, and a kingston.com URL to visit to validate the purchase—and slapped the certificate on the back of the retail package right in front of my eyes.

A freshly purchased Kingston microSD card. It was just like new!

One vendor particularly interested me. There was literally a mom, a pop, and one young child sitting in a small stall of the mobile phone market. They were busily slapping dozens of non-Kingston cards into Kingston retail packaging. They had no desire to sell to me, but I was persistent. This card interested me in particular because it also had the broken *D* logo, but no Kingston marking. The preceding photo is the card and the package it came in; the card is Sample 4 in the next section, where you can see a detailed analysis of seven different microSD cards from my shopping trip.

Gathering Data

After collecting my samples, I read out their card ID information by checking their */sys* entries under Linux, and then decapsulated (that is, dissolved) their packages with nitric acid. As you can see in the photos in the following table, my decapsulation technique was pretty crude. Most of the damage to the cards came from removing dissolved encapsulant with acetone and a Q-tip. I had to get a little rough, which didn't do the bond wires any favors. But it was good enough for my purposes.

Here's all the basic information I pulled from those cards:

Sample 1 The irregular card that started this whole investigation. It was purchased through a sanctioned Kingston distributor in China, and to the best of my knowledge, none were shipped to Chumby's end customers. MID = 0x000041, OEMID = 0x3432, serial = 0x960, name = SD2GB.

Sample 2 A normal card that I purchased from the same sanctioned Kingston distributor in China where I bought Sample 1. It was typical of microSD cards actually shipped in the first lot of chumby Ones. MID = 0x000002, OEMID = 0x544D, serial = 0x9C62CAE6, name = SA02G.

Sample 3 A Kingston card purchased through a major US retail chain. MID = 0x000002, OEMID = 0x544D, serial = xA6EDFA97, name = SD02G. Note how the MID and OEMID are identical to those Sample 2, but not Sample 1.

Sample 4 The non-Kingston card I saw slapped into Kingston-marked packaging, bought on the open market in Shenzhen. MID = 0x000012, OEMID = 0x3456, serial = 0x253, name = MS. Note the low serial number.

Sample 5 A device from a more established retailer in the Shenzhen market. I bought it because it had the XXX.A00LF marking, like my original irregular card. MID = 0x000027, OEMID = 0x5048, serial = 0x7CA01E9C, name = SD2GB.

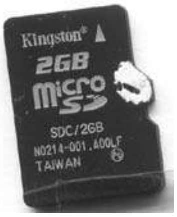
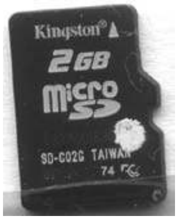
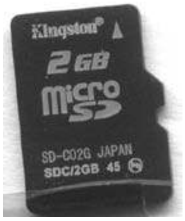
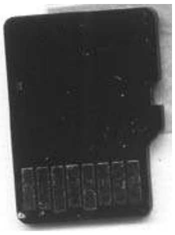
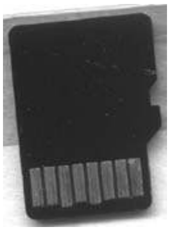
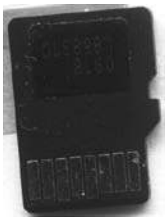
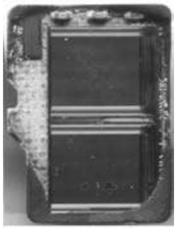
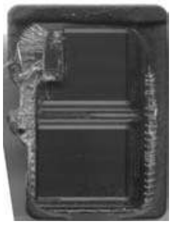
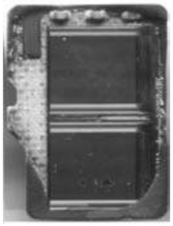
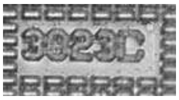
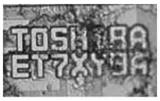
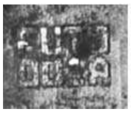
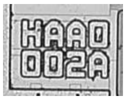
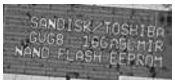
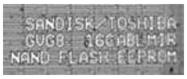
Sample 6 A SanDisk card bought on the open market from a sketchy shop run by a sassy chain-smoking girl who wouldn't stop texting. I actually acquired three total SanDisk cards from different sketchy sources, but all of them checked out with the same CID info, so I opened only one. MID = 0x000003, OEMID = 0x5344, serial = 0x114E933D, name = SU02G.

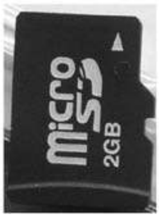
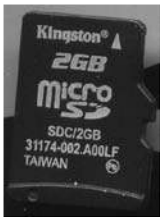
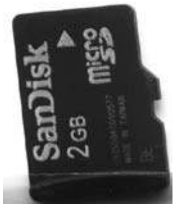
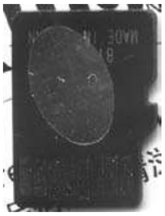
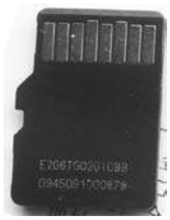
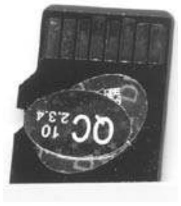
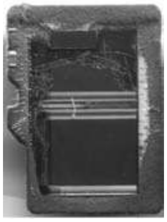
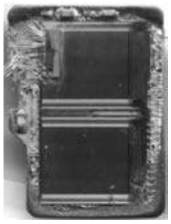
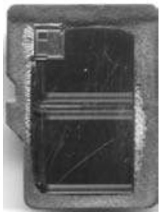
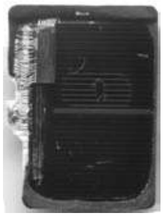
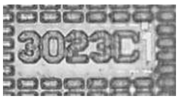
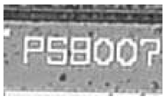
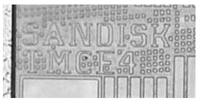
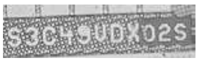
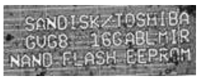
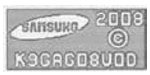
Sample 7 A Samsung card that I bought from a Samsung wholesale distributor. I didn't scan this one before decapsulating it, and the card actually had no markings on the outside (it was blank, with just a laser mark on the back), so I didn't photograph it. From appearances alone, it was the sketchiest of the bunch, but it was one of the best built. You can't judge a book by its cover! MID = 0x00001B, OEMID = 0x534D, serial = 0xB1FE8A54, name = 00000.

That's a lot of data, and I had my work cut out for me drawing some kind of useful conclusion from it all.

NOTE *Interestingly, one SanDisk card from three in Sample 6 turned out to be used and only quick-formatted. With help from some recovery software, I found DLLs, WAVs, maps, and VeriSign certificates belonging to Navione's Careland GPS. Someday, I'll acquire lots of refurb microSD cards and collect interesting data from them.*

A Breakdown of All the Cards Collected for the Investigation

	Sample 1: Original Kingston card from authorized Kingston distro	**Sample 2: Normal Kingston card from authorized Kingston distro**	**Sample 3: US retail Kingston card**
Front marking			
Back marking			
Decapsulated			
Controller die marking		 	
Flash die marking		(SanDisk/ Toshiba flash)	

Sample 4: Fake card bought from SZ market	**Sample 5: Questionably authentic Kingston card bought from SZ market**	**Sample 6: SanDisk card bought from SZ market**	**Sample 7: Samsung card bought from authorized Samsung distro**
			Samsung card image missing
			Samsung card image missing
			
			
(SanDisk/ Toshiba flash)		(SanDisk/ Toshiba flash)	

Summarizing My Findings

Here are the most interesting high-level conclusions I drew from my survey:

- The "normal" Kingston cards (Samples 2 and 3) were fabricated by Toshiba, as indicated by the flash die markings and their OEMIDs. 0x544D is *TM* in ASCII, presumably for *Toshiba Memory*. These cards employ Toshiba controllers and Toshiba memory chips, and seem to be of good quality. Thankfully, they were only ones sent to Chumby customers.
- The irregular card (Sample 1) used the same controller chip as the outright fake (Sample 4) I bought in the market. Both the irregular Kingston and the fake Kingston had low serial numbers and wacky ID information. Both of these cards exhibited abnormal operation under certain circumstances. I still hesitate to call Kingston's irregular card a fake, as that's a very strong accusation, but its construction was similar to another card of clearly questionable quality, which leads me to question Kingston's choice of authorized manufacturing partners.
- The irregular card is the only card in the group that does not use a stacked CSP construction. Instead, it uses *side-by-side bonding*—that is, the microcontroller and the memory chip are simply placed next to each other. Stacked CSPs place the microcontroller on top of the memory chip. This is significantly more complex than side-by-side placement, because the chips must first have their inert back-side material ground off to make the overall height of the stack fit inside such a slim package. Despite the difficulty, stacking chips is popular because it allows vendors to cram more silicon into the same footprint.
- The only two memory chip foundries in this sample set were Toshiba/SanDisk and Samsung. (SanDisk and Toshiba co-own the factory that makes their memory chips.)

- Samsung's NAND die, which is the most expensive part of a microSD card, is about 17 percent larger than dies from Toshiba/SanDisk. This means that Samsung microSD cards should naturally carry a slightly higher price than Toshiba/SanDisk cards. However, Samsung can offset that against the ability to place the same bare die that normally gets crammed inside a microSD package into thin small outline package (TSOP) devices suitable for board-level machine assembly instead. If demand for microSD cards slumps, Samsung can slap excess bare dies inside TSOP packages and sell those to third parties that do conventional machine assembly of chips. Plus, Samsung also doesn't have a middleman like Kingston to eat away at margins.

I knew (like many others in manufacturing) that Kingston isn't a semiconductor manufacturer, in that it owns no fabrication facilities, but this research implied that Kingston does no original design of its own. I hoped to at least find a Kingston-branded controller chip inside the Kingston cards, even if the chip was fabricated by a foundry. I also expected to see Kingston sourcing memory chips from a broader variety of companies. Being able to balance the supply chain and be less dependent on a single, large competitor for chips would be a significant value-add to customers, giving Kingston leverage to negotiate a better price that few others can achieve. But every Kingston card I bought had a SanDisk/Toshiba memory chip inside. The only "value-add" that I saw was in the selection of the controller chip.

Oddly enough, of all the vendors, Kingston quoted Chumby with the best lead times and pricing, despite SanDisk and Samsung making all their own silicon and thereby having lower inherent costs. This told me that Kingston must have a very low margin on its microSD cards, which could explain why irregular cards found their way into its supply chain. Kingston is also probably more willing to talk to smaller accounts like Chumby because, as a channel brand, Kingston can't compete against OEMs like SanDisk or Samsung for the biggest contracts from the likes of Nokia and Apple.

So, the irregular microSD card I pulled from the chumby One production line may not have been counterfeit, but it was still a child of the remarking ecosystem in China. Kingston is more of a channel trader and less of a technology provider, and is probably seen by SanDisk and Toshiba as a demand buffer for their production output. I also wouldn't be surprised if SanDisk/Toshiba sold Kingston less-than-perfect parts, keeping the best of the lot for themselves. Thus I'd expect Kingston cards to have slightly more defective sectors, but thanks to the magic of error correction and spare sectors this fact is transparent to end users.

As a result, Kingston plays an important role in stabilizing microSD card prices and improving fab margins. But the potential conflict of interest seems staggering, and I'm still very curious about how this ecosystem came to be. Buying a significant amount of a competitor's technology from a competitor's fab yet still selling at a competitive price is counterintuitive to me, and perhaps my greatest folly in investigating that irregular microSD card was expecting something different.

FAKE FPGAS

Anyone who has done manufacturing in China for a while will have more than one story about irregularities in the supply chain. Here's another one of my favorite stories, which highlights some of the core incentives that drive agents to cheat.

The White Screen Issue

It was March 2013, and I was wrapping up the first volume production run of a bespoke robotics controller board codenamed Kovan.* At the conclusion of any production run, I

* *Kovan is open hardware; you can read more about it and download the source on the Kosagi wiki at* http://www.kosagi.com/w/index.php?title=Kovan_Main_Page.

always review the list of issues encountered in production, to identify areas of improvement. Manufacturing is a Sisyphean struggle toward perfection: every run has some units you just have to scrap, and the difference between profit and loss is how well you can manage the scrap rate.

On this run, one particular problem, dubbed the "white screen issue" after its most obvious symptom, was the dominant problem. About 4 percent of the total run exhibited this problem, accounting for almost 80 percent of unit failures. I had the factory send me a few samples of the failed units to analyze in more detail.

As I've often discovered when analyzing failed units, the most obvious symptom of the problem was only tangentially related to the root cause. The LCD screen appeared white on these units because the FPGA failed to configure. An FPGA, short for *Field Programmable Gate Array*, is essentially a blob of logic and memory devices embedded in a dense network of wires that can be configured at runtime to behave a certain way. The behavior of the FPGA is typically described in a high-level language that resembles a programming language like C (for instance, Verilog) or Ada (like VHDL), which is then compiled into a configuration bitstream.

FPGAs are very handy for implementing time-sensitive hardware interfaces that software would have trouble emulating. In this particular application, the FPGA controlled everything from the motors to the sensors and even the LCD. When the FPGA failed to configure, the LCD didn't receive sync and data signals, leading it to show a blank, white screen instead of the expected factory test patterns.

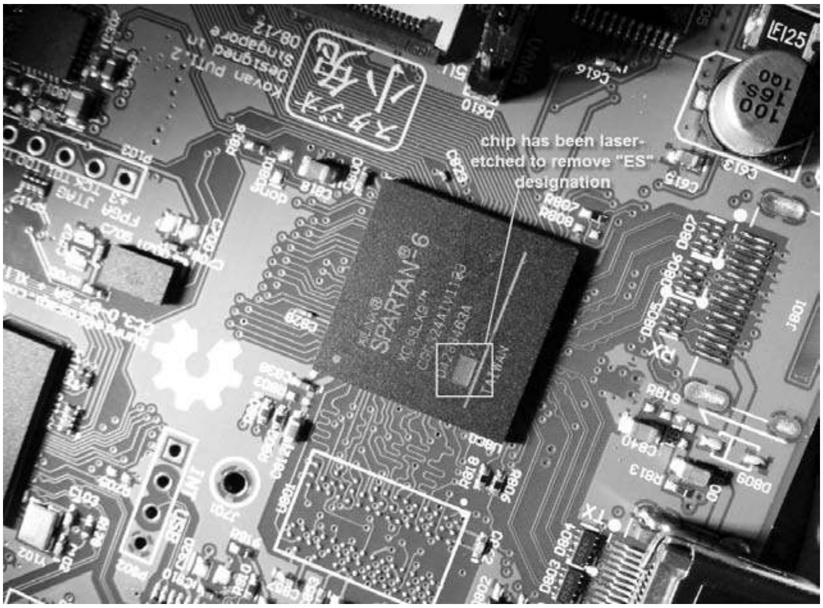
FPGA failure was a big deal. For starters, the FPGA was the most expensive part on the board by a long shot, at around $11 per chip. I was also worried this problem could point to a deeper design issue. Perhaps the FPGA's power regulators were unstable, or maybe there was an issue with the boot sequence that aggravated a corner case in configuration timing that would creep into the "good" production units as they aged. The situation definitely warranted a deeper investigation.

Incorrect ID Codes

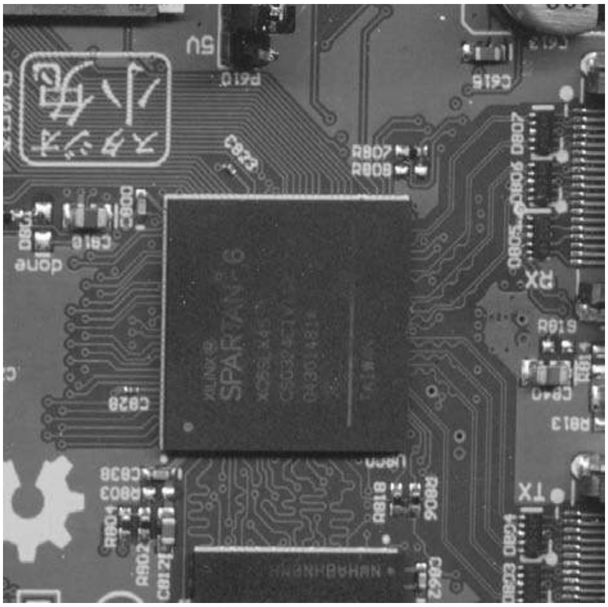
I hooked up the debug console, dug into the problem, and discovered that the failure was linked to the FPGA not responding with the correct ID code. The ID code is checked via queries over a test access bus known as *JTAG*. Most users don't check an FPGA ID before programming, but we designed an ID code check into Kovan because we allowed customers to specify what capacity FPGA they wanted to use for a given production lot. Some applications are more demanding than others, while others are more cost-sensitive. As a result, a customer could have a mixed inventory of FPGAs, and we wanted to be able to detect and protect the hardware from an accidental mismatch between the bitstream and the FPGA.

But this was a single production lot, and in theory all the FPGAs should have been the same. How, then, could the FPGA report a mismatched ID code at all? I scratched my head for a while and suspected a bug in our JTAG implementation, until I looked up the reported ID code. It was a known code—but for silicon marked as "Engineering Samples" from Xilinx, the vendor that makes these FPGAs. Engineering samples are preproduction units sold by Xilinx that have some minor known bugs but are sufficiently functional for most applications, to the point where most customers wouldn't see a difference, *except* for the ID code.

I looked closer at the PCB, and for the first time, I noticed that a small, white rectangle was laser-etched into the FPGA's surface. The rectangle was right below the part number, where the "ES" designator for an engineering sample would normally be marked. Someone had blasted the letters off and sold us engineering samples as full production units!



An engineering sample FPGA on a Kovan board



For contrast, an FPGA of the same type that hasn't been tampered with

The problem was very clearly a supply chain issue, not a design issue. Someone in the chain was taking ES silicon, blasting off the letters, and blending them in with legitimate units at a rate of around 3 to 5 percent. Typically, Xilinx would require that all ES silicon in distributor's inventories be scrapped once production units become available, but the ES units were almost fully functional, to the point where most applications would be unaffected. A production bitstream would load seamlessly into an ES part, and nobody would know the difference. The only way to tell them apart would be by doing an ID code check, which as I noted previously is atypical.

Thus, slipping ES silicon into production lots would likely go unnoticed. Mixing ES parts in at a rate of 3 to 5 percent was also very clever: a low mix rate makes substitutions very hard to catch without 100 percent prescreening of the parts. Even in production, if the ES silicon were marginal, it would be maddeningly difficult to nail down the root cause of an issue due to its rarity.

In fact, there's a correlation between manufacturing difficulty and the use of FPGAs. Usually if your design calls for an FPGA, you're pushing boundaries on multiple fronts, and so a scrap rate of a few percent is to be expected. The margin on FPGA-powered hardware is also often fat enough that a 4 percent failure rate might simply be accepted by the end customer. Thus, whoever did this knew exactly what they were doing; it was virtually risk-free money.

Finally, it's important to note that most vendors in a supply chain survive on single-digit margins, so finding an extra 3 to 5 percent of "free money" on the most expensive part on a board virtually doubles profitability. That provides a very strong incentive to cheat, especially if you think you won't be caught.

The Solution

The resolution to this problem was quite interesting. I met with the managers and CEO of AQS, the CM charged with producing Kovan, briefed them about the problem, and showed them the evidence I had accumulated. When my presentation ended, the CEO didn't point a finger at upstream vendors or partners. Instead, he immediately looked his staff in the eyes and asked, "Did any of you do this?" He understood better than anyone else in the room that any individual buyer or manager would effectively double their take-home pay that month if they could pull off this cheat without getting caught.

In other words, the truly remarkable part of this situation is how rarely the problem I experienced happens, given what's at stake and how hard these problems are to catch. And while I do have a few good bar stories to tell about fakes in the supply chain, remember that I've also shipped hundreds of thousands of units of good product. The majority of people in China are hard-working, honest people who pass on easy opportunities to cheat me and turn a profit. It's important not to generalize the whole based on the bad actions of a few.

At the end of the day, the vendor who sold us the chips didn't admit fault, but they did replace all remarked units at their own cost. (We still had to pay for the labor cost to replace the chips and recertify the boards.) This is about the closest you can get to an amicable resolution in China when you're not a giant like Apple or Foxconn. I did send a note to Xilinx HQ about potential misbehavior by one of their authorized vendors, but in the end, I'm a small customer and the substitution of parts could have happened literally anywhere on the supply chain. Even the courier delivering the packages could have done the swap.

It wouldn't be worth the cost to Xilinx in terms of manpower, relationships, and focus to investigate the problem and rat out the one bad actor in literally hundreds of possible suspects. But I'd like to imagine that at least a memo was sent around, and whoever was swapping in the ES parts got scared enough that they stopped.

CLOSING THOUGHTS

At the end of the day, a permissive IP ecosystem has benefits and drawbacks. As an engineer and a designer, I prefer to be in an ecosystem where ideas are accessible, even if it means I have to be on guard for occasional problems with fake goods. Put another way, a fundamental prerequisite for virality is the ability to make copies. The explosion of interest in hardware startups is in part thanks to the highly competitive manufacturing ecosystem that could flourish only in a product-over-patent culture.

Westerners who come to China without understanding the principles of *gongkai* and *guanxi* often feel like they're being cheated. But once you understand the rules and learn how to use them to drive your interests, you won't feel like the game is rigged against you anymore.

In the US IP system, honor has little economic value, and law trumps honor. For example, patent trolling is a perfectly legal, and very profitable, way to make a living. In the Chinese system, however, reputation can trump law. This opens the door for corruption but also crowd-sources the enforcement of social and moral values, driving a market value for honor, especially in local, tightly knit communities.

Of course, the approach of making money by locking up ideas and selling the rights to them is patently incompatible with a permissive IP ecosystem. Thankfully, the notion that ideas are community property dovetails nicely with my open source philosophies. In the next part of the book, I'll talk more about my experiences creating open hardware and building businesses rooted in these principles.