GROUWSTRA

IC & Software Patent

How is a Chip or Firmware Patent Different than Other Patents?

What About a Software Patent?

This Patent Stuff and My Semiconductor Business – Part 6

Welcome to this post about patents and chips. Not a lot has been written about this combination, but there is a lot to know, especially for the innovators and entrepreneurs themselves. In this three-weekly series, I talk about various aspects, from my dual points of view of a patent agent and a semiconductor entrepreneur. If you like the article and read it on LinkedIn, give it a thumbs up, and/or click on Follow. If you like to work with us for your next patent, "contact us" info is on [*www.icswpatent.com*](http://www.icswpatent.com)*. You can also subscribe/unsubscribe for short email alerts when the next post is available.*

This time we're getting into the heart of the matter, so to say. Are chip patents, firmware patents, and software patents any different than any other patents? The answers are *no*, *somewhat*, and *yes*—at least, for US law. From a business perspective, yes, they are different and it really helps if you understand the specifics.

Let's brush over software first. In the US, it used to be that you could patent just about anything crazy. If you want to know how crazy, just google "US patent 6368227". That patent is scheduled to expire this November 17, after which date you can relax when your kids to take a swing (in the US). But things have changed a lot in the past five years or so, and now software patent applications are quite critically examined before they're possibly allowed. There are new rules and there is a lot of new case law telling what can and cannot be allowed. It is still possible to patent software, but it needs to (actually!) be innovative, and a whole lot more. It helps a lot if the software is tied to specific hardware—and that is where firmware comes in. Firmware is generally easier to patent than, say, software that runs on an all-purpose computer. Mind you that for any software, you can't patent it if it is an algorithm that you could do inside your head, or with the aid of a pencil and paper, even if it would take 130 years to do so, and even if you specifically state that you perform that algorithm on a computer.

Luckily, for us semiconductor people, we have at least two advantages: first, our circuits can be touched and thus fall into a patently patentable category. Second, chip innovators have been rumored to be (censored). Some clearly brilliant. As a result, regular patent practitioners may burn out after 10 years of software patent applications, but those of us working on semiconductor patent applications keep doing it even after retirement—just because it is more fun.

I digress—let's get back to the issue. How is a chip patent application different than other patents?

First, the invention is often much broader than the inventor tells you. Yes, this means that the inventor might be even more of a genius than he or she already believes! The inventor is usually focused on solving a specific problem for the design or product that (s)he's working on; finds a solution; and then digs deep in an implementation. He or she doesn't consider the fact that the basic idea is applicable much wider, or the fact that it could be implemented rather differently.

Another issue is that the law says that the patent must describe the invention in an enabling way. You receive a 20- or 21-year privilege, and after that, your invention is shared by the public. In other words, twenty (one) years from now, when the patent expires, others in the community are allowed to start making, selling, and using your invention. Selling and using may not be so much of an issue. But to make it, those others in the community should fully understand it first. The law gives them the right, and if the patent doesn't clearly explain what the invention is and how it works, it is not going to be granted, or worse, it may be granted, and subsequently successfully challenged by a competitor—just because the story was garbled or took a couple of shortcuts. Unfortunately, some of the most brilliant inventors are not the most brilliant explainers, and it can be really difficult (like pulling teeth) to get out of them how (and why) the thing really works.

Personally, I do the pulling teeth, gently but confidently, and find that the chip design inventors are usually happy to respond to my prodding questions. I will not write anything in a patent application unless I feel that I understand it. I will almost never just copy an inventor's description of how it works—because either it will give me a hard time later in the prosecution process or it will yield a half-baked patent whose value may be unnecessarily compromised. Or both. Oh, and it will likely also cost you more because it gives extra rounds of office actions to respond to, trying to fix something that cannot be fixed properly anymore. My approach is uncommon, as far as I can see, but not unique. It means that sometimes I have to spend a lot extra (of my) time to get to the bottom—I don't usually charge by the hour for patent work—or I may have to decline an invention that is beyond my field. But I have come across other patent practitioners who happily used the inventor's description, and praised the inventor for the "terrific job they have done writing it and saving you money".

Another issue with electronics is that there can be so many different circuits that all do the same thing. Once you've figured out how to solve a particular problem, there can be thousands of different circuit variations or even many different architectures that all implement the new solution. Say, the invention requires that you add two signals. Signals can be represented by a digital number (there are different digital representations for numbers), or by a voltage, or by a current, or by a pulse width, or yet something else. Each of these could lead to at least one different implementation, and a very different circuit or architecture. Architectures may need to all be described, or at least the most important ones. Different applications can also yield different circuits, for instance depending on whether external signals are single-ended or differential. Newer chip processes may require different circuits to implement the same thing than an older process would. In an RF or mixed-signal circuit, a metal stack that includes a thick metal top layer can yield a different implementation that without thick metal. Then, there is the possibility that the invention could be performed by dedicated and optimized logic, or it could sit in an FPGA, or it could run as firmware on one or more quasi-dedicated DSPs, or it could even run on an ARM or other processor.

To write a strong patent application, somebody needs to analyze the options and the different solutions, and somebody needs to decide what to include and how. And then, still, it is often necessary to describe the invention both in the form of whatever hardware or circuits it is and could be implemented in, as well as in a method that explains what the hardware is really doing.

You can't expect your inventor to do all that.

Personally, I think writing and pursuing a chip patent is inspiring and fun—but it is very different than for other types of inventions.

Now how about software? There is of course software in the semiconductor industry. I know that some of my readers make EDA tools. Others make verification IP and tools. The software runs on general-purpose computers, and that causes extra checks and balances that make patenting much harder. It is rather common that because of this, the path to allowance of the claims is longer than for hardware. It can be twice or three times as long. It is normal that in a first Office Action all claims are rejected. It is not uncommon that in a third Office Action still all claims are rejected. However, as always, a true innovation will go through the system much quicker than a marginal invention. By now, the USPTO's procedures for examining a software application have crystalized, and are well documented. The procedure is not simple though, and the USPTO guidelines include many examples, including a number specifically tied to recent case law. Essentially, they document in what circumstances software is acceptable even when not tied to specific hardware. But the material is broader, and the guidelines can even be applied to hardware inventions. Software patenting requires experience and expertise. And some thick skin is also helpful. Not all practitioners will work on software patent applications.

So how about firmware? Firmware is tied to specific hardware. That makes it much easier to work with, and frankly, much more fun. You chip patent practitioner will be glad to help you just as if it's a regular hardware patent application.

Upcoming:

7 Woohoo! I Invented a Huge Improvement over My Competitor's Invention!
8 I'll Be A Billionaire Soon Enough. But Now I'll Just Buy This Book on Patent Writing on thriftbooks.com.
9 My CTO Can't Explain His Invention to Me. But He Is the Smartest Guy in the World.
10 Should I Do a Provisional, Non-Provisional, Or a PCT?
11 My Invention is Vital for My Business Plan. But I Don't Have Much Money Yet. How Can I Save?

Published so far (find them on www.icswpatent.com or #ThisPatentStuff):

1 So You Got This Great Idea That Will Wipe Out Competition. Now What?
2 Developing an IP Protection Strategy for Your Semiconductor Company – PART I
3 Developing an IP Protection Strategy for Your Semiconductor Company – PART II
4 In What Countries Should I Patent, Anyway?
5 Choosing the Right Patent Person for Your Inventions
6 How is a Chip or Firmware Patent Different than Other Patents? What About a Software Patent?

Disclaimer

Please do not construe anything in this article as legal advice: it isn't. The article contains my private opinions, with where possible the point of view of a semiconductor industry entrepreneur and/or a patent agent fighting for the inventor and the entrepreneur. If you need a strong patent on your circuit and/or system, I might be your guy.

www.icswpatent.com