What is a software patent?

By Van Thompson, Director, Consulting Services, IP Services, TechInsights

Software patenting practices need to change in response to recent US rulings. Van Thompson, TechInsights, argues that multiple, more narrowly claimed patents and focusing on software controlling hardware are the answer.

There has been a lot of talk the past several years about software patent rulings related to business methods and patentable material (ex. is the concept inventive if it is computer assisted?). Going back to 1981, the Supreme Court ruled in Diamond v. Diehr that the software controlling the physical rubber molding process, primarily based on a known mathematical equation, didn't eliminate the molding process' patentability. This ruling led to confusion as one could argue that the rubber molding process itself, not the software, was the improvement.

Business method patents gained legitimacy in 1998 with the State Street Bank & Trust v. Signature Financial Group ruling. This case revolved around a patent that included software that controlled mutual fund investment decisions. State Street wanted the method to be deemed unpatentable since they argued it was simply a mathematical equation, but the court ruled the method was patentable "because it corresponded to a useful, concrete or tangible thing" (i.e. the transformation of data, representing discrete dollar amounts).

However, business method patents were again thrown in disarray with the Bilski v. Kappos decision in 2010. The Bilski decision essentially rejected the State Street Bank & Trust v. Signature Financial Group ruling and created a new standard by which business method patents are to be judged. A key point in the ruling included "The machine-or-transformation test is not the sole test for deciding whether an invention is a patent-eligible "process."" Judge Stevens later added "The primary concern is that patents on business methods may prohibit a wide swath of legitimate competition and innovation." The court later summarized that a process is patentable if:

i) It is tied to a particular machine or apparatus, or
ii) It transforms a particular article into a different state or thing.

The recent Alice Corporation vs. CLS Bank International ruling further limited business methods patents by specifying non-patentable methods, generally referred to as abstract ideas, can't become patentable through software implementation. In his ruling, Justice Clarence Thomas explicitly stated "We hold that the claims at issue are drawn to the abstract idea of intermediated settlement, and that merely requiring generic computer implementation fails to transform that abstract idea into a patent-eligible invention." Language of that nature narrows business method patents significantly as legal teams will need to prove something isn't an abstract idea while also trying to demonstrate that the software provides an improvement over existing business methods or processes.

It has become blatantly obvious that software patenting practices going forward need to change. Historically in all mediums, patent prosecutors typically try to draft claims as broadly as possible to provide maximum coverage for their patents. Alice sends a clear message to not get greedy as those patents will not hold up in court even if they somehow get issued by the USPTO. It is a given that narrowing the claims within a specific patent will be a necessity, but that doesn't mean you won't be able to properly protect your innovations. A strategy involving the development of multiple conservative patents with narrower claims covering smaller components of the

technology can still be used to provide adequate coverage of a company's future products. Creating a portfolio in this manner will undoubtedly increase the upfront costs of a patenting program. However, the long view is a higher quality portfolio that has a much better chance of surviving the courts. The patents will not only be individually stronger, but will also have broader coverage within a larger group, meaning the opposition has to deal with more than one all-encompassing patent.

It is also evident that additional details in the specification are essential for greater success in the software patenting realm. Details on how the software is not only coded but also an improvement on the prior art will be crucial in showing the implementation is not just another abstract idea or a software representation of a formula/method known to many for several years or even decades. Patent prosecutors need to be able to show that the software isn't simply representing a process anyone can complete in their head. Details outlining the number of calculations completed to solve the problem or even lines of innovative code could be very helpful in revealing how the software is a key differentiator as compared to what was done previously. Patent prosecutors may even want to include lines of code within the claims to properly narrow what is covered. As with anything in high tech, if the coding scheme is fundamental and/or particularly cost effective for future products, others will follow a similar approach to remain competitive.

Exceedingly detailed specifications will also be a major benefit to a corporation's continuation practice. More information in the specification will lead to multiple claim improvement opportunities as long as open patent applications exist. This exercise buys companies much more time in viewing what techniques are adopted in the industry and, with the proper specification backing, patent prosecutors could develop IP that reads on products already sold (i.e. generating revenue) in the market.

Even taking the aforementioned patenting practices and the ground-breaking technological advancements in the software space into account, many still believe that the complications related to the enforcement of business method patents apply to all software patents. Like other technology groupings in the high tech sector (semiconductor, wireless, automotive, etc.), the key question for software patents should be, "Is this patent enforceable and, if so, who is the addressable market?" In other words, is the patent detectable and can one infer, or better, conclusively show that a competitor is using my patented technology? The answer to all of these questions is a resounding yes and the sweet spot from both a market capitalisation and evidence of use creation perspective is software used to control hardware.

Software controlling hardware is an intriguing area in the technology sector as it is highly involved in the consolidation of multiple features into one device. Much like the semiconductor world with formerly distinct analog, RF, memory and processor IC's being condensed into one SoC (System-on-Chip), everything consumer electronics related is being aggressively designed into one mobile device. Cell phones and tablets are now the primary camera, TV, GPS, alarm clock and soon to be medical device of choice for many consumers. Most advancements in the consolidation process are largely driven by software, which highlights why this is such an exciting time for software patent development and enforcement.

There are countless avenues to take when searching your software related portfolio for patents of interest. Software is used to control temperature, battery life, graphics, image capture and compression, etc., and every cell phone, tablet or PC has an operating system that accesses certain portions of the hardware for key functionality. Areas of focus should involve features that are contained on a sole product as arguments pertaining to shared infringement can arise if multiple parties (ex. a patent including a wireless product and base station or external server elements) are implicated. Patents specific to operating systems, Application Programming Interfaces (APIs), product drivers (ex. Graphics), image processing or accessing library files have potential, to name a few.

From a detectability perspective, software analysis with respect to hardware control is generally quite cost effective and low risk. Most projects consist of a feasibility study to determine the overall scope and extent of what is required to support the given patent(s). This allows one to move on from an uncertain, time consuming or cost prohibitive project to the low hanging fruit for your licensing campaign. Finding where a function is implemented can be much harder than testing the functionality on many products. When located, the proper

expert can write a program to execute a desired function on the target product and later complete the functional testing and analysis.

Furthermore, once key features in the control of specific hardware are identified, it opens the door for additional patents in your arsenal related to functional testing, board level tracing or even IC related reverse engineering. This broadening of exposure on a given target is gold for any lawyer or licensor pursuing a case. In addition, the details of how the software controls the hardware are clearly visible and tangible when the applicable sector is identified and tested, which is key in answering the "tied to a particular machine or apparatus" and "transforms a particular article into a different state or thing" questions outlined by the Supreme Court in the Bilski case.

These factors lead to another reason why software controlling hardware patents are so appealing. In most if not all cases, the software for a given product is created by the product vendor, meaning you are not bringing other parties such as IC manufacturers into play when completing software patent analysis on a product. This can eliminate concerns about patent exhaustion and can significantly reduce the apportionment risk as well. Two scenarios are quite favourable in this instance:

i) The patent is used solely by the product software.
ii) The patent is used by the product software and product hardware manufactured by the same vendor.

In these scenarios, it is easy to extrapolate an argument for higher damages since the patent has an increased utilisation or apportionment on the targeted product.

Business method patents have dominated the discussion with software patents for over thirty years. There have been multiple rulings used to determine what is patentable, what is abstract and what state was transformed in order to create a truly innovative process. Even with these rulings, the general consensus is that the enforceability of business method patents is still uncertain. However, software controlling hardware patents are not only innovative, but they are also tangible, risk averse and cost effective to pursue. Not only that, the addressable market is growing (worldwide smartphone shipments in 2020 are expected to reach about 2.5B alone), consolidation of products creates additional features to analyse and the impact of patent exhaustion and apportionment is reduced given that, in many cases, one party is the sole target of the patent.

Patenting practices will and have to change to better handle recent rulings. However, it is still possible to protect your technology using multiple, more narrowly claimed patents and amply detailed patent specifications that enable an effective patent continuation program. All of this will lead to higher quality, more assertable portfolios in the long run.

So, what is a software patent? It isn't simply a business method or abstract idea. Legal experts and licensing professionals primarily focus on enforceable patents with a large potential market and limited risk. In the software field, those patents control hardware.

Van Thompson has worked with TechInsights, a technology consulting firm that helps the world's leading patent owners protect and grow their businesses, since 1998. As Director, Consulting Services, Mr. Thompson provides leadership to TechInsights' global team of Principals and Consultants whilst working directly with several clients including external licensing agencies and in-house corporate counsel of leading technology firms. Areas of specialisation include patent portfolio management; reverse engineering for competitive intelligence or licensing purposes; and IP due diligence & risk management. He is a Professional Engineer (Ontario) and holds a BSc in Electrical Engineering from Queen's University. Van can be reached at VThompson@techinsights.com