No. 13-298

IN THE

Supreme Court of the United States

ALICE CORP. PTY. LTD,

Petitioner,

v.

CLS BANK INTERNATIONAL ET AL.,

Respondents.

On Writ Of Certiorari to the
United States Court Of Appeals
For The Federal Circuit

BRIEF OF *AMICUS CURIAE* IEEE-USA IN SUPPORT OF NEITHER PARTY

CHRIS J. KATOPIS
Counsel of Record
1308 CLIFTON STREET, N.W.
WASHINGTON, DC 20009
Tel. (202) 200-9490
chriskatopis@gmail.com

28 January 2014

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Pages

CASES

STATUTES

OTHER AUTHORITIES

STATEMENT OF INTEREST OF *AMICUS CURIAE* [1]

IEEE-USA is an organizational unit of The Institute of Electrical and Electronics Engineers, Inc. (IEEE), the world's largest organization for technical professionals, and a leading educational and scientific association for the advancement of technology. IEEE-USA supports the nation's prosperity and competiveness by fostering technological innovation for the benefit of all, including more than 200,000 U.S. engineers, scientists, and allied professionals who are members of the IEEE.

As part of its mission, IEEE-USA seeks to ensure that U.S. intellectual property law serves to promote the progress of science and the useful arts in a way that is consistent with the principles set forth by our Nation's Founders. IEEE-USA's members have a substantial stake in the United States patent system. Our membership includes inventors who create and use cutting-edge technology, researchers who are involved in scientific

---

[1] Pursuant to Supreme Court Rule 37.6, counsel listed on the cover states that this brief was authored by *amicus curiae* IEEE-USA and reviewed by counsel, and that counsel for a party did not author this brief in whole or in part. Nor did counsel for a party make a monetary contribution intended to fund the preparation or submission of the brief. In addition, all parties have consented to the filing of this *amicus* brief, and their consent letters are on file with the Clerk's office as of December 11, 2013.

discovery, authors of journal articles in the fields of electronics and computer science, entrepreneurs, and employees of firms that acquire, license, and market patented technology. Because of the stake that IEEE-USA membership has in the outcome of this case, and because of its public interest in promoting activities in the fields of electronics and computer science, IEEE-USA is fully aware of the need for pioneer developers of technology to be able to capture the value of their inventions, as well as the need of future market entrants to be able to clearly identify where there is still room for development of follow-on technologies. Because of such awareness, IEEE-USA fully understands the need for a balanced patent system. IEEE-USA submits that its broad experience and balanced perspective will be helpful to this Court in its deliberations. IEEE-USA has filed a number of *amicus* briefs in a number of cases, and it is notable that this Court adopted the "foreseeable bar" standard suggested by IEEE-USA, in its decision in *Festo Corp. v. Shoketsu Kinzoku Kogyo Kabushiki Co.,* 535 U.S. 722 (2002).

SUMMARY OF ARGUMENT

Computer-implemented inventions as a class are too important to deny patent protection. There are nearly 1 million software-related U.S. patents in force today on which the public relies. In this case the Court is asked to decide whether computer-implemented inventions are patentable under 35 U.S.C. § 101. On both legal and technological grounds, IEEE-USA believes the answer is clearly yes.

While the parties and other *amici* will cover sufficiently the law and patent policy, IEEE-USA's expertise is in the technological innovation at issue in this case. On the basis of this expertise, IEEE-USA submits that phrases such as "software patent" or "patent on software" are technologically inaccurate and misleading. Thus it is necessary to present the technological facts about software-implemented inventions, which the IEEE-USA is uniquely qualified to explain. In turn, this explanation will make clear why these modern-day inventions are exactly the type of inventions that the patent system was created 224 years ago to protect. See *Bilski v. Kappos*, 130 S.Ct. 3218 (2010) (stating that Section 101 of the U.S. patent law is a "dynamic provision designed to encompass new and unforeseen inventions").

Ultimately, this brief is necessary given widespread misunderstanding among lawyers, judges, and commentators about the nature of the innovations produced by the economically dynamic and vibrant software industry—an industry made possible in part by the intellectual property protections afforded to its innovators.

ARGUMENT

I. The Court's jurisprudence includes recognition that the broad scope of patent eligible subject matter encompasses computer-implemented inventions

The Supreme Court has held that the categories of patent-eligible subject matter recited in Section 101 are broad in their scope and limited only by three important judicially created exceptions. That is, "[l]aws of nature, natural phenomena and abstract ideas" are excluded from patent eligibility because such fundamental discoveries represent "the basic tools of scientific and technological work" *Gottschalk v. Benson*, 409 U.S. 63, 67 (1972). Thus, the Court has concluded that abstract ideas are to be treated as being equivalent to laws of nature and natural phenomena. Accordingly, patent-eligible inventions cannot include representations of relationships that always existed, as well as things that are purely of nature and not made by man. *Parker v. Flook*, 98 S.Ct. 2522, 2527 (1978) and *Diamond v. Chakrabarty*, 100 S.Ct. 2204, 2207 (1980). However, the method or means of application of a scientific principle, law of nature, or abstract idea, for the production of a useful result, is a proper subject for a patent.

In this regard, this Court's decision in *Diamond v. Diehr*, 450 U.S. 175 (1981) taught that in a § 101 inquiry, each element of an invention is measured against a § 101 standard, that is, whether there is at least one element that is neither *"abstract"* (purely in the human mind) nor *"natural"* (unaided by

human intervention). *Diehr* explained that the analysis for the *kind* of invention eligible under § 101 is "wholly apart" from the analysis for whether the invention or any element is *new* and differentiated from the prior art.

Instead, the *Diehr* Court noted that Diehr's claim—*as a whole*—applied well-known scientific and abstract algorithm concepts to *a new context*, an improved and practical process for curing rubber. *Diehr*, 450 U.S. at 184. In essence, any single limitation that could not be performed in the human mind or by nature unaided by human intervention—*e.g.*, measuring a temperature of a physical process, or opening an industrial mold press—converted Diehr's recognition of applicability of an "abstract" principle, such as an algorithm, into a patentable *application* of that principle. The *Diehr* Court expressly held that the old-or-new status of individual components of Diehr's invention was *irrelevant* to a § 101 analysis. *Id.* at 193 n. 15. However, because Diehr's claim recited language that could not be performed by nature unaided, and could not be performed in the "abstract" by the human mind, the claim *as a whole* satisfied § 101. *Diehr*, 450 U.S. at 191.

As pointed out in this Court's recent decision, *Bilski v. Kappos*, 130 S.Ct. 3218 (2010), Section 101 of the U.S. patent law is a "dynamic provision designed to encompass new and unforeseen inventions." *Id.* at 3227, citing. *J. E. M. Ag Supply, Inc. v. Pioneer Hi-Bred Int'l, Inc.*, 534 U.S. 124, 135, 122 S.Ct. 593, 151 L.Ed.2d 508 (2001). As this Court recognized in *Bilski*, a physical-based "machine-or-

transformation test may well provide a sufficient basis for evaluating processes similar to those in the Industrial Age —for example, inventions grounded in a physical or other tangible form. But there are reasons to doubt whether the test should be the sole criterion for determining the patentability of inventions in the Information Age." *Id*.

IEEE-USA notes this Court's cautionary approach in *Bilski*; that a "machine-or-transformation" test may be too restrictive for an "Information Age," computer based economy. As IEEE-USA will present in the next section, a machine-or-transformation test is certainly inappropriate, unless "transformation" is interpreted to include the kinds of physical transformations that are the foundation of digital computers and microprocessor based technologies.

Further, IEEE-USA urges the court to consider reliance by industry, dating from at least the Federal Circuit's opinion in *In re Alappat*, 33 F. 3d 1526 (Fed. Cir. 1994) (en banc), on the patentability of computer-implemented inventions. In that decision, the Federal Circuit held that a computer program "is not a disembodied mathematical concept which may be characterized as an 'abstract idea.'" *Id.* at 1544. A computer program, or a sub-program, according to that Court, is the digital equivalent of "a specific machine." *Id.* Thus, a software program, when combined with the hardware of a computer, qualifies as an invention of a digital "machine." For example, a word processing program is a digital equivalent of a typewriter – what the former does in computerized digital format the latter does in a mechanical, analog

format. *See* Adam Mossoff, "A Brief History of Software Patents (and Why They're Valid)" (Sept. 2013), at 6, http://cpip.gmu.edu/wp-content/uploads/2013/08/A-Brief-History-of-Software-Patents-Adam-Mossoff1.pdf.

The importance of the *In re Alappat* decision for the industry was greatly magnified by decisions shortly thereafter. In the case of *Lotus Dev. Corp. v. Borland Int'l, Inc.*, the First Circuit held that Lotus could not copyright its pull-down menus because these were a functional "method of operation" (*i.e.*, a utilitarian design, and not an expressive text capable of receiving copyright protection). *Lotus*, 49 F.3d 807, 815 (1st Cir. 1995). This Court (as a result of a 4-4 split) affirmed the First Circuit ruling. *Lotus*, 516 U.S. 233, 116 S. Ct. 804; 133 L. Ed. 2d 610 (1996).

As this Court has repeatedly recognized, when contrasting patents against other IP regimes, such as copyright and trademark, "it is the province of patent law" to secure "new product designs or functions." *Qualitex Co. v. Jacobsen Products Co., Inc.*, 514 U.S. 159, 164 (1995); *Elmer v. ICC Fabricating*, 67 F.3d 1571, 1580 (Fed. Cir. 1995) ("patent law, not trade dress law, is the principal means for providing exclusive rights in useful product features"). *See also Baker,* 101 U.S. at 102 ("[T]he exclusive right to the art or manufacture … is the province of letters-patent, not copyright").

Other than trade secrecy, an option that is often not available (e.g., if use of the invention is inherently disclosing to the public or if, as is common

today, employee turnover is high), a patent is essentially the only practical option for securing engineering innovations as embodied in or implemented using a computer program.

II. Computer-implemented inventions that use mathematical algorithms, and/or implement business methods, are like any other real, and physics-based, technology

IEEE-USA submits that phrases like "software patent," or a "patent on software," are technologically inaccurate and misleading. IEEE-USA fully supports the framing accepted by the Court, that the kind of inventions at issue here are best described as "computer-implemented inventions," but wishes to assist the Court with further clarification of the phrase.

IEEE-USA believes that it is in a unique position to contribute technical expertise to the debate over so-called "software patents." The field of electrical engineering has been the basis of virtually all computing hardware since the field of general-purpose computing's very beginnings in the 1940s. The IEEE, through its predecessor organizations (the Institute of Radio Engineers and the American Institute of Electrical Engineers), has been an important part of electrical engineering's development since 1884.

II.A Equivalence of software and hardware

From the time of its founding, and up to today, the IEEE has continued to adapt to the field's many changes. Among the field's most important trends, that started from at least the 1970s, and is continuing to present day, is the increasing importance of the digital side of electrical engineering. This growth of importance is due primarily to two factors:

1. With each new generation of digital hardware technology, general-purpose computing hardware has become faster and/or less expensive. (Each new generation of digital hardware can also possess additional or alternative advantages, such as greater miniaturization and/or lower power consumption. For clarity of exposition, we have limited our focus to increased speed and/or lowered cost.)

2. The principle of equivalency, that holds that special-purpose programming of general-purpose hardware can be made equivalent to special-purpose hardware. (The equivalency can be made exact when the conversion is from digital special-purpose hardware to special-purpose programming of general-purpose hardware. On the replacement of analog special-purpose hardware, there will always remain situations, such as transducing by sensors, where analog hardware will remain essential. Also, quantization, a necessary step for substitution with digital techniques, is inherently an approximate representation, of continuous real world values.)

With each new advance of general-purpose hardware, on the frontiers of processing speed and/or affordability, additional possibilities for utilizing the principle of equivalency emerge as commercially viable. Thus, over the decades, there has been a continual trend, no less vibrant today, of re-implementing functionality of a special-purpose hardware solution as an equivalent functionality using special-purpose programming of general-purpose hardware.

Also, as general-purpose hardware becomes more common, economies of scale emerge that further encourage this trend. It is this trend, towards increased use of the principle of equivalency, which has been the main driving force behind the appearance, before this Court today, of a case such as *CLS Bank International v. Alice Corp. Pty. Ltd.*

An example technology, with extensive current investment in re-implementing functionality with software, is the field of wireless communication. Since the earliest days of radio-frequency technology, which began about 100 years ago, radio signals were modulated (*e.g.*, varied in frequency, amplitude, or phase) using analog circuitry. This is because digital devices, even when they became available, were historically unable to operate at radio frequency in an economically viable way. Currently, however, there is a great deal of investment in re-implementing radio signal modulation, with general-purpose digital hardware programmed to run special-purpose software. (Such software is called "embedded firmware" or "microcode." *See* Fanny

Mlinarsky, "Multimode wireless devices: It's the software, stupid!" *Mobile Handset DesignLine TechOnLine Community website*, October 13, 2009, http://www.octoscope.com/English/Collaterals/Articles/octoScope_MultimodeSoftware_DesignLine_20091013.pdf.)

Another and related example is the ubiquitous presence of software-related inventions in smart phones, as documented by economists at the U.S. Patent and Trademark Office ("PTO"). *See* S. Graham and S. Vishnubhakat, "Of Smart Phone Wars and Software Patents," 27 *Journal of Economic Perspectives*, 67 (Winter 2013), ("PTO 2003"). This paper states that the PTO reviewed all 73 patents involved in some of the high-profile litigation, among four major firms in the smart phone industry: Motorola, Microsoft, Apple, and Samsung. The PTO found that 65 of the patents (9 out of 10) included at least one software-related claim. PTO 2003 at 73.

The functional equivalence, between hardware and software, means that the choice, of how much of a particular system to implement with each, is really just a practical, market-driven, design issue. A primary driver, towards increased use of software, is the speed with which software can be written, and rewritten. Implementing as much as possible in software is almost always the most cost-effective way to reduce time-to-market. Once on the market, in order to stay competitive, products often need frequent updating, with new or modified features. An implementation in software almost always provides the quickest and most cost-effective route, by which such updates can be accomplished.

The main advantage of a hardware implementation of an algorithm (relative to software) is that it is almost always faster. If the extra speed is of paramount importance, the extra time-to-market, to produce a hardware implementation, becomes the economically sensible choice. An example area, where speed can be the predominant factor, is real-time video processing. This is because each frame of a video represents a large amount of data (at least a megapixel, for U.S. HDTV), and, in order that any motion represented by the video appear realistic (*i.e.*, take place in real time), the frames must be produced as a continuous, steady stream (for U.S. HDTV, frames must be output at a constant rate of 30 per second).

Further, as will be explained below, there are additional important reasons why the distinction, between a so-called "software-implemented system" and a so-called "hardware-implemented system," is really very artificial. As will be explained below, every general-purpose computer is ultimately based on specialized hardware. The general-purpose character of general-purpose computers results from the fact that, eventually, all such systems can perform the same functionality – albeit at very different speeds. (Speed differences are, of course, very important in practical applications.)

For this reason, every "software" solution includes at least some special-purpose hardware. Conversely, many "hardware" solutions contain at least some software. Also, to the extent special purpose hardware solutions are still needed, there

has been a long term trend towards making its design more like the writing of software. Rather than working with wires and circuits, today's hardware designer is much more likely to be writing programs, or "software," that is automatically translated into a circuit. The languages in which such hardware is written can be the same as those used for software (*e.g.*, the "C" programming language). More frequently, the hardware is programmed in a language, called a "Hardware Description Language," specifically designed for expressing hardware.

A practical design choice, like choosing how much to implement in software and how much in hardware, that is constantly changing and evolving due to market conditions, should not be a controlling factor (or a factor at all) as to whether the implemented method is patent-eligible.

II.B The execution of software depends on real, physics based, processes

A computer is always a real physical system. It shares many of the same qualities of any other physical system. For example, research into the fundamental connections between physics and information processing has shown that *any computation*, if it is to be performed in less than an infinite amount of time, *must consume at least some energy*.

Another way of saying this is that, in order to "calculate" or, more correctly, "compute," a computer

must make changes to its physical state. Being able to reliably change state is a general engineering problem that has been addressed, and continues to be addressed, using engineering solutions.

Almost all computer design, at least to date, has been of a kind that is very intolerant of errors in the hardware. Even a few "flipped bits" (*i.e.*, bits of data that, for some physical reason, spontaneously change from "1" to "0" or from "0" to "1"), among the billions of bits that are often involved in executing a typical computer program, can cause that program to "crash" (*i.e.*, be forced to totally end its current execution).

The choice of binary arithmetic, as the basis for almost all current computers, was not driven by mathematics. Binary arithmetic was chosen because it can be implemented, entirely, with very simple on-off switches, and because on-off switches can be made extremely reliable.

It is a precept of engineering design that one can minimize the opportunities for errors by minimizing the number of operations or steps in any system or process. Accordingly, when seeking to give a physical representation to numbers, it is generally most reliable if the physical system used undergoes the least number of physical transformations. Binary arithmetic reduces that number of transformations to the least possible: two.

II.B.1 Every bit of information, stored in a computer, depends on a physical device

Every "bit" of information, in every binary-based computer system, is really only a particular form of physical device, capable of changing between one of two physical states.

Over the decades, a vast array of physical devices have been designed, for storing a bit of information for use by general-purpose computers. The following is only a small sample of the types of physical storage devices.

Hard Drives: The "hard drive" of most computers is still based upon magnetic technology. In a magnetic memory technology, each bit of information stored is represented by the direction at which a small item of magnetic material is magnetized. In a hard drive, the magnetic material is organized into the shape of a "disk" or "platter," so that it can literally be spun in a circle.

Digital memories based on magnetic technology have been long known to have the following advantage: they are referred to as "non-volatile." This means that, once a small item of magnetic material has been magnetized in a particular direction, it will hold that magnetization for a very long time (*e.g.*, at least years and often decades). A disadvantage of magnetically-based memories is generally their relatively slow speed of access.

RAM: The "RAM" (or Random Access Memory) of most digital computers is currently based upon

the principle of charge storage. Each bit of information stored in a RAM is based upon whether a tiny amount of electric charge is, or is not, stored, on a tiny piece of electrically conductive material.

A well-known disadvantage of this type of RAM is its volatility. This is because the electric charge, once stored, will tend to "leak off" quickly (*e.g.*, within seconds). For these types of memories, the way information is continuously kept, for the hours or days a computer may continuously operate, is by a constant scanning and "refreshing," where needed, of each electric charge stored. This type of RAM is often called "Dynamic" RAM, or DRAM, because, unlike magnetic memories, it must constantly be refreshed in order to maintain its contents.

Microprocessor Chip: A microprocessor chip, which forms the heart of such things as personal computers (*e.g.*, desktops, laptops, and tablets), will include storage that needs to be particularly fast. Each bit of such storage can be accomplished with a kind of circuit referred to generally as a "latch" (or "flip flop").

The basic principle of a latch is as follows. A latch has two main modes of operation, which we shall refer to as "storage mode," in which the latch maintains the content already written into it, and "write mode," during which the contents of the latch can be changed. In storage mode, the latch maintains its content (which is either a "0" or a "1") through use of a self-reinforcing feedback loop. In particular, the output of the latch is fed back into its "feedback" input. For example, if a "0" is currently

stored in the latch, the "0" output of the latch is fed back into its feedback input, and that feedback causes the latch to continue to output a "0." A "1" is stored in the latch in the same way, except the signal, which forms the feedback, is of an opposite polarity. Because of the self-reinforcing feedback, so long as power is supplied, a latch can retain, indefinitely, the bit of information that has been stored.

In write mode, the feedback loop is temporarily broken. Because of this, the latch's output is able to provide the same bit of information present at the latches "data" input. When the transition back to storage mode is made, the latch will retain (or remember) the data present at the latch's data input, because of the re-establishment of the feedback loop.

The net result is that changing the contents of a latch is actually a physical transformation, of the polarity of its feedback signal.

II.B.2 The special-purpose hardware of general-purpose computers

Thus far has been described, as a physical system, the most basic capability of any information processing device – the ability to store information. We will now address the three basic kinds of special purpose hardware necessary, in any so-called "general purpose" computer, for making use of the information stored.

First, special-purpose hardware is needed for determining how a pattern of bits of information, stored in an area of the computer referred to as the program (or algorithm) store, are recognized (or interpreted) as a particular kind of "instruction" to be performed (*e.g.*, an instruction to "add" two numbers). Second, some kind of sequencing hardware is needed, for determining an ordering by which instructions are executed. This ordering is often accomplished by a special-purpose item of digital hardware called a "program counter." In addition, the instruction set for a computer almost always includes instructions that change the order in which future instructions are executed. Third, special-purpose hardware is needed to actually perform the operation specified by an instruction. The two main types of operations, performed by an instruction, can be referred to as "mathematical" or "flow-of-control." As the name suggests, mathematical operations perform some kind of mathematical function or operation. The bits operated on, by mathematical instructions, are stored in an area of the computer referred to as the "data" store. The bits of a data store are often called data "variables" or "records." In contrast, flow-of-control operations change the sequencing, according to which future instructions are executed.

An example mathematical operation is performed by the above-mentioned "add" instruction. This instruction requires a special-purpose item of hardware, called an "adder," to actually perform the addition. Some instructions will perform both a mathematical operation and a flow-of-control operation.

To provide a concrete example, of where the program and data stores of a computer may reside, each can be a region of "main memory" (with main memory being implemented by the RAM technology discussed above). It should be noted that both the program store and data store could be designed to utilize the same (or overlapping) regions of a single main memory, as a matter of design choice. Thus, so-called "data" of the data store, which is operated upon in response to program-store instructions, can later be interpreted, by the computer, as instructions.

Regardless of the operation type, the operations performed by a single instruction are very simple. It is rare to find a general-purpose computer where any of the hard-wired mathematical operations go beyond the basic operations of arithmetic. Many operations are so simple, they are hard to even understand as representing the "processing" of data. For example, many instructions just have the net effect of copying an item of data, from one region of the data store to another.

The total set of instructions, that are actually hard-wired into a particular general-purpose computer, is called the computer's "instruction set." Thus, at its most basic (or hard-wired) level, the capabilities of a general-purpose computer are extremely primitive and simple.

Everything else about a general-purpose computer, that makes it easier for humans to use, is a result of "software." The "intelligence" or

"abstraction" abilities, provided by such software, are simply a result of vast numbers (*e.g.*, millions) of instructions in the program store, physically processing vast numbers (*e.g.*, billions) of bits in the data store.

For example, many people have heard of IBM's "Watson" computer, which appears to be able to play the game of "Jeopardy!" in many cases better than a human. Most people would probably call this a kind of "intelligent" behavior. However, such intelligent behavior is still relatively uncommon, and most programs are important not because they appear to be "intelligent," but because they provide a level of "abstraction," when dealing with a particular task, that allows humans to work faster. Another way of saying this is that the computer does those parts, of accomplishing a kind of task, that a computer is particularly good at (*e.g.*, crunching lots of numbers through complex equations), leaving the human to focus on those parts of a task in which human intelligence provides the greatest "value add."

II.B.3 What is "general," about a general-purpose computer?

Even though every general-purpose computer is based upon a particular specialized set of choices for its hardware configuration (*e.g.*, such as the particular instructions included in its hard-wired instruction set), all such computers are referred to as "general-purpose" for the following reason: as long as a few well-known basic capabilities are included in the hardware of a computer design, every general-

purpose computer can, *eventually*, perform any of the operations or computations of any other general-purpose computer. Remarkably, it has been shown that the vast array of potential differences in specialized hardware and specialized instruction set, possible when designing a general-purpose computer, amount only to *differing the speed*, at which certain general-purpose computers can accomplish certain kinds of algorithms, in comparison with other kinds of general-purpose computers accomplishing other kinds of algorithms.

Most users of a type of device they would call a "personal computer" (as mentioned above, these include desktops, laptops, and tablets) have an intuitive understanding that their device has extremely wide (or "general-purpose") applicability, due to the fact that tens of thousands of "apps" (*i.e.*, separate items of application software) are typically available for them.

For purposes of this *Amicus* Brief of IEEE-USA, however, the phrase "general-purpose computer" has been used in its broadest sense, as referring to the core internals of how a device operates, and not to the externals, as a user or consumer may experience them. For example, there are many products that a consumer experiences as special-purpose, but that are really, internally, a combination of general-purpose hardware with special-purpose programming. Such combinations are often referred to as "embedded systems" or "microcontrollers." Even now, a typical office, home or automobile contains many such embedded

systems, each of which is performing, from the user's perspective, a limited repertoire of functionalities.

II.C Claim 33 of the '479 patent

As an example of the foregoing discussion (of the physics and special-purpose hardware of program execution), changes in state occur when carrying out the steps of claim 33 of Alice's U.S. Patent 5,970,479 ("the '479 patent"), on which particular focus has been placed. The steps of claim 33 provide clear evidence that the method described therein has been sufficiently applied (although IEEE-USA takes no position as to whether the '479 patent should be found valid, with respect to statutory criteria other than §101).

Step (a) requires establishing a shadow credit record and a shadow debit record for each stakeholder party. This requires that a computer – the computer of the supervisory institution – create data files assigned to individual stakeholder parties, and change each to an appropriate state. The data files may be stored in main memory or on an external mass storage device (such as a hard drive).

Similarly, step (b) also requires the supervisory institution computer to perform machine operations. These include initializing the shadow credit record, and the shadow debit record, with bit values corresponding to a start-of–balance value. The start-of-balance values are obtained from each exchange institution. These machine operations include accessing the computer's data store, in order to write

start-of balance values into each shadow credit and debit record.

Next, in step (c), in response to every transaction received from each stakeholder (*e.g.*, via a communications network), the supervisory institution computer performs a series of machine operations, as directed by its allocated process, that adjusts each respective party's shadow credit record or shadow debit record (with each of the adjustments being made in chronological order). The series of machine operations, when so performed, produce further changes of state in response to every transaction that results in an exchange obligation. A stakeholder is only permitted certain transactions (as determined by its computer controlled process), for purposes of adjusting its shadow credit record and shadow debit record (*i.e.* those which do not result in the value of the shadow debit record being made less than the value of the shadow credit record at any time).

Lastly, in step (d), at the end-of-day, the supervisory institution computer, as directed by its process, instructs exchange-institution computers to exchange credits or debits (*e.g.*, via a communications network). The exchange is made to the credit and debit records of the respective stakeholder parties, according to the adjustments specified in permitted transactions. The credits and debits are made irrevocable, with time invariant obligations placed on the exchange institutions.

It seems clear that the method steps of claim 33 require the performance of a number of database

operations (*e.g.*, writing to and updating records in files, based on specific types of transactions). Among other changes, changes of state are caused in the shadow credit and debit records. Thus, carrying out the method steps of claim 33 requires a computer to perform actions upon physical things or objects, sufficient to conclude that the method has been applied, and is therefore patent eligible.

II.D Expect the unexpected

The trend, of increasing use of the principle of equivalency, is far from being the only kind of change to which the IEEE is adapting. In the most recent decades, there has been a fascinating trend towards convergence, among the previously-separate physical sciences. For example, fields as previously distinct as molecular biology and electrical engineering are experiencing ever increasing levels of interaction, with educational institutions responding by offering majors in this precise area. Such interactions are certain to produce many "unforeseen inventions," as the Court says in *Bilski*, for decades to come.

III. Software is pervasive. Computer-implemented inventions are too important to a 21st century economy to deny patent protection

The software industry has undergone (and continues to undergo) spectacular growth since the decisions of *In re Alappat* and *Lotus*, in the mid-

1990s. In early 1996, the PTO adopted and published its "Examination Guidelines for Computer-Related Inventions," 61 *Fed. Reg*. 7478, (February 28, 1996) on which applicants relied to obtain patent claims to protect their technology developments. Figure 1 and Table 1 in the Appendix depict the historical sharp rise of software-related patents in force following these developments. They show that as of the end of 2012, nearly 1 million such patents were in force, a substantial fraction of all 2.23 million utility patents in force at the end of that year (See Table 2). The fate of these patent assets and the vast investments made in reliance thereupon depend on this Court's decision in this case. Going forward, an analysis by the United States Government Accountability Office ("GAO"), shows that approximately 50% of issued patents in recent years are software-related patents. GAO, *Intellectual Property: Assessing Factors that Affect Patent Infringement Litigation Could Help Improve Patent Quality*; GAO-13-465, (August 2013).

Patents on computer-implemented inventions are crucial to investment in innovation, because the most important and innovative ideas are often the most risky. Patents reduce that risk, and thereby increase the incentive to invest. The provision of property rights for software-implemented inventions thereby helps ensure that capital flows to new good ideas.

Once initial investments have been made, and a learning curve is established, patents continue to be important for assuring further investment will also be protected. The processes of continuing the

operation of a software-based company (*e.g.*, coding, testing, debugging, integrating with other software, seeking any regulatory approval, developing marketing and distribution channels) are far more expensive than just the initial invention (and its "seed capital"). Patents are essential to protecting that continued investment.

IV. The consequences of a failure to preserve the patent-eligibility of software-implemented solutions are numerous and harmful to our country

Our 21st century economy depends on software-implemented systems. The Court must protect the public's settled expectation that nearly 1 million patents on such systems, particularly those that protect large investments in system development, will remain valid.

A failure to preserve the patent-eligibility of software-implemented solutions could result in a number of unintended consequences. More specifically, holding method, system, and product inventions, similar to those described in the Alice patents in suit, to be patent-ineligible could cause more inventors and companies to maintain trade secrets, which could result in a reduction of knowledge sharing in the engineering and academic arenas, fewer collaborations and strategic alliances among inventors, suppression of free exchange of technological ideas, and a reduction in the amount of investment for innovation itself.

In addition, because developers will take all necessary steps to ensure that their vast research and developments remain protected to the extent possible, a failure to preserve the patent-eligibility of software-implemented solutions may prompt developers to develop ways to maintain or extend trade secrecy in such software-related solutions. Holding software implemented method, system, and product inventions to be patent-ineligible could cause new software protection mechanisms to prohibit unauthorized use, sale, or reverse engineering of such systems.

For example, in order to frustrate reverse engineering, developers may utilize modern techniques to incorporate software implemented solutions into field-programmable chips, even at the loss of design flexibility or system performance. Developers may incorporate reverse engineering blocking mechanisms into product design. Further, developers may even choose to eliminate distribution of software documentation in order to shield the nature of such inventions outside of their companies. More employees may be required to sign covenants not to compete. Thus, we could see a return to the economic landscape of marketing and sale of software-implemented systems in the 1960s and 1970s, in which the design and implementation details relating to software-implemented innovations were protected through secrecy agreements.

The failure to preserve the patent-eligibility of software implemented solutions could result in greater reliance on copyright protection and possibly cause more patent applications to be first filed in

countries outside of the U.S. that offer greater protection of software implemented inventions (*e.g.* Australia, Japan, and others). Lastly, the failure to preserve patent-eligibility of software implemented solutions could result in companies transferring their software development operations outside the United States, resulting in a loss of jobs in the high-tech sector and a drain of U.S.-based knowledge out of the country.

IEEE-USA respectfully submits that failing to preserve the patent-eligibility of software implemented solutions, or even diminishing the scope of their patent-eligibility, will lessen our country's ability to fulfill the directives of the U.S. Constitution "[t]o promote the progress of science and useful arts."

CONCLUSION

IEEE-USA therefore urges this Court to rule on the question presented in the affirmative.

Respectfully submitted,

IEEE-USA
Amicus Curiae

By:

Chris J. Katopis
Counsel of Record
1308 Clifton Street, N.W.
Washington, DC 2000
Tel. (202) 200-9490
chriskatopis@gmail.com

Counsel for Amicus Curiae
IEEE-USA

28 JANUARY 2014

APPENDIX

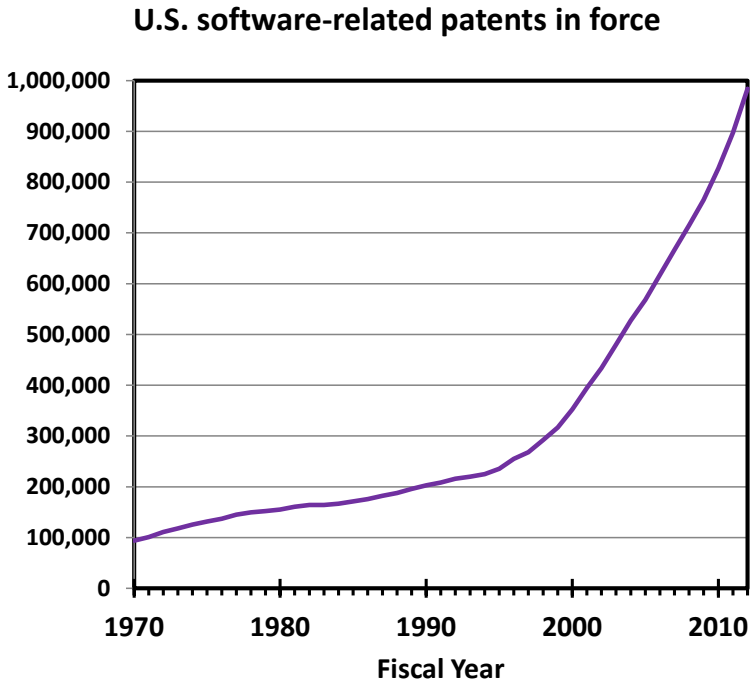

Figure 1

"Software-related" patents were identified using the definitions provided in PTO 2003 at 77 (footnote 7). This definition is based on the US patent classes and subclasses that were determined by PTO experts to contain patent applications or issued patents containing some element of either general purpose software or software that is specific to some form of hardware. This is the same definition used in the GAO 2003 report. Using these classes and subclasses, a search on FeePatentsOnline.com ("FPO Database") confined to appropriate date ranges produced the counts shown in the attached tables.

For comparison, the number of utility patents in force by the end of each fiscal year is also provided. The number of issued patents each year was obtained from the PTO's annual reports. Table 1 shows the results for software-related patents in force whereas Table 2 pertains to all utility patents in force.

The PTO's Technology Assessment and Forecast ("TAF") database was used to obtain general patent maintenance statistics in order to account for early expiration of patents before their nominal term. The 17-year patent term laws were changed in 1995 under the GATT legislation and 35 U.S.C. § 154(a)(2) provides that patents filed on or after June 8, 1995 have a term of 20 years from the priority application date. Under § 154(c)(1) the term of a patent that was in force on or that results from an application filed before June 8, 1995 is the greater of the 20-year term, or 17 years from grant. Because no patents expired yet under § 154(a)(2), the only two categories of patents analyzed separately are those having the 17-year term, shown as "Old Patents" in the tables, and those governed by § 154(c)(1), shown as "patents subject to GATT effects." Expirations from both categories were combined.

Patent term adjustments under 35 U.S.C. § 154(b) began taking effect only for patents issued after 2000 and as such, it is not manifested in patent expirations for patents issued to date. The number of patents extended due to the FDA approval process under 35 U.S.C. § 156 is negligible for the purposes of this analysis, as PTO data in 78 *Fed. Reg*. 31886 (May 28, 2013) shows

that the terms of only about 70 such patents per year are extended.

The results obtained in this analysis for U.S. utility patents in force are within 1% of the limited results published by the World Intellectual Property Organization for the years 2004-2012. See "WIPO IP Statistics Data Center," Patents In Force Indicator, at http://www.wipo.int/ipstats/en/statistics/patents/.

Table 1

FY	PATENTS SUBJECT TO GATT EFFECTS				MAINTENANCE			OLD PATENTS		Software-related patents in force
	Application date > June-7-1975							Issued < June-8-1978		
	Issued	Nominal term expired (long pendency)	Nominal term expired (mid pendency)	Nominal term expired (short pendency)	Total expired in FY due to failure to renew	Fraction issued in FY that expired any time before nominal term	Total issued in FY that expired any time before nominal term	Issued	Nominal term expired	
t	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*k*	*m*
Source	FPO Database				Scaled from PTO TAF database			FPO Database*		Calculated **
1970								12,620	5,002	94,138
1971								11,123	4,548	100,713
1972								13,733	3,384	111,062
1973								12,081	4,933	118,211
1974								12,709	5,568	125,352
1975								11,875	5,714	131,512
1976								13,044	7,268	137,288
1977								15,343	7,659	144,972
1978	11,636								6,790	149,819
1979	9,459								7,234	152,044
1980	10,436								7,383	155,097
1981	12,057						25		6,263	160,891
1982	11,917						1,596		8,401	164,407
1983	10,765						5,480		10,976	164,196
1984	13,208						8,126		10,879	166,524
1985	14,616				0	0.6390	9,340		9,948	171,192
1986	15,974				416	0.6286	9,902		10,986	175,764
1987	20,324				1,411	0.6205	12,682		12,620	182,056
1988	18,697				2,019	0.6068	11,418		11,123	187,611
1989	24,179				2,227	0.5893	14,191		13,733	195,831
1990	21,966				2,886	0.5905	13,156		12,081	202,829
1991	23,476				5,114	0.5927	14,607		12,709	208,482
1992	25,891				6,698	0.5725	14,524		11,875	215,799
1993	26,106				8,835	0.5593	14,846		13,044	220,026
1994	29,818				9,845	0.5478	16,394		15,343	224,656
1995	31,873	391			13,027	0.5338	16,930		7,484	235,627
1996	33,806	1,035	1,059		12,460	0.5134	18,130			254,879
1997	36,020	1,569	4,950	2,703	13,516	0.5181	18,607			268,161
1998	48,898	2,214	6,209	3,474	12,832	0.5261	27,055			292,334
1999	50,336	2,344	7,018	2,658	14,098	0.5087	27,239			316,881
2000	59,438	2,797	7,204	2,824	13,522	0.4940	27,787			352,367
2001	62,692	4,102	6,145	2,369	13,828	0.3232	19,543			394,584
2002	61,885	4,962	7,048	1,824	17,177	0.3313	20,937			433,895
2003	69,023	4,167	6,066	2,058	18,634	0.3115	20,771			479,719
2004	73,500	4,544	6,957	3,588	20,107	0.2986	20,878			527,464
2005	68,188	3,430	8,862	4,850	20,755	0.1290	8,142			568,386
2006	79,827	4,302	7,891	6,918	23,366	0.1466	12,033			617,396
2007	79,946	3,285	9,590	7,376	22,615	0.1340	10,033			666,576
2008	80,348	3,495	8,543	10,287	23,172	0.1292	9,931			714,756
2009	85,351	4,251	9,510	10,137	25,529	0.0000				765,053
2010	106,902	4,264	10,699	10,471	34,550	0.0000				826,912
2011	115,020	4,994	10,817	10,940	32,844	0.0000				897,371
2012	132,244	5,167	12,788	11,025	32,818	0.0000				983,862

Notes:

* Except for 1995, where applications filed on or before June 7, 1995 are shown per search in FPO database

** Software-related patents in force at yeat *t* calculated as follows

For $t < 1995$: $m(t)=m(t-1)+a(t)+h(t)-e(t)-k(t)+g(t-17)$; *For* $t \geq 1995$:

$m(t)=m(t-1)+a(t)+h(t)-b(t)-c(t)-d(t)-e(t)-k(t)+[b(t)g(t-17)/a(t-17)+c(t)g(t-18)/a(t-18)+d(t)g(t-19)/a(t-19)]$

Last terms involving column *g* are corrections to avoid double counting of expirations included in column *e*

Table 2

FY	PATENTS SUBJECT TO GATT EFFECTS				MAINTENANCE			OLD PATENTS		Utility Patents in Force
	Application date > June-7-1975							Issued < June-8-1978		
	Issued	Nominal term expired in FY (long pendency)	Nominal term expired in FY (mid pendency)	Nominal term expired in FY (short pendency)	Total expired in FY due to failure to renew	Fraction issued in FY that expired any time before nominal term	Total issued in FY that expired any time before nominal term	Issued	Nominal term expired	
t	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*k*	*m*
Source	Annual Report	FPO Database (Utility Patents)			Annual Report	PTO TAF database		Annual Report*		Calculated **
1970								66,334	41,739	856,119
1971								70,370	37,602	888,887
1972								83,215	28,643	943,459
1973								67,491	39,416	971,534
1974								79,262	43,574	1,007,222
1975								70,132	41,928	1,035,426
1976								75,311	50,174	1,060,563
1977								84,197	48,458	1,096,302
1978	65,963								45,454	1,116,811
1979	51,686								49,277	1,119,220
1980	56,618								52,532	1,123,306
1981	66,617						136		42,538	1,147,385
1982	59,449						7,962		51,385	1,155,449
1983	54,744						27,869		65,237	1,144,956
1984	66,753						41,070		70,022	1,141,687
1985	69,667				1	0.632	44,025		61,596	1,149,757
1986	71,301				1,121	0.629	44,816		61,949	1,157,988
1987	82,141				6,172	0.621	50,972		66,334	1,167,623
1988	77,317				11,224	0.607	46,914		70,370	1,163,346
1989	95,831				12,416	0.589	56,473		83,215	1,163,546
1990	88,974				12,060	0.591	52,539		67,491	1,172,969
1991	91,822				19,134	0.593	54,420		79,262	1,166,395
1992	99,405				28,603	0.572	56,909		70,132	1,167,065
1993	96,675				38,475	0.559	54,074		75,311	1,149,954
1994	101,270				38,859	0.548	55,477		84,197	1,128,168
1995	101,895	2,214			48,604	0.534	54,396		42,425	1,136,820
1996	104,900	5,658	6,003		60,392	0.513	53,860			1,169,667
1997	111,977	8,513	27,047	15,321	54,485	0.518	58,016			1,176,278
1998	139,297	12,235	33,686	18,981	41,063	0.526	73,284			1,209,635
1999	142,852	11,691	38,778	14,419	52,289	0.509	72,663			1,236,955
2000	164,486	14,222	35,938	15,604	47,958	0.494	81,264			1,299,804
2001	169,571	20,734	31,248	11,820	49,077	0.323	54,806			1,386,743
2002	160,839	23,651	35,620	9,274	53,724	0.331	53,288			1,466,896
2003	171,493	18,600	28,914	10,399	57,770	0.311	53,417			1,559,066
2004	169,295	18,366	31,052	17,102	63,552	0.299	50,546			1,640,011
2005	151,077	14,182	35,816	21,649	67,534	0.129	19,490			1,696,345
2006	162,509	17,051	32,633	27,959	72,654	0.147	23,829			1,755,756
2007	160,306	13,306	38,009	30,502	67,122	0.134	21,475			1,815,887
2008	154,699	13,668	34,606	40,771	67,127	0.129	19,991			1,866,976
2009	165,213	16,322	37,198	41,062	66,330	0.000				1,926,915
2010	207,915	15,791	41,079	40,956	79,993	0.000				2,013,634
2011	221,350	16,962	40,056	42,004	82,146	0.000				2,109,560
2012	246,464	16,520	43,430	40,828	80,050	0.000				2,230,643

Notes:

* Except for 1995, where applications filed on or before June 7, 1995 are shown per search in FPO database

** Utility patents in force at yeat *t* calculated as follows

For $t < 1995$: $m(t)=m(t-1)+a(t)+h(t)-e(t)-k(t)+g(t-17)$; For $t \geq 1995$:

$m(t)=m(t-1)+a(t)+h(t)-b(t)-c(t)-d(t)-e(t)-k(t)+[b(t)g(t-17)/a(t-17)+c(t)g(t-18)/a(t-18)+d(t)g(t-19)/a(t-19)]$

Last terms involving column *g* are corrections to avoid double counting of expirations included in column *e*