APPENDIX A

United States Court of Appeals for the Federal Circuit

WISCONSIN ALUMNI RESEARCH FOUNDATION,
Plaintiff-Appellee

v.

APPLE INC.,
Defendant-Appellant

2017-2265, 2017-2380

Appeals from the United States District Court for the Western District of Wisconsin in No. 3:14-cv-00062-wmc, Judge William M. Conley.

Decided: September 28, 2018

MORGAN CHU, Irell & Manella LLP, Los Angeles, CA, argued for plaintiff-appellee. Also represented by CHRISTOPHER ABERNETHY, GARY N. FRISCHLING, ALAN J. HEINRICH, AMY E. PROCTOR, JASON SHEASBY.

WILLIAM F. LEE, Wilmer Cutler Pickering Hale and Dorr LLP, Boston, MA, argued for defendant-appellant. Also represented by ANDREW J. DANFORD, FELICIA H. ELLSWORTH, LAUREN B. FLETCHER, KEITH SYVERSON.

Before PROST, *Chief Judge,* BRYSON and O'MALLEY, *Circuit Judges.*

PROST, *Chief Judge.*

The Wisconsin Alumni Research Foundation ("WARF") sued Apple Inc. for infringement of U.S. Patent No. 5,781,752 ("the '752 patent"). After a two-week, bifurcated trial, a jury found Apple liable for infringement and awarded over $234 million in damages. The district court denied Apple's post-trial motions for judgment as a matter of law and for a new trial. Because no reasonable juror could have found infringement based on the evidence presented during the liability phase of trial, we reverse the district court's denial of Apple's motion for judgment as a matter of law. With respect to invalidity, we affirm the grant of summary judgment in favor of WARF.

I

The technology at issue relates to how computer processors execute a computer program's instructions. Computer programs are made up of lists of program instructions written in "program order." Although these instructions could be executed sequentially—i.e., in program order—some processors execute program instructions "out-of-order" to improve computer performance.

Of course, when executing instructions out-of-order, the processor must obtain the same result as if it had executed the instructions in program order. This can be complicated by the fact that "data dependencies" may exist between individual program instructions. A data dependence exists between two instructions if one instruction relies upon data produced or modified by an earlier, or "older," instruction in the program order. To illustrate, the parties discuss "store" and "load" instructions that access the same location in memory. Memory can be

thought of as a set of places to store data, where each place has an address by which the contents of that place can be accessed. *See* J.A. 1713 (testimony of Apple's expert, Dr. Colwell). A "store" instruction, or simply a "store," writes data to a given location in memory, overwriting any data that had previously been stored in that memory location. A "load instruction," or "load," reads data from a given memory location and then uses that data to perform some function. *Id.* at 1714. A data dependence exists between a store instruction and a load instruction if (1) the store instruction appears earlier than the load instruction in the program order; and (2) the store and load instructions will access the same memory location—i.e., the same address—if and when the store and load instructions are executed. In such a scenario, the load instruction depends on the store instruction having been executed first so that the data the load instruction reads from memory is current and correct.

At the time a processor decides whether to allow instructions to execute out-of-order, it may be unclear whether a data dependence exists between given store and load instructions. This is called an "ambiguous dependency." J.A. 1429 (testimony of WARF's expert, Dr. Conte), 1715 (testimony of Dr. Colwell). This ambiguity can occur, for example, if the address where the store instruction will store data has not yet been determined, due to some independent calculation. Without knowing the ultimate storage location, the processor cannot determine whether the store and load instructions will access the same memory location and, thus, cannot determine whether a data dependence exists between those store and load instructions.

Even where an ambiguous dependency exists, the processor may nonetheless choose to execute the potentially dependent load instruction before the store instruction has finished executing. This is called "speculation" because the processor is effectively speculating that no

data dependence exists between those store and load instructions. A "mis-speculation" occurs if a data dependence does exist between the two instructions, and the processor executes the dependent load instruction before the store instruction. If a processor correctly speculates—in other words, if the processor correctly guesses that a load instruction is *not* dependent on an earlier store instruction that has not yet executed—processor performance may be improved because the processor did not needlessly delay execution of that load instruction. J.A. 1431–32 (testimony of Dr. Conte). But, if a processor mis-speculates, the processor essentially has to discard work it has already performed and re-do the work in the correct order. J.A. 1433 (testimony of Dr. Conte), 1717–19 (testimony of Dr. Colwell). This recovery process is called "squashing" or "flushing." J.A. 1433 (testimony of Dr. Conte). As might be expected, mis-speculations do not help processor performance, and may in fact harm performance. In short, while out-of-order execution of instructions with ambiguous dependencies may improve performance in cases where the processor speculates correctly, performance may be decreased by mis-speculation.

One method to minimize mis-speculation is for the processor to make an informed decision as to whether it should speculate. This is called "prediction." This case concerns a particular prediction method used to increase the accuracy of processor speculation such that mis-speculations are minimized.

A

The '752 patent, which expired on December 26, 2016, describes a specific prediction technique for an out-of-order processor. In this case, WARF asserted independent claims 1 and 9, as well as dependent claims 2, 3, 5, and 6. Claim 1 reads:

> 1. In a processor capable of executing program instructions in an execution order differing from their program order, the processor further having a data speculation circuit for detecting data dependence between instructions and detecting a mis-speculation where a [load] instruction dependent for its data on a [store] instruction of earlier program order, is in fact executed before the [store] instruction, a data speculation decision circuit comprising:
>
> a) a predictor receiving a mis-speculation indication from the data speculation circuit to produce a prediction associated with the particular [load] instruction and based on the mis-speculation indication; and
>
> b) a prediction threshold detector preventing data speculation for instructions having a prediction within a predetermined range.

'752 patent claim 1.[1] Claim 9 reads:

> 9. In a processor capable of executing program instructions in an execution order differing from the program order of the instructions, the processor further having a data speculation circuit for detecting data dependence between instructions and detecting a mis-speculation where a [load] instruction dependent for its data on a [store] instruction of earlier program order, is in fact executed before the [store] instruction, a data speculation decision circuit comprising:

---

[1] The modifications to the claim language reflect the parties' substitutions, for clarity, of the term "load" for "data consuming," and the term "store" for "data producing." *See* Appellant's Br. 12 n.1; Appellee's Br. 13. We adopt this helpful substitution in this opinion.

> a) a prediction table communicating with the data speculation circuit to create an entry listing a particular [load] instruction and [store] instruction each associated with a prediction when a mis-speculation indication is received; and
>
> b) an instruction synchronization circuit only instructing a processor to delay a later execution of the particular [load] instruction if the prediction table includes an entry.

'752 patent claim 9.[2]

According to the claim language, when the data speculation circuit detects a mis-speculation, it sends a mis-speculation indication to a predictor. *Id.* at claim 1. The predictor then produces a prediction, based on the mis-speculation indication, as to whether a data dependence likely exists between the corresponding load and store instructions. A higher prediction value indicates a greater likelihood of data dependence and, therefore, a greater likelihood that a mis-speculation will occur if those instructions are executed out-of-order. *Id.* at col. 11 ll. 29–32. The prediction may be "updated based on historical mis-speculations detected by the data speculation circuit." *Id.* at col. 8 ll. 8–9. Going forward, the predictor "provides a dynamic indication to the data speculation circuit . . . as to whether data speculation should be performed." *Id.* at col. 8 ll. 1–3. And, if the prediction for a given load instruction exceeds a certain "predetermined range," speculation is prevented. *Id.* at claim 1.

---

[2] The parties have not distinguished these claims for purposes of the infringement issues on appeal.

B

The products accused of infringement in this case are Apple's A7, A8, and A8X integrated circuit chips, which include one or more processors. These processors include a Load-Store Dependency Predictor ("LSD predictor"), which is the technology at issue in this case.

The LSD predictor detects data dependences between load and store instructions and uses a prediction table to make predictions based on those dependences. Each entry in the prediction table includes (among other things) a load tag, a store tag, and a prediction (or "counter"). The load tag is generated by taking certain information about a load instruction, such as its address, and creating a 12-bit load tag using a hashing function. Because the load tags are limited to 12 bits, only 4,096 load tags are available. The hashing algorithm uses a one-way hash, meaning that a given load tag cannot be expanded back to the load instruction that generated that load tag. Moreover, based on this hashing algorithm, it is possible for multiple load instructions to hash to the same load tag.

When multiple load instructions hash to the same load tag, it is possible for multiple instructions to update the same prediction in the LSD predictor's prediction table. This is called "aliasing." *See* J.A. 2237 (testimony of Apple's expert, Dr. August), 2294 (same); *see also* Appellant's Br. 23–24; Appellee's Br. 14. This means that a given instruction's history may impact the behavior of all load instructions that share the same load tag. J.A. 2166 ll. 15–18, 2168 ll. 7–11 (testimony of Dr. August).

C

WARF filed this patent infringement suit against Apple in January 2014.[3] Apple answered and asserted counterclaims for declaratory judgment of non-infringement and invalidity of the '752 patent.

Before trial, both Apple and WARF moved for summary judgment with respect to Apple's counterclaims and defenses of anticipation under 35 U.S.C. § 102, based on U.S. Patent No. 5,619,662 ("Steely"). Specifically, Apple asserted that claims 1–3, 5, 6, and 9 of the '752 patent were invalid as anticipated by Steely. The district court granted summary judgment of no anticipation in favor of WARF.

Once the case proceeded to trial, the district court bifurcated the trial into two phases: liability and damages. After the liability phase, the jury found the asserted claims infringed and not invalid.[4] After the damages phase of trial, the jury awarded WARF over $234 million in damages.

After trial, Apple moved for judgment as a matter of law ("JMOL") and, in the alternative, for a new trial. The district court denied Apple's post-trial motions in their entirety. Apple timely appealed. This court has jurisdiction under 28 U.S.C. § 1295(a)(1).

---

[3] WARF has since filed a second infringement suit against Apple with respect to additional products released by Apple that WARF also believes infringe the '752 patent. *See* Compl., *Wis. Alumni Research Found. v. Apple Inc.*, No. 3:15-cv-00621-WMC (W.D. Wis. Sept. 25, 2015), ECF No. 1. That case is currently stayed pending the outcome of this appeal. *See* J.A. 20346.

[4] We note that the invalidity issues presented to the jury are not before this court on appeal.

II

"We review a district court's denial of JMOL or a new trial under the law of the regional circuit." *LifeNet Health v. LifeCell Corp.*, 837 F.3d 1316, 1322 (Fed. Cir. 2016). The Seventh Circuit reviews a district court's denial of a motion for judgment as a matter of law de novo. *Clarett v. Roberts*, 657 F.3d 664, 674 (7th Cir. 2011). In doing so, the appellate court "review[s] the record as a whole to 'determine whether the evidence presented, combined with all reasonable inferences permissibly drawn therefrom, is sufficient to support the verdict when viewed in the light most favorable to the party against whom the motion is directed.'" *Id.* (quoting *Erickson v. Wis. Dep't of Corr.*, 469 F.3d 600, 601 (7th Cir. 2006)). A jury verdict will be overturned only if no reasonable juror could have found in the non-movant's favor. *Id.*

A

Apple contends that no reasonable juror could have found that Apple's processors literally infringe the asserted claims of the '752 patent.[5] Specifically, Apple argues that its processors satisfy neither the "particular" nor the "mis-speculation" limitations recited in each of the claims.

With respect to the "particular" limitation, independent claim 1 requires a predictor that "produce[s] a prediction associated with the particular [load] instruction." '752 patent claim 1. Likewise, independent claim 9 requires a prediction table that "create[s] an entry listing a particular [load] instruction and [store] instruction each associated with a prediction." *Id.* at claim 9.

---

[5] WARF abandoned its theory of infringement under the doctrine of equivalents before trial, and has proceeded only on a theory of literal infringement.

Although neither party asked the district court to construe "particular" before trial, WARF moved during trial to preclude Apple's expert, Dr. August, from testifying that a prediction cannot be associated with a "particular" load instruction if each load tag represents multiple load instructions. J.A. 18646–62. Specifically, WARF argued that Apple's expert should have been forbidden from making any suggestion that each prediction must be associated with one and only one load instruction.

Apple responded by arguing that the term "particular" should carry its plain and ordinary meaning, and that its expert's theory of non-infringement was consistent with that meaning. J.A. 18728 ("Apple has always maintained that the phrase—which uses only an ordinary word—does not require construction; the plain and ordinary meaning should apply."); J.A. 18730 ("Apple believes that the word 'particular' does not require any construction, because the '752 patent uses the word in its ordinary sense."); J.A. 18728 ("With respect to the 'particular' limitation, Dr. August has consistently applied the plain and ordinary meaning of the claim language."). Apple explained that the plain and ordinary meaning of "particular" meant that the claimed "prediction" must be associated with a *single* load instruction (i.e., one and only one load instruction), rather than with a *group* of load instructions. *See* J.A. 18729–33; *see also* J.A. 144 (Dist. Ct. Op. (summarizing Apple's argument)).

The district court denied WARF's motion to exclude the testimony of Apple's expert. J.A. 142–46. In doing so, the district court agreed with Apple that the term "particular" should be given its plain and ordinary meaning and thus ruled that no jury instruction was necessary to define that term. J.A. 145. Consistent with Apple's understanding of the plain and ordinary meaning, the district court reasoned that "[f]rom the court's reading of claim 1 as a whole, it contemplates a *single* load instruction." J.A. 144 (emphasis added). In the district court's

view, this was "consistent with the plain meaning of the claim terms 'the' and 'the particular.'" J.A. 145. The court thus "conclude[d] that claim 1 discloses a prediction associated with a *single* load instruction." J.A. 145 (emphasis added).[6]

On appeal, WARF does not dispute the district court's decision to give the term "particular" its plain and ordinary meaning. *See* Appellee's Br. 11, 26. Instead, WARF appears to disagree with the district court's understanding of the plain meaning. Giving a term its plain and ordinary meaning does not leave the term devoid of any meaning whatsoever. Instead, "the 'ordinary meaning' of a claim term is its meaning to the ordinary artisan after reading the entire patent." *Phillips v. AWH Corp.*, 415 F.3d 1303, 1321 (Fed. Cir. 2005) (en banc). In our view, the plain meaning of "particular," as understood by a person of ordinary skill in the art after reading the '752 patent, requires the prediction to be associated with a *single* load instruction. A prediction that is associated with more than one load instruction does not meet this limitation.[7]

---

[6] We note that the district court, in its order denying Apple's JMOL, stated that "a reasonable jury could conclude that a prediction was associated with a particular load instruction even if that same prediction may be associated with other load instructions." J.A. 205 (alteration omitted).

[7] WARF contends that this view of the plain meaning reads out the preferred embodiment of the '752 patent, which purportedly uses partial instruction addresses to identify load instructions, similar to the load tags used in Apple's products. *See* Appellee's Br. 28–29. We are unpersuaded. Figure 1 of the patent shows a program stored in memory "at a plurality of physical addresses 19 here depicted as xx1–xx6 where the values xx indicate

Applying the plain and ordinary meaning of the term "particular," and drawing all reasonable inferences from the evidence in favor of WARF, we hold that no reasonable juror could have found literal infringement in this case. As explained above, each entry in Apple's LSD prediction table includes, among other things, a load tag and a prediction. Each load tag is generated by hashing information about a load instruction, such as its address, down to a 12-bit load tag. Only 4,096 load tags are possible. And because of the way Apple's hashing algorithm is designed, multiple load instructions may hash to the same load tag. Each load tag can therefore be associated with a *group* of load instructions—namely, all of the load instructions that hash to the same load tag. The practical effect of this is that a given load instruction's history will impact the prediction associated with *all* load instructions that hash to that same load tag.

WARF first contends that the "prediction" corresponding to a load tag will necessarily remain associated with a "particular" load instruction that mis-speculates because that load instruction will always hash to the same 12-bit load tag. Appellee's Br. 13. But, even accepting that a load instruction will always generate the same 12-bit load tag, *see* J.A. 2518 ll. 19–24, this is insufficient to satisfy

---

some higher ordered address bits that may be ignored in this example." '752 patent col. 6 ll. 62–67; *see also id.* at col. 5 ll. 44–49 (discussing Fig. 2). Although Figures 5–8, which show the prediction table, then refer to the instructions as "LD 8" and "ST 10," there is no indication in the specification that instruction addresses are hashed or truncated before being added to the prediction table. Instead, the specification explains that the prediction table of Figure 5 is reviewed to determine if a "particular" instruction "identified by its physical address" is in the prediction table. *Id.* at col. 11 ll. 3–7.

this claim limitation because this argument ignores the plain and ordinary meaning of the term "particular," as described above. Under that meaning, it is not enough that an instruction hash to the same tag every time; the dispositive issue is whether *other* instructions *also* hash to that tag, such that the prediction is associated with a group of instructions, rather than a *particular* instruction.[8]

WARF's second argument for upholding the jury verdict appears to be that, even if the prediction must be associated with a single load instruction, the products still infringe in at least *some* circumstances—i.e., those in which aliasing does not occur. Appellee's Br. 15–18. Certainly, a product that "sometimes, but not always, embodies a claim nonetheless infringes." *Broadcom Corp. v. Emulex Corp.*, 732 F.3d 1325, 1333 (Fed. Cir. 2013) (alteration omitted) (quoting *Bell Commc'ns Research, Inc. v. Vitalink Commc'ns Corp.*, 55 F.3d 615, 622–23 (Fed. Cir. 1995)). But after reviewing the evidence and drawing all reasonable inferences in favor of WARF, we find that there is insufficient evidence to support WARF's theory that Apple's load tags are *sometimes* associated with a single load instruction.

---

[8] On this point, WARF also argues that even if multiple instructions hash to the same load tag, "the 'prediction' merely becomes associated with *two* loads, including 'the particular [load]' that mis-speculated." Appellee's Br. 16. WARF then contends that infringement still exists because the preamble of the claims uses the word "comprising," which allows for additional, unrecited elements. *Id.* at 15–16. But "'[c]omprising' is not a weasel word with which to abrogate claim limitations," *Spectrum Int'l, Inc. v. Sterilite Corp.*, 164 F.3d 1372, 1380 (Fed. Cir. 1998), and WARF's application of that term here would frustrate the plain meaning of "particular" as used in this patent.

The evidence WARF points us to in support of this theory is sparse. WARF contends that the frequency of "aliasing" in Apple's products is low (specifically, 0.1%), which WARF takes to mean that load tags represent a single load instruction at least sometimes (in fact, 99.9% of the time). This conclusion, however, does not follow from the evidence cited by WARF.

First, the inference WARF seeks to draw from the evidence cited is not reasonable. The 0.1% statistic comes from an email from Stephan Meier, an Apple engineer, that pertains to testing various hashing functions. J.A. 1499 (testimony of Dr. Conte). During trial, the parties disputed the meaning of the 0.1% statistic, with WARF arguing that it represents the *frequency* of aliasing, and Apple arguing that it represents the *performance impact* of aliasing, *see, e.g.*, J.A. 2238 ll. 10–18 (cross-examination testimony of Dr. August). Although there are a few isolated statements from Apple's fact and expert witnesses that WARF argues support its theory, *see* J.A. 2238–40, the most thorough explanation of this piece of evidence comes from WARF's expert, and his testimony undermines the inference WARF seeks to draw. According to WARF's expert, Mr. Meier was trying to determine the "performance impact" of using different hashing functions to determine which hashing function performs best, including an analysis of how many bits should be used for the load tag. J.A. 1499–500 (testimony of Dr. Conte). As explained by WARF's expert, Mr. Meier concluded that a 9-bit load tag would cause a loss of "0.9 percent in performance"; a 10-bit load tag would cause a 0.4% loss in performance; and a 12-bit load tag would cause less than a 0.1% loss in performance. J.A. 1500 (testimony of Dr. Conte). Despite this explanation, which indicates that Mr. Meier's statistics were indeed describing overall performance, WARF's expert jumped to the conclusion that "aliasing is very rare." J.A. 1501 (testimony of Dr. Conte). But, in light of Dr. Conte's testimony,

it is unreasonable to infer that the 0.1% statistic was referring to the *frequency* of aliasing.

Second, even accepting WARF's unreasonable view of this evidence (that the *frequency* of aliasing is 0.1%), this does not support an inference that load tags sometimes represent a single load instruction. "Aliasing" does not simply refer to two load instructions hashing to the same load tag. Instead, "aliasing" occurs when two load instructions actually update the same prediction in operation because they share the same load tag. *See* J.A. 2294 (testimony of Dr. August) ("Q: First, what's the difference between the load tags' grouping of load instructions and the concept of aliasing? A: So the grouping is always present. Aliasing is when the program is running, what is the performance impact of that grouping."); *see also* Appellant's Br. 23–24; Appellee's Br. 14. It is therefore not reasonable to infer that load instructions rarely hash to the same load tag, merely because the frequency of load instructions actually updating the same prediction during operation is low.

Finally, WARF points to Apple's technical documentation, arguing that certain language in the documentation demonstrates that Apple's LSD predictor "uniquely" identifies load instructions. Appellee's Br. 14 (citing J.A. 10131); *see also* J.A. 1489 l. 8–1490 l. 8 (testimony of Dr. Conte, discussing J.A. 10131). Apple points out, however, that the documentation merely states that the LSD predictor "can be thought of" as uniquely identifying load instructions, Reply Br. 2–3 (quoting J.A. 10131), and that "in practice" the load tags are the result of applying the hashing algorithm. *Id.*; *see also* J.A. 2178 ll. 3–22 (testimony of Dr. August). Reading the quote in context, it is not reasonable to infer that the load tags, in practice, uniquely identify load instructions. And even if this inference were reasonable, it would not be enough to support a finding that Apple's processors actually practice the "particular" limitation.

In short, there is not substantial evidence to support WARF's theory that, in Apple's LSD predictor, a prediction (by way of a load tag) is at least sometimes associated with a single load instruction. And, given that only 4,096 load tags are possible, and that Apple's operating system alone contains millions of load instructions, the only reasonable inference to draw is that load tags will always represent multiple load instructions. *See* J.A. 1605–06 (testimony of Dr. Conte), 2296–97 (testimony of Dr. August).[9]

In sum, drawing all reasonable inferences in favor of WARF, there is insufficient evidence to support the jury's finding that Apple's products literally satisfy the "particular" limitation. As this conclusion is sufficient to set aside the jury's infringement finding, we need not address Apple's arguments regarding the "mis-speculation" limitation.

B

Apple also contends that the district court erred in granting summary judgment of no anticipation based on the Steely prior art reference. The district court determined that Steely did not disclose the "prediction" claimed in the '752 patent. *Wis. Alumni Research Found. v. Apple, Inc.*, No. 14-cv-062-WMC, 2015 WL 4668247, at *13–16 (W.D. Wis. Aug. 6, 2015) ("*MSJ Order*"). In Apple's view, this determination was based on an incorrect construction of the term "prediction." Apple also contends that, even under the court's construction, a genuine dispute of material fact exists, making summary judgment improper.

---

[9] Although WARF's brief states that programs can have fewer than 4,096 load instructions, WARF has not pointed us to any evidence to support this assertion. *See* Appellee's Br. 17.

1

Claim construction is ultimately a legal question reviewed de novo, with any subsidiary fact-findings regarding extrinsic evidence reviewed for clear error. *Teva Pharm. USA, Inc. v. Sandoz, Inc.*, 135 S. Ct. 831, 841 (2015).

The parties dispute the construction of the term "prediction." WARF contends that a "prediction" must be dynamic, meaning it is capable of receiving updates. Apple contends that while a "prediction" includes dynamic predictions, the term is also broad enough to include static predictions (i.e., those incapable of receiving updates). The district court agreed with WARF, concluding that a prediction, as used in the patent, must be "capable of receiving updates." *MSJ Order* at *13.

On appeal, Apple argues that the plain and ordinary meaning of "prediction" encompasses both dynamic and static predictions. Apple further contends that the patent's specification does not limit a "prediction" to being dynamic, and that by requiring that the prediction be capable of receiving updates, the district court improperly imported a limitation from the preferred embodiment. *See* Appellant's Br. 38–39. Apple's arguments are unpersuasive, as explained below.

First, "the 'ordinary meaning' of a claim term is its meaning to the ordinary artisan after reading the entire patent." *Phillips*, 415 F.3d at 1321. Reading the patent as a whole, it is clear that the claimed prediction must be capable of receiving updates. The term "prediction" is used throughout the specification to describe a prediction value that updates based on a given load instruction's historical mis-speculation behavior. *See* '752 patent col. 11 ll. 33–35 ("Normally the prediction 109 starts at zero when an entry is first made in the prediction table 44 and is incremented and decremented as will be described below."); *see also id.* at col. 8 ll. 7–11 ("The prediction

provided by the predictor circuit 33, as will be described, is updated based on historical mis-speculations detected by the data speculation circuit 30. For this reason, the data speculation circuit 30 must communicate with the predictor circuit 33 on an ongoing basis."). Specifically, the prediction is updated as new information is gathered regarding the likelihood of future mis-speculation. *Id.* at col. 12 l. 61–col. 13 l. 3 ("[T]he predictor circuit 33 must also make adjustments in its prediction table 44 if there is a mis-speculation, ... [T]he prediction table 44 is checked to see whether the LOAD/STORE pair causing the mis-speculation is in the prediction table 44 already. If so then at process block 302, the prediction 109 is updated toward synchronize so that this mis-speculation may be avoided in the future."); *id.* at col. 12 ll. 14–17 ("In this case, the prediction that there was a need to synchronize was wrong and so at process block 120 the prediction 109 is decremented toward the do not synchronize state."); *id.* at col. 12 ll. 50–55 ("In this case, the prediction 109 is updated toward the synchronize condition indicating that the prediction that there was a need to synchronize was correct as there is in fact a LOAD instruction waiting to be synchronized."). Where, as here, "a patent 'repeatedly and consistently' characterizes a claim term in a particular way, it is proper to construe the claim term in accordance with that characterization." *GPNE Corp. v. Apple Inc.*, 830 F.3d 1365, 1370 (Fed. Cir. 2016) (quoting *VirnetX, Inc. v. Cisco Sys., Inc.*, 767 F.3d 1308, 1318 (Fed. Cir. 2014)).

Second, Apple has not pointed us to any portion of the specification that describes a static prediction. Although Apple directs our attention to "alternative embodiments" for obtaining the prediction—methods other than incrementing, such as "various weighting schemes" or "complex pattern matching techniques"—none of the passages concerning these embodiments describe a static prediction. *See* Appellant's Br. 39 (citing '752 patent col. 14

ll. 6–14); Reply Br. 13 (citing same). Instead, the embodiments merely illustrate methods other than "simply incrementing it in value for each speculation" for calculating the value of the prediction. '752 patent col. 14 ll. 8–9. In short, by allowing the claimed "prediction" to also include static predictions, Apple's proposed construction would "expand the scope of the claims far beyond anything described in the specification." *Kinetic Concepts, Inc. v. Blue Sky Med. Grp., Inc.*, 554 F.3d 1010, 1019 (Fed. Cir. 2009); *see id.* (limiting the term "wound" to "skin wound," rather than allowing it to encompass "pus pockets," where all of the examples in the specification involved skin wounds).

In sum, rather than improperly reading a limitation from the preferred embodiment into the claims, the district court's construction, with which we agree, properly reads the claim term in the context of the entire patent.

2

Apple next contends that the district court erred in granting summary judgment of no anticipation, even under the district court's construction of "prediction," which requires that the prediction be "capable of receiving updates." Appellant's Br. 40–42. Specifically, it contends that a genuine factual dispute exists as to whether Steely discloses predictions capable of receiving updates.

Applying Seventh Circuit law, "[w]e review the grant of summary judgment *de novo*, construing all facts and drawing all inferences 'in the light most favorable to the non-moving party.'" *Austin v. Walgreen Co.*, 885 F.3d 1085, 1087 (7th Cir. 2018) (quoting *Zuppardi v. Wal-Mart Stores, Inc.*, 770 F.3d 644, 649 (7th Cir. 2014)).

Steely discloses out-of-order processors that use past mis-speculations (or "collisions") to predict whether load instructions should be allowed to execute out-of-order. *See* Steely col. 2 ll. 63–66. Steely uses "tags" to indicate

whether instructions that were "previously reordered and executed had a collision" and to "ascertain whether the . . . instructions can be reordered." *See id.* at col. 2 l. 64–col. 3 l. 1. Specifically, Steely assigns "tags" to load and store instructions that mis-speculate (or "collide"). Although Steely discloses multiple techniques for generating tags, the parties focus on the first technique disclosed. According to this technique, "when a pair of load and store instructions cause a problem the first time"—i.e., when they are executed out-of-order and a mis-speculation occurs—"a portion of the address in memory which resulted in a load-store collision are [sic] saved." *Id.* at col. 48 ll. 1–4. The example in Steely uses five bits of the memory address as the tag. *Id.* at col. 48 ll. 26–28. That tag is associated with the load and store instructions that mis-speculated when reordered. *Id.* at col. 48 ll. 26–29, 37–50; *see also* J.A. 15069 ¶ 162 (invalidity report of Dr. Colwell). The next time that same pair of load and store instructions is called, the instructions' tags are compared. If the tags match, those instructions will not be executed out-of-order. Steely col. 48 ll. 33–36; J.A. 15069 ¶¶ 162–63, 15136–37 ¶¶ 301–02 (invalidity report of Dr. Colwell).

Apple contends that the outcome of this comparison is a "prediction," as it indicates the likelihood of mis-speculation if those instructions are executed out-of-order. We agree that this is a reasonable inference to draw at the summary judgment stage. The only question, then, is whether the outcome of that comparison is also "capable of receiving updates," as is required under the proper construction of the term "prediction."

Apple's expert, Dr. Colwell, provided an example in his invalidity report explaining how the outcome of this comparison can change. He reasoned that the outcome may change because, "[a]s more mis-speculations occur, one or both of the tags of the same pair of load and store instructions may change to different values, resulting in different outcomes from a comparison of the tags."

J.A. 15137–38 ¶ 303. In other words, if the store instruction from the first load-store pair is reordered with a *different* load instruction, and a mis-speculation occurs, both of those instructions would receive a tag based on the memory address that the instructions were accessing. According to Dr. Colwell, this necessarily causes the store instruction's tag to change. And because the store instruction's tag has changed, that tag will no longer match the tag of the original load instruction. Thus, Dr. Colwell concludes that the outcome of the comparison of the original load-store pair will be different. *Id.* Based on this example, Apple contends that Steely discloses a variable "prediction" that is "capable of receiving updates."

WARF responds that Dr. Colwell's example is mere speculation regarding how Steely *might* be implemented, and that Steely itself never discloses that tags, once generated, can change. Appellee's Br. 45.

We agree with WARF that no reasonable juror could find that Steely's specification discloses the behavior described in Dr. Colwell's example regarding changing tags. Apple points us to just two statements from the specification as support for this disclosure: (1) a statement regarding the size of the tag storage table, *see* Steely col. 47 ll. 56–60; and (2) a statement that tags "will be stored" in that table, *see id.* at col. 48 ll. 29–30. Based on the size of the table, Apple contends that a fact-finder could infer that each instruction can be associated with only a *single* tag (as opposed to, for example, allowing an instruction to carry multiple tags). So, the argument goes, because tags "will be stored," when a given instruction receives a new tag, the new tag is stored in the table, and the former tag is necessarily changed. But the inference Dr. Colwell would have a fact-finder draw from the size of the table is not reasonable. And although Apple cites additional evidence in attempt to bolster this theory (namely, uncorroborated inventor testimony and another

reference stating generally that Steely uses a "prediction"), such evidence is insufficient to create a genuine dispute of material fact.

We therefore agree with the district court that no reasonable juror could find that Steely discloses the "prediction" limitation of the '752 patent's claims. We therefore affirm the district court's grant of summary judgment on this issue.

III

For the reasons stated above, we reverse the district court's denial of Apple's JMOL motion with respect to non-infringement, but affirm its grant of summary judgment with respect to Apple's anticipation defense based on Steely.

AFFIRMED-IN-PART AND REVERSED-IN-PART

COSTS

The parties shall bear their own costs.