NOTE: Pursuant to Fed. Cir. R. 47.6, this disposition is not citable as precedent. It is a public record.

United States Court of Appeals for the Federal Circuit

03-1153

INTERGRAPH CORPORATION,

Plaintiff-Appellee,

v.

INTEL CORPORATION,

Defendant-Appellant.

DECIDED: February 11, 2004

Before MAYER, Chief Judge, LOURIE and CLEVENGER, Circuit Judges.

LOURIE, Circuit Judge.

DECISION

Intel Corporation appeals from the decision of the United States District Court for the Eastern District of Texas finding that Intel infringed Intergraph Corporation's United States Patents 5,560,028 and 5,794,003. Intergraph Corp. v. Intel Corp., No. 2:01CV160 (E.D. Tex. Oct. 10, 2002) ("Findings and Conclusions"). Because we conclude that the district court erred in construing the claim term "pipeline identifier," we vacate the judgment of infringement and remand for the district court to determine in the first instance whether the accused devices infringe the '028 and '003 patents under our revised claim construction.

BACKGROUND

The technology in this case relates to microprocessors that execute multiple computer instructions simultaneously, or in parallel. In order to execute instructions in parallel, a microprocessor must, among other things, determine which instructions can be issued in the same clock cycle and route those instructions to the pipelines that will process them. The patents in suit describe two conventional approaches to parallel processing: The first approach, known as a "superscalar" architecture, employs hardware to determine which instructions can be issued in parallel and to schedule those instructions for processing by the available pipelines. See '028 patent, col. 1, ll. 35-60. The second approach, known as a "wide-word" or "very long instruction word" ("VLIW") architecture, uses software, such as a compiler, to group independent instructions together and then positions each individual instruction within a group so that it aligns with the pipeline that will process it. See id., col. 1, l. 61 to col. 2, l. 18.

Intergraph owns the '028 and '003 patents, which are directed to a computer architecture for processing groups of computer instructions in parallel through a combination of superscalar and VLIW techniques. The '028 patent, entitled "Software Scheduled Superscalar Computer Architecture," claims a system and methods for grouping together instructions that are executable in parallel and then using "pipeline identifiers" to route individual instructions to appropriate processing pipelines. Claim 20 reads as follows:

In a computing system having a plurality of processing pipelines in which groups of individual instructions are executable, each individual instruction in a group executable in parallel by the plurality of processing pipelines, a method for transferring each individual instruction in a group to be executed through a crossbar switch having a first set of connectors coupled to a very long instruction word storage for receiving individual instructions therefrom, a second set of connectors coupled to the plurality of processing pipelines, and switches between the first set and the second set. Of [sic] connectors, the method comprising:

retrieving the very long instruction word from a main memory;

storing in the very long instruction word storage, the very long instruction word, the very long instruction word having a set of individual instructions including at least one group of individual instructions to be executed in parallel, each individual instruction in the at least one group having embedded therein a unique pipeline identifier indicative of the processing pipeline which will execute that individual instruction, the very long instruction word storage also including at least one other individual instruction not in the at least one group of individual instructions, the at least one other individual instruction having embedded therein a different pipeline identifier; and

using the unique pipeline identifiers of the individual instructions in the at least one group of individual instructions to control the switches between the first set of connectors and the second set of connectors to thereby supply each individual instruction in the at least one group to be executed in parallel to an appropriate processing pipeline.

Id., col. 16, ll. 28-61 (emphases added). Claim 21 depends from claim 20.

The ’003 patent, entitled “Instruction Cache Associative Crossbar Switch System,” is directed more particularly to a system and methods for routing individual instructions to appropriate processing pipelines through an “associative crossbar switch.” Claim 1 recites a computing system comprising:

means for forming groups of software-scheduled instructions, software-scheduled instructions within each of the groups executable in parallel; and

a super-scaler cache for routing each of the software-scheduled instructions within the groups to be executed in parallel to an appropriate instruction pipeline of a plurality of instruction pipelines, the super-scaler cache comprising:

super-scaler storage for holding one group of the groups of software-scheduled instructions, each software-scheduled instruction within the one group having embedded therein an instruction pipeline identifier of a plurality of instruction pipeline identifiers;

an associative crossbar having a first set of connectors coupled to the super-scaler storage for receiving each of the software-scheduled instructions therefrom, and a second set of connectors coupled to the plurality of instruction pipelines; and

means responsive to the instruction pipeline identifier of each of the software-scheduled instructions, for coupling appropriate connectors of the first set of connectors to appropriate connectors of the second set of connectors, to thereby supply each of the software-scheduled instructions to the appropriate instruction pipeline for parallel execution.

’003 patent, col. 7, ll. 28-55 (emphases added).

Relatedly, claim 6 recites “a method for transferring software-scheduled instructions to be executed through an associative crossbar switch in a super-scaler cache,” wherein the method comprises the following steps:

forming groups of software-scheduled instructions, software-scheduled instructions within each group being executable in parallel;

storing in the super-scaler storage in the super-scaler cache one group of the groups of software-scheduled instructions to be executed in parallel, each software-scheduled instruction in the one group having embedded therein an instruction pipeline identifier of a plurality of instruction pipeline identifiers; and

using the instruction pipeline identifier of each of the software-scheduled instructions to control switches in the associative crossbar switch in the super-scaler cache between the first set of connectors and the second set of connectors to thereby supply each of the software-scheduled instructions to an appropriate instruction pipeline.

Id., col. 9, ll. 1-24 (emphases added). Claim 7 depends from claim 6. Claims 11, 14, and 15 depend from claims 1, 6, and 7, respectively, and further require that a compiler form the groups of software-scheduled instructions or determine the pipeline identifiers.

Finally, claim 22 recites a super-scaler cache for similarly routing software-scheduled instructions to appropriate instruction pipelines. The super-scaler cache comprises super-scaler storage, an associative crossbar, selection means for supplying output signals in response to pipeline identifiers, and switching means for receiving those output signals and then connecting connectors to thereby supply each software-scheduled instruction in a set of software-scheduled instructions to the appropriate instruction pipeline. Id., col. 11, ll. 32-64.

Intel manufactures the Itanium (“Merced”) and Itanium 2 (“McKinley”) microprocessors. In those devices, instructions are organized into “bundles.” Each bundle comprises three “syllables” and a “template,” which indicates the type of each instruction. A “steering crossbar” in the Itanium device and “dispersal multiplexers” in the Itanium 2 device ultimately route data from the bundles to “execution units” where they can be processed in parallel.

In July 2001, Intergraph brought suit against Intel in the United States District Court for the Eastern District of Texas, alleging that Intel’s Itanium and Itanium 2 microprocessors infringed the ’028 and ’003 patents. Intel filed a counterclaim seeking a declaratory judgment that both patents are invalid. [1] In June 2002, the court issued a Markman order construing a multitude of disputed claim terms. Intergraph Corp. v. Intel Corp., No. 2:01-CV-160 (TJW) (E.D. Tex. June 3, 2002) (“Markman Order”). Of relevance to this appeal are the four following claim interpretations: First, the court construed the term “pipeline identifier” in both the ’028 patent and the ’003 patent to mean “a designation indicative of a processing pipeline,” further explaining that a “pipeline identifier” need not identify the explicit processing pipeline that will execute an instruction. Id., slip op. at 6, 18. Second, the court interpreted the term “groups of individual instructions” in the ’028 patent to mean “a collection of one or more individual instructions that can be executed simultaneously (i.e., can be dispatched to parallel pipelines simultaneously)." Id. at 4. Similarly, it interpreted the term "groups [or sets] of software-scheduled instructions" in the '003 patent to mean "sets of one or more instructions that have been identified by software as being capable of being processed in parallel." Id. at 15. Finally, the court construed the term "associative crossbar" or "associative crossbar switch" to mean "a crossbar in which the data being routed includes the routing instructions." Id. at 10, 17.

Following a bench trial, the district court found that Intel literally infringed claims 20 and 21 of the '028 patent and claims 1, 6, 7, 11, 14, 15, and 22 of the '003 patent.[2] In its findings of fact and conclusions of law, the court began by construing the term "instruction" in both patents to mean "the smallest unit of work capable of being processed at any particular stage in the computer." Findings and Conclusions, slip op. at 3-4, 17-18. The court then engaged in a claim-by-claim and limitation-by-limitation infringement analysis. For both patents, the court found that the templates in the accused devices contain "pipeline identifiers" that control the "crossbars" (i.e., the steering crossbar in the Itanium microprocessor and the dispersal multiplexers in the Itanium 2 microprocessor) to supply individual "instructions" to the appropriate processing pipelines. Id. at 7-8, 10-12, 15. For the '003 patent, the court rejected Intel's argument that the accused devices do not infringe because they do not have an "associative crossbar" located physically within the cache, concluding instead that the claims do not contain such a requirement. Id. at 9. The court further found that the "crossbars" in the accused devices, with their associated dispersal logic, are "associative" because the instructions that are routed through them include the templates, which in turn include the routing instructions. Id. at 9-10. Accordingly, the court entered a final judgment of infringement and permanently enjoined Intel from making, using, or selling the Itanium and Itanium 2 microprocessors. Intergraph Corp. v. Intel Corp., No. 2-01-CV-160-TJW, slip op. at 1 (E.D. Tex. Oct. 30, 2002) ("Final Judgment and Permanent Injunction").

Intel timely appealed to this court. We have jurisdiction pursuant to 28 U.S.C. § 1295(a)(1).

DISCUSSION

A determination of infringement requires a two-step analysis. "First, the court determines the scope and meaning of the patent claims asserted and then the properly construed claims are compared to the allegedly infringing device." Cybor Corp. v. FAS Techs., Inc., 138 F.3d 1448, 1454 (Fed. Cir. 1998) (en banc) (citations omitted). Step one, claim construction, is an issue of law that we review de novo. Id. at 1456. Step two, comparison of the claims to the accused device, requires a determination that every claim limitation or its equivalent is found in the accused device. Warner-Jenkinson Co. v. Hilton Davis Chem. Co., 520 U.S. 17, 29 (1997). Those determinations are questions of fact that we review for clear error when tried without a jury. Ultra-Tex Surfaces, Inc. v. Hill Bros. Chem. Co., 204 F.3d 1360, 1363 (Fed. Cir. 2000).

On appeal, Intel challenges the district court's claim constructions and findings of infringement with respect to five claim terms. We discuss each of the contested claim terms below.[3]

A. "Pipeline Identifier"

Intel first takes issue with the district court's construction of the claim term "pipeline identifier" found in both the '028 patent and the '003 patent. Pointing to the claim language, the specification, and the prosecution history, Intel argues that a "pipeline identifier" must identify the specific pipeline to which an instruction will be routed. Intel asserts that, under its proposed construction, the accused devices do not have "pipeline identifiers" because their templates do not identify specific pipelines. Even under the district court's claim construction, however, Intel challenges as clearly erroneous the court's finding that the templates in the accused devices are "pipeline identifiers," arguing that the templates are not "unique" identifiers embedded within each syllable.

Intergraph responds that the district court correctly interpreted the term "pipeline identifier" to require only identification of the type of pipeline through which an instruction will be processed. Intergraph maintains that the claim language and the specification compel such an interpretation and that the specification's disclosure of a "specific" pipeline relates only to a preferred embodiment. According to Intergraph, the one prosecution history remark regarding a "specific" pipeline should not negate the contrary meaning found in the patent itself. With respect to infringement, Intergraph contends that the district court correctly found that the template bits associated with each instruction in the accused devices satisfy the "pipeline identifier" limitations.

We agree with Intel that the district court erred in its construction of the claim term "pipeline identifier."[4] We commence our analysis, as we must, with the claim language. To begin with, some claim language can be read to suggest that a "pipeline identifier" identifies a particular pipeline. For example, claim 20 of the '028 patent recites "a unique pipeline identifier indicative of the processing pipeline which will execute that individual instruction." (emphases added). And claims 1 and 22 of the '003 patent refer to using "pipeline identifiers" to supply each individual instruction to "the appropriate instruction pipeline." (emphasis added). At the same time, however, other claim language can be read to suggest that a "pipeline identifier" need not identify a specific pipeline. For instance, claim 20 also recites "using the unique pipeline identifiers . . . [to control switches and] thereby supply each individual instruction in the at least one group to be executed in parallel to an appropriate processing pipeline." (emphasis added). Claim 6 of the '003 patent similarly refers to using "pipeline identifiers" to supply each instruction to "an appropriate instruction pipeline." (emphasis added). Still, that language does not clarify the degree of specificity required by the term "pipeline identifier," for both the specific pipeline that will execute an instruction and the type of pipeline that will execute an instruction are "appropriate" pipelines. We therefore read the claim language as being susceptible to both parties' proposed definitions.

We turn next to the specifications, which also offer only equivocal support for each party's proposed definition. On the one hand, the specifications of both patents contain statements suggesting that a "pipeline identifier" must identify a specific pipeline. For example, the "Summary of the Invention" portions disclose "a pipeline identifier indicative of the pipeline for executing [an] instruction." '028 patent, col. 3, ll. 22-23 (emphasis added); see '003 patent, col. 2, ll. 10-11. Even more persuasively, but limited in context to describing a preferred embodiment of the invention, the specifications also disclose a "pipeline tag indicative of the specific pipeline to which [an] instruction should be dispatched." '028 patent, col. 5, ll. 14-16 (emphasis added); '003 patent, col. 3, ll. 43-45 (emphasis added). Furthermore, Figures 10 and 11 of the '028 patent and Figure 3 of the '003 patent depict "pipeline identifiers" that identify specific pipelines.[5] More precisely, they show pipeline identifier bits 1, 3, and 6 as signaling that their corresponding instructions will be routed to pipelines 1, 3, and 6, respectively. '028 patent, figs. 10, 11; '003 patent, fig. 3.

On the other hand, the specifications also contain statements suggesting that a "pipeline identifier" need identify only the type of pipeline to which an instruction will be routed. For example, both specifications explain that, in some embodiments, the compiler's determination of "the appropriate pipeline for execution of an individual instruction . . . . is essentially a determination of the type of instruction provided." '028 patent, col. 6, ll. 56-59 (emphasis added); '003 patent, col. 3, ll. 64-67 (emphasis added). That statement is qualified, however, by the following sentence, which provides examples of pipeline types and states that "load instructions will be sent to the load pipeline [and] store instructions to the store pipeline." '028 patent, col. 6, ll. 59-61 (emphases added); '003 patent, col. 3, l. 67 to col. 4, l. 2 (emphases added). Similarly, the statement that "[p]referably, the pipeline tag will correspond to the type of functional unit required for execution of [an] instruction" is qualified by its ensuing example, which identifies a particular execution unit—namely, "floating point unit 1." '028 patent, col. 3, ll. 15-17 (emphasis added). Thus, the two examples that refer to identifying pipelines by type actually identify specific pipelines. We therefore find them to be of little assistance to our claim construction analysis.[6]

Finally, we arrive at the prosecution history, which, unfortunately for Intergraph, resolves any ambivalence. A patentee may, of course, act as his own lexicographer and define a claim term in either the specification or the prosecution history. Mycogen Plant Sci., Inc. v. Monsanto Co., 243 F.3d 1316, 1327 (Fed. Cir. 2001). We conclude that Intergraph did just that during prosecution of the '028 patent. In the first office action, the examiner rejected claims 1-23 as indefinite under 35 U.S.C. § 112, stating, among other reasons, that it was "unclear what constitutes a pipeline identifier" in claims 20 and 21. In response, Intergraph amended claim 20 and explained that a "pipeline identifier" is a "tag . . . indicative of the specific processing pipeline to which [an] instruction will be dispatched." (emphasis added). With that statement, Intergraph clearly and unambiguously defined the term "pipeline identifier" as referring to the particular pipeline to which an instruction will be routed, and Intergraph cannot now obtain a broader interpretation of that term. See Honeywell Inc. v. Victor Co. of Japan, 298 F.3d 1317, 1323-24 (Fed. Cir. 2003).

Nevertheless, Intergraph argues that that prosecution history statement referred only to a preferred embodiment and therefore should not be applied to limit the scope of the claims. We disagree. The examiner's question regarding the meaning of the term "pipeline identifier" was raised in the context of rejecting claim 20, so Intergraph's response is at least relevant to that claim's use of the term. Moreover, just before defining the term "pipeline identifier," Intergraph stated in its response to the examiner that the "pipeline identifiers are described throughout the specification, for example, beginning on page 8 at line 15." Although the text at page 8, line 15 does indeed appear to refer to the embodiment depicted in Figure 1, Intergraph explicitly identified its discussion of pipeline identifiers as occurring "throughout the specification," with the page 8, line 15 reference being only one example of its use of the term. We therefore read Intergraph's prosecution history statement as not being limited in scope to a preferred embodiment, but rather as indicating the meaning that Intergraph ascribed to the term "pipeline identifier" throughout the '028 patent.

Intergraph's statement made during prosecution of the '028 patent is also relevant to our interpretation of the '003 patent. Because the claim language and the specification of the '003 patent have not resolved the ambiguity surrounding the term "pipeline identifier," we find the prosecution history of the '028 patent to be highly relevant to our construction of the term "pipeline identifier" in the '003 patent. The '003 patent incorporates by reference copending U.S. Patent Application 08/147,800, which led to the '028 patent, and more particularly cites that application for a description of the overall system within which the '003 patent's "associative crossbar switch" is implemented. '003 patent, col. 2, ll. 52-58. Because the overall system disclosed in the '800 application plainly includes "pipeline identifiers" and because the two patents in suit contain overlapping (and at times identical) claim language and disclosures regarding "pipeline identifiers," we consider that Intergraph intended that term to have the same meaning in both patents. We therefore apply Intergraph's definition of the term "pipeline identifier" made during prosecution of the '028 patent to the '003 patent as well.

In sum, we conclude that Intergraph expressly defined the term “pipeline identifier” during prosecution of the ’028 patent. In light of that prosecution history, as well as the claim language and the specifications, we hold that, for both the ’028 patent and the ’003 patent, a “pipeline identifier” must identify the specific processing pipeline to which an instruction will be dispatched.

Having revised the district court’s construction of the claim term “pipeline identifier,” we turn next to the question of infringement. Unfortunately, we are unable to determine from the record whether the templates in the accused devices identify the specific pipelines to which instructions will be routed, and the district court made no findings of fact regarding that aspect of the accused devices. We therefore vacate the district court’s finding of infringement and remand for the court to determine in the first instance whether the accused devices have “pipeline identifiers” under our construction of that term.

B. “Groups of Individual Instructions”

Intel next contests the district court’s construction of the term “groups of individual instructions” in the ’028 patent and the term “groups [or sets] of software-scheduled instructions” in the ’003 patent. Intel argues that those limitations require that the instructions within a group can actually be executed in parallel by the claimed processing pipelines, and not just that they are free from certain data dependencies. Intel further contends that the district court clearly erred in finding that the accused devices form “groups of instructions” in the compiler because the accused devices’ “independence groups” may be subject to data dependencies and hardware constraints that preclude parallel execution and because the accused devices’ “issue groups” are formed by hardware downstream of main memory.

In response, Intergraph defends the district court’s construction of the “groups of instructions” limitations, arguing that the claim language refers to groups of instructions that are “executable,” not “actually executed,” in parallel. According to Intergraph, the recited compiler determines only which instructions can be processed in parallel and allows for situations in which the microprocessor itself determines which instructions will ultimately be processed in parallel. Intergraph also argues that the district court did not clearly err in finding that the “groups of instructions” limitations read on the accused devices, pointing to evidence that the accused devices' compilers create instruction groups such that all instructions in any given group can be executed in parallel.

We agree with Intergraph that the district court did not err in its construction of the "groups of instructions" limitations. Claim 20 of the '028 patent and claims 1, 6, and 22 of the '003 patent simply refer to instructions within a group as being "executable in parallel." The plain language thus suggests, and the district court so concluded, that the recited instruction groups "can be" or are "capable of being" executed in parallel. Nowhere do the claims require that the instructions within a group be free of all data dependencies and hardware constraints that could later impede parallel execution. Indeed, the specifications of both patents suggest otherwise. In the context of describing the formation of instruction groups, both patents disclose only that the compiler checks for data dependencies that would preclude parallel execution. See '028 patent, col. 5, ll. 21-23 ("[A]ll instructions in the group must be capable of simultaneous execution; e.g., there cannot be data dependency between instructions."); '003 patent, col. 3, ll. 58-63 ("During the compilation, the instructions are checked for data dependencies, dependence upon previous branch instructions, or other conditions that preclude their execution in parallel with other instructions. The result of the compilation is identification of a set or group of instructions which can be executed in parallel."). In their later discussions of which instructions will actually be dispatched to which pipelines, however, both specifications acknowledge that "[t]he particular pipeline to which a given instruction word is dispatched will depend upon hardware constraints as well as data dependencies." '028 patent, col. 10, ll. 41-43; '003 patent, col. 6, ll. 23-25. The patents thus disclose that data dependencies are checked for during the formation of instruction groups and that hardware constraints may be considered at some time before it is determined which pipelines will actually execute which instructions. We see nothing in the patents that requires more than data dependencies to be considered when the recited instruction groups are formed.[7] We therefore conclude that the district court did not err in construing the terms "groups of individual instructions" and "groups [or sets] of software-scheduled instructions."

Nor did the district court clearly err in finding that the accused devices satisfy the "groups of instructions" limitations. Intel's primary allegation of factual error—that the instructions in the accused devices' "independence groups" are not executable in parallel because they have not been cleared for all data dependencies and hardware constraints—is basically a restatement of its claim construction argument, which we have already rejected. In any event, Intergraph points to ample evidence in the record to support the court's finding that the accused microprocessors have "groups of instructions." For example, an article describing the architecture of the accused devices states that "[t]he compiler creates instruction groups so that all instructions in an instruction group can be safely executed in parallel." In addition, Intel's expert agreed that software in the Itanium microprocessors checks the instructions within a group for certain data dependencies that would preclude parallel execution. Another Intel witness described the "independence groups" in the accused devices as groups of instructions that are independent and can be issued in parallel, and still another Intel witness testified that software in the Itanium 2 microprocessor forms groups of instructions that it believes can be executed in parallel. In view of that evidence, we cannot say that the district court clearly erred in finding that the accused devices satisfy the "groups of instructions" limitations in the '028 and '003 patents.

C. "Instructions"

Intel's next contention is that the district court erroneously construed the term "instruction," found in both the '028 and the '003 patents, to have different meanings in different parts of the claims. Intel urges us to interpret the term "instruction" to mean "a complete instruction corresponding to a traditional programmer visible command that uniquely describes one basic computer operation." Under either the district court's construction or its own proposed construction, Intel contends that the accused devices do not infringe because they do not store "instructions" with embedded pipeline identifiers or route "instructions" through the crossbar as the claims require.

Intergraph counters that the district court's construction of the term "instruction" is both consistent and correct because, at every stage, an "instruction" is the smallest unit of work capable of being processed. Intergraph further argues that the accused devices' syllables and templates are the "instructions" in the storage stage and that the accused devices' syllables are the "instructions" in the routing stage.

We agree with Intergraph that the district court did not err in construing the claim term "instruction." Claim 20 of the '028 patent and claims 1, 6, and 22 of the '003 patent refer to storing "instructions" in VLIW or super-scaler storage and then routing "instructions" to appropriate processing pipelines. Taking into account both the storage stage and the routing stage, the district court interpreted the term "instructions" to mean simply "the smallest unit of work capable of being processed at any particular stage in the computer." That construction does not run afoul of the general rule that a claim term should be interpreted consistently throughout a claim, see Phonometrics, Inc. v. N. Telecom Inc., 133 F.3d 1459, 1465 (Fed. Cir. 1998); Southwall Techs., Inc. v. Cardinal IG Co., 54 F.3d 1570, 1579 (Fed. Cir. 1995), for it treats an "instruction" as always being the "smallest unit of work capable of being processed." Moreover, the patent does not require that an "instruction" contain a specific, or even unvarying, number of bits. On the contrary, the specification of the '028 patent shows that an "instruction" may contain 64 bits in the storage stage and only 57 bits in the routing stage, after the group and pipeline identifiers have been discarded. See '028 patent, col. 7, l. 56 ("As shown [in Figure 7], the instructions are expanded to 64 bit length . . . ."); id., col. 9, ll. 30-62 & fig. 10 (depicting 64-bit "instructions" as each consisting of a 57-bit word, a 3-bit group identifier, and a 4-bit pipeline identifier); id., col. 10, ll. 31-32 & fig. 11 (referring to the 57-bit words as "instructions" that are executed by the processing pipelines). It is therefore irrelevant whether the number of bits composing an "instruction" is constant throughout all of the recited stages of processing. We accordingly affirm the district court's construction of the term "instructions."

We also agree with Intergraph that the district court did not clearly err in finding that the accused devices have "instructions" in both the storage and the routing stages. First, the record supports the finding that the accused devices' syllables and templates constitute "instructions" in the storage stage. To be sure, the parties offered conflicting evidence regarding whether the accused devices' templates ever reach the rotate buffer and hence ever reside in VLIW or super-scaler storage. Nonetheless, the district court reasonably chose to rely on the testimony of Intergraph's expert, who stated that, based on his review of an Intel design document, both the syllables and the templates in the accused devices are sent to the rotate buffer. See Findings and Conclusions, slip op. at 5. Moreover, that Intel document also appears to show bundles, which include both syllables and templates, as leaving the rotator in the accused devices. It was therefore not clearly erroneous for the court to resolve this factual dispute by finding that the accused devices have "instructions," which include embedded pipeline identifiers in the form of the templates, in the storage stage. Id. at 7. Second, the record also supports the finding that the accused devices' syllables constitute "instructions" in the routing stage. On this point, Intel's own expert agreed that the accused devices' syllables are the smallest units that an execution pipeline needs for processing, thereby fitting the court's definition of "instructions" in the routing stage. We therefore conclude that the district court's finding that the accused devices have the recited "instructions" was not clearly erroneous.

D. "Crossbar"

Intel also challenges the district court's construction of the "crossbar" limitation in the '003 patent, arguing that the recited crossbar must be implemented within the cache. Intel asserts that the claim language, the specification, and the prosecution history all require that the crossbar be physically located within the cache. Intel further argues that there can be no infringement under its proposed definition of the term "crossbar" because the accused devices have significant decoding and processing circuitry located between the cache and the crossbar.

Intergraph responds that nothing in the claim language, the specification, or the prosecution history specifies the location or the position of the crossbar in the claimed invention. Intergraph further maintains that the district court's finding that the accused microprocessors satisfy the "crossbar" limitations should be affirmed because there was no claim construction error.

We agree with Intergraph that the district court did not err in construing the asserted claims of the '003 patent as not requiring the crossbar to be located within the cache. Claims 1 and 22 recite a "super-scaler cache comprising . . . an associative crossbar," while claim 6 refers to "an associative crossbar switch in a super-scaler cache." Intel argues that the plain meaning of the words "in" and "comprising" mandates that the crossbar be located within the cache. We disagree, as neither "comprising" nor "in" necessarily conveys information about the location or position of the crossbar relative to the cache. The word "comprising" in claims 1 and 22 is a term of art that simply means that the crossbar is an essential, but not necessarily the only, element of the cache. See Genentech, Inc. v. Chiron Corp., 112 F.3d 495, 501 (Fed. Cir. 1997). The word "in" may similarly be used to indicate that the crossbar is a constituent of the cache; or it may be used to indicate that the crossbar is within the bounds or area of the cache. See Webster's New World Dictionary 680 (3d college ed. 1988). Nothing in the claim language, however, clearly points to one particular meaning over another for that ordinary word. We therefore conclude that the claim language does not compel Intel's proposed claim construction.

The specification of the '003 patent likewise does not require the crossbar to be located within the cache. Indeed, Figures 2, 4, and 5 depict the instruction cache and the crossbar as two distinct entities; they certainly do not show the crossbar as being located within the cache. And the specification, at most, suggests that the preferred embodiment places the crossbar in the cache, stating only that the placement of the crossbar switch earlier in the instruction pipeline "allows the crossbar to be a part of the cache itself," '003 patent, col. 1, ll. 46-47 (emphasis added), and that "[i]n a preferred embodiment of this invention the associative crossbar is implemented in the instruction cache," id., col. 2, ll. 22-23. Without more, however, we will not read that purported aspect of the preferred embodiment into the broader claim language. See RF Del., Inc. v. Pac. Keystone Techs., Inc., 326 F.3d 1255, 1263 (Fed. Cir. 2003).

The prosecution history of the '003 patent also fails to convince us that the recited crossbar must be located within the cache. Intel asserts that Intergraph distinguished the prior art during prosecution on the basis that the claimed invention places the associative crossbar within the cache. However, the prosecution history shows that Intergraph distinguished the Blaner and Hiller references by arguing that neither disclosed using an associative crossbar switch, let alone a cache that includes an associative crossbar switch. The prosecution history also shows that Intergraph distinguished the Rustad reference by pointing out that it did not disclose the recited means and steps for forming and storing groups of software-scheduled instructions.[8] Intergraph's other prosecution history remarks regarding the recited associative crossbar switch largely repeat the language found in the specification of the ’003 patent, which we have already found to be unconvincing. We therefore do not read Intergraph’s prosecution history statements as clearly disclaiming crossbars that are not physically located within the cache.

Having affirmed the district court’s claim construction, we discern no clear error in the court’s factual finding that the accused devices satisfy the “crossbar” limitations. Intel essentially concedes as much. We therefore affirm the court’s finding that the accused devices have “crossbars” as claimed in the ’003 patent.

E. “Associative Crossbar”

Intel lastly argues that the district court clearly erred in finding that the crossbars in the accused devices are “associative.” More specifically, Intel asserts that only the syllables, which do not include routing instructions, are routed to the crossbars in the accused devices and that the accused devices consequently lack “associative crossbars.”

Intergraph responds that the district court did not clearly err in finding that the crossbars in the accused devices are “associative” because the templates in the accused devices are not stripped off from the syllables upstream of the crossbars. Alternatively, Intergraph argues that the accused devices’ crossbars include their associated dispersal logic and thus undisputedly contain the routing instructions, which are part of the templates.

Although we reject Intergraph’s first argument,[9] we agree with Intergraph that the district court’s finding that the crossbars in the accused devices are “associative”—i.e., that the data being routed through the crossbars include the routing instructions—was not clearly erroneous. The court found that the “crossbar” in the Itanium microprocessor includes both the steering crossbar and its associated dispersal logic and that the “crossbar” in the Itanium 2 microprocessor includes both the dispersal multiplexers and their associated dispersal logic. Findings and Conclusions, slip op. at 10. Under that characterization of the accused devices, which is not clearly erroneous, the data being routed clearly include the templates, which in turn include the routing instructions. We therefore uphold the district court's finding that the accused devices have "associative crossbars."

CONCLUSION

For the foregoing reasons, we conclude that the district court erred in its construction of the claim term "pipeline identifier." We therefore vacate the court's finding of infringement with respect to both the '028 patent and the '003 patent, and we remand for the court to consider in the first instance whether the accused devices infringe the two patents in suit under our narrower claim construction. We do, however, affirm the district court's claim constructions and infringement findings for all of the other claim limitations on appeal. Accordingly, the decision of the district court is vacated and remanded.

---

[1] Intel also sought to have the two patents in suit declared unenforceable, but later abandoned that counterclaim. See Intergraph Corp. v. Intel Corp., No. 2-01-CV-160-TJW, slip op. at 1 (E.D. Tex. Oct. 30, 2002) ("Final Judgment and Permanent Injunction").

[2] The court also held that Intel failed to prove by clear and convincing evidence that the '028 and '003 patents are invalid. Findings and Conclusions, slip op. at 16.

[3] Because the two patents in suit contain overlapping claim language and specifications, we discuss them in parallel. However, we have considered each patent individually and note those differences between the two patents that are relevant to our claim construction analysis.

[4] Both parties argue, and we agree, that the term "pipeline identifier" should be interpreted consistently throughout the claims in each patent. See Rexnord Corp. v. Laitram Corp., 274 F.3d 1336, 1342 (Fed. Cir. 2001).

[5] Intergraph argues that Figures 12 and 13 of the '028 patent illustrate "pipeline identifiers" that identify pipelines by type. Those figures, however, illustrate an instruction frame and a group of instructions. They, along with their accompanying text, depict instruction types and group tags but, importantly, do not show "pipeline identifiers." See '028 patent, col. 10, ll. 44-67. We therefore find Figures 12 and 13 to be unhelpful to our interpretation of the term "pipeline identifier."

[6] Elsewhere, the specifications, like the claim language, ambivalently disclose that a "pipeline identifier" determines "an appropriate pipeline." E.g., '028 patent, col. 3, ll. 13, 28-29; see also '028 patent, col. 5, ll. 31-32 (stating that each instruction in a group is routed to "its appropriate pipeline, as determined by the pipeline tag of the instruction").

[7] Moreover, we reject Intel's argument that during prosecution of the '028 patent Intergraph defined the "groups of individual instructions" as consisting of instructions that are actually executed simultaneously. In the first office action, the examiner stated that it was unclear whether the individual instructions within a group or the groups themselves were executable in parallel. Intergraph clarified that the former interpretation was correct, explaining that "the goal of the invention is to execute each individual instruction in a group in parallel." Nothing in Intergraph's response contradicts the district court's construction of the term "groups of individual instructions."

[8] During prosecution Intergraph did state: "Fig[ure] 1 is the only figure in Rustad et al. that illustrates a crossbar switch in conjunction with a cache. Furthermore, the crossbar switch is not within a cache." That Figure 1 was of a prior art reference cited in Rustad and depicted an instruction cache and a crossbar in nearly the same way as do the block diagrams in the '003 patent. Intergraph did not meaningfully explain what distinction, if any, existed between its claimed invention and Rustad's Figure 1 and, in any event, did not rely on any such distinction to overcome the examiner's § 103 rejection, which was based on Rustad itself. Intergraph's one vague statement regarding Figure 1 therefore does not amount to a clear disavowal of claim scope.

[9] The evidence cited by Intergraph supports only that the templates leave the rotate buffer; it does not show that the templates actually reach the steering crossbar in the Itanium device or the dispersal multiplexers in the Itanium 2 device.