

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

KEKER & VAN NEST LLP
ROBERT A. VAN NEST - # 84065
rvannest@kvn.com
CHRISTA M. ANDERSON - # 184325
canderson@kvn.com
DANIEL PURCELL - # 191424
dpurcell@kvn.com
633 Battery Street
San Francisco, CA 94111-1809
Telephone: 415 391 5400
Facsimile: 415 397 7188

KING & SPALDING LLP
DONALD F. ZIMMER, JR. - #112279
fzimmer@kslaw.com
CHERYL A. SABNIS - #224323
csabnis@kslaw.com
101 Second Street, Suite 2300
San Francisco, CA 94105
Tel: 415.318.1200
Fax: 415.318.1300

KING & SPALDING LLP
SCOTT T. WEINGAERTNER
(Pro Hac Vice)
sweingaertner@kslaw.com
ROBERT F. PERRY
rperry@kslaw.com
BRUCE W. BABER (Pro Hac Vice)
1185 Avenue of the Americas
New York, NY 10036
Tel: 212.556.2100
Fax: 212.556.2222

IAN C. BALLON - #141819
ballon@gtlaw.com
HEATHER MEEKER - #172148
meekerh@gtlaw.com
GREENBERG TRAURIG, LLP
1900 University Avenue
East Palo Alto, CA 94303
Tel: 650.328.8500
Fax: 650.328.8508

Attorneys for Defendant
GOOGLE INC.

UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA
SAN FRANCISCO DIVISION

ORACLE AMERICA, INC.,

Plaintiff,

v.

GOOGLE INC.,

Defendant.

Case No. 3:10-cv-03561 WHA

**GOOGLE INC.'S OPPOSITION TO
ORACLE AMERICA, INC.'S RULE 50(A)
MOTION AT THE CLOSE OF ALL
EVIDENCE FOR PHASE II**

Dept.: Courtroom 8, 19th Floor
Judge: Hon. William Alsup

TABLE OF CONTENTS

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

I. INTRODUCTION 1

II. A REASONABLE JURY COULD *ONLY* FIND THAT GOOGLE DID NOT INFRINGE THE ASSERTED CLAIMS OF THE 104 PATENT..... 2

 A. Dalvik bytecode instructions only contain indexes, which are not “symbolic references” because they refer to numeric memory locations..... 2

 1. Field indexes are not symbolic references..... 3

 2. The Court’s claim construction applies to data in the constant pool tables as well as the actual field data in an instance object. 5

 3. Indexes are not symbolic references because they are not resolved..... 8

 B. Dexopt does not infringe the ’104 patent..... 9

 C. The Court should reject Oracle’s post-trial attempt to re-constitute the term “symbolic reference.” 10

III. A REASONABLE JURY COULD ONLY FIND THAT GOOGLE DOES NOT INFRINGE THE ASSERTED CLAIMS OF THE ’520 PATENT. 11

 A. The jury is entitled to credit Professor Parr’s testimony explaining why pattern matching differs from simulating execution..... 12

 B. Simulating execution, properly understood, involves the simulation of actual Java virtual machine operations. 13

IV. ORACLE’S PATENT INFRINGEMENT CLAIMS ARE BARRED BY GOOGLE’S EQUITABLE DEFENSES. 14

 A. Oracle’s “additional facts” are contradicted by the evidence 15

 B. Oracle’s patent infringement claims are barred by equitable estoppel. 16

 C. Oracle’s patent infringement claims are barred by laches. 16

 D. Oracle’s patent infringement claims are barred by waiver. 17

 E. Oracle’s patent infringement claims are barred by implied license..... 18

V. GOOGLE CHOSE NOT TO PRESENT ITS DEFENSES OF PATENT MISUSE, USE BY THE UNITED STATES, UNCLEAN HANDS, AND EXPRESS LICENSE, AND THEREFORE DOES NOT OPPOSE ORACLE’S MOTION ON THESE DEFENSES. 18

VI. CONCLUSION..... 19

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

TABLE OF AUTHORITIES

Federal Cases

A.C. Aukerman Co. v. R.L. Chaides Const. Co.
960 F.2d 1020 (Fed. Cir. 1992)..... 15, 16

De Forest Radio Tel. Co. v. United States
273 U.S. 236 (1927)..... 17

Eon-Net LP v. Flagstar Bancorp
653 F.3d 1314 (Fed. Cir. 2011)..... 13

In re Katz Interactive Call Processing Patent Litigation
712 F. Supp. 2d 1080 (C.D. Cal. 2010) 16

Lucent Technologies, Inc. v. Gateway, Inc.
580 F. Supp. 2d 1016 (S.D. Cal. 2008) *aff'd in part, vacated in part, remanded*,
580 F.3d 1301 (Fed. Cir. 2009)..... 16

McCoy v. Mitsuboshi Cutlery, Inc.
67 F.3d 917 (Fed. Cir. 1995)..... 18

Wang Laboratories, Inc. v. Mitsubishi Electronics Am., Inc.
103 F.3d 1571 (Fed. Cir. 1997)..... 17

1 **I. Introduction**

2 The Court should deny Oracle's motion for judgment as a matter of law. Under the
3 Court's claim construction, no reasonable jury could find that Android infringes Claims 11, 27,
4 29, 39, 40, and 41 of U.S. Patent No. RE38,104 (the '104 patent). The construction of "symbolic
5 references" does not, as Oracle suggests, limit the term "data" to "the actual field data in an
6 instance object." Moreover, the terms "dynamically" and "statically" do not take on the tortured
7 meanings Oracle now ascribes to them in yet another attempt to modify the Court's claim
8 construction.

9 Likewise, Oracle's request for judgment as a matter of law that Google infringes the '520
10 patent is based on no more than the view of Oracle's counsel as to its desired claim construction
11 for the term "simulating execution." Yet neither Oracle's own expert, nor even Oracle itself in its
12 brief, focuses on the actual claim language at issue. Oracle is also incorrect that the disputed
13 claim language necessarily encompasses pattern matching. As Google explained at trial, a pattern
14 matcher will produce outcomes different than what results from simulating execution, and
15 Google's dx tool proves the point, as it does not return identical results when it is applied to
16 bytecode that would be indistinguishable for purposes of simulating execution. As explained in
17 Google's own motion for judgment as a matter of law, Oracle's current theory is in essence a
18 belated doctrine of equivalents argument. It should be rejected.

19 Oracle's motion as to Google's equitable defenses also fails. The evidence at trial
20 established that Sun publicly and privately applauded Android for years. From the time of Mr.
21 Schwartz's blog post in 2007 through Mr. Ellison's efforts to form an Oracle-Google partnership
22 in 2010, Sun (and later Oracle) never once raised the issue of alleged patent infringement. The
23 fact that, during that time, Sun still desired to work with Google and hoped to convince Google to
24 buy a TCK license does not undermine Google's equitable defenses because the parties never
25 discussed any patents, let alone a license for the patents-in-suit. To the contrary, Sun actively
26 encouraged Google and even built products to run on Android. During this time, Google invested
27 money, time, and resources in developing Android and bringing a smart phone to market. This
28 undisputed evidence is more than sufficient to grant judgment in Google's favor on its affirmative

1 defenses of equitable estoppel, laches, waiver, and implied license.

2 For all of these reasons, Oracle’s motion should be denied.

3 **II. A reasonable jury could *only* find that Google did not infringe the asserted claims of**
 4 **the 104 patent.**

5 “When *I* use a word,” Humpty Dumpty said,
 6 in rather a scornful tone, “it means just what I
 choose it to mean—neither more nor less.”

7 “The question is,” said Alice, “whether you
 8 *can* make words mean so many different things.”

9 “The question is,” said Humpty Dumpty,
 “which is to be master—that’s all.”

10 —Lewis Carroll, *Through the Looking Glass*

11 TX 2564 at 5/852 (“The Java Language Specification”).

12 **A. Dalvik bytecode instructions only contain indexes, which are not “symbolic**
 13 **references” because they refer to numeric memory locations.**

14 Oracle’s theory has morphed yet again. In his opening report, Dr. Mitchell called the
 15 indexes in the Dalvik bytecode instructions “numeric references,” acknowledging the undisputed
 16 fact that they identify locations in constant pool tables. Then, in his trial testimony, Dr. Mitchell
 17 called the indexes symbolic references, claiming that those indexes are names. Now, in its
 18 motion, Oracle argues that indexes qualify as *both* numeric and symbolic references. This
 19 schizophrenic theory permeates its motion for judgment as a matter of law:

- 20 • “A field index—also called field@CCCC generally or ‘01’ in specific examples of
 21 field indices in the trial testimony—is a reference to data to be obtained in
 22 accordance with a corresponding numerical reference, and identifies that data by a
 name other than the numeric memory location of the data.” [Oracle Mot. for
 JMOL at 2.]
- 23 • “For the field index in the IGET instruction to be a symbolic reference, it is
 24 enough that it identifies—‘specifies,’ in Mr. McFadden’s words—*some* data to be
 obtained, by something other than the data’s location.” (*Id.* at 3.)
- 25 • “That indexes such as the ‘01’ in ‘52 01’ (the bytecode for IGET field@0001) may
 26 *also* indicate the location of information in the dex file’s constant pool is *not*
 27 *relevant*. The claims do not require that a reference *exclusively* identify data
 28 symbolically to qualify as a symbolic reference.” (*Id.* at 4.)
- “Thus, even if a field index were a numeric memory location in the Field ID table,
 it would still nevertheless be a symbolic reference, because it *also* identifies the

1 value of a field in an instance object—the only data that is relevant to the claims—
2 by a name other than its location.” (*Id.* at 6.)

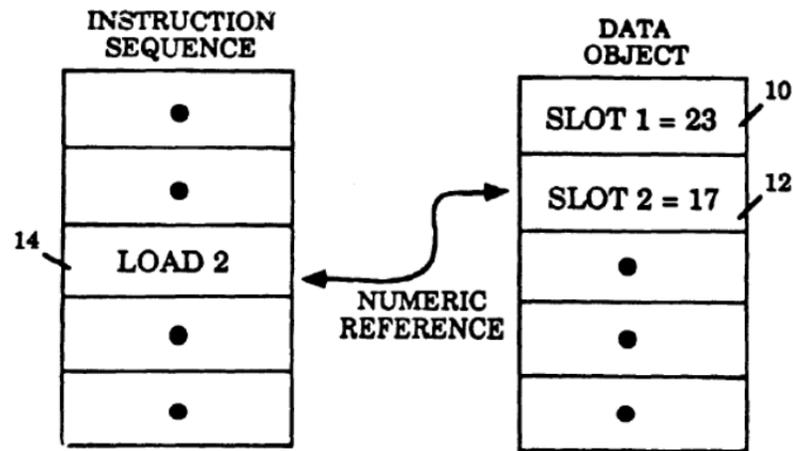
3 As discussed below, not only should these new—and illogical—theories be rejected as a matter of
4 law, they highlight the fact that the indexes in Dalvik bytecode instructions are indeed numeric
5 rather than symbolic references. Accordingly, judgment as a matter of law should be entered in
6 favor of Google, not Oracle.

7 **1. Field indexes are not symbolic references.**

8 There is no dispute that Dalvik bytecode instructions—the instructions evaluated by both
9 Resolve.c and the dexopt tool—consist of an opcode and operand. [TX 737 (“Each instruction
10 starts with the named opcode and is optionally followed by one or more arguments.”); RT
11 3221:8-10, 3234:4-7 (McFadden); RT 3590:23-3591:25 (Bornstein); RT 3844:14-3855:12,
12 3925:4-18 (August).] Nor is there any dispute that the accused operands are indexes that take the
13 form “field@CCCC” and reference locations in tables. [See Oracle Mot. for JMOL at 2-3; TX
14 46.106; TX 735 (“There are separately enumerated and indexed constant pools for references to
15 strings, types, fields, and methods.”); TX 737; RT 3732:15-19, 3736:16-23, 3755:8-9, 3765:9-12
16 (McFadden); 3858:5-12, 3858:21-359:11, 3918:13-23, 3923:20-24, 3925:19-3926-9 (August).]
17 Even Dr. Mitchell agrees. [RT 3488:6-8, 3488:19-23, 3489:10-12 (Dr. Mitchell) (the “classIdx”
18 index in Dalvik instructions provides information “to a location in another table”); RT 3496:12-
19 3497:6 (the “field Idx” in Dalvik instructions is “an index to a specific location in the field
20 table”).] These indexes refer to locations in the constant pool tables, which are data in the .dex
21 file; for example, “field@0001” refers to the entry 1 in the Field IDs table. [See Oracle Mot. for
22 JMOL at 4.]

23 Oracle focuses its analysis on the IGET instruction in Dalvik bytecode. [See *id.* at 3.] It
24 argues that the IGET instruction corresponds to the “LOAD ‘y’” instruction in Figure 1B of the
25 ’104 patent. [*Id.*] That is the wrong example. As Oracle notes, “[t]he IGET instruction contains
26 three operands—vA, vB, and field@CCCC—where the third operand field@CCCC is the field
27 index.” [*Id.* at 3 (citing TX 735 at 6; RT 3221:8-10 (McFadden)).] That index identifies an entry
28

1 in the Field ID table. As such, the IGET instruction in the Dalvik bytecode instructions actually
 2 corresponds to the “LOAD 2” instruction depicted in Figure 1A the ’104 patent.



3
4
5
6
7
8
9
10
11 **Figure 1A**
12 **Prior Art**

13 The instruction, LOAD 2, directs the computer to go to Slot 2 in the table, much like the IGET
 14 instruction 52 01 instructs the computer to go to entry 01 in the Field ID table. Even Oracle
 15 concedes this is how the IGET instruction works. [*Id.* at 4.] As such, the operands for IGET
 16 instructions in Dalvik bytecode—the indexes—qualify as numeric references rather than
 17 symbolic references. Because those indexes refer to data by a numeric memory location—entry
 18 01 in the Field IDs table—they do not meet the Court’s construction of “symbolic references.”

19 In light of all the evidence presented at trial, Oracle now concedes that indexes in the
 20 instruction stream indicate the location of information in the constant pool tables, making them
 21 numeric references in at least that respect. “That indexes such as the ‘01’ in ‘52 01’ (the bytecode
 22 for IGET field@0001) may also indicate the location of information in the dex file’s constant
 23 pool is not relevant.” [*See* Oracle Mot. for JMOL at 4 (emphases removed).] Oracle’s new
 24 argument that this is not relevant, however, is insincere—indeed, the whole issue is whether the
 25 indexes qualify as symbolic or numeric references. Oracle continues, “[t]he claims do not require
 26 that a reference *exclusively* identify data symbolically to qualify as a symbolic reference.” [*Id.*]
 27 Along those same lines, Oracle argues that “even if a field index were a numeric memory location
 28 in the Field ID table, it would still nevertheless be a symbolic reference, because it *also* identifies

1 the value of a field in an instance object—the only data that is relevant to the claims—by a name
 2 other than its location.” [*Id.* at 6 (emphasis in original).] This is the first time that Oracle has
 3 posited the idea of “hybrid” numeric-symbolic references, which conflicts with the patent and the
 4 Court’s Claim Construction Order. [*See* Dkt. No. 137 at 20 (“The ’104 patent teaches two
 5 different types of data references: numeric references and symbolic references.”); *see also* TX
 6 4015.] And, tellingly, neither Dr. Mitchell nor any other witness testified that indexes could
 7 simultaneously qualify as *both* numeric and symbolic references. In fact, Oracle cites nothing
 8 from the record evidence in support of those new-found “dual-status” references. They are
 9 simply an after-the-fact argument developed by Oracle’s counsel to try to salvage its case.

10 **2. The Court’s claim construction applies to data in the constant pool**
 11 **tables as well as the actual field data in an instance object.**

12 Oracle’s tortured re-interpretation of the Court’s claim construction provides no basis for
 13 judgment as a matter of law. As is clear on its face, the Court’s construction distinguishes
 14 between using names to represent data (i.e., symbolic references) and using numeric memory
 15 locations:

16 a reference that identifies data by a name other than the numeric memory location
 17 of the data, and that is resolved dynamically rather than statically

18 [Dkt. No. 137 at 22.] Dr. Mitchell agrees. [RT 3480:12-15.] There is no limitation regarding the
 19 type of data being referenced.

20 Regardless, Oracle postulates that “the ‘data’ in the Court’s construction of symbolic
 21 reference is the actual field data ‘obtained’ by Dalvik—the value of a field in an ‘instance
 22 object’—rather than the constant pool information in the Android dex file that is the focus of
 23 Google’s arguments.” [Oracle Mot. for JMOL at 5 (parenthetical with claim construction
 24 omitted); *see also* RT 4019:3-7 (Mitchell).] This reading is much more restrictive than what the
 25 ordinary meaning of the word “data” implies. According to Dr. Mitchell, the “data” must be the
 26 actual field data in the instance object, as opposed to any data in the .dex file, including the data
 27 in the constant pool tables. [RT 4025:6-21 (Mitchell) (“So in order for that to happen, we have to
 28 find where that actual data is in the object using this symbolic reference. . . .”)] As explained in

1 Google’s motion for judgment as a matter of law, to turn an index—which is a numeric reference
2 to a location in a table—into a symbolic reference, Dr. Mitchell must resort to improperly re-
3 defining “symbolic reference” to mean “a reference that identifies *the actual field data in the*
4 *instance object* by a name other than the numeric memory location of the *actual field data in the*
5 *instance object*.” [Google Mot. for JMOL at 3 (emphases indicating Dr. Mitchell’s apparent
6 changes to Court’s construction).] This does not conform to the Court’s ruling on claim
7 construction, which simply distinguishes between a reference to data—any data—by a name
8 rather than location. [See Claim Construction Order (Dkt. No. 137) at 20-22; RT 3480:12-15
9 (Mitchell).]

10 Oracle misplaces much emphasis on the fact that the figures in the ’104 patent show that
11 the symbolic reference refers to data that is “obtained.” [Oracle Mot. for JMOL at 5.] It further
12 emphasizes that the experts agree that the purpose of the LOAD instruction described in the
13 patent is to obtain the value from the data object. [*Id.*] But there are two problems with these
14 arguments. First, the fact that an instruction loads a particular piece of data does not mean that
15 the intervening data structures—in the case of Android, the constant pool tables—do not also
16 qualify as data. It’s all data, as explained by Dr. August on cross-examination. [3955:22-24 (“Q.
17 Now, just to clean up a few other things. You labeled this ‘data,’ but this is actually the constant
18 pool; true, sir? A. Constant pool is data.”).] The simple figures of the ’104 patent, which contain
19 only instructions and the ultimate data object, do not prove otherwise. Put another way, the fact
20 that Figures 1A, 1B, and 8 in the ’104 patent reflect entries in a “Data Object” does not mean that
21 other data cannot also meet the construction of “symbolic references.” Indeed, if this is the only
22 “data” that the claim construction refers to, then every other reference in the system must be
23 deemed “symbolic.” Then Oracle’s interpretation becomes a self-fulfilling prophecy: by
24 identifying a single piece of data as “the data” for purposes of the claim construction, Oracle’s
25 interpretation means that every other piece of data is a symbolic reference. This is simply not
26 consistent with the patent or the Claim Construction Order. [See TX 4015; Dkt. No. 137 at 20-
27 22.]

28

1 Second, as Dr. August explained in his testimony, the actual field data in the instance
2 object may itself be a reference to other data; for example, another string in the string data table.

3 Q. What can be in this instance object by way of data?

4 A. Well, it could be a symbolic reference. It could be a numeric
5 reference. Or it could be other non-reference data.

6 ***

7 Q. Let's talk about in Dalvik. Is the data at these entries always
8 non-reference data, as we talked about in your technical
9 tutorial?

10 A. No, it's not.

11 Q. How do you know that?

12 A. Because I write programs. And you can write programs that
13 contain references in instances of objects.

14 [RT 4002:14-16; 4003:11-17.] Thus Oracle's characterization of the value of the actual field data
15 in the instance object as some form of "ultimate" or "meaningful" data in Android is a false
16 premise, unsupported by any evidence.

17 By claiming that data in the constant pool tables does not count, and that only the value of
18 the actual field data in an instance object qualifies as the data, Oracle's motion turns the claim
19 construction on its head. [Oracle Mot. for JMOL at 6 ("The Field ID table and the other parts of
20 the dex file constant pool information are not 'data' within the meaning of the claims.").] As
21 discussed above, Oracle's new argument would mean that all references in the instruction stream
22 qualify as symbolic references, because, despite the fact that those references are indexes that
23 point to the location of data (in constant pool tables), they ultimately lead to the loading of other
24 data once the symbolic reference—the name in the string table—is determined and thereafter
25 resolved. This is an illogical reading of the '104 patent, which clearly provides for an instruction
26 stream that contains *both* symbolic and numeric references as separate and distinct elements.
27 [See, e.g., TX 4015 at Claim 11.]
28

1 **3. Indexes are not symbolic references because they are not resolved.**

2 Oracle contends that field indices are resolved to pointers; therefore, they must be
3 symbolic references. [Oracle Mot. for JMOL at 7.] Oracle has it backwards. Field indices are
4 *not* resolved. As Mr. McFadden explained in the context of discussing Resolve.c:

5 Q. And are indexes in this process being resolved?

6 A. No.

7 Q. Why do you say no?

8 A. Well, resolution implies something is unknown, something is
9 ambiguous. If you have an index, you know exactly where
10 you're going. You have the location.

11 For symbols, you don't know where you're going and you
12 won't know until you have resolved the ambiguity. So it
13 doesn't really make sense to say that you "resolve an index."

14 [RT 3650:18-3651:1.] The record confirms that indexes are numeric rather than symbolic
15 references.

16 Indexes in Dalvik bytecode instructions always point to the numeric memory location of
17 data, both before and after resolution of the symbolic references in the .dex file string data. In
18 Resolve.c, the instruction stream does not change; the index field@CCCC is used both before and
19 after resolution. It is simply used to look at entries in the Resolved Fields Table and the Field ID
20 table. [RT 3636:13-3637:20 (McFadden).] In dexopt, although the bytecode instruction operand
21 is rewritten, it is simply changed from one numeric reference to another numeric reference. [RT
22 3746:22-3747:14 (McFadden); 3933:5-3934:19 (August).] Specifically, the index value
23 "field@CCCC" is replaced with another numeric memory location—the offset
24 "fieldoff@CCCC." [RT 3746:22-3747:14 (McFadden); 3933:5-3934:19 (August); TX 737 at 3;
25 TX 46.106 (showing instruction format kFmt22c uses the data at location "field@CCCC" and
26 instruction format kFmt 22cs uses the data at location "field offset CCCC"); see also TX 737 at 1
27 ("Suggested static linking formats have an additional 's' suffix, . . .").] This is made clear in the
28 dexopt documentation, which notes that dexopt is simply replacing one location with another:
"For virtual method calls, replace the method index with a vtable index" and "For instance field
get/put, replace the field index with a byte offset." [TX 739.] In sum, either the index in the

1 instruction stream stays the same, as in the case of *Resolve.c*, or it gets rewritten by an index or
2 offset to another location.

3 True symbolic references—names in the string table—require real resolution. These types
4 of references do not tell the computer where to go next; they require a search, i.e., a “resolution.”
5 [RT 3646:1-23, 3766:18-3767:4 (McFadden); 3848:20-3849:17 (August).] In contrast, indexes
6 do not require any searching (or “resolving”) because they tell the computer exactly which
7 location in memory to access next.

8
9 THE COURT: The premise being that the certain of said instructions containing
one or more symbolic references.

10 And you’re saying that -- well, what is “01”? Is that, in your view, a symbolic
11 reference?

12 THE WITNESS: No. “01” is a numeric reference because it gives you an actual
numeric memory location. Right here (indicating). This is the location, 01.

13 There’s no resolution, no search. Nothing – nothing expensive about figuring out
14 what that instruction is referring to when it goes to the field ID table.

15 [RT 3865:11-20 (August).] This is precisely the distinction recognized in the Court’s Claim
16 Construction Order. [Dkt. No. 137 at 21 (noting that for symbolic references “[t]he claimed
17 invention would use a dynamic subroutine to interpret this symbolic reference—it would have to
18 figure out that ‘y’ means ‘17’ or that ‘y’ means ‘the data stored in memory slot 2,’ and *then* get
19 the data called y (col. 5:13–19).”)]

20 **B. Dexopt does not infringe the ’104 patent.**

21 Oracle’s request for judgment as a matter of law on infringement of claims 27 and 29
22 should also be denied. Oracle’s claim that the Android dexopt tool resolves symbolic references
23 dynamically rather than statically rests on Dr. Mitchell’s erroneous opinion that an optimization
24 (e.g., resolution) based on information *about* a runtime environment is “dynamic.” But there is a
25 wealth of documentary evidence and fact and expert testimony testimony that proves dexopt is a
26 static optimization. [RT 3730:16-22 (McFadden); RT 3940:17-20 (August); RT 3595:21-24
27 (Bornstein); TX 32 at 35, TX 816 at 24:05; TX 735 (defining opcodes that “are reasonable
28 candidates for static linking”); TX 737 (defining “statically linked” instruction formats); TX 739

1 (dexopt performs optimizations that “can be inferred statically”).] Meanwhile, Oracle’s reliance
 2 on TX 1094 is specious. As is clear from the document, it was an incomplete cut-and-paste job
 3 from TX 739 (also TX 105). And the copied reference to runtime information, as explained by
 4 Andy McFadden, does not refer to optimization performed by dexopt. [See RT 3734:20-3735:7.]
 5 Accordingly, no reasonable jury could conclude that dexopt infringes claims 27 and 29 of the
 6 ’104 patent.

7
 8 **C. The Court should reject Oracle’s post-trial attempt to re-construe the term
 “symbolic reference.”**

9 Oracle seeks judgment as a matter of law on infringement of claims 27 and 29 of the ‘104
 10 patent based on yet another request that the Court’s construction of “symbolic reference” be re-
 11 construed. [Dkt. 1168 at 9-11 (Section III.C).] This latest attempt to change the meaning of a
 12 claim construction that the parties have relied on for over a year and through the close of evidence
 13 at trial represents Oracle’s third bite. [See Dkt. 132, 1128.] Google has previously explained
 14 why this request should be rejected as both highly prejudicial and as waived by Oracle. [See Dkt.
 15 1134 at 1-2 (incorporated by reference herein).] In addition, there is no basis to conclude, as
 16 Oracle has, that the term “dynamic” is given a special meaning in the ’104 patent. Indeed, the
 17 record at trial proves just the opposite.

18 For example, Oracle’s expert commended the Court’s construction of “symbolic
 19 reference,” stating that “this is a *great, clear statement* from the Court about what symbolic
 20 reference means.” [RT 3302:23-3303:1 (Mitchell) (emphasis added).] Dr. Mitchell also testified
 21 that “dynamic” resolution, as used in the Court’s construction of symbolic reference, is resolution
 22 that is done at runtime:

23 Q. So resolution from -- if there were a symbolic reference that was
 24 resolved to a numeric reference, even if that were so, you would also
 need to show that that was done dynamically; correct?

25 A. Yes.

26 Q. And you said that that means done at runtime essentially, right?

27 A. Yes, that’s one way that it can be done dynamically.

28 [RT 3502:25-3503:7.] In light of this clear testimony from Oracle’s own expert, Oracle cannot

1 now be heard to complain that “dynamic,” as that term is used in the context of the Court’s
2 definition of “symbolic reference,” is ambiguous.

3 Further, Oracle’s request, while couched as seeking clarification of the meaning of the
4 term dynamic in the context of the ’104 patent, is nothing more than another attempt to eliminate
5 the “dynamic” nature of the symbolic resolution disclosed and claimed by the ’104 patent. Oracle
6 posits that the ’104 patents requirement of dynamic resolution is satisfied when symbolic
7 references are “resolved to identify the memory location of the underlying data based on memory
8 conditions that exist *at whatever time the resolution occurs.*” [Dkt. 1168 at 11 (emphasis added).]
9 By definition, symbolic references must be resolved; thus, Oracle’s definition of dynamic would
10 apply to every symbolic reference, because resolution must happen at some time.

11 The ’104 patent is clear, however, that the claimed symbolic references are resolved
12 during execution—i.e., at runtime. For example, the ’104 patent discloses that the main
13 interpretation routine determines if the reference is symbolic, invokes the dynamic field reference
14 routine that resolves the symbolic reference, and then re-executes the instruction. [TX 4015 at
15 5:10-23.] This passage clearly defines the dynamic nature of the ’104 patent’s symbolic
16 resolution. It is performed by the interpretation routine—i.e., while interpreting the instruction
17 containing a symbolic reference. After this runtime resolution occurs, the resolved instruction is
18 re-executed. Indeed, the fact that the patent describes the second pass as “reexecution” confirms
19 that the symbolic reference resolution occurs during execution.

20 Given the testimony of Oracle’s own expert and the clear import of the ’104 patent
21 disclosure as captured in the Court’s construction of “symbolic reference,” Oracle has provided
22 no basis for its belated request to redo the claim construction. More importantly, applying the
23 current construction, no reasonable jury could find that claims 27 and 29 are infringed by dexopt.

24 **III. A reasonable jury could only find that Google does not infringe the asserted claims**
25 **of the ’520 patent.**

26 Oracle asks for judgment as a matter of law of infringement based on its argument that
27 “simulating execution” does not require any actual operations on a Java virtual machine to be
28 simulated, and that the term also encompasses the distinct method (which results in different

1 outcomes in certain circumstances) of pattern matching. Neither of these contentions are
2 supportable, and Oracle’s arguments should be rejected.

3
4 **A. The jury is entitled to credit Professor Parr’s testimony explaining why
pattern matching differs from simulating execution**

5 To reach his opinion on non-infringement, Professor Parr not only looked at the source
6 code, but also performed multiple experiments to confirm his understanding. His second
7 experiment showed beyond any doubt that the dx tool is not simulating execution as required by
8 the ’520 patent. He inserted an extra piece of bytecode that put a zero into the zeroth entry in an
9 array—where a zero already existed at the inception of the array. Such an extraneous piece of
10 code, when run through a program that simulated execution of byte code instructions, would
11 create an efficient instruction with no problem. [RT 3810:4-13 (Parr).] However, the dx tool
12 failed to create a proper static array initialization instruction when it was fed that code. [RT
13 3809:20-3810:3 (Parr).] As Professor Parr explained, the code was set up to “punt” when it did
14 not find the pattern it sought. [RT3801:7-18 (Parr)]. Hence, Professor Parr concluded that the dx
15 tool code could not be simulating execution of the byte code. Dr. Mitchell’s so-called
16 “experiment,” referenced in Oracle’s JMOL brief at 14:23-15:2, in no way contradicts or even
17 addresses Professor Parr’s findings. Dr. Mitchell’s work consisted of running the dx tool on a
18 standard Java byte code pattern and confirming that it created a single instruction to initialize a
19 static array. [RT 3345:8-3346:1 (Mitchell).] But the fact that the byte code started in one form
20 and ended in another does not tell how it got there; and Dr. Mitchell admitted that, for a method
21 claim, it is not enough simply to show that “with the same input you’d get the same output from a
22 piece of code.” [RT 4054:24-4055:4 (Mitchell).] Accordingly, Dr. Mitchell’s experiment is
23 insufficient to show that pattern matching infringes the ’520 patent.

24 Oracle’s contention that simulated execution includes the far different process of pattern
25 matching because the claims “do not exclude pattern matching” is incorrect. The burden is on
26 Oracle to show infringement of the claims as properly construed—not on Google to prove that the
27 claims do not cover what the dx tool does. Oracle never requested construction of “simulating
28 execution” in the ’520 patent; its position was that the phrase takes its plain meaning. Yet

1 nothing in the plain meaning of that phrase shows that pattern matching is covered. Indeed, it is
2 undisputed that the '520 patent makes no mention of pattern matching. [RT 3521:10-12
3 (Mitchell); TX 4011.] Oracle continues to highlight the source code comments from the class
4 simulator.java in an attempt to prove its case. But both experts agree that the code that identifies
5 the static initialization of arrays is *not* found in that class; it is in the separate BytecodeArray
6 class. [RT 3799:14-3800:7 (Parr); 3519:19-3520:3 (Mitchell); 4061:9-4062:3 (Mitchell).] And
7 BytecodeArray, in which Oracle concedes both the parseInstruction and parseNewarray methods
8 are found, Mot. at 14:17-19, is the class that identifies static initialization through pattern
9 matching rather than through simulated execution. [RT 3799:14-3801:18 (Parr).] Oracle cannot
10 show that, as a matter of law, pattern matching must be encompassed within “simulating
11 execution.” At most, this is a doctrine of equivalents argument that Oracle never made, and
12 Google (not Oracle) is entitled to judgment on it. [See Brief in Support of Google’s Motion for
13 Judgment as a Matter of Law on Counts V and VII of Oracle’s Amended Complaint (ECF No.
14 1166) at 8:21-9:25.]

15
16 **B. Simulating execution, properly understood, involves the simulation of actual
Java virtual machine operations.**

17 At trial, Professor Parr, who has been programming in Java for nearly two decades now,
18 explained that “there is no meaningful definition of simulating execution of a stack machine
19 without manipulation of a stack. That means pushing, popping, things like that.” [RT 3794:17-
20 19 (Parr).] Oracle’s own expert, Dr. Mitchell, did not disagree—far from it. He agreed that “in a
21 Java bytecode system, instructions operate by pushing and popping and replacing values from the
22 top of an operand stack.” [RT 4058:3-6 (Mitchell).] Google is not attempting to read language
23 from the specification into the claims—it is explaining how the claim language would be
24 understood by those of skill in the art in light of the specification, which is to require simulating
25 execution of the actions set forth in the byte code instructions in order to obtain the initial values
26 of the static array. [TX 4011 at Fig. 3, 5:52-54.]

27 In contrast to Professor Parr’s thoughtful explanation of what the relevant claim language
28 would mean to one of ordinary skill in the art, Dr. Mitchell seldom used the actual language of

1 “simulating execution” in his testimony, eschewing that phrase for the use of the more general
 2 word “simulation.” [See, e.g., RT 4032 (“is properly called simulation or not”; “dx tool simulates
 3 bytecode”; “pattern matching as a portion of simulation”; “the process is properly called
 4 simulation”) (Mitchell).] Indeed, Oracle’s counsel repeatedly uses other words in its JMOL
 5 motion to describe what the dx tool does. [Oracle Mot. for JMOL at 12:11-12 (“the dx tool
 6 examines the bytecodes”); *id.* at 14:20 (“bytecode instructions are examined without being
 7 executed”); *id.* at 15:5-7 (dx tool “examining” the byte codes) (emphases added).] Yet the claim
 8 language of the ’520 patent is clear that what is required is not “simulating” or “examining,” but
 9 rather “simulating execution.” TX 4011 at 9:54-57, 12:7-8. Given that Oracle itself repeatedly
 10 describes what the dx tool does in terms other than simulating execution, a reasonable jury plainly
 11 could find that the claim language is not satisfied by the dx tool.¹

12 **IV. Oracle’s patent infringement claims are barred by Google’s equitable defenses.**

13 For the reasons stated in Google’s Proposed Findings of Fact and Conclusions of Law
 14 submitted after Phase I (Dkt. 1047), Google, not Oracle, is entitled to judgment on its equitable
 15 defenses.² Just as it did in post-trial briefing following Phase I, Oracle’s Motion simply recites its
 16 own gloss on disputed evidence and utterly fails to address the evidence favoring Google. Oracle
 17 also argues that “additional facts” elicited in Phase II support its made-for-litigation story that
 18 both Sun and Google always believed that Google needed a license for Android—a license that
 19 Oracle now opportunistically claims in this phase was a primarily about patents, not copyrights.

20 ¹ Oracle also trots out a claim differentiation argument that was neither found in Dr. Mitchell’s
 21 expert report, nor testified to by any witness at trial. It is therefore inappropriate to make the
 22 argument on this record and at this point in the case. However, had Oracle attempted to make the
 23 argument at trial, there is a simple response: Unasserted claim 3 of the ’520 patent requires that
 24 three separate steps (allocating a stack, reading a bytecode that manipulates the stack, and
 25 performing stack manipulation on the stack) be included within the third step (the play executing
 26 step) of claim 1. A method that allocated a stack as part of the second step of claim 1 (“receiving
 27 the class into a preloader”) would not infringe claim 3, as it would not have the stack allocation
 28 step as part of the play executing step, but it would still infringe claim 1. Thus, even if claim
 differentiation were an iron-clad rule (and it is not, *see Eon-Net LP v. Flagstar Bancorp*, 653 F.3d
 1314, 1323 (Fed. Cir. 2011)), claim differentiation does not mean that claim 1 does not include
 stack manipulation.

² The jury already has found that Google proved that “Sun and/or Oracle engaged in conduct Sun
 and/or Oracle knew or should have known would reasonably lead Google to believe that it would
 not need a license to use the structure, sequence, and organization of the copyrighted compilable
 code.” Dkt. 1089 (Answer to Special Interrogatory 4A).

1 [Oracle Mot. for JMOL at 16.] Oracle is wrong. Indeed, in most instances, Oracle’s gloss on the
2 evidence is expressly contradicted by other evidence in the record—evidence it pretends doesn’t
3 exist. Oracle’s motion should be denied.

4 **A. Oracle’s “additional facts” are contradicted by the evidence**

5 In Phase I, the evidence established that (1) Sun was well aware of Google’s plans for
6 Android at least as early as April 2006, (2) Sun publicly encouraged Android after it was
7 announced and the SDK was released in November 2007, (3) Sun never once told Google that
8 Android infringed Sun’s copyrights or patents, and (4) Google relied on Sun’s active
9 encouragement and failure to assert its rights by investing time, money, and resources into
10 Android. [See GFOF 37-92.] This evidence is equally relevant to Phase II, but Oracle ignores it.
11 Instead, Oracle tries to make hay out of “additional facts” elicited in Phase II. Oracle’s “facts”
12 are not facts at all; they are attorney argument, and they are contradicted by the evidence.

13 For example, Oracle states “[o]ne of Google’s primary goals in these negotiations was to
14 obtain a license to Sun’s patents.” [Oracle Mot. for JMOL at 16:25-26 (citing TX 2714, TX 22,
15 and TX 618).] But Andy Rubin, the head of the Android project at Google, testified that the
16 documents Oracle cites in support of this “fact” addressed Google’s desire to develop an open
17 source license with patent protection for downstream licensees, not a desire to license specific
18 patents from Sun. [See, e.g., RT 3190:25-3192:21.] Vineet Gupta, Sun’s lead negotiator during
19 the Sun-Google partnership discussions, confirmed that Sun and Google did not specifically
20 negotiate for a patent license, and that he never presented any Java-related patents to Google
21 during the negotiations. [RT 3771:21-24; TX 3542.] Nothing in the Phase II (or Phase I)
22 evidence supports Oracle’s position that Sun and Google were negotiating for a license to specific
23 Java-related patents.

24 Oracle also states as an “additional fact” that Tim Lindholm “was aware of the ‘104 patent
25 in particular.” [Oracle Mot. for JMOL at 17.] This is plainly incorrect. Mr. Lindholm testified
26 that he had never even read the ‘685 patent, let alone the ‘104 patent, which has entirely different
27
28

1 claims than the '685 patent. [RT 3023:9-13; TX 4015.³] Nor is there any evidence that any
2 member of the Android team had ever seen either of the two patents-in-suit. Oracle's vague
3 assertions about Google's awareness of unidentified "Sun patents," Mot. at 17, is irrelevant to
4 Google's equitable defenses.

5 In short, none of the "additional facts" cited by Oracle in its Motion overcome the
6 overwhelming evidence in support of Google's equitable defenses. [See GFOF at 37-92.]

7 **B. Oracle's patent infringement claims are barred by equitable estoppel.**

8 For the reasons explained in Google's Proposed Findings of Fact and Conclusions of Law,
9 and its Response to Oracle's proposed findings and conclusions, Google has proven each of the
10 elements of its equitable estoppel defense. [Dkt. 1047 at GCOL 32-33, 1079 at 59-62; *see also*
11 *A.C. Aukerman Co. v. R.L. Chaides Const. Co.*, 960 F.2d 1020, 1042 (Fed. Cir. 1992) (listing
12 elements).] Rather than address these facts, Oracle's motion makes much of the fact that Sun
13 and Google discussed Android-related opportunities after Google launched Android. [Oracle
14 Mot. for JMOL at 20-21.] Not even Oracle's own gloss on those discussions supports its motion
15 here.

16 That Sun attempted to convince Google to buy a TCK license and make Android Java-
17 compatible, Mot. at 20-21, is irrelevant to Google's estoppel defense. What is relevant is that Sun
18 never informed Google that it was violating either the '104 or '520 patent, until July 20, 2010,
19 less than three weeks before the lawsuit was filed. [RT 3181:25-3182:10 (Rubin); *see also* TX
20 1074.] Indeed, Sun and Google never even discussed a license to any specific patents, let alone
21 the patents-in-suit. [RT 3190:25-3192:21 (Rubin); RT 3181:25-3182:10 (Gupta).] Discussions
22 about a potential license unrelated to the patents-in-suit are simply irrelevant to Google's
23 affirmative defense of equitable estoppels. [See Dkt. 1047 at GCOL 32-34.]

24 **C. Oracle's patent infringement claims are barred by laches.**

25 For the reasons stated in Google's Proposed Findings of Fact and Conclusions of Law,
26 and its Response to Oracle's proposed findings and conclusions, Google has proven each element

27 ³ In any event, Mr. Lindholm was not involved at any point with the design or development of the
28 Dalvik virtual machine specifically or even Android generally. [RT 3033:22-3035:1.]

1 of laches. [Dkt. 1047 at 31-32, 1079 at 64-66; *see also Aukerman*, 960 F.2d at 1032.]

2 That Sun occasionally tried to convince Google to buy a license for Android—but did not
3 identify or even allude to the patents-in-suit—does not excuse Oracle’s delay in filing suit. Each
4 of the cases Oracle cites for the proposition that negotiations are an excuse for unreasonable delay
5 involve negotiations focused on the specific patents at issue. *See Lucent Techs., Inc. v. Gateway,*
6 *Inc.*, 580 F. Supp. 2d 1016, 1053 (S.D. Cal. 2008) *aff’d in part, vacated in part on other grounds,*
7 *remanded*, 580 F.3d 1301 (Fed. Cir. 2009) (“The evidence showed that for several years leading
8 up to the start of the litigation underlying this case, beginning around 1998 through 2002 and
9 2003, Lucent engaged in efforts to sell a license for the ’226 to computer manufacturers Gateway
10 and Dell.”); *In re Katz Interactive Call Processing Patent Litig.*, 712 F. Supp. 2d 1080, 1110
11 (C.D. Cal. 2010) (patentee contacted alleged infringer “regarding its patent portfolio,” patentee
12 “asked for additional information regarding any potential infringement,” and alleged infringer
13 considered taking a license for the specific patent portfolio). In this case, however, the record is
14 clear that Sun and Google never negotiated over the patents-in-suit, or any specific patents for
15 that matter. [RT 3190:25-3192:21 (Rubin); RT 3181:25-3182:10 (Gupta).]

16 Oracle also ignores Mr. Rubin’s testimony that, during the time of Sun’s and Oracle’s
17 delay in filing suit, Google invested in Android by hiring engineers, creating Google applications
18 for Android, and spending time and money to help Google and its partners bring phones to
19 market. [RT 1715:2-1717:25; Dkt. 1047 at GFOF 82-86.] Oracle cites no case suggesting that
20 the “prejudice” requirement for laches cannot be met where the defendant invests time, money,
21 and resources in a still-developing product in reliance upon the plaintiff’s delay and at a time
22 when the defendant could have changed course had the plaintiff filed suit earlier.

23 **D. Oracle’s patent infringement claims are barred by waiver.**

24 For the reasons stated in Google’s Proposed Findings of Fact and Conclusions of Law,
25 and its Response to Oracle’s proposed findings and conclusions, Google has proven that Sun
26 waived its rights to enforce the ’104 and ’520 patents. [Dkt. 1047 at GCOL 35, 1079 at 56-
27 58.] For the reasons explained above, Oracle’s desire to sell Google a license does not contradict
28 the evidence establishing that Oracle waived its right to pursue legal action against Google.

1 **E. Oracle’s patent infringement claims are barred by implied license**

2 For the reasons stated in Google’s Proposed Findings of Fact and Conclusions of Law,
3 and its Response to Oracle’s proposed findings and conclusions, Google has proven that the entire
4 course of conduct between Sun and Google over the relevant time period created an implied
5 license for Google to use the ’104 and ’520 patents in Android. [Dkt. 1047 at GCOL 34-35, 1079
6 at 62-63.] Oracle’s attempt to limit the doctrine of implied license to a specific factual situation
7 unlike the situation here is unavailing.

8 Oracle cites two cases that address a specific situation in which an implied license can
9 arise. [Oracle Mot. for JMOL at 23-24 (citing *Zenith Elecs. Corp. v. PDI Comm’n Sys., Inc.*, 522
10 F.3d 1348, 1360 (Fed. Cir. 2008); *Jacobs v. Nintendo of Am, Inc.*, 370 F.3d 1097, 1100 (Fed. Cir.
11 2004)).] But these cases are not the *only* situations where an implied license can arise. Indeed,
12 the Federal Circuit, relying on *De Forest Radio Tel. Co. v. United States*, 273 U.S. 236 (1927),
13 has recognized there are “different categories of conduct” which can give rise to implied license:

14 Since *De Forest*, this court and others have attempted to identify and isolate
15 *various avenues to an implied license*. As a result, courts and commentators relate
16 that implied licenses arise by acquiescence, by conduct, by equitable estoppel
17 (estoppel in pais), or by legal estoppel. These labels describe not different kinds of
licenses, but rather *different categories of conduct which lead to the same*
conclusion: an implied license.

18 *Wang Labs., Inc. v. Mitsubishi Elecs. Am., Inc.*, 103 F.3d 1571, 1580 (Fed. Cir. 1997) (internal
19 citations omitted) (emphases added).

20 Contrary to Oracle’s attempt to limit the doctrine, the general test for whether there is an
21 implied license is whether “the entire course of conduct between a patent or trademark owner and
22 an accused infringer” creates such an implied license. *McCoy v. Mitsuboshi Cutlery, Inc.*, 67 F.3d
23 917, 920 (Fed. Cir. 1995). The evidence here supports a finding of implied license.

24 **V. Google chose not to present its defenses of patent misuse, use by the United States,
25 unclean hands, and express license, and therefore does not oppose Oracle’s motion
on these defenses.**

26 In Section VI of its Motion, Oracle moves for judgment as a matter of law as to Google's
27 Sixth (patent misuse), Eighth (use by the U.S.) and Nineteenth (unclean hands) defenses, as well
28 as Google's defense of express license. Although Google believes these defenses have merit,

1 Google chose not to present these defenses at trial. Google therefore does not oppose Section VI
2 of Oracle's Motion.

3 **VI. Conclusion**

4 For the foregoing reasons, Google requests that the Court deny Oracle's Rule 50 Motion
5 at the close of all the Phase II evidence.

6 Dated: May 17, 2012

KEKER & VAN NEST LLP

7
8 By: /s/ Robert A. Van Nest
ROBERT A. VAN NEST

9 Attorneys for Defendant
10 GOOGLE INC.

11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28