

1 MORRISON & FOERSTER LLP
 MICHAEL A. JACOBS (Bar No. 111664)
 2 mjacobs@mofo.com
 KENNETH A. KUWAYTI (Bar No. 145384)
 3 kkuwayti@mofo.com
 MARC DAVID PETERS (Bar No. 211725)
 4 mdpeters@mofo.com
 DANIEL P. MUINO (Bar No. 209624)
 5 dmuino@mofo.com
 755 Page Mill Road, Palo Alto, CA 94304-1018
 6 Telephone: (650) 813-5600 / Facsimile: (650) 494-0792

7 BOIES, SCHILLER & FLEXNER LLP
 DAVID BOIES (Admitted *Pro Hac Vice*)
 8 dboies@bsfllp.com
 333 Main Street, Armonk, NY 10504
 9 Telephone: (914) 749-8200 / Facsimile: (914) 749-8300
 STEVEN C. HOLTZMAN (Bar No. 144177)
 10 sholtzman@bsfllp.com
 1999 Harrison St., Suite 900, Oakland, CA 94612
 11 Telephone: (510) 874-1000 / Facsimile: (510) 874-1460

12 ORACLE CORPORATION
 DORIAN DALEY (Bar No. 129049)
 13 dorian.daley@oracle.com
 DEBORAH K. MILLER (Bar No. 95527)
 14 deborah.miller@oracle.com
 MATTHEW M. SARBORARIA (Bar No. 211600)
 15 matthew.sarboraria@oracle.com
 500 Oracle Parkway, Redwood City, CA 94065
 16 Telephone: (650) 506-5200 / Facsimile: (650) 506-7114

17 *Attorneys for Plaintiff*
 ORACLE AMERICA, INC.

19 UNITED STATES DISTRICT COURT
 20 NORTHERN DISTRICT OF CALIFORNIA
 21 SAN FRANCISCO DIVISION

22 ORACLE AMERICA, INC.
 23 Plaintiff,
 24 v.
 25 GOOGLE INC.
 26 Defendant.

Case No. CV 10-03561 WHA
**ORACLE AMERICA, INC.'S
 RULE 50(A) MOTION AT THE
 CLOSE OF ALL EVIDENCE FOR
 PHASE II (PATENT PHASE)**
 Dept.: Courtroom 8, 19th Floor
 Judge: Honorable William H. Alsup

TABLE OF CONTENTS

	Page
I. INTRODUCTION	1
II. LEGAL STANDARD FOR JUDGMENT AS A MATTER OF LAW.....	1
III. NO REASONABLE JURY COULD FIND THAT GOOGLE DID NOT INFRINGE THE ASSERTED CLAIMS OF THE '104 PATENT	2
A. Android's Resolve.c infringes claims 11, 39, 40, and 41 of the '104 patent because Dalvik bytecode instructions contain symbolic references	2
1. A field index is a symbolic reference that is contained in a Dalvik bytecode instruction	2
2. The data that is obtained is the "data" that determines whether a reference is symbolic or numeric, not the constant pool information used to perform symbolic reference resolution.....	5
3. Conversion of instruction stream indices to numeric memory locations confirms that the indices are symbolic reference— numeric references are not resolved.....	7
B. Android dexopt infringes claims 27 and 29 of the '104 patent.....	8
C. Under the correct claim construction, dexopt resolves symbolic references "dynamically rather than statically"	9
IV. NO REASONABLE JURY COULD FIND THAT GOOGLE DID NOT INFRINGE THE ASSERTED CLAIMS OF THE '520 PATENT	12
A. '520 Patent Background.....	12
B. Android's dx Tool Simulates Execution Of Bytecodes To Identify The Static Initialization Of Arrays	13
1. Dr. Mitchell's Analysis	13
2. Dr. Parr Sought To Read Additional Limitations Into The Claim Language	15
V. GOOGLE'S EQUITABLE DEFENSES FAIL.....	16
A. Additional Facts Relevant to Google's Equitable Defenses	16
B. Google Has Not Shown that Equitable Estoppel Bars Oracle's Patent Infringement Claims.....	20
C. Google Has Not Shown that the Doctrine of Laches Applies to Oracle's Patent Infringement Claims.....	21
D. Google Has Not Shown that Oracle or Sun Waived Its Right to Assert Patent Infringement Claims.....	22

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 CONTENTS
(continued)

	Page
E. Google Has Not Shown that Oracle or Sun Gave It an Implied License to Use Oracle’s Patents	23
VI. ALTERNATIVE GOOGLE DEFENSES THAT GOOGLE PLED BUT DID NOT PRESENT TO THE JURY FAIL	24
A. Google Has Not Shown Patent Misuse, Use by the United States, or Unclean Hands	24
B. Google Has Not Shown that Oracle or Sun Gave It an Express License to Use Oracle’s Patents	24
VII. CONCLUSION	24

TABLE OF AUTHORITIES

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

Page(s)

CASES

A.C. Aukerman Co. v. R.L. Chaides Const. Co.,
960 F.2d 1020 (Fed. Cir. 1992) (*en banc*)..... 21, 22

Adidas-Am., Inc. v. Payless Shoesource, Inc.,
546 F. Supp. 2d 1029 (D. Or. 2008) 22

Carborundum Co. v. Molten Metal Equip. Innovations,
72 F.3d 872 (Fed. Cir. 1995)..... 24

Carpet Seaming Tape Licensing Corp. v. Best Seam, Inc.,
694 F.2d 570 (9th Cir. 1982)..... 21

Dream Games of Ariz., Inc. v. PC Onsite,
561 F.3d 983 (9th Cir. 2009)..... 24

Forrett v. Richardson,
112 F.3d 416 (9th Cir. 1997)..... 1

Gasser Chair Co. v. Infanti Chair Mfg. Corp.,
60 F.3d 770 (Fed. Cir. 1995)..... 20

Geo M. Martin Co. v. Alliance Mach. Sys. Int’l LLC,
618 F.3d 1294 (Fed. Cir. 2010)..... 2

In re Katz Interactive Call Processing Patent Litig.,
712 F. Supp. 2d 1080 (C.D. Cal. 2010) 22

Jacobs v. Nintendo of Am., Inc.,
370 F.3d 1097 (Fed. Cir. 2004)..... 23

Lucent Techs., Inc. v. Gateway, Inc.,
580 F. Supp. 2d 1016 (S.D. Cal. 2008) 22

Qualcomm Inc. v. Broadcom Corp.,
548 F.3d 1004 (Fed. Cir. 2008)..... 22, 23

U.S. v. Amwest Surety Ins. Co.,
54 F.3d 601 (9th Cir.1995)..... 23

Wang Labs., Inc. v. Mitsubishi Elecs. Am., Inc.,
103 F.3d 1571 (Fed. Cir. 1997)..... 23

White v. Ford Motor Co.,
312 F.3d 998 (9th Cir. 2002)..... 1, 2

1 *Winn v. Opryland Music Group, Inc.*,
2 22 Fed. Appx. 728 (9th Cir. 2001) 22

3 *Zenith Elecs. Corp. v. PDI Commc'ns Sys.*,
4 522 F.3d 1348 (Fed. Cir. 2008)..... 23

5 **STATUTES**

6 Fed. R. Civ. P. 50(a)(1) 1

7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

1 **I. INTRODUCTION**

2 Google infringes Claims 11, 27, 29, 39, 40, and 41 of United States Patent No. RE38,104
3 (“the ’104 patent”) and Claims 1 and 20 of United States Patent No. 6,061,520 (“the ’520
4 patent”). Google infringes the ’104 patent in two ways: the Resolve.c resolution functions that
5 are part of the Dalvik Virtual Machine infringe Claims 11, 39, 40, and 41 and the dexopt tool that
6 is also part of the Dalvik VM infringes Claims 27 and 29. Google infringes the ’520 through the
7 operation of the dx tool, which is part of the Android SDK used by developers. Given the
8 evidence in the record, Google’s patent infringement can be determined as a matter of law, as no
9 reasonable jury could find for Google.¹

10 Google’s indirect infringement can be determined as a matter of law along with direct
11 infringement. The parties stipulated that indirect infringement (induced and contributory
12 infringement) shall follow from a finding of direct infringement and need not be submitted to the
13 jury. ECF No. 1139. In finding direct infringement as a matter of law, the Court by extension
14 should also find indirect infringement by Google.

15 Google failed to meet its burden of proving its equitable defenses and either failed to
16 preserve or present evidence on other affirmative defenses.

17 For these reasons, Oracle is entitled to judgment as a matter of law.

18 **II. LEGAL STANDARD FOR JUDGMENT AS A MATTER OF LAW**

19 Judgment as a matter of law is appropriate when “a party has been fully heard on an issue
20 during a jury trial and the court finds that a reasonable jury would not have a legally sufficient
21 evidentiary basis to find for the party on that issue.” Fed. R. Civ. P. 50(a)(1). In the Ninth
22 Circuit, “[t]he test is whether ‘the evidence, construed in the light most favorable to the
23 nonmoving party, permits only one reasonable conclusion, and that conclusion is contrary to that
24 of the jury.’” *White v. Ford Motor Co.*, 312 F.3d 998, 1010 (9th Cir. 2002) (quoting *Forrett v.*
25 *Richardson*, 112 F.3d 416, 419 (9th Cir. 1997)). The Federal Circuit applies the law of the
26

27 ¹ Depending on the type of claim, Google directly infringes by installing code on devices, running
28 the devices, or developing applications. Because these elements of the claims were not disputed,
Oracle does not address this aspect of direct infringement.

1 regional circuit when reviewing a district court’s grant of judgment as a matter of law in patent
2 actions. *Geo M. Martin Co. v. Alliance Mach. Sys. Int’l LLC*, 618 F.3d 1294, 1300 (Fed. Cir.
3 2010) (citing *White*).

4 **III. NO REASONABLE JURY COULD FIND THAT GOOGLE DID NOT INFRINGE**
5 **THE ASSERTED CLAIMS OF THE ’104 PATENT**

6 The record evidence proves as a matter of law that (1) Android’s Resolve.c infringes
7 claims 11, 39, 40, and 41 of the ’104 patent and that (2) Android dexopt infringes claims 27 and
8 29 of the ’104 patent. No reasonable jury could find otherwise.

9 **A. Android’s Resolve.c infringes claims 11, 39, 40, and 41 of the ’104 patent**
10 **because Dalvik bytecode instructions contain symbolic references**

11 The only dispute with respect to infringement by Android’s Resolve.c is whether Dalvik
12 bytecode instructions contain “symbolic references.” RT 4106:21-22 (Jacobs); 4154:6-11 (Van
13 Nest). Android source code and documentation, as well as the experts’ and Google engineers’
14 testimony, confirm that Dalvik bytecode instructions indeed contain symbolic references: the
15 field indices and other indices that are operands in Dalvik instructions. For resolve.c, the
16 question of “dynamic” in the Court’s construction was not presented, as resolve.c indisputably
17 resolves references dynamically. Whether or not the references are *also* pointers to table entries,
18 as Google argues, is legally irrelevant.

19 **1. A field index is a symbolic reference that is contained in a Dalvik**
20 **bytecode instruction**

21 A field index in a Dalvik bytecode instruction meets the Court’s definition of “symbolic
22 reference.” The Court construed the term “symbolic reference” as “a reference that identifies data
23 by a name other than the numeric memory location of the data, and that is resolved dynamically
24 rather than statically.” ECF No. 137 at 22.

25 A field index—also called field@CCCC generally or “01” in specific examples of field
26 indices in the trial testimony—is a reference to data to be obtained in accordance with a
27 corresponding numerical reference, and identifies that data by a name other than the numeric
28 memory location of the data. RT 3228:14-3229:25 (McFadden); RT 3303:2-3304:20 (Mitchell).
In order to obtain data from the data object containing the value of the field, the Dalvik VM uses

1 the resolver functions of Resolve.c to resolve the field index to a numeric memory location that is
2 then used to obtain the value. RT 3646:24-3647:25 (McFadden); RT 3308:18-3309:24
3 (Mitchell). The Dalvik VM resolves type indices,² method indices, and string indices in much the
4 same way as field indices, and they are symbolic references for the same reason. See RT
5 3239:17-21 (McFadden); 3310:4-3311:1 (Mitchell).

6 The Dalvik bytecode instruction that was the focus of both parties' evidence and argument
7 is the IGET instruction, which corresponds to the "LOAD 'y'" instruction in the '104 patent. RT
8 3297:10-3302:2 (Mitchell); RT 3956:2-3961:6 (August). The IGET instruction (together with the
9 IPUT instruction) "performs the identified object instance field operation with the *identified field*,
10 loading or storing into the value register." TX 735 at 6, emphasis added. The IGET instruction
11 contains three operands—vA, vB, and field@CCCC—where the third operand field@CCCC is
12 the field index. TX 735 at 6; RT 3221:8-10 (McFadden). The field index in the IGET instruction
13 identifies the field—that is, it specifies the field from which the data is to be obtained by IGET.
14 Google's Mr. McFadden testified:

15 Q. Can you explain what the iget instruction is?

16 A. That is the instance field get instruction. What that means is there is an object
17 somewhere and you need to get a piece of data out of it. The data is stored in
18 fields. So what this instruction does is it finds the instance of the object and
retrieves the data from the specified field.

19 RT 3221:2-7 (McFadden), *see also* RT 3968:10-15 (August). Dr. Mitchell confirmed that the
20 resolve instance field function call takes the field index as an argument and returns the resolved
21 field, which is stored and then used by Dalvik to obtain the actual field data. RT 3311:23-
22 3312:19 (Mitchell); 3318:4-3319:13 (Mitchell).

23 For the field index in the IGET instruction to be a symbolic reference, it is enough that it
24 identifies—"specifies," in Mr. McFadden's words—*some* data to be obtained, by something other
25 than the data's location. This satisfies the Court's construction of "symbolic reference."
26
27

28 ² A class index is a kind of type index. TX 736 at 2.

1 Android engineers McFadden and Bornstein testified that the field index contained in the
2 instructions is not the numeric memory location of the value of the field. RT 3614:22-3615:16
3 (Bornstein); 3761:19-3762:6 (McFadden). In fact, “the Dalvik IGET instruction *never* contains
4 the numerical memory location of the actual field data that it is supposed to get.” *Id.* 3761:14-18
5 (McFadden) (emphasis added). Dr. August also testified that the field index was not the
6 numerical memory location of the actual field data in an object. RT 3970:20-3971:3 (August).
7 Dr. Mitchell confirmed that the field indices and other indices, which are contained in the Dalvik
8 bytecode instructions, “are names that are used when there is data as the program runs to find the
9 location of data, but they, themselves, are not the location of the program data in any sense.” RT
10 3533:21-25 (Mitchell). This testimony proved that the field index in the IGET instruction is a
11 symbolic reference to the field, because the field index is not and cannot be the numeric memory
12 location of the value of the field.

13 That indexes such as the “01” in “52 01” (the bytecode for IGET field@0001) may *also*
14 indicate the location of information in the dex file’s constant pool is *not relevant*. The claims do
15 not require that a reference *exclusively* identify data symbolically to qualify as a symbolic
16 reference. Moreover, there is no requirement in the ’104 patent or the Court’s claim construction
17 that the reference to data in a symbolic reference be direct. Indeed, by definition, a symbolic
18 reference is not the numeric memory location of the data it refers to, so it is necessarily indirect.

19 Because the indexes in question *at least* identify data by names other than numeric
20 memory locations, they qualify as symbolic references. This reasoning is confirmed by the
21 Court’s construction of the terms “resolve” and “resolving” to mean “*at least* determining the
22 numerical memory-location reference that corresponds to the symbolic reference.” (emphasis
23 added). The process of resolution may also include intermediate steps that refer to the constant
24 pool, but so long as the symbolic reference “at least” used to determine the corresponding
25 numerical memory-location, the Court’s construction of both “symbolic reference” and “resolve”
26 is met. This provides an independent basis on which JMOL of infringement should be granted.

1 **2. The data that is obtained is the “data” that determines whether a**
2 **reference is symbolic or numeric, not the constant pool information**
3 **used to perform symbolic reference resolution**

4 An additional basis on which JMOL of infringement should be granted is that the “data”
5 in the Court’s construction of symbolic reference (“a reference that identifies data by a name
6 other than the numeric memory location of the data”) is the actual field data “obtained” by
7 Dalvik—the value of a field in an “instance object”—rather than the constant pool information in
8 the Android dex file that is the focus of Google’s arguments.

9 That data from an instance object is the “data” that the claimed symbolic reference refers
10 to follows from the claim language of the ’104 patent. For this issue, Claim 11 is representative:

11 11. An apparatus comprising:

12 a memory containing intermediate form object code constituted by a set of
13 instructions, certain of said instructions containing one or more symbolic
14 references; and

15 a processor configured to execute said instructions containing one or more
16 symbolic references by determining a numerical reference corresponding to said
17 symbolic reference, storing said numerical references, and obtaining data in
18 accordance to said numerical references.

19 TX 4015, 7:5-14. Applying the Court’s construction of “symbolic reference,” the instructions
20 contain references to data that identify the data by a name other than the numeric memory
21 location of the data; the references are resolved to numeric memory locations, which are then
22 stored and used to “obtain” the data.

23 FIG. 1B of the ’104 patent (below) shows that a symbolic reference refers to the data that
24 is “obtained.” The figure illustrates the execution of the instruction “LOAD ‘y.’” In this
25 instruction, the symbolic reference “y” refers to the data in the data object. RT 3298:20-21
26 (Mitchell). “[A]n *instruction that accesses or fetches y*, such as the Load instruction 14’
27 illustrated in FIG. 1, references the variable y by the symbolic name ‘y.’” TX 4015, 1:37-39.
28 The symbolic reference *is to the data obtained*, not some other information. The experts agree
29 that the purpose of the LOAD instruction described in the patent is to obtain the value from the
30 data object. RT 3298:20-24 (Mitchell), 3960:25-3961:6 (August).

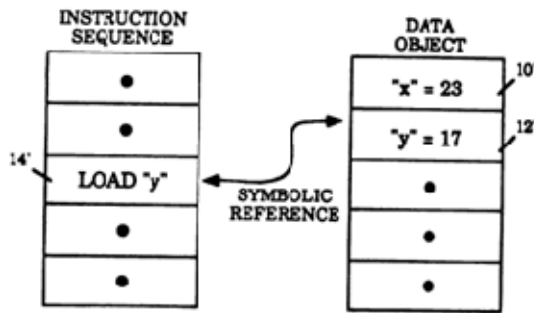


Figure 1B
Prior Art

A field index in a Dalvik IGET instruction is a symbolic reference because it identifies data in the instance object by a name (the index) other than the numeric memory location. McFadden, Bornstein, August, and Mitchell all testified that the field index is not the numeric memory location of the actual field data in an object. RT 3614:22-3615:16 (Bornstein), 3761:14-3762:6 (McFadden), 3790:20-3971:3 (August), 3533:21-25 (Mitchell).

The Field ID table and the other parts of the dex file constant pool information are not “data” within the meaning of the claims. Referring to the description of the dex file format (TX 736), Mr. McFadden testified that the Field ID table is not stored in the data area of a dex file:

Q. So what this description of the overall file layout of a dex file shows is that the Field ID table is not stored in the Data area of a dex file; true, sir?

A. It’s not stored in the section that’s labeled “Data.”

Q. Not stored in the section labeled “Data” by TX 736, Google’s official definition of the dex file format; true, sir?

A. True.

RT 3754:13-19 (McFadden). The constant pool information is not the data that the instructions refer to or obtain. Mr. McFadden testified that, while the IGET instruction obtains actual field data from an object and stores it in a Dalvik register, it does not obtain the field index, string ID index, or the strings “fun” or “byte” and store any of them in a Dalvik register. RT 3759:10-3760:23 (McFadden). 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 value of a field in an instance object—the only data that is relevant to the claims—by a name other than its location.

1 **3. Conversion of instruction stream indices to numeric memory locations**
2 **confirms that the indices are symbolic reference—numeric references**
3 **are not resolved**

4 The fact that the field indices are resolved to numeric references further confirms that they
5 are symbolic references. If they were numeric references and not symbolic references, there
6 would be no need to convert them to numeric references. But because Dalvik does convert field
7 indices (and other instruction stream indices) to pointers (numerical references), no reasonable
8 jury could conclude that the indices are not symbolic references.

9 Google’s Mr. McFadden testified:

10 Q. The Dalvik VM stores pointers that result from resolving the indexes?

11 A. Yes.

12 Q. And the Dalvik VM then pulls them out of storage on subsequent Dalvik
13 bytecode executions?

14 A. Yes.

15 RT 3236:6-11 (McFadden). Mr. McFadden’s source code comments explain that the Dalvik
16 resolving functions convert an index contained in the instruction stream into a pointer:

17 When a class, method, field, or string constant is referred to from Dalvik bytecode,
18 **the reference takes the form of an integer index value.** This value indexes into
19 an array of type_id_item, method_id_item, field_id_item, or string_id_item in the
20 DEX file. The first three themselves contain (directly or indirectly) indexes to
21 strings that **the resolver uses to convert the instruction stream index into a**
22 **pointer to the appropriate object or struct.**

23 TX 46.14 at 1 (emphases added). Mr. McFadden confirmed that this was an accurate description
24 of Dalvik. RT 3236:12-19 (McFadden). He also testified that if the instruction stream index
25 were the numeric memory location, it would already be a pointer and there would be no reason to
26 convert it to a pointer. RT 3234:22-3235:13 (McFadden).

27 That Dalvik resolves a field index to a numeric memory location means that a reasonable
28 jury could come to only one conclusion: a field index is a symbolic reference. Under Google’s
29 view, Dalvik resolves a numeric reference into a numeric reference. As its own witnesses
30 testified, there would be no reason to do that. (*Id.*) The Court should grant judgment of
31 infringement as a matter of law in Oracle’s favor.

1 **B. Android dexopt infringes claims 27 and 29 of the '104 patent**

2 Oracle is also entitled to judgment as a matter of law on infringement of '104 patent
3 claims 27 and 29 by dexopt. Google's engineers testified that dexopt resolves symbolic
4 references into numerical references. *See, e.g.*, RT 3769:8-12 (McFadden). There were only two
5 disputed issues regarding infringement: whether Dalvik dexopt bytecode instructions contain
6 symbolic references and whether dexopt resolves symbolic references dynamically rather than
7 statically. *See, e.g.*, RT 3841:2-19 (August). The first issue is the same as that with respect to
8 Android's *Resolve.c* and should be resolved in Oracle's favor as discussed above. With respect
9 to the second issue, the evidence at trial showed that dexopt resolves symbolic references
10 dynamically rather than statically. No reasonable jury could find otherwise.

11 The Android source code documentation and admissions by Google engineers establish
12 that dexopt resolves references dynamically. Mr. McFadden admitted that the resolution process
13 depends on the conditions actually existing on the handset; dexopt needs to rerun when those
14 conditions change by way of a system update. RT 3769:13-17 (McFadden). *See id.* at RT
15 3255:20-25 (admitting need to run dexopt when performing system update because memory
16 layout could change). Dr. Mitchell agreed. RT 3330:24-3331:21 (discussing McFadden
17 testimony). That is sufficient under the ordinary meaning of "dynamic."

18 Dexopt is performed with a running Dalvik virtual machine. That dexopt runs at
19 "runtime" is another sufficient, although not necessary, basis on which to show dynamic
20 reference resolution. Dexopt must process dex files while the Dalvik Virtual Machine is running
21 because it needs information only available at runtime. Android engineer Dan Bornstein admitted
22 that dexopt processes dex files while the Dalvik Virtual Machine is running. RT 3580:21-23
23 (Bornstein). Similarly, when asked whether "dexopt processes the dex files when the Dalvik
24 Virtual machine is running," Google expert David August responded, "Sometimes." RT 3988:14-
25 3989:23 (August).

26 Google's internal documentation confirms that dexopt optimizations require information
27 only available at runtime. TX 105, part of the Android documentation for dexopt, explains that
28 dexopt is "really just a back door into the VM. It performs an abbreviated VM initialization,

1 loads zero or more DEX files from the bootstrap class path, and then sets about verifying and
2 optimizing whatever it can from the target DEX.” TX 105 at 2. It states, in reference to the
3 optimizations performed by the Dalvik optimizer, that “Some of these require information only
4 available at runtime, others can be inferred statically when certain assumptions are made.” *Id.* at
5 3. As Dr. Mitchell explains, TX 105 shows “the sense in which [dexopt] is dynamic and it’s part
6 of the runtime environment of the Android platform.” RT 3321:22-3332:16, 3989:21-23.
7 Similarly, in TX 1094, HTC developer Kant Kang asks why dexopt has to execute during runtime
8 instead of compile time, noting that it causes “extra cpu usage.” TX 1094. The response from
9 Android engineers is: “What you are seeing is normal behavior” and for an explanation as to
10 “why some of these optimizations can only be performed at runtime,” they quote from TX 105.
11 *Id.* When showed this document, Google’s paid fact witness, Dan Bornstein, tried to dispute the
12 obvious, claiming the Android engineer who responded to this customer query “must have just
13 been confused.” RT 3580:24-3584:8.

14 Notwithstanding these admissions from its own engineers and documents, Google
15 maintains that dexopt is “static” because Google documents sometimes refer to the dexopt
16 symbolic reference resolution process as “static linking.” But calling an apple a “banana” does
17 not change the fact that it is an apple. Likewise, calling a dynamic process “static,” as in TX 816,
18 does not change the fact that it is dynamic.

19 The ordinary meaning of “dynamic” does not require “at runtime.” Mr. McFadden
20 admitted that dexopt is dynamic if “dynamic” means “depending on conditions on the handset
21 which can change from time to time.” *See, e.g.*, RT 3769:23-3770:1 (McFadden). The Court
22 should grant JMOL in Oracle’s favor.

23 **C. Under the correct claim construction, dexopt resolves symbolic references**
24 **“dynamically rather than statically”**

25 As set forth in Oracle’s objection to Jury Instruction 11 (ECF 1128), the Court’s
26 construction of “symbolic reference” should be adjusted to accurately reflect the meaning of the
27 terms “dynamic” and “static” as used in the ’104 patent and the Court’s May 9, 2011 Claim
28

1 Construction Order. Under the proper construction, Google’s non-infringement argument based
2 on dynamic vs. static resolution disappears.

3 The ’104 patent is unambiguous as to what “static” and “dynamic” mean in connection
4 with numeric and symbolic references. The patent uses the terms “static” and “dynamic” as
5 adjectives to characterize numeric and symbolic references, respectively: “[T]he main
6 interpretation routine determines if the data reference is *static, i.e., numeric, or dynamic, i.e.,*
7 *symbolic . . .*” TX 4015, 5:11-13 (emphasis added). A numeric reference is “static” because it
8 does not change – it directly references the memory location of data. *Id.* at 5:24-31 (describing
9 “static field reference routine” used to handle numeric references). A symbolic reference is
10 “dynamic” because it changes – it must be resolved to identify the memory location of data. *Id.*
11 at 5:13-23 (describing “dynamic field reference routine” used to handle symbolic references).

12 The Court’s Claim Construction Order explained this very clearly – numeric references
13 are “static” because they identify a memory location directly, while symbolic references are
14 “dynamic” because they require resolution to a memory location:

15 *A numeric data reference* was one that identified data directly by its memory-
16 location address. For example, the command “load the data stored in memory slot
17 2” contains a numeric reference to the data stored in slot 2 (col. 1:26–41). The
18 claimed invention would use a *static subroutine* to interpret this numeric data
19 reference — all it would have to do is go get whatever data is stored in slot 2

20 *A symbolic data reference*, on the other hand, did not identify data directly by its
21 memory-location address. Instead, a symbolic reference identified data by a
22 “symbolic name” (col. 1:64–67). For example, the command “load the data called
23 y” contains a symbolic reference to the data called y. The claimed invention
24 would use a *dynamic subroutine* to interpret this symbolic reference — it would
25 have to figure out that “y” means “17” or that “y” means “the data stored in
26 memory slot 2,” and then get the data called y (col. 5:13–19).

27 ECF 137 at 20-21, emphasis added.

28 Despite the clarity of the patent and the Court’s Claim Construction Order, the actual
claim construction of “symbolic reference” is ambiguous as to the meaning of “dynamic” and
“static.” As submitted to the jury, “symbolic reference” is construed to mean “a reference that
identifies data by a name other than the numeric memory location of the data, *and that is resolved*
dynamically rather than statically.” ECF 1141 at 5, emphasis added. Oracle originally objected

1 to the inclusion of the last phrase (in italics), noting that confusion might arise over the meaning
2 of “dynamic.” ECF 132 at 2. The Court declined to remove that phrase, but explained:

3 Because this word [“dynamic”] comes directly from the ’104 patent, its use therein
4 will further inform the construction of ‘symbolic reference.’ The word ‘dynamic’
5 is not being imported from a vacuum.

6 ECF 137 at 22. The Court should now apply the correct meaning of “dynamic” to dispose of
7 Google’s non-infringement arguments.

8 Google exploited the ambiguity in the construction of “symbolic reference” to argue that
9 “dynamic” refers to *the timing* of symbolic reference resolution, rather than the nature of the
10 symbolic references. In its closing argument and through its witnesses, Google sought to
11 establish that “dynamic” is sometimes understood in the industry to mean resolution “at runtime,”
12 while “static” may be understood to mean resolution at “install time.” See RT 3762:23-3763:19
13 (McFadden). Irrespective of how those terms may otherwise be used, *that is not how they are*
14 *used in the ’104 patent.* Instead, “dynamic” refers to the changeable nature of symbolic
15 references – they must be resolved to identify the memory location of the underlying data based
16 on memory conditions that exist at whatever time the resolution occurs. TX 4015 at 5:10-31;
17 ECF 137 at 20-21. This is the meaning of “dynamic” that must be applied to resolve Google’s
18 non-infringement argument.

19 Applying the correct meaning of “dynamic,” there is no question that Android’s dexopt
20 dynamically resolves symbolic into numerical references. Google’s Andy McFadden confirmed
21 that the index number contained in the IGET instruction is converted into a pointer, which he
22 admitted was a numerical reference. RT 3234:4-18 (McFadden). While Mr. McFadden and
23 Google deny that the index number is a symbolic reference (as previously addressed), there is no
24 dispute that dexopt replaces it with a numerical reference. RT 3739:13-3741:16 (McFadden).
25 That is the meaning of “dynamic” as used in the ’104 patent. Mr. McFadden further
26 acknowledged that the resolution of index numbers into pointers is also “dynamic” in that it
27 depends on conditions existent on the handset at the time of resolution. RT 3769:8-3770:1
28 (McFadden). Under the correct interpretation of “dynamic,” Google cannot dispute that dexopt
resolves references dynamically.

1 **IV. NO REASONABLE JURY COULD FIND THAT GOOGLE DID NOT INFRINGE**
2 **THE ASSERTED CLAIMS OF THE '520 PATENT**

3 Oracle proved that Google's dx tool infringes Claims 1 and 20 of the '520 patent. Google
4 concedes that all steps of Claims 1 and 20 are performed except the "simulating execution" step.
5 But the indisputable evidence is that "simulating execution" *is* performed by the dx tool code;
6 indeed, it is *expressly described* in the code comments. Google's defense rests solely on reading
7 non-existent limitations into the claims. Absent that legal impropriety, no reasonable jury could
8 find against Oracle.

9 Oracle's expert, Dr. Mitchell, testified that the dx tool simulates the execution of
10 bytecodes to determine the static initialization of arrays. The purpose of the method is to reduce
11 the number of bytecodes needed to initialize arrays. To that end, the dx tool examines the
12 bytecodes *without executing them* to determine the static initialization that they perform. The
13 comments in the dx tool code confirm this. TX 46.16 at line 37 ("Class which knows how to
14 simulate the effects of executing bytecode"). That is the very definition of "simulating
15 execution" recited in the patent claims.

16 Google's two counter-arguments rely on reading non-existent limitations into the claim
17 language: (1) The dx tool does not infringe because it does not manipulate a data stack in
18 determining static initialization of arrays; and (2) the dx tool does not infringe because it uses
19 pattern matching to determine static initialization. Neither is a defense. The asserted claims do
20 not require stack manipulation, so its absence from the dx tool is not exculpatory. And the
21 asserted claims do not exclude pattern matching, so even if that forms part of the dx tool process,
22 it still qualifies as "simulating execution" of bytecodes.

23 **A. '520 Patent Background**

24 The invention of the '520 patent addresses a problem with static array initialization by the
25 Java virtual machine. TX 4011 ('520 patent), 1:57-2:58; RT 3334:16-3335:19 (Mitchell). Static
26 arrays are lists of data items (such as numbers) that are used by Java programs. RT 3334:20-25
27 (Mitchell). For static arrays to work, they need to be initialized in the Java virtual machine. TX
28

1 4011, 1:57-2:52. Initialization occurs via a method (“<clinit>” or “class initialization”) that uses
2 a series of Java bytecodes to initialize the static array. *Id.*

3 As an example, the patent shows a static array written in Java source code. *Id.* at 1:65.
4 When this source code is compiled, the Java compiler produces a long list of Java bytecode
5 instructions to initialize the array. *Id.*, 2:26-57; RT 3335:10-19 (Mitchell). This set of bytecode
6 instructions is larger than the array itself and takes up more memory space. *Id.*

7 To reduce the bytecodes needed for initialization, the invention simulates execution of the
8 bytecode instructions to determine the static initialization they perform. *Id.*, 2:64-3:7. The long
9 list of instructions is then replaced with a shorter instruction indicating the static initialization of
10 the array. *Id.*, 3:66-4:17. In this way, the invention saves memory space.

11 **B. Android’s dx Tool Simulates Execution Of Bytecodes To Identify The Static**
12 **Initialization Of Arrays**

13 Claims 1 and 20 of the ’520 patent are infringed by Android’s dx tool. As indicated in the
14 patent claim handouts, Google admits that all steps of Claims 1 and 20 are performed by the dx
15 tool except for the following steps:

16 *From Claim 1*

17 simulating execution of the byte codes of the clinit method against a memory
18 without executing the byte codes to identify the static initialization of the array

19 *From Claim 20*

20 simulating execution of the code to identify the static initialization of the array.

21 TX 1106 at 7-8. Google’s code demonstrates that the dx tool does, in fact, perform these
22 “simulating execution” steps.

23 **1. Dr. Mitchell’s Analysis**

24 The dx tool is part of the Android SDK, a platform developers use to write and compile
25 Android applications. RT 3549:25-3550:2 (Bornstein), 3253:23-3254:2 (McFadden). The first
26 version of the dx tool was written at Google by former engineer Dan Bornstein. RT 3547:20-21
27 (Bornstein). Android applications written in the Java programming language are first compiled to
28 Java bytecode using a Java compiler. RT 3547:5-10 (Bornstein). The Android dx tool is then

1 used to transform the Java bytecode files into Android dex code files. RT 3547:14-19
2 (Bornstein).

3 Because the dx tool processes Java bytecode compiled by a Java compiler, it faces the
4 same problem described in the '520 patent – namely, that files containing static arrays will have
5 long lists of bytecode instructions for array initialization. RT 3338:15-3339:1 (Mitchell). The dx
6 tool solves this problem using the patented technique; it simulates execution of the instructions
7 and replaces them with a single instruction to initiate the array. RT 3338:19-3339:1 (Mitchell).

8 A code file called “Simulator.java” within the dx tool simulates execution of the
9 initialization bytecodes to figure out what they do. TX 46.16 (Simulator.java); RT 3340:5-
10 3341:16 (Mitchell). The engineer comments in the code clearly state that Simulator.java is a class
11 designed to “simulate the effects of executing bytecode”:

```
12 36 /**  
13 37 * Class which knows how to simulate the effects of executing bytecode.  
14 38 *  
15 39 * <p><b>Note:</b> This class is not thread-safe. If multiple threads  
16 40 * need to use a single instance, they must synchronize access explicitly  
17 41 * between themselves.</p>  
18 42 */  
19 43 public class Simulator {  
20 44     /**
```

21 TX 46.16 at lines 37-43, 86-105. The file calls upon the parseInstruction and parseNewarray
22 methods to assist with understanding the instructions. TX 46.16 at line 99; TX 46.17 at lines 211,
23 887; RT 3341:17-3344:7 (Mitchell). As a result of Simulator.java and the methods it invokes, the
24 bytecode instructions are examined without being executed, their static initialization is
25 determined, and a shorter “fast instruction” is generated to replace the long list of bytecode
26 instructions. *Id.* This precisely matches the “simulating execution” step of the asserted claims.

27 To confirm the infringing functionality of the dx tool, Dr. Mitchell performed an
28 experiment. RT 3344:8-3346:1 (Mitchell). He created a file containing a static array of ten
elements and compiled the file to Java bytecode using a Java compiler. *Id.* The resulting
bytecode contained roughly 50 instructions to initialize the array. *Id.* He then ran the bytecode
through the dx tool. *Id.* The resulting dex file had a single, succinct instruction to initialize the
array, rather than the list of 50 instructions. *Id.* This confirmed that the dx tool, using

1 Simulator.java, had simulated execution of the instructions, determined their static initialization,
2 and replaced them with a shorter instruction, just as recited in the asserted claims. *Id.*

3 **2. Dr. Parr Sought To Read Additional Limitations Into The Claim**
4 **Language**

5 Google's expert, Dr. Parr, conceded that the dx tool does "identify the static initialization
6 of the array" by examining the "byte codes of the clinit method against a memory" and "without
7 executing the byte codes." RT 3793:2-5, 3807:10-14, 3820:12-22, 3821:16-23, 3822:17-3823:13.
8 With those concessions, Dr. Parr acknowledged that the sole remaining issue was whether the dx
9 tool process for identifying static initializations could be characterized as "simulating execution."

10 On this issue, Dr. Parr offered two arguments, both dependent on importing additional
11 limitations into the asserted claims. First, Dr. Parr said that the dx tool cannot be simulating
12 execution of bytecodes because it does not manipulate a stack to determine static initializations of
13 arrays. RT 3794:15-3795:21, 3801:19-21. In support, he pointed to an exemplary embodiment in
14 the patent specification involving stack manipulation. *Id.* However, as Dr. Parr himself
15 conceded, the asserted claims make no mention of stack manipulation. TX 4011, 9:47-62, 12:3-7;
16 RT 3794:20-23. Nor has the term "simulating execution" been construed to require stack
17 manipulation. Indeed, dependent Claim 3 includes "stack manipulation" as an express limitation,
18 establishing that the limitation is not part of independent Claim 1. The absence of stack
19 manipulation in the dx tool is irrelevant to infringement, as the asserted claims do not require that
20 feature.

21 Second, Dr. Parr argued that the dx tool identifies static initializations through pattern
22 matching, which he contends is distinguishable from "simulating execution" of bytecodes. RT
23 3798:22-3799:3. While Dr. Parr admitted that the process of creating and initializing arrays
24 begins in the Simulator.java file, he argued that the identification of static initializations occurs in
25 another file and involves pattern matching. RT 3800:2-3801:18, 3830:12-19, 3834:8-16,
26 3834:25-3835:5. Again, this argument relies on limiting the meaning of "simulating execution,"
27 this time to exclude any pattern matching. The term has not been construed that way. There is
28 nothing in the claim language to suggest that "simulating execution" cannot be achieved through

1 pattern matching. The meaning of “simulating execution” is apparent from the claim language
2 itself:

3 simulating execution of the byte codes of the clinit method against a memory
4 *without executing the byte codes* to identify the static initialization of the array

5 TX 4011, Claim 1 (emphasis added). In the context of the surrounding language, “simulating
6 execution” is performed on “byte codes of the clinit method,” “without executing the byte codes,”
7 “to identify the static initialization of the array.” Dr. Parr has admitted that those elements are
8 performed by the dx tool. Whether or not this occurs through pattern matching, it is still
9 “simulating execution” – *i.e.*, identifying static initialization of bytecodes *without actually*
10 *executing them.*

11 **V. GOOGLE’S EQUITABLE DEFENSES FAIL**

12 Google’s equitable defenses to Oracle’s copyright infringement claim overlap with its
13 equitable defenses to Oracle’s patent infringement claims. Oracle’s proposed findings of fact and
14 conclusions of law submitted after Phase I reflect the state of the record at the end of that phase
15 and are hereby incorporated by reference. *See* ECF Nos. 1048 at 11-26, 30-35; ECF 1081 at 16-
16 37, 55-70.

17 In addition, Oracle identifies below further evidence introduced in Phase II of the trial
18 relating to the equitable defenses, and highlights other evidence particularly relevant to those
19 defenses in the context of Oracle’s patent claims. Google did not submit new evidence in Phase
20 II that is material to its equitable defenses. Under the law that applies to Google’s equitable
21 defenses in the patent context, judgment should be entered in Oracle’s favor.

22 **A. Additional Facts Relevant to Google’s Equitable Defenses**

23 During Phase I, Oracle introduced extensive evidence of the negotiations between Oracle
24 and Google beginning in 2005 and of Google’s internal recognition of the need to take a license
25 from Sun. *See, e.g.*, ECF 1048 at ¶¶ 64-65, 85-86, 98-99. One of Google’s primary goals in
26 these negotiations was to obtain a license to Sun’s patents. *See, e.g.*, TX 2714 at 1 (Feb. 6, 2006
27 Rubin email) (“If you and I can define the open source license and include patent protection, then
28 Eric will be 100% supportive”); TX 22 at 8 (Apr. 24, 2006 presentation entitled “Android/Sun

1 final approval” and describing proposed license as “includes patent grants”); TX 618 §§ 1.17,
2 1.22, 1.32, 3.1(d) (draft licensing agreement including license to Sun patents).

3 The key Google participants in the negotiations were already familiar with Sun’s patents.
4 Andy Rubin had negotiated a license with Sun at his prior company, Danger, which included
5 license rights to Sun patents. TX 1026 § 2.1(Sun-Danger license); TX 565 (Aug. 2, 2007 Gupta
6 email) (“Andy cannot say he is not aware of the licensing requirements- as he had to go thru this
7 at Danger- and we discussed this during Project Android Phase, and then during the Sun/Google
8 collaboration attempt as well”). Mr. Rubin was specifically aware that Sun had patents relating to
9 the virtual machine. After initially resisting on cross-examination, Mr. Rubin finally admitted
10 that he “had discussions with Sun about patents relating to the virtual machine.” RT 3204:6-
11 3205:3 (Rubin). Mr. Rubin also authored several emails that showed he was aware of the need to
12 take a license to Sun’s patents. In March 2006, for example, Mr. Rubin wrote, “I don’t see how
13 you can open java without sun, since they own the brand and ip.” TX 18 at 1. Mr. Rubin
14 acknowledged at trial that by “IP” he meant “Patents, copyrights and the like.” RT 1355:1-4
15 (Rubin). Six months later, in November 2006, in response to Sun’s announcement that it was
16 open sourcing the Java platform, Mr. Rubin cautioned: “They still have patents and trademarks.”
17 TX 155 at 1. Mr. Rubin explained what he meant by this remark:

18 Look, like I said before, I assume they're running a business, they're inventing
19 intellectual property, they're protecting it through the patent system. Through
20 GPL, I didn't know what they were, *but I knew that it was dangerous to use the
stuff without knowing exactly what it was.*

21 TX 1128 at Rubin Dep. Tr. 16:4-16 (emphasis added).

22 Tim Lindholm was another key Google representative in the negotiations. He assisted
23 Rubin by “helping negotiate with my old team at Sun for a critical license.” TX 17 at 1. Before
24 joining Google, Mr. Lindholm worked at Sun, specifically with the Java Virtual Machine, and
25 was a named inventor on more than 10 Sun patents, many or all of which relate to Java virtual
26 machines. RT 2993:5-24. He was aware of the ’104 patent in particular. Mr. Lindholm co-
27 authored the book, The Java Virtual Machine Specification, with Frank Yellin. TX 25.

28 Mr. Yellin also now works for Google. RT 2997:14-2998:15. Chapter 9 of that book states, “The

1 technique documented in this chapter is covered by U.S. Patent 5,367,685.” TX 25 at 389. The
2 ’685 patent is the predecessor to the ’104. *See* TX 4015 at 1. Google’s argument that all claims
3 of the ’104 patent are broader than the claims of the ’685 patent (ECF 311) means that the ’104
4 would cover the technology described in the chapter as well.

5 Despite his concern that it was “dangerous” to proceed without knowing what Sun’s
6 intellectual property covered (TC 1128 at Rubin Dep. Tr. 16:4-16), Mr. Rubin never asked
7 Mr. Lindholm—or anyone else on his team—to review any Sun patents or investigate whether
8 Android technology might infringe one of Sun’s patents. RT 3140:17-3141:1 (Rubin), 3027:11-
9 3028:4 (Lindholm).

10 To counter this evidence, in its closing statement, Google trotted out the Jonathan
11 Schwartz November 2007 blog post yet again, claiming it showed Sun had no concerns about
12 Google’s patent infringement. RT 4193:2-13 (Google closing). But Android’s source code was
13 not publicly released until October 21, 2008—*almost a full year later*. RT 1719:10-18 (Rubin).
14 Google had not even finished developing the Android source code in November 2007, let alone
15 publicly released it. *See* RT 1507:20-1508:18 (Schmidt). Mr. Schmidt’s alleged conversations
16 with Mr. Schwartz similarly took place in late 2007 and early 2008 (*id.* at 1537:3-18)—again,
17 well before Google’s source code was released and Sun could have known of Google’s
18 infringement.

19 When the source code was released, Sun was in discussions with Google over a license to
20 Java. *See, e.g.*, TX 1058 (Oct. 7, 2008 Gupta email to Rubin) (“Many thanks for taking time to
21 kick the discussion off yesterday”). These discussions continued. On November 24, 2008,
22 Mr. Rubin wrote that Sun had asked him “to certify Android through the Java process and
23 become licensees of Java.” TX 1002. In February 2009, Brett Slatkin proposed to Eric Schmidt
24 that Google buy the rights to Java and “solve all these lawsuits we’re facing.” TX 406 at 1.
25 Schmidt wrote back “Certainly a clever idea. I’ll ask our team to pursue.” TX 406 at 1. The
26 proposal was also forwarded to Tim Lindholm and Bob Lee, the lead Android core libraries
27 developer. TX 326.

1 There is a particularly revealing set of emails relating to a meeting that took place between
2 the parties on April 29, 2009. In an email written immediately after the meeting, which was
3 admitted in Phase II, Oracle's account manager for Java technology and source code licensing,
4 Leo Cizek writes that he confronted Google's Martin Buccholz about the need for Google to take
5 a license for Android:

6 Vineet,

7 Tom and I spoke with Martin Buccholz today. I delivered the message that they
8 have only two options: OpenJDK or Commercial Use, which would require
9 compatibility. I also explained that using Java in the context of customer-facing
10 applications is considered by Sun to be commercial use. I also explained that if
11 they choose the commercial use/compatibility option, it would have ramifications
12 throughout Google, and I gave Android as an example.

13 Martin replied: "The Android group did not use any Java code in developing
14 Dalvik; they only used the Java specifications."
15 Unfortunately I did not have a tape recorder running at the time!

16 I replied that Sun's position is that the spec license agrmts require that any s/w
17 created from them which is for commercial use be compatible.

18 TX 531.

19 Mr. Buccholz sent a parallel email internally at Google that same afternoon:

20 As expected, this does not look promising. Leo says Sun has an inflexible
21 licensing model where Open JDK never gets any support of any kind, and the
22 commercial version of the code does (and mostly Sun is thinking of support for
23 binaries). *Also, Sun would want any discussions with Google to involve other
24 inter-company issues, in particular Android, which I'm sure we would want to
25 keep separate. Android seems to be a big deal at Sun.* Leo suggested that his boss,
26 Vineet Gupta, CTO for OEM Software Systems Engineering, should have a chat
27 with Tim Lindholm (Tim knows both of these guys) and I agreed that would be a
28 good idea.

TX 1029 (emphasis added).

 Tim Lindholm's initial response, after stating, "I guess this isn't surprising," was to agree
to meet with Mr. Gupta. *Id.* But then he thought better of it:

*Actually, having said that I wonder whether this is too close to dangerous
territory, and with too little chance of anything positive coming of it for us to be
messing around? We really don't want to inadvertently stir anything up for
Android, and Leo has said pretty clearly that they don't have anything for us as
regards security patches. I suspect we should step away, and only respond further
if Sun chases after us.*

1 *Id.* (emphasis added). Mr. Bornstein responded (“No surprise that I think this is exactly what we
2 should do.”). *Id.*

3 Google stepped away, but it could not hide for long. Oracle’s acquisition of Sun closed
4 on January 27, 2010. ECF 525 at 8 Stipulated Fact No. 3. One of Mr. Ellison’s first acts was to
5 meet personally with Mr. Schmidt about Android in March 2010 to persuade Google to become
6 compatible with Java. RT 312:18-20 (Ellison). Several meetings took place between the two
7 companies over the next few months, including a meeting between Mr. Ellison and Mr. Page,
8 meetings with Andy Rubin and his boss, Alan Eustace, and a meeting on July 20, 2010 where
9 Google admits the ’104 and ’520 patent were specifically discussed. RT 391:15-395:2 (Kurian),
10 2309:22-2314:12 (Catz). The lawsuit was filed on August 12, 2010.

11 **B. Google Has Not Shown that Equitable Estoppel Bars Oracle’s Patent**
12 **Infringement Claims**

13 To prevail on its estoppel defense, Google must prove that (1) Sun/Oracle, “through
14 misleading conduct, led” Google “to reasonably infer that” Sun/Oracle “did not intend to enforce
15 its patent against” Google; (2) Google “relied on this conduct;” and (3) “due to the reliance,”
16 Google “will be materially prejudiced if” Oracle “is allowed to proceed on its claim.” *Gasser*
17 *Chair Co. v. Infanti Chair Mfg. Corp.*, 60 F.3d 770, 776 (Fed. Cir. 1995).

18 Google failed to prove any of these elements. The test for patent estoppel is similar to the
19 one for copyright estoppel, so Oracle refers the Court to its analysis on that subject. *See*
20 ECF 1049 at 29-31; ECF 1081 at 60-63. In particular, Google has no credible claim that it relied
21 on Sun/Oracle’s conduct to its detriment or that its reliance was reasonable. The jury so advised
22 in its Phase I ruling. ECF 1089 ¶ 4.B. While this advisory verdict was specifically in the context
23 of the SSO of the copyrighted code, Google did not introduce any new material evidence in phase
24 II that provides support for this claim, so the verdict applies equally here.

25 The evidence shows that negotiations between the parties over Java licensing continued
26 throughout the relevant time period and beyond. *See, e.g.*, RT at 492:18-22 (Page) (“I’m not sure
27 they’ve ever broken off. Continue to have discussions to this day”); TX 531, 1002, 1029, 1058,
28 ECF 1049 ¶¶ 85, 99-100, 133. Overwhelming evidence at trial showed Google was aware that

1 Sun had patents that covered the virtual machine technology and that it faced potential legal
2 action by Sun in connection with Android. *See, e.g.*, RT 3204:6-3205:3 (Rubin); RT 2993:4-24
3 (Lindholm); TX 18, TX 25 at 389, TX 1029, ECF 1049 ¶¶ 62-65, 130, 132, ECF 1081 ¶ 66.
4 Google’s argument is one made up by the lawyers in hindsight. In all of its discussions with Sun
5 and Oracle, Google never claimed that it had been led by Sun to believe that it didn’t need a
6 license or had relied on any such belief. ECF 134, RT at 2316:1-9 (Catz). Google decided on its
7 Android development path and implemented the infringing technology regardless of any Sun or
8 Oracle statements, actions, or inactions, and Google has never proven otherwise. TX 1029, ECF
9 1049 ¶¶ 62-65, 96, 98, 114, 117. Google knew the risks of operating without a license but
10 decided to proceed anyway.

11 **C. Google Has Not Shown that the Doctrine of Laches Applies to Oracle’s Patent**
12 **Infringement Claims**

13 Google has not produced evidence to raise a colorable laches defense. In patent, as in
14 copyright, laches requires proof of the following elements: (1) “the plaintiff delayed filing suit
15 for an unreasonable and inexcusable length of time from the time the plaintiff knew or reasonably
16 should have known of its claim against the defendant,” and (2) “the delay operated to prejudice or
17 injury of the defendant.” *A.C. Aukerman Co. v. R.L. Chaides Const. Co.*, 960 F.2d 1020, 1032
18 (Fed. Cir. 1992) (*en banc*). “A district court must weigh all pertinent facts and equities in making
19 a decision on the laches defense.” *Id.* at 1034 (“the length of delay, the seriousness of prejudice,
20 the reasonableness of excuses, and the defendant’s conduct or culpability must be weighed”).

21 Here, Oracle filed suit on August 12, 2010, less than two years from the first time that
22 Google made the code in Android available to the public, on October 21, 2008. RT 1719:10-18
23 (Rubin). Because Oracle brought suit within less than six years of learning of the infringement,
24 there is no presumption of laches, and “the burden is upon the defendant to show that the delay
25 was unexcused and that the defendant suffered injury as a result of the delay.” *Carpet Seaming*
26 *Tape Licensing Corp. v. Best Seam, Inc.*, 694 F.2d 570, 580 (9th Cir. 1982); *A.C. Aukerman*,
27 960 F.2d at 1038.

1 While Oracle did not delay unreasonably, negotiations with the accused infringer would
2 justify delay in filing suit in any case. *See A.C. Aukerman Co.*, 960 F.2d at 1033. Courts have
3 found delay reasonable or excusable where evidence shows that for several years leading up to
4 the start of litigation, plaintiff engaged in efforts or negotiations to license the defendant. *In re*
5 *Katz Interactive Call Processing Patent Litig.*, 712 F. Supp. 2d 1080, 1110-11 (C.D. Cal 2010).
6 *Lucent Techs., Inc. v. Gateway, Inc.*, 580 F. Supp. 2d 1016, 1053 (S.D. Cal. 2008) (granting
7 plaintiff’s JMOL of advisory verdict—and holding defendant “did not prove the laches factors by
8 a preponderance of the evidence, and even if it had, the Court would exercise its discretion and
9 decline to apply laches in light of all the circumstances of this case.”), *vacated-in-part on other*
10 *grounds* by 525 F.3d 1200 (Fed. Cir. 2008).

11 Google also failed to demonstrate material prejudice by showing that it took actions or
12 suffered consequences that it would not have had Sun/Oracle brought suit earlier. “The courts
13 must look for a change in the economic position of the alleged infringers during the period of
14 delay.” *A.C. Aukerman*, 960 F.2d at 1033. Google’s policy was to push forward and develop
15 Android even if it risked “making enemies along the way,” and it did not change its position in
16 reliance on Oracle’s inaction. *See Oracle’s Proposed Findings of Fact*, ECF 1049, at 19-24.

17 Finally, “laches is not available in a case of willful infringement.” *Cf. A.C. Aukerman*,
18 960 F.2d at 1033 (conscious copying may constitute “particularly egregious conduct which would
19 change the equities significantly in the plaintiff’s favor.”); *Winn v. Opryland Music Group, Inc.*,
20 22 Fed. Appx. 728, 729 (9th Cir. 2001). As the evidence shows willful infringement here,
21 Google may not assert the defense of laches. *See Oracle’s Proposed Findings of Fact*, ECF 1049,
22 at 19-24.

23 **D. Google Has Not Shown that Oracle or Sun Waived Its Right to Assert Patent**
24 **Infringement Claims**

25 To prevail on its claim for waiver, Google must prove by clear and convincing evidence
26 that Oracle or Sun, with full knowledge of the material facts, intentionally relinquished rights to
27 enforce the ’104 or the ’520 patents. *Qualcomm Inc. v. Broadcom Corp.*, 548 F.3d 1004, 1019-
28 1020 (Fed. Cir. 2008). “An implied waiver of rights will be found where there is ‘clear, decisive

1 and unequivocal' conduct which indicates a purpose to waive the legal rights involved." *Adidas-*
2 *Am., Inc. v. Payless Shoesource, Inc.*, 546 F. Supp. 2d 1029, 1074 (D. Or. 2008) (quoting *U.S. v.*
3 *Amwest Surety Ins. Co.*, 54 F.3d 601, 602–03 (9th Cir.1995)).

4 Google has presented no credible evidence that Sun/Oracle intentionally relinquished
5 rights. Neither Sun nor Oracle ever relinquished anything. Instead both made numerous attempts
6 to negotiate a license with Google. *See* section V.A. above and ECF 1081 at 68-70. Moreover,
7 neither the November 2007 blog post from Mr. Schwartz nor the alleged meetings between Mr.
8 Schwartz and Mr. Schmidt in late 2007 and early 2008 can support Google's waiver defense
9 because both took place before Google released its source in October 2008. RT 1719:10-18
10 (Rubin). Sun therefore could not have had "full knowledge" of the relevant facts at the time of
11 Schwartz's blog post. *Qualcomm*, 548 F.3d at 1019-20.

12 **E. Google Has Not Shown that Oracle or Sun Gave It an Implied License to Use**
13 **Oracle's Patents**

14 The doctrine of implied license has no application here. In the patent infringement
15 context, "[t]he implied license defense is typically presented 'when a patentee or its licensee sells
16 an article and the question is whether the sale carries with it a license to engage in conduct that
17 would infringe the patent owner's rights.'" *Zenith Elecs. Corp. v. PDI Commc'ns Sys.*, 522 F.3d
18 1348, 1360 (Fed. Cir. 2008) (granting JMOL against implied license defense, holding that "an
19 implied license arising from sale of a component to be used in a patented combination extends
20 only for the life of the component whose sale and purchase created the license."); *Jacobs v.*
21 *Nintendo of Am., Inc.*, 370 F.3d 1097, 1100 (Fed. Cir. 2004) (implied license was derived from
22 express license between Jacobs and purchaser that specifically authorized purchaser to sell
23 accelerometers for infringing uses). Relatedly, for there to be an implied license, the plaintiff
24 must have made an "affirmative grant of consent or permission" to the accused infringer to use
25 the patented inventions. *Wang Labs., Inc. v. Mitsubishi Elecs. Am., Inc.*, 103 F.3d 1571, 1581
26 (Fed. Cir. 1997).

1 This case could hardly be more different from the ones cited above. There was no “sale”
2 by Oracle to Google, and no affirmative grant of permission. The record shows the reverse. *See*
3 *e.g.*, ECF 1049 at 15-19.

4 **VI. ALTERNATIVE GOOGLE DEFENSES THAT GOOGLE PLED BUT DID NOT**
5 **PRESENT TO THE JURY FAIL**

6 **A. Google Has Not Shown Patent Misuse, Use by the United States, or Unclean**
7 **Hands**

8 In its answer, Google asserted the defenses of patent misuse (Sixth Defense, Google Inc.’s
9 Answer To Plaintiff’s Amended Complaint For Patent And Copyright Infringement And
10 Amended Counterclaims (“Google Answer”) (ECF 51) at 10), use by the United States (Eighth
11 Defense, Google’s Answer at 10), and unclean hands (Nineteenth Defense, Google’s Answer at
12 12). Google did not identify these defenses as remaining to be decided in the parties’ October 13,
13 2011 Joint Proposed Pretrial Order. (ECF 525 at 1-6.) The Court ruled that only those claims or
14 defenses set forth in the Joint Proposed Pretrial Order remained in the case.

15 (1/4/2012 Final Pretrial Order (ECF 675) at 1.) *See Dream Games of Ariz., Inc. v. PC Onsite*, 561
16 F.3d 983, 996 (9th Cir. 2009) (“[P]arties are typically considered bound by the statements of
17 claims made in their pretrial order.”). Accordingly, these defenses must fail.

18 **B. Google Has Not Shown that Oracle or Sun Gave It an Express License to Use**
19 **Oracle’s Patents**

20 The Court should grant judgment as a matter of law to Oracle on Google’s express license
21 defense. An express license is a defense to patent infringement. *See Carborundum Co. v. Molten*
22 *Metal Equip. Innovations*, 72 F.3d 872, 878 (Fed. Cir. 1995). Google presented no evidence that
23 any user of Android (including Google itself) held an express license from Oracle or Sun to the
24 ’104 or ’520 patents. Nor did Google argue that defense to the jury. Accordingly, Oracle should
25 be granted judgment as a matter of law on this defense.

26 **VII. CONCLUSION**

27 For the foregoing reasons, Oracle is entitled to judgment in its favor on its patent
28 infringement claims and against Google on Google’s defenses.

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

Dated: May 16, 2012

MORRISON & FOERSTER LLP

By: /s/ Michael A. Jacobs
Michael A. Jacobs
Attorneys for Plaintiff
ORACLE AMERICA, INC.