IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF ILLINOIS
EASTERN DIVISION

CASCADES COMPUTER INNOVATION, LLC., Plaintiff, v. MOTOROLA MOBILITY, INC. and SAMSUNG ELECTRONICS CO., LTD. Defendants.	Civil Action No. 1:11-cv-4574 Honorable Robert W. Gettleman

THIRD AMENDED COMPLAINT FOR PATENT INFRINGEMENT

Cascades Computer Innovation, LLC ("Cascades") complains of Defendants Motorola Mobility, Inc. and Samsung Electronics Co., Ltd., and alleges the following:

NATURE OF THE SUIT

1. This is a suit for patent infringement arising under the patent laws of the United States, Title 35 of the United States Code § 1 et seq. This Court has exclusive jurisdiction over the subject matter of this Complaint under 28 U.S.C. § 1338(a).

PARTIES

2. Cascades is an Illinois limited liability company having its principal place of business at 500 Skokie Boulevard, Suite 350, Northbrook, IL 60062.

3. Defendant Motorola Mobility, Inc. ("Motorola") is a Delaware corporation headquartered at 600 N. U.S. Highway 45, Libertyville, Illinois 60048.

4. Samsung Electronics Co., Ltd. ("Samsung") is a foreign corporation with corporate headquarters at 250, 2-ga, Taepyung-ro, Jung-gu, Seoul 100-742, Republic of Korea.

5. Cascades' claim for patent infringement against Motorola and Samsung arises under the patent laws of the United States, including 35 U.S.C. §§271 and 281. Consequently, this Court has original subject matter jurisdiction over this suit pursuant to 28 U.S.C. §§1331 and 1338.

6. Motorola sells products throughout the United States and conducts substantial business in this judicial district, including providing the products accused of infringement in this judicial district. Motorola maintains its headquarters in this judicial district and also maintains retail sales offices in this judicial district.

7. Motorola is subject to both specific and general personal jurisdiction of this Court because, among other things, it has established continuous and systematic contacts with Illinois and in this judicial district; it has committed acts within Illinois and this judicial district giving rise to this action, and it has minimum contacts with the forum such that the exercise of jurisdiction over it would not offend traditional notions of fair play and substantial justice. As an example, Motorola has established distribution networks placing products that are covered by claims of Cascades' United States Patent No. 7,065,750 into the stream of commerce such that those products flow into Illinois and this district. Motorola has also committed acts of patent infringement and/or contributed to others' acts of patent infringement within this district.

8. Samsung sells products throughout the United States and conducts substantial business in this judicial district, including providing the products accused of infringement in this judicial district.

9. Samsung is subject to both specific and general personal jurisdiction of this Court because, among other things, it has established continuous and systematic contacts with Illinois and in this judicial district; it has committed acts within Illinois and this judicial district giving rise to this action, and it has minimum contacts with the forum such that the exercise of jurisdiction over it would not offend traditional notions of fair play and substantial justice. As an example, Samsung has established distribution networks placing products that are covered by claims of Cascades' United States Patent No. 7,065,750 into the stream of commerce such that those products flow into Illinois and this district. Samsung has also committed acts of patent infringement and/or contributed to others' acts of patent infringement within this district.

10. Venue is proper in this judicial district under 28 U.S.C. §§ 1391 and/or 1400(b).

PATENT AT ISSUE

11. On June 20, 2006, United States Patent No. 7,065,750 ("the '750 patent"), entitled "Method and Apparatus for Preserving Precise Exceptions in Binary Translated Code," was duly and legally issued by the United States Patent and Trademark Office. Cascades owns the exclusive license and right to sue for past, present and future infringement of the '750 Patent.

12. Motorola is now and has been infringing and/or contributorily infringing the '750 patent, literally and under the doctrine of equivalents, by, among other things, using, offering to sell, selling, re-selling and/or importing products that are covered by one or more claims of the '750 patent. Such infringing products include, but are not limited to, cell phone products such as its Motorola DROID^3 smartphones. An illustrative claim chart demonstrating how Motorola and its customers practice the

method of claim 15 of the '750 patent using the Motorola DROID smartphones with their Android operating system and software is attached as Exhibit A. Other Motorola products that use the Android operating system and software infringe in the same way. In fact, Motorola openly admits this on its website:

> BASKING RIDGE, N.J., and LIBERTYVILLE, Ill. – July 7, 2011 – Verizon Wireless and Motorola Mobility, Inc. (NYSE: MMI), today announced the new Android™-powered DROID 3 by Motorola, a global smartphone that delivers power for work and play without making compromises.
>
> DROID 3 by Motorola is the world's thinnest full QWERTY smartphone, and still delivers the power of a dual-core 1 GHz processor for fast multi-tasking. Customers can take stunning photos with the 8-megapixel camera or capture the moment in 1080p HD video. Equipped with Android 2.3, the DROID 3 by Motorola features a brilliant 4-inch qHD display, a roomy 5-row QWERTY keyboard and 3G Mobile Hotspot capabilities, with the ability to connect up to five Wi-Fi-enabled devices. DROID 3 by Motorola delivers the power needed to conquer the day whether customers are at home, work or somewhere in between.
>
> **Additional features**
>
> - Powered by Android™ 2.3 Gingerbread

http://mediacenter.motorola.com/Press-Releases/Verizon-Wireless-Introduces-the-Next-Generation-Droid-Delivering-Power-and-Performance-The-Droid-3-by-Motorola-3735.aspx. Motorola's personnel likewise demonstrate Motorola's Droid smartphones using Android (See, e.g., http://www.youtube.com/watch?v=gJyRGc7juG4), and infringement occurs through this use in the manner described in Exhibit A. Most recently, Motorola's Senior Vice President, Alain Mutricy, demonstrated Motorola's Droid 4, the successor to the Droid 3, which he indicates is being upgraded to Android 4.0: http://www.youtube.com/watch?v=GHNZlpwBx4o

13. Further, Motorola is a contributory infringer of the '750 patent because it knew at least when the original complaint was filed and, thus, knew then (and now

knows) that the steps of the claimed method described in claim 15, for example, were carried out by users of Motorola's phones, such as the Motorola DROID[3] smartphone, that employ Android operating systems and software. Direct infringement by customer/end-users is illustrated in the attached Exhibit A. Motorola's customers are direct infringers and Motorola contributes to their infringement in that Motorola's customers carry out each and every step of the method defined, for example, in claim 15, as illustrated in the claim chart, Exhibit A. Motorola is fully aware of such direct infringement and encourages, aids, and assists it. Motorola also knows there are no substantial non-infringing uses of the accused products and Motorola knows its phones using the Android operating system and software are especially designed and made to use the Android operating system and software and, thus, are designed to practice, for example, the method of claim 15 of the '750 patent. Motorola knows this occurs when a Motorola customer/end-user operates a phone in the manner Motorola directs, instructs and teaches.

14. Motorola does more than simply sell products that use the Android operating system. It directs customers to use and shows them how to use the Android operating system and, thus, facilitates such use. Motorola's customers, in turn, actually use the identified cell phones to carry out each step of method claim 15, which is an act of direct infringement. The conditions of such use are known to Motorola and set forth in Exhibit A. Specifically, use of the method occurs when the phone is turned on and made functional. When the Android operating system starts up, the Dalvik Virtual Machine in the phone looks through the applications installed on the phone and builds a tree of dependencies. This dependency tree optimizes the byte code for every application and stores it in the Dalvik cache. The applications are then run using the

optimized byte code. Motorola is fully aware that its phones using Android operating systems are designed to operate and do, in fact, operate in this way. The claimed method is performed when a user of the cellular phones operates the device for their intended purpose using the Android operating system, e.g., allowing the Dalvik Virtual Machine to optimize the byte code for each application.

15. Further, Motorola's manufacture, sale, offer to sell and importation of the identified cell phones is a direct infringement of apparatus/system claim 1, for example, in that smartphones made and sold by Motorola such as the DROID^3 smartphone include each element of claim 1, including a non-optimizing foreign code execution module, an optimizing binary translator, a host CPU, a documentation tracker and a recovery mechanism, all as required by claim 1.

16. Motorola makes the DROID^3 cellular telephone.

17. Motorola DROID^3 phone includes both hardware and software components.

18. Motorola imports, sells, and offers to sell the DROID^3 phone to customers in the United States.

19. Motorola customers use the DROID^3 phones in the United States.

20. The Motorola DROID^3 phone:

a. includes a binary translation system.

b. includes an optimizing translator.

c. includes a binary translated code translated from foreign code.

d. designates a set of recovery points in the optimized binary translated code during optimized translation of the foreign code.

e. generates a set of documentations during the optimized translation of the foreign code.

21. The Motorola DROID[3] phone uses one of the documentations in the set of documentations corresponding to executed optimized binary translated code when an exception arises during its execution to recover a foreign state corresponding to a recovery point for the exception.

22. The Motorola DROID[3]:

a. includes a non-optimizing foreign code execution module.

b. includes a non-optimizing foreign code execution module dedicated to foreign state for each binary operation executed.

c. includes an optimizing binary translator.

d. includes a binary translator configured to translate foreign binary operations into optimized sequences of host operations.

e. includes a host CPU.

f. includes a host CPU configured to execute host operations.

g. includes a host CPU to execute the host operations.

h. includes a documentation generator.

i. includes a document generator configured to generate a set of documentations for optimized sequences of host operations.

j. includes a documentation tracker.

k. includes a documentation tracker configured to record host operation addresses at appointed points of the host operation sequences being executed.

l. includes a recovery mechanism.

m. includes a recovery mechanism configured to select a documentation in the set of documentations using a host operation address corresponding to the selected documentation.

23. Samsung is now and has been infringing and/or contributorily infringing the '750 patent, literally and under the doctrine of equivalents, by, among other things, using, offering to sell, selling, re-selling and/or importing products that are covered by one or more claims of the '750 patent. Such infringing products include, but are not limited to, cell phone products such as its Samsung Nexus S smartphones. An illustrative claim chart demonstrating how Samsung practices the method of claim 15 of the '750 patent using the Samsung Nexus smartphones with their Android operating system and software is attached as Exhibit B. Other Samsung products that use the Android operating system and software infringe in the same way. In fact, Samsung openly admits this:

> **OVERLAND PARK, Kan. – March 21, 2011** – Sprint (NYSE: S) extends its 4G device innovation lead once again with the upcoming availability of the 20th 4G device and fourth 4G phone, **Nexus S™ 4G1** from Google™. Coming to Sprint this spring, it will also be able to take advantage of the unprecedented controls and services enabled by Google Voice™ integration built into the Sprint Network.. Manufactured by Samsung Telecommunications America (Samsung Mobile), a leading global mobile phone provider and the No. 1 mobile phone provider in the United States2, Nexus S 4G comes packed with a pure Google experience using Android™ 2.3, Gingerbread, the fastest version of Android available for smartphones. It is powered by a 1GHz Samsung application processor that produces rich 3D-like graphics, faster upload and download times and supports HD-like multimedia content along with a dedicated Graphics Processing Unit (GPU) to make playing mobile games, browsing the Web and watching videos a fast, fluid and smooth experience.
>
> "Nexus S 4G shows the strong commitment Sprint has to Android, and when combined with our 4G network capabilities, it gives customers the option of a pure Google experience," said Fared Adib, vice president – Product Development, Sprint. "As the first 4G smartphone with Android 2.3, Nexus S 4G delivers on the promise of the advanced data capabilities

> of 4G to deliver an incredible Web browsing experience, offers quick and easy access to future Android updates and access to the services built into Google Voice."
>
> It is designed with Samsung's brilliant Super AMOLED™ touchscreen technology providing a premium viewing experience. The 4-inch Contour Display features a curved design for a more comfortable look and feel in the user's hand or along the side of the face. It also offers a screen that is bright with higher color contrast, meaning colors are incredibly vibrant and text is crisp at any size and produces less glare than on other smartphone displays when outdoors, so videos, pictures and games look their best and the sun won't wash them out
>
> Sprint Nexus S 4G customers will be among the first to receive Android software upgrades and new Google mobile apps. In many cases, the device will get the updates and new apps as soon as they are available.
>
> "We're excited to partner with Sprint on Nexus S 4G, which brings innovative hardware by Samsung and innovations on the Android platform, to create a powerful smartphone experience," said Andy Rubin, vice president of Engineering at Google.

(http://www.samsung.com/us/news/newsRead.do?news_seq=19830&page=18&gltype=localnews.) Samsung employees and corporate officers, including J.K. Shin, the President and Head of Mobile Communications at Samsung Electronics, and Kevin Packingham, Senior Vice President of Product Innovations, demonstrated a successor to the Nexus S, the Galaxy Nexus smartphone at a recent keynote also broadcast via webcam to introduce this latest model of Samsung phone with the Android OS. (http://androidandme.com/2011/10/news/video-miss-the-google-and-samsung-event-last-night-catch-the-full-keynote/)

24. Further, Samsung is a contributory infringer of the '750 patent because it knew at least when the original complaint was filed and, thus, knew then (and now knows) that the steps of the claimed method described in claim 15, for example, were carried out by users of Samsung's phones, such as the Samsung Nexus S smartphone, employing Android operating systems and software. Direct infringement by

customer/end-users is illustrated in the attached Exhibit B. Samsung's customers are direct infringers and Samsung contributes to their infringement in that Samsung's customers carry out each and every step of the method defined, for example, in claim 15, as illustrated in the claim chart, Exhibit B. Samsung is fully aware of such direct infringement and encourages, aids and assists it. Samsung also knows there are no substantial non-infringing uses of the accused products and Samsung knows its phones using the Android operating system and software are especially designed and made to use the Android operating system and software and, thus, are designed to practice, for example, the method of claim 15 of the '750 patent. Samsung knows this occurs when a Samsung customer/end-user operates a phone in the manner Samsung directs, instructs and teaches.

25. Samsung does more than simply sell products that use the Android operating system. It directs customers to use and shows them how to use the Android operating system and, thus, facilitates such use. Samsung's customers, in turn, actually use the identified cell phones to carry out each step of method claim 15, which is an act of direct infringement. The conditions of such use are known to Samsung and set forth in Exhibit B. Specifically, use of the method occurs when the phone is turned on and made functional. When the Android operating system starts up, the Dalvik Virtual Machine in the phone looks through the applications installed on the phone and builds a tree of dependencies. This dependency tree optimizes the byte code for every application and stores it in the Dalvik cache. The applications are then run using the optimized byte code. Samsung is fully aware that its phones using Android operating systems are designed to operate and do, in fact, operate in this way. The claimed method is performed when a user of the cellular phones operates the device for their

intended purpose using the Android operating system, e.g., allowing the Dalvik Virtual Machine to optimize the byte code for each application.

26. Further, Samsung's manufacture, sale, offer to sell and importation of the identified cell phones is a direct infringement of apparatus/system claim 1, for example, in that smartphones made and sold by Samsung such as the Nexus S smartphone include each element of claim 1, including a non-optimizing foreign code execution module, an optimizing binary translator, a host CPU, a documentation tracker and a recovery mechanism, all as required by claim 1.

27. Samsung makes the Nexus S cellular telephone.

28. Samsung's Nexus S phone includes both hardware and software components.

29. Samsung imports, sells, and offers to sell the Nexus S phone to customers in the United States.

30. Samsung customers use the Nexus S phones in the United States.

31. The Samsung Nexus S phone:

a. includes a binary translation system.

b. includes an optimizing translator.

c. includes a binary translated code translated from foreign code.

d. designates a set of recovery points in the optimized binary translated code during optimized translation of the foreign code.

e. generates a set of documentations during the optimized translation of the foreign code.

32. The Samsung Nexus S phone uses one of the documentations in the set of documentations corresponding to executed optimized binary translated code when an

exception arises during its execution to recover a foreign state corresponding to a recovery point for the exception.

33. The Samsung Nexus S:

a. includes a non-optimizing foreign code execution module.

b. includes a non-optimizing foreign code execution module dedicated to foreign state for each binary operation executed.

c. includes an optimizing binary translator.

d. includes a binary translator configured to translate foreign binary operations into optimized sequences of host operations.

e. includes a host CPU.

f. includes a host CPU configured to execute host operations.

g. includes a host CPU to execute the host operations.

h. includes a documentation generator.

i. includes a document generator configured to generate a set of documentations for optimized sequences of host operations.

j. includes a documentation tracker.

k. includes a documentation tracker configured to record host operation addresses at appointed points of the host operation sequences being executed.

l. includes a recovery mechanism.

m. includes a recovery mechanism configured to select a documentation in the set of documentations using a host operation address corresponding to the selected documentation.

PRAYER FOR RELIEF

WHEREFORE, Cascades prays for the following relief:

A. A judgment finding Motorola and Samsung have each infringed and contributorily infringed the '750 patent;

B. A judgment that the '750 patent is valid and enforceable;

C. A permanent injunction enjoining Motorola and Samsung, their agents, officers, assigns and others acting in concert with them, from infringing, inducing infringement of, and/or contributing to infringement of the '750 patent;

D. An award of damages adequate to compensate Cascades for the infringement of the '750 patent that has occurred;

E. An award of pre-judgment interest and post-judgment interest on the damages awarded;

F. A determination that this is an exceptional case and an award of Cascades' attorneys' fees pursuant to 35 U.S.C. § 285 and any other applicable statute or law, and an award to Cascades of its costs; and,

G. Such other relief as the Court deems equitable under the circumstances.

JURY DEMAND

Plaintiff demands a trial by jury on all issues triable to a jury.

Raymond P. Niro (rniro@nshn.com)
Arthur A. Gasey (gasey@nshn.com)
Paul C. Gibbons (gibbons@nshn.com)
Anna B. Folgers (afolgers@nshn.com
NIRO, HALLER & NIRO
181 W. Madison, Suite 4600
Chicago, IL 60602
(312) 236-0733
Fax: (312) 236-3137

Attorneys for Cascades Computer Innovation, LLC

CERTIFICATE OF SERVICE

The undersigned hereby certifies that on February 10, 2012 the foregoing **THIRD AMENDED COMPLAINT FOR PATENT INFRINGEMENT** was filed electronically with the Clerk of the Court for the Northern District of Illinois using the Court's Electronic Case Filing System, which will send notification to the registered participants of the ECF System as listed in the Court's Notice of Electronic Filing.

I certify that all parties in this case are represented by counsel who are CM/ECF participants.

_____

Attorneys for Cascades Computer Innovation, LLC

EXHIBIT A TO THIRD AMENDED COMPLAINT FOR PATENT INFRINGEMENT

US 7,065,750 Claim 15
v. Motorola Droid Family

US 7,065,750 to Babaian

US007065750B2

(12) **United States Patent**
Babaian et al.

(10) **Patent No.: US 7,065,750 B2**
(45) **Date of Patent: Jun. 20, 2006**

(54) **METHOD AND APPARATUS FOR PRESERVING PRECISE EXCEPTIONS IN BINARY TRANSLATED CODE**

(75) Inventors: **Boris A. Babaian**, Moscow (RU); **Andrew V. Yakushev**, Sergiev-Posad (RU); **Sergey A. Rozhkov**, Moscow (RU); **Vladimir M. Gushchin**, Moscow (RU)

(73) Assignee: **Elbrus International**, George Town (KY)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 723 days.

(21) Appl. No.: **09/838,552**

(22) Filed: **Apr. 18, 2001**

(65) **Prior Publication Data**

US 2002/0092002 A1 Jul. 11, 2002

Related U.S. Application Data

(63) Continuation-in-part of application No. 09/505,652, filed on Feb. 17, 2000

(60) Provisional application No. 60/120,348, filed on Feb. 17, 1999, provisional application No. 60/120,376, filed on Feb. 17, 1999, provisional application No. 60/120,380, filed on Feb. 17, 1999, provisional application No. 60/120,457, filed on Feb. 17, 1999, provisional application No. 60/120,458, filed on Feb. 17, 1999, provisional application No. 60/120,459, filed on Feb. 17, 1999, provisional application No. 60/120,508, filed on Feb. 17, 1999.

(51) **Int. Cl.**
G06F 9/455 (2006.01)

(52) **U.S. Cl.** **717/136**

(58) **Field of Classification Search** 717/129, 717/130, 136, 151, 158; 712/23, 24, 203, 212/212, 228, 230, 235, 244, 237; 714/15, 714/17, 20, 35, 48, 49, 702, 707, 747
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,537,559 A * 7/1996 Kase et al 712/244

(Continued)

OTHER PUBLICATIONS

Cherneff, A. Hendeg, M. Hookway, R. Reeve, C. Rubin, N. Tye, T. Bharadwaj, Ya S. Yates, J. "FX32 a profile-directed binary translator", Mar./Apr. 1998, IEEE Micro, p. 56-64, retrieved from IEEE Jul. 7, 2004.*

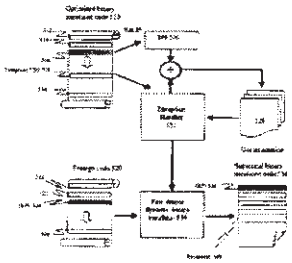
(Continued)

Primary Examiner – Wei Y. Zhen
Assistant Examiner – Mary Steelman
(74) *Attorney, Agent, or Firm* – Townsend and Townsend and Crew LLP

(57) **ABSTRACT**

Precise exceptions handling in the optimized binary translated code is achieved by transitioning execution to the non-optimized step-by-step foreign code execution means in accordance with one of the several coherent foreign states designated during the optimized translation of the foreign code. A method to improve the operation by avoiding complete foreign state updates in the optimized code, an apparatus to track the switching between the states and a method to recompute the complete foreign state in accordance to the current state identification, execution context and additional documentation provided during the translation time are proposed.

18 Claims, 4 Drawing Sheets



US 7,065,750 Claim 15

15. A method of recomputing a dedicated foreign state (A) in a binary translation system (B) from documentation generated by an optimizing translator (C) in a case of an exception arising during execution of optimized binary translated code translated from a foreign code (D), the method comprising: designating a set of recovery points in the optimized binary translated code during optimized translation of the foreign code, wherein each recovery point represents a foreign state (E), generating a set of documentations during the optimized translation of the foreign code (F), wherein each documentation in the set of documentations corresponds to a recovery point in the optimized binary translated code and describes operations required to calculate a corresponding foreign state for the recovery point (G) and using one of the documentations in the set of documentations corresponding to executed optimized binary translated code when an exception arises during its execution to recover a foreign state corresponding to a recovery point for the exception, wherein the foreign state is recovered by executing the operations for the one of the documentations (H).

US 7,065,750 Claim 15

foreign state (A)

```
/* Create the PC reconstruction slot if not already done */
extern ArmLIR *genCheckCommon(CompilationUnit *cUnit, int dOffset,
                              ArmLIR *branch,
                              ArmLIR *pcrLabel)
{
    /* Forget all def info (because we might rollback here.  Bug #2367397 */
    dvmCompilerResetDefTracking(cUnit);

    /* Set up the place holder to reconstruct this Dalvik PC */
    if (pcrLabel == NULL) {
        int dPC = (int) (cUnit->method->insns + dOffset);
        pcrLabel = dvmCompilerNew(sizeof(ArmLIR), true);
        pcrLabel->opCode = kArmPseudoPCReconstructionCell;
        pcrLabel->operands[0] = dPC;
        pcrLabel->operands[1] = dOffset;
        /* Insert the place holder to the growable list */
        dvmInsertGrowableList(&cUnit->pcReconstructionList, pcrLabel);
    }
    /* Branch to the PC reconstruction code */
    branch->generic.target = (LIR *) pcrLabel;
    return pcrLabel;
}
```

Note: Dalvik PC – Program Counter for the interpreted (foreign) code

Dalvik code analysis, CodegenCommon.c

US 7,065,750 Claim 15

a binary translation system

Note: JIT is part of all the releases of Android since 2.2. Source – code analysis

Dalvik JIT

Posted by Tim Bray on 25 May 2010 at 2:57 PM

[This post is by Dan Bornstein, virtual-machine wrangler. — Tim Bray]

As the tech lead for the Dalvik team within the Android project, I spend my time working on the virtual machine (VM) and core class libraries that sit beneath the Android application framework. This layer is mostly invisible to end users, but done right, it helps make Android devices run smoothly and improves developer productivity.

The 2.2 release is particularly pleasing to me, as it is the first release since before 1.0 in which we have been able to deliver significantly new VM technology. And unlike much of what my team and I do, it is something that can be experienced directly by end users.

"Dalvik" isn't exactly a household word (at least in my country), and most people wouldn't know a virtual machine if it hit them in the face, but when you tell them you were able to make their existing device work better — run faster, use less battery — they will actually take notice!

What Makes This Possible?

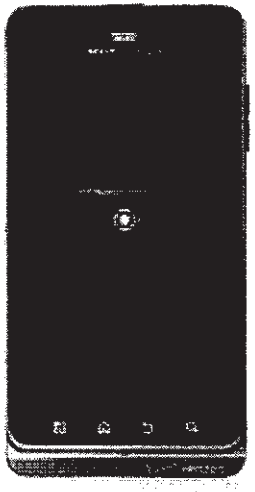
We added a Just In Time (JIT) compiler to the Dalvik VM. The JIT is a software component which takes application code, analyzes it, and actively translates it into a form that runs faster, doing so while the application continues to run. If you want to learn more about the design of the Dalvik JIT, please watch the excellent talk from Google I/O 2010 given by my colleagues Bill Buzbee and Ben Cheng, which should be posted to YouTube very soon.

From: Android developers blog, http://android-developers.blogspot.com/2010/05/dalvik-jit.html

US 7,065,750 Claim 15

a binary translation system

DROID 3 BY MOTOROLA



ANDROID™ PLATFORM
Android 2.3 (Gingerbread)

Source: http://www.motorola.com/Consumers/US-EN/Consumer-Product-and-Services/Mobile-Phones/ci.DROID-3-by-MOTOROLA-US-EN.alt

US 7,065,750 Claim 15

optimizing translator (C)

Dalvik Interpreter

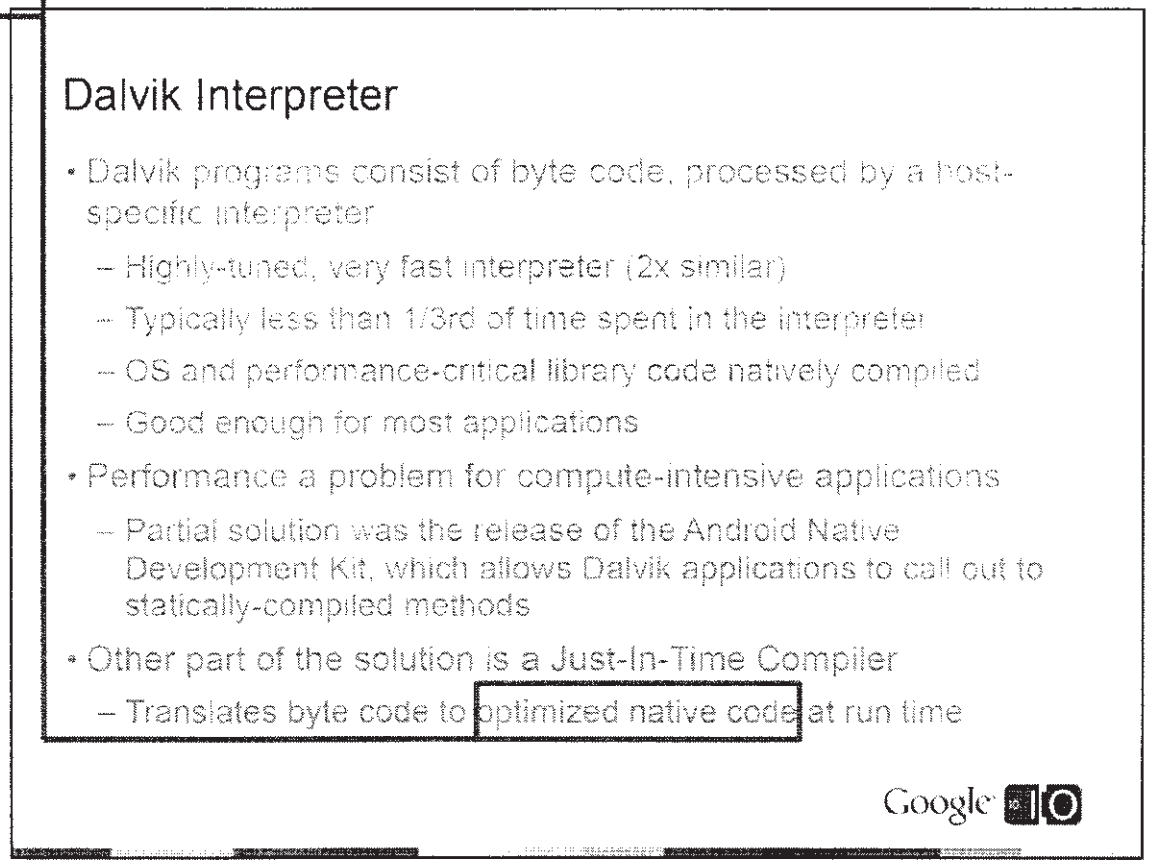
- Dalvik programs consist of byte code, processed by a host-specific interpreter
  - Highly-tuned, very fast interpreter (2x similar)
  - Typically less than 1/3rd of time spent in the interpreter
  - OS and performance-critical library code natively compiled
  - Good enough for most applications
- Performance a problem for compute-intensive applications
  - Partial solution was the release of the Android Native Development Kit, which allows Dalvik applications to call out to statically-compiled methods
- Other part of the solution is a Just-In-Time Compiler
  - Translates byte code to optimized native code at run time

Google I/O

Source: http://dl.google.com/googleio/2010/android-jit-compiler-androids-dalvik-vm.pdf

US 7,065,750 Claim 15

binary translated code translated from a foreign code (D),



Source: http://dl.google.com/googleio/2010/android-jit-compiler-androids-dalvik-vm.pdf

US 7,065,750 Claim 15

designating a set of recovery points in the optimized binary translated code during optimized translation of the foreign code, wherein each recovery point represents a foreign state (E)

```
/* Create the PC reconstruction slot if not already done */
extern ArmLIR *genCheckCommon(CompilationUnit *cUnit, int dOffset,
                              ArmLIR *branch,
                              ArmLIR *pcrLabel)
{
    /* Forget all def info (because we might rollback here.  Bug #2367397 */
    dvmCompilerResetDefTracking(cUnit);

    /* Set up the place holder to reconstruct this Dalvik PC */
    if (pcrLabel == NULL) {
        int dPC = (int) (cUnit->method->insns + dOffset);
        pcrLabel = dvmCompilerNew(sizeof(ArmLIR), true);
        pcrLabel->opCode = kArmPseudoPCReconstructionCell;
        pcrLabel->operands[0] = dPC;
        pcrLabel->operands[1] = dOffset;
        /* Insert the place holder to the growable list */
        dvmInsertGrowableList(&cUnit->pcReconstructionList, pcrLabel);
    }
    /* Branch to the PC reconstruction code */
    branch->generic.target = (LIR *) pcrLabel;
    return pcrLabel;
}
```

Dalvik code analysis, CodegenCommon.c

US 7,065,750 Claim 15

generating a set of documentations during the optimized translation of the foreign code (F),

```
/* Load the Dalvik PC into r0 and jump to the specified target */
static void handlePCReconstruction(CompilationUnit *cUnit,
                                   ArmLIR *targetLabel)
{
    ArmLIR **pcrLabel =
        (ArmLIR **) cUnit->pcReconstructionList.elemList;
    int numElems = cUnit->pcReconstructionList.numUsed;
    int i;
    for (i = 0; i < numElems; i++) {
        dvmCompilerAppendLIR(cUnit, (LIR *) pcrLabel[i]);
        /* r0 = dalvik PC */
        loadConstant(cUnit, r0, pcrLabel[i]->operands[0]);
        genUnconditionalBranch(cUnit, targetLabel);
    }
}
```

Dalvik code analysis, CodegenDriver.c

US 7,065,750 Claim 15

each documentation in the set of documentations corresponds to a recovery point in the optimized binary translated code and describes operations required to calculate a corresponding foreign state for the recovery point

```
/* Load the Dalvik PC into r0 and jump to the specified target */
static void handlePCReconstruction(CompilationUnit *cUnit,
                                   ArmLIR *targetLabel)
{
    ArmLIR **pcLabel =
        (ArmLIR **) cUnit->pcReconstructionList.elemList;
    int numElems = cUnit->pcReconstructionList.numUsed;
    int i;
    for (i = 0; i < numElems; i++) {
        dvmCompilerAppendLIR(cUnit, (LIR *) pcLabel[i]);
        /* r0 = dalvik PC */
        loadConstant(cUnit, r0, pcLabel[i]->operands[0]);
        genUnconditionalBranch(cUnit, targetLabel);
    }
}
```

Dalvik code analysis, CodegenDriver.c

US 7,065,750 Claim 15

using one of the documentations in the set of documentations corresponding to executed optimized binary translated code when an exception arises during its execution to recover a foreign state corresponding to a recovery point for the exception, wherein the foreign state is recovered by executing the operations for the one of the documentations [H].

```
/* Now create a special block to host PC reconstruction code */
lastBB->next = dvmCompilerNewBB(kPCReconstruction);
lastBB = lastBB->next;
lastBB->id = numBlocks++;

/* And one final block that publishes the PC and raise the exception */
lastBB->next = dvmCompilerNewBB(kExceptionHandling);
lastBB = lastBB->next;
lastBB->id = numBlocks++;
```

Continued

Dalvik code analysis, FrontEnd. c

US 7,065,750 Claim 15

using one of the documentations in the set of documentations corresponding to executed optimized binary translated code when an exception arises during its execution to recover a foreign state corresponding to a recovery point for the exception, wherein the foreign state is recovered by executing the operations for the one of the documentations (H).

```
void dvmCompilerMIR2LIR(CompilationUnit *cUnit)
{   ......
```

```
            case kExceptionHandling:
                labelList[i].opCode = kArmPseudoEHBlockLabel;
                if (cUnit->pcReconstructionList.numUsed) {
                    loadWordDisp(cUnit, rGLUE, offsetof(InterpState,
                                 jitToInterpEntries.dvmJitToInterpPunt),
                                 r1);
                    opReg(cUnit, kOpBlx, r1);
                }
                break;

 * 3) dvmJitToInterpPunt: use the fast interpreter to execute the next
 *    instruction(s) and stay there as long as it is appropriate to return
 *    to the compiled land. This is used when the jit'ed code is about to
 *    throw an exception.
```

Continued

Dalvik code analysis, codegendriver.c, InterpDefs.h

US 7,065,750 Claim 15

using one of the documentations in the set of documentations corresponding to executed optimized binary translated code when an exception arises during its execution to recover a foreign state corresponding to a recovery point for the exception, wherein the foreign state is recovered by executing the operations for the one of the documentations (H).

```
    .global dvmJitToInterpPunt
dvmJitToInterpPunt:
    ldr     r10, [rGLUE, #offGlue_self]  @ callee saved r10 <- glue->self
    mov     r2,#kSVSPunt                 @ r2<- interpreter entry point
    mov     r3, #0
    str     r3, [r10, #offThread_inJitCodeCache] @ Back to the interp land
    b       jitSVShadowRunEnd            @ doesn't return
```

Dalvik code analysis, footer.S

EXHIBIT B TO THIRD AMENDED COMPLAINT FOR PATENT INFRINGEMENT

US 7,065,750 Claim 15
v. Nexus/Galaxy Family

US 7,065,750 to Babaian

(12) **United States Patent**
Babaian et al.

(10) **Patent No.: US 7,065,750 B2**
(45) **Date of Patent: Jun. 20, 2006**

(54) **METHOD AND APPARATUS FOR PRESERVING PRECISE EXCEPTIONS IN BINARY TRANSLATED CODE**

(75) Inventors: **Boris A. Babaian**, Moscow (RU); **Andrew V. Yakushev**, Sergiev-Posad (RU); **Sergey A. Rozhkov**, Moscow (RU); **Vladimir M. Gushchin**, Moscow (RU)

(73) Assignee: **Elbrus International**, George Town (KY)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 723 days

(21) Appl. No.: **09/838,552**

(22) Filed: **Apr. 18, 2001**

(65) **Prior Publication Data**

US 2002/0092002 A1 Jul. 11, 2002

Related U.S. Application Data

(63) Continuation-in-part of application No. 09/505,652, filed on Feb. 17, 2000

(60) Provisional application No. 60/120,348, filed on Feb. 17, 1999, provisional application No. 60/120,376, filed on Feb. 17, 1999, provisional application No. 60/120,380, filed on Feb. 17, 1999, provisional application No. 60/120,457, filed on Feb. 17, 1999, provisional application No. 60/120,458, filed on Feb. 17, 1999, provisional application No. 60/120,459, filed on Feb. 17, 1999, provisional application No. 60/120,504, filed on Feb. 17, 1999.

(51) **Int. Cl.**
G06F 9/455 (2006.01)

(52) **U.S. Cl.** **717/136**

(58) **Field of Classification Search** 717/129, 717/130, 136, 151, 158; 712/23, 24, 203, 212, 228, 230, 235, 244, 237; 714/15, 714/17, 20, 35, 48, 49, 702, 707, 747
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,537,559 A * 7/1996 Kato et al. 712/244

(Continued)

OTHER PUBLICATIONS

Chernoff, A. Herdeg, M. Hookway, R. Reeve, C. Rubin, N. Tye, T. Bharadwaj, Yo S. Yates, J. "FX!32 a profile-directed binary translator", Mar.-Apr. 1998, IEEE Micro, p. 56-64, retrieved from IEEE Jul. 7, 2004 *

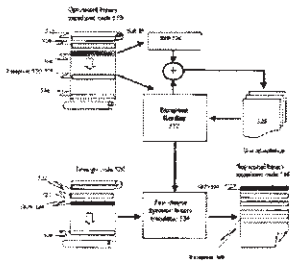
(Continued)

Primary Examiner—Wei Y. Zhen
Assistant Examiner—Mary Steelman
(74) *Attorney, Agent, or Firm*—Townsend and Townsend and Crew LLP

(57) **ABSTRACT**

Precise exceptions handling in the optimized binary translated code is achieved by transitioning execution to the non-optimized step-by-step foreign code execution means in accordance with one of the several coherent foreign states designated during the optimized translation of the foreign code. A method to improve the operation by avoiding complete foreign state updates in the optimized code, an apparatus to track the switching between the states and a method to recompute the complete foreign state in accordance to the current state identification, execution context and additional documentation provided during the translation time are proposed.

18 Claims, 4 Drawing Sheets



US 7,065,750 Claim 15

15. A method of recomputing a dedicated foreign state (A) in a binary translation system (B) from documentation generated by an optimizing translator (C) in a case of an exception arising during execution of optimized binary translated code translated from a foreign code (D), the method comprising: designating a set of recovery points in the optimized binary translated code during optimized translation of the foreign code, wherein each recovery point represents a foreign state (E); generating a set of documentations during the optimized translation of the foreign code (F), wherein each documentation in the set of documentations corresponds to a recovery point in the optimized binary translated code and describes operations required to calculate a corresponding foreign state for the recovery point (G); and using one of the documentations in the set of documentations corresponding to executed optimized binary translated code when an exception arises during its execution to recover a foreign state corresponding to a recovery point for the exception, wherein the foreign state is recovered by executing the operations for the one of the documentations (H).

US 7,065,750 Claim 15

foreign state (A)

```
/* Create the PC reconstruction slot if not already done */
extern ArmLIR *genCheckCommon(CompilationUnit *cUnit, int dOffset,
                              ArmLIR *branch,
                              ArmLIR *pcrLabel)
{
    /* Forget all def info (because we might rollback here.  Bug #2367397 */
    dvmCompilerResetDefTracking(cUnit);

    /* Set up the place holder to reconstruct this Dalvik PC */
    if (pcrLabel == NULL) {
        int dPC = (int) (cUnit->method->insns + dOffset);
        pcrLabel = dvmCompilerNew(sizeof(ArmLIR), true);
        pcrLabel->opCode = kArmPseudoPCReconstructionCell;
        pcrLabel->operands[0] = dPC;
        pcrLabel->operands[1] = dOffset;
        /* Insert the place holder to the growable list */
        dvmInsertGrowableList(&cUnit->pcReconstructionList, pcrLabel);
    }
    /* Branch to the PC reconstruction code */
    branch->generic.target = (LIR *) pcrLabel;
    return pcrLabel;
}
```

Note: Dalvik PC – Program Counter for the interpreted (foreign) code

Dalvik code analysis, CodegenCommon.c

US 7,065,750 Claim 15

a binary translation system

Note: JIT is part of all the releases of Android since 2.2. Source – code analysis

Dalvik JIT

Posted by Tim Bray on 25 May 2010 at 2:57 PM

[This post is by Dan Bornstein, virtual-machine wrangler. — Tim Bray]

As the tech lead for the Dalvik team within the Android project, I spend my time working on the virtual machine (VM) and core class libraries that sit beneath the Android application framework. This layer is mostly invisible to end users, but done right, it helps make Android devices run smoothly and improves developer productivity.

The 2.2 release is particularly pleasing to me, as it is the first release since before 1.0 in which we have been able to deliver significantly new VM technology. And unlike much of what my team and I do, it is something that can be experienced directly by end users.

"Dalvik" isn't exactly a household word (at least in my country), and most people wouldn't know a virtual machine if it hit them in the face, but when you tell them you were able to make their existing device work better — run faster, use less battery — they will actually take notice!

What Makes This Possible?

We added a Just In Time (JIT) compiler to the Dalvik VM. The JIT is a software component which takes application code, analyzes it, and actively translates it into a form that runs faster, doing so while the application continues to run. If you want to learn more about the design of the Dalvik JIT, please watch the excellent talk from Google I/O 2010 given by my colleagues Bill Buzbee and Ben Cheng, which should be posted to YouTube very soon.

From: Android developers blog, http://android-developers.blogspot.com/2010/05/dalvik-jit.html

US 7,065,750 Claim 15

a binary translation system

Meet the Nexus S with Android 2.3

Published: December 6, 2010

Samsung and Google deliver Nexus S, the world's first handset to feature the latest version of Google's Android™ platform. Powered by Android 2.3, Gingerbread, this smartphone is packed with powerful technology and the latest in hardware features.

Source: http://www.samsung.com/us/article/meet-the-nexus-s-with-android-2-3

US 7,065,750 Claim 15

optimizing translator (G)

Dalvik Interpreter

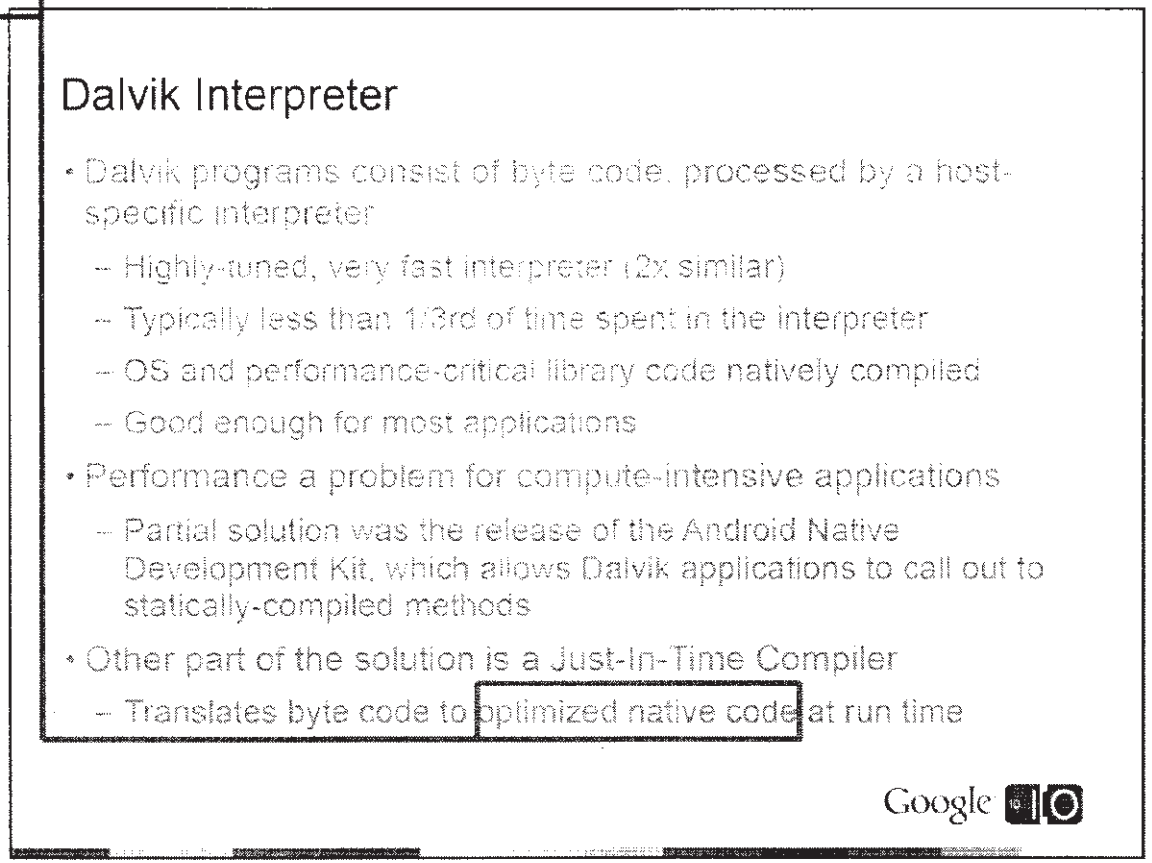
- Dalvik programs consist of byte code, processed by a host-specific interpreter
  - Highly-tuned, very fast interpreter (2x similar)
  - Typically less than 1/3rd of time spent in the interpreter
  - OS and performance-critical library code natively compiled
  - Good enough for most applications
- Performance a problem for compute-intensive applications
  - Partial solution was the release of the Android Native Development Kit, which allows Dalvik applications to call out to statically-compiled methods
- Other part of the solution is a Just-In-Time Compiler
  - Translates byte code to optimized native code at run time

Google I/O

Source: http://dl.google.com/googleio/2010/android-jit-compiler-androids-dalvik-vm.pdf

US 7,065,750 Claim 15

binary translated code translated from a foreign code (D),



Source: http://dl.google.com/googleio/2010/android-jit-compiler-androids-dalvik-vm.pdf

US 7,065,750 Claim 15

designating a set of recovery points in the optimized binary translated code during optimized translation of the foreign code, wherein each recovery point represents a foreign state (E)

```
/* Create the PC reconstruction slot if not already done */
extern ArmLIR *genCheckCommon(CompilationUnit *cUnit, int dOffset,
                              ArmLIR *branch,
                              ArmLIR *pcrLabel)
{
    /* Forget all def info (because we might rollback here.  Bug #2367397 */
    dvmCompilerResetDefTracking(cUnit);

    /* Set up the place holder to reconstruct this Dalvik PC */
    if (pcrLabel == NULL) {
        int dPC = (int) (cUnit->method->insns + dOffset);
        pcrLabel = dvmCompilerNew(sizeof(ArmLIR), true);
        pcrLabel->opCode = kArmPseudoPCReconstructionCell;
        pcrLabel->operands[0] = dPC;
        pcrLabel->operands[1] = dOffset;
        /* Insert the place holder to the growable list */
        dvmInsertGrowableList(&cUnit->pcReconstructionList, pcrLabel);
    }
    /* Branch to the PC reconstruction code */
    branch->generic.target = (LIR *) pcrLabel;
    return pcrLabel;
}
```

Dalvik code analysis, CodegenCommon.c

US 7,065,750 Claim 15

generating a set of documentations during the optimized translation of the foreign code (F),

```
/* Load the Dalvik PC into r0 and jump to the specified target */
static void handlePCReconstruction(CompilationUnit *cUnit,
                                   ArmLIR *targetLabel)
{
    ArmLIR **pcrLabel =
        (ArmLIR **) cUnit->pcReconstructionList.elemList;
    int numElems = cUnit->pcReconstructionList.numUsed;
    int i;
    for (i = 0; i < numElems; i++) {
        dvmCompilerAppendLIR(cUnit, (LIR *) pcrLabel[i]);
        /* r0 = dalvik PC */
        loadConstant(cUnit, r0, pcrLabel[i]->operands[0]);
        genUnconditionalBranch(cUnit, targetLabel);
    }
}
```

Dalvik code analysis, CodegenDriver.c

US 7,065,750 Claim 15

each documentation in the set of documentations corresponds to a recovery point in the optimized binary translated code and describes operations required to calculate a corresponding foreign state for the recovery point

```
/* Load the Dalvik PC into r0 and jump to the specified target */
static void handlePCReconstruction(CompilationUnit *cUnit,
                                   ArmLIR *targetLabel)
{
    ArmLIR **pcrLabel =
        (ArmLIR **) cUnit->pcReconstructionList.elemList;
    int numElems = cUnit->pcReconstructionList.numUsed;
    int i;
    for (i = 0; i < numElems; i++) {
        dvmCompilerAppendLIR(cUnit, (LIR *) pcrLabel[i]);
        /* r0 = dalvik PC */
        loadConstant(cUnit, r0, pcrLabel[i]->operands[0]);
        genUnconditionalBranch(cUnit, targetLabel);
    }
}
```

Dalvik code analysis, CodegenDriver.c

US 7,065,750 Claim 15

using one of the documentations in the set of documentations corresponding to executed optimized binary translated code when an exception arises during its execution to recover a foreign state corresponding to a recovery point for the exception, wherein the foreign state is recovered by executing the operations for the one of the documentations (H).

```
/* Now create a special block to host PC reconstruction code */
lastBB->next = dvmCompilerNewBB(kPCReconstruction);
lastBB = lastBB->next;
lastBB->id = numBlocks++;

/* And one final block that publishes the PC and raise the exception */
lastBB->next = dvmCompilerNewBB(kExceptionHandling);
lastBB = lastBB->next;
lastBB->id = numBlocks++;
```

Continued

Dalvik code analysis, FrontEnd. c

US 7,065,750 Claim 15

using one of the documentations in the set of documentations corresponding to executed optimized binary translated code when an exception arises during its execution to recover a foreign state corresponding to a recovery point for the exception, wherein the foreign state is recovered by executing the operations for the one of the documentations (M).

```
void dvmCompilerMIR2LIR(CompilationUnit *cUnit)
{   ......
```

```
            case kExceptionHandling:
                labelList[i].opCode = kArmPseudoEHBlockLabel;
                if (cUnit->pcReconstructionList.numUsed) {
                    loadWordDisp(cUnit, rGLUE, offsetof(InterpState,
                                 jitToInterpEntries.dvmJitToInterpPunt),
                                 r1);
                    opReg(cUnit, kOpBlx, r1);
                }
                break;

 * 3) dvmJitToInterpPunt: use the fast interpreter to execute the next
 *    instruction(s) and stay there as long as it is appropriate to return
 *    to the compiled land. This is used when the jit'ed code is about to
 *    throw an exception.
```

Continued

Dalvik code analysis, codegendriver.c, InterpDefs.h

US 7,065,750 Claim 15

using one of the documentations in the set of documentations corresponding to executed optimized binary translated code when an exception arises during its execution to recover a foreign state corresponding to a recovery point for the exception, wherein the foreign state is recovered by executing the operations for the one of the documentations [6d].

```
    .global dvmJitToInterpPunt
dvmJitToInterpPunt:
    ldr     r10, [rGLUE, #offGlue_self]  @ callee saved r10 <- glue->self
    mov     r2,#kSVSPunt                 @ r2<- interpreter entry point
    mov     r3, #0
    str     r3, [r10, #offThread_inJitCodeCache] @ Back to the interp land
    b       jitSVShadowRunEnd            @ doesn't return
```

Dalvik code analysis, footer.S

US 7,065,750 Applicability to other Samsung products

US 7,065,750 is also applicable to at least following Samsung products

- Nexus S
- Gem
- Gem Touchscreen
- Continuum
- Mesmerize i500
- Fascinate
- Showcase i500
- Showcase Galaxy S
- Acclaim
- Galaxy Indulge
- Capacitive
- Infuse 4G
- Sidekick 4G
- S 4G
- Vibrant
- Epic 4G
- S 4G
- Replenish
- Galaxy Prevail
- Intercept Galaxy Tab (tablet)
- Galaxy S (tablet)