(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
18 May 2006 (18.05.2006)

PCT

(10) International Publication Number
WO 2006/052946 A2

(51) International Patent Classification:
G06F 7/00 (2006.01)

(21) International Application Number:
PCT/US2005/040388

(22) International Filing Date:
8 November 2005 (08.11.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/626,252	8 November 2004 (08.11.2004)	US
60/626,292	8 November 2004 (08.11.2004)	US
60/626,293	8 November 2004 (08.11.2004)	US

(71) Applicant *(for all designated States except US)*: **INNOPATH SOFTWARE, INC.** [US/US]; 400 Caribbean Drive, Sunnyvale, California 94089 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **GU, Jinsheng** [CN/US]; 400 Caribbean Drive, Sunnyvale, California 94089 (US). **MANAPETTY, Premjith** [IN/US]; 400 Caribbean Drive, Sunnyvale, California 94089 (US).

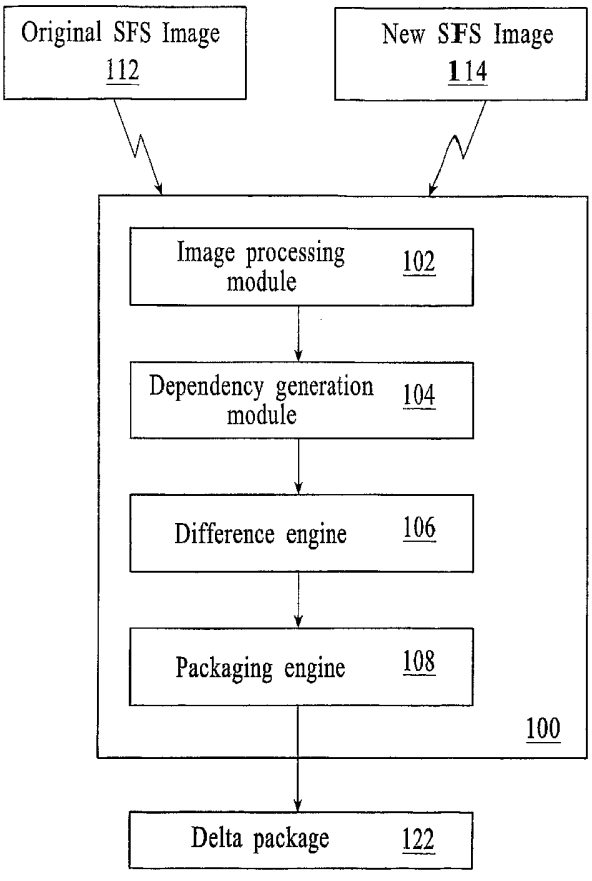
(74) Agents: **COURTNEY, Barbara** et al.; Courtney Staniford & Gregory LLP, P.o. Box 9686, San Jose, California 95157 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI,

[Continued on next page]

(54) Title: STATIC FILE SYSTEM DIFFERENCING AND UPDATING



(57) Abstract: Systems and methods are provided for static file system (SFS) differencing and updating. The differencing and updating includes portion-level differencing and block-level updating of units of an original image (referred to as blocks). The differencing and updating splits SFS images into portions based on block information and the image structure. A delta file is generated for each portion (portion-level differencing) of the new SFS image; the delta file includes information of differences between the portion of the new SFS image and the portion of the original SFS image to which the new SFS image portion corresponds. The delta files are transferred to a device where the target SFS image of the device is updated block-by-block using information of the delta files. The block-by-block update reconstructs all portions of the new SFS image in a device block in host device RAM and writes the reconstructed block into host device ROM.

WO 2006/052946 A2

FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Static File System Differencing and Updating

Inventors:

Jinsheng Gu

Premjith Manapetty

Related Application

This application claims the benefit of United States Patent Application Numbers 60/626,252, 60/626,292, and 60/626,293, all filed November 8, 2004. This application is related to United States Patent Application Numbers (number not yet assigned; attorney docket number DOGO.P020; entitled "Updating Compressed Read-Only Memory File System (CRAMFS) Images") and (number not yet assigned; attorney docket number DOGO.P021; entitled "Reorganizing Images in Static File System Differencing and Updating"), both filed November 8, 2005.

Technical Field

The disclosed embodiments relate to updating static file system images using difference files.

Background

Software running on a processor, microprocessor, and/or processing unit to provide certain functionality often changes over time and also increases in complexity. The changes can result from the need to correct bugs, or errors, in the software files, adapt to evolving technologies, or add new features, to name a few. In particular, software hosted on mobile processing devices, for example mobile wireless devices, often includes numerous software bugs that require correction. Software includes one or more computer programs, algorithms, files, and/or code pertaining to operation of the host device. Software can be divided into smaller units that are referred to as modules or components.

Portable processor-based devices like mobile processing devices typically include a real-time operating system (RTOS) in which all software components of the device are linked as a single large executable image. Further, file system support has become common recently due to the availability of compact storage and more demanding functionalities in these mobile wireless devices. In addition, the single

large image needs to be preloaded, or embedded, into the device using a slow communication link like a radio, infrared, or serial link.

Obstacles to updating the software of mobile processing devices include the time, bandwidth, and cost associated with delivering the updated file to the device, as well as limited resources of the device available for use in updating new files once received. As one example, static file system images are required to be updated in-place (e.g., in read-only memory (ROM)) because of resource limitations (e.g., limitations relating to RAM, ROM, time for downloading the image, power consumption, etc) of the host device. Consequently, there is a need for generating difference files that are sized appropriately for efficient transfer to host devices and for use in updating software of resource-limited devices.

Incorporation By Reference

Each publication, patent, and/or patent application mentioned in this specification is herein incorporated by reference in its entirety to the same extent as if each individual publication and/or patent application was specifically and individually indicated to be incorporated by reference.

Brief Description of the Figures

Figure 1 is a block diagram of a static file system (SFS) differencing system, under an embodiment.

Figure 2 is a flow diagram for SFS differencing, under an embodiment.

Figure 3 is a flow diagram for SFS differencing, under another embodiment.

Figure 4 is an example SFS image following splitting of the image blocks into portions, under an embodiment.

Figure 5 is a block diagram of an SFS differencing and updating system, under an embodiment.

Figure 6 is a flow diagram for SFS updating, under an embodiment.

Figure 7 is a flow diagram for in-place updating of SFS images in devices, under an embodiment.

In the drawings, the same reference numbers identify identical or substantially similar elements or acts. To easily identify the discussion of any particular element or act, the most significant digit or digits in a reference number

refer to the Figure number in which that element is first introduced (e.g., element 100 is first introduced and discussed with respect to Figure 1).

Detailed Description

Systems and methods are provided for static file system (SFS) differencing and updating. The systems and methods for SFS differencing and updating include portion-level differencing and block-level updating of update units of an original image (referred to as device blocks or blocks) as described below. The differencing and updating of an embodiment splits SFS images into a series of portions based on block information and the SFS image structure. A delta file is generated for each portion (portion-level differencing) of the new SFS image, and the delta file includes information of differences between the portion of the new SFS image and the portion(s) of the original SFS image to which the new SFS image portion corresponds (a new portion can depend on more than one original portion, and in addition, the original portion(s) might not be in the same location as the new portion). The delta files are transferred to a device for use in updating images of the device to the new SFS image. The target SFS image of the device is updated block-by-block using information of the delta files. The block-by-block update of an embodiment reconstructs all portions of the new SFS image in a device block in random access memory (RAM) of the host device and then writes the reconstructed block into ROM of the host device.

A SFS, also referred to as a read-only file system, is a file system that can not be modified during run time. Examples of SFSs include but are not limited to the Symbian Z drive (also referred to as the "ROM drive"), the Linux compressed ROM file system (CRAMFS), the encrypted file systems, and file systems that might store operating system executables, built-in applications, essential multimedia information, and default language files, to name a few.

The SFS differencing and updating of embodiments described below receives images of a static file system. The images, which include an original image and a new image, each include a number of blocks, for example super blocks, data blocks, etc. The SFS differencing splits the images by using information of the blocks to split the images into multiple portions. Differences are determined between content of the images by determining differences between the portions of the original image and the new image, where the differences are generated for each

portion of the new image. The differences include byte-level differences between the portions, but are not so limited. A delta file is generated that includes the differences for each portion of the new image.

The SFS differencing and updating of embodiments includes updating by which the SFS of an image hosted on a portable device is updated in-place on the portable device. The updating receives the delta file at a portable device via at least one coupling. Dependent ones of the original portions hosted on the portable device are assembled, and at least one of the is identified that corresponds to the delta file received, where the new portion location in ROM is encoded in the SFS delta package associated with its corresponding delta file in an embodiment. The updating reconstructs at least one new portion on the portable device that corresponds to the delta file identified. The reconstructed new portions of the new image are written to the read-only memory (ROM) of the portable device.

In the following description, numerous specific details are introduced to provide a thorough understanding of, and enabling description for, embodiments of the SFS differencing and updating. One skilled in the relevant art, however, will recognize that the SFS differencing and updating can be practiced without one or more of the specific details, or with other components, systems, etc. In other instances, well-known structures or operations are not shown, or are not described in detail, to avoid obscuring aspects of the SFS differencing and updating.

Figure 1 is a block diagram of a SFS differencing system 100, under an embodiment. The SFS differencing system includes an image processing module 102, a dependency generation module 104, a difference engine or module 106, and a packaging engine or module 108, but is not so limited. The dependency generation module 104 of an embodiment is coupled to the image processing module 102. The difference engine 106 of an embodiment is coupled to the dependency generation module 104. The packaging engine 108 of an embodiment is coupled to the difference engine 106.

The SFS differencing system 100 of an embodiment couples among components of a host computer system (not shown), where the components can include at least one of a processor, a controller, a memory device, and/or a bus, but are not so limited. One or more of the components or modules 102-108 of the SFS differencing system 100 run under control of at least one algorithm, program, or routine. A host computer system processor couples among the components of the

host computer system and the components 102-108 of the SFS differencing system 100 under program control. While the image processing module 102, dependency generation module 104, difference engine 106, and packaging engine 108 are shown as separate blocks, some or all of these blocks 102-108 can be monolithically integrated onto a single chip, distributed among a number of chips or components of a host system, and/or provided by one or some combination of programs or algorithms. The programs or algorithms when present can be implemented in software algorithm(s), firmware, hardware, and any combination of software, firmware, and hardware.

The term "module" as generally used herein refers to a logically separable part of a program, and is interchangeable with one or more terms including software, algorithm, program, component, and unit. The term "component" as generally used herein refers to one or more of the parts that make up a system; a component may be software and/or hardware and may be subdivided into other components. The terms "module", "component", and "unit" may be used interchangeably or defined to be subelements of one another in different ways depending on the context. The term "processor" as generally used herein refers to any logic processing unit, such as one or more central processing units (CPUs), digital signal processors (DSPs), application-specific integrated circuits (ASIC), etc.

In operation the SFS differencing system 100 receives at least one original SFS image 112 and at least one new SFS image 114 and performs portion-level differencing to generate one or more delta files as described below. The delta files are assembled into a delta package 122 for transfer to a portable or mobile device, also referred to as a client device. These differences include byte-level differences between one or more portions of blocks of the compared images, but are not so limited. The SFS differencing system 100 generates the delta file in a processor-based or computer system or running under a processor-based or computer system. The computer system on which or under which the SFS differencing system runs includes any collection of computing components and devices operating together, as is known in the art. The computer system can also be a component or subsystem within a larger computer system or network.

Contents of the delta file provide an efficient representation of the differences between the original image 112 and the new image 114. The delta file can include meta-data along with actual data of replacement and/or insertion

operations that represent the differences between the new or current version of the associated file and previous versions of the file, as described in the United States Patent Number 6,925,467 issued to InnoPath Software, Inc. of Sunnyvale, California on August 2, 2005. The SFS differencing system 100 provides any differences between the original image 112 and the new image 114 in delta files of the delta package 122 using a minimum number of bytes and a pre-defined format or protocol, thereby providing a delta file optimized in space.

The SFS differencing system 100 performs portion-level differencing, and **Figure 2** is a flow diagram for SFS differencing 200, under an embodiment. The SFS differencing is performed, for example, using the SFS differencing system 100 described above and elsewhere herein. The SFS differencing 200 receives 202 images of a static file system. The images include an original image and a new image, but are not so limited. Each image also includes a number of blocks, for example super blocks, data blocks, etc. The blocks are of a pre-specified size (e.g., 64 KB, 128 KB, etc.) but are not so limited. The blocks of the received images are split 204 into a number of portions using information of the blocks for example. The SFS differencing 200 determines 206 differences between content of the images by determining differences between the portions of each of the original image and the new image. A delta file is generated 208 that includes information of the differences in content or data between each portion of the new image and one or more aligned portions of the original image.

As another example, **Figure 3** is a flow diagram for SFS differencing 300, under an embodiment. The SFS differencing 300 receives 302 images of a static file system, as described above with reference to Figure 2. The received original image is divided 304 into one or more original sections or portions, and the received new image is divided 304 into one or more new sections or portions. The SFS differencing identifies 306 dependency alignments between the original sections and the new sections. A delta file is generated 308 for at least one of the new sections. The delta file includes but is not limited to differences between a new section and one or more original section(s), where the new section depends on the original section(s). The delta files of alternative embodiments may include differences between at least one new section and at least one original section on which the new sections depend.

The image processing module 102, as described above with reference to **Figure 1**, receives each of an original SFS image 112 and a new SFS image 114. The image processing module 102 of an embodiment operates to begin the portion-level differencing of the received images by parsing the specific image area and extracting information of the images. The information extracted about the images is used in differencing and updating operations. This parsing includes decoding the information of the static file system structure and internal format to get related information for use in performing SFS image differencing and updating. The information resulting from the parsing includes, but is not limited to, the locations and sizes of blocks of the image (e.g., super blocks, data blocks, etc.), the compression library used (if compressed), the type of encryption used to encrypt the image (e.g., encryption algorithm, encryption key, etc.) (if encrypted), to name a few.

Based on the target device block information, the image processing module 102 divides or splits the SFS image into a number or series of portions. **Figure 4** is an example SFS image 400 following splitting into portions 402 by the image processing module 102, under an embodiment. The portions 402 include portions or parts of a block (e.g., "Block 4") of the image but are not so limited. For example, following splitting of an embodiment Block 2 includes portions 402-2a, 402-2b, and 402-2c, and Block 4 includes portions 402-4a and 402-4b. Following the splitting operation a block may contain any number of portions as appropriate to the data content of the block and/or the specific structure of the SFS image. The portions include for example a super block/header portion, a control meta-data portion, a file portion, etc. The SFS image processing module 102 may perform decompression (when the image is compressed), decryption (when the image is encrypted) and/or any other processing needed in order for the image processing module 102 to extract data from the received image.

The image processing module 102 of an embodiment also outputs a file which includes mapping information like the SFS image/file name and one or more locations of the SFS image/file name, for example. This file is referred to herein as a hint file but is not so limited.

The dependency generation module 104, also referred to herein as a dependency analysis module or dependency generator, determines or generates a mapping or alignment between the original images and the new images. The alignment includes information as to portions of the new image that depend on

portions of the original image. The alignment of an embodiment also can include information as to the sequence by which the portions of the new image are to be updated during the in-place update of the original image hosted in the processor-based portable device. The dependency generation module 104 uses information of the specific structure of the SFS as domain knowledge (e.g., information from the hint file) in performing the alignment, and the alignment is determined relative to the block boundaries of the images, but is not so limited.

The difference engine 106 generally determines or computes the portion-level differences between data of each new portion of the new image and data of the original portion of the original image. More specifically, the difference engine 106 uses information from the dependency generation module 104 to read or gather one or more portions of the old image on which each portion of the new image depends. The difference engine 106 of an embodiment computes the differences between content of the new portion and content of the original portion upon which the new portion depends. The identified differences in data between the original and new portions are encoded into a file that is referred to herein as a delta file. In addition to the encoded differences in data between original and new portions, the delta file can include control information like dependency information of dependencies between new portions and original portions. Furthermore, the delta file can include verification information (e.g., checksum values, cyclic redundancy codes (CRCs), etc.) of the delta body, the original portion of an image to be updated, and the corresponding dependency information

While the difference engine 106 described above determines differences between original and new images at the portion-level, the difference engine of alternative embodiments may determine differences using a different granularity (e.g., multiple portions, etc.). Furthermore, while the difference engine 106 described above generates a delta file for each portion of a new image that is different from the portion of the original image on which it depends, alternative embodiments may generate more than one delta file for each portion or may include difference information of multiple portions in one delta file.

The packaging engine 108 receives the delta files generated by the difference engine 106 and assembles the delta files into a delta package or difference package. The packaging engine 108 of an embodiment assembles the delta files into the delta package according to an update order or sequence, but is not so limited. The

packaging engine 108 can also perform additional operations on the delta file and/or delta package, for example encrypting the delta package.

While the SFS differencing system 100 described above is described as including the image processing module 102, dependency generation module 104, difference engine 106, and packaging engine 108 as separate modules or components, the embodiment is not so limited. Alternative embodiments for example can include functionality of the modules 102-108 in one or more modules and/or distribute the functionality of modules 102-108 among any number of modules or system components.

The SFS differencing and updating of embodiments include updating by which SFS images hosted on a device like a portable electronic device are updated in-place on the device, as described above. The updating receives the delta file at a portable device via at least one coupling. Dependent ones of the original portions hosted on the portable device are assembled, and at least one of the dependent original sections is identified that corresponds to the delta file received. The updating reconstructs at least one new portion on the portable device that corresponds to the delta file identified. The reconstructed new portions of the new image are written to the ROM of the portable device.

Figure 5 is a block diagram of an SFS differencing and updating system 500, under an embodiment. The system 500 includes an SFS differencing system 100 and an SFS updating system 550. The SFS differencing system 100 includes an image processing module 102, a dependency generation module 104, a difference engine 106, and a packaging engine 108, as described above with reference to Figure 1, but is not so limited. In operation the SFS differencing system 100 receives at least one original SFS image 112 and at least one new SFS image 114 and performs portion-level differencing to generate a delta package 122 that includes one or more delta files as described herein.

The SFS differencing system 100 generates the delta file in a processor-based or computer system or running under a processor-based or computer system. The computer system on which or under which the SFS differencing system 100 runs includes any collection of computing components and devices operating together, as is known in the art. The computer system can also be a component or subsystem within a larger computer system or network.

The SFS updating system 550 is hosted on a processor-based device, and receives the delta file and performs updates to original images hosed on the portable device. The processor-based device or system on which or under which the SFS updating system 550 runs includes any collection of computing components and devices operating together, as is known in the art. The computer system can also be a component or subsystem within a larger computer system or network. The processor-based device or system can include mobile devices, for example, cellular telephones, personal computers, portable computing devices, portable telephones, portable communication devices, subscriber devices or units, and personal digital assistants (PDAs). The mobile devices, also referred to as "mobile communication devices," "portable communication devices" and "communication devices," can include all such devices and equivalents, and are not limited to communication devices that are wireless.

The SFS differencing system 100 and SFS updating system 550 communicate via a communication path 506. The communication path 506 includes any medium by which files are communicated or transferred between processor-based devices or systems. Therefore, the communication path 506 includes wireless couplings or connections, wired couplings or connections, and hybrid wireless/wired couplings or connections. The communication path 506 also includes couplings or connections to and/or through networks or network components including local area networks (LANs), metropolitan area networks (MANs), wide area networks (WANs), proprietary networks, interoffice or backend networks, and the Internet. The communication path 506 can include various network components (not shown) of a communication service provider or carrier, but is not so limited. Furthermore, the communication path 506 includes removable fixed mediums like floppy disks, hard disk drives, and CD-ROM disks, as well as telephone lines, buses, and electronic mail messages.

The delta package 122 is transferred or transmitted to the SFS updating system 550 via the communication path 506. The SFS updating system 550 includes an update module 552 that uses information of the delta package 122 to perform an in-place update 554 of the SFS image 112P hosted on the portable device. The update module 552 generally reconstructs the new image 114P at the portable device by applying contents of the delta package 122 to portions of the original SFS image 112P hosted on the portable device. The reconstructed portions of the new image

114P are written to the ROM of the portable device, for example, by writing the new image 114P over the hosted original image 112P. Upon completion of this SFS update process, the SFS image now hosted on the portable device is substantially identical to the new SFS image 114 received in the first SFS differencing system 100.

Figure 6 is a flow diagram for SFS updating 600, under an embodiment. The SFS updating 600 of an embodiment is used in resource-limited computing devices for example. The SFS updating receives 602 the delta file at a portable device via at least one coupling. Dependent original portions of the SFS image hosted on the portable device are assembled 604. The updating 600 identifies 606 at least one of the dependent original sections that correspond to the received delta file. The updating 600 reconstructs 608 at least one new portion on the portable device that corresponds to the delta file identified. The reconstructed new portions of the new image are subsequently written to the memory (e.g., ROM) of the portable device, for example, simultaneous with or subsequent to processing of all delta files received in a delta package.

Figure 7 is a flow diagram for in-place updating 700 of SFS images in devices, under an embodiment. The updating 700 can be performed for example by an update module 552 as described above with reference to Figure 5. The updating 700 uses the control information encoded in the received delta package to determine 702 that data of the delta package is not corrupt, thereby verifying the integrity of the delta package contents. The integrity of each SFS image portion to be updated can also be verified. If any data of the delta package is corrupt, the error is reported 704.

When it is determined 702 that the data of the delta package is not corrupt, the updating 700 sets 706 a pointer to the start of the delta package in order to begin the update. The portion update order and device block update order is implicitly encoded in the delta package of an embodiment. It is determined 708 that a delta file is present at the start of the update and the update proceeds. The delta file could be encrypted for security purposes and, in that case, the delta file is decrypted (using appropriate keys) before or simultaneous with the start of the update process. The updating 700 uses information of the delta file contents (e.g., control information) to decode 712 dependency formation information, read 712 related portions of the original image (if necessary), unzip and/or decrypt 712 dependent portions of the

original image (if necessary), and assemble 712 dependent content of original portions of the original image hosted on the portable device.

The updating 700 generates or reconstructs the new portion of the image being updated by applying 714 the contents of the delta file to the dependent content of the original portion of the original image. The new portion once reconstructed is written to an area of RAM in the host device, and is zipped and/or encrypted 714 as appropriate to the SFS image. The new portion is placed 714 into a specific location in the block of the original image so as to replace the original portion to which it corresponds. After each block of the new image is created or reconstructed, that particular block is written to the ROM; alternative embodiments may write the reconstructed blocks to the ROM at other points in the update process. The update process then proceeds with processing the delta file for the next block.

The updating 712 and 714 described above continues until the end of the delta package is reached and all delta files of the delta package have been processed. In so doing, a determination 716 is made as to whether the delta package includes further delta files for use in updating additional portions of the original image. When the delta package includes unprocessed delta files corresponding to additional portions of the original image to be updated, operation returns to read and apply these unprocessed delta files. When it is determined 716 that all delta files of the delta package have been processed and applied to the original image, the updating 700 writes 718 the new SFS image to the ROM of the portable device. The updating of an embodiment overwrites the original SFS image with the new SFS image, but alternative embodiments can write the new SFS image to one or more other areas of ROM or other device memory areas.

Referring to **Figures 1, 2, 3, 5, 6, and 7**, the operations of the processes are under control of at least one processor, but are not so limited. Those skilled in the relevant art can create source code, microcode, program logic arrays or otherwise implement the SFS differencing and/or SFS updating of an embodiment based on these flow diagrams and the detailed description provided herein. The algorithm or routine operating according to these flow diagrams is stored as program code in machine-readable or computer-readable memory areas or devices of a computer system (e.g., non-volatile memory) that forms part of the associated processors, in the associated memory areas, in removable media, such as disks, or hardwired or preprogrammed in chips, such as electronically erasable programmable ROM

("EEPROM") semiconductor chips, or in any combination of these components, but is not so limited.

The SFS differencing and updating of an embodiment includes a device for differencing static file system images. The device of an embodiment comprises a receiver that receives images of a static file system, the images including an original image and a new image. The device of an embodiment also comprises a pre-processor that divides the original image into numerous original sections and divides the new image into numerous new sections. The device of an embodiment comprises a dependency generator that identifies dependency alignments between the plurality of original sections and the plurality of new sections. The device of an embodiment comprises a difference engine that generates a delta file for at least one of the new sections, wherein the delta file includes differences between the at least one new sections and at least one of the original sections on which the at least one new sections depends.

The device of an embodiment further comprises a packaging engine that assembles the delta file for the at least one of the new sections according to an update sequence.

The SFS differencing and updating of an embodiment includes a method comprising splitting blocks of SFS images into portions based on block information and image structure information, wherein the SFS images include original images and new images. The method of an embodiment comprises performing portion-level differencing by generating a delta file for a new portion of the new image, wherein the delta file includes information of differences between the new portion and one of more corresponding original portions of the original image. The method of an embodiment comprises transferring the delta file to a client device. The method of an embodiment comprises updating a target SFS image of the client device using information of the delta file by reconstructing all portions of the new image in a device block in random access memory of the host device and writing the device block into read-only memory of the host device.

The SFS differencing and updating of an embodiment includes a method comprising receiving images of a static file system, the images including an original image and a new image. The method of an embodiment comprises dividing the original image into a plurality of original sections and divides the new image into a plurality of new sections. The method of an embodiment comprises identifying

dependency alignments between the plurality of original sections and the plurality of new sections. The method of an embodiment comprises generating a delta file for at least one of the new sections, wherein the delta file includes differences between the at least one new sections and at least one of the original sections on which the at least one new sections depends.

The SFS differencing and updating of an embodiment includes a system comprising a receiver that receives images of a static file system, the images including an original image and a new image. The system of an embodiment comprises a pre-processor coupled to the receiver that divides the original image into a plurality of original sections and divides the new image into a plurality of new sections. The system of an embodiment comprises a dependency generator coupled to the pre-processor that identifies dependency alignments between the plurality of original sections and the plurality of new sections. The system of an embodiment comprises a difference engine coupled to the dependency generator that generates a delta file for at least one of the plurality of new sections that is different from at least one of the plurality of original sections on which the at least one new section depends, the delta file including coded differences between a new section and one or more original sections. The system of an embodiment comprises a packaging engine coupled to the difference engine that assembles the delta files into a delta package.

The system of an embodiment comprises an update engine in a portable device, wherein the portable device receives the delta package via at least one coupling, wherein the update engine assembles dependent original sections of the plurality of original sections hosted on the portable device, identifies at least one delta file of the delta package that corresponds to at least one of the dependent original sections, and

reconstructs at least one new section that corresponds to the at least one delta file identified.

The update engine of an embodiment receives the delta package and verifies integrity of contents of at least one delta file of the delta package.

The update engine of an embodiment reconstructs the at least one new section in a first memory area of the portable device.

The first memory area of an embodiment is in random access memory (RAM).

The update engine of an embodiment continues identifying delta files of the delta package that correspond to at least one of the dependent original sections and reconstructing new sections that correspond to the delta files identified.

The update engine of an embodiment determines that all delta files of the delta package have been applied to the original sections hosted on the portable device and in response to the determination writes the reconstructed new sections to a second memory area of the portable device.

The update engine of an embodiment writes each block of the reconstructed new sections to a second memory area.

The second memory area of an embodiment is in read-only memory (ROM).

The SFS differencing and updating of an embodiment includes a method comprising receiving images of a static file system, the images including an original image and a new image, wherein the images include a plurality of blocks. The method of an embodiment comprises splitting the images by using information of the plurality of blocks to split the images into a plurality of portions. The method of an embodiment comprises determining differences between content of the images by determining differences between the plurality of portions of the original image and the new image. The method of an embodiment comprises generating a delta file that includes the differences for at least one portion.

The method of an embodiment comprises transferring the delta file to a portable wireless device that hosts the original image.

The method of an embodiment comprises receiving the delta file at a portable device via at least one coupling. The method of an embodiment comprises assembling dependent original portions of the plurality of original portions hosted on the portable device. The method of an embodiment comprises identifying at least one of the dependent original portions that corresponds to the delta file received. The method of an embodiment comprises reconstructing at least one new portion that corresponds to the at least one delta file identified.

The method of an embodiment comprises assembling a plurality of the delta files into a delta package.

The method of an embodiment comprises transferring the delta package to a portable wireless device that hosts the original image.

The method of an embodiment comprises receiving the delta package at the portable device via at least one coupling. The method of an embodiment comprises

assembling dependent original portions of the plurality of original portions hosted on the portable device. The method of an embodiment comprises identifying at least one delta file of the delta package that corresponds to at least one of the dependent original portions. The method of an embodiment comprises reconstructing at least one new portion that corresponds to the at least one delta file identified.

The SFS differencing and updating of an embodiment includes computer readable media including executable instructions which, when executed in a processing system, determine differences between images by receiving images of a static file system, the images including an original image and a new image, wherein the images include a plurality of blocks. The media further determines differences between images by splitting the images by using information of the plurality of blocks to split the images into a plurality of portions. The media further determines differences between images by determining differences between content of the images by determining differences between the plurality of portions of the original image and the new image. The media further determines differences between images by generating a delta file that includes the differences for at least one portion.

The media further determines differences between images by transferring the delta file to a portable wireless device that hosts the original image.

The media further determines differences between images by receiving the delta file at a portable device via at least one coupling. The media further determines differences between images by assembling dependent original portions of the plurality of original portions hosted on the portable device. The media further determines differences between images by identifying at least one of the dependent original portions that corresponds to the delta file received. The media further determines differences between images by reconstructing at least one new portion that corresponds to the at least one delta file identified.

The media further determines differences between images by assembling a plurality of the delta files into a delta package.

The media of an embodiment may transfer the delta package to a portable wireless device that hosts the original image.

The media of an embodiment receives the delta package at the portable device via at least one coupling. The media of an embodiment assembles dependent original portions of the plurality of original portions hosted on the portable device.

The media of an embodiment identifies at least one delta file of the delta package that corresponds to at least one of the dependent original portions. The media of an embodiment reconstructs at least one new portion that corresponds to the at least one delta file identified.

Aspects of the SFS differencing and updating described above may be implemented as functionality programmed into any of a variety of circuitry, including programmable logic devices (PLDs), such as field programmable gate arrays (FPGAs), programmable array logic (PAL) devices, electrically programmable logic and memory devices and standard cell-based devices, as well as application specific integrated circuits (ASICs). Some other possibilities for implementing aspects of the SFS differencing and updating include: microcontrollers with memory (such as electronically erasable programmable read only memory (EEPROM)), embedded microprocessors, firmware, software, etc. Furthermore, aspects of the SFS differencing and updating may be embodied in microprocessors having software-based circuit emulation, discrete logic (sequential and combinatorial), custom devices, fuzzy (neural) logic, quantum devices, and hybrids of any of the above device types. Of course the underlying device technologies may be provided in a variety of component types, e.g., metal-oxide semiconductor field-effect transistor (MOSFET) technologies like complementary metal-oxide semiconductor (CMOS), bipolar technologies like emitter-coupled logic (ECL), polymer technologies (e.g., silicon-conjugated polymer and metal-conjugated polymer-metal structures), mixed analog and digital, etc.

Unless the context clearly requires otherwise, throughout the description and the claims, the words "comprise," "comprising," and the like are to be construed in an inclusive sense as opposed to an exclusive or exhaustive sense; that is to say, in a sense of "including, but not limited to." Words using the singular or plural number also include the plural or singular number respectively. Additionally, the words "herein," "hereunder," "above," "below," and words of similar import, when used in this application, refer to this application as a whole and not to any particular portions of this application. When the word "or" is used in reference to a list of two or more items, that word covers all of the following interpretations of the word: any of the items in the list, all of the items in the list and any combination of the items in the list.

The above description of illustrated embodiments of the SFS differencing and updating is not intended to be exhaustive or to limit the invention to the precise form disclosed. While specific embodiments of, and examples for, the SFS differencing and updating are described herein for illustrative purposes, various equivalent modifications are possible within the scope of the SFS differencing and updating, as those skilled in the relevant art will recognize. The teachings of the SFS differencing and updating provided herein can be applied to other processing systems and communication systems, not only for the SFS differencing and updating systems described above.

The elements and acts of the various embodiments described above can be combined to provide further embodiments. These and other changes can be made to the SFS differencing and updating in light of the above detailed description. Furthermore, aspects of the SFS differencing and updating can be modified, if necessary, to employ the systems, functions and concepts of the various patents and applications described above to provide yet further embodiments of the SFS differencing and updating.

In general, in the following claims, the terms used should not be construed to limit the SFS differencing and updating to the specific embodiments disclosed in the specification and the claims, but should be construed to include all processing systems that operate under the claims to provide file differencing and updating. Accordingly, the SFS differencing and updating is not limited by the disclosure, but instead the scope of the SFS differencing and updating is to be determined entirely by the claims.

While certain aspects of the SFS differencing and updating are presented below in certain claim forms, the inventors contemplate the various aspects of the SFS differencing and updating in any number of claim forms. For example, while only one aspect of the SFS differencing and updating is recited as embodied in computer-readable medium, other aspects may likewise be embodied in computer-readable medium. Accordingly, the inventors reserve the right to add additional claims after filing the application to pursue such additional claim forms for other aspects of the SFS differencing and updating.

Claims

What is claimed is:

1. A device for differencing static file system images, comprising:
a receiver that receives images of a static file system, the images including an original image and a new image;
a pre-processor that divides the original image into a plurality of original sections and divides the new image into a plurality of new sections;
a dependency generator that identifies dependency alignments between the plurality of original sections and the plurality of new sections; and
a difference engine that generates a delta file for at least one of the new sections, wherein the delta file includes differences between the at least one new sections and at least one of the original sections on which the at least one new sections depends.

2. The device of claim 1, further comprising a packaging engine that assembles the delta file for the at least one of the new sections according to an update sequence.

3. A method comprising:
splitting blocks of SFS images into portions based on block information and image structure information, wherein the SFS images include original images and new images;
performing portion-level differencing by generating a delta file for a new portion of the new image, wherein the delta file includes information of differences between the new portion and one of more corresponding original portions of the original image;
transferring the delta file to a client device; and
updating a target SFS image of the client device using information of the delta file by reconstructing all portions of the new image in a device block in random access memory of the host device and writing the device block into read-only memory of the host device.

4. A method comprising:

receiving images of a static file system, the images including an original image and a new image;

dividing the original image into a plurality of original sections and divides the new image into a plurality of new sections;

identifying dependency alignments between the plurality of original sections and the plurality of new sections; and

generating a delta file for at least one of the new sections, wherein the delta file includes differences between the at least one new sections and at least one of the original sections on which the at least one new sections depends.

5. A system comprising:

a receiver that receives images of a static file system, the images including an original image and a new image;

a pre-processor coupled to the receiver that divides the original image into a plurality of original sections and divides the new image into a plurality of new sections;

a dependency generator coupled to the pre-processor that identifies dependency alignments between the plurality of original sections and the plurality of new sections;

a difference engine coupled to the dependency generator that generates a delta file for at least one of the plurality of new sections that is different from at least one of the plurality of original sections on which the at least one new section depends, the delta file including coded differences between a new section and one or more original sections; and

a packaging engine coupled to the difference engine that assembles the delta files into a delta package.

6. The system of claim 5, further comprising an update engine in a portable device, wherein the portable device receives the delta package via at least one coupling, wherein the update engine assembles dependent original sections of the plurality of original sections hosted on the portable device, identifies at least one delta file of the delta package that corresponds to at least one of the dependent original sections, and

reconstructs at least one new section that corresponds to the at least one delta file identified.

7. The system of claim 6, wherein the update engine receives the delta package and verifies integrity of contents of at least one delta file of the delta package.

8. The system of claim 6, wherein the update engine reconstructs the at least one new section in a first memory area of the portable device.

9. The system of claim 8, wherein the first memory area is in random access memory (RAM).

10. The system of claim 8, wherein the update engine continues identifying delta files of the delta package that correspond to at least one of the dependent original sections and reconstructing new sections that correspond to the delta files identified.

11. The system of claim 10, wherein the update engine determines that all delta files of the delta package have been applied to the original sections hosted on the portable device and in response to the determination writes the reconstructed new sections to a second memory area of the portable device.

12. The system of claim 11, wherein the update engine writes each block of the reconstructed new sections to a second memory area.

13. The system of claim 11, wherein the second memory area is in read-only memory (ROM).

14. A method comprising:

receiving images of a static file system, the images including an original image and a new image, wherein the images include a plurality of blocks;

splitting the images by using information of the plurality of blocks to split the images into a plurality of portions;

determining differences between content of the images by determining differences between the plurality of portions of the original image and the new image;

generating a delta file that includes the differences for at least one portion.

15. The method of claim 14, further comprising transferring the delta file to a portable wireless device that hosts the original image.

16. The method of claim 14, further comprising:

receiving the delta file at a portable device via at least one coupling;

assembling dependent original portions of the plurality of original portions hosted on the portable device;

identifying at least one of the dependent original portions that corresponds to the delta file received; and

reconstructing at least one new portion that corresponds to the at least one delta file identified.

17. The method of claim 14, further comprising assembling a plurality of the delta files into a delta package.

18. The method of claim 17, further comprising transferring the delta package to a portable wireless device that hosts the original image.

19. The method of claim 18, further comprising:

receiving the delta package at the portable device via at least one coupling;

assembling dependent original portions of the plurality of original portions hosted on the portable device;

identifying at least one delta file of the delta package that corresponds to at least one of the dependent original portions; and

reconstructing at least one new portion that corresponds to the at least one delta file identified.

20. Computer readable media including executable instructions which, when executed in a processing system, determine differences between images by:

receiving images of a static file system, the images including an original image and a new image, wherein the images include a plurality of blocks;

splitting the images by using information of the plurality of blocks to split the images into a plurality of portions;

determining differences between content of the images by determining differences between the plurality of portions of the original image and the new image;

generating a delta file that includes the differences for at least one portion.

21. The media of claim 20, wherein the media further determines differences between images by transferring the delta file to a portable wireless device that hosts the original image.

22. The media of claim 20, wherein the media further determines differences between images by:

receiving the delta file at a portable device via at least one coupling;

assembling dependent original portions of the plurality of original portions hosted on the portable device;

identifying at least one of the dependent original portions that corresponds to the delta file received; and

reconstructing at least one new portion that corresponds to the at least one delta file identified.

23. The media of claim 20, wherein the media further determines differences between images by assembling a plurality of the delta files into a delta package.

24. The media of claim 23, further comprising transferring the delta package to a portable wireless device that hosts the original image.

25. The media of claim 24, further comprising:

receiving the delta package at the portable device via at least one coupling;

assembling dependent original portions of the plurality of original portions hosted on the portable device;

identifying at least one delta file of the delta package that corresponds to at least one of the dependent original portions; and

reconstructing at least one new portion that corresponds to the at least one delta file identified.

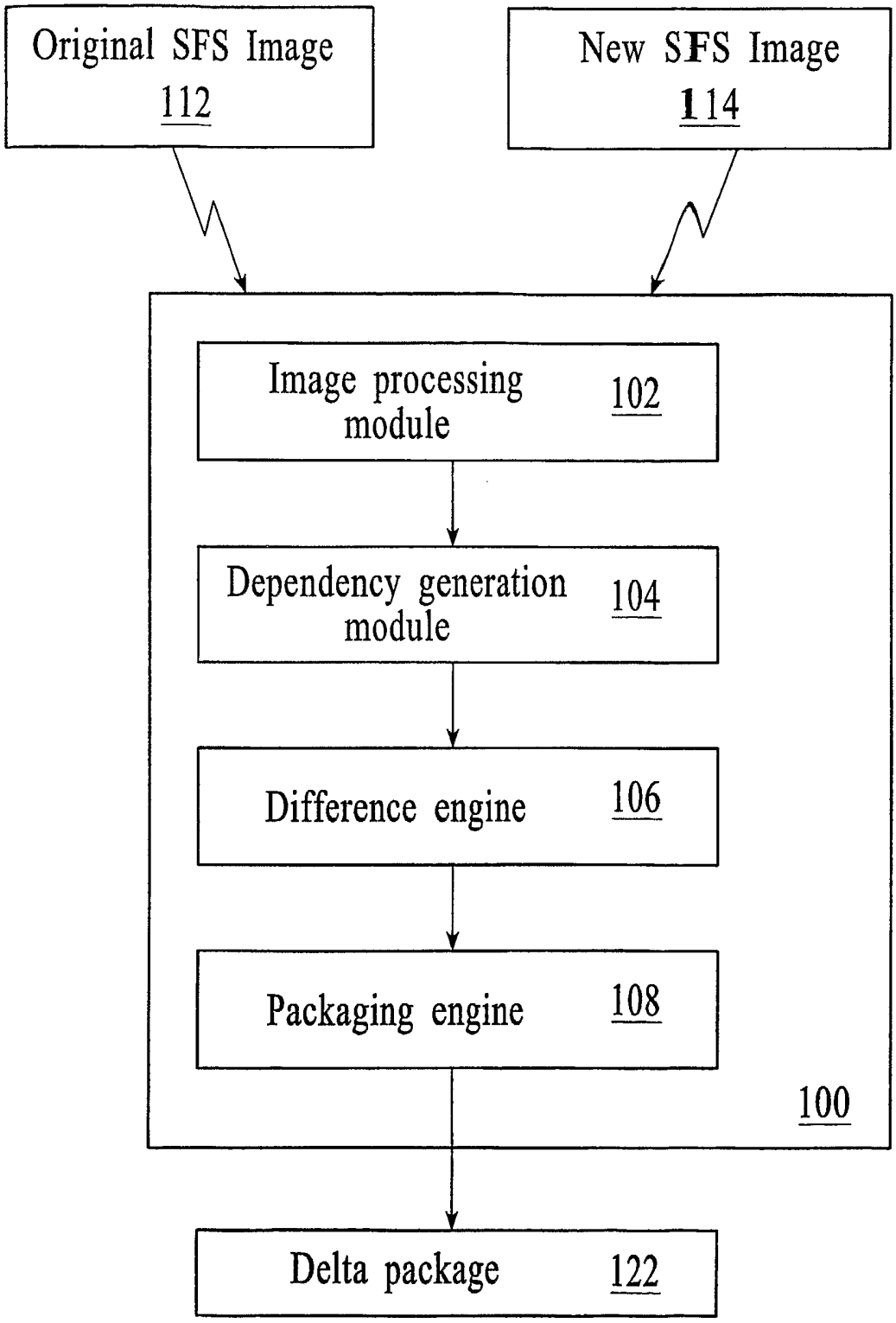

Original SFS Image
112
New SFS Image
114
Image processing module
102
Dependency generation module
104
Difference engine
106
Packaging engine
108
100
Delta package
122

FIG.1

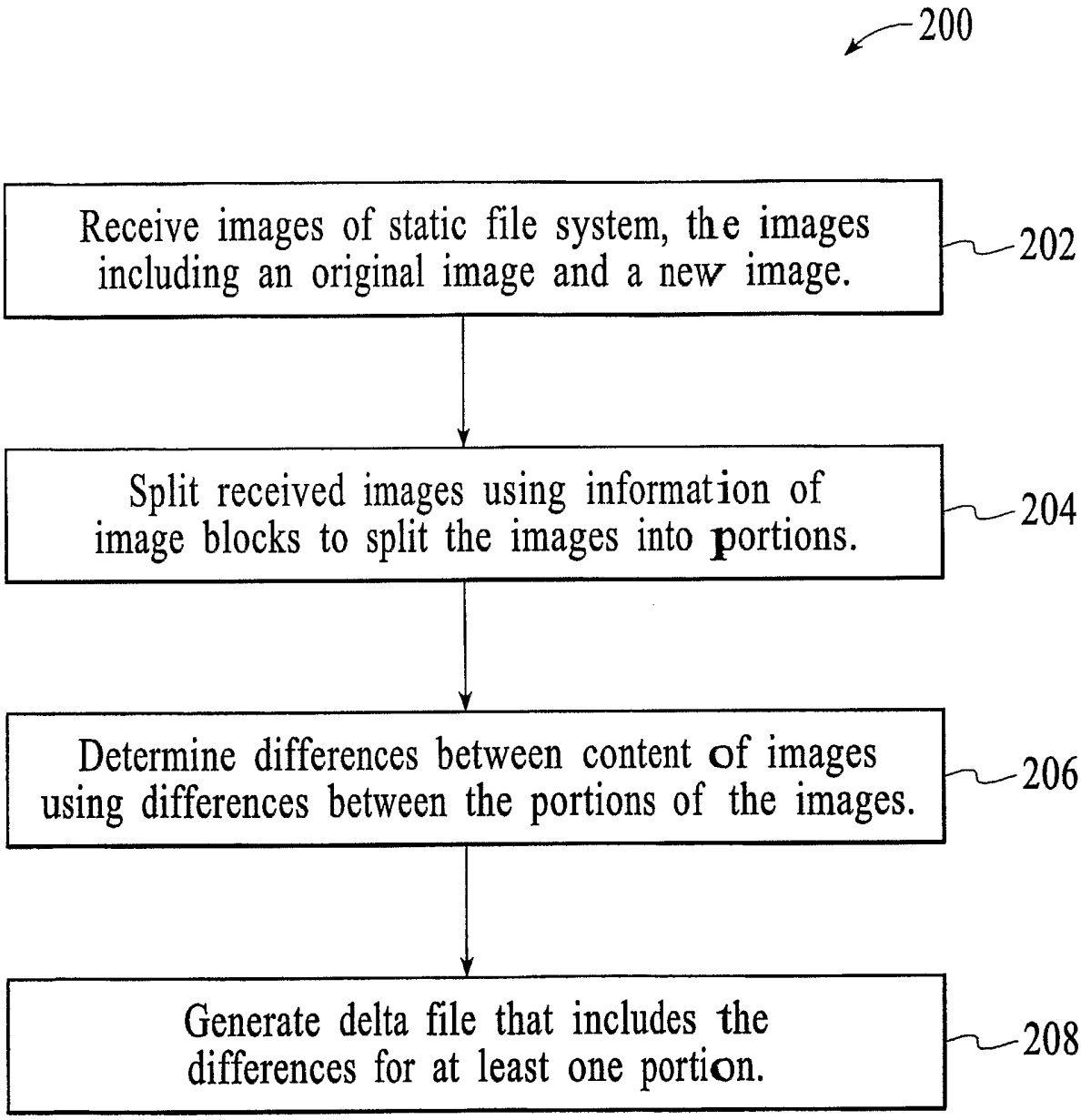

200
Receive images of static file system, the images including an original image and a new image.
202
Split received images using information of image blocks to split the images into portions.
204
Determine differences between content of images using differences between the portions of the images.
206
Generate delta file that includes the differences for at least one portion.
208

FIG.2

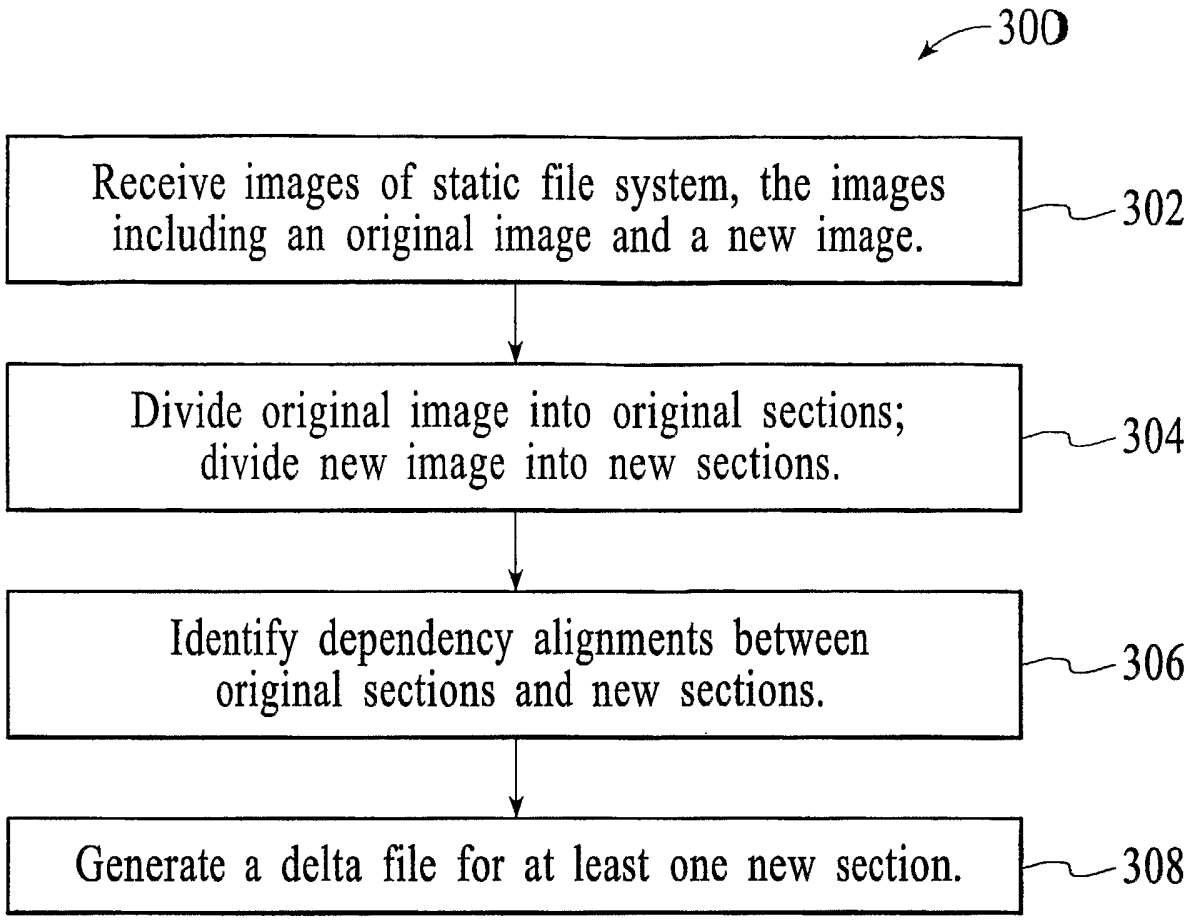

300
Receive images of static file system, the images including an original image and a new image.
302
Divide original image into original sections; divide new image into new sections.
304
Identify dependency alignments between original sections and new sections.
306
Generate a delta file for at least one new section.
308

FIG.3

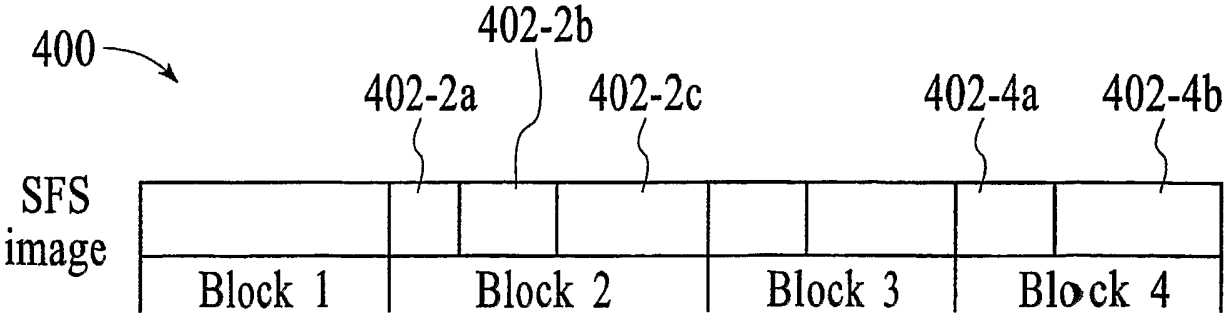

400
402-2b
402-2a
402-2c
402-4a
402-4b
SFS image
Block 1
Block 2
Block 3
Block 4

FIG.4

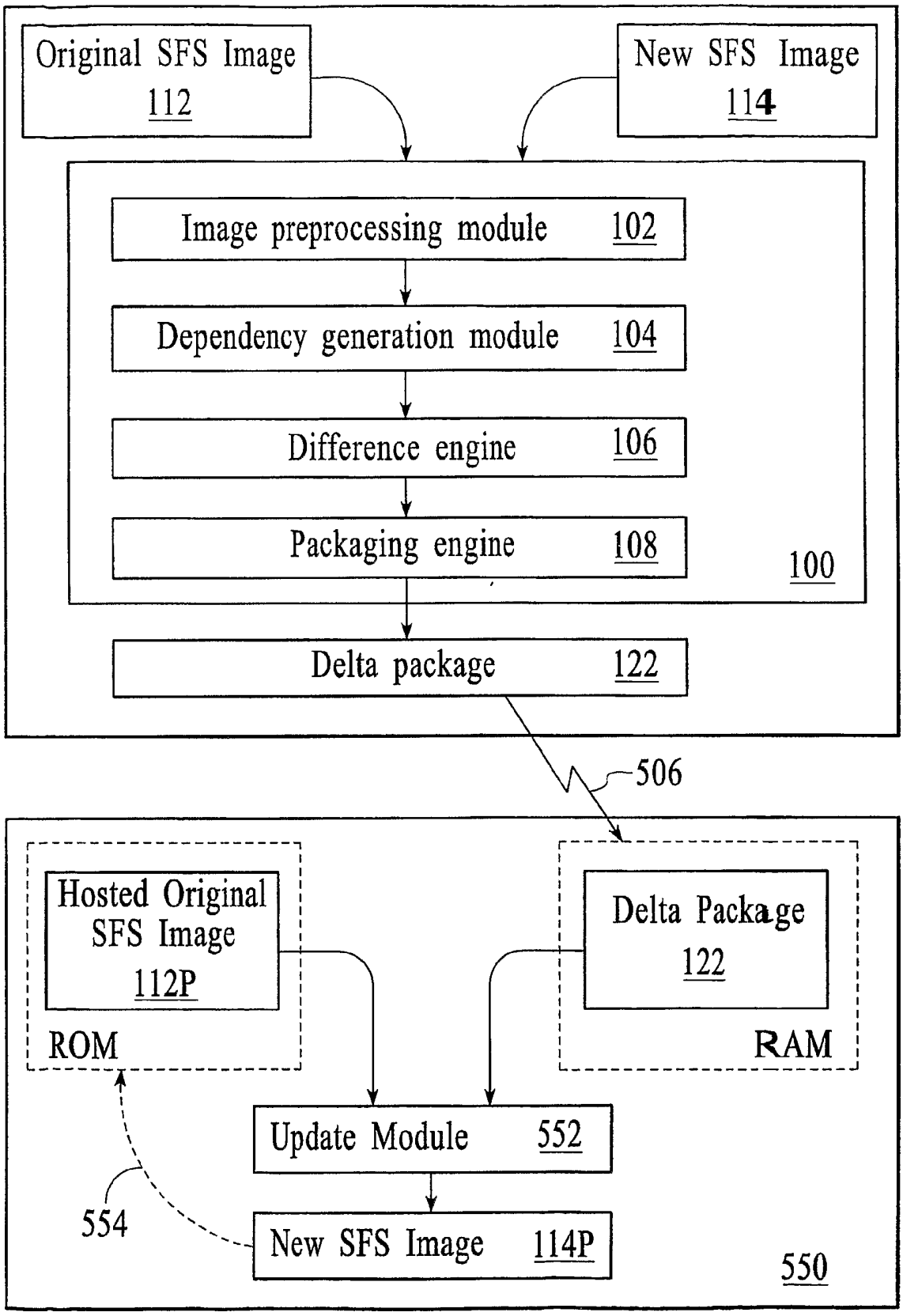

Original SFS Image
112
New SFS Image
114
Image preprocessing module 102
Dependency generation module 104
Difference engine 106
Packaging engine 108
100
Delta package 122
506
Hosted Original SFS Image
112P
ROM
Delta Package
122
RAM
Update Module 552
554
New SFS Image 114P
550
500

FIG 5

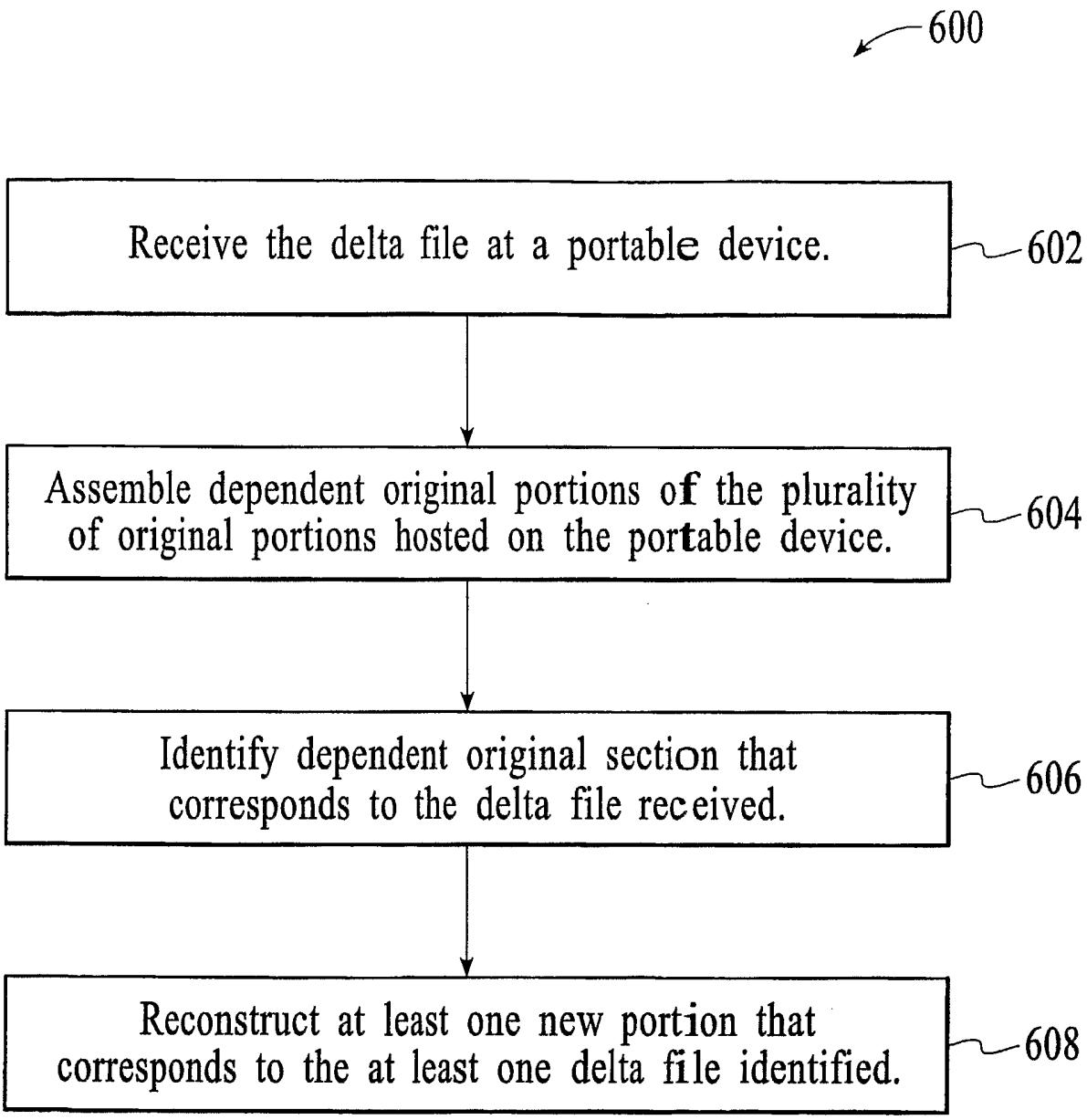

600
Receive the delta file at a portable device.
602
Assemble dependent original portions of the plurality of original portions hosted on the portable device.
604
Identify dependent original section that corresponds to the delta file received.
606
Reconstruct at least one new portion that corresponds to the at least one delta file identified.
608

FIG.6

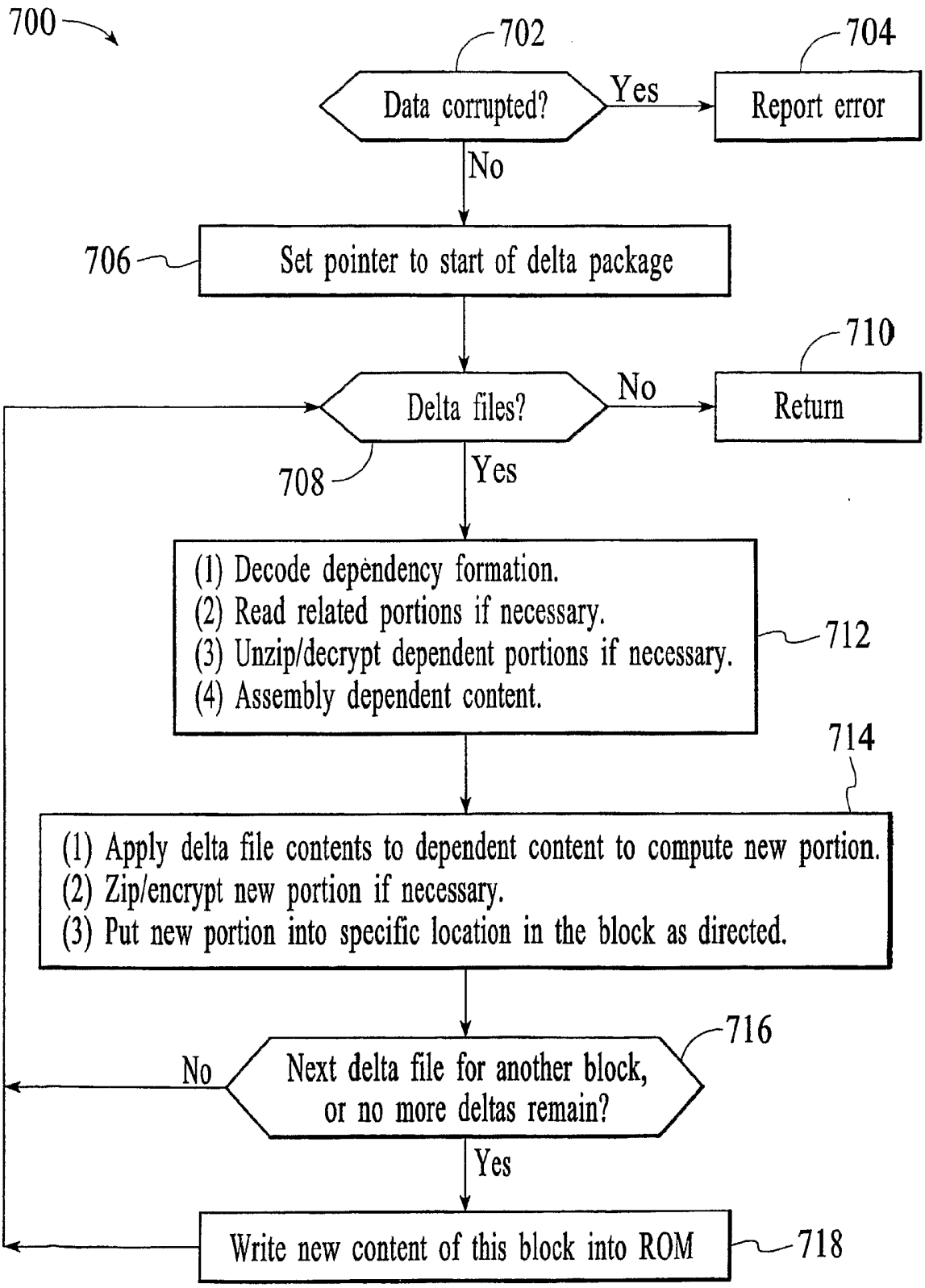

700
702
Data corrupted?
Yes
704
Report error
No
706
Set pointer to start of delta package
710
Delta files?
No
Return
708
Yes
(1) Decode dependency formation.
(2) Read related portions if necessary.
(3) Unzip/decrypt dependent portions if necessary.
(4) Assembly dependent content.
712
714
(1) Apply delta file contents to dependent content to compute new portion.
(2) Zip/encrypt new portion if necessary.
(3) Put new portion into specific location in the block as directed.
716
No
Next delta file for another block, or no more deltas remain?
Yes
Write new content of this block into ROM
718

FIG 7