



- (51) **International Patent Classification:**
G06F 13/14 (2006.01) G06F 9/00 (2006.01)
G06F 3/00 (2006.01)
- (21) **International Application Number:**
PCT/US2009/033269
- (22) **International Filing Date:**
5 February 2009 (05.02.2009)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
61/026,438 5 February 2008 (05.02.2008) US
- (71) **Applicant (for all designated States except US):** VI-ASAT, INC. [US/US]; 6155 El Camino Real, Carlsbad, CA 92009 (US).
- (72) **Inventors; and**
- (75) **Inventors/Applicants (for US only):** OWENS, John, R. [US/US]; 1928 High Ridge Avenue, Carlsbad, CA 92008 (US). ANDOLINA, John, C. [US/US]; 1981 Pinewood Road, Vista, CA 92081 (US). SHANKEN, Stuart [US/US]; 5028 Almondwood Way, San Diego, CA 92130 (US). QUINTANA, Richard, L. [US/US]; 6482 Kite Place, Carlsbad, CA 92011 (US).
- (74) **Agents:** FRANKLIN, Thomas, D. et al.; Townsend And Townsend And Crew LLP, Two Embarcadero Center, 8th Floor, San Francisco, CA 94111-3834 (US).
- (81) **Designated States (unless otherwise indicated, for every kind of national protection available):** AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, 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, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: TRUSTED FIELD-PROGRAMMABLE LOGIC CIRCUITRY

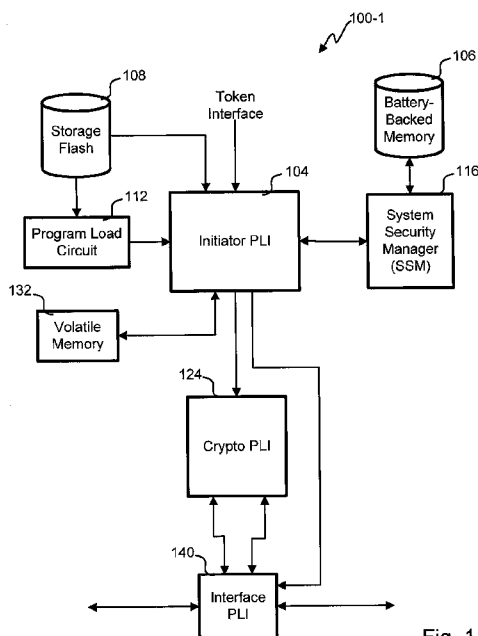


Fig. 1A

(57) **Abstract:** In one embodiment, a method for trusted booting of a cryptographic processor system is disclosed. Default image(s) is loaded into a field-programmable logic chip (FPLC). The default image(s) cannot perform cryptographic processing, but can perform a first algorithm that is unclassified. A processor, internal or external to the FPLC, can be used with the default image. A multi-layer or multi-part key has portions stored in two different places. A protected image is decrypted with the multi-layer key using the first algorithm and loaded into the FPLC. Cryptographic processing is performed using a second algorithm classified or produced by the government. In another embodiment, a method for securing a field-programmable logic chip (FPLC) is disclosed. Information is cryptographically processed within the FPLC. An error condition is detected outside of the FPLC and the error condition is communicated to the FPLC to disrupt an image(s) within the FPLC. Optionally, at least a portion of a key can be erased such that cryptographic processing is curtailed or eliminated. In yet another embodiment, a method for operating a field-programmable logic chip (FPLC) is disclosed. Operation of the FPLC includes a configuration state and a cryptographic processing state. Switching between states is controlled by a state machine. Each state has one or more images. Transferring between states causes some or all images from the other state being overwritten.

WO 2009/100249 A2

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

TRUSTED FIELD-PROGRAMMABLE LOGIC CIRCUITRY

[0001] This application claims the benefit of and is a non-provisional of co-pending US Provisional Application Serial No. 61/026,438 filed on February 5, 2008, which is hereby expressly incorporated by reference in its entirety for all purposes.

5

BACKGROUND

[0002] This disclosure relates in general to field-programmable logic chip or circuit (FPLC) and, but not by way of limitation, to FPLC used in traffic processing such as cryptographic processing.

10 [0003] Programmable circuitry is common in logic design, but achieving the level of security and high-assurance desired by the governments, individuals and certain businesses has been difficult. Programmability is seen as a risk to achieving security and high-assurance. This is especially true when programmability is utilized in the field. One could imagine the programmability feature being used to cause the FPLC to operate in a mode that would not have the required security.

15 [0004] Cryptographic circuitry can fail or be compromised. Where such a circuit enters an error mode, there is no recovery. Failure can result in insecure functioning that is not desirable. The cryptographic circuit can erase keys to prevent further activity. Even without keys, the cryptographic circuit can perform in undesirable modes when malfunctioning. With programmability, the risk of these malfunctions is greater.

20 [0005] FPLCs have many advantages over fixed circuitry, but cost is generally not one of the advantages. Various images and soft cores are loaded into FPLCs. The size of the FPLC is chosen to accommodate all the images needed for a given design. When smaller or fewer FPLCs are possible, the costs of producing a unit decreases.

SUMMARY

25 [0006] In one embodiment, a method for trusted booting of a cryptographic processor system is disclosed. Default image(s) is loaded into a field-programmable logic circuit or chip (FPLC). The default image(s) cannot perform cryptographic processing, but can perform a first algorithm that is unclassified. A processor, internal or external to the FPLC, can be used with the default image. A multi-layer or multi-part key has portions stored in
30 two different places. A protected image is decrypted with a key using the first algorithm

and loaded into the FPLC. Cryptographic processing is performed using a second algorithm classified or developed by the government.

[0007] In one embodiment, the present disclosure provides a method for booting with multi-layered security that cryptographically processing information. The method includes
5 loading a default image into a FPLC. The default image prevents the FPLC from passing information from a first port to a second port. The default image includes a first algorithm that is unclassified by the government and allows operational software to function. The method also includes loading a multi-layered key. The first layer of the multilayer key is stored separate from a second layer of the multilayer key and the first and second layers are
10 used to formulate the multi-layer key. The method also includes decrypting a protected image using the multi-layered key and the first algorithm to produce a decrypted image from the encrypted protected image. The method also includes loading the decrypted image into the FPLC. The also further includes performing cryptographic processing using the decrypted image and a classified second algorithm. The cryptographic processing operates
15 on information passing from the first port to the second port.

[0008] In one embodiment, the present disclosure provides a cryptographic processing system for booting a FPLC with multi-layered keys. The system includes a first port for receiving information for cryptographic processing and a second port for transmitting information after cryptographic processing. The system also includes a default image
20 loaded in the FPLC. The default image prevents the FPLC from passing information from a first port to a second port. The default image includes a first algorithm that is unclassified by the government. The default image allows operational software to function. The system also includes a multi-layered key. A first layer of the multilayer key is stored separate from a second layer of the multilayer key and the first and second layers of the multilayer key are
25 used to formulate the multi-layer key. The system also includes a protected image that is decrypted using the multi-layered key and the first algorithm to produce a decrypted image. The decrypted image is loaded into the FPLC. Cryptographic processing is performed using the decrypted image and a second algorithm, which is classified by the government. The cryptographic processing operates on information passing from the first port to the second
30 port.

[0009] In one embodiment, the present disclosure provides a cryptographic processing system for booting a FPLC with multi-layered keys. The cryptographic processing system

includes a first means for loading a default image into the FPLC. The default image prevents the FPLC from passing information from a first port to a second port. The default image includes a first algorithm that is unclassified by the government and allows operational software to function. The cryptographic processing system also includes a
5 second means for loading a multi-layered key. A first layer of the multilayer key is stored separate from a second layer of the multilayer key. The first and second layers are used to formulate the multi-layer key. The cryptographic processing system also includes a means for decrypting a protected image using the multi-layered key and the first algorithm to produce a decrypted image. The cryptographic processing system also includes a third
10 means for loading the decrypted image into the FPLC. The cryptographic processing system also includes a means for cryptographically processing that uses the decrypted image and a second algorithm, which is classified by the government. The cryptographic processing operates on information passing from the first port to the second port.

[0010] In another embodiment, a method for securing a FPLC is disclosed. Information is
15 cryptographically processed within the FPLC. An error condition is detected outside of the FPLC and the error condition is communicated to the FPLC to disrupt an image(s) within the FPLC. Optionally, at least a portion of a key can be erased such that cryptographic processing is curtailed or eliminated.

[0011] In one embodiment, the present disclosure provides a method for providing system
20 security for a FPLC. The method includes cryptographically processing information within the FPLC and detecting an error condition. The detecting operation is performed external to the FPLC. The method also includes conveying into the FPLC the error condition. The method also includes disrupting an image in the FPLC for one or more soft cores loaded when the error condition is detected where the image cannot perform cryptographic
25 processing after disruption even if keys are present.

[0012] In one embodiment, the present disclosure provides a cryptographic processing system for providing system security for a FPLC. The cryptographic processing system includes a cryptographic soft core at least partially within the FPLC that cryptographically processes information. The cryptographic processing system also includes a system
30 security manager that detects an error condition. The detection is performed external to the FPLC, the cryptographic soft core in the FPLC is disrupted when the error condition is

detected and the cryptographic soft core cannot perform cryptographic processing after disruption even if keys are present.

[0013] In one embodiment, the present disclosure provides a cryptographic processing system for providing system security for a FPLC. The cryptographic processing system includes a means for cryptographically processing information within the FPLC. The system also includes a means for detecting an error condition. The detection is performed external to the FPLC. The system also includes a means for disrupting an image in the FPLC for one or more soft cores loaded when the error condition is detected. The image cannot perform cryptographic processing after disruption even if keys are present.

10 [0014] In one embodiment, a method for operating a FPLC is disclosed. Operation of the FPLC includes a configuration state and a cryptographic processing state. Switching between states is controlled by a state machine. Each state has one or more images. Transferring between states causes some or all images from the other state being overwritten.

15 [0015] In one embodiment, the present disclosure provides a method for operating a FPLC for cryptographic processing. The method includes operating in a configuration state using a configuration processing soft core. The configuration state is used to configure a cryptographic processing state with configuration information. The configuration processing soft core consumes a first portion of the FPLC. The method also includes switching from the configuration state to the cryptographic processing state. The switching operation is controlled by a state machine. The method also includes operating in the cryptographic processing state using a cryptographic processing soft core. The cryptographic processing state performs cryptographic processing and the cryptographic processing soft core consumes a second portion of the FPLC. The first portion and the second portion overlap within the FPLC such that only one of the configuration processing soft core or the cryptographic processing soft core can be loaded into the FPLC at a time.

20 [0016] In one embodiment, the present disclosure provides a cryptographic traffic processor for processing information using a FPLC. The processor includes a configuration processing soft core, a cryptographic processing soft core, and a state machine having a configuration state and a traffic processing state. The configuration state uses the configuration processing soft core and is used to configure the traffic processing state with configuration information. The configuration processing soft core consumes a first portion

of the FPLC. The configuration state switches to the traffic processing state under the control of the state machine. The traffic processing state uses the traffic processing soft core. The traffic processing state performs traffic processing. The traffic processing soft core consumes a second portion of the FPLC. The first portion and the second portion
5 overlap at different times within the FPLC such that only one of the configuration processing soft core or the traffic processing soft core can be loaded into the FPLC at a time.

[0017] In one embodiment, the present disclosure provides a cryptographic traffic processor for processing information using a FPLC. The processor includes a means for
10 operating in a configuration state using a configuration processing soft core. The configuration state is used to configure a traffic processing state with configuration information and the configuration processing soft core consumes a first portion of the FPLC. The processor also includes a means for switching from the configuration state to the traffic processing state, in which the switching is controlled by a state machine. The
15 processor also includes a means for operating in the traffic processing state using a traffic processing soft core. The traffic processing state performs traffic processing. The traffic processing soft core consumes a second portion of the FPLC. The first portion and the second portion overlap within the FPLC such that only one of the configuration processing soft core or the traffic processing soft core can be loaded into the FPLC at a time.

[0018] Further areas of applicability of the present disclosure will become apparent from the detailed description provided hereinafter. It should be understood that the detailed description and specific examples, while indicating various embodiments, are intended for purposes of illustration only and are not intended to necessarily limit the scope of the disclosure.

BRIEF DESCRIPTION OF THE DRAWINGS

[0019] The present disclosure is described in conjunction with the appended figures:

FIGS. 1A, 1B and 1C depict block diagrams of embodiments of a cryptographic processor system;

FIG. 2A depicts a block diagram of an embodiment of a system security manager (SSM);
30

FIG. 2B depicts a block diagram of an embodiment of a fail-safe SSM;

FIG. 3 illustrates a flowchart of an embodiment of a process for booting a cryptographic processor system;

FIGs. 4A, 4B, 4C and 4D depict block diagrams of embodiments of a traffic processing system;

5 FIGs. 5A, 5B, 5C, 5D, 5E, 5F, 5G, and 5H depict diagrams of embodiments of a layout of a field-programmable logic chip (FPLC) implementing a traffic processor;

FIGs. 6A and 6B depict diagrams of embodiments of a state machine used to control the traffic processor; and

10 FIG. 7 illustrates a flowchart of an embodiment of a process for cryptographically processing information in a two state configuration.

[0020] In the appended figures, similar components and/or features may have the same reference label. Further, various components of the same type may be distinguished by following the reference label by a dash and a second label that distinguishes among the similar components. If only the first reference label is used in the specification, the
15 description is applicable to any one of the similar components having the same first reference label irrespective of the second reference label.

DETAILED DESCRIPTION

[0021] The ensuing description provides preferred exemplary embodiment(s) only, and is not intended to limit the scope, applicability or configuration of the disclosure. Rather, the
20 ensuing description of the preferred exemplary embodiment(s) will provide those skilled in the art with an enabling description for implementing a preferred exemplary embodiment. It being understood that various changes may be made in the function and arrangement of elements without departing from the spirit and scope as set forth in the appended claims.

I. Trusted Boot

25 [0022] Referring first to **FIG. 1A**, a block diagram of an embodiment of a cryptographic processor system 100-1 is shown. There are a number of field-programmable logic chips or circuits (FPLCs) in this embodiment, which may be field-programmable gate arrays (FPGA), programmable logic devices (PLDs), complex PLDs (CPLDs), or any other circuit
30 that can be programmed with some logic or a soft core after manufacture, for example, in the field. Programmability in the field includes programmability while manufacturing a system including the FPLC or programmability while the system is deployed with an end

user. A FPLC is a circuit chip or die in it's own package or chips or dice in a multi-chip module.

[0023] A programmable logic image (PLI) is a soft core of functionality that can be programmed into a FPLC or otherwise implemented. The PLI could include general purpose processor, a state machine, an application specific processor, a cryptofunction, and/or configuration information and parameters. A number of PLIs may be in a single FPLC or a single PLI may be spread out among a number of FPLCs. The blocks shown in the figures can be combined or split in various embodiments.

[0024] A number of PLIs are used to process traffic or more specifically, cryptographically process traffic. Plain text information is received by the interface PLI 140 for encryption processing, and cipher text information is output by the interface PLI 140. Conversely, cipher text information is received by the interface PLI 140 for decryption processing, and plain text information is output by the interface PLI 140. The interface PLI 140 can pass information without cryptographic processing in some cases. For example, the three PLIs 104, 124, 140 in this embodiment could be implemented in one or two FPLCs with the first holding the initiator PLI, the second holding the interface PLI and the crypto PLI divided between the two FPLCs.

[0025] A crypto PLI 124 performs cryptographic processing in a traffic processing state. These are just representative blocks for performing cryptographic processing and could be combined or separated in various embodiments. If loaded into the same FPLC, PLIs can be isolated from each other with a buffer of unused cells and controlled interfaces between the isolated areas. Signals from one PLI can be kept from routing outside the isolated area except where a deliberate port is configured to pass information between PLIs. In this way, isolation can be achieved in the same device unless interaction through a port is desired.

[0026] Soft cores for the various PLIs in their various versions are held in a storage flash 108. These soft cores are sometimes referred to as images. Additionally, key fragments or layers can be held in the storage flash 108. The storage flash 108 can additionally hold software to boot and run any processor of the cryptographic processor system 100. Any type of flash memory or non-volatile can be used for the storage flash 108.

[0027] This embodiment also includes volatile memory 132. A processing core within the initiator PLI 104 loads software from the storage flash 108 and uses the volatile memory 132 for program operation and variable storage. RAM, SRAM or any type of volatile

memory could be used for the volatile memory 132. Other embodiments could use non-volatile memory, for example, MRAM for the volatile memory 132.

[0028] A battery or other power source (not shown) is used to allow a battery-backed memory 106 to retain its contents even when main power is interrupted or lost. SRAM or
5 DRAM can be used for the battery-backed memory. The battery-backed memory 106 may also hold key layers. In one embodiment, some key layers are stored in the storage flash 108 while others are stored on the battery-backed memory 106. Further, key layers can be held on a token that is removably coupled to the cryptographic processor system 100 through a token interface. Any type of non-volatile memory can be used for the storage
10 flash 108, the battery-backed memory 106 or token.

[0029] The program load circuit 112 loads soft cores for the PLIs into one or more FPLCs. The program load circuit 112 could be implemented with a CPLD, for example. The various images or soft cores in the storage flash 108 are loaded into the programmable logic in a particular sequence that can be controlled by the program load circuit 112 and/or
15 the loaded PLIs. Certain PLIs can be loaded into the programmable logic and later removed such that other PLIs can recover some of the programmable logic when the removed PLIs are not needed.

[0030] An initiator PLI 104 may assist in this process and perform other configuration once the soft core of the initiator PLI 104 is loaded and functioning. Certain configuration
20 actions such as loading keys, built-in test and other housekeeping functions can be performed by the initiator PLI 104. In some embodiments, the soft core of the initiator PLI 104 may be removed if not needed and reloaded when it is needed. Memory can be used to pass information from the initiator PLI 104 to other PLIs to use when the initiator PLI is not loaded.

25 [0031] As further described below, the system security manager (SSM) 116 monitors for errors and alarms before taking remedial action on the PLIs, FPLCs and/or key layers. Redundancy can be used by the SSM 116 to operate in a failsafe, trusted and/or high-assurance manner.

[0032] Referring next to **FIG. 1B**, shows a block diagram of another embodiment of the
30 cryptographic processor system 100-2 that uses a fail-safe SSM 118 instead of a SSM 116. The fail-safe SSM 118 includes redundancy and/or other high-assurance circuits as further described below. This embodiment also includes a processor chip 128 instead or in addition

to a processing soft core within the initiator PLI 104. The processor chip 128 is a hardware processor separate from the FPLC holding the initiator PLI 104. Volatile memory 132 is used by the processor chip 128 for program operation and variable storage.

[0033] With reference to **FIG. 1C**, a block diagram of yet another embodiment of a cryptographic processor system 100-3 is shown. This embodiment includes a security manager PLI 136 that is embedded in a FPLC. Some or all of the FPLCs used in the cryptographic processor system 100 could have its own security manager PLI 136. The security manager PLI could be used in addition to a SSM 116 or a fail-safe SSM 118 in various embodiments. The security manager PLI 136 is further described below.

10 [0034] Referring next to **FIG. 3**, a flow chart of an embodiment of a process 300 for booting a cryptographic processor system 100 is shown. The depicted portion of the process begins in block 304 where the storage flash 108 is loaded with one or more default images. Some FPLCs have a decryption algorithm built into the chip to be programmed that uses a fixed key to allow decryption of images loaded into that chip. For example,
15 Xilinx™ and Altera™ could provide on-chip advanced encryption standard (AES) decryption of images using a predetermined key that is fixed and battery-backed on the chip. Block 304 would include encrypting the default image(s) with the appropriate key prior to storage in the storage flash 108.

[0035] The default images provide just enough logic to get the cryptographic processor system 100 running in a configuration state, but not enough to allow data throughput in this
20 embodiment. For example, the initiator PLI 104 can be a default image in some embodiments. The default images support built-in-test to allow checking that the logic circuitry at least has some functionality and that the circuit card was assembled properly. Other embodiments could do certain unclassified traffic processing with the default images.
25 A loaded default image allows running and updating of the operational software. Additionally, the default images include another decryption algorithm that is unclassified. This unclassified decryption algorithm is a soft core that can be loaded into FPLC to allow decrypt and load of additional logic in the same chip or another chip.

[0036] In the United States, the government classifies certain cryptographic algorithms
30 while others are unclassified. Classified algorithms are not available to the general public and are controlled by government regulation. Each country can have their own classified and unclassified cryptographic algorithms which may vary from other countries. As the

control of classified and unclassified algorithms differs in a particular country, embodiments use a mixture in certain circumstances.

[0037] In block 308, protected images are loaded into the storage flash 108. These may be optionally encrypted to allow decoding with the on-chip decryption algorithm. The protected images are encrypted to allow decryption with the unclassified decryption algorithm and an appropriate key. The protected images allow full cryptographic processing when properly loaded and enabled with the appropriate key(s). In one embodiment, a particular portion of a FPLC or a whole programmable FPLC could start out with a default image that is later replaced wholly or in part with another protected or unprotected image. Other embodiments could keep some or all of the default image functioning alongside protected images.

[0038] Operation of the cryptographic processor system 100 is begun in block 316. Prior blocks 304, 308 and 312 could be performed at the factory in one embodiment. Block 312 is likely to be done at least partially in the field as layers of the multi-layered key may change over time or be erased. Booting may be begun by application of power to the cryptographic processor system 100 or by a reset operation or other remedial action.

[0039] One or more default images are retrieved from the image flash storage 108 and loaded into programmable logic by the program load circuit 112 in block 320. For example, the initiator PLI 104 could be loaded as a default image. The default image(s) may be encrypted and a previously-loaded default image, such as the initiator PLI 104, could automatically decrypt the default image before loading into the programmable logic. Other embodiments may use the cryptographic function built into some FPLCs as explained above. Yet other embodiments could use the built-in cryptographic function and a soft core cryptographic function to utilize double decryption. Embodiments may have various blocks implemented in one or more PLIs to separate different functions within a given FPLC.

[0040] Once the initiator PLI 104 has the default image loaded in unencrypted form, a general purpose processor is available as a soft core that was part of the default image. Other embodiments could use a hard core for the processor used by the initiator PLI and external to any FPLC. Memory is available for processing and retrieval of operational software that is booted in block 324. Once the default software is loaded there is enough intelligence to accept new software and/or keys, but no cryptographic processing of traffic can be done in this embodiment of the default software. In this embodiment, the initiator

PLI 104 has the unclassified decryption algorithm as a soft core or available in software. The PLIs operate with default images in block 324.

[0041] The initiator PLI retrieves the remainder of the default images in block 326.

Those default images are loaded using optional decryption before built-in-test is performed on each of the other PLIs. During the loading process, verification of checksums, CRCs or hashes can be performed to confirm the default images were loaded correctly. The initiator PLI 104 can reload the default images where there is an error in the checksum, CRC or hash.

[0042] A determination in block 328 analyzes whether there are multi-layer keys present to continue booting the cryptographic processor system 100. The multi-layer key is reconstituted by retrieving various layers or portions from different locations. For example, the storage flash 108, the token and/or the battery-backed memory 106 may all have a portion of the multi-level key. Where one or more layers of the multi-layer key is missing processing loops back to block 312 to wait for loading of a missing key layer(s). If the multi-layer key is present, processing goes from block 328 to 332, but goes from block 328 to block 312 if the multi-layer key is missing.

[0043] The multi-layered key(s) are loaded into the cryptographic processor system 100 in block 312. The multi-layered key has multiple portions that are all needed by the unclassified decryption algorithm to decrypt the protected images. Layers of the multi-layered key(s) can be erased if error conditions are found. Without all of the layers, the multi-layered key will not be usable. A battery-backed memory 106 and/or flash may be used to store the various layers of the multi-layered key. This embodiment stores various layers or values of the keys in the storage flash 108, the battery-backed memory 106 and/or a token coupled to a token interface. The layers may additionally be encrypted, for example, the layer retrieved from the storage flash 108 is encrypted in one embodiment. Some locations may store multiple layers or portions of the multi-layer key, for example, the battery-backed memory 106 could have two portions that can be individually erased or scrambled to destroy the multi-layer key. The condition for destroying each layer of the key could be different to protect against different threats. Once the multi-layered key is loaded, processing goes from block 312 back to block 328.

[0044] In block 332, the initiator PLI 104 loads a soft core version of the unclassified decryption algorithm. The unclassified decryption algorithm could be AES, triple-DES or

any other appropriate algorithm . The multi-layer key is constituted by retrieving layers from two or more locations. The protected soft cores are retrieved from the storage flash 108 and decrypted by the initiator PLI 104 in block 336. Some embodiments may also use the AES encryption built into the FPLC to utilize double decryption. That second order of protection of the AES encryption would involve decryption again as the protected image is loaded into a particular FPLC. Additionally, the initiator PLI 104 may calculate a checksum or hash as an soft core is loaded. The initiator PLI 104 can compare the checksum or hash against a predetermined value to confirm the soft core was loaded correctly. The initiator PLI 104 can reload the protected image where there is an error in the checksum or hash.

[0045] The PLIs loaded with their default or protected images can be tested various built in tests (BIT) in block 338. For example, known result verifications could be performed where a known input is fed to one or more PLIs to determine if a known output is produced during a known answer verification. Other possible testing of the PLIs can be performed such as scan chain testing, check words, check sums, boundary scan, integrity test, and other tests. After completion of block 338, the cryptographic processor system 100 is in a trusted state. Beyond testing, the trusted state is implemented with redundancy in this embodiment.

[0046] Monitoring is performed in block 344 for error conditions after the known answer and other verifications. Where they are conditions that could indicate a security concern, one or more layers of the multi-layer key are deleted in block 348 before looping back to block 312 to wait for a new key layer load, which may be manually or automatically performed. From block 312, the booting process could loop back to either block 316 or block 338 depending on the severity of the error. With other errors detected in block 344, processing goes from block 344 to block 338 to test the PLIs again without removal of one or more key layers. In the absence of errors detected in block 344, the cryptographic processor system 100 is available to encrypt and decrypt traffic in a fully operational mode in block 340. Testing continues in block 344 during operation either periodically, upon certain events or when errors are suspected.

II. System Security Manager

[0047] FPGAs and/or FPLCs may have security manager PLI 136 for programming into the FPLC. For example, Xilinx™ and Altera™ have envisioned a security manager PLI 136.

The security manager PLI 136 is a soft core in this embodiment. Periodically, the security manager PLI 136 can check the other soft cores loaded into the FPLC to confirm they match what was originally loaded. Changes in the loaded programming could be detected in this manner. When an error condition is detected, the security monitor can erase the programming from within the FPGA or FPLC such that it returned to an inoperable state. The programmed-in security manager PLI 136 is not redundant. Additionally, these approaches presume that the security manager PLI 136 is operating properly despite other problems detected within the FPLC.

[0048] With reference to FIG. 2A, an embodiment of a SSM 116 is shown. In one embodiment, we include a SSM 116, that has circuitry external to the FPLCs housing the PLIs. The SSM 116 can activate the internal security monitoring circuit to erase the programming in a FPLC and/or may just overwrite the programming, reset the logic or otherwise prevent further operation of the FPLC. The SSM 116 can work in conjunction with the security monitor or replace the function of the security monitor. In this embodiment, the SSM 116 is outside the FPLC used for other PLIs, but could be implemented in an ASIC, FPGA, CPLD, or PLD. In this embodiment, a CPLD is used to implement the SSM 116 and is not field reprogrammable. Other soft cores or PLIs could be included in the FPLC used by the SSM 116 in other embodiments.

[0049] Many conditions are observed by the SSM 116, that may cause security measures to take place. Things like battery voltage over or under specification, tamper of any circuitry or enclosure, alarm conditions, triggering of a FPGA's or FPLC's security monitor are all conditions that are observed. Based on an analysis of the threat, the SSM 116 can erase/overwrite/reset PLIs and/or FPLCs, keys, and/or key portions or layers. For example, the system security monitor 116 may receive an indication that a particular FPGA security monitor found a single point failure and erased the FPGA. The system security monitor 116 could respond by writing an erasing or initialization program into the FPGA before reprogramming it once again. Certain conditions only result in erasing and/or reprogramming a portion of a PLI, a whole PLI, multiple PLIs, a FPLC, or multiple FPLCs.

[0050] This embodiment has a number of security functions that are activated based upon how the inputs are interpreted by a threat analysis circuit 216. The threat analysis circuit 216 can activate an erase circuit 204, a PLI wipe circuit 212 and/or a re-fill and test PLI (RATP) circuit 208. The erase key circuit 204 could erase, overwrite and/or otherwise

disable a key. A layer of a multi-layer key may be erased or overwritten to effectively disable use of the multi-layer key even though other layers of the multi-layer key are still available. Where keys should be disabled, for example, the erase key circuit 204 can erase layers in the battery-backed memory 106, the token and/or the storage flash 108.

- 5 [0051] The testing of the PLI in the RATP circuit 208 can be built-in-test, security monitor tests, known answer verifications or the like. Re-filling of the PLI can be of the original image, a default image or a null image. Although this embodiment of the RATP circuit 208 and PLI wipe circuit 212 operate at the PLI level, other embodiments could also optionally operate on one or more FPLCs.
- 10 [0052] Referring next to **FIG. 2B**, an embodiment of a fail-safe SSM 118 is shown that uses redundant SSMs 116. The embodiment of the SSM 116 in FIG. 2A does not operate in a failsafe mode that is trusted according to some criterions. This embodiment duplicates of all the circuitry in the SSM 116 such that both copies would have to perform the same way or that would cause an alarm condition. In this way, any failure of one of the SSMs 116
- 15 would cause an erasure of the various PLIs, FPLCs, key layers, and/or keys. A consistency check circuit 216 compares inputs and outputs of the parallel SSMs 116 to assure there are matches. If one SSM 116 goes haywire, the PLI containing the fail-safe SSM 118 can be erased after possibly erasing one or more keys or key layers.

III. Overlapping State Areas for Programmable Crypto Processing Circuits

- 20 [0053] High-assurance and classified applications generally avoid use of PLIs or FPLCs. There are concerns that the reprogramability of these devices will leave them vulnerable to compromise. When operating these devices certain logic circuits are only used in certain states. For example, crypto processing systems configure the traffic engine before operating the traffic engine to process information. An embodiment reuses at least some of the same
- 25 device resources for a configuration state and a cryptographic processing state.

- [0054] Referring to **FIG. 4A**, a block diagram of an embodiment of a traffic processing system 400-1 is shown. The traffic processing could be cryptographic or other processing on data. This embodiment processes data in a manner that can tolerate delays associated with switching into a configuration state when necessary to perform configuration and any
- 30 key management before switching back to a traffic processing state to operate upon more data. The traffic processing system 400 has a program load circuit 112 that loads multiple

soft cores into the traffic processor 404 from the storage flash 108. The program load circuit loads soft cores as a function of the operational state of the traffic processor 404.

[0055] Included in the traffic processor 404 at various times are a traffic processing soft core 424, a configuration processing soft core 416, a persistent soft core 408, traffic ports 5 418, a program memory 412, and a configuration information store 420. Soft cores are outlined in the figure with dashed lines and are loaded into a FPLC as images by the program load circuit 112. The program memory 412 holds software for execution by the configuration processing soft core 416. The software can be loaded by the program load circuit 112. One or more storage media within the FPLC that implements the traffic 10 processor are used for the program memory and the configuration information store 420 in this embodiment.

[0056] The program load circuit 112 has pointers to know where the various images are loaded in the storage flash 108 for the various states. The next state is communicated to the program load circuit 112 and the pointer is found to know which addresses from the storage 15 flash 108 to feed into the traffic processor FPLC 404. The stream of programming information is fed from the storage flash 108 into the programming interface of the FPLC by the program load circuit.

[0057] The configuration processing soft core 416 performs configuration for the other states of operation, for example, key loading and management, decryption of classified 20 images, built-in test, etc. In this embodiment, the configuration processing soft core 416 includes a processor, but other embodiments could perform the same actions without use of a processor. The produced configuration information is recorded in the configuration information store 420 and includes various things such as decoded keys, operational parameters, cryptographic algorithm variables, data ports to use, configuration of data 25 passed to/from the traffic processing soft core 424.

[0058] The persistent soft core 408 could be used for loading PLIs or soft cores, optionally decrypting PLIs or soft cores, managing keys and security, and/or a state machine for flipping between the various cores used by the various states. In this embodiment, the persistent soft core 408 aids in loading images and storing semaphores or parameters that 30 are passed between states. The state machine for flipping between various cores could be external to the traffic processor 404 in some embodiments. Other embodiments could have transitions to another state decided by a loaded PLI or soft core. For example, when the

traffic processing soft core 404 detected an error that required reset, it could trigger loading the configuration processing soft core 416 and pass it the appropriate parameters.

5 [0059] For FPLCs that have partial reconfiguration, the state machine and image loading logic can be within the FPLCs to allow the state machine to remain during reprogramming of the FPLC. Other embodiments could have the image loading logic external when the FPLC does not support partial reconfiguration. The reconfiguration of the entire FPLC could triggered by a state machine that would be overwritten in the process of reconfiguring. The new configuration could have a new state machine capable of triggering a transition to another state that would use the external image loading logic to load a new
10 image into the FPLC.

[0060] Once configuration is complete, this embodiment has no further need for a general-purpose processor. The traffic processing soft core 424 and traffic ports 418 can be loaded by the program load circuit and consume some of the same resources in the device that were consumed by the configuration processing soft core 416. The traffic processing
15 soft core 424 does not use any software and instead is controlled by state engines in this embodiment. Information needed for the traffic processing soft core 424 are available from the configuration information store 420 in the traffic processing state. The traffic ports 418 are used by the traffic processing soft core 424 to send and receive information with the traffic processor 404. Other embodiments could avoid use of general-purpose processors
20 for any state and rely on state machines for each state instead.

[0061] If additional configuration is needed at some point, data processing temporarily ceases and the traffic processor 404 returns to the configuration state. Data for processing could be buffered or otherwise delayed until return to the traffic processing state once the configuration state completes. The traffic processor 404 flops between states and the
25 program load circuit 112 loads the necessary soft cores to allow operation with a reduction in resources for the device.

[0062] With reference to **FIG. 4B**, a block diagram of another embodiment of the traffic processing system 400-2 is shown. This embodiment uses a volatile memory 132 external to the FPLC implementing the traffic processor 404 to store the program memory 412 and
30 the configuration information store 420. A memory interface (not shown) can be another soft core that is used to interact with the volatile memory 132.

[0063] Referring next to **FIG. 4C**, a block diagram of yet another embodiment of the traffic processing system 400-3 is shown. This embodiment of the traffic processing system 400-3 has an initiator PLI 104 as its configuration processing soft core 416. Additionally, the traffic processing soft core 424 is replaced with the crypto PLI 124. The functionality of the initiator PLI 104 and the traffic processing soft core 424 is described above.

[0064] With reference to **FIG. 4D**, a block diagram of still another embodiment of the traffic processing system 400-4 is shown. In this embodiment, the configuration information store 420 is retained within the FPLC of the traffic processor 404. The FPLC could have embedded memory or could use a soft core to implement memory to hold the configuration information between states.

[0065] With reference to **FIGs. 5A & 5B**, diagrams of an embodiment of a layout of a traffic processor 404 is shown in two different states. This embodiment of the traffic processor includes memory within the FPLC. The memory is used to store the software memory 412 and the configuration information 420. FIG. 5A shows the loaded soft cores used in a configuration state, and FIG. 5B shows the soft cores loaded in a traffic processing state for this embodiment. The configuration state in FIG. 5A, has a persistent soft core 408, a configuration processing soft core 416 in addition to the onboard memory. In transitioning to the traffic processing state shown in FIG. 5B, the configuration processing soft core 416 is overwritten with a traffic processing soft core 424 and traffic ports 418. Traffic ports 418 are not used in the configuration state 604 as these traffic ports are used to pass traffic.

[0066] Referring next to **FIGs. 5C & 5D**, diagrams of another embodiment of a layout of a traffic processor 404 is shown in two different states. The layout of the soft cores in a configuration state is shown FIG. 5C, and the layout of the soft cores in a traffic processing state is shown in FIG. 5D. This embodiment uses an external battery-backed memory 106. A memory interface soft core 508 is present in both states to interface to the battery-backed memory 106. Processing can flip back and forth between the two states as necessary during normal operation. Certain soft cores partially or wholly overlap for different states.

[0067] With reference to **FIGs. 5E & 5F**, diagrams of one embodiment of a layout of a traffic processor 404 is shown in two different states. This traffic processor 404 performs cryptographic processing, for example, using the cryptographic processor system 100. In a first state shown in FIG. 5E, an initiator PLI 104 is used to perform configuration, key

management, etc. The crypto PLI 124 and interface PLI 140 are loaded in a second state in a manner that overlaps partially or wholly the initiator PLI 104. Operation can move between these states during normal operation of the traffic processor 404.

5 [0068] Referring next to **FIGs. 5G & 5H**, diagrams of another embodiment of a layout of a traffic processor 404 is shown in two different states. This traffic processor 404 performs cryptographic processing, for example, using the cryptographic processor system 100. Operation moves between at least two states, with one having an initiator PLI 104 loaded and the other having a crypto PLI 124 loaded. In this embodiment, there is no overlap between the PLIs swapped out between states. While in one state there are one or more
10 PLIs that are inoperable unless there is a transition to the state that would use that PLI.

[0069] Referring next to **FIG. 6A**, an embodiment of a state machine 600-1 used to control the traffic processor 404 is shown. This embodiment operates in two states, namely, a configuration state 604 and a traffic processing state 608. After reset or boot, processing enters the configuration state 604. Once configuration is complete, processing goes to the
15 traffic processing state 608. When more configuration is needed, processing goes from the traffic processing state 608 to the configuration state 604 before returning back to the traffic processing state 608. Appropriate images are loaded into the device when switching between states. The size of the FPLC is dictated by the largest amount of logic in any state.

[0070] With reference to **FIG. 6B**, another embodiment of the state machine 600-2 used
20 to control the traffic processor 404 is shown. This embodiment controls a cryptographic processor system 100 in three states. A control and key management state 612 and a cryptographic processor setup state 616 use a configuration processing soft core 416, but a cryptographic processing state 620 has no need for a general-purpose processor. Between each state, different soft cores can be loaded into the cryptographic processor system 100.
25 In some embodiments, the control and key management state 612 and a cryptographic processor setup state 616 may use the same soft cores such that loading images between states is not required.

[0071] Referring next to **FIG. 7**, an embodiment of a flow diagram showing a process 700
30 for cryptographically processing information in a two state configuration. The depicted portion of the process begins in block 704 where the storage flash 108 is loaded with configuration images used for the various soft cores. In block 708, the storage flash 108 is loaded with the traffic state images used in the various soft cores for that state.

Additionally, software for a general purpose processing core can be loaded into the storage flash 108 or directly into the program memory 412.

5 [0072] Normal operation of the cryptographic processor system 100 begins in block 712 when the PLI or FPLC is reset or powered-up. The configuration image(s) are loaded into the device in block 716. The traffic processor 404 begins operation in the configuration state to initialize the PLIs in block 720 to begin the configuration state and assembles configuration information for use in other state(s) in block 724. The configuration information is stored in the configuration information store 420 in block 728.

10 [0073] Transitioning into block 732 activates the traffic processing state by loading traffic processing soft core(s) into the traffic processor 404. Loading of new soft cores may be preceded by overwriting and/or erasing of prior soft cores or may simply be accomplished by writing the new soft cores over the prior soft cores. In the traffic processing state, the traffic image is configured with the stored traffic configuration information upon activating the traffic processing soft core in block 736. Processing traffic takes place in block 740.
15 Other embodiments could perform cryptographic processing in block 740, for example, using the cryptographic processor system 100. So long as configuration is not needed or errors are not detected in block 744, traffic processing continues. Any error detected in block 744 is recorded in block 748 before looping back to block 712 where the device is rebooted. Other embodiments could take any number of remedial measures depending on
20 the error encountered, for example keys or a key layer(s) could be destroyed in a cryptographic application.

[0074] A number of variations and modifications of the disclosed embodiments can also be used. For example, functions that are implemented in PLIs could be performed in software. Code could be run on soft cores within a FPLC or a pre-programmed
25 microprocessor circuit.

[0075] Specific details are given in the above description to provide a thorough understanding of the embodiments. However, it is understood that the embodiments may be practiced without these specific details. For example, circuits may be shown in block diagrams in order not to obscure the embodiments in unnecessary detail. In other instances,
30 well-known circuits, processes, algorithms, structures, and techniques may be shown without unnecessary detail in order to avoid obscuring the embodiments.

[0076] Implementation of the techniques, blocks, steps and means described above may be done in various ways. For example, these techniques, blocks, steps and means may be implemented in hardware, software, or a combination thereof. For a hardware implementation, the processing units may be implemented within one or more application specific integrated circuits (ASICs), digital signal processors (DSPs), digital signal processing devices (DSPDs), programmable logic devices (PLDs), field-programmable gate arrays (FPGAs), soft core processors, hard core processors, controllers, micro-controllers, microprocessors, other electronic units designed to perform the functions described above, and/or a combination thereof. Software can be used instead of or in addition to hardware to perform the techniques, blocks, steps and means.

[0077] Also, it is noted that the embodiments may be described as a process which is depicted as a flowchart, a flow diagram, a data flow diagram, a structure diagram, or a block diagram. Although a flowchart may describe the operations as a sequential process, many of the operations can be performed in parallel or concurrently. In addition, the order of the operations may be re-arranged. A process is terminated when its operations are completed, but could have additional steps not included in the figure. A process may correspond to a method, a function, a procedure, a subroutine, a subprogram, etc. When a process corresponds to a function, its termination corresponds to a return of the function to the calling function or the main function.

[0078] Furthermore, embodiments may be implemented by hardware, software, scripting languages, firmware, middleware, microcode, hardware description languages, and/or any combination thereof. When implemented in software, firmware, middleware, scripting language, and/or microcode, the program code or code segments to perform the necessary tasks may be stored in a machine readable medium such as a storage medium. A code segment or machine-executable instruction may represent a procedure, a function, a subprogram, a program, a routine, a subroutine, a module, a software package, a script, a class, or any combination of instructions, data structures, and/or program statements. A code segment may be coupled to another code segment or a hardware circuit by passing and/or receiving information, data, arguments, parameters, and/or memory contents. Information, arguments, parameters, data, etc. may be passed, forwarded, or transmitted via any suitable means including memory sharing, message passing, token passing, network transmission, etc.

[0079] For a firmware and/or software implementation, the methodologies may be implemented with modules (e.g., procedures, functions, and so on) that perform the functions described herein. Any machine-readable medium tangibly embodying instructions may be used in implementing the methodologies described herein. For
5 example, software codes may be stored in a memory. Memory may be implemented within the processor or external to the processor. As used herein the term "memory" refers to any type of long term, short term, volatile, nonvolatile, or other storage medium and is not to be limited to any particular type of memory or number of memories, or type of media upon which memory is stored.

10 [0080] Moreover, as disclosed herein, the term "storage medium" may represent one or more memories for storing data, including read only memory (ROM), random access memory (RAM), magnetic RAM, core memory, magnetic disk storage mediums, optical storage mediums, flash memory devices and/or other machine readable mediums for storing information. The term "machine-readable medium" includes, but is not limited to portable
15 or fixed storage devices, optical storage devices, wireless channels, and/or various other storage mediums capable of storing that contain or carry instruction(s) and/or data.

[0081] While the principles of the disclosure have been described above in connection with specific apparatuses and methods, it is to be clearly understood that this description is made only by way of example and not as limitation on the scope of the disclosure.

CLAIMS

WHAT IS CLAIMED IS:

- 1 1. A method for booting with multi-layered security that
2 cryptographically processes information, the method comprising:
3 loading a default image into a field-programmable logic chip (FPLC),
4 wherein:
5 the default image prevents the FPLC from passing
6 information from a first port to a second port,
7 the default image includes a first algorithm that is
8 unclassified by the government, and
9 the default image allows operational software to function;
10 loading a multi-layered key, wherein:
11 a first layer of the multilayer key is stored separate from a
12 second layer of the multilayer key, and
13 the first and second layers are used to formulate the multi-
14 layer key;
15 decrypting a protected image using the multi-layered key and the first
16 algorithm to produce a decrypted image, wherein the protected image is encrypted;
17 loading the decrypted image into the FPLC; and
18 performing cryptographic processing using the decrypted image and a second
19 algorithm, which is classified by the government, wherein the cryptographic processing
20 operates on information passing from the first port to the second port.
- 1 2. The method as recited in claim 1, wherein the unclassified algorithm
2 is software implemented.
- 1 3. The method as recited in claim 1, wherein the operation software is
2 run on a processor external to the FPLC.
- 1 4. The method as recited in claim 1, wherein the unclassified algorithm
2 is a soft core.
- 1 5. The method as recited in claim 1, wherein the loading the default
2 image is performed after decrypting the default image on-chip within the FPLC.

1 6. The method as recited in claim 1, wherein the loading the multi-layer
2 key includes loading the first and second layers of the multi-layered key from at least two
3 different chips.

1 7. The method as recited in claim 1, wherein the FPLC includes a
2 plurality of FPGAs.

1 8. The method as recited in claim 1, wherein the default image is
2 isolated from the decrypted image within a single field-programmable gate array (FPGA).

1 9. A cryptographic processing system for booting a field-programmable
2 logic chip (FPLC) with multi-layered keys, the cryptographic processing system
3 comprising:

4 a first port for receiving information for cryptographic processing;

5 a second port for transmitting information after cryptographic processing;

6 a default image loaded in the FPLC, wherein:

7 the default image prevents the FPLC from passing
8 information from a first port to a second port,

9 the default image includes a first algorithm that is
10 unclassified by the government, and

11 the default image allows operational software to function;

12 a multi-layered key, wherein:

13 a first layer of the multilayer key is stored separate from a
14 second layer of the multilayer key, and

15 the first and second layers are used to formulate the multi-
16 layer key;

17 a protected image that is decrypted using the multi-layered key and the first
18 algorithm to produce a decrypted image, wherein:

19 the decrypted image is loaded into the FPLC, and

20 cryptographic processing is performed using the decrypted
21 image and a second algorithm, which is classified by the government,
22 wherein the cryptographic processing operates on information passing from
23 the first port to the second port.

1 10. The cryptographic processing system as recited in claim 9, wherein
2 the unclassified algorithm is software implemented.

1 11. The cryptographic processing system as recited in claim 9, wherein
2 the unclassified algorithm is a soft core.

1 12. The cryptographic processing system as recited in claim 9, wherein
2 the loading the default image is performed after decrypting the default image on-chip within
3 the FPLC.

1 13. The cryptographic processing system as recited in claim 9, wherein
2 the loading the multi-layer key includes loading the first and second layers of the multi-
3 layered key from at least two different chips.

1 14. The cryptographic processing system as recited in claim 9, wherein
2 the FPLC includes a plurality of FPGAs.

1 15. The cryptographic processing system as recited in claim 9, wherein
2 the default image is isolated from the decrypted image within a single field-programmable
3 gate array (FPGA).

1 16. A cryptographic processing system for booting a field-programmable
2 logic chip (FPLC) with multi-layered keys, the cryptographic processing system
3 comprising:

4 first means for loading a default image into a field-programmable logic chip
5 (FPLC), wherein:

6 the default image prevents the FPLC from passing
7 information from a first port to a second port,

8 the default image includes a first algorithm that is
9 unclassified by the government, and

10 the default image allows operational software to function;

11 second means for loading a multi-layered key, wherein:

12 a first layer of the multilayer key is stored separate from a
13 second layer of the multilayer key, and

14 the first and second layers are used to formulate the multi-
15 layer key;
16 means for decrypting a protected image using the multi-layered key and the
17 first algorithm to produce a decrypted image;
18 third means for loading the decrypted image into the FPLC; and
19 means for cryptographically processing that uses the decrypted image and a
20 second algorithm, which is classified by the government, wherein the cryptographic
21 processing operates on information passing from the first port to the second port.

1 17. The cryptographic processing system as recited in claim 16, wherein
2 the unclassified algorithm is software implemented.

1 18. The cryptographic processing system as recited in claim 16, wherein
2 the unclassified algorithm is a soft core.

1 19. The cryptographic processing system as recited in claim 16, wherein
2 the loading the default image is performed after decrypting the default image on-chip within
3 the FPLC.

1 20. The cryptographic processing system as recited in claim 16, wherein
2 the loading the multi-layer key includes loading the first and second layers of the multi-
3 layered key from at least two different chips.

1 21. The cryptographic processing system as recited in claim 16, wherein
2 the FPLC includes a plurality of FPGAs.

1 22. The cryptographic processing system as recited in claim 16, wherein
2 the default image is isolated from the decrypted image within a single field-programmable
3 gate array (FPGA).

1 23. A method for providing system security for a field-programmable
2 logic chip (FPLC), the method comprising:
3 cryptographically processing information within the FPLC;
4 detecting an error condition, wherein the detection is performed external to
5 the FPLC;
6 conveying into the FPLC the error condition; and

7 disrupting an image in the FPLC for one or more soft cores loaded when the
8 error condition is detected where the image cannot perform cryptographic processing after
9 disruption even if keys are present.

1 24. The method as recited in claim 23, wherein the key is used to
2 program the image into the FPLC.

1 25. The method as recited in claim 23, wherein the conveying is
2 performed by disrupting programming of the FPLC.

1 26. The method as recited in claim 23, further comprising erasing at least
2 a portion of a key used for the cryptographic processing.

1 27. The method as recited in claim 23, wherein the key is broken into
2 portions stored in at least two different memories.

1 28. The method as recited in claim 23, wherein the disrupting the image
2 includes writing a different image into the FPLC over at least part of the image.

1 29. The method as recited in claim 23, wherein the disrupting the image
2 is performed by sate machine.

1 30. The method as recited in claim 23, further comprising detecting the
2 error condition using circuitry that is redundantly implemented.

1 31. A cryptographic processing system for providing system security for
2 a field-programmable logic chip (FPLC), the cryptographic processing system comprising:
3 a cryptographic soft core at least partially within the FPLC that
4 cryptographically processes information;
5 a system security manager that detects an error condition, wherein:
6 the detection is performed external to the FPLC,
7 the cryptographic soft core in the FPLC is disrupted when the
8 error condition is detected where the image, and
9 the cryptographic soft core cannot perform cryptographic
10 processing after disruption even if keys are present.

1 32. The cryptographic processing system as recited in claim 31, wherein
2 the key is used to program the cryptographic soft core into the FPLC.

1 33. The cryptographic processing system as recited in claim 31, wherein
2 at least a portion of a key used for the cryptographic processing is disrupted after detecting
3 the error condition

1 34. The cryptographic processing system as recited in claim 31, wherein
2 the key is broken into portions stored in at least two different memories.

1 35. The cryptographic processing system as recited in claim 31, further
2 comprising a security manager programmable logic image (PLI) internal to the FPLC.

1 36. The cryptographic processing system as recited in claim 31, wherein
2 the disrupting the image includes writing a different image into the FPLC.

1 37. The cryptographic processing system as recited in claim 31, wherein
2 the disrupting the image is performed by sate machine.

1 38. The cryptographic processing system as recited in claim 31, further
2 comprising detecting the error condition using circuitry that is redundantly implemented.

1 39. A cryptographic processing system for providing system security for
2 a field-programmable logic chip (FPLC), the cryptographic processing system comprising:
3 means for cryptographically processing information within the FPLC;
4 means for detecting an error condition, wherein the detection is performed
5 external to the FPLC;
6 means for disrupting an image in the FPLC for one or more soft cores loaded
7 when the error condition is detected where the image cannot perform cryptographic
8 processing after disruption even if keys are present.

1 40. The cryptographic processing system as recited in claim 39, wherein
2 the key is used to program the image into the FPLC.

1 41. The cryptographic processing system as recited in claim 39, further
2 comprising means for erasing at least a portion of a key used for the cryptographic
3 processing.

1 42. The cryptographic processing system as recited in claim 39, wherein
2 the key is broken into portions stored in at least two different memories.

1 43. The cryptographic processing system as recited in claim 39, wherein
2 the disrupting the image includes means for writing a different image into the FPLC.

1 44. The cryptographic processing system as recited in claim 39, wherein
2 the disrupting the image is performed by sate machine.

1 45. The cryptographic processing system as recited in claim 39, wherein
2 the means for detecting the error condition uses circuitry that is redundantly implemented.

1 46. A method for operating a field-programmable logic chip (FPLC) for
2 cryptographic processing, the method comprising:

3 operating in a configuration state using a configuration processing soft core,
4 wherein:

5 the configuration state is used to configure a cryptographic
6 processing state with configuration information, and

7 the configuration processing soft core consumes a first
8 portion of the FPLC;

9 switching from the configuration state to the cryptographic processing state,
10 where the switching is controlled by a state machine;

11 operating in the cryptographic processing state using a cryptographic
12 processing soft core, wherein:

13 the cryptographic processing state performs cryptographic
14 processing,

15 the cryptographic processing soft core consumes a second
16 portion of the FPLC, and

17 the first portion and the second portion overlap within the
18 FPLC such that only one of the configuration processing soft core or the
19 cryptographic processing soft core can be loaded into the FPLC at a time.

1 47. The method as recited in claim 46, wherein the FPLC isolates
2 portions within the FPLC without routing certain signals between the isolated portions.

1 48. The method as recited in claim 46, further comprising switching from
2 the cryptographic processing state to the configuration state.

1 49. The method as recited in claim 46, further comprising overwriting the
2 first portion with the second portion.

1 50. The method as recited in claim 46, wherein the configuration state
2 performs key management for the cryptographic processing state.

1 51. The method as recited in claim 46, wherein:
2 the configuration state uses a soft core of a general-purpose processor within
3 a third portion,
4 the first portion is at least partially coextensive in area with the third portion,
5 and
6 the cryptographic processing state overwrites at least part of the third portion
7 with the second portion.

1 52. The method as recited in claim 46, wherein the cryptographic
2 processing state is controlled by one or more state machines to process traffic without use of
3 software.

1 53. The method as recited in claim 46, wherein the configuration
2 information is stored within the FPLC.

1 54. The method as recited in claim 46, wherein the FPLC is a single
2 field-programmable gate array (FPGA).

1 55. The method as recited in claim 46, wherein the FPLC is programmed
2 with a flash memory using a circuit external to the FPLC.

1 56. A cryptographic traffic processor for processing information using a
2 field-programmable logic chip (FPLC), the cryptographic traffic processor comprising:
3 a configuration processing soft core;
4 a cryptographic processing soft core; and
5 a state machine having a configuration state and a traffic processing state,
6 wherein:

7 the configuration state uses the configuration processing soft
8 core,
9 the configuration state is used to configure the traffic
10 processing state with configuration information,
11 the configuration processing soft core consumes a first
12 portion of the FPLC,
13 the configuration state switches to the traffic processing state
14 under the control of the state machine,
15 the traffic processing state uses the traffic processing soft
16 core,
17 the traffic processing state performs traffic processing,
18 the traffic processing soft core consumes a second portion of
19 the FPLC, and
20 the first portion and the second portion overlap at different
21 times within the FPLC such that only one of the configuration processing
22 soft core or the traffic processing soft core can be loaded into the FPLC at a
23 time.

1 57. The traffic processor as recited in claim 56, wherein the FPLC
2 isolates portions within the FPLC without routing certain signals between the isolated
3 portions.

1 58. The traffic processor as recited in claim 56, wherein:
2 the configuration state uses a soft core of a general-purpose processor within
3 a third portion,
4 the first portion is at least partially coextensive in area with the third portion,
5 and
6 the traffic processing state overwrites at least part of the third portion with
7 the second portion.

1 59. The traffic processor as recited in claim 56, wherein the traffic
2 processing state is controlled by one or more state machines to process traffic without use of
3 software.

1 60. The traffic processor as recited in claim 56, wherein the configuration
2 information is stored within the FPLC.

1 61. The traffic processor as recited in claim 56, wherein the FPLC is a
2 single field-programmable gate array (FPGA).

1 62. The traffic processor as recited in claim 56, wherein the FPLC is
2 programmed with a flash memory using a circuit external to the FPLC.

1 63. A cryptographic traffic processor for processing information using a
2 field-programmable logic chip (FPLC), the cryptographic traffic processor comprising:

3 means for operating in a configuration state using a configuration processing
4 soft core, wherein:

5 the configuration state is used to configure a traffic
6 processing state with configuration information, and

7 the configuration processing soft core consumes a first
8 portion of the FPLC;

9 means for switching from the configuration state to the traffic processing
10 state, where the switching is controlled by a state machine;

11 means for operating in the traffic processing state using a traffic processing
12 soft core, wherein:

13 the traffic processing state performs traffic processing,

14 the traffic processing soft core consumes a second portion of
15 the FPLC, and

16 the first portion and the second portion overlap within the
17 FPLC such that only one of the configuration processing soft core or the
18 traffic processing soft core can be loaded into the FPLC at a time.

1 64. The traffic processor as recited in claim 63, wherein the FPLC
2 isolates portions within the FPLC without routing certain signals between the isolated
3 portions.

1 65. The traffic processor as recited in claim 63, wherein:
2 the configuration state uses a soft core of a general-purpose processor within
3 a third portion,

4 the first portion is at least partially coextensive in area with the third portion,
5 and
6 the traffic processing state overwrites at least part of the third portion with
7 the second portion.

1 66. The traffic processor as recited in claim 63, wherein the traffic
2 processing state is controlled by one or more state machines to process traffic without use of
3 software.

1 67. The traffic processor as recited in claim 63, wherein the configuration
2 information is stored within the FPLC.

1 68. The traffic processor as recited in claim 63, wherein the FPLC is a
2 single field-programmable gate array (FPGA).

1 69. The traffic processor as recited in claim 63, wherein the FPLC is
2 programmed with a flash memory using a circuit external to the FPLC.

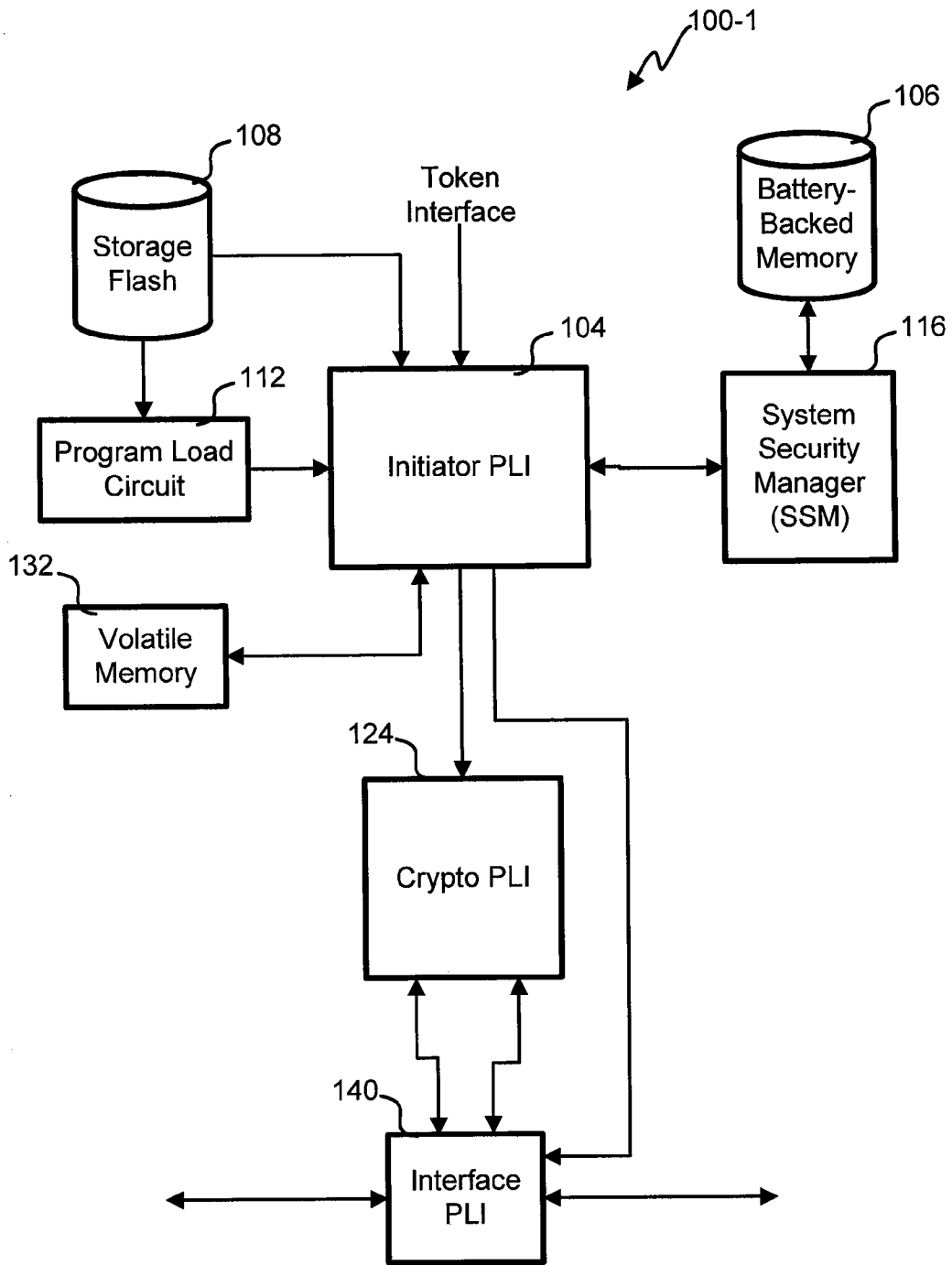


Fig. 1A

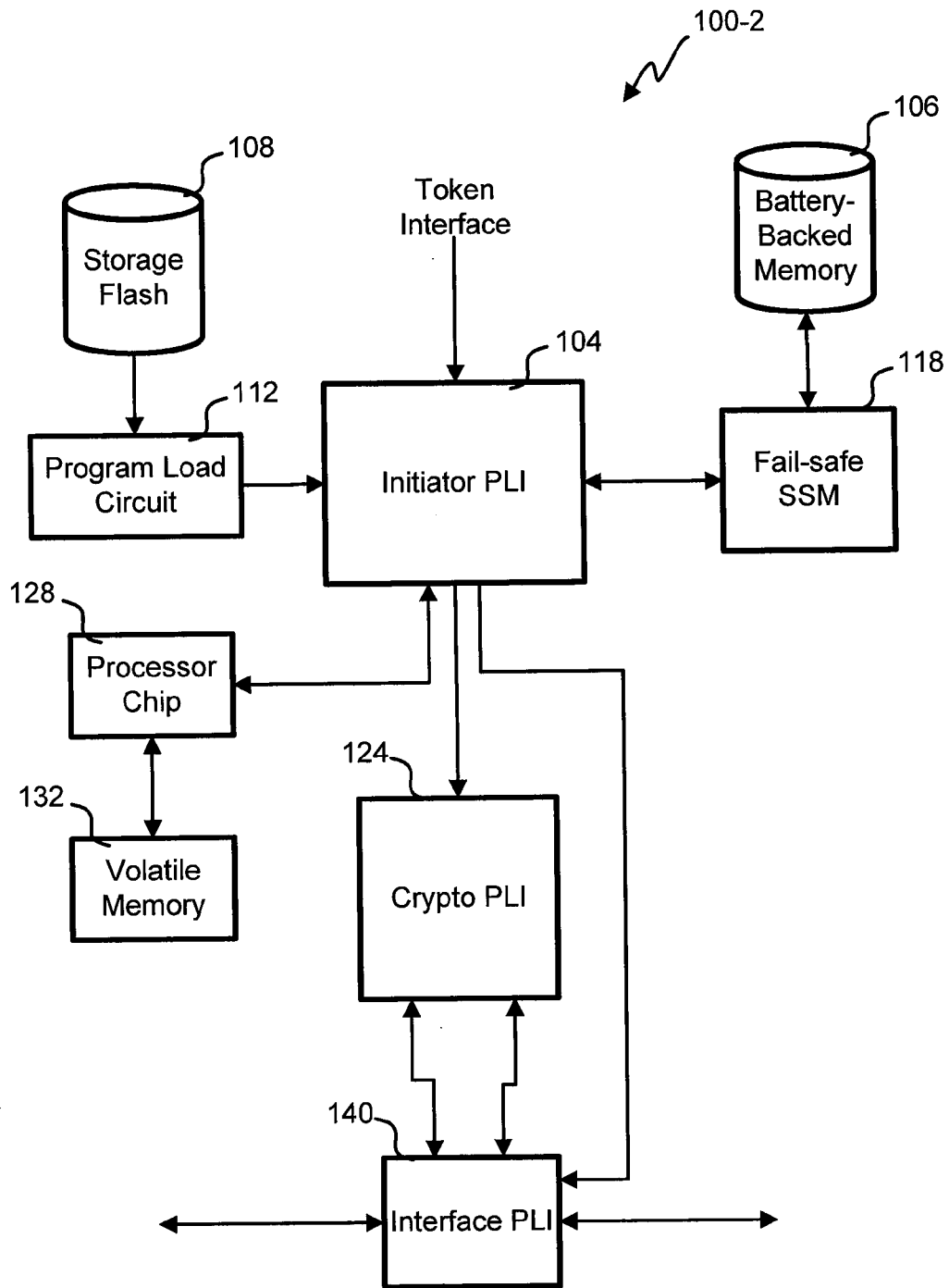


Fig. 1B

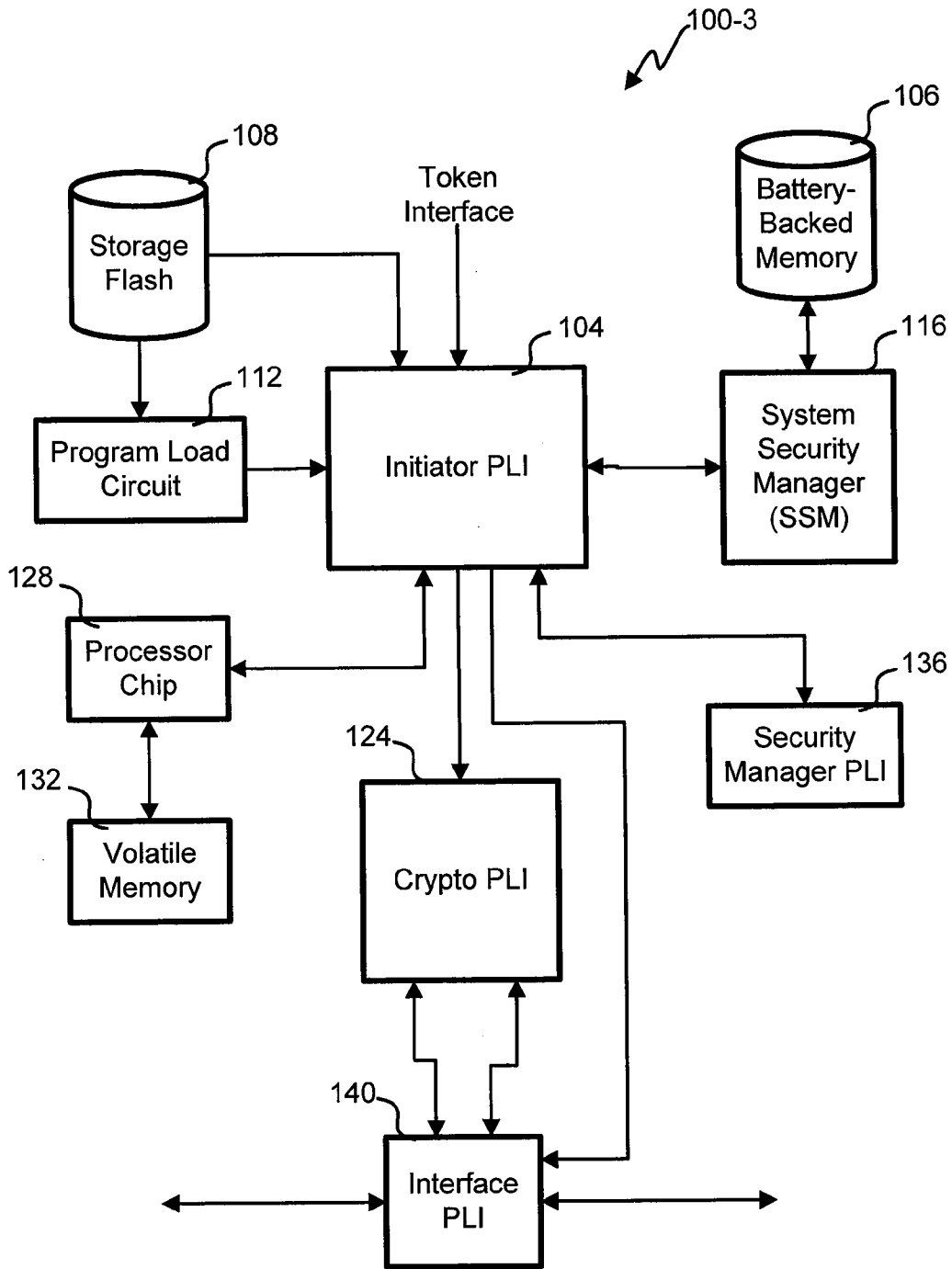


Fig. 1C

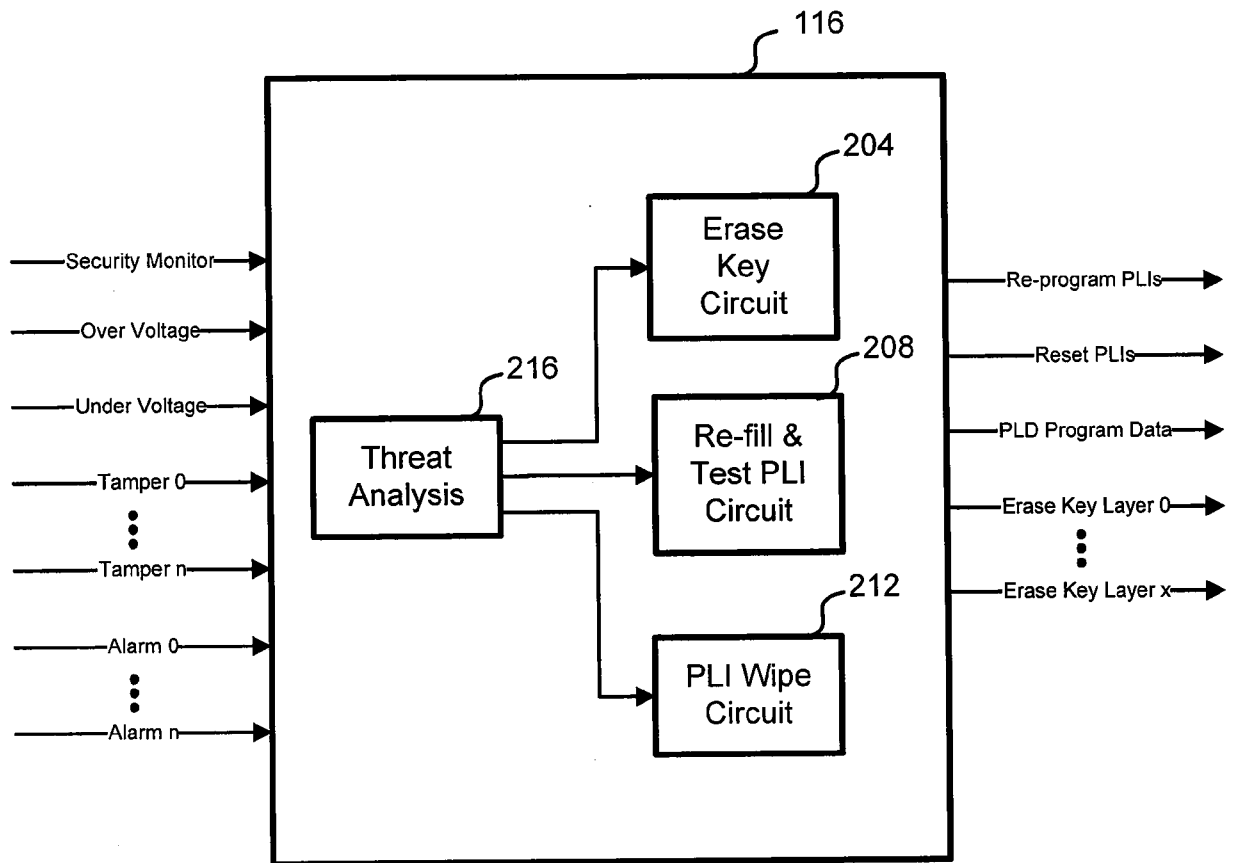


Fig. 2A

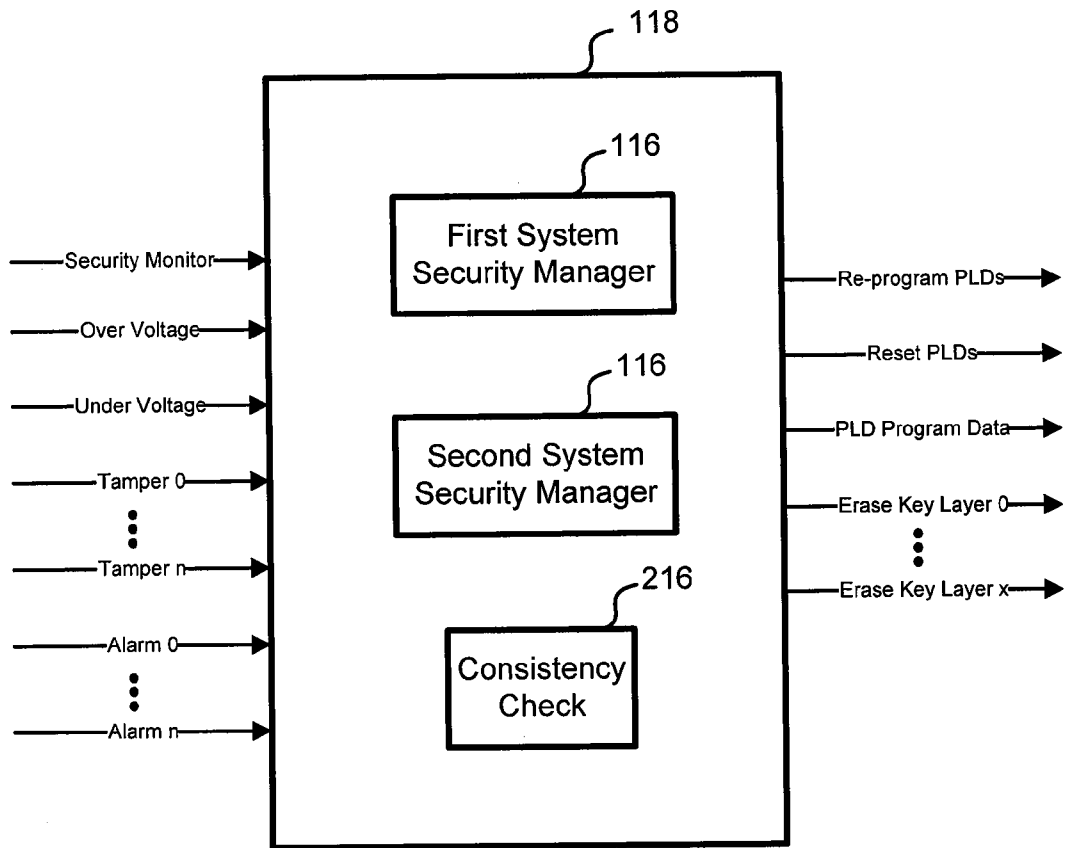


Fig. 2B

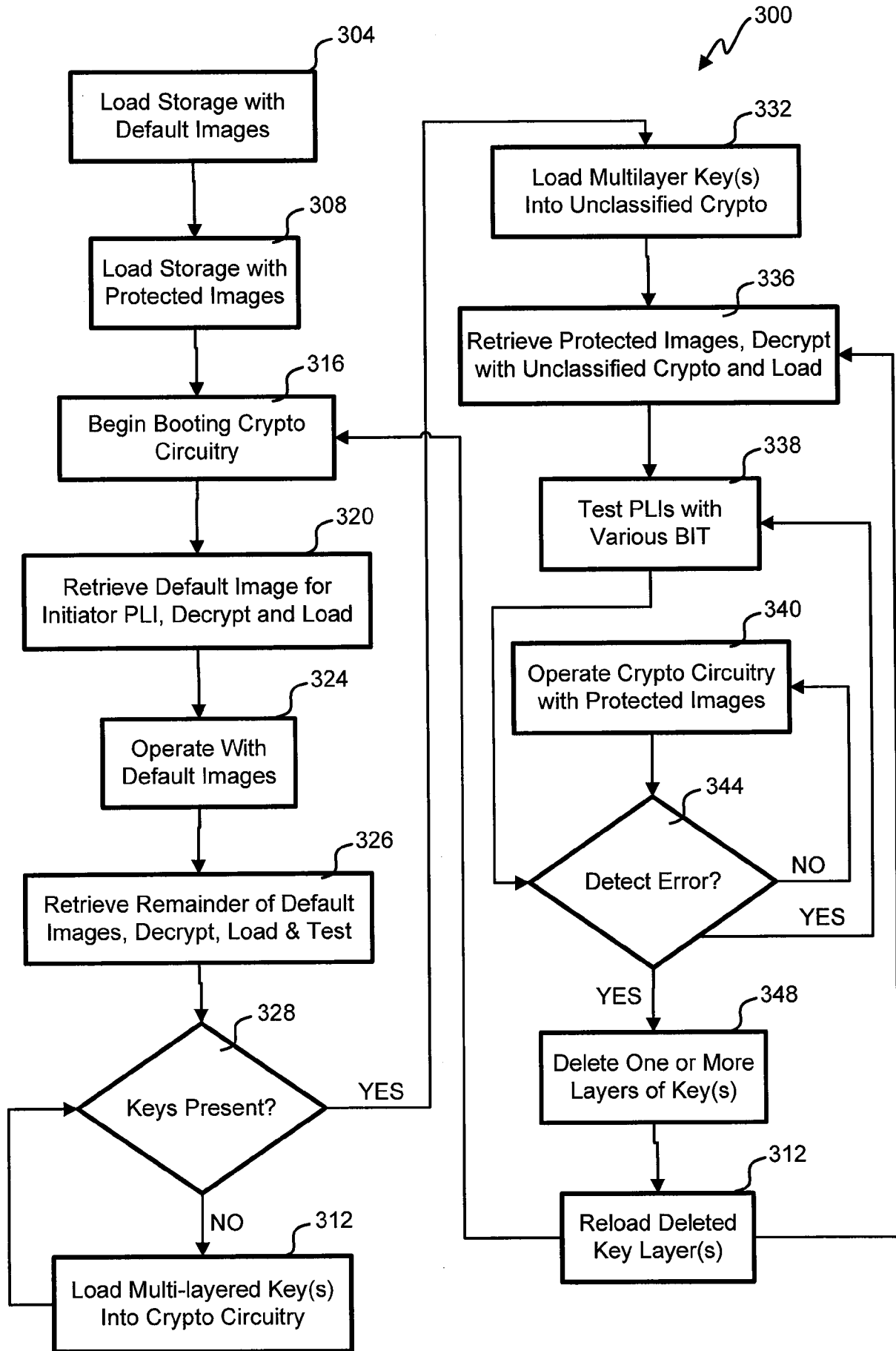


Fig. 3

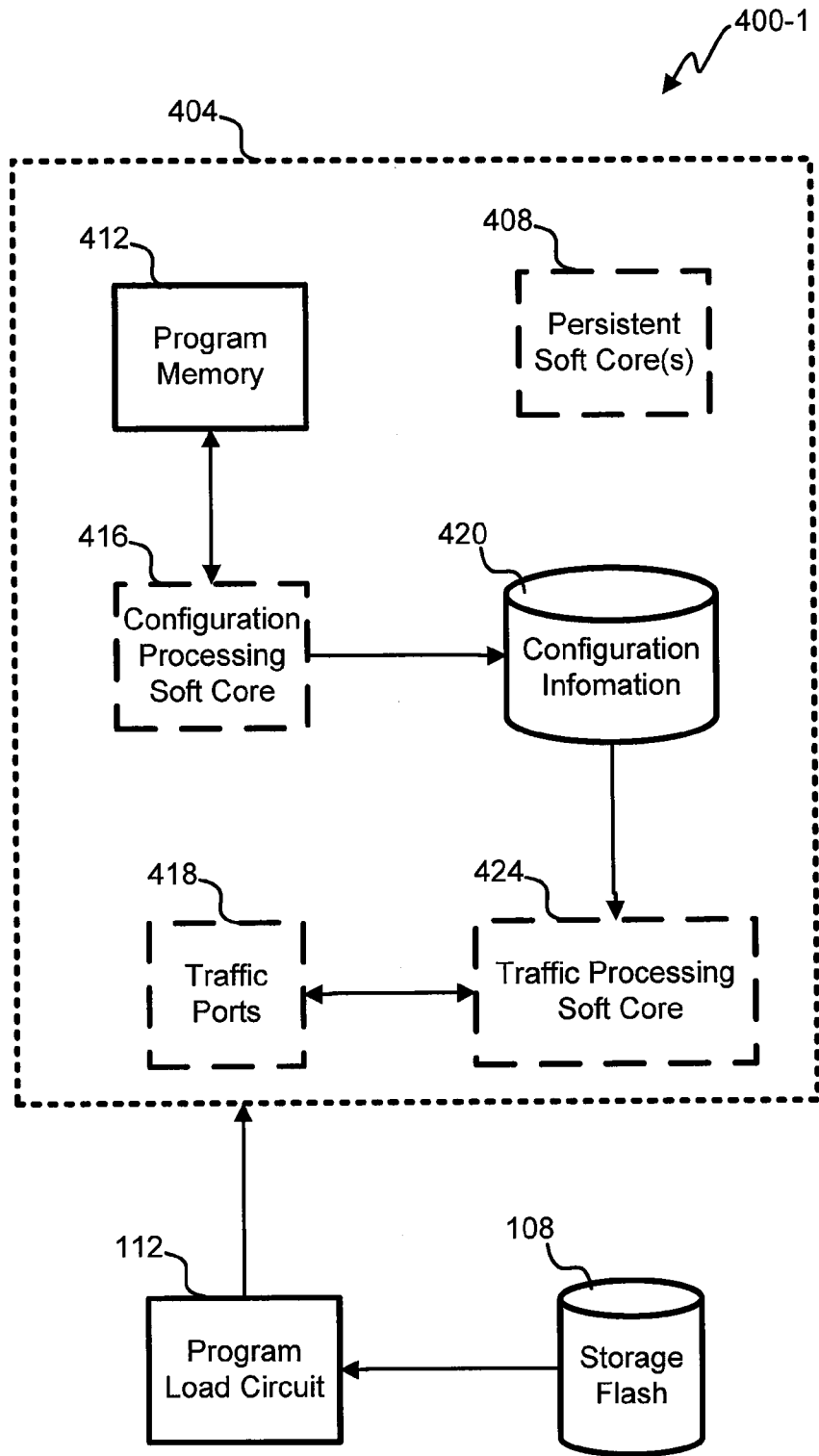


Fig. 4A

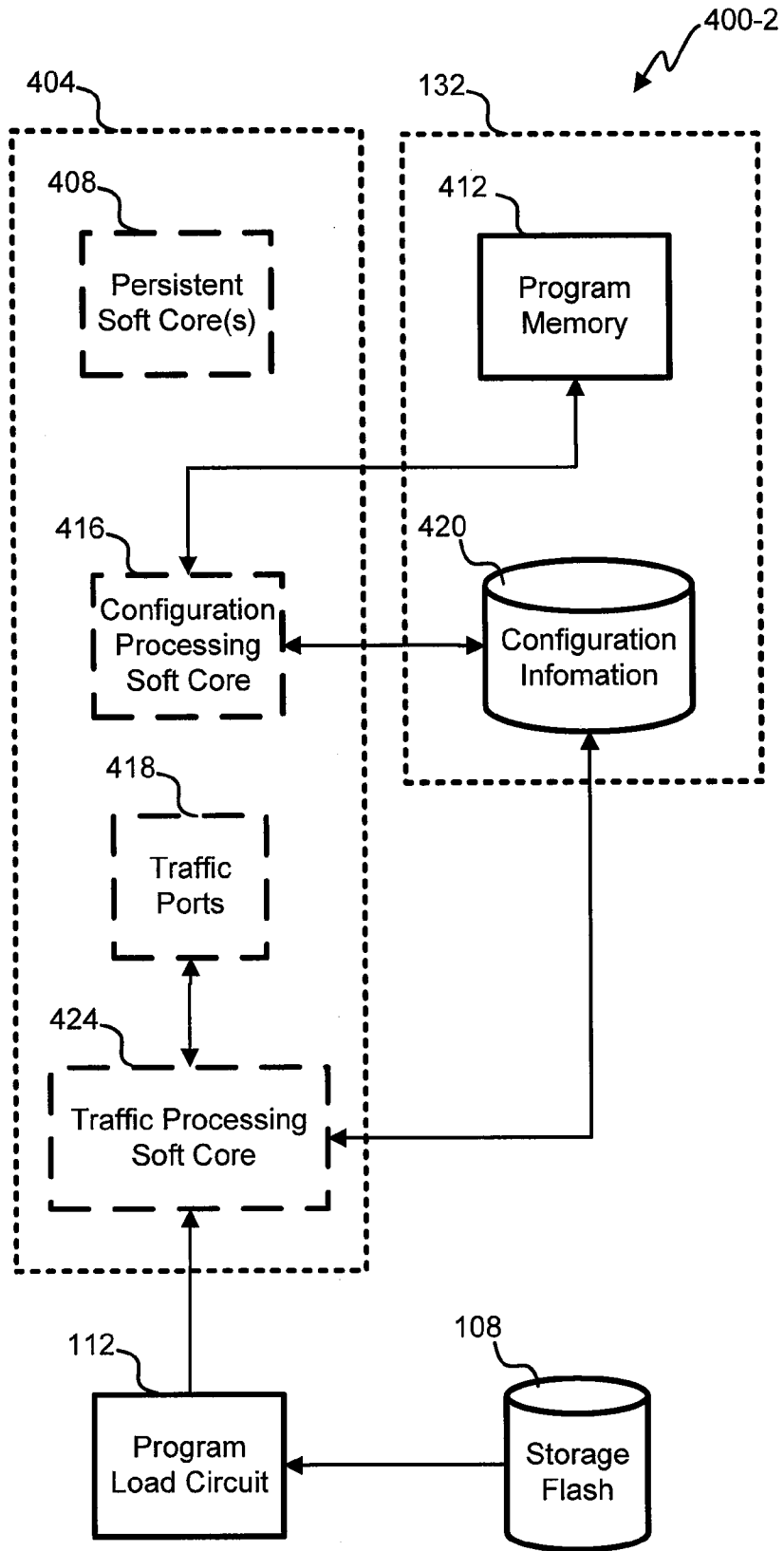


Fig. 4B

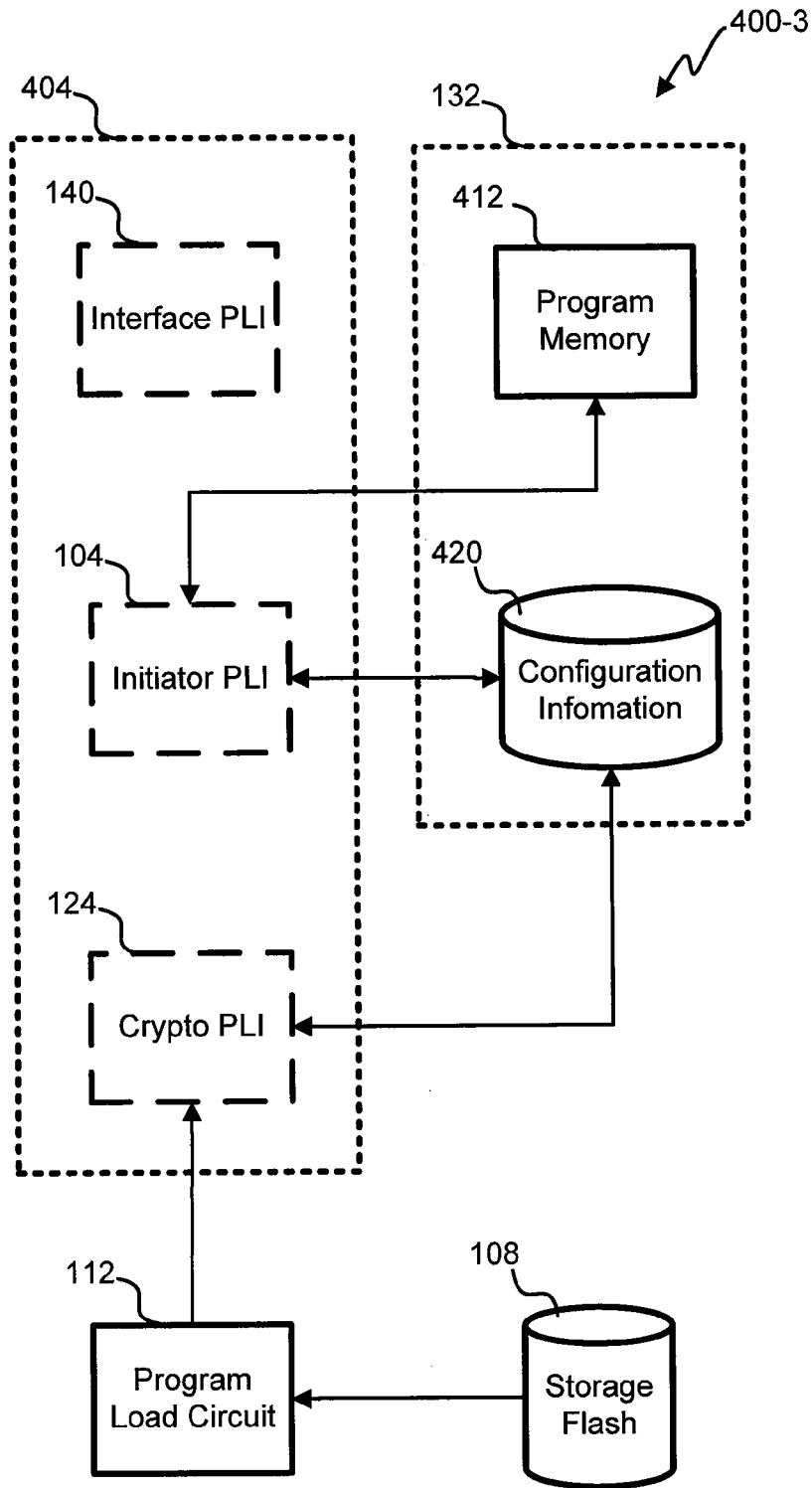


Fig. 4C

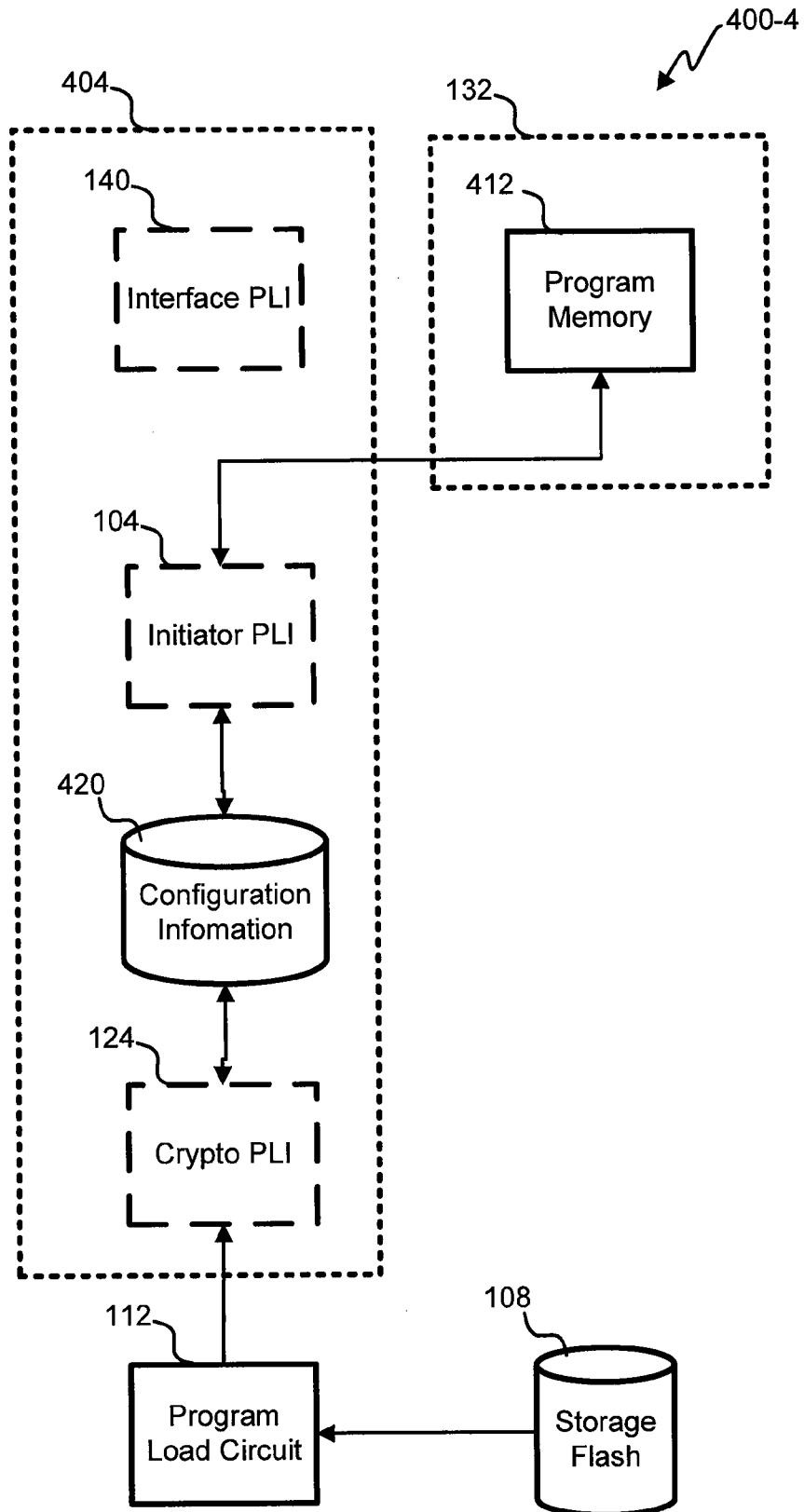


Fig. 4D

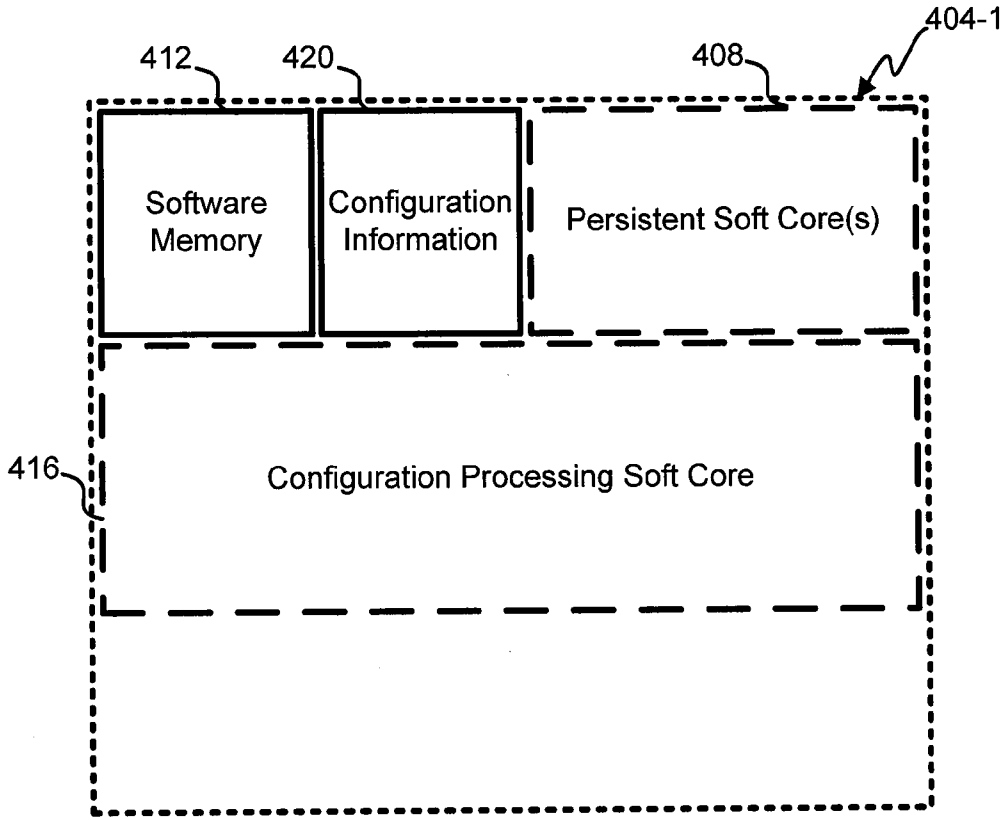


Fig. 5A

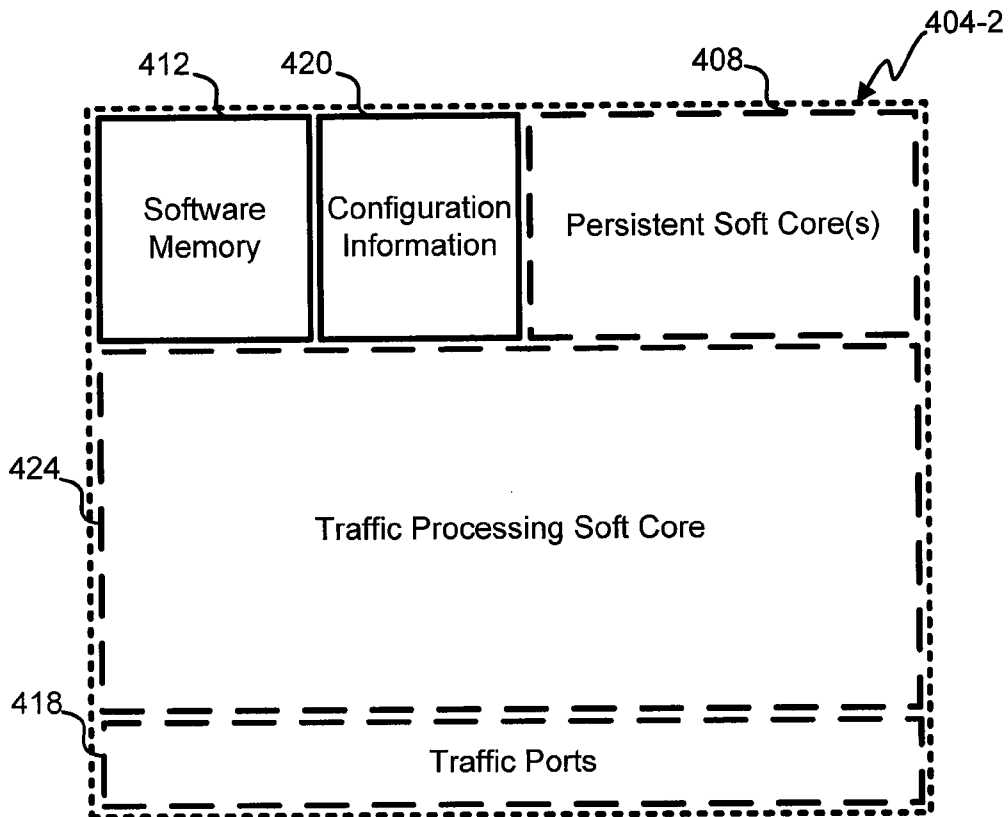


Fig. 5B

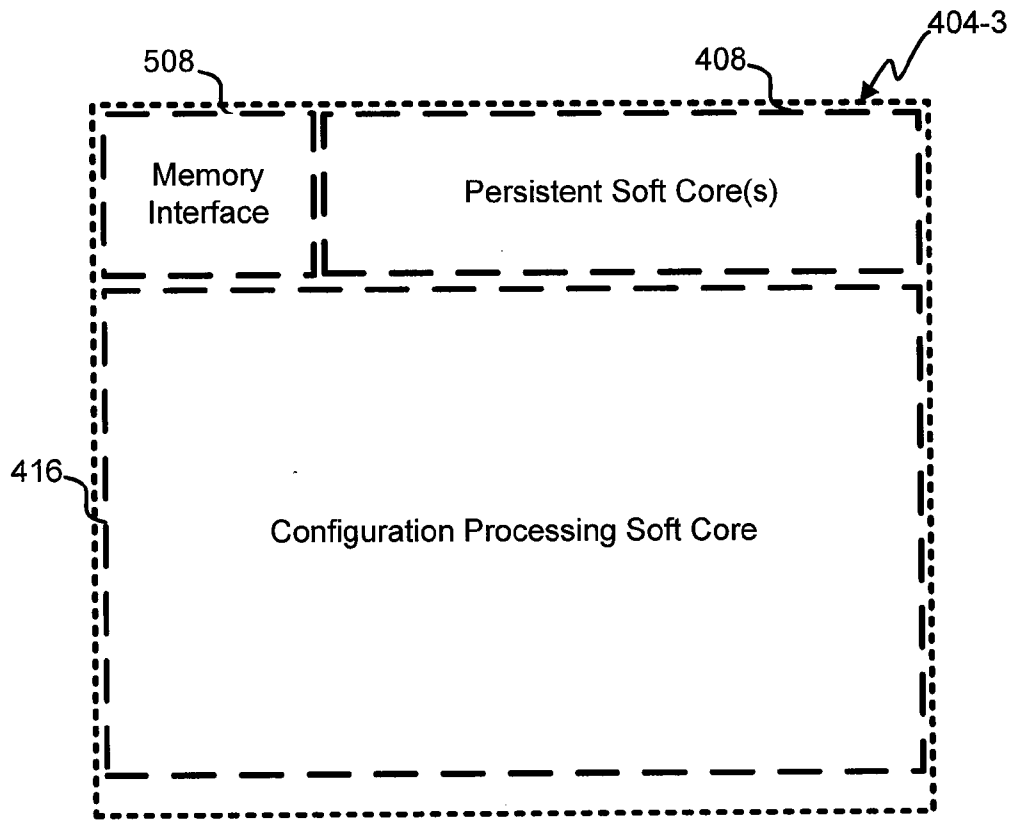


Fig. 5C

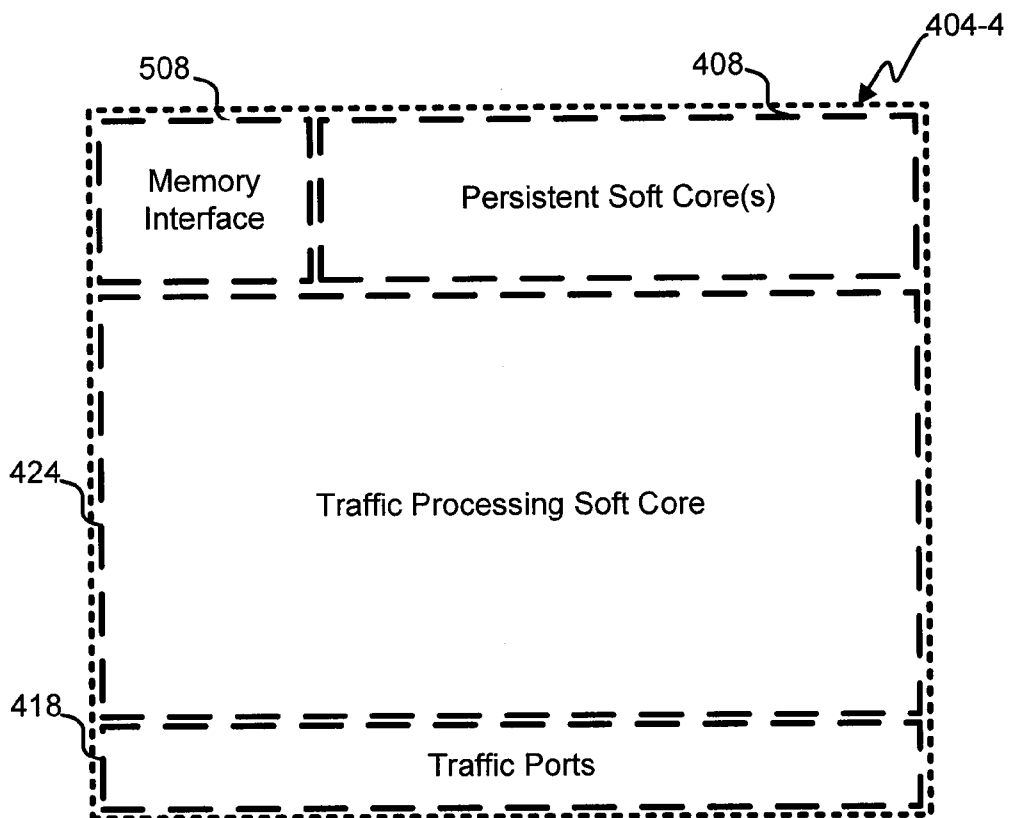


Fig. 5D

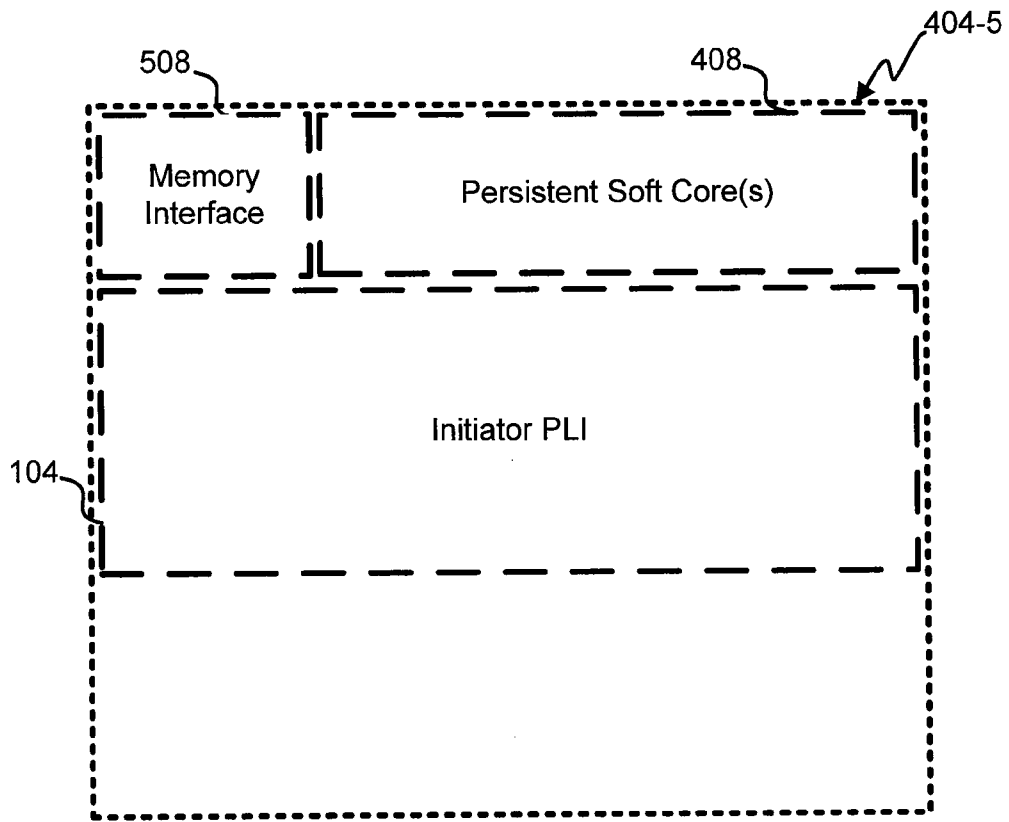


Fig. 5E

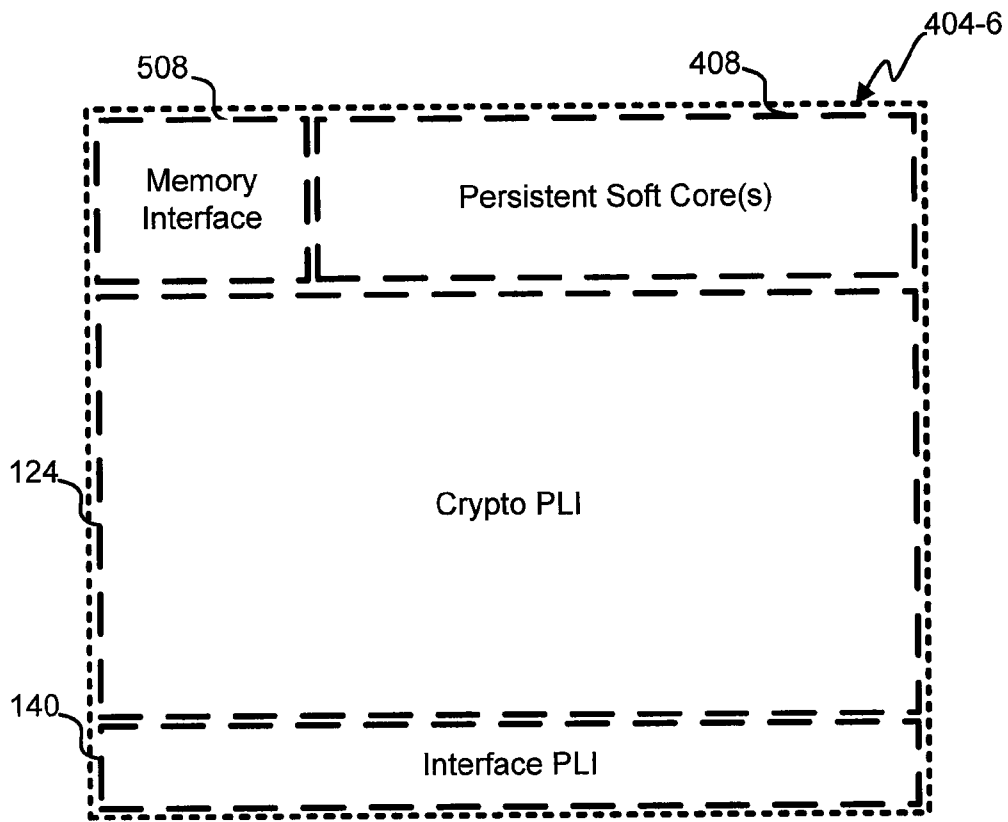


Fig. 5F

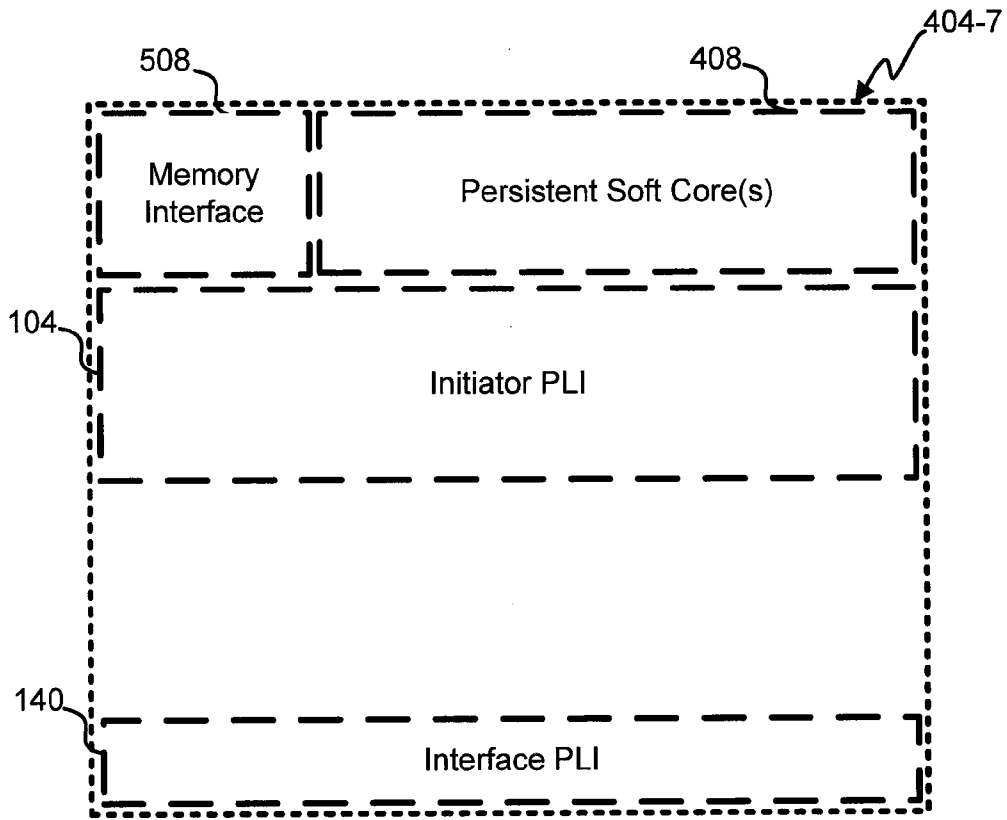


Fig. 5G

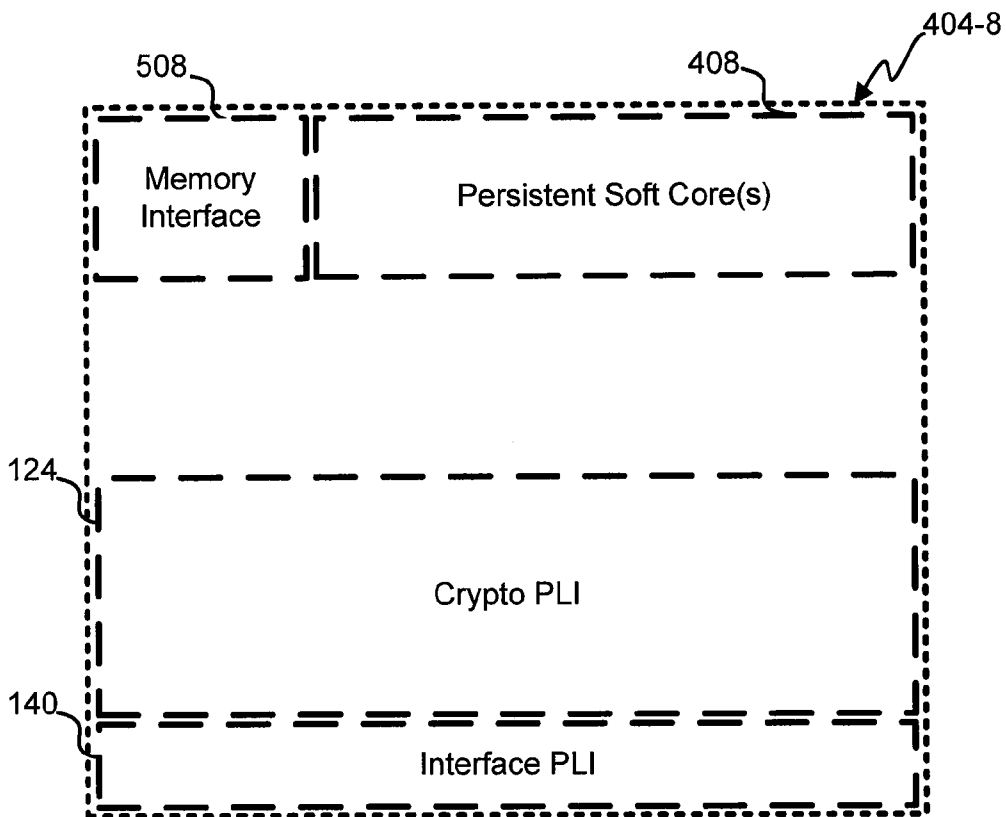


Fig. 5H

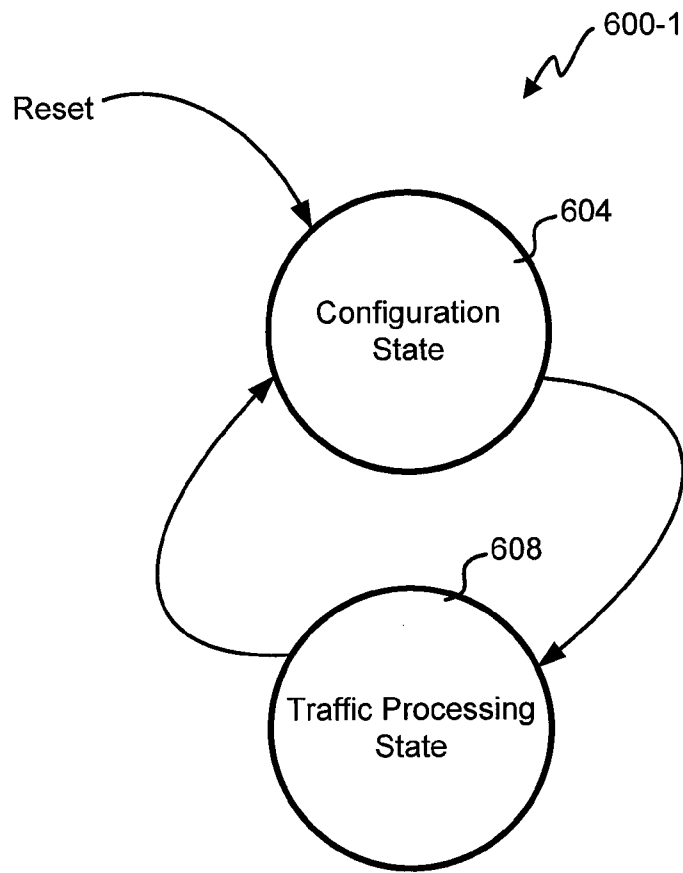


Fig. 6A

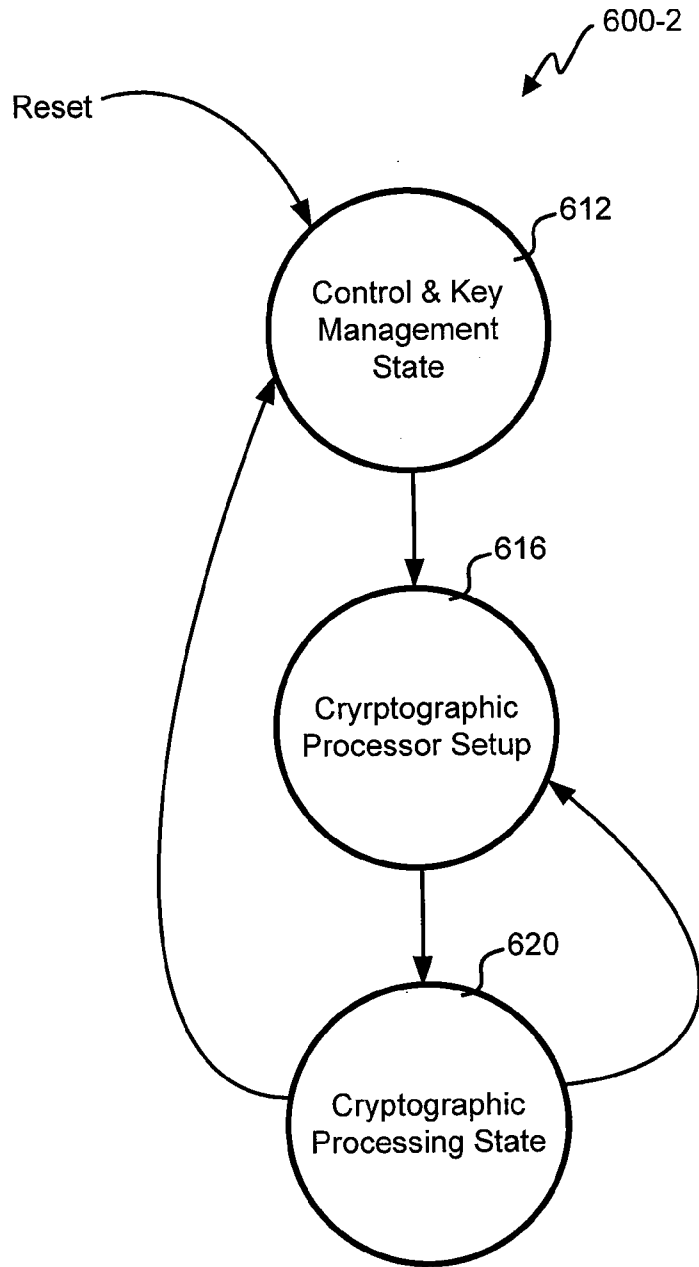


Fig. 6B

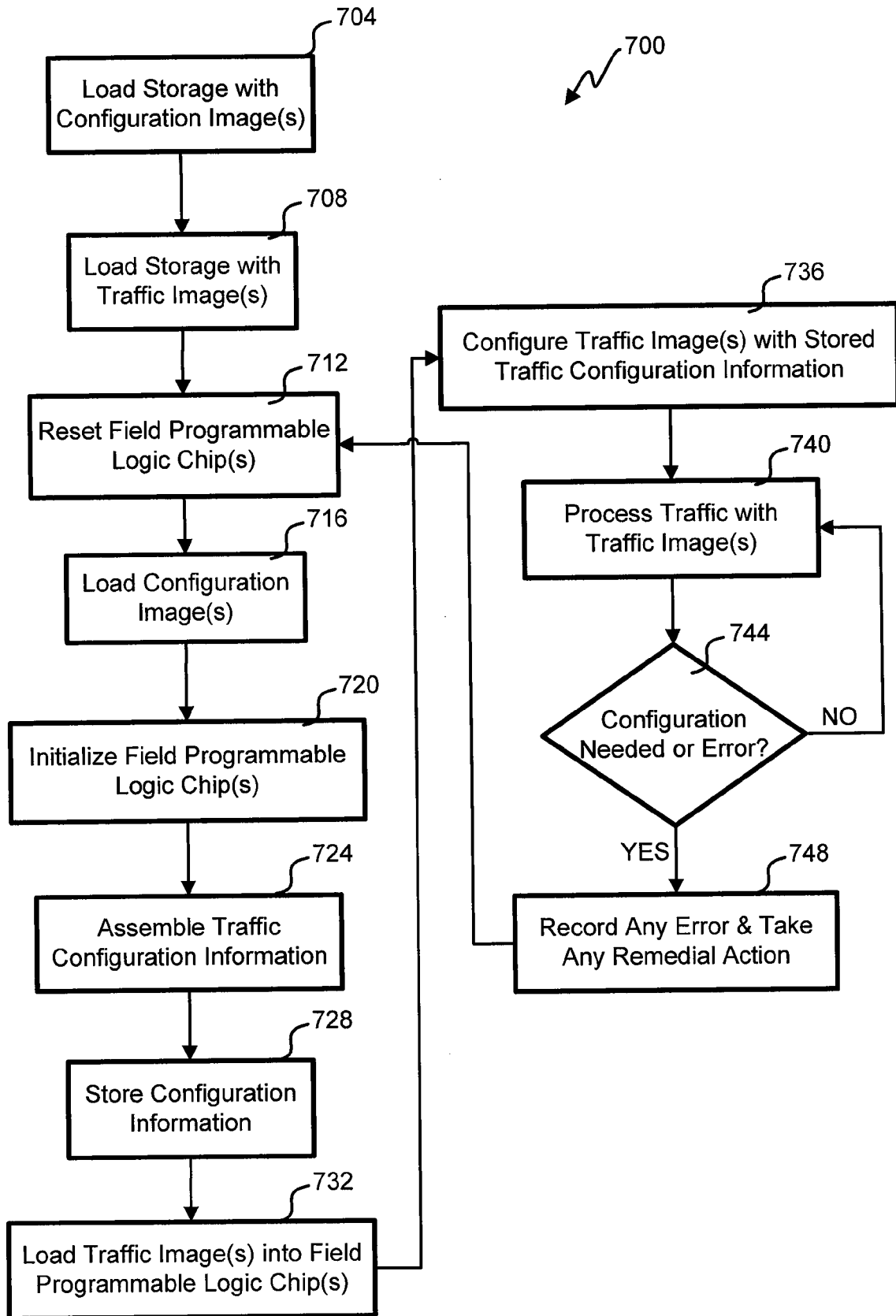


Fig. 7