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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
16 October 2003 (16.10.2003)

PCT

(10) International Publication Number
WO 03/085498 A2

(51) International Patent Classification[7]: G06F 1/00

(21) International Application Number: PCT/US03/08764

(22) International Filing Date: 20 March 2003 (20.03.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/112,124 29 March 2002 (29.03.2002) US

(71) Applicant: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US).

(72) Inventors: **SUTTON, James, II**; 20205 NW Paulina Drive, Portland, OR 97229 (US). **GRAWROCK, David**; 8285 Southwest 184th Avenue, Aloha, OR 97007 (US).

(74) Agents: **MALLIE, Michael, J.**; Blakely, Sokoloff, Tayor & Zafman, 7th Floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025 et al. (US).

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

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (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.

WO 03/085498 A2

(54) Title: SYSTEM AND METHOD FOR RESETTING A PLATFORM CONFIGURATION REGISTER

(57) Abstract: A method and apparatus for resetting and modifying special registers in a security token is described. In one embodiment, a register may be reset when a reset flag is true when a special transmission on a bus demonstrates the mutual locality of the associated processor and chipset. A modify flag may also be used to indicate whether the register contents may be modified. Modifications may also be dependent upon demonstration of mutual locality.

SYSTEM AND METHOD FOR RESETTING A PLATFORM CONFIGURATION REGISTER

FIELD

[0001] The present disclosure relates generally to microprocessor systems, and more specifically to microprocessor systems that may operate in a trusted or secured environment.

BACKGROUND

[0002] The increasing number of financial and personal transactions being performed on local or remote microcomputers has given impetus for the establishment of "trusted" or "secured" microprocessor environments. The problem these environments try to solve is that of loss of privacy, or data being corrupted or abused. Users do not want their private data made public. They also do not want their data altered or used in inappropriate transactions. Examples of these include unintentional release of medical records or electronic theft of funds from an on-line bank or other depository. Similarly, content providers seek to protect digital content (for example, music, other audio, video, or other types of data in general) from being copied without authorization.

[0003] One component of such a trusted microprocessor system may be a trusted platform module (TPM), as disclosed in the Trusted Computing Platform Alliance (TCPA) Main Specification, version 1.1a, 1 December 2001, issued by the Trusted Computing Platform Alliance (available at www.trustedpc.com at the time of filing of the present application). The TPM may include several special registers, called platform configuration registers (PCR). The PCRs may be used to store encrypted digests of files, including software programs, which may later be retrieved to further the process of authentication. As the contents of these PCRs are therefore significant to the secure or trusted nature of the system, the PCRs should not be reset by general-purpose operations of the system. Current design limits the PCRs to be reset only upon a general system reset (e.g. power-on reset).

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

[0005] **Figure 1** is a diagram of an exemplary trusted or secured software environment, according to one embodiment of the present invention.

[0006] **Figure 2** is a diagram of certain exemplary trusted or secured software modules and exemplary system environment, according to one embodiment of the present invention.

[0007] **Figure 3** is a detailed diagram of the security token of the system of Figure 2A, according to one embodiment of the present invention.

[0008] **Figure 4** is a schematic diagram of an exemplary microprocessor system adapted to support the secured software environment of Figure 1, according to one embodiment of the present invention.

[0009] **Figure 5** is a flowchart showing resetting of platform configuration registers, according to one embodiment of the present invention.

[0010] **Figure 6** is a flowchart showing resetting of platform configuration registers, according to another embodiment of the present invention.

DETAILED DESCRIPTION

[0011] The following description describes techniques for resetting or modifying special registers used in a trusted or secured environment in a microprocessor system. In the following description, numerous specific details such as logic implementations, software module allocation, encryption techniques, bus signaling techniques, and details of operation are set forth in order to provide a more thorough understanding of the present invention. It will be appreciated, however, by one skilled in the art that the invention may be practiced without such specific details. In

other instances, control structures, gate level circuits and full software instruction sequences have not been shown in detail in order not to obscure the invention. Those of ordinary skill in the art, with the included descriptions, will be able to implement appropriate functionality without undue experimentation. The invention is disclosed in the form of a microprocessor system. However, the invention may be practiced in other forms of processor such as a digital signal processor, a minicomputer, or a mainframe computer.

[0012] In one embodiment, a new kind of platform configuration register (PCR) is used in conjunction with a trusted platform module (TPM) within a security token. Each new PCR may have associated with it a pair of flags that may be used to indicate whether or not the PCR may be either reset or have its contents modified. The security token may have a bus interface that may receive special locality-confirming messages. These locality-confirming messages may indicate both that the security token is in direct physical connection with other system resources and also that trusted software desires to reset or modify PCR contents.

[0013] Referring now to Figure 1, a diagram of an exemplary trusted or secured software environment is shown, according to one embodiment of the present invention. In the Figure 1 embodiment, trusted and untrusted software may be loaded simultaneously and may execute simultaneously on a single computer system. A secure virtual machine monitor (SVMM) 350 selectively permits or prevents direct access to hardware resources 380 from one or more untrusted operating systems 340 and untrusted applications 310 through 330. In this context, "untrusted" does not necessarily mean that the operating system or applications are deliberately misbehaving, but that the size and variety of interacting code makes it impractical to reliably assert that the software is behaving as desired, and that there are no viruses or other foreign code interfering with its execution. In a typical embodiment, the untrusted code might consist of the normal operating system and applications found on today's personal computers.

[0014] SVMM 350 also selectively permits or prevents direct access to hardware resources 380 from one or more trusted or secure kernels 360 and one or more trusted applications 370. Such a trusted or secure kernel 360 and trusted applications 370 may be limited in size and functionality to aid in the ability to perform trust analysis upon it. The trusted application 370 may be any software code, program, routine, or set of routines which is executable in a secure environment. Thus, the trusted application 370 may be a variety of applications, or code sequences, or may be a relatively small application such as a Java applet.

[0015] Instructions or operations normally performed by operating system 340 or kernel 360 that could alter system resource protections or privileges may be trapped by SVMM 350, and selectively permitted, partially permitted, or rejected. As an example, in a typical embodiment, instructions that change the processor's page table that would normally be performed by operating system 340 or kernel 360 would instead be trapped by SVMM 350, which would ensure that the request was not attempting to change page privileges outside the domain of its virtual machine.

[0016] Referring now to Figure 2, a diagram of certain exemplary trusted or secured software modules and exemplary system environment 200 is shown, according to one embodiment of the present invention. In the Figure 2 embodiment, processor 202, processor 212, processor 222, and optional other processors (not shown) are shown. In other embodiments, the number of processors may differ, as may the boundary of various components and functional units.

[0017] Processors 202, 212, 222 may contain certain special circuits or logic elements to support secure or trusted operations. For example, processor 202 may contain secure enter (SENTER) logic 204 to support the execution of special SENTER instructions that may initiate trusted operations. SENTER is one example of a privileged instruction, defined as a machine instruction that usually can only be executed when in a special mode and usually available to the operating system or systems but not to other users. Privileged operations may occur upon the execution of

privileged instructions. Processor 202 may also contain bus message logic 206 to support special bus messages on system bus 230 in support of special SENTER operations. In alternate embodiments, memory control functions of chipset 240 may be allocated to circuits within the processors, and for multiple processors may be included on a single die. In these embodiments, special bus messages may also be sent on busses internal to the processors. The use of special bus messages may increase the security or trustability of the system for several reasons. Circuit elements such as processors 202, 212, and 222 or chipset 240 may only issue or respond to such messages if they contain the appropriate logic elements of embodiments of the present disclosure, or their equivalents. Therefore successful exchange of the special bus messages may help ensure proper system configuration. Special bus messages may also permit activities that should normally be prohibited, such as resetting a platform configuration register 286. The ability of potentially hostile untrusted code to spy on certain bus transactions may be curtailed by allowing special bus messages to be issued only in response to special security instructions.

[0018] Additionally, processor 202 may contain secure memory to support secure initialization operations. In one embodiment secure memory 208 may be an internal cache of processor 202, perhaps operating in a special mode.

[0019] A "chipset" may be defined as a group of circuits and logic that support memory and input/output (I/O) operations for a connected processor or processors. Individual elements of a chipset may be grouped together on a single chip, a pair of chips, or dispersed among multiple chips, including processors. In the Figure 2 embodiment, chipset 240 may include circuitry and logic to support memory and I/O operations to support processors 202, 212, and 222. In one embodiment, chipset 240 may interface with a number of memory pages 250 through 262 and a device-access page table 248 containing control information indicating whether non-processor devices may access the memory pages 250 through 262. Chipset 240 may include device-access logic 247 that may

permit or deny direct memory access (DMA) from I/O devices to selected portions of the memory pages 250 through 262. In some embodiment the device access logic 247 may contain all relevant information required to permit or deny such accesses. In other embodiments, the device access logic 247 may access such information held in the device access page table 248. The functions of chipset 240 may further be allocated among one or more physical devices in alternate embodiments. Chipset 240 may additionally include its own bus message logic 242 to support special bus messages on system bus 230 in support of special SENTER operations.

[0020] Chipset 240 may support standard I/O operations on I/O busses such as peripheral component interconnect (PCI), accelerated graphics port (AGP), universal serial bus (USB), low pin count (LPC) bus, or any other kind of I/O bus (not shown). An interface 290 may be used to connect chipset 240 with token 276, containing one or more platform configuration registers (PCR) 286, 294. In one embodiment, interface 290 may be the LPC bus (Low Pin Count (LPC) Interface Specification, Intel Corporation, rev. 1.0, 29 December 1997) modified with the addition of certain security enhancements. One example of such a security enhancement would be a locality confirming message, utilizing a previously-reserved message header and address information targeting a platform configuration register (PCR) 286 within token 276. In one embodiment, token 276 may contain special security features, and in one embodiment may include the trusted platform module (TPM) 281 disclosed in the Trusted Computing Platform Alliance (TCPA) Main Specification, version 1.1a, 1 December 2001, issued by the TCPA (available at the time of filing of the present application at www.trustedpc.com). The interface 290 may provide a data path for various two-way communications between chipset 240 and token 276. The security bus logic 266 of chipset 240 may include circuits to generate and transmit locality-confirming messages to token 276.

[0021] Two software components identified in system environment 200 are a Secure Virtual Machine Monitor (SVMM) 282 module and a Secure Initialization Authenticated Code (SINIT-AC) 280 module. The SVMM 282

module may be stored on a system disk or other mass storage, and moved or copied to other locations as necessary. In one embodiment, prior to beginning the secure launch process SVMM 282 may be moved or copied to one or more memory pages 250 through 262. Following the secure enter process, a virtual machine environment may be created in which the SVMM 282 may operate as the most privileged code within the system, and may be used to permit or deny direct access to certain system resources by the operating system or applications within the created virtual machines.

[0022] Some of the actions required by the secure enter process may be beyond the scope of simple hardware implementations, and may instead advantageously use a software module whose execution can be implicitly trusted. In one embodiment, these actions may be performed by Secure Initialization (SINIT) code. Such action may include checking critical portions of the system configuration to ensure that is supports the correct instantiation of the secure environment, configuring the device access logic 247 and device access page table 248, and calculate and register the SVMM 282 identity prior to transferring system control to it. Here "register" means placing a trust measurement of SVMM 282 into a register, for example into PCR 286 or into PCR 294. When this last action is taken, the trustworthiness of the SVMM 282 may be inspected by a potential system user.

[0023] The SINIT code may be produced by the manufacturer of the processors or of the chipsets. For this reason, the SINIT code may be trusted to aid in the secure launch of chipset 240. In order to distribute the SINIT code, in one embodiment a well-known cryptographic hash is made of the entire SINIT code, producing a value known as a "digest". One embodiment produces a 160-bit value for the digest. The digest may then be encrypted by a private key, held in one embodiment by the manufacturer of the processor, to form a digital signature. When the SINIT code is bundled with the corresponding digital signature, the combination may be referred to as SINIT authenticated code (SINIT-AC)

280. Copies of the SINIT-AC 280 may be later validated as discussed below.

[0024] Any logical processor may initiate the secure launch process, and may then be referred to as the initiating logical processor (ILP). In the present example processor 202 becomes the ILP, although any of the processors on system bus 230 could become the ILP. Neither memory-resident copy of SINIT-AC 280 nor memory-resident copy of SVMM 282 may be considered trustworthy at this time since, among other reasons, the other processors or the DMA devices may overwrite memory pages 250 – 262.

[0025] The ILP (processor 202) then executes a privileged instruction. This privileged instruction may in one embodiment be referred to as a secured enter (SENTER) instruction, and may be supported by SENTER logic 204. Execution of the SENTER instruction may cause the ILP (processor 202) to issue special bus messages on system bus 230. In other embodiments, other privileged instructions may be executed.

[0026] After issuing the special bus message, the ILP (processor 202) may check to see when and if the other processors, which may be referred to as responding logical processors (RLPs) are ready to enter secure operations. When the RLPs are ready, the ILP (processor 202) may move both a copy of SINIT-AC 280 and key 284 into secure memory for the purpose of authenticating and subsequently executing the SINIT code included in SINIT-AC 280. In one embodiment, this secure memory may be an internal cache of the ILP (processor 202), perhaps operating in a special mode. Key 284 represents the public key corresponding to the private key used to encrypt the digital signature included in the SINIT-AC 280 module, and is used to verify the digital signature and thereby authenticate the SINIT code. In one embodiment, key 284 may already be stored in the processor, perhaps as part of the SENTER logic 204. In another embodiment, key 284 may be stored in a read-only key register 244 of chipset 240, which is read by the ILP. In yet another embodiment, either the processor or the chipset's key register 244 may actually hold a cryptographic digest of key 284, where key 284 itself is included in the

SINIT-AC 280 module. In this last embodiment, the ILP reads the digest from key register 244, calculates an equivalent cryptographic hash over the key 284 embedded in SINIT-AC 280, and compares the two digests to ensure the supplied key 284 is indeed trusted.

[0027] A copy of SINIT-AC and a copy of a public key may then exist within secure memory. The ILP may now validate the copy of SINIT-AC by decrypting the digital signature included in the copy of the SINIT-AC using the copy of a public key. This decryption produces an original copy of a cryptographic hash's digest. If a newly-calculated digest matches this original digest then the copy of SINIT-AC and its included SINIT code may be considered trustable.

[0028] The ILP may now register the unique identity of the SINIT-AC module by writing the SINIT-AC module's cryptographic digest value to a platform configuration register 286 in the security token 276, as outlined below. The ILP's execution of its SENTER instruction may now terminate by transferring execution control to the trusted copy of the SINIT code held within the ILP's secure memory. The trusted SINIT code may then perform its system test and configuration actions and may register the memory-resident copy of SVMM, in accordance with the definition of "register" above.

[0029] Registration of the memory-resident copy of SVMM may be performed in several manners. In one embodiment, the SENTER instruction running on the ILP writes the calculated digest of SINIT-AC into PCR 286 within the security token 276. Subsequently, the trusted SINIT code may write the calculated digest of the memory-resident SVMM to the same PCR 286 or another PCR 294 within the security token 276. If the SVMM digest is written to the same PCR 286, the security token 276 hashes the original contents (SINIT digest) with the new value (SVMM digest) and writes the result back into the PCR 286. In embodiments where the first (initializing) write to PCR 286 is limited to the SENTER instruction, the resulting digest may be used as a root of trust for the system.

[0030] Once the trusted SINIT code has completed its execution, and has registered the identity of the SVMM in a PCR, the SINIT code may transfer ILP execution control to the SVMM. In a typical embodiment, the first SVMM instructions executed by the ILP may represent a self-initialization routine for the SVMM. The ILP may in one embodiment cause each of the RLPs to join in operations under the supervision of the now-executing copy of SVMM. From this point onwards, the overall system is operating in trusted mode as outlined in the discussion of Figure 1 above.

[0031] Referring now to Figure 3, a detailed diagram of the security token 276 of the system of Figure 2 is shown, according to one embodiment of the present invention. Token 276 may include a TPM 278 containing several prior-art PCRs 232, 234, 236, 238. These prior-art PCRs may be reset only at a general system reset event. The present TPM 278 may contain 16 or more prior-art general-purpose PCRs.

[0032] Additionally, in one embodiment token 276 may include PCRs 286, 294 of the present invention. Each PCR 286, 294 may be associated with flags indicating the behavior of the PCR under differing conditions. For example, PCR 286 has associated with it a reset flag 288 and a modify flag 292, and PCR 294 has associated with it a reset flag 296 and a modify flag 298. When reset flag 288 has the logical value true, PCR 286 may be reset under the command of a locality-confirming message arriving on the interface 290. When modify flag 292 has the logical value true, PCR 286 may have its contents modified under the command of a locality-confirming message arriving on the interface 290. In one embodiment, an encrypted digest of SINIT-AC 280 is intended to be written into PCR 286 and an encrypted digest of SVMM 282 is intended to be written into PCR 294. These digests may be exhibited upon request as part of an attestation process that may demonstrate to a local or remote user that system environment 200 is operating in a secure or trusted manner.

[0033] In another embodiment, digests of SINIT-AC 280 and of SVMM 282 may be written into a single PCR, such as PCR 286. In this

embodiment, first the digest of SINIT-AC is written into PCR 286. Then the contents of PCR 286 are cryptologically combined with a digest of SVMM 282, and the resulting combined digest written into PCR 286. In one embodiment, circuitry within token 276, configured to support PCR 286, may take the value of an incoming write request over interface 290, cryptologically combine this value with the current value within PCR 286, and then place the resulting new value within PCR 286. This process may be referred to as an "extend" operation. The value of the incoming write request may be said to be "extended onto" the current value within PCR 286. This extended value within PCR 286, representing a combined digest, may then be exhibited upon request as part of an attestation process that may demonstrate to a local or remote user that system environment 200 is operating in a secure or trusted manner.

[0034] Reset flags 288, 296 and modify flags 292, 298 may be set to logic true values at the time of manufacture of token 276. This will permit the resetting and content modification of PCR 286, 294 upon receipt of a locality-confirming bus message on interface 290. In other embodiments, reset flags 288, 296 and modify flags 292, 298 may have their values changed between true and false depending upon specific security actions taken under the control of TPM 278.

[0035] In the Figure 3 embodiment, exemplary additional PCRs 201, 203, and 205 of an embodiment are shown. In alternate embodiments fewer or additional PCRs in accordance with an embodiment of the present invention may be utilized. Additional PCRs such as PCRs 201, 203, and 205 may be utilized to store security-related information, such as digests of code, in other embodiments. Any additional PCRs, such as exemplary PCRs 201, 203, and 205, may have the associated reset flags 207, 209, and 211, respectively, and the associated modify flags 213, 215, and 217, respectively.

[0036] The PCRs 286, 294 may be reset or modified at times other than general system resets when the system environment 200 causes locality-confirming messages to be sent from chipset 240 to token 276. In one embodiment, the locality-confirming messages may be generated by

security bus logic 266 of chipset 240 in response to the bus message logic 242 receiving a SENTER special bus message over system bus 230. The SENTER special bus message in turn may result from the execution of a SENTER instruction in a processor. Such a SENTER instruction may be executed days, weeks, or months subsequent to any general system reset event. Details of the SENTER process, transferring system operation from insecure operations to secure operations, are described below in connection with Figures 3 through 7.

[0037] In the Figure 3 embodiment, token 276 is shown in an exemplary trusted or secure personal computer environment. Specifically, token 276 is shown connected with chipset 240 via an interface 290. In alternate embodiments, token 276 of an embodiment of the present invention may be used in other system environments. Examples of such environments may include a data server environment, a cellular or other form of mobile telephone, a mobile data port, an automotive system, a cable or satellite television system, or a personal digital assistant (PDA). These examples should not be construed as limiting, as there are many other system environments that may utilize a token of a present embodiment. An interface 290 or alternate equivalent interface may connect the other embodiment token to circuitry within one of the above exemplary system environments, wherein chipset 240 may be a portion of, for example, an integrated cellular telephone controller or a PDA controller. These embodiments are in accordance with the definition of "chipset" included in the discussion of Figure 2 above.

[0038] Referring now to Figure 4, one embodiment of a microprocessor system 400 adapted to support the secured software environment of Figure 1 is shown. CPU A 410, CPU B 414, CPU C 418, and CPU D 422 may be similar to the processors of Figure 1, but may be configured with additional microcode or logic circuitry to support the execution of special instructions. In one embodiment, this additional microcode or logic circuitry may be the SENTER logic 204 of Figure 2. These special instructions may support the issuance of special bus messages on system bus 420. In one embodiment, the issuance of special bus messages may

be supported by circuitry such as the bus message logic 206 of Figure 2. Similarly chipset 430 may be similar to chipset 130 but may support the above-mentioned special cycles on system bus 420. Additionally chipset 430 may possess additional page protection circuitry for protecting page tables within physical memory 434. In one embodiment, the page protection circuitry may be the device access logic 247 of Figure 2.

[0039] In one embodiment, chipset 430 interfaces an additional data bus called a modified low pin count (LPC) bus 450 through a LPC bus logic (LPC-BL) 452. Modified LPC bus 450 may be used to connect chipset 430 with a security token 454. Token 454 may in one embodiment include the trusted platform module (TPM) 460 envisioned by the TCPA. In one embodiment, token 454 may include SINIT PCR 470 and SVMM PCR 480. Each PCR 470, 480 may be associated with flags indicating the behavior of the PCR under differing conditions. For example, SINIT PCR 470 has associated with it a reset flag 472 and a modify flag 474, and SVMM PCR 480 has associated with it a reset flag 482 and a modify flag 484. When reset flag 472 has the logical value true, SINIT PCR 470 may be reset under the command of a locality-confirming message arriving on the modified LPC bus 450. When modify flag 474 has the logical value true, SINIT PCR 470 may have its contents modified under the command of a locality-confirming message arriving on the modified LPC bus 450. In one embodiment, an encrypted digest of SINIT-AC is intended to be written into SINIT PCR 470 and an encrypted digest of SVMM is intended to be written into SVMM PCR 480. These digests may be exhibited upon request as part of an attestation process that may demonstrate to a local or remote user that microprocessor system 400 is operating in a secure or trusted manner.

[0040] In an alternate embodiment, digests of SINIT-AC and of SVMM may be written into a single PCR, such as SINIT PCR 470. In this embodiment, first the digest of SINIT-AC is written into SINIT PCR 470. Then the contents of SINIT PCR 470 are cryptologically combined with a digest of SVMM, and the resulting combined digest written into SINIT PCR 470. In one embodiment, circuitry within token 454, configured to

support SINIT PCR 470, may take the value of an incoming write request over modified PC bus 450, cryptologically combine this value with the current value within SINIT PCR 470, and then place the resulting new value within SINIT PCR 470. This combined digest may then be exhibited upon request as part of an attestation process that may demonstrate to a local or remote user that microprocessor system 400 is operating in a secure or trusted manner.

[0041] Reset flags 472, 482 and modify flags 474, 484 may be set to logic true values at the time of manufacture of token 454. This will permit the resetting and content modification of SINIT PCR 470 and SVMM PCR 480 upon receipt of a locality-confirming bus message on modified LPC bus 450. In other embodiments, reset flags 472, 482 and modify flags 474, 484 may have their values changed between true and false depending upon specific security actions taken within microprocessor system 400, such as a SENTER or other privileged instruction execution.

[0042] In order to use a SVMM 350 in a microprocessor system such as microprocessor system 400, several actions may need to be taken. One action may be that the system configuration should be tested for trustworthiness, both in terms of hardware configuration, the SVMM 350 itself, and any software that loads and starts the SVMM 350. Another action may be to register the SVMM 350 identity and transfer system control to it. When this last action is taken the trustworthiness of the SVMM 350 may be inspected by a potential system user.

[0043] In one embodiment, a copy of SINIT-AC may be stored in the fixed media 444 for use when the microprocessor system 400 needs to enter secure operations. When a logical processor, for example physical processor CPU 414, desires to initialize system wide secure operations using the SVMM 350, it may load a copy of SINIT-AC into memory 434 to form a memory-resident copy of SINIT-AC. Any logical processor starting up secure operations in this manner may be referred to as an initiating logical processor (ILP). In the present example CPU B 414 becomes the ILP. CPU B 414 as ILP may then load a copy of SVMM into memory to

form a memory-resident copy of SVMM. This copy of SVMM may be on non-contiguous pages of memory, but the chipset 430 will maintain a page table for it. Neither SINIT-AC nor SVMM may be considered trustworthy at this time since, among other reasons, the other processors or the DMA devices may overwrite memory 434.

[0044] The ILP then may execute a special or privileged instruction implemented in microcode or other processor logic, such as SENTER. Execution of the SENTER instruction may cause the ILP to issue special bus messages on system bus 420, and then wait considerable time intervals for subsequent system actions. In other embodiments, other privileged instructions may be executed.

[0045] After issuing the special bus message, the ILP (CPU B 414) checks to see when and if all the RLPs have properly responded. When the RLPs are ready to enter secure operations, the ILP may move both the memory-resident copy of SINIT-AC and a cryptographic public key of chipset 430 into secure memory. In one embodiment, this secure memory may be the internal cache of the ILP, CPU B 414. Copy of SINIT-AC and copy of public key may then exist within internal cache. The ILP may now validate the cache-resident copy of SINIT-AC by decrypting the digital signature of this copy of the SINIT-AC using the cache-resident copy of public key. This decryption produces an original copy of a cryptographic hash's "digest". If a newly-calculated "digest" matches this original "digest" then the cache-resident copy of SINIT-AC may be considered trustable.

[0046] At this point during the execution of the SENTER instruction, a pair of locality-confirming messages may be generated by LPC-BL 452. This allows the resetting of SINIT PCR 470 and the writing of the SINIT digest into SINIT PCR 470.

[0047] The ILP may now issue another special bus message, via system bus 420 signaling the waiting RLP's (CPU A 410, CPU C 418, CPU D 422) and chipset 430 that secured operations are going to be initiated. The trusted cache-resident copy of SINIT-AC may then test and validate the copy of SVMM residing in physical memory 434. In one embodiment,

SINIT-AC may read the page table entries for the copy of SVMM located in memory 434. SINIT-AC may then command chipset 430 to prevent DMA access from I/O devices to those page table entries. Validation and subsequent registration of the memory-resident copy of SVMM may be performed in several manners. Again during this portion of the execution of the SENTER instruction, another pair of locality-confirming messages may be generated by LPC-BL 452. This allows the resetting of SVMM PCR 480 and the writing of the SVMM digest into SVMM PCR 480.

[0048] At this point the memory-resident copy of SVMM may begin execution, at the ending of execution of the SENTER instruction. The ILP may in one embodiment cause each of the RLPs to join in operations under the supervision of the now-executing memory-resident copy of SVMM. From this point onwards the overall system is operating in trusted mode as outlined in the discussion of Figure 1 above.

[0049] Referring now to Figure 5, a flowchart showing resetting of platform configuration registers is shown, according to one embodiment of the present invention. The method shown in Figure 5 may be used for the secure or trusted computer environments of Figure 1 and Figure 3. The method may also be used in other environments, such as a cellular or other form of mobile telephone, automobile or other security systems, or in a digital or satellite television or other media exchange system. In these latter embodiments, circuits within the named equipment may serve as the equivalent of the chipset 240 or chipset 430 for the purpose of sending locality-confirming messages to a security token.

[0050] The Figure 5 embodiment starts in block 520, where a first locality-confirming message is sent over an interface, addressed to a first PCR. Data within the first locality-confirming message indicates a reset is to be performed, and in block 530 the reset flag of the first PCR is checked to see if resets may be allowed. If so, then the first PCR is reset. Then in block 540 a second locality-confirming message is sent, addressed to the first PCR. Data within the second locality-confirming message indicates a digest (or other data) write is to be performed, and in block 550 the modify flag of the first PCR is checked to see if

modifications to the value contained within the first PCR may be allowed. If so, then the first PCR is written with the digest contained within the second locality-confirming message.

[0051] In block 560, a third locality-confirming message may then be sent that is addressed to a second PCR. Data within the third locality-confirming message indicates a reset is to be performed, and in block 570 the reset flag of the second PCR is checked to see if resets may be allowed. If so, then the second PCR is reset. Then in block 580 a fourth locality-confirming message is sent, addressed to the second PCR. Data within the fourth locality-confirming message indicates a digest (or other data) write is to be performed, and in block 590 the modify flag of the second PCR is checked to see if modifications to the value contained within the second PCR may be allowed. If so, then the second PCR is written with the digest contained within the fourth locality-confirming message.

[0052] Referring now to Figure 6, a flowchart showing resetting of platform configuration registers is shown, according to another embodiment of the present invention. The method of Figure 6 may be used in various environments as listed above in connection with the method of Figure 5. The Figure 6 embodiment starts in block 620, where a first locality-confirming message is sent over an interface, addressed to a first PCR. Data within the first locality-confirming message indicates a reset is to be performed, and in block 630 the reset flag of the first PCR is checked to see if resets may be allowed. If so, then the first PCR is reset. Then in block 640 a second locality-confirming message is sent, addressed to the first PCR. Data within the second locality-confirming message indicates a digest (or other data) write is to be performed, and in block 650 the modify flag of the first PCR is checked to see if modifications to the value contained within the first PCR may be allowed. If so, then the first PCR is written with the digest contained within the second locality-confirming message.

[0053] Then in block 660 a third locality-confirming message is sent, addressed again to the first PCR. Data within the third locality-confirming message indicates a digest (or other data) write is to be

performed, and in block 670 the modify flag of the first PCR is checked to see if modifications to the value contained within the first PCR may still be allowed. If so, then in block 680 the token cryptographically hashes the digest contained within the third locality-confirming message onto the existing contents of the first PCR. Then in block 690 the resulting combined digest may be written into the first PCR. This process may be the extend process as described above in connection with Figure 2.

[0054] In the foregoing specification, the invention has been described with reference to specific exemplary embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention as set forth in the appended claims. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

CLAIMS

What is claimed is:

1. A system, comprising:
a circuit including a first register; and
a first bus coupled to said circuit to convey a locality confirming message.

2. The system of claim 1, further comprising a first flag coupled to said first register.

3. The system of claim 2, wherein said first flag indicates permission to reset said first register.

4. The system of claim 3, wherein said first register is capable of being reset during execution of a privileged instruction by a processor.

5. The system of claim 3, further comprising a second flag coupled to said first register to indicate permission to modify a content of said first register.

6. The system of claim 1, further comprising a first flag coupled to said first register to indicate permission to modify a content of said first register.

7. The system of claim 6, wherein said content of said first register is capable of being modified during execution of a privileged instruction by a processor.

8. The system of claim 1, wherein said first register is capable of containing a first digest of a first code.

9. The system of claim 8, wherein said first register is capable of containing a second digest of a second code extended onto said first digest.

10. The system of claim 8, further comprising a second register to contain a second digest of a second code.

11. The system of claim 1, further comprising a chipset to transmit said locality confirming message across said first bus.

12. The system of claim 11, wherein said chipset transmits said locality confirming message responsively to a special bus message.

13. The system of claim 12, wherein said special bus message is conveyed on a second bus.

14. The system of claim 13, further comprising a processor to transmit said special bus message on said second bus.

15. The system of claim 14, wherein said processor transmits said special bus message during an execution of a privileged instruction.

16. An apparatus, comprising:

a first register to store a first digest; and

a first flag coupled to said first register to indicate whether to permit said first register to reset responsive to a locality confirming message.

17. The apparatus of claim 16, wherein said first flag may be set at manufacture.

18. The apparatus of claim 16, further comprising a second flag coupled to said first register to indicate whether to permit contents of said first register to change responsive to said locality confirming message.

19. The apparatus of claim 16, further comprising a second register to store a second digest.

20. The apparatus of claim 19, further comprising a third flag coupled to said second register to indicate whether to permit said second register to reset responsive to said locality confirming message.

21. The apparatus of claim 20, further comprising a fourth flag coupled to said second register to indicate whether to permit contents of said second register to change responsive to said locality confirming message.

22. The apparatus of claim 16, further comprising a circuitry supporting said first register responsive to a write instruction to extend said first digest with a second digest.

23. A chipset, comprising:

bus message logic to receive a special bus message from a processor; and

security bus logic to send a locality confirming message responsive to said special bus message.

24. The chipset of claim 23, when said special bus message is issued during the period of execution of a secured enter instruction.

25. The chipset of claim 23, wherein said locality confirming message includes instructions to reset a register.

26. The chipset of claim 23, wherein locality confirming message includes instructions to modify the contents of a register.

27. A method, comprising:

transmitting a first locality confirming message on a first bus; and

determining whether to reset a first register responsively to said first locality confirming message.

28. The method of claim 27, further comprising determining whether to reset said first register responsive to a first flag.

29. The method of claim 28, further comprising transmitting a second locality confirming message on said first bus; and further comprising determining whether to modify a first content of said first register responsively to said second locality confirming message.

30. The method of claim 29, further comprising determining whether to modify said first content of said first register responsive to a second flag.

31. The method of claim 30, further comprising transmitting a third locality confirming message on said first bus; and further comprising determining whether to reset a second register responsively to said third locality confirming message.

32. The method of claim 31, further comprising transmitting a fourth locality confirming message on said first bus; and further comprising determining whether to modify a second content of said second register responsively to said fourth locality confirming message.

33. The method of claim 30, further comprising transmitting a third locality confirming message on said first bus; and further

comprising determining whether to modify a second content of said first register responsively to said third locality confirming message.

34. The method of claim 31, further comprising modifying said second content of said first register using a cryptological combination with a digest.

35. An apparatus, comprising:
a first register to store a first digest; and
a first flag coupled to said first register to indicate whether to permit contents of said first register to be modified responsive to a locality confirming message.

36. The apparatus of claim 35, wherein said first flag may be set at manufacture.

37. The apparatus of claim 35, further comprising a second flag coupled to said first register to indicate whether to permit said first register to reset responsive to said locality confirming message

38. A method, comprising:
transmitting a first locality confirming message on a first bus; and
determining whether to modify contents of a first register responsively to said first locality confirming message.

39. The method of claim 38, further comprising determining whether to modify contents of said first register responsive to a first flag.

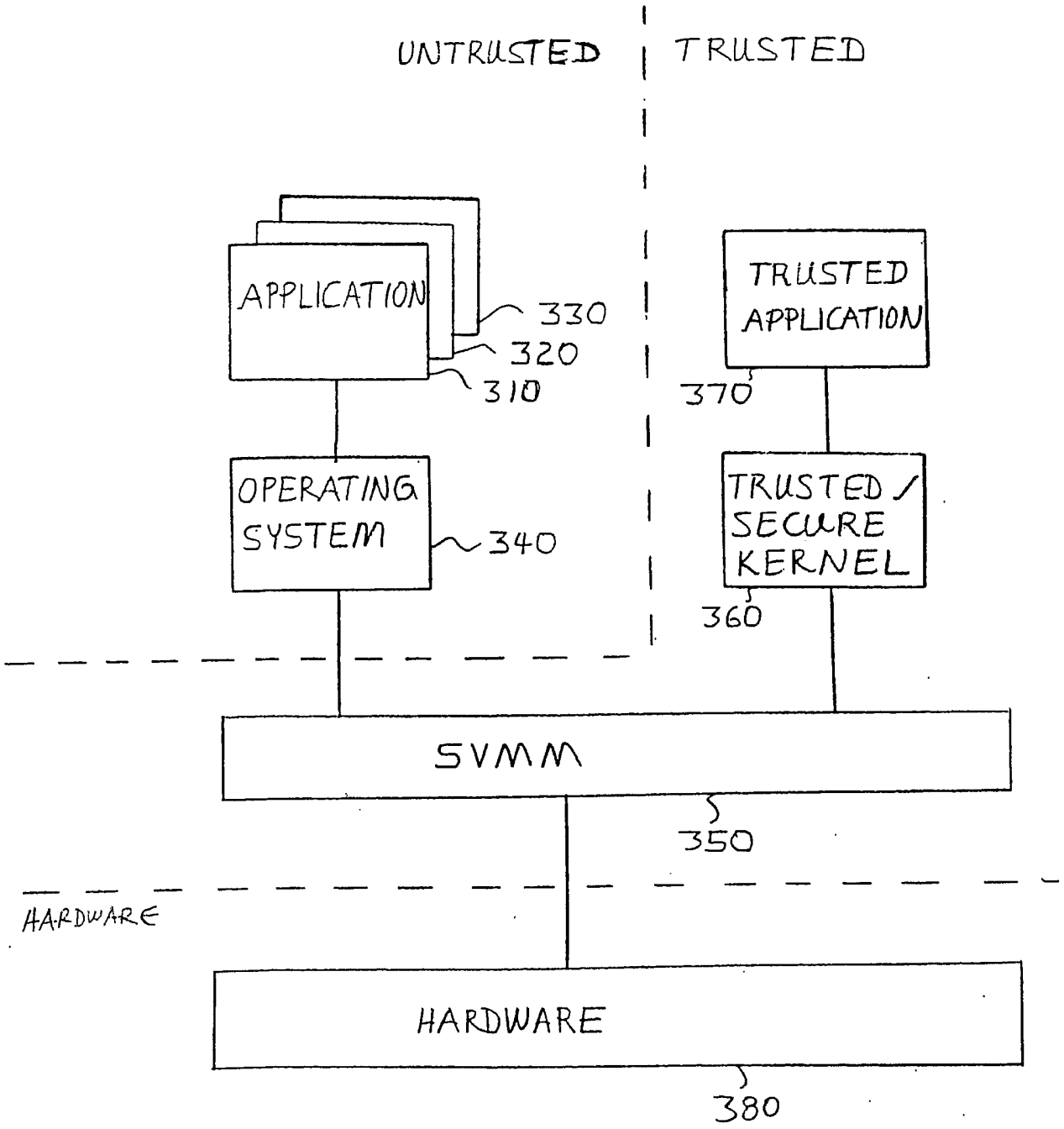
UNTRUSTED
TRUSTED
APPLICATION
330
320
310
TRUSTED APPLICATION
370
OPERATING SYSTEM
340
TRUSTED / SECURE KERNEL
360
SVMM
350
HARDWARE
HARDWARE
380

FIGURE 1

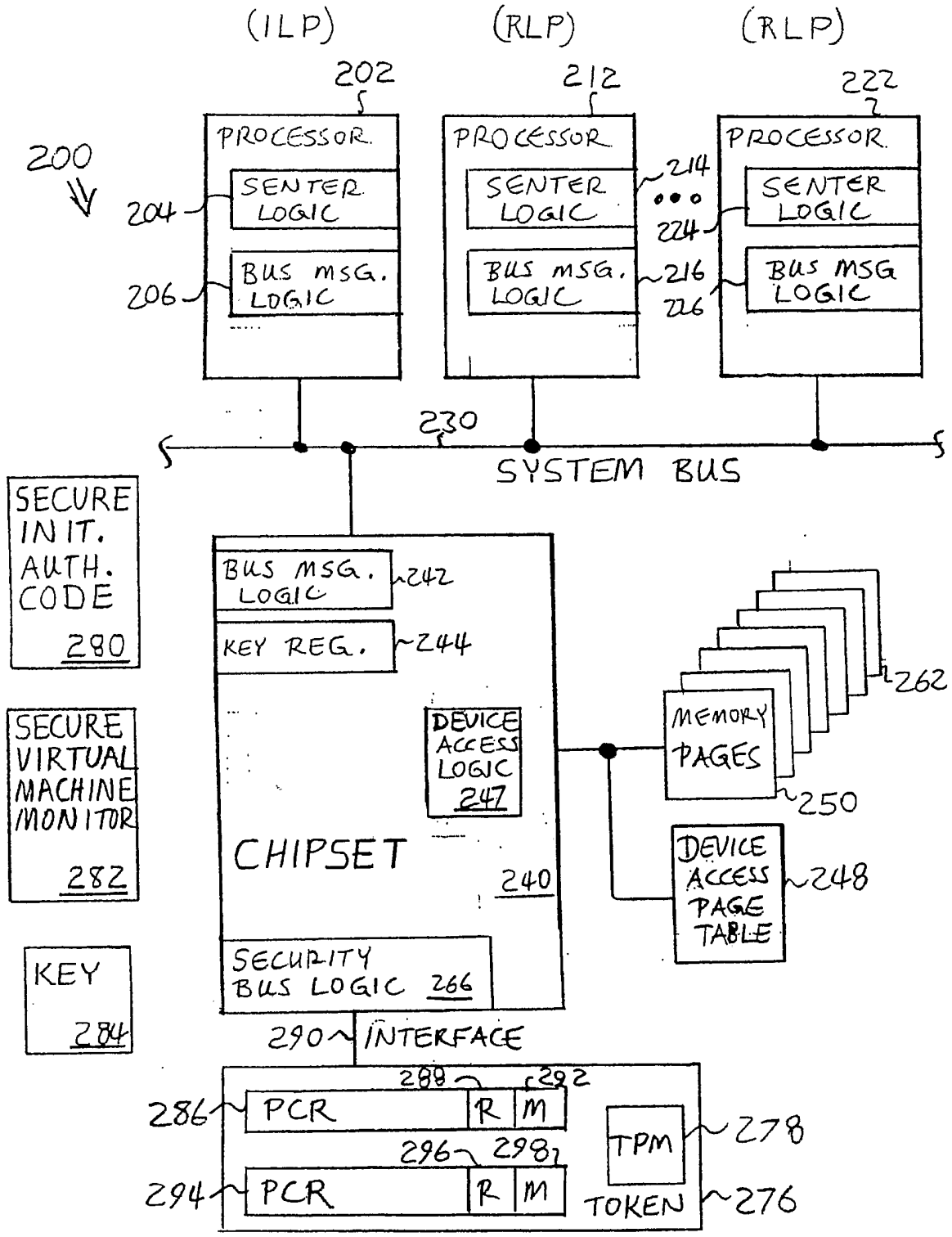
(ILP)
(RLP)
(RLP)
202
212
222
200
PROCESSOR
PROCESSOR
PROCESSOR
204
SENTER LOGIC
SENTER LOGIC
214
SENTER LOGIC
224
206
BUS MSG. LOGIC
BUS MSG. LOGIC
216
BUS MSG LOGIC
226
230
SYSTEM BUS
SECURE INIT. AUTH. CODE 280
BUS MSG. LOGIC 242
KEY REG. 244
262
MEMORY PAGES
250
DEVICE ACCESS LOGIC 247
SECURE VIRTUAL MACHINE MONITOR 282
CHIPSET
240
DEVICE ACCESS PAGE TABLE
248
KEY 284
SECURITY BUS LOGIC 266
290
INTERFACE
288
292
286
PCR
R
M
296
298
294
PCR
R
M
TPM
278
TOKEN
276

FIGURE 2

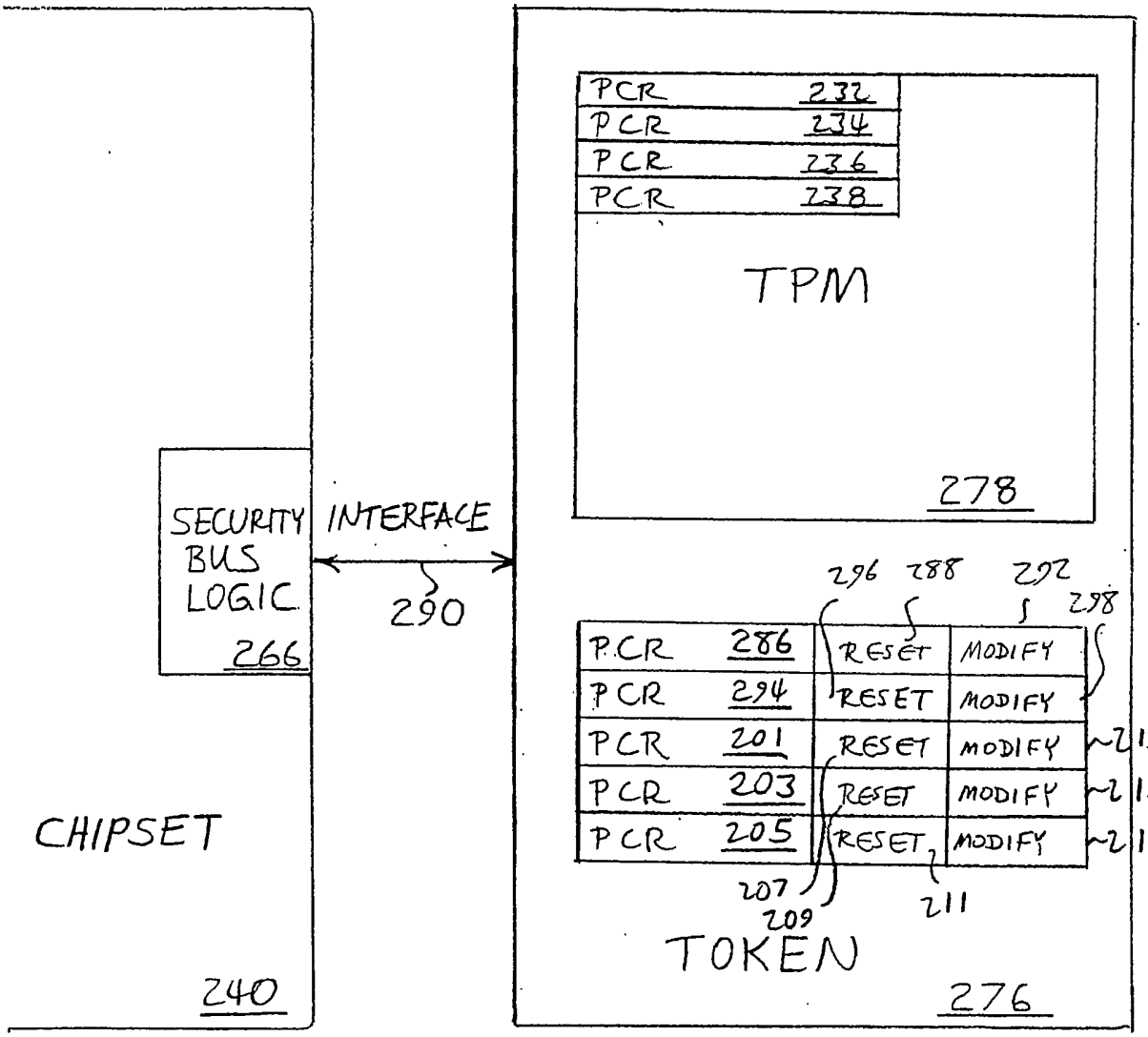
PCR 232
PCR 234
PCR 236
PCR 238
TPM
278
SECURITY BUS LOGIC
266
INTERFACE
290
CHIPSET
240
296
288
292
298
PCR 286
PCR 294
PCR 201
PCR 203
PCR 205
RESET
MODIFY
213
215
217
207
209
211
TOKEN
276

FIGURE 3

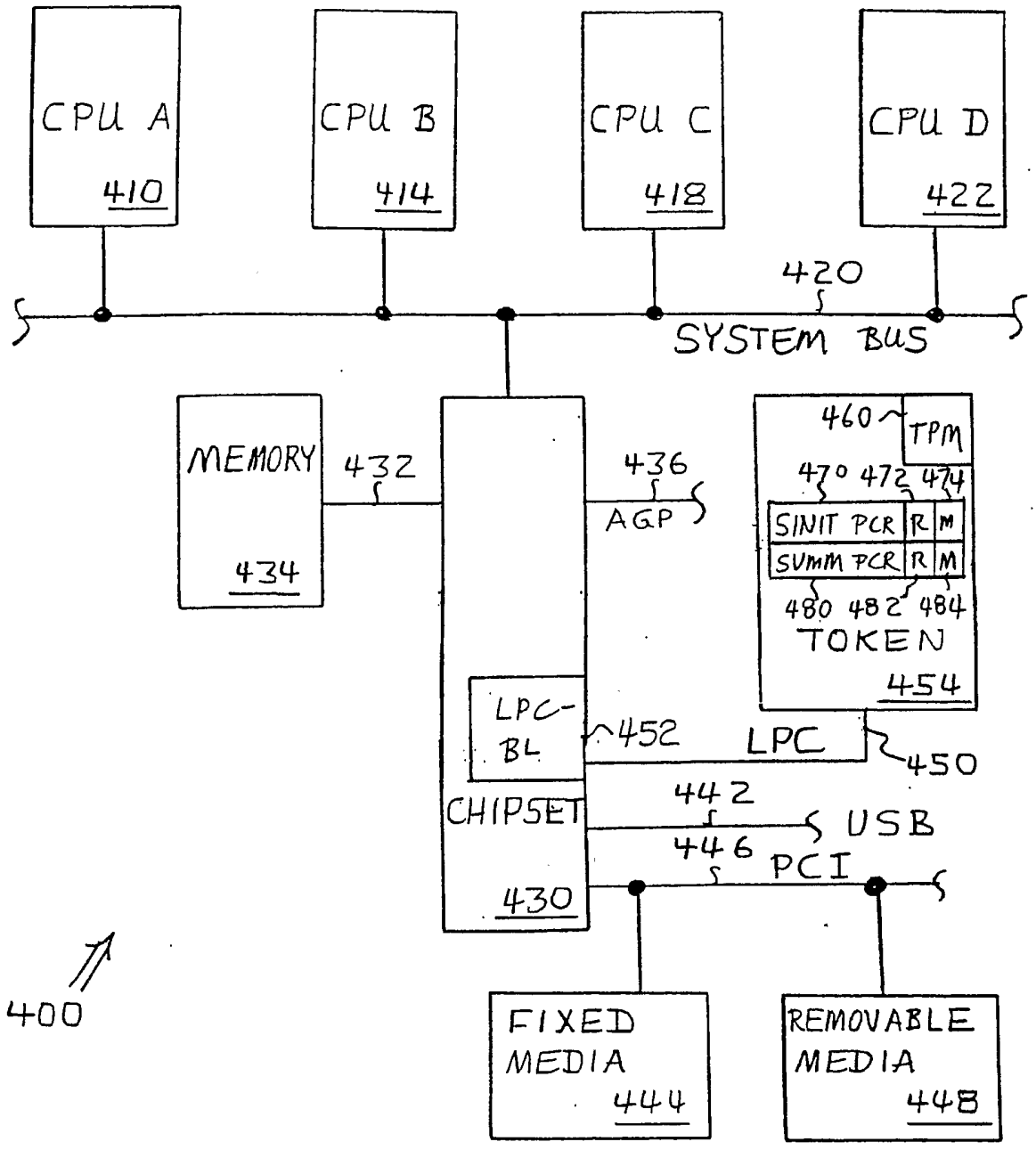
CPU A
410
CPU B
414
CPU C
418
CPU D
422
420
SYSTEM BUS
MEMORY
434
432
436
AGP
460
TPM
470
472
474
SINIT PCR
R
M
SVMM PCR
R
M
480
482
484
TOKEN
454
LPC-
BL
452
LPC
450
CHIPSET
430
442
USB
446
PCI
400
FIXED
MEDIA
444
REMOVABLE
MEDIA
448

FIGURE 4

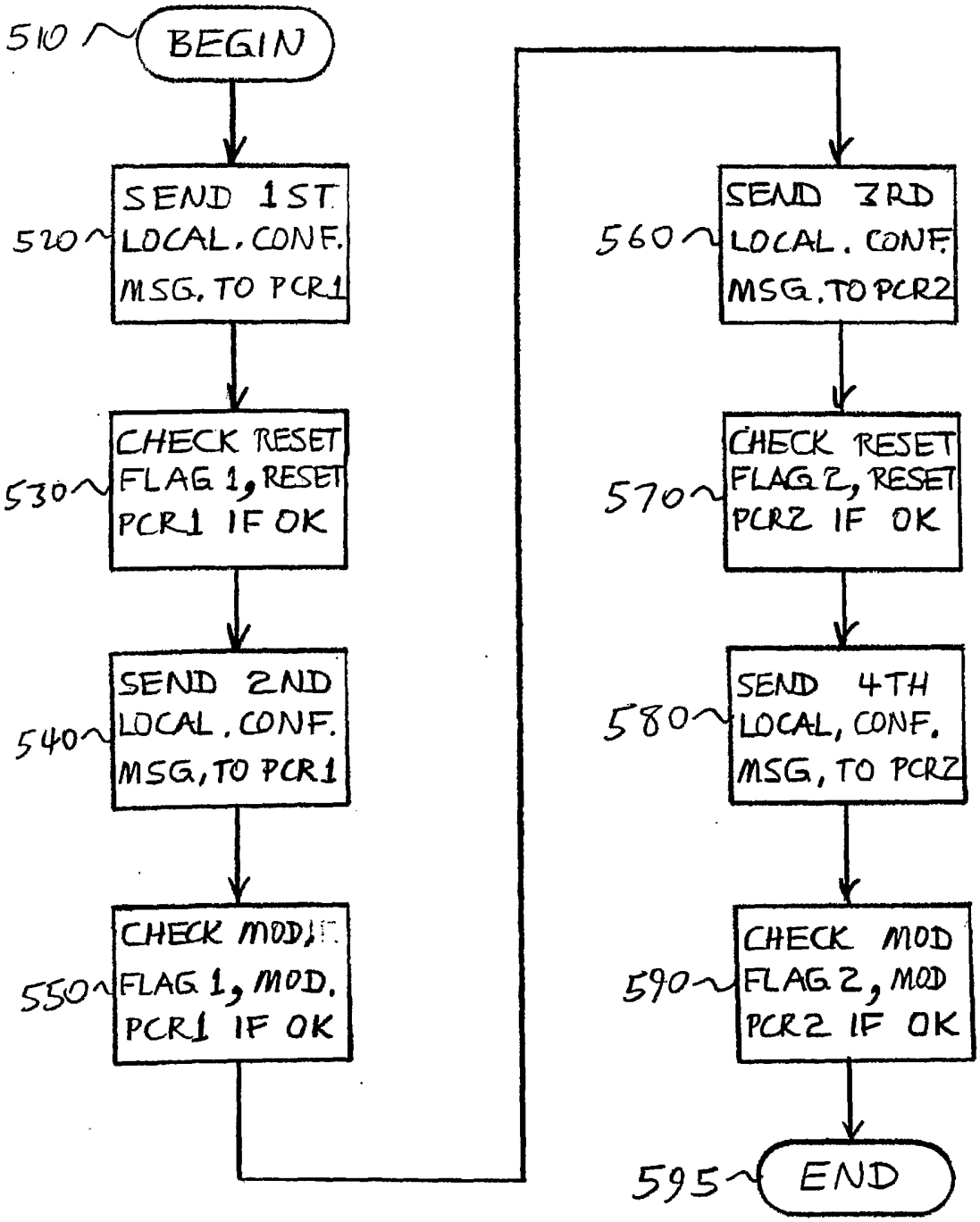
510
BEGIN
520
SEND 1ST LOCAL. CONF. MSG. TO PCR1
530
CHECK RESET FLAG 1, RESET PCR1 IF OK
540
SEND 2ND LOCAL. CONF. MSG, TO PCR1
550
CHECK MOD FLAG 1, MOD. PCR1 IF OK
560
SEND 3RD LOCAL. CONF. MSG. TO PCR2
570
CHECK RESET FLAG 2, RESET PCR2 IF OK
580
SEND 4TH LOCAL, CONF. MSG, TO PCR2
590
CHECK MOD FLAG 2, MOD PCR2 IF OK
595
END

FIGURE 5

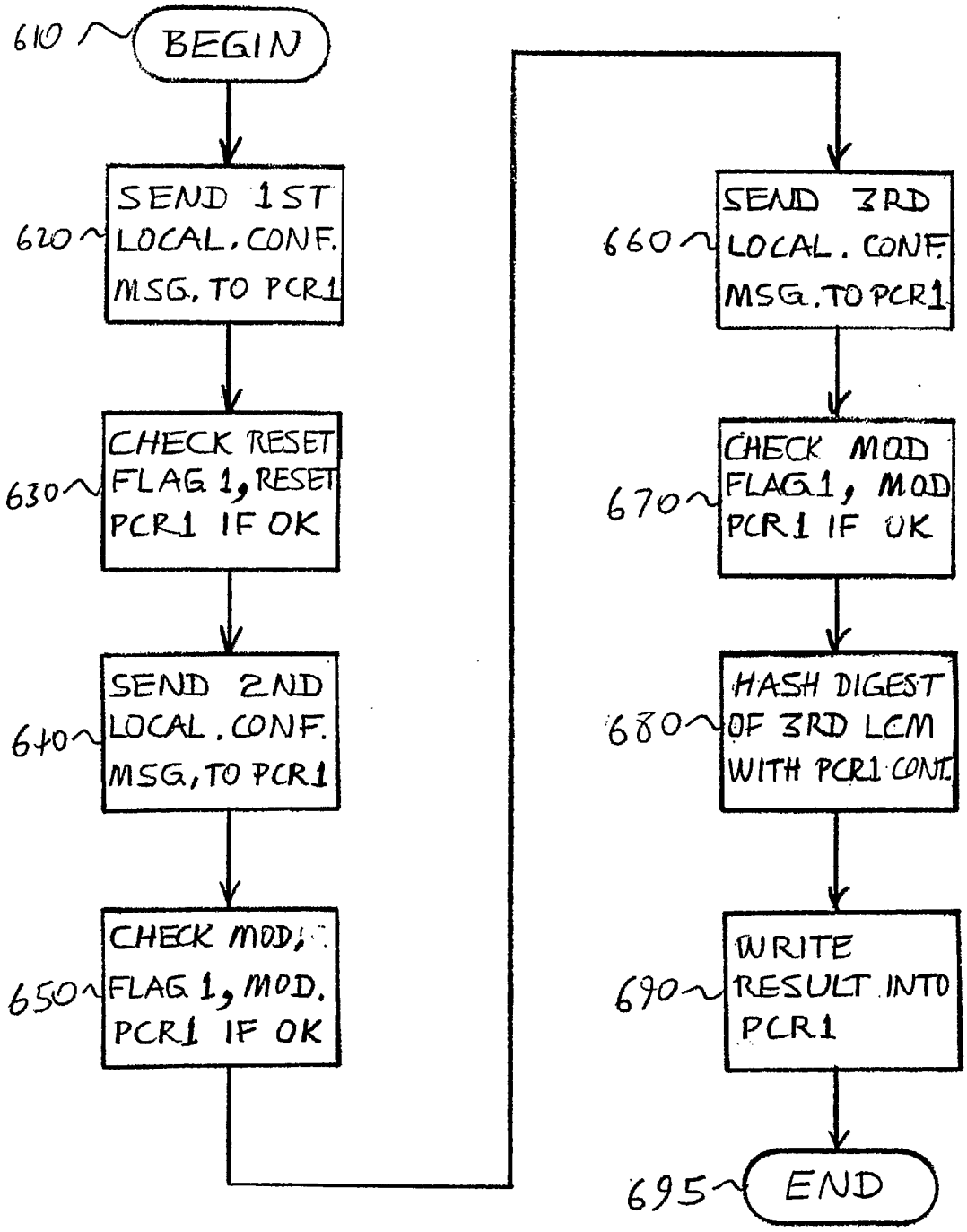
610
BEGIN
620
SEND 1ST LOCAL. CONF. MSG. TO PCR1
630
CHECK RESET FLAG 1, RESET PCR1 IF OK
640
SEND 2ND LOCAL. CONF. MSG, TO PCR1
650
CHECK MOD; FLAG 1, MOD. PCR1 IF OK
660
SEND 3RD LOCAL. CONF. MSG. TO PCR1
670
CHECK MOD FLAG 1, MOD PCR1 IF OK
680
HASH DIGEST OF 3RD LCM WITH PCR1 CONT.
690
WRITE RESULT INTO PCR1
695
END

FIGURE 6