(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 103 92 470 T5** 2005.04.07

(12) # Veröffentlichung

der internationalen Anmeldung mit der
(87) Veröffentlichungs-Nr.: **WO 03/085497**
in deutscher Übersetzung (Art. III § 8 Abs. 2 IntPatÜG)
(21) Deutsches Aktenzeichen: **103 92 470.1**
(86) PCT-Aktenzeichen: **PCT/US03/08762**
(86) PCT-Anmeldetag: **20.03.2003**
(87) PCT-Veröffentlichungstag: **16.10.2003**
(43) Veröffentlichungstag der PCT Anmeldung
in deutscher Übersetzung: **07.04.2005**

(51) Int Cl.[7]: **G06F 1/00**

(30) Unionspriorität:
10/112,169 29.03.2002 US

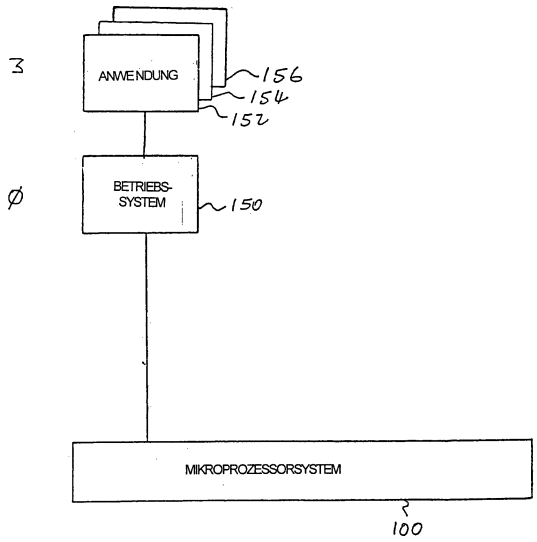
(71) Anmelder:
Intel Corporation, Santa Clara, Calif., US

(74) Vertreter:
BOEHMERT & BOEHMERT, 28209 Bremen

(72) Erfinder:
Sutton II, James, Portland, Oreg., US; Grawrock, David, Aloha, Oreg., US

(54) Bezeichnung: **System und Verfahren zum Ausführen von Initialisierungsbefehlen einer gesicherten Umgebung**

(57) Hauptanspruch: System, das folgendes umfaßt:
einen ersten logischen Prozessor, der einen sicheren Speicher umfaßt, um einen gesicherten Eingabebefehl auszuführen, und
einen Chipsatz, um vor einem Zugriff auf einen Secure Virtuel Machine Monitor durch eine Nicht-Prozessor-Einrichtung zu schützen.



Beschreibung

Gebiet der Erfindung

[0001] Die vorliegende Erfindung betrifft allgemein Mikroprozessorsysteme und insbesondere Mikroprozessorsysteme, die in einer vertrauenswürdigen oder gesicherten Umgebung betrieben werden.

Hintergrund

[0002] Die wachsende Zahl von finanziellen und personengebundenen Transaktionen, die auf lokalen oder entfernten Mikrocomputern ausgeführt werden, gaben einen Anstoß zum Aufbauen von "vertrauenswürdigen" oder "gesicherten" Mikroprozessor-Umgebungen. Das Problem, das diese Umgebungen zu lösen versuchen, ist der Verlust der Geheimhaltung oder daß Daten verfälscht oder mißbraucht werden. Benutzer wollen nicht, daß ihre Privatdaten öffentlich werden. Sie wollen außerdem nicht, daß ihre Daten verändert werden oder in unangebrachten Transaktionen benutzt werden. Beispiele davon umfassen eine unabsichtliche Abgabe von medizinischen Aufzeichnungen oder einen elektronischer Diebstahl von Finanzmitteln aus einer Online-Bank oder anderen Hinterlegungsstellen. In ähnlicher Weise versuchen Provider von Inhalten, den digitalen Inhalt (zum Beispiel Musik, andere Audio- oder Videodaten oder allgemein andere Arten von Daten) vor einem Kopieren ohne Autorisierung zu schützen.

[0003] Existierende vertrauenswürdige Systeme können einen vollständig geschlossenen Satz vertraulicher Software benutzen. Dieses Verfahren ist relativ einfach zu implementieren, hat jedoch den Nachteil, daß es die gleichzeitige Verwendung von üblichen, kommerziell erhältlichen Betriebssystemen und Anwendungssoftware nicht gestattet. Dieser Nachteil beschränkt die Akzeptanz solcher vertraulicher Systeme.

Kurze Beschreibung der Zeichnungen

[0004] Die vorliegende Erfindung wird im Wege von Beispielen, und nicht mittels einer Einschränkung, in den Figuren und den beigefügten Zeichnungen dargestellt, wobei gleiche Bezugszeichen sich auf ähnliche Elemente beziehen und die Figuren Folgendes zeigen:

[0005] **Fig.** 1 ist ein Diagramm einer beispielhaften Softwareumgebung, die in einem Mikroprozessorsystem ausgeführt wird.

[0006] **Fig.** 2 ist ein Diagramm von bestimmten beispielhaften vertrauenswürdigen oder gesicherten Softwaremodulen und eine beispielhafte Systemumgebung gemäß einer Ausführungsform der vorliegenden Erfindung.

[0007] **Fig.** 3 ist ein Diagramm einer beispielhaften vertrauenswürdigen oder gesicherten Softwareumgebung gemäß einer Ausführungsform der vorliegenden Erfindung.

[0008] **Fig.** 4A ist ein schematisches Diagramm eines beispielhaften Mikroprozessorssystems, das dafür eingerichtet ist, die gesicherte Softwareumgebung der **Fig.** 3 zu unterstützen, gemäß einer Ausführungsform der vorliegenden Erfindung.

[0009] **Fig.** 4B ist ein schematisches Diagramm eines beispielhaften Mikroprozessorsystems, das dafür eingerichtet ist, die gesicherte Softwareumgebung der **Fig.** 3 zu unterstützen, gemäß einer alternativen Ausführungsform der vorliegenden Erfindung.

[0010] **Fig.** 5 ist ein schematisches Diagramm eines beispielhaften Mikroprozessorsystems, das dafür eingerichtet ist, die gesicherte Softwareumgebung der **Fig.** 3 zu unterstützen, gemäß einer alternativen Ausführungsform der vorliegenden Erfindung.

[0011] **Fig.** 6 ist eine Darstellung einer Zeitrahmenleiste der Ausführung von Softwarekomponenten gemäß einer Ausführungsform der vorliegenden Erfindung.

[0012] **Fig.** 7 ist ein Ablaufdiagramm der Software und anderer Prozeßblöcke gemäß einer Ausführungsform der vorliegenden Erfindung.

Detailliere Beschreibung

[0013] Die nachfolgende Beschreibung beschreibt Techniken zum Initiieren einer vertrauenswürdigen oder gesicherten Umgebung in einem Mikroprozessorsystem. In der nachfolgenden Beschreibung werden zahlreiche spezielle Details, wie Logikimplementierungen, Softwaremodulzuordnungen, Verschlüsselungstechniken, Bussignalisierungstechniken und Details des Ablauf dargestellt, um ein vollständigeres Verständnis der vorliegenden Erfindung zu liefern. Ein Fachmann wird jedoch erkennen, daß die Erfindung auch ohne diese speziellen Details ausgeführt werden kann. In anderen Fällen werden Steuerstrukturen, Gatelevel-Schaltungen und vollständige Softwarebefehlsequenzen nicht im Detail gezeigt, um nicht von der Erfindung abzulenken. Die Durchschnittsfachleute werden jedoch mit den eingefügten Beschreibungen in der Lage sein, die geeigneten Funktionen ohne unverhältnismäßiges Experimentieren zu implementieren. Die Erfindung wird in Form eines Mikroprozessorsystems offenbart. Die Erfindung kann jedoch auch in anderen Prozessorformen ausgeführt werden, wie mit einem Digitalsignalprozessor, einem Minicomputer oder einem Großrechner.

[0014] Bezug nehmend auf **Fig.** 1 wird eine Dia-

gramm einer beispielhaften Softwareumgebung gezeigt, die in einem Mikroprozessorsystem ausgeführt wird. Die in **Fig.** 1 gezeigte Software ist nicht vertrauenswürdig (unzuverlässig). Beim Betreiben in einem Level mit hoher Berechtigung macht es die Größe und das ständige Aktualisieren des Betriebsystems **150** sehr schwierig, irgendeine Vertrauenswürdigkeitsanalysen in einer zeitnahen Weise auszuführen. Viele der Betriebssysteme sitzen in einem Berechtigungsring null (0), dem höchsten Berechtigungslevel. Die Anwendungen **152**, **154** und **156** besitzen eine viel geringere Berechtigung und befinden sich typischerweise in einem Berechtigungsring drei (3). Die Existenz der verschiedenen Berechtigungsringe und die Aufteilung des Betriebssystems **150** und der Anwendung **152**, **154** und **156** in diesen verschiedenen Berechtigungsringe macht es scheinbar möglich, die Software der **Fig.** 1 in einem vertrauenswürdigen Modus zu betreiben, basierend auf einer Beschlußfassung, den Einrichtungen, die von dem Betriebssystem **150** zur Verfügung gestellt werden, zu vertrauen. In der Praxis ist die Fassung einer solchen Vertrauensentscheidung jedoch oft unpraktikabel. Faktoren, die zu diesem Problem beitragen, umfassen die Größe (Zahl der Zeilen des Codes) des Betriebssystems **150**, die Tatsache, daß das Betriebssystem **150** zahlreiche Updates (neue Codemodule und Korrekturprogramme) empfangen haben kann, und die Tatsache, daß das Betriebssystem **150** auch Codemodule enthalten kann, wie Treiber für Einrichtungen, die von Seiten geliefert werden, die von dem Betriebssystementwickler verschieden sind. Das Betriebssystem **150** kann ein herkömmliches sein, wie Microsoft®, Linux oder Solaris® oder kann ein anderes einschlägig bekanntes oder ein anderweitig erhältliches Betriebssystem sein. Die speziellen Arten oder Namen der Anwendungen oder der Betriebssysteme, die ausgeführt werden oder ablaufen, sind nicht kritisch.

[0015] Bezug nehmend auf **Fig.** 2 wird ein Diagramm von bestimmten beispielhaften vertrauenswürdigen oder gesicherten Softwaremodulen und einer beispielhaften Systemumgebung **200** gemäß einer Ausführungsform der vorliegenden Erfindung gezeigt. In der Ausführungsform der **Fig.** 2 sind ein Prozessor **202**, ein Prozessor **212**, ein Prozessor **222** und optional andere Prozessoren (nicht gezeigt) als separate Hardwareeinheiten gezeigt. In anderen Ausführungsformen kann die Anzahl der Prozessoren anders sein, sowie die Grenzen der verschiedenen Komponenten und der funktionellen Einheiten. In einigen Ausführungsformen können die Prozessoren durch separate Hardwareausführungsthreads oder durch "logische Prozessoren" ersetzt werden, die auf einem oder mehreren physikalischen Prozessoren ablaufen.

[0016] Die Prozessoren **202**, **212**, **222** können verschiedene spezielle Schaltungen oder logische Elemente enthalten, um sichere oder vertrauenswürdige Operationen zu unterstützen. Zum Beispiel kann der Prozessor **202** eine Sicherheitseingabe (SENTER)-Logik **204** enthalten, um die Ausführung spezieller SENTER-Befehle zu unterstützen, die vertrauenswürdige Operationen initiieren können. Der Prozessor **202** kann auch eine Busnachrichtenlogik **206** enthalten, um Spezialbusnachrichten auf einem Systembus **230** in Unterstützung spezieller SENTER-Operationen zu unterstützen. In alternativen Ausführungsformen können Speichersteuerfunktionen eines Chipsatzes **240** Schaltungen in den Prozessoren zugeordnet sein und mehrere Prozessoren können auf einem einzelnen Chip enthalten sein. In diesen Ausführungsformen können Spezialbusnachrichten auch auf Bussen innerhalb der Prozessoren gesendet werden. Die Benutzung von Spezialbusnachrichten kann die Sicherheit oder Vertrauenswürdigkeit des Systems aus verschiedenen Gründen erhöhen. Schaltungselemente, wie die Prozessoren **202**, **212** und **222** oder der Chipsatz **240**, können nur solche Nachrichten ausgeben oder beantworten, wenn diese geeignete Logikelemente von Ausführungsformen der vorliegenden Offenbarung enthalten. Daher kann ein erfolgreicher Austausch der Spezialbusnachrichten dabei helfen, eine korrekte Systemkonfiguration sicherzustellen. Spezialbusnachrichten können auch Aktivitäten ermöglichen, die normalerweise verboten werden sollten, wie das Zurücksetzen eines Plattformkonfigurationsregisters **278**. Die Möglichkeit, potentiell gefährlichen, nicht vertrauenswürdigen Code auf verschiedenen Bustransaktionen ausfindig zu machen, kann eingeschränkt werden, indem ermöglicht wird, daß Spezialbusnachrichten nur in Antwort auf spezielle Sicherheitsbefehle ausgegeben werden.

[0017] Zusätzlich kann der Prozessor **202** einen sicheren Speicher **208** enthalten, um Sicherheitsinitialisierungsoperationen zu unterstützen. In einer Ausführungsform kann der sicheren Speicher **208** ein interner Cache des Prozessors **202** sein, der etwa in einem speziellen Modus betrieben wird. In anderen Ausführungsformen kann der sichere Speicher **208** ein Spezialspeicher sein. Andere Prozessoren, wie der Prozessor **212** und der Prozessor **222**, können auch eine SENTER-Logik **214**, **224**, eine Busnachrichtenlogik **216**, **226** und sichere Speicher **218**, **228** umfassen.

[0018] Ein "Chipsatz" kann als eine Gruppe von Schaltungen und Logiken definiert sein, die einen Speicher und Eingabe/Ausgabe (I/O)-Operationen für einen oder mehrere angeschlossene Prozessoren unterstützen. Einzelne Elemente eines Chipsatzes können zusammen auf einem einzelnen Chip, oder auf einem Paar von Chips gruppiert sein, oder über viele Chips, einschließlich von Prozessoren, verteilt sein. In der Ausführungsform der **Fig.** 2 kann der Chipsatz **240** einen Schaltkreis und eine Logik um-

fassen, um einen Speicher und I/O-Operationen zu unterstützen, um die Prozessoren **202**, **212** und **222** zu unterstützen. In einer Ausführungsform kann der Chipsatz **240** mit einer Anzahl von Speicherseiten **250** bis **262** in einer Einrichtungszugriffseitentabelle **248** verbunden sein, die Steuerinformationen enthält, welche angeben, ob Nichtprozessor-Einrichtungen auf die Speicherseiten **250** bis **262** zugreifen können. Der Chipsatz **240** kann eine Speicherzugriffslogik **247** umfassen, die einen direkten Speicherzugriff (DMA) aus I/O-Einrichtungen zum Auswählen von Teilen der Speicherseiten **250** bis **262** erlauben oder verbieten kann. In einigen Ausführungsformen kann die Einrichtungszugriffslogik **247** alle relevanten Informationen enthalten, um solche Zugriffe zu erlauben oder abzulehnen. In anderen Ausführungsformen kann die Einrichtungszugriffslogik **247** auf solche Informationen zugreifen, die in der Einrichtungszugriffsseitentabelle **248** enthalten sind. Die tatsächliche Zahl der Speicherseiten ist nicht wichtig und wird sich abhängig von den Systemerfordernissen ändern. In anderen Ausführungsformen können die Speicherzugriffsfunktionen außerhalb des Chipsatzes **240** sein. Die Funktionen des Chipsatzes **240** können in alternativen Ausführungsformen weiterhin auf eine oder mehrere physikalische Einrichtungen verteilt sein.

[0019] Der Chipsatz **240** kann zusätzlich seine eigene Busnachrichtenlogik **242** umfassen, um Spezialbusnachrichten auf einem Systembus **230** in Unterstützung der Spezial-SENTER-Operationen zu unterstützen. Einige dieser Spezialbusnachrichten können eine Übertragung der Inhalte eines Schlüsselregisters **240** zu einem Prozessor **202**, **212** oder **222** umfassen oder ermöglichen ein spezielles ALL_JOINED-Flag **274**, das von einem Prozessor **202**, **212** oder **222** untersucht wird. Zusätzliche Merkmale der Busnachrichtenlogik **242** können darin bestehen, Busaktivitäten durch die Prozessoren in einem "EXISTS"-Register **272** zu registrieren und eine bestimmte Spezialbusnachrichtaktivität durch die Prozessoren in einem "JOINS"-Register **272** zu speichern. Eine Übereinstimmung der Inhalte des EXISTS-Register **272** und des JOINS-Registers **272** kann benutzt werden, um das spezielle ALL_JOINED-Flag **274** zu setzen, um anzuzeigen, daß alle Prozessoren in dem System in dem Sicherheitseingabeprozeß teilnehmen.

[0020] Der Chipsatz **240** kann standardisierte I/O-Operationen auf I/O-Bussen unterstützen, wie Peripheral Component Interconnect (PCI), Accelerated Graphics Port (AGP), Universal Serial Bus (USB), Low Pin Count (LPC) Bus oder andere Arten eines I/O-Busses (nicht gezeigt). Eine Schnittstelle **290** kann benutzt werden, um den Chipsatz **240** mit einem Token **276** zu verbinden, der eine oder mehrere Plattformkonfigurationsregister (PCR) **287**, **797** enthält. In einer Ausführungsform kann die Schnittstelle **290** der LPC-Bus (Low Pin Count (LPC) Interface Specification, Intel Corporation, rev. 1.0, 29. Dezember 1997) sein, der mit dem Zusatz bestimmter Sicherheitsverbesserungen modifiziert ist. Ein Beispiel einer solchen Sicherheitsverbesserung wäre eine Ortsbestätigungsnachricht, die einen vorher reservierten Nachrichtenkopf und Adressinformationen verwendet, die auf ein Plattformkonfigurationsregister (PCR) **278** in dem Token **276** zeigen. In einer Ausführungsform kann der Token **276** Spezialsicherheitsmerkmale enthalten, und in einer Ausführungsform kann er ein Vertrautheitsplattformmodul (TPM) **281** umfassen, das in Trusted Computing Platform Alliance (TCPA) Main Specification, Version 1.1a, 1. Dezember 2001, erschienen von TCPA, offenbart ist (zur Zeit der Einreichung der vorliegenden Anmeldung erhältlich über www.trustedpc.com).

[0021] Zwei Softwarekomponenten, die in der Systemumgebung **200** identifiziert sind, sind ein Secure Virtual Machine Monitor (SVMM) **282** Modul und ein Secure Initialization Authenticated Code (SINIT-AC) **280** Modul. Das Modul SVMM **282** kann in einer Systemdiskette oder in einem anderen Massenspeicher gespeichert sein und, falls notwendig, an andere Orte verschoben oder kopiert werden. In einer Ausführungsform kann vor dem Beginn des Sicherheitsstartprozesses das SVMM **282** auf eine oder mehrere der Speicherseiten **250** bis **262** verschoben oder kopiert werden. Nach dem Sicherheitseingabeprozeß kann eine virtuelle Maschinenumgebung erzeugt werden, in der das SVMM **282** als der Code mit der höchsten Berechtigung in dem System betrieben wird, und kann benutzt werden, um den direkten Zugriff auf bestimmte Systemressourcen durch das Betriebssystem oder Anwendungen innerhalb der erzeugten virtuellen Maschinen zu erlauben oder abzulehnen.

[0022] Einige der Aktionen, die von dem Sicherheitseingabeprozeß benötigt werden, können außerhalb des Umfangs einfacher Hardwareimplementierungen liegen und können statt dessen vorteilhafterweise ein Softwaremodul verwenden, dessen Ausführung uneingeschränkt vertrauenswürdig sein kann. In einer Ausführungsform können diese Aktionen durch einen Sicherheitsinitialisierungs(SINIT)-Code ausgeführt werden. Drei beispielhafte Aktionen werden hier dargestellt. Diese Aktionen sollten jedoch nicht als Beschränkung betrachtet werden. Eine Aktion kann es erforderlich machen, daß verschiedene Steuerungen, die kritische Punkte der Systemkonfiguration darstellen, getestet werden, um sicherzustellen, daß die Konfiguration die korrekte Instantiierung der Sicherheitsumgebung unterstützt. In einer Ausführungsform kann ein erforderlicher Test sein, daß die Speichersteuerkonfiguration, die von dem Chipsatz **240** zur Verfügung gestellt wird, nicht zuläßt, daß zwei oder mehr unterschiedliche Systembusadressen den gleichen Ort in den Speicherseiten **250** bis **262** berühren. Eine zweite Aktion kann darin

bestehen, die Einrichtungszugriffsseitentabelle **248** und die Einrichtungszugriffslogik **247** zu konfigurieren, um jene Speicherseiten, die von der speicheransässigen Kopie des SVMM **282** benutzt werden, vor einer Beeinflussung durch Nicht-Prozessor-Einrichtungen zu schützen. Eine dritte Aktion kann darin bestehen, die Identität des SVMM **282** Moduls zu berechnen und zu registrieren und ihm eine Systemsteuerung zu übertragen. Hierin bedeutet "Registrieren" das Einsetzen einer Vertrauenswürdigkeitsmaßeinheit des SVMM **282** in ein Register, zum Beispiel in das PCR **278** oder in das PCR **279**. Wenn diese letztere Aktion vorgenommen wird, kann die Vertrauenswürdigkeit des SVMM **282** durch einen potentiellen Systembenutzer überprüft werden.

[0023] Der SINIT-Code kann von dem Hersteller der Prozessoren oder der Chipsätze produziert werden. Zu diesem Zweck kann der SINIT-Code vertrauenswürdig sein, um bei den Sicherheitsstarts des Chipsatzes **240** zu helfen. Um den SINIT-Code zu verteilen, wird in einer Ausführungsform ein gut bekannter Verschlüsselungshash aus dem gesamten SINIT-Code gemacht, wodurch ein als "Digest" bekannter Wert erzeugt wird. Eine Ausführungsform erzeugt einen 160-Bit-Wert für das Digest. Das Digest kann anschließend durch einen privaten Schlüssel verschlüsselt werden, der in einer Ausführungsform von dem Hersteller des Prozessors aufbewahrt wird, um eine digitale Signatur zu bilden. Wenn der SINIT-Code mit der entsprechenden Digitalsignatur verknüpft wird, kann die Kombination als SINIT-Authentifizierungscode (SINIT-AC) **280** bezeichnet werden. Kopien des SINIT-AC **280** können später überprüft werden, wie nachfolgend erläutert wird.

[0024] Der SINIT-AC **280** kann auf einer Systemdiskette oder einem anderen Massenspeicher in einem festen Medium gespeichert werden und, falls notwendig, an andere Orte verschoben oder kopiert werden. In einer Ausführungsform kann vor dem Beginn des Sicherheitsstartprozesses der SINIT-AC **280** in die Speicherseiten **250** bis **262** verschoben oder kopiert werden, um eine speicheransässige Kopie des SINIT-AC zu bilden.

[0025] Ein beliebiger logischer Prozessor kann den Sicherheitsstartprozeß initiieren und kann dann als initiierender logischer Prozessor (ILP) bezeichnet werden. In dem vorliegenden Beispiel wird der Prozessor **202** der ILP, obgleich ein beliebiger Prozessor auf dem Bussystem **230** der ILP werden könnte. Weder die speicheransässige Kopie des SINIT-AC **280** noch die speicheransässige Kopie des SVMM **282** können zu dieser Zeit als vertrauenswürdig betrachtet werden, da, neben anderen Gründen, die anderen Prozessoren oder die DMA-Einrichtungen die Speicherseiten **250** bis **262** überschreiben können.

[0026] Der ILP (Prozessor **202**) führt einen Spezialbefehl aus. Dieser Spezialbefehl wird als ein Sicherheitseingabe (SENTER)-Befehl bezeichnet und kann von einer SENTER-Logik **204** unterstützt werden. Die Ausführung des SENTER-Befehls kann bewirken, daß der ILP (Prozessor **202**) Spezialbusnachrichten auf dem Systembus **230** ausgibt und anschließend erhebliche Zeitintervalle auf nachfolgende Systemaktionen wartet. Nachdem die Ausführung von SENTER beginnt, wird eine dieser Spezialnachrichten (eine SENTER BUS NACHRICHT), auf dem Systembus **230** verbreitet. Jene von dem ILP verschiedene Prozessoren, die als antwortende logische Prozessoren (RLPs) bezeichnet werden können, antworten auf die SENTER BUS NACHRICHT mit einem internen, nicht maskierbaren Ereignis. In dem vorliegenden Beispiel umfassen die RLPs den Prozessor **212** und den Prozessor **222**. Alle RLPs müssen aktuelle Operationen beenden, eine RLP-Bestätigung (ACK)-Spezialbusnachricht auf dem Systembus **230** senden und dann einen Wartezustand einnehmen. Es wird darauf hingewiesen, daß der ILP auch seine eigene ACK-Nachricht über den Systembus **230** sendet.

[0027] Der Chipsatz **240** kann ein Paar von Registern enthalten, das "EXISTS"-Register **270** und das "JOINS"-Register **272**. Diese Register können benutzt werden, um zu verifizieren, daß der ILP und alle RLPs korrekt auf die SENTER BUS NACHRICHT antworten. In anderen Ausführungsformen kann der Chipsatz **240** allen operationellen logischen Prozessoren in dem System nachgehen, indem er eine "1" in das entsprechende Bit des EXISTS-Register **270** auf jeder Systembustransaktion schreibt, die von dem Prozessor ausgeführt wird. In dieser Ausführungsform enthält jede Transaktion auf dem Systembus **230** ein Identifizierungsfeld, das die Kennung des logischen Prozessors enthält. In einer Ausführungsform besteht dies aus einer physikalischen Prozessorkennung und einer Kennung für den Hardwareausführungsthread in jedem physikalischen Prozessor. Wenn beispielsweise ein Thread, der auf dem Prozessor **222** ausgeführt wird, eine Bustransaktion auf dem Systembus **230** hervorruft, würde der Chipsatz **240** diese logische Prozessorkennung in der Transaktion sehen und eine "1" an den entsprechenden Ort **286** in dem EXISTS-Register **270** schreiben. Während des Sicherheitsstartprozesses, wenn der gleiche Thread auf dem Prozessor **222** seine ACK-Nachricht auf dem Systembus **230** sendet, würde der Chipsatz **240** auch dies sehen und könnte eine "1" an den entsprechenden Ort **288** in dem JOINS-Register **272** schreiben. (Im Beispiel der **Fig.** 2 ist zur Verdeutlichung jeder physikalische Prozessor mit nur einem einzelnen ausgeführten Thread gezeigt. In alternativen Ausführungsformen können die physikalischen Prozessoren mehrere Threads unterstützen und dabei mehrere logische Prozessoren). Wenn die Inhalte des JOINS-Registers **272** zu den Inhalten des EXISTS-Registers **270** passen,

kann der Chipsatz **240** eine ALL_JOINED-Flag **246** setzen, das anzeigt, daß alle Prozessoren korrekt auf die SENTER BUS NACHRICHT geantwortet haben.

[0028] In einer anderen Ausführungsform können das EXISTS-Register **270** und das JOINS-Register **272** damit fortfahren, nach dem Setzen des ALL_JOINED-Flags **246** zur Sicherheit beizutragen. Während der Zeit nach dem Setzen des ALL_JOINED-Flags **246** bis zum Ende der Vertrauenswürdigkeits- oder Sicherheitsoperationen, kann der Chipsatz **240** damit fortfahren, die Buszyklen zu überwachen und mit dem JOINS-Register **272** zu vergleichen. Wenn der Chipsatz während dieser Periode eine Bustransaktion von einem logischen Prozessor sieht, die nicht aktuell in dem JOINS-Register **272** identifiziert ist, kann der Chipsatz **240** annehmen, daß dieser logische Prozessor etwas spät "erschienen" ist. Dies würde implizieren, daß ein solcher logischer Prozessor nicht an dem Sicherheitsstartprozeß teilgenommen hat und daher einen Angreifer (Sicherheitsbedrohung) darstellen könnte. Unter solchen Umständen kann der Chipsatz **240** entsprechend antworten, um diesen Angreifer aus der gesicherten Umgebung herauszuhalten. In einer Ausführungsform kann der Chipsatz **240** ein Systemreset unter solchen Umständen betreiben. In einer zweiten Ausführungsform kann eine ähnliche Ermittlung eines "verspäteten" Prozessors durch jeden logischen Prozessor erreicht werden, der ein spezielles reserviertes Signal auf dem Systembus nach jeder Transaktion ausgibt, die der Verbreitung der ACK-Busnachricht folgt. Wenn in dieser Ausführungsform der Chipsatz **240** nach dem Setzen des ALL_JOINED-Flags **246** Bustransaktion beobachtet, die von einem Prozessor ohne ein bestätigtes Spezialsignal initiiert wurde, kann der Chipsatz **240** wieder annehmen, daß dieser logische Prozessor irgendwie "verspätet" erschienen ist und einen Angreifer darstellen kann.

[0029] Nach dem Erscheinen der SENTER BUS NACHRICHT fragt der ILP (Prozessor **202**) die ALL_JOINED-Flag **246** ab, um zu sehen, wann und ob alle Prozessoren korrekt mit ihren ACKs geantwortet haben. Wenn das Flag **246** nie gesetzt ist, sind verschiedene Implementierungen möglich. Ein Überwachungstimer in dem ILP oder dem Chipsatz oder einem anderen Ort können ein Systemreset hervorrufen. Alternativ kann das System hängen, was einen Reset durch den Betreiber erforderlich macht. In beiden Fällen wird die Beanspruchung einer Sicherheitsumgebung geschützt (dadurch daß der Sicherheitsstartprozeß nicht beendet wird, bis alle Prozessoren teilnehmen), obgleich das System möglicherweise nicht weiterarbeitet. Im normalen Betrieb wird nach kurzer Zeit die ALL_JOINED-Flag **246** gesetzt und der ILP kann sicherstellen, daß alle anderen logischen Prozessoren in einen Wartezustand eingetreten sind.

[0030] Wenn das ALL_JOINED-Flag **246** gesetzt ist, kann der ILP (Prozessor **202**) sowohl eine Kopie des SINIT-AC **280** als auch einen Schlüssel **284** in einen sicherer Speicher **208** verschieben, um den SINIT-Code, der in dem SINIT-AC **280** enthalten ist, zu authentifizierten und nachfolgend auszuführen. In einer Ausführungsform kann dieser Sicherheitsspeicher **208** ein interner Cache des ILP (Prozessor **202**) sein, der etwa in einem speziellen Modus betrieben wird. Der Schlüssel **284** stellt einen öffentlichen Schlüssel dar, der dem privaten Schlüssel entspricht, der zum Verschlüsseln der digitalen Signatur benutzt wird, die in dem SINIT-AC **280** Modul enthalten ist, und wird benutzt, um die digitale Signatur zu überprüfen und dabei den SINIT-Code zu authentifizieren. In einer Ausführungsform kann der Schlüssel **284** bereits in dem Prozessor gespeichert sein, etwa als ein Teil der SENTER-Logik **204**. In einer anderen Ausführungsform kann der Schlüssel in einem Nur-Lese-Schlüsselregister **204** des Chipsatzes **240** gespeichert sein, der von dem ILP gelesen wird. In noch einer anderen Ausführungsform kann das Schlüsselregister **244** von entweder dem Prozessor oder dem Chipsatz tatsächlich ein Verschlüsselungs-Digest des Schlüssels **284** enthalten, wobei der Schlüssel **284** selbst in dem SINIT-AC **280** Modul enthalten ist. In dieser letzteren Ausführungsform liest der ILP das Digest aus dem Schlüsselregister **244**, berechnet einen äquivalenten Verschlüsselungshash über den Schlüssel **284**, der in dem SINIT-AC **280** eingebettet ist, und vergleicht die zwei Digests, um sicherzustellen, daß der zur Verfügung gestellte Schlüssel **284** tatsächlich vertrauenswürdig ist.

[0031] Eine Kopie des SINIT-AC und eine Kopie des öffentlichen Schlüssels können sich dann in dem sicheren Speicher **208** befinden. Der ILP kann nun die Kopie des SINIT-AC durch Entschlüsseln der digitalen Signatur, die in der Kopie des SINIT-AC enthalten ist, unter Verwendung der Kopie des öffentlichen Schlüssels überprüfen. Diese Entschlüsselung erzeugt eine Originalkopie eines Digest des Verschlüsselungshashs. Wenn ein neu berechneter Digest zu diesem Originaldigest paßt, kann die Kopie des SINIT-AC und der darin enthaltene SINIT-Code als vertrauenswürdig betrachtet werden.

[0032] Der ILP kann nun eine andere Spezialbusnachricht, eine SENTER CONTINUE NACHRICHT, über den Systembus **230** ausgeben, die den wartenden RLPs (Prozessor **212**, Prozessor **222**) und dem Chipsatz **240** signalisiert, daß gesicherte Operationen initiiert werden. Der ILP kann nun die eindeutige Identität des SINIT-AC-Moduls registrieren, indem er den Wert des Verschlüsselungsdigits des SINIT-AC-Moduls in ein Plattformkonfigurationsregister **272** in dem Sicherheitstoken **276** schreibt, wie nachfolgend dargestellt. Die Ausführung des SENTER-Befehls des ILPs kann nun durch Übertragen der Ausführungssteuerung auf der vertrauenswürdi-

ge Kopie des SINIT-Codes, der in dem sicherer Speicher **208** des ILP gehalten ist, beendet werden. Der vertrauenswürdige SINIT-Code kann anschließend seinen Systemtest und Konfigurationsaktionen ausführen und die speicheransässige Kopie des SVMM registrieren gemäß der obigen Definition von "Registrieren".

[0033] Die Registrierung der speicheransässigen Kopie des SVMM kann auf verschiedene Weisen ausgeführt werden. In einer Ausführungsform schreibt der SENTER-Befehl, der auf dem ILP abläuft, das berechnete Digest des SINIT-AC in das PCR **278** in dem Sicherheitstoken **276**. Nachfolgend kann der vertrauenswürdige SINIT-Code das berechnete Digest des speicheransässigen SVMM zu dem gleichen PCR **278** oder in ein anderes PCR **279** in dem Sicherheitstoken **276** schreiben. Wenn das SVMM-Digest in das gleiche PCR **278** geschrieben wird, zerhackt der Sicherheitstoken **276** die Originalinhalte (SINIT-Digest) mit den neuen Wert (SVMM-Digest) und schreibt das Ergebnis zurück in das PCR **278**. In Ausführungsformen, bei denen das erste (initiierende) Schreiben in das PCR **278** auf den SENTER-Befehl beschränkt ist, kann das resultierende Digest als eine Wurzelvertrauenswürdigkeit für das System benutzt werden.

[0034] Wenn der vertrauenswürdige SINIT-Code seine Ausführung beendet hat und die Identität des SVMM in einem PCR registriert hat, kann der SINIT-Code eine ILP-Ausführungssteuerung zu dem SVMM übertragen. In einer typischen Ausführungsform können die ersten SVMM-Befehle, die von dem ILP ausgeführt werden, Selbstinitialisierungsroutinen für das SVMM darstellen. Das ILP kann in einer Ausführungsform einzelne Spezialbusnachrichten einer RLP JOIN NACHRICHT zu jedem RLP senden, wodurch jeder der RLPs bei Operationen unter Überwachung der nun ausgeführten Kopie des SVMM teilnimmt. Von diesem Punkt an wird das gesamte System in einem vertrauenswürdigen Modus betrieben, wie bei der Erläuterung der **Fig.** 3 nachfolgend dargestellt wird.

[0035] Bezug nehmend auf **Fig.** 3 wird ein Diagramm einer beispielhaften vertrauenswürdiger oder gesicherten Softwareumgebung gemäß einer Ausführungsform der vorliegenden Erfindung gezeigt. In der Ausführungsform der **Fig.** 3 kann vertrauenswürdige und nicht vertrauenswürdige Software gleichzeitig geladen werden und kann gleichzeitig auf einem einzelnen Computersystem ausgeführt werden. Ein SVMM **350** erlaubt oder verhindert selektiv den direkten Zugriff auf Hardwareressourcen **380** von einem oder mehreren nicht vertrauenswürdigen Betriebssystemen **340** und nicht vertrauenswürdigen Anwendungen **310** bis **330**. In diesem Zusammenhang bedeutet "nicht vertrauenswürdig" nicht notwendigerweise, daß das Betriebssystem oder die Anwendungen absichtlich nicht richtig funktionieren, aber daß die Größe und die Auswahl eines interagierenden Codes es unmöglich machen, zuverlässig festzustellen, daß sich die Software wie gewünscht verhält und daß keine Viren oder anderer fremder Code bei seiner Ausführung stört. In einer typischen Ausführungsform kann der nicht vertrauenswürdige Code aus dem normalen Betriebssystem und Anwendungen bestehen, die auf heutigen Personalcomputern zu finden sind.

[0036] Das SVMM **350** erlaubt oder verhindert außerdem selektiv einen direkten Zugriff auf Hardwareressourcen **380** aus einem oder mehreren vertrauenswürdigen oder sicheren Kernen **360** und einem oder mehreren vertrauenswürdigen Anwendungen **370**. Ein solcher vertrauenswürdiger oder sicherer Kern **360** und solche vertrauenswürdigen Anwendungen **370** können in der Größe und in der Funktionalität begrenzt sein, um bei der Möglichkeit zu helfen, an ihnen eine Vertrauenswürdigkeitsanalyse vorzunehmen. Die vertrauenswürdige Anwendung **370** kann ein beliebiger Softwarecode, ein Programm, eine Routine oder ein Satz von Routinen sein, der in einer sicheren Umgebung ausführbar ist. Daher kann die vertrauenswürdige Anwendung **370** eine Auswahl von Anwendungen oder Codesequenzen sein oder kann eine relativ kleine Anwendung, wie ein Java-Applet, sein.

[0037] Befehle oder Operationen, die normalerweise von einem Betriebssystem **340** oder einem Kern **360** ausgeführt werden und einem Systemressourcenschutz oder Berechtigungen umwandeln könnten, können von der SVMM **350** abgefangen werden und selektiv erlaubt, teilweise erlaubt oder zurückgewiesen werden. Als ein Beispiel würden in einer typischen Ausführungsform Befehle, welche die Seitentabelle des Prozessors ändern, die normalerweise von dem Betriebssystem **340** oder dem Kern **360** ausgeführt würden, statt dessen von dem SVMM **350** abgefangen werden, wodurch sichergestellt wird, daß die Aufforderung nicht versucht, Seitenberechtigungen außerhalb der Domain in der virtuellen Maschine zu ändern.

[0038] Bezug nehmend auf **Fig.** 4A wird eine Ausführungsform eines Mikroprozessorsystems **400** gezeigt, der dazu eingerichtet ist, die gesicherte Softwareumgebung der **Fig.** 3 zu unterstützen. Eine CPU A **410**, eine CPU B **414**, eine CPU C **418** und eine CPU D **422** können mit einem zusätzlichen Mikrocode oder einer Logikschaltung ausgestattet sein, um die Ausführung der Spezialbefehle zu unterstützen. In einer Ausführungsform kann dieser zusätzliche Mikrocode oder die logische Schaltung die SENTER-Logik **204** der **Fig.** 2 sein. Diese Spezialbefehle können die Ausgabe von Spezialbusnachrichten auf dem Systembus **420** unterstützen, welche eine korrekte Synchronisation der Prozesse ermöglichen,

während die Sicherheitsumgebung gestartet wird. In einer Ausführungsform kann die Ausgabe von Spezialbusnachrichten von einer Schaltung unterstützt werden, wie der Busnachrichtenlogik **206** der **Fig.** 2. In gleicher Weise kann der Chipsatz **430** ähnlich zu dem Chipsatz **240** sein und kann die oben erwähnten Spezialzyklen auf dem Systembus **420** unterstützen. Die Zahl der physikalischen Prozessoren kann bei der Implementierung bei einer speziellen Ausführungsform variieren. In einer Ausführungsform können die Prozessoren Mikroprozessoren der Klasse Intel® Pentium® sein. Der Chipsatz **430** kann mit Massenspeichereinrichtungen, wie einem festen Medium **444** oder einem entfernbaren Medium **448**, über einen PCI-Bus **446** oder alternativ über USB **442**, einem Integrated Controller Electronics (IDE)-Bus (nicht gezeigt), einem Small Computer Systems Interconnect (SCSI)-Bus (nicht gezeigt) oder anderen I/O-Bussen verbunden sein. Die festen Medien **444** oder die entfernbaren Medien **448** können Magnetplatten, ein Magnetband, magnetische Disketten, magneto-optische Laufwerke, CD-ROM, DVD-ROM, Flash Memory Cards oder andere Formen von Massenspeichern sein.

[0039] In der Ausführungsform der **Fig.** 4A sind vier Prozessoren CPU A **410**, CPU B **414**, CPU C **418** und CPU D **422** als vier separate Hardwareeinheiten gezeigt. In anderen Ausführungsformen kann die Zahl der Prozessoren differieren. Allerdings können die physikalisch diskreten Prozessoren durch separate Hardwareausführungsthreads ersetzt werden, die auf einen oder mehreren physikalischen Prozessoren ablaufen. Im letzteren Fall bearbeiten diese Threads viele der Attribute der zusätzlichen physikalischen Prozessoren. Um einen allgemeinen Ausdruck zur Beschreibung zu haben, wenn eine beliebige Mischung von mehreren physikalischen Prozessoren und mehreren Threads auf Prozessoren verwendet wird, kann der Ausdruck "logischer Prozessor" benutzt werden, um entweder einen physikalischen Prozessor oder einen Thread, der in einem oder mehreren physikalischen Prozessoren ausgeführt wird, zu beschreiben. Damit kann ein Prozessor mit einem einzelnen Thread als ein einzelner logischer Prozessor betrachtet werden und Prozessoren mit mehreren Threads oder mehreren Kernen können als mehrere logische Prozessoren betrachtet werden.

[0040] In einer Ausführungsform ist der Chipsatz **430** mit einem modifizierten LPC-Bus **450** verbunden. Der modifizierte LPC-Bus **450** kann benutzt werden, um den Chipsatz **430** mit einem Sicherheitstoken **454** zu verbinden. Der Token **454** kann in einer Ausführungsform das TPM **471** umfassen, das von der Trusted Computing Platform Alliance (TCPA) zur Verfügung gestellt wird.

[0041] Bezug nehmend auf die **Fig.** 4B wird eine alternative Ausführungsform eines Mikrocomputersystems **490** gezeigt, das dazu eingerichtet ist, die gesicherte Softwareumgebung der **Fig.** 3 zu unterstützen. Im Unterschied zu der Ausführungsform der **Fig.** 4A kann die CPU A **410** und die CPU B **414** mit dem Chipsatz **428** über einen Systembus A **402** verbunden sein, während die CPU C **418** und die CPU D **422** mit dem Chipsatz **428** über einen Systembus B **404** verbunden sein kann. In anderen Ausführungsformen können mehr als zwei Systembusse verwendet werden. In anderen alternativen Ausführungsformen können Punkt-zu-Punkt-Busse benutzt werden. Spezialbefehle können die Ausgabe von Spezialbusnachrichten auf dem Systembus A **402** und auf dem Systembus B **404** unterstützen, die eine korrekte Synchronisierung der Prozessoren ermöglichen während dem Starten der Sicherheitsumgebung. In einer Ausführungsform kann die Ausgabe einer Spezialbusnachricht von einer Schaltung unterstützt werden, wie von der Busnachrichtenlogik **206** der **Fig.** 2.

[0042] In einer Ausführungsform ist der Chipsatz **428** zur Aufrechterhaltung der Übereinstimmung und der Kohärenz zwischen dem Systembus A **402** und dem Systembus B **404** verantwortlich. Wenn eine standardisierte oder eine Spezialbusnachricht über den Systembus A **402** gesendet wird, spiegelt der Chipsatz **428** diese Nachricht (falls geeignet) auf dem Systembus B **404** wider und umgekehrt.

[0043] In einer alternativen Ausführungsform behandelt der Chipsatz **428** den Systembus A **402** und den Systembus B **404** als unabhängige Untersysteme. Beliebige Spezialbusnachrichten, die auf dem Systembus A **402** ausgegeben werden, wirken nur auf Prozessoren an diesem Bus. In gleicher Weise wirken Spezialbusnachrichten, die auf dem Systembus B **404** ausgegeben werden, nur auf Prozessoren an diesem Bus. Jeglicher geschützter Speicher, der gegenüber dem Systembus A **402** aufgebaut ist, ist nur von Prozessoren ansprechbar, die mit dem Systembus A **402** verbunden sind, und die Prozessoren an dem Systembus B **404** werden als nichtvertrauenswürdige Einrichtungen behandelt. Um Zugriff auf irgendeinen geschützten Speicher zu erlangen, der für die CPU A **410** und die CPU B **414** auf dem Systembus A **402** aufgebaut wurde, müssen die Prozessoren CPU C **418** und CPU D **422** auf dem Systembus B **404** ihre eigene SENTER-Prozesse ausführen, wodurch eine registrierte Umgebung erzeugt wird, die gleich zu der von den Prozessoren auf dem Systembus A **402** erzeugten ist.

[0044] Bezug nehmend auf **Fig.** 5 wird ein schematisches Diagramm eines beispielhaften Mikroprozessorsystems **500**, das dafür eingerichtet ist, die gesicherte Softwareumgebung der **Fig.** 3 zu unterstützen, gemäß einer alternativen Ausführungsform der vorliegenden Erfindung gezeigt: Im Unterschied zu der Ausführungsform der **Fig.** 4A an kann jeder Pro-

zessor (zum Beispiel die CPU **510**) bestimmte Chipsatzfunktionen (zum Beispiel die Chipsatzfunktionen **594**) umfassen, die zum Beispiel Speichersteuerfunktionen und Einrichtungszugriffslogikfunktionen ausführen. Diese Chipsatzfunktionen ermöglichen dabei die direkte Verbindung von Speicher (zum Beispiel dem Speicher A **502**) mit dem Prozessor. Andere Chipsatzfunktionen können in einem separaten Chipsatz **530** verbleiben. Spezialbusnachrichten können über den Systembus **520** ausgegeben werden.

[0045] Jeder Prozessor kann einen indirekten Zugriff auf einen Speicher vornehmen, der mit anderen Prozessoren verbunden ist. Diese Zugriffe können jedoch erheblich langsamer, im Vergleich zu Zugriffen auf prozessoreigenen Speicher sein. Vor dem Start des SENTER-Prozesses kann die Software Kopien von SINIT-AC **566** und SVMM **574** von einem festen Medium **544** in einem lokalen Speicher **504** verschieben, und dabei eine Kopie des SINIT-AC **556** und eine Kopie des SVMM **572** bilden. In dieser Ausführungsform kann der Speicher **504** ausgewählt werden, da direkt von dem Prozessor auf ihn zugegriffen werden kann, der dafür bestimmt ist, der ILP zu sein. Im Beispiel der **Fig.** 5 ist dies die CPU B **514**. Alternativ können die Kopien des SINIT-AC **566** und der SVMM **574** in anderen Speichern plaziert werden, die mit anderen (nicht-ILP) Prozessoren verknüpft sind, so lange der ILP **514** die Möglichkeit hat, auf diese Speicher zuzugreifen. Die CPU B ILP **514** startet den Sicherheitseingabeprozeß durch Abgeben des SENTER-Befehls, wie bereist bei **Fig.** 2 beschrieben, und mit ähnlichen Konsequenzen und ausgegeben Buszyklen. Der Chipsatz **530** kann das EXISTS-Register **576**, das JOINS-Register **580** und das ALL_JOINED-Flag **584**, wie im Zusammenhang mit **Fig.** 2 beschrieben, verwenden, um zu bestimmen, ob alle Prozessoren korrekt auf die SENTER BUS NACHRICHT geantwortet haben, und signalisiert diese Information an den ILP. Der ILP (CPU B **514**) kann wieder die speicheransässige Kopie des SINIT-AC **556** in einen sicheren Speicher **560** zusammen mit einer Kopie des öffentlichen Schlüssels **564** verschieben. Nach der Überprüfung und Registrierung des SINIT-AC **556** kann der ILP damit fortfahren, die speicheransässige Kopie des SVMM **572** zu überprüfen und zu registrieren.

[0046] Bezug nehmend auf die **Fig.** 6 wird eine Zeichnung einer Zeitrahmenleiste von verschiedenen Operationen gemäß einer Ausführungsform der vorliegenden Erfindung gezeigt. Die Zeitrahmenleiste der **Fig.** 6 zeigt den gesamten Zeitablauf der Operationen, die im Zusammenhang mit dem beispielhaften System diskutiert wurden, das im Zusammenhang mit **Fig.** 2 vorhergehend diskutiert wurde. Wenn die Software festlegt, daß sichere oder vertrauenswürdige Operationen gewünscht sind, lokalisiert eine Software ein SINIT-AC **280** und ein SVMM **282**, die für einen nachfolgenden SENTER-Befehl verfügbar sind, und macht eine Kopie davon. In diesem Beispiel lädt die Software eine Kopie der SINIT-AC **280** und eine Kopie des SVMM **282** in eine oder mehrere Speicherseiten **250** bis **262**. Ein Prozessor, in dem vorliegenden Beispiel der Prozessor **202**, wird dann als der ILP ausgewählt, welcher den SENTER-Befehl zur Zeit **612** ausgibt. Zu der Zeit **614** gibt der SENTER-Befehl des ILP die SENTER BUS NACHRICHT **616** aus. Der ILP gibt dann sein eigenes SENTER-ACK **608** zur Zeit **618** aus, bevor er einen Zustand zum Warten auf das Chipsatz-Flag zur Zeit **628** einnimmt.

[0047] Jeder RLP, wie der Prozessor **222**, antwortet auf die SENTER BUS NACHRICHT **616** durch Abschließen des aktuellen Befehls während der Zeit **620**. Der RLP gibt dann seinen SENTER-ACK **622** aus und nimmt einen Zustand **634** ein, bei dem er auf eine SENTER CONTINUE NACHRICHT wartet.

[0048] Der Chipsatz **240** verbringt die Zeit **624** damit, das JOINS-Register **272** in Reaktion auf die SENTER-ACK Nachrichten, die auf dem Systembus **230** beobachtet werden, zu setzen. Wenn die Inhalte des JOINS-Registers **272** zu den Inhalten des EXISTS-Registers **270** passen, setzt der Chipsatz **240** die ALL_JOINED-Flag **246** zur Zeit **626**.

[0049] Zu dieser Zeit bleibt der ILP in einer Schleife während er die ALL_JOINED-Flag **246** abruft. Wenn die ALL_JOINED-Flag **246** gesetzt ist und der ILP feststellt, daß die ALL_JOINED-Flag **246** zur Zeit **630** gesetzt ist, kann der ILP die SENTER CONTINUE NACHRICHT während der Zeit **632** ausgeben. Wenn die SENTER CONTINUE NACHRICHT auf dem Systembus **230** zur Zeit **636** verbreitet ist, können die RLPs in einen Wait-for-Join-Zustand eintreten. Zum Beispiel tritt der RLP des Prozessors **222** in einen Wait-for-Join-Zustand in der Zeitperiode **638** ein.

[0050] Nach dem Ausgeben der SENTER CONTINUE NACHRICHT kann der ILP (in der Zeitperiode **640**) den öffentlichen Schlüssel des Schlüsselregisters **244** des Chipsatzes **240** und eine Kopie des SINIT-AC in seinen sicheren Speicher **208** geben, um eine Kopie des Schlüssels und eine Kopie des SINIT-AC zu bilden. In anderen Ausführungsformen kann das Schlüsselregister **244** ein Digest des öffentlichen Schlüssels enthalten und der tatsächliche öffentliche Schlüssel kann in oder mit der SINIT-AC enthalten sein. Nach der Authentifizierung der Kopie des SINIT-AC, wie oben in Verbindung mit **Fig.** 2 beschrieben, kann der ILP tatsächlich die Kopie des SINIT-AC in dem sicheren Speicher **208** ausführen.

[0051] Nachdem die Kopie des SINIT-AC in dem sicheren Speicher **208** die Ausführung beginnt, überprüft und registriert sie die speicheransässige Kopie des SVMM (während der Zeitperiode **640**). Nachdem

die Kopie des SVMM dem PCR **278** des Sicherheitstoken **276** registriert ist, beginnt die speicheransässige Kopie des SVMM selbst mit der Ausführung. Zu dieser Zeit, während die Zeitperiode **650** anhält, werden SVMM-Operationen in dem ILP aufgebaut.

[0052] Unter den ersten Dingen, welche die ILP SVMM-Operation macht, ist die Ausgabe einer einzelnen RLP JOIN NACHRICHT an den Systembus **230**. Ein Beispiel ist eine JOIN NACHRICHT **644** des Prozessors **222**. Diese Nachricht kann den Ort in dem Speicher enthalten, an dem der RLP Prozessor **222** bei der Ausführung der registrierten speicheransässigen Kopie des SVMM anknüpft. Alternativ können die ILP SVMM-Operationen einen Speicherort an einem vorbestimmten Ort in dem Chipsatz oder dem Speicher registriert haben, und nach dem Empfangen der JOIN NACHRICHT ruft der RLP seine Startadresse von diesem Ort auf. Nach dem Empfangen der JOIN NACHRICHT des Prozessors **222** und der Ermittlung seiner Startadresse springt der RLP Prozessor **222** (während der Zeitperiode **646**) an diesen Ort und verknüpft die Ausführung der registrierten, speicheransässigen Kopie des SVMM.

[0053] Nachdem alle RLPs die registrierte, speicheransässige Kopie des SVMM verknüpft haben, werden gesicherte Operationen über das Mikrocomputersystem **200** aufgebaut.

[0054] Bezug nehmend auf **Fig.** 7 wird ein Ablaufdiagramm der Software und anderer Prozeßblöcke gemäß einer Ausführungsform der Erfindung gezeigt. Zur Klarheit zeigt die **Fig.** 7 nur Prozeßblöcke für einen einzelnen beispielhaften RLP. In anderen Ausführungsformen können verschiedene antwortende logische Prozessoren vorhanden sein.

[0055] Der Prozeß **700** beginnt beim Block **710**, wenn ein logischer Prozessor eine Kopie der SINIT-AC und der SVMM-Module macht, die für einen Zugriff durch einen nachfolgenden SENTER-Befehl verfügbar sind. In diesem Beispiel lädt der ILP im Block **712** den SINIT-AC und den SVMM-Code aus einem Massenspeicher in einen physikalischen Speicher. In alternativen Ausführungsformen kann dies ein beliebiger logischer Prozessor tun, und nicht nur der ILP. Ein Prozessor wird der ILP durch Ausführen des SENTER-Befehls, wie im Block **714** gezeigt. Im Block **716** gibt der ILP SENTER-Befehl eine SENTER BUS NACHRICHT aus. Der ILP gibt dann Block **718** seine eigene SENTER-ACK Nachricht an den Chipsatz aus. Der ILP nimmt dann einen Wartezustand ein, der als Entscheidungsblock **720** gezeigt ist, und wartet darauf, daß der Chipsatz sein ALL_JOINED-Flag setzt.

[0056] Nachdem jeder RLP die SENTER BUS NACHRICHT im Block **770** empfängt, stoppt er die Ausführung mit dem Ende des aktuellen Befehls und gibt im Block **772** seinen eigenen SENTER-ACK aus. Jeder RLP nimmt dann einen Wartezustand ein, als Entscheidungsblock **774** gezeigt, und wartet, daß eine SENTER CONTINUE NACHRICHT von dem ILP ankommt.

[0057] Der Chipsatz setzt die entsprechenden Bits in dem JOINS-Register, wenn die SENTER ACK Nachricht empfangen ist. Wenn die Inhalte des JOINS-Registers gleich zu den Inhalten des EXISTS-Registers sind, setzt der Chipsatz sein ALL_JOINED-Flag, die dem ILP signalisiert, von dem Entscheidungsblock **720** fortzufahren. Nach Verlassen des Entscheidungsblocks **720** über den JA-Pfad, gibt der ILP dann eine SENTER CONTINUE NACHRICHT im Block **722** aus. Dies signalisiert jedem RLP, von dem Entscheidungsblock **774** fortzufahren. Jeder RLP nimmt dann einen zweiten Wartezustand ein, der als Entscheidungsblock **776** gezeigt ist, und wartet auf eine SENTER JOIN NACHRICHT.

[0058] In der Zwischenzeit verschiebt der ILP Block **724** den öffentlichen Schlüssel des Chipsatzes und die speicheransässige Kopie des SINIT-AC in seinen eigenen sicheren Speicher zur sicheren Ausführung. Der ILP benutzt im Block **726** den Schlüssel, um die sicherheitsspeicheransässige Kopie des SINIT-AC zu überprüfen und führt sie dann aus. Die Ausführung des SINIT-AC kann Tests der Systemkonfigurationen und der SVMM-Kopie vornehmen, registriert dann die SVMM-Identität und beginnt schließlich mit der Ausführung des SVMM im Block **728**. Als Teil der Aktionen, die in Block **728** ausgeführt werden, kann der ILP SINIT-Code die Einrichtungszugriffsseitentabelle **248** und die Einrichtungszugriffslogik **247** des Speichers und des Chipsatzes konfigurieren, um jene Speicherseiten, die von der speicheransässigen Kopie des SVMM **282** benutzt werden, vor einem Eingriff durch Nicht-Prozessor-Einrichtungen zu schützen, wie in Block **754** gezeigt.

[0059] Nachdem der ILP die Ausführung unter der Steuerung des SVMM beginnt, sendet der ILP-Block **730** eine individuelle SENTER JOIN NACHRICHT zu jedem RLP. Nach dem Ausgeben der SENTER JOIN NACHRICHT beginnt der ILP im Block **732** mit den SVMM-Operationen.

[0060] Der Empfang der SENTER JOIN NACHRICHT bewirkt, daß jeder RLP den Wartezustand, der in dem Block **776** dargestellt ist, entlang dem JA-Pfad verläßt und beginnt mit den SVMM-Operationen im Block **780**. Die SENTER JOIN NACHRICHT kann die SVMM-Einsprungstelle enthalten, zu der der RLP beim Verknüpfen der SVMM-Operationen abzweigt. Alternativ kann der ILP SVMM-Code die geeignete RLP-Einsprungstelle in einem Systemort (zum Beispiel in dem Chipsatz) registrieren, der von der RLP nach dem Empfang der SENTER JOIN NACHRICHT abgerufen wird.

[0061] Obgleich verschiedene offenbarte Ausführungsformen ein oder mehrer Prozessoren enthalten (entweder logische oder physikalische Prozessoren), ist zu verstehen, daß solche Mehrfachprozessoren und/oder Systeme mit mehreren Threads in größerem Detail beschrieben werden, um die zusätzliche Komplexität zu erklären, die mit dem Sichern eines Systems mit mehreren logischen oder physikalischen Prozessoren verbunden ist. Eine Ausführungsform, die wahrscheinlich auch in weniger komplexen Systemen vorteilhaft ist, kann nur einen Prozessor benutzen. In einigen Fällen kann der eine physikalische Prozessor mehrere Threads aufweisen und daher mehrere logische Prozessoren umfassen (und demgemäß einen ILP und einen RLP, wie beschrieben). In anderen Fällen kann jedoch ein System mit einem einzelnen Prozessor und einem einzelnen Thread benutzt werden und immer noch die offenbarten Sicherheitsbearbeitungstechniken anwenden. In solchen Fällen ist möglicherweise kein RLP vorhanden. Die Sicherheitsbearbeitungstechniken wirken jedoch immer noch, um die Wahrscheinlichkeit zu verringern, daß Daten gestohlen oder in einer nichtautorisierter Weise manipuliert werden können.

[0062] In der vorhergehenden Beschreibung wurde die Erfindung in Bezug auf bestimmte beispielhafte Ausführungsformen davon beschrieben. Es wird jedoch deutlich, daß verschiedene Modifikationen und Änderungen daran vorgenommen werden können, ohne von dem breiteren Gedanken und dem Umfang der Erfindung, wie er in den nachfolgenden Ansprüchen dargelegt ist, abzuweichen. Die Beschreibung und die Zeichnungen sind daher eher in einem beispielhaften Sinne als in einem beschränkenden Sinne zu verstehen.

Zusammenfassung

[0063] Ein Verfahren und eine Vorrichtung zum Initialisieren von Sicherheitsoperationen in einem Mikroprozessorsystem wird beschrieben. In einer Ausführungsform initiiert ein initialisierender logischer Prozessor den Prozeß durch anhalten der Ausführung der anderen logischen Prozessoren und anschließend durch Laden einer Initialisierungs- und Secure Virtual Machine Monitor Software in einen Speicher. Der Initialisierungsprozessor lädt dann die Initialisierungssoftware in einen sicheren Speicher zur Authentifizierung und zur Ausführung. Die Initialisierungssoftware authentifiziert und registriert die Secure Virtual Machine Monitor Software vor den sicheren Systemoperationen.

Patentansprüche

1. System, das folgendes umfaßt:
einen ersten logischen Prozessor, der einen sicheren Speicher umfaßt, um einen gesicherten Eingabebefehl auszuführen, und
einen Chipsatz, um vor einem Zugriff auf einen Secure Virtuel Machine Monitor durch eine Nicht-Prozessor-Einrichtung zu schützen.

2. System nach Anspruch 1, bei dem der gesicherte Eingabebefehl dazu dient, den ersten logischen Prozessor zu veranlassen, eine Spezialbusnachricht an einen zweiten logischen Prozessor auszugeben, um den zweiten logischen Prozessor mit dem ersten logischen Prozessor in sicheren Operationen zu synchronisieren.

3. System nach Anspruch 1, bei dem sich der sichere Speicher in einem Cache des ersten logischen Prozessors befindet.

4. System nach Anspruch 1, bei dem der sichere Speicher vor einem Zugriff durch Schaltungen geschützt ist, die von dem ersten logischen Prozessor verschieden sind.

5. System nach Anspruch 1, das weiterhin einen Sicherheitstoken umfaßt, der ein Plattformkonfigurationsregister aufweist, um ein Digest zu speichern.

6. System nach Anspruch 1, das weiterhin einen zweiten logischen Prozessor umfaßt, um auf eine erste Spezialbusnachricht von dem gesicherten Eingabebefehl zu antworten.

7. System nach Anspruch 6, bei dem der zweite logische Prozessor dazu dient, die Ausführung eines aktuellen Befehls abzuschließen und eine zweite Spezialbusnachricht in Antwort auf die erste Spezialbusnachricht auszugeben.

8. System nach Anspruch 7, bei dem der Chipsatz dazu dient, ein Flag in Antwort auf den Empfang der zweiten Spezialbusnachricht zu setzen.

9. System nach Anspruch 8, bei dem der zweite logische Prozessor an einen Einsprungspunkt des Secure Virtual Machine Monitor in Reaktion auf eine dritte Spezialbusnachricht springt.

10. Verfahren, das folgendes umfaßt:
Synchronisieren eines ersten logischen Prozessors und eines zweiten logischen Prozessors,
Authentifizieren eines Initialisierungscodemoduls,
Authentifizieren eines Secure Virtuel Machine Monitor und
Ausführen eines Secure Virtuel Machine Monitor.

11. Verfahren nach Anspruch 10, das weiterhin ein Senden einer Spezialbusnachricht zu dem zweiten logischen Prozessor umfaßt, um den Secure Virtuel Machine Monitor auf dem zweiten logischen Prozessor reagierend auszuführen.

12. Verfahren nach Anspruch 10, bei dem das

Synchronisieren eine Spezialbusnachricht umfaßt, um den zweiten logischen Prozessor zu veranlassen, eine Ausführung anzuhalten und eine Bestätigung zu senden.

13. Verfahren nach Anspruch 12, bei dem das Synchronisieren ein Setzen eines Flags in dem Chipsatz in Reaktion auf die Bestätigung umfaßt.

14. Verfahren nach Anspruch 10, bei dem das Authentifizieren eines Initialisierungscodemoduls ein Verschieben einer Kopie des Initialisierungscodemoduls und eines öffentlichen Schlüssels in einem sicheren Speicher umfaßt.

15. Verfahren nach Anspruch 14, bei dem das Authentifizieren eines Initialisierungscodemoduls ein Vergleichen eines ersten Digest des Initialisierungscodemoduls mit einem zweiten Digest des Initialisierungscodemoduls umfaßt.

16. Verfahren nach Anspruch 10, bei dem das Authentifizieren eines Secure Virtual Machine Monitors ein Ausführen des Initialisierungscodemoduls umfaßt.

17. Verfahren nach Anspruch 16, bei dem das Authentifizieren eines Secure Virtual Machine Monitors ein Registrieren des Virtual Machine Monitors in einem Plattformkonfigurationsregister umfaßt.

18. Vorrichtung, die folgendes umfaßt:
eine Einrichtung zum Synchronisieren eines ersten logischen Prozessors und eines zweiten logischen Prozessors,
eine Einrichtung zum Authentifizieren eines Initialisierungscodemoduls,
eine Einrichtung zum Authentifizieren eines Secure Virtual Machine Monitors und
eine Einrichtung zum Ausführen des Secure Virtual Machine Monitors in dem ersten logischen Prozessor.

19. Vorrichtung nach Anspruch 18, die weiterhin eine Einrichtung zum Senden einer ersten Spezialbusnachricht zu einem zweiten logischen Prozessor umfaßt, um den zweiten Secure Virtuel Machine Monitor in dem zweiten logischen Prozessor auszuführen.

20. Vorrichtung nach Anspruch 18, die weiterhin eine Einrichtung zum Verschieben einer Kopie des Initialisierungscodes und eines öffentlichen Schlüssels in einem sicheren Speicher umfaßt.

21. Vorrichtung nach Anspruch 20, die weiterhin eine Einrichtung zum Vergleichen eines ersten Digests des Initialisierungscodemoduls mit einem zweiten Digest des Initialisierungscodemoduls umfaßt.

22. Vorrichtung nach Anspruch 18, die weiterhin eine Einrichtung zum Registrieren des Secure Virtual Machine Monitors umfaßt.

23. Prozessor, der folgendes umfaßt:
eine sichere Eingabelogik zum Ausführen eines ersten Befehls zum Aufrufen einer sicheren Operationsinitialisierung und zum Ermitteln eines Zeitpunktes, um mit einer Ausführung eines Secure Initialization Authenticated Code fortzufahren, und
eine Busnachrichtenlogik, um eine erste Spezialbusnachricht in Reaktion auf den ersten Befehl zu senden und um eine zweite Spezialbusnachricht in Reaktion auf den ermittelten Zeitpunkt zu senden.

24. Prozessor nach Anspruch 23, bei dem der Zeitpunkt einer Ausgabe einer Bestätigung eines ersten logischen Prozessor nachfolgt.

25. Prozessor nach Anspruch 23, bei dem die sichere Eingabelogik weiterhin dazu dient, ein Flag-Register in einem Chipsatz aufzurufen, um den Zeitpunkt zu ermitteln.

26. Prozessor nach Anspruch 23, bei dem die sichere Eingabelogik weiterhin dazu dient, einen Schlüssel einzugeben und ein Codemodul nach dem Zeitpunkt zu authentifizieren.

27. Prozessor nach Anspruch 23, bei dem die Busnachrichtenlogik weiterhin dazu dient, eine dritte Spezialbusnachricht zu senden, die einen Codeeinsprungspunkt umfaßt.

28. Chipsatz, der folgendes umfaßt:
eine Busnachrichtenlogik, die auf eine erste Spezialbusnachricht von dem ersten logischen Prozessor reagiert, um eine Sicherheitsoperation vorzubereiten, und
ein Register, um eine Bestätigung von einem zweiten logischen Prozessor in Reaktion auf die erste Spezialbusnachricht zu speichern.

29. Chipsatz nach Anspruch 28, bei dem der Chipsatz dazu dient, das Register mit einer logischen Prozessoraktivität zu vergleichen, um zu ermitteln, wann dem ersten logischen Prozessor zu signalisieren ist, mit einer Sicherheitsoperationsinitialisierung fortzufahren.

30. Chipsatz nach Anspruch 29, bei dem das Signal das Setzen eines Flags umfaßt.

31. Chipsatz nach Anspruch 28, der weiterhin eine Einrichtungszugriffslogik umfaßt, um einen Secure Virtual Machine Monitor zu sperren.

32. Chipsatz nach Anspruch 28, der weiterhin ein Schlüsselregister umfaßt, um einen Schlüssel zu dem ersten logischen Prozessor nach der ersten

Spezialbusnachricht zu senden.

33. System, das folgendes umfaßt:
einen logischen Prozessor mit einer sicheren Eingabelogik und einer ersten Busnachrichtenlogik, die auf die sichere Eingabelogik reagiert, und
einen Chipsatz mit einer zweiten Busnachrichtenlogik, um eine erste Spezialbusnachricht von dem ersten Busnachrichtensystem zu empfangen, und mit einem Flag, das in Reaktion auf eine Bestätigung gesetzt wird.

34. System nach Anspruch 33, das weiterhin einen Secure Initialization Authenticated Code umfaßt, um Sicherheitsoperationen in Reaktion auf die sichere Eingabelogik zu initiieren.

35. System nach Anspruch 34, das weiterhin einen Schlüssel umfaßt, der von dem logischen Prozessor zum Authentifizieren des Secure Initialization Authenticated Code benutzt wird.

36. System nach Anspruch 34, bei dem die erste Busnachrichtenlogik eine zweite Spezialbusnachricht ausgibt, und bei dem der logische Prozessor den Secure Initialization Authenticated Code in einem sicheren Speicher nach der zweiten Spezialbusnachricht verschiebt.

37. System nach Anspruch 34, das weiterhin einen Secure Virtual Machine Monitor umfaßt.

38. System nach Anspruch 37, bei dem der Secure Initialization Authenticated Code eine Initialisierung des Secure Virtual Machine Monitors ausführt.

39. System nach Anspruch 38, bei dem die Initialisierung eine Authentifizierung umfaßt, und bei dem der Chipsatz eine Einrichtungszugriffslogik umfaßt, um vor einem Nicht-Prozessor-Zugriff auf den Secure Virtual Machine Monitor in Reaktion auf die Initialisierung zu schützen.

40. System nach Anspruch 38, bei dem die erste Busnachrichtenlogik eine dritte Spezialbusnachricht in Reaktion auf die Initialisierung ausgibt.

41. System nach Anspruch 40, bei dem die dritte Spezialbusnachricht einen Codeeinsprungpunkt für den Secure Virtual Machine Monitor umfaßt.

42. Verfahren, das folgendes umfaßt:
Übertragen einer Spezialbusnachricht,
Authentifizieren eines Initialisierungscodes in einem ersten logischen Prozessor,
Authentifizieren eines Secure Virtual Machine Monitor und
Ausführen des Secure Virtual Machine Monitor in dem ersten logischen Prozessor.

43. Verfahren nach Anspruch 42, das weiterhin ein Übertragen einer Bestätigung in Reaktion auf die erste Busnachricht umfaßt.

44. Verfahren nach Anspruch 42, das weiterhin ein Anhalten einer Ausführung in dem zweiten logischen Prozessor und ein Senden einer Bestätigung umfaßt.

45. Verfahren nach Anspruch 44, das weiterhin ein Setzen eines Flags in einem Chipsatz in Reaktion auf die Bestätigung umfaßt.

46. Verfahren nach Anspruch 42, bei dem die Authentifizierung eines Initialisierungscodes ein Verschieben einer Kopie des Initialisierungscodes und eines öffentlichen Schlüssels in einen sicheren Speicher umfaßt.

47. Verfahren nach Anspruch 46, bei dem die Authentifizierung eines Initialisierungscodes ein Vergleichen eines ersten Digests des Initialisierungscodes mit einem zweiten Digest des Initialisierungscodes umfaßt.

48. Verfahren nach Anspruch 42, bei dem die Authentifizierung eines Secure Virtual Machine Monitors ein Ausführen des Initialisierungscodes umfaßt.

49. Verfahren nach Anspruch 48, bei dem die Authentifizierung eines Secure Virtual Machine Monitors ein Registrieren des virtuellen Maschinenmonitors in einem Plattformkonfigurationsregister umfaßt.

Es folgen 8 Blatt Zeichnungen

Anhängende Zeichnungen

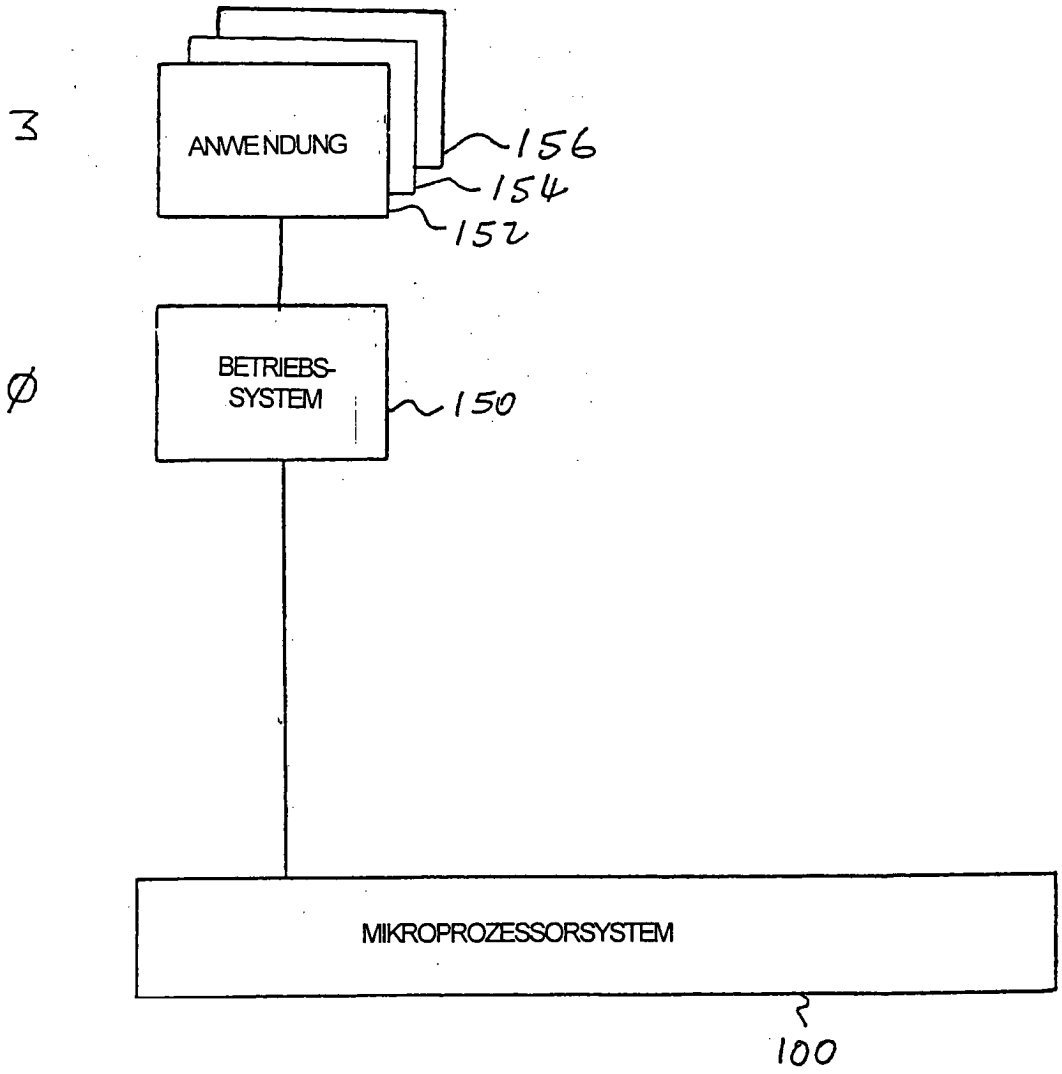


Fig. 1

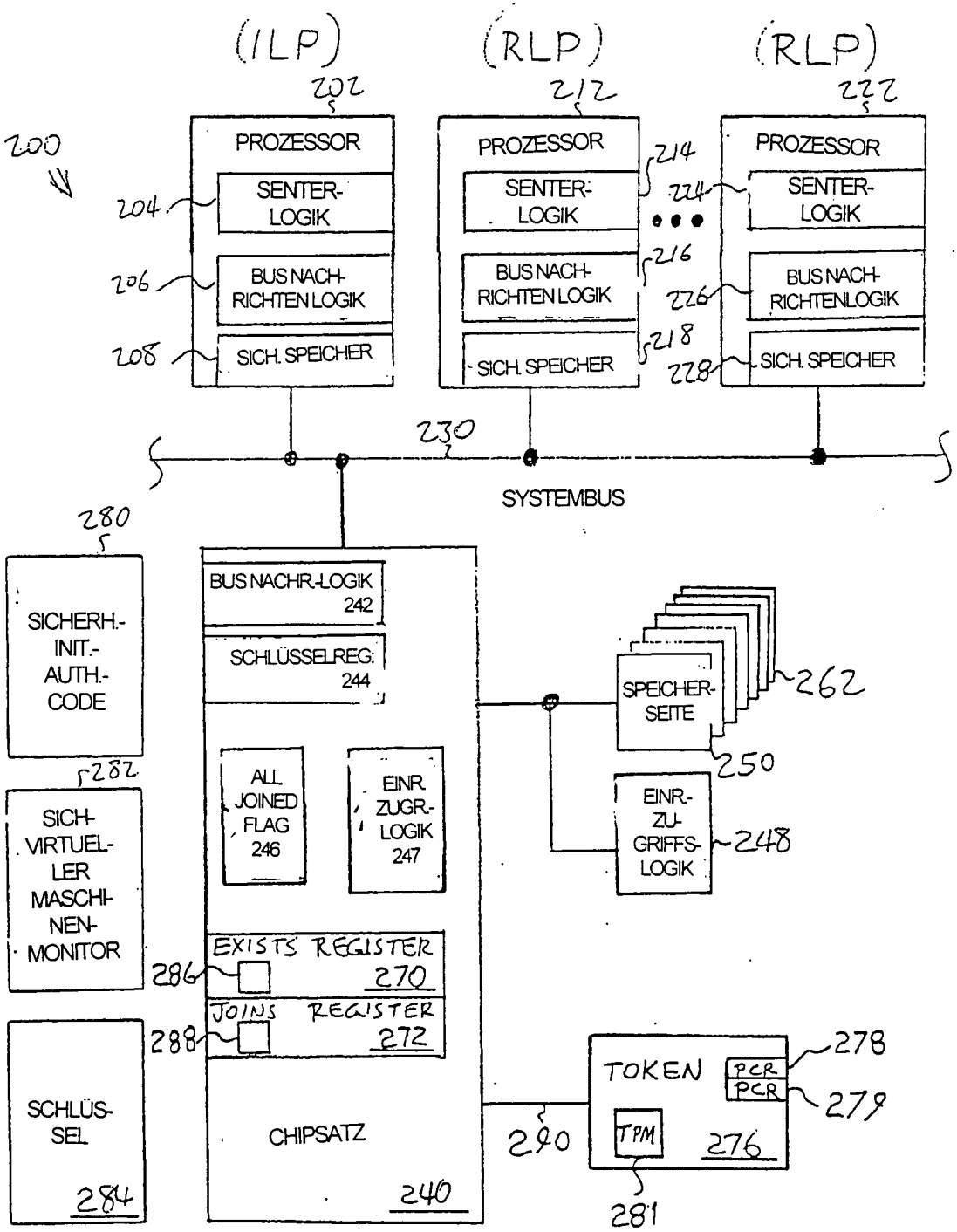

(ILP)
(RLP)
(RLP)
202
212
222
200
PROZESSOR
PROZESSOR
PROZESSOR
204
SENTER-LOGIK
SENTER-LOGIK
SENTER-LOGIK
206
BUS NACH-RICHTEN LOGIK
BUS NACH-RICHTEN LOGIK
BUS NACH-RICHTENLOGIK
208
SICH. SPEICHER
SICH. SPEICHER
SICH. SPEICHER
214
216
218
224
226
228
230
SYSTEMBUS
280
SICHERH.-INIT.-AUTH.-CODE
282
SICH-VIRTUELLER MASCHINEN-MONITOR
BUS NACHR-LOGIK 242
SCHLÜSSELREG. 244
ALL JOINED FLAG 246
EINR. ZUGR.-LOGIK 247
SPEICHER-SEITE
262
250
EINR.-ZU-GRIFFS-LOGIK
248
EXISTS REGISTER 270
286
JOINS REGISTER 272
288
SCHLÜSSEL 284
CHIPSATZ 240
TOKEN
PCR
PCR
278
279
290
TPM 276
281

Fig. 2

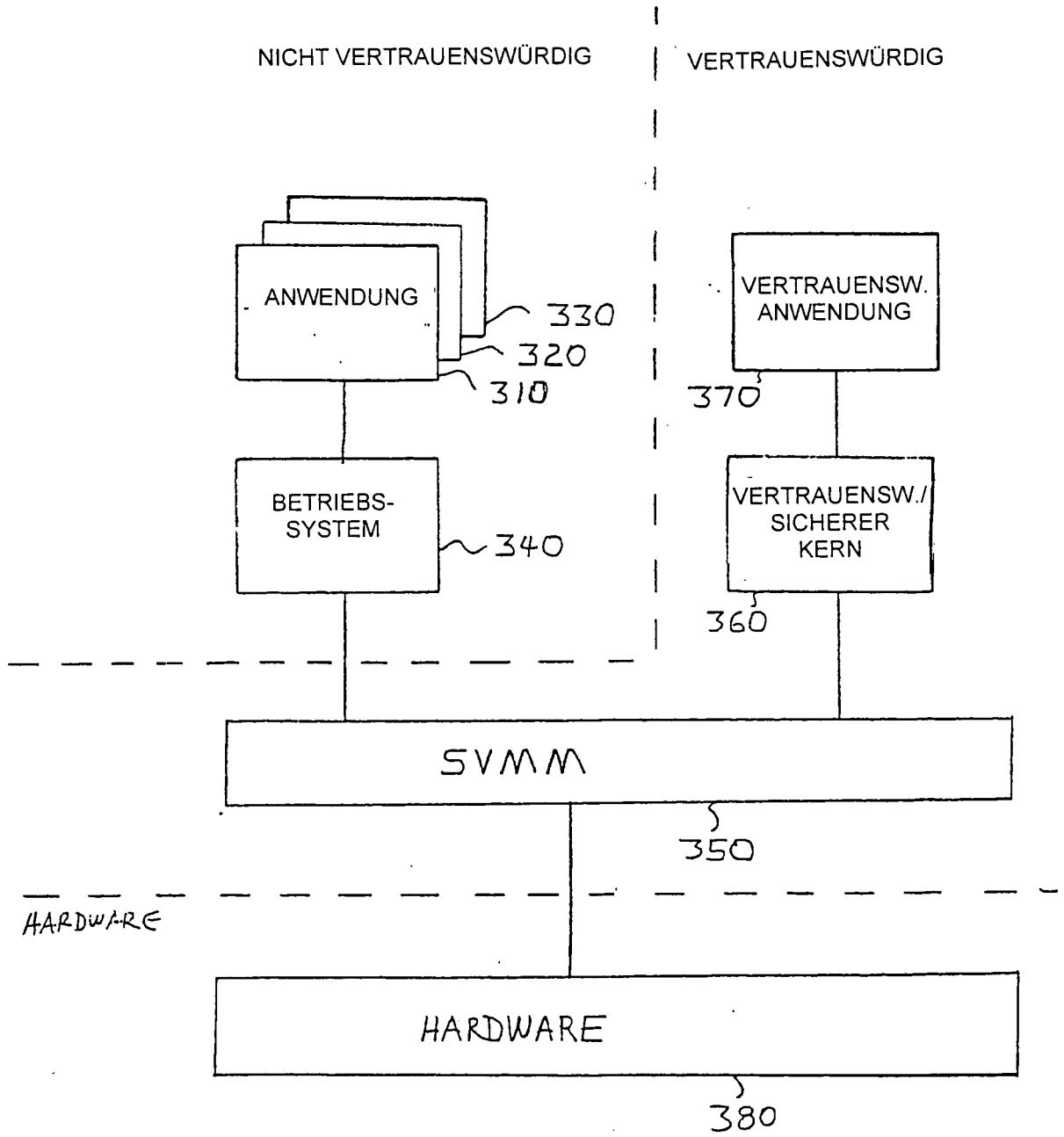

NICHT VERTRAUENSWÜRDIG
VERTRAUENSWÜRDIG
ANWENDUNG
330
320
310
VERTRAUENSW. ANWENDUNG
370
BETRIEBS-SYSTEM
340
VERTRAUENSW./ SICHERER KERN
360
SVMM
350
HARDWARE
HARDWARE
380

Fig. 3

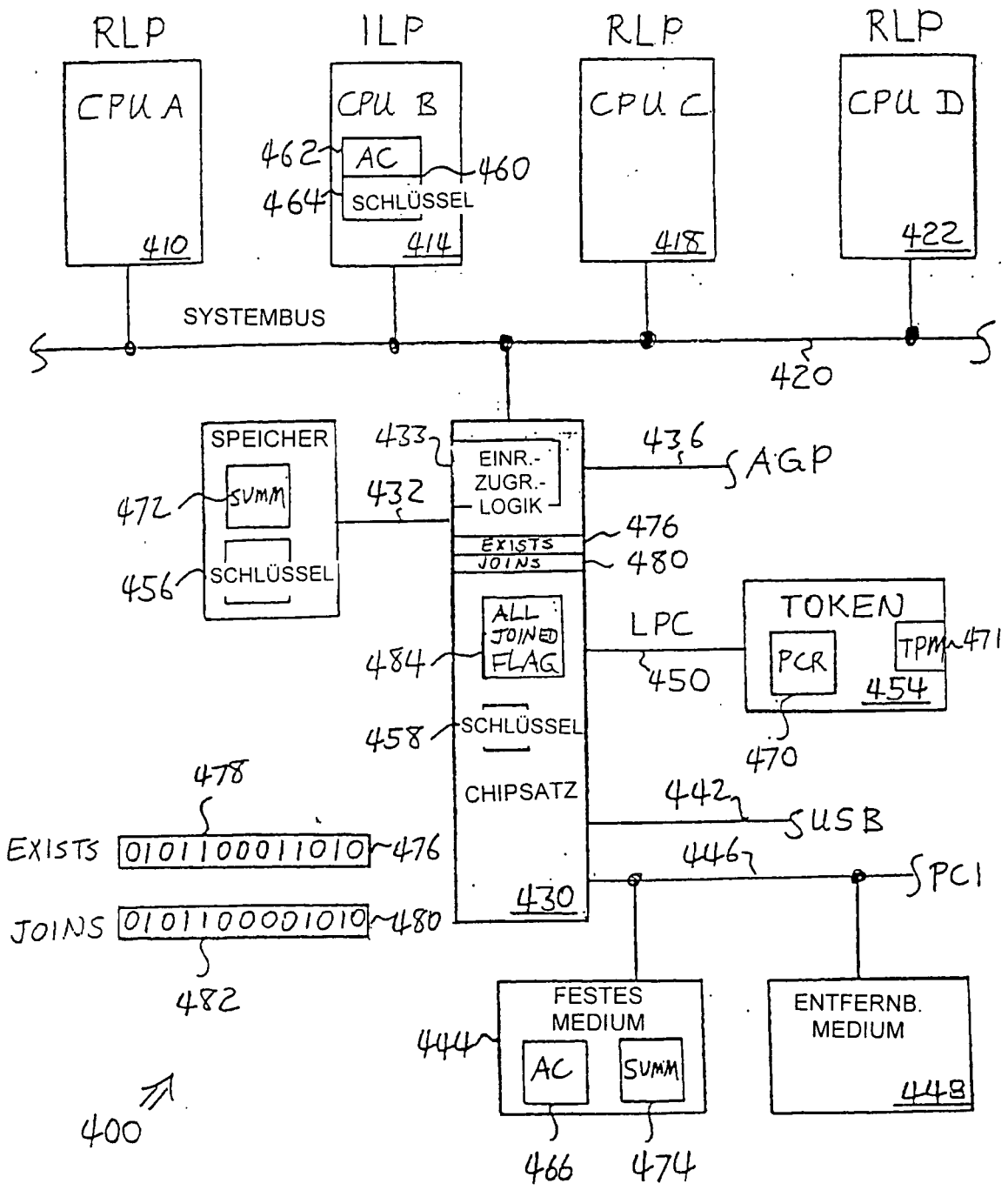

RLP
CPU A
410
ILP
CPU B
462
AC
460
464
SCHLÜSSEL
414
RLP
CPU C
418
RLP
CPU D
422
SYSTEMBUS
420
SPEICHER
472
SUMM
456
SCHLÜSSEL
432
433
EINR.-
ZUGR.-
LOGIK
436
AGP
476
EXISTS
480
JOINS
ALL
JOINED
FLAG
484
LPC
450
TOKEN
TPM
471
PCR
454
470
458
SCHLÜSSEL
CHIPSATZ
430
442
USB
446
PCI
478
EXISTS
0101100011010
476
JOINS
0101100001010
480
482
FESTES
MEDIUM
444
AC
SUMM
466
474
ENTFERNB.
MEDIUM
448
400

Fig. 4A

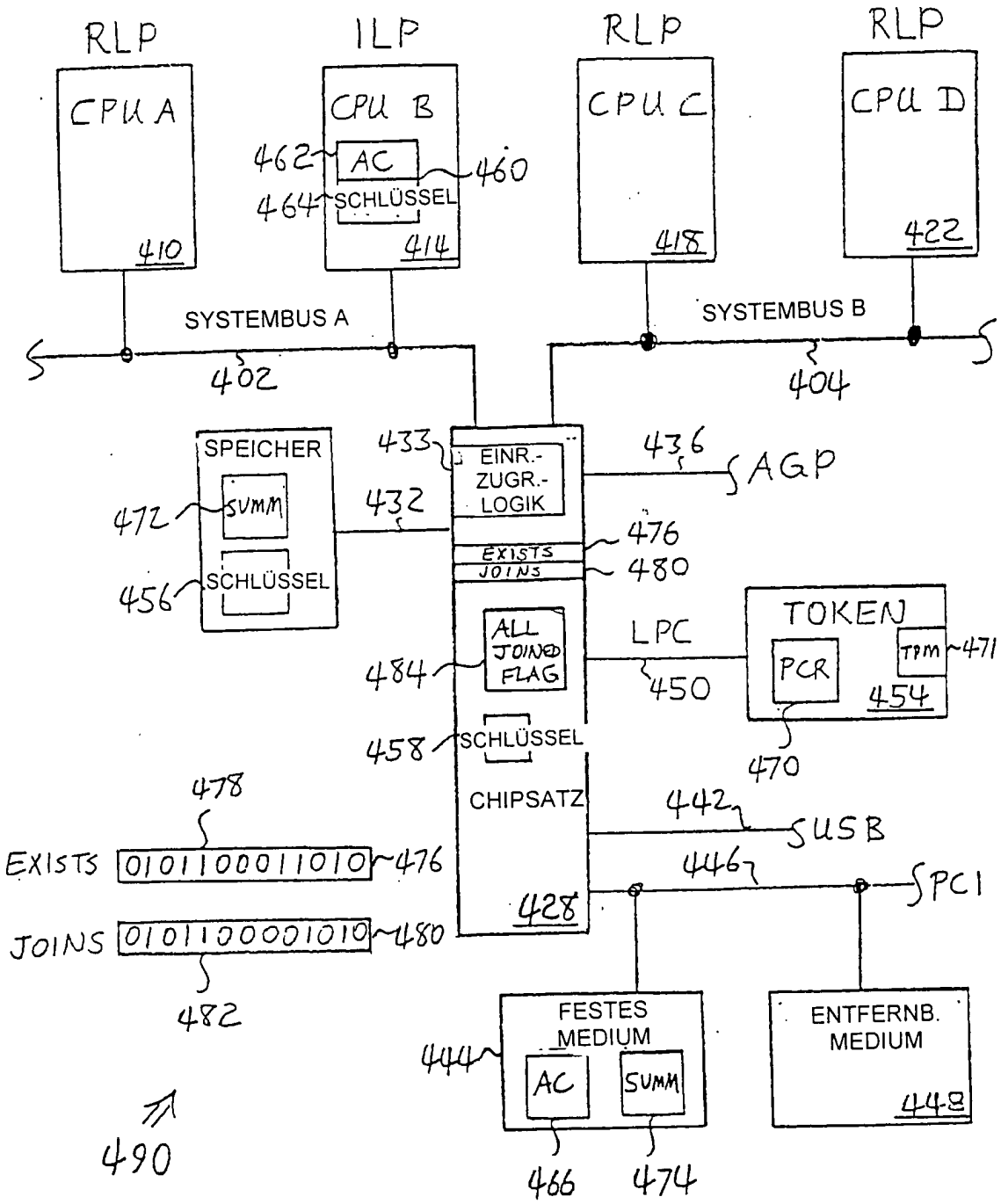

RLP
CPU A
410
ILP
CPU B
462
AC
460
464
SCHLÜSSEL
414
RLP
CPU C
418
RLP
CPU D
422
SYSTEMBUS A
402
SYSTEMBUS B
404
SPEICHER
472
SUMM
456
SCHLÜSSEL
432
433
EINR.-
ZUGR.-
LOGIK
436
AGP
476
EXISTS
480
JOINS
ALL
JOINED
FLAG
484
LPC
450
TOKEN
TPM
471
PCR
454
470
458
SCHLÜSSEL
CHIPSATZ
428
442
USB
446
PCI
478
EXISTS
0101100011010
476
JOINS
0101100001010
480
482
490
FESTES
MEDIUM
444
AC
SUMM
466
474
ENTFERNB.
MEDIUM
448

Fig. 4B

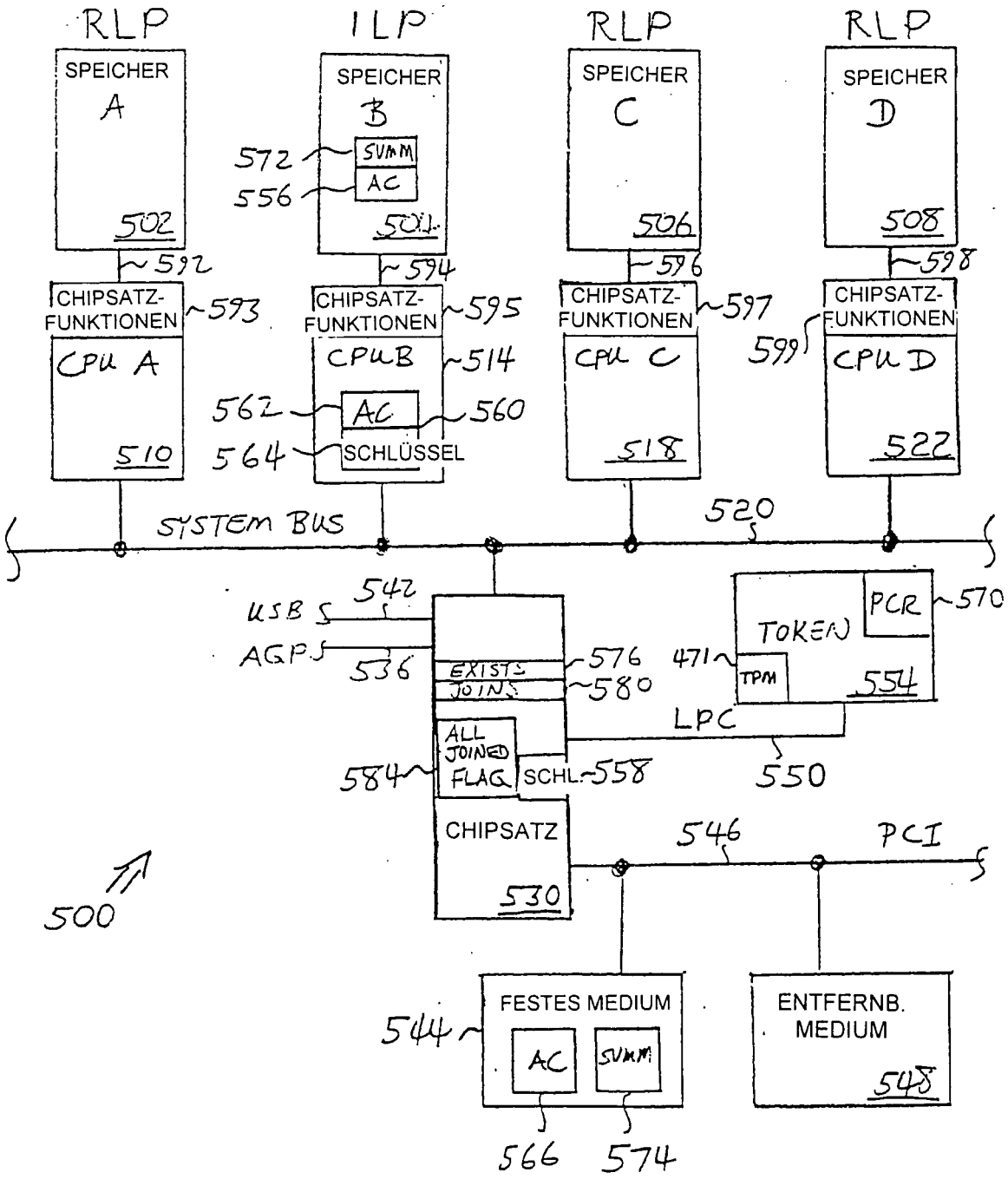

RLP
SPEICHER
A
502
592
CHIPSATZ-
FUNKTIONEN
593
CPU A
510
ILP
SPEICHER
B
572
SVMM
556
AC
504
594
CHIPSATZ-
FUNKTIONEN
595
CPU B
514
562
AC
560
564
SCHLÜSSEL
RLP
SPEICHER
C
506
596
CHIPSATZ-
FUNKTIONEN
597
CPU C
518
RLP
SPEICHER
D
508
598
CHIPSATZ-
FUNKTIONEN
599
CPU D
522
SYSTEM BUS
520
542
USB
AGP
536
EXISTS
JOINS
576
580
ALL
JOINED
FLAG
584
SCHL
558
CHIPSATZ
530
500
TOKEN
PCR
570
471
TPM
554
LPC
550
546
PCI
FESTES MEDIUM
544
AC
SVMM
566
574
ENTFERNB.
MEDIUM
548

Fig. 5

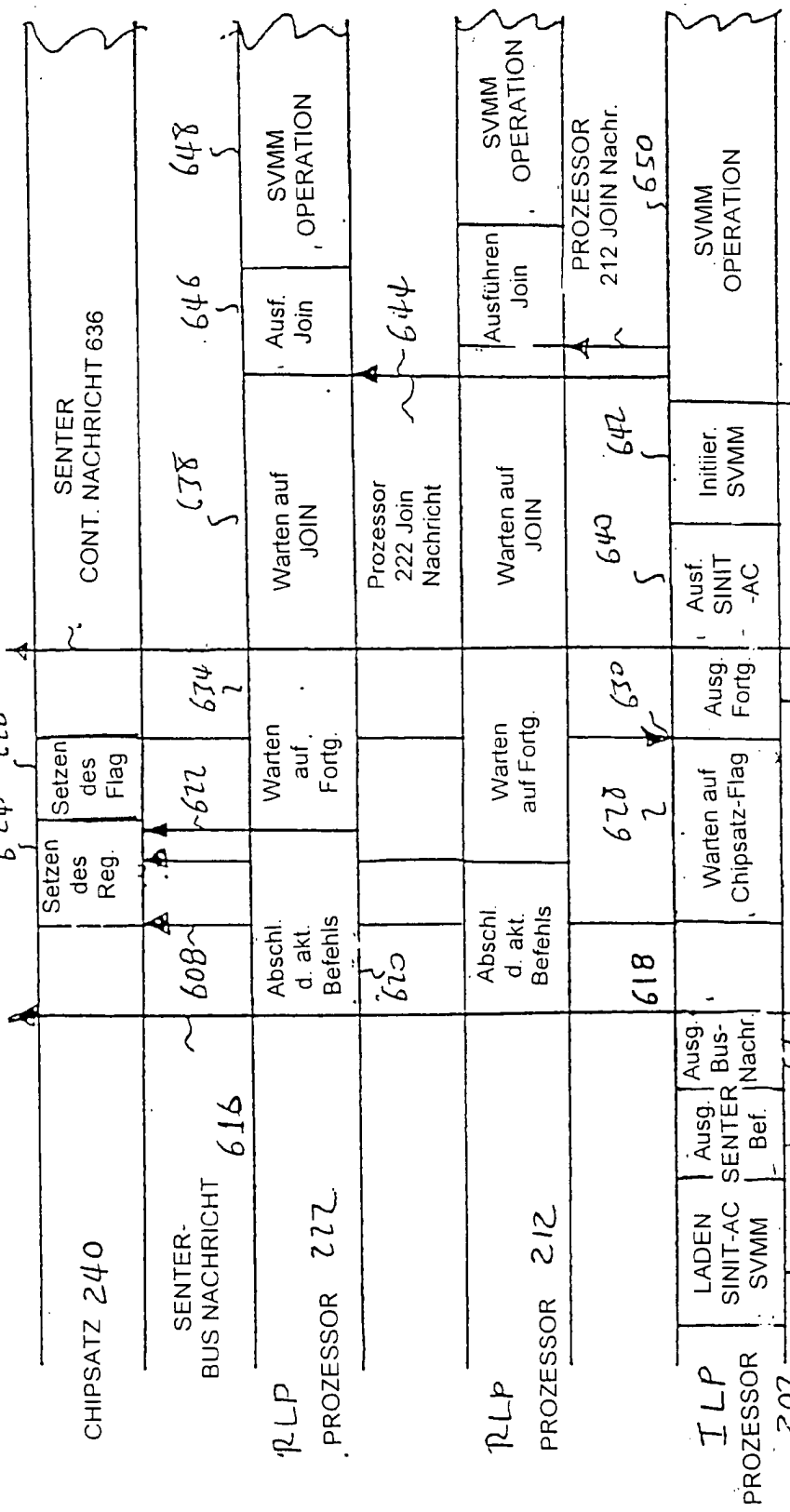

CHIPSATZ 240
SENTER-BUS NACHRICHT 616
RLP PROZESSOR 222
RLP PROZESSOR 212
ILP PROZESSOR 202
LADEN SINIT-AC SVMM
Ausg. SENTER Bef.
Ausg. Bus-Nachr.
Abschl. d. akt. Befehls
Setzen des Reg.
Setzen des Flag
Warten auf Fortg.
Warten auf Chipsatz-Flag
Ausg. Fortg.
Ausf. SINIT -AC
Initiier. SVMM
SENTER CONT. NACHRICHT 636
Warten auf JOIN
Prozessor 222 Join Nachricht
Ausf. Join
Ausführen Join
PROZESSOR 212 JOIN Nachr.
SVMM OPERATION
608
610
612
614
616
618
620
624
626
628
630
632
634
636
638
640
642
644
646
648
650

Fig. 6

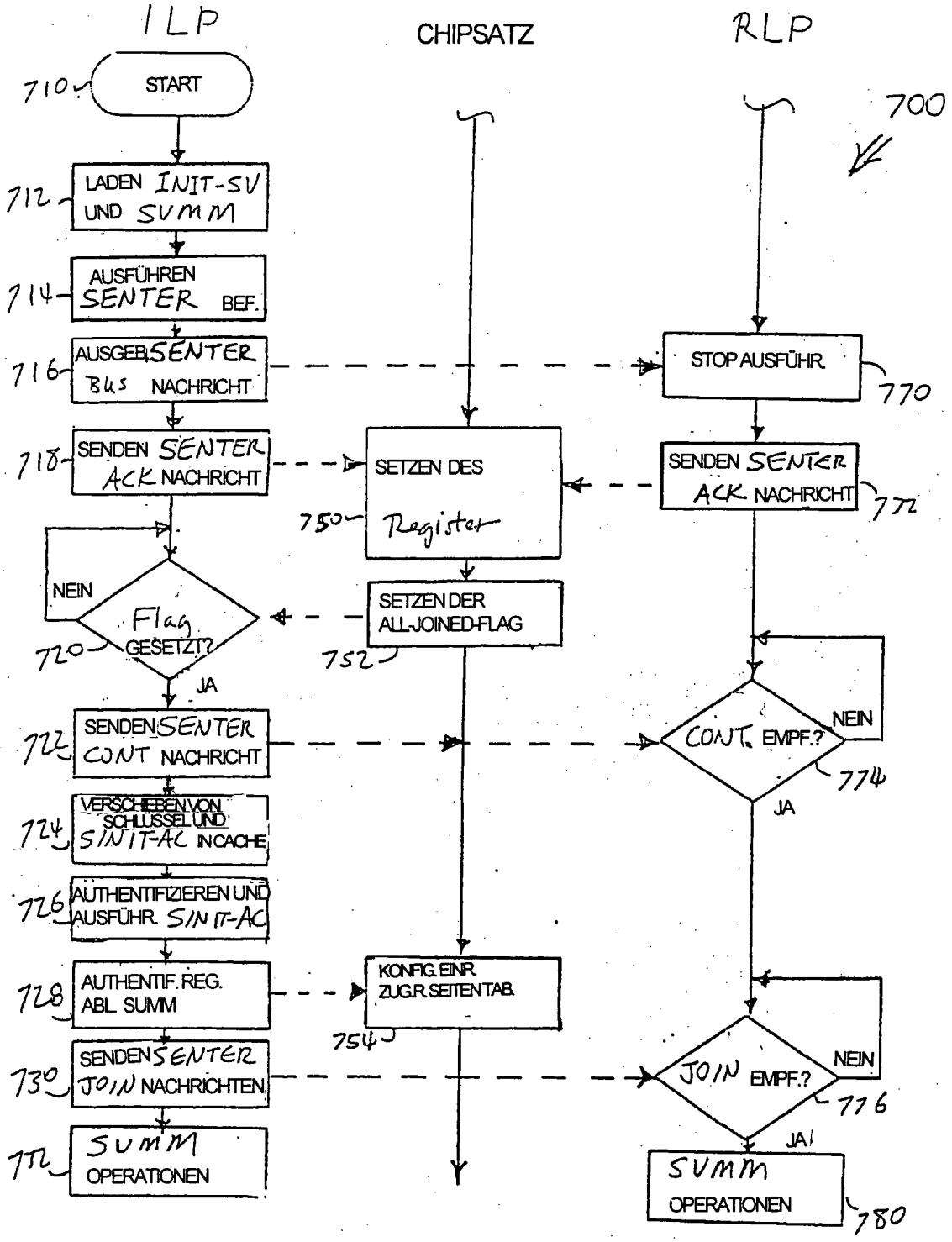

ILP
CHIPSATZ
RLP
700
START
710
LADEN INIT-SV UND SVMM
712
AUSFÜHREN SENTER BEF.
714
AUSGEB. SENTER BUS NACHRICHT
716
STOP AUSFÜHR.
770
SENDEN SENTER ACK NACHRICHT
718
SETZEN DES Register
750
SENDEN SENTER ACK NACHRICHT
772
NEIN
Flag GESETZT?
720
SETZEN DER ALL-JOINED-FLAG
752
JA
SENDEN SENTER CONT NACHRICHT
722
CONT. EMPF.?
774
NEIN
JA
VERSCHIEBEN VON SCHLÜSSEL UND SINIT-AC IN CACHE
724
AUTHENTIFIZIEREN UND AUSFÜHR. SINIT-AC
726
AUTHENTIF. REG. ABL. SUMM
728
KONFIG. EINR. ZUGR. SEITENTAB.
754
SENDEN SENTER JOIN NACHRICHTEN
730
JOIN EMPF.?
776
NEIN
JA
SVMM OPERATIONEN
732
SVMM OPERATIONEN
780

Fig. 7