(19) Deutsches Patent- und Markenamt

(10) **DE 10 2018 126 731 A1** 2019.06.27

(12) # Offenlegungsschrift

(21) Aktenzeichen: **10 2018 126 731.2**
(22) Anmeldetag: **26.10.2018**
(43) Offenlegungstag: **27.06.2019**

(51) Int Cl.: ***G06F 12/14*** (2006.01)

(30) Unionspriorität:
15/854,278 **26.12.2017** **US**

(71) Anmelder:
INTEL CORPORATION, Santa Clara, Calif., US

(74) Vertreter:
Samson & Partner Patentanwälte mbB, 80538 München, DE

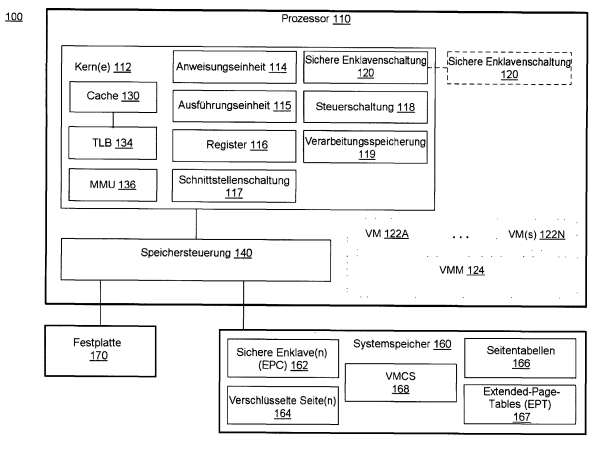
(72) Erfinder:
Rozas, Carlos, Portland, OR, US; Anati, Ittai, Ramat Hasharon, IL; McKeen, Francis, Portland, Oreg., US; Zmudzinski, Krystof, Forest Grove, OR, US; Alexandrovich, Ilya, Yokneam Illit, IL; Chakrabarti, Somnath, Portland, OR, US; Caspi, Dror, Kiryat Yam, IL; Ozsoy, Meltem, Hilsboro, OR, US

Prüfungsantrag gemäß § 44 PatG ist gestellt.

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen.

(54) Bezeichnung: **Freigabeanweisung, um Seitenblock während des Auslagerns umzukehren**

(57) Zusammenfassung: Eine sichere Enklavenschaltung speichert eine Enklave-Page-Cache-Map, um Inhalte einer sicheren Enklave in Systemspeicher zu verfolgen, der sichere Daten speichert, die eine Seite mit einer virtuellen Adresse umfassen. Eine Ausführungseinheit soll, als Reaktion auf eine Anforderung, die Seite von der sicheren Enklave zu räumen: die Erzeugung von Übersetzungen der virtuellen Adresse blocken; einen oder mehrere aktuell auf die sicheren Daten in der sicheren Enklave zugreifende Hardware-Threads aufzeichnen; einen Inter-Processor-Interrupt an einen oder mehrere dem einen oder den mehreren Hardware-Threads zugeordnete Kerne senden, um den einen oder die mehreren Hardware-Threads dazu zu veranlassen, die sichere Enklave zu verlassen und Translation-Lookaside-Buffer des einen oder der mehreren Kerne zu leeren; und als Reaktion auf das Erkennen eines der virtuellen Adresse für die Seite in der sicheren Enklave zugeordneten Seitenfehlers, die Erzeugung von Übersetzungen der virtuellen Adresse freigeben.



Beschreibung

Technisches Gebiet

[0001] Die Offenbarung bezieht sich auf sicheres Auslagern in eine und aus einer sicheren Enklave von Speicher, und insbesondere auf eine Freigabeanweisung, um einen Seitenblock während des sicheren Auslagerns umzukehren.

Hintergrund

[0002] Sicheres Auslagern, auch bezeichnet als Page-Swapping, verwendet in modernen Prozessoren Metadaten für eine Seite, die entfernt und in Systemspeicher oder auf Festplatte zurückgeschrieben (zum Beispiel ausgelagert) werden soll. Die Metadaten werden von einem Prozessor für Zugriffskontrolle und kryptographische Operationen, die sichere Seiten schützen, verwendet. Diese Metadaten sollen gespeichert werden, wenn die Seite von sicherem Speicher entfernt wird. Einige der Metadaten werden geheim gehalten, und einige von ihnen können sicher freigelegt, zum Beispiel außerhalb von sicherem Speicher gespeichert werden. Beide Sätze dieser Metadaten werden jedoch bei erneutem Laden der Seite von einer Festplatte verifiziert. Bei herkömmlichem sicherem Auslagern werden die Metadaten zwischen sicherem Speicher und unsicherem Speicher aufgeteilt, und somit müssen Auslagerungsprozesse auch diese Aufteilung in Metadaten verfolgen, was sowohl Zurückschreiboperationen (um eine Seite auf Festplatte zurückzuspeichern und die Metadaten zu speichern), als auch Ladeoperationen (was die Seite in Speicher wiederherstellt und Zugriff auf die gespeicherten Metadaten erneuert) erschwert. Zum Beispiel sollte Verarbeitungslogik, die Auslagerungsprozesse ausführt, berechnen, wo jedes Feld jedes Satzes von Metadaten ist (sowohl in dem sicheren, als auch in unsicherem Speicher), und wie groß jedes Feld jedes Satzes von Metadaten, auf den zugegriffen werden soll, ist.

[0003] Darüber hinaus treten während einer Zurückschreiboperation eine Reihe von Prozessortransaktionen auf, die das Speichern der Metadaten, das Verschlüsseln der Seitendaten und das Speichern der verschlüsselten Seite in Speicher und auf Festplatte beinhalten. Um die Seite wiederherzustellen, kopiert in den meisten Implementierungen der Prozessor externe Metadaten und die verschlüsselte Seite in sicheren Speicher. Der Prozessor führt eine authentifizierte Entschlüsselungsoperation aus, die sowohl die Metadaten, als auch die Seite authentifiziert, und verifiziert entweder die verschlüsselten Inhalte oder die entschlüsselten Inhalte. Diese Reihe von Prozessortransaktionen, die die Zurückschreib- und Ladeoperationen beinhalten, sind hinsichtlich Verarbeitungsressourcen und Taktzyklen teuer.

Figurenliste

Fig. 1A ist ein Blockdiagramm eines beispielhaften Systems, das verbessertes sicheres Auslagern ausführt, gemäß verschiedenen Implementierungen.

Fig. 1B ist ein Blockdiagramm einer beispielhaften sicheren Enklave innerhalb eines Systemspeichers des Systems von **Fig. 1A**, das die Verwendung einer Unified-Metadata-Structure (UMDS) beinhaltet, gemäß Implementierungen.

Fig. 2 ist ein Blockdiagramm eines Verfahrens für das Übersetzen virtueller Adressen für eine Seite letztendlich in eine physikalische Host-Adresse, gemäß Implementierungen.

Fig. 3 ist ein Blockdiagramm einer sicheren Enklavenschaltung des Prozessors von **Fig. 1A**, gemäß Implementierungen.

Fig. 4 ist ein Flussdiagramm eines Verfahrens für einen Auslagerungsfluss einer Seite in einen und aus einem sicheren Speicher, gemäß Implementierungen.

Fig. 5 ist ein Flussdiagramm eines Verfahrens für das Freigeben der Erzeugung weiterer Übersetzungen einer Seite nach Empfang eines der Seite zugeordneten Seitenfehlers, gemäß Implementierungen.

Die **Fig. 6A** und **Fig. 6B** sind ein Flussdiagramm eines Verfahrens für das Freigeben der Erzeugung weiterer Übersetzungen der Seite nach Empfang eines der Seite zugeordneten Seitenfehlers, gemäß einer alternativen Implementierung.

Fig. 7A ist ein Diagramm einer Unified-Metadata-Structure-Information (UMDSINFO) -Datenstruktur, gemäß Implementierungen.

Fig. 7B ist ein Diagramm eines Datenvektors, der ein Teil der UMDSINFO-Datenstruktur ist, gemäß Implementierungen.

Fig. 8 ist ein Diagramm einer erweiterten Versions-Array-Seite, gemäß Implementierungen.

Fig. 9 ist ein Diagramm einer Unified-Metadata-Structure (UMDS), gemäß Implementierungen.

Fig. 10 ist ein Flussdiagramm eines Verfahrens für das Weiterleiten eines geblockten Bits zwischen der UMDSINFO und Flags der UMDS während eines Zurückschreibens, auf das später während eines Ladens verwiesen werden kann, um eine Seite für sicheren Speicher wiederherzustellen, gemäß Implementierungen.

Fig. 11 ist ein Flussdiagramm eines Verfahrens für die Erzeugung einer Versions-Array-Seite vor sicheren Auslagerungsoperationen, gemäß Implementierungen.

Fig. 12A ist ein Blockdiagramm zum Veranschaulichen sicherer Enklaven und Systemspeicherdatenstrukturen während eines sicheren Zurückschreibens, gemäß Implementierungen.

Fig. 12B ist ein Flussdiagramm eines Verfahrens für das Ausführen eines sicheren Zurückschreibens unter Bezugnahme auf die Datenstrukturen von **Fig. 12A**, gemäß einer Implementierung.

Die **Fig. 13A**, **Fig. 13B** und **Fig. 13C** sind ein Flussdiagramm eines Verfahrens für das Ausführen eines sicheren Zurückschreibens, gemäß verschiedenen anderen Implementierungen.

Fig. 14A ist ein Blockdiagramm zum Veranschaulichen sicherer Enklaven und Systemspeicherdatenstrukturen während eines sicheren Ladens, gemäß Implementierungen.

Fig. 14B ist ein Flussdiagramm eines Verfahrens für das Ausführen eines sicheren Ladens unter Bezugnahme auf die Datenstrukturen von **Fig. 14A**, gemäß einer Implementierung.

Die **Fig. 15A** und **Fig. 15B** sind ein Flussdiagramm eines Verfahrens für das Ausführen eines sicheren Ladens, gemäß verschiedenen anderen Implementierungen.

Fig. 16A ist ein Blockdiagramm zum Veranschaulichen einer Mikroarchitektur für einen Prozessor, der verbessertes sicheres Auslagern implementiert, gemäß Implementierungen.

Fig. 16B ist ein Blockdiagramm zum Veranschaulichen einer In-Order-Pipeline und einer Registerumbenennungsstufe, Out-of-Order-Ausgabe-/Ausführungs-Pipeline, die durch den Prozessor von **Fig. 5A** implementiert wird, der verbessertes sicheres Auslagern implementiert, gemäß Implementierungen.

Fig. 17 veranschaulicht ein Blockdiagramm der Mikroarchitektur für einen Prozessor oder eine integrierte Schaltung, der/die verbessertes sicheres Auslagern implementiert, gemäß einer Implementierung.

Fig. 18 ist ein Blockdiagramm eines Computersystems, das Hardware-Unterstützung für verbessertes sicheres Auslagern implementiert, gemäß einer Implementierung

Fig. 19 ist ein Blockdiagramm eines Computersystems, das Hardware-Unterstützung für verbessertes sicheres Auslagern implementiert, gemäß einer anderen Implementierung.

Fig. 20 ist ein Blockdiagramm eines System-on-a-Chip, das verbessertes sicheres Auslagern implementiert, gemäß einer Implementierung.

Fig. 21 veranschaulicht ein Blockdiagramm für ein Computersystem, das verbessertes sicheres Auslagern implementiert, gemäß einer Implementierung.

Fig. 22 veranschaulicht ein Blockdiagramm für ein Computersystem, das Hardware-Unterstützung für verbessertes sicheres Auslagern implementiert, gemäß einer anderen Implementierung.

Ausführliche Beschreibung

[0004] Implementierungen der Offenbarung beschreiben eine Freigabeanweisung, um einen Seitenblock während sicheren Auslagerns umzukehren In einer Implementierung kann, nachdem eine Seite in sicherem Speicher für eine Zeitdauer inaktiv (zum Beispiel ohne Zugriff) war, Verarbeitungslogik (zum Beispiel innerhalb eines Verarbeitungskerns) bestimmte Operationen in Vorbereitung, um die Seite von dem sicheren Speicher zu entfernen und sie auf Festplatte zurückzuschreiben (zum Beispiel auszulagern), ausführen. Sicherer Speicher ist geschützter Speicher und kann hierin als Enklave-Page-Cache (EPC), eine sichere Enklave oder „Enklave“ bezeichnet werden. Die Seiten innerhalb der Enklave können als EPC-Seiten bezeichnet werden. Eine Enklave kann bestimmte Adressbereiche von Systemspeicher als sicher beschreiben, und daher erfolgt Interaktion mit diesem Adressbereich durch sichere Anweisungen (auch Software-Guard-Instructions oder SGX genannt, von Intel® Corporation in Santa Clara, Kalifornien). Wenn eine Seite ausgelagert wird, werden der Seite zugeordnete Metadaten für spätere Verwendung gespeichert (nachdem die Seite in sicheren Speicher wiederhergestellt wird), und diese verschlüsselten Daten für die Seite werden in regulären Systemspeicher (außer-

halb der sicheren Enklave) oder auf einer Festplatte (zum Beispiel Hard-Disk-Drive (HDD), Solid-State-Drive (SSD) oder einer anderen permanenten Speicherung) zurückgespeichert. Die Verarbeitungslogik kann eine Enklavenzurückschreibanweisung (zum Beispiel EWB-Anweisung) ausführen, um die Daten zu verschlüsseln, die verschlüsselten Daten in Systemspeicher (außerhalb der sicheren Enklave) zu schreiben, und auch um die verschlüsselten Daten auf Festplatte zurückzuschreiben.

[0005] In Vorbereitung für die Ausführung der EWB-Anweisung kann die Verarbeitungslogik zuerst weitere Übersetzungen der virtuellen Adresse für Zugriff auf die Seite blocken, zum Beispiel durch Ausführen einer Enklavenblockanweisung (zum Beispiel EBLOCK-Anweisung). Zum Beispiel kann die Verarbeitungslogik die Seite (zum Beispiel eine virtuelle Adresse für die Seite) als nicht in Übersetzungstabellen und erweiterten Übersetzungstabellen vorhanden markieren, den Ablauf einer Zeitdauer, während derer nicht auf die Seite zugegriffen wird, erkennen, und ein „Block“-Bit in einer Enklave-Page-Cache-Map (EPCM) für die zu der virtuellen Adresse abgebildete EPC-Seite setzen, um Erzeugung der weiteren Übersetzungen der virtuellen Adresse zu blocken. Die Verarbeitungslogik kann auch einen oder mehrere Hardware-Threads aufzeichnen, die aktuell auf die sicheren Daten in der sicheren Enklave zugreifen (zum Beispiel eine ETRACK-Anweisung ausführen), und einen Inter-Processor-Interrupt (IPI) an einen oder mehrere dem einen oder den mehreren Hardware-Threads zugeordnete Kerne senden, um den einen oder die mehreren Hardware-Threads dazu zu veranlassen, die sichere Enklave zu verlassen und Translation-Lookaside-Buffer (TLB) des einen oder der mehreren Kerne zu leeren.

[0006] Nach dieser Reihe von Vorbereitungen, um die EWB-Anweisung auszuführen, ist es möglich, dass die Verarbeitungslogik einen Seitenfehler erkennt, der anzeigt, dass ein System-Agent oder eine Anwendungskomponente Zugriff auf die Seite in der Enklave versucht hat. Da die Seite jedoch geblockt worden ist und Einträge in den TLB geleert wurden, kann es keinen Zugriff auf die Seite geben. Üblicherweise würde, als Reaktion auf die Erkennung eines der virtuellen Adresse für die Seite in der sicheren Enklave zugeordneten Seitenfehlers, die Verarbeitungslogik mit Ausführung der EWB-Anweisung fortfahren, optional die verschlüsselten Daten auf Festplatte schreiben, optional die verschlüsselten Daten von Festplatte lesen, und eine Enklavenladeanweisung (zum Beispiel ELD-Anweisung) ausführen, um die Seite in den sicheren Speicher wiederherzustellen. Jedoch verursacht dies erheblichen Mehraufwand, der viele Verarbeitungsressourcen verbraucht, und somit die Leistung von sicherer Speicheroperation in Fällen, wo ein Seitenfehler erkannt wird, bevor eine EWB-Anweisung ausgeführt wird, herabsetzt. Die hierin offenbarten Implementierungen stellen Wege bereit, um die Erzeugung der weiteren Übersetzung der virtuellen Adresse freizugeben, um mit Seitenzugriff ohne das Ausführen der EWB- und ELD-Anweisungen fortzufahren.

[0007] Zum Beispiel kann die Verarbeitungslogik eine Enklavenfreigabeanweisung (zum Beispiel EUNBLOCK-Anweisung) ausführen, um das Bit in der EPCM für die virtuelle Adresse zu löschen und die Seite als in den Übersetzungstabellen vorhanden zu markieren. Dies kann Architekturstatus für neue Übersetzungseinträge in den TLB und erneuerten Zugriff auf die an der virtuellen Seitenadresse in dem sicheren Speicher gespeicherten Daten setzen. Ausführung der EUNBLOCK-Anweisung kann die für die Ausführung der Anweisung verwendeten Prozessortaktzyklen im Vergleich zu dem vorherigen Ansatz, bei dem die EWB-Anweisung ausgeführt wird, um ein Zurückschreiben auszuführen, gefolgt von einer ELD-Anweisung, um die Seite von dem Systemspeicher (oder der Festplatte) zu dem sicheren Speicher wiederherzustellen, erheblich reduzieren.

[0008] In einigen Implementierungen wird der Seitenfehler nach dem Start der EWB-Anweisungsausführung empfangen, in diesem Fall müssen die EWB- und ELD-Anweisungen abgeschlossen werden, um Zugriff auf die Seite wiederherzustellen. Bei herkömmlichem sicherem Auslagern werden die einer Seite zugeordneten Metadaten zwischen sicherem Speicher und unsicherem Speicher aufgeteilt, und somit müssen Auslagerungsprozesse wie EWB- und ELD-Anweisungen auch diese Aufteilung in Metadaten verfolgen, was sowohl Zurückschreiboperationen (um eine Seite auf Festplatte zurückzuspeichern und die Metadaten zu speichern), als auch Ladeoperationen (was die Seite in Speicher wiederherstellt und Zugriff auf die gespeicherten Metadaten erneuert) erschwert. Zum Beispiel sollte Verarbeitungslogik, die die Auslagerungsprozesse ausführt, berechnen, wo jedes Feld jedes Satzes von Metadaten ist (sowohl in dem sicheren, als auch in unsicherem Speicher), und wie groß jedes Feld jedes Satzes von Metadaten, auf den zugegriffen werden soll, ist.

[0009] Die offenbarten Implementierungen können eine Unified-Metadata-Structure (UMDS) einsetzen, um alle sich auf eine EPC-Seite beziehenden Metadaten zu speichern, einschließlich der Versions- und Message-Authentication-Code (**MAC**) -Werte, die unten detaillierter diskutiert werden. Die UMDS kann in einem Slot einer Extended-Version-Array (EVA) -Seite des sicheren Speichers gespeichert werden. Die EVA-Seite kann an einer bestimmten Adresse in sicherem Speicher gespeichert werden, sodass die Metadaten nicht an mehreren Plätzen gespeichert oder von dort abgerufen werden müssen, einer innerhalb des sicheren Speichers und

einer außerhalb sicheren Speichers. Auf diese Weise wird die Ausführung der EWB- und ELD-Anweisungen gestrafft, wodurch die Notwendigkeit des Zugriffs auf mehrere Adressplätze mit flexiblen Mengen von Metadaten, jedoch mit einem einzelnen Zugriff auf einen Slot innerhalb der EVA-Seite, die die UMDS speichert, vermieden wird.

[0010] Als ein weiterer Vorteil der offenbarten UMDS-basierten Architektur kann ein virtueller Maschinenmonitor (VMM) (zum Beispiel ein Hypervisor) die EBLOCK-Anweisung ausführen, um ein Blockstatusbit zu erzeugen, um einen geblockten Status einer aus dem sicheren Speicher auszulagernden Seite anzuzeigen . Dieses Blockstatusbit kann Passage des in der EPCM (oben diskutiert) gesetzten geblockten Bits zu einer UMDS-Informations (UMDSINFO) -Datenstruktur oder innerhalb der UMDS selbst, von denen beide unten detaillierter diskutiert werden, auslösen. Einfügen des geblockten Bits in eine UMDS-bezogene Struktur kann es den EWB- und den ELD-Anweisungen erlauben, auf das geblockte Bit zu verweisen, um während des sicheren Auslagerns den geblockten Status der Seite (zum Beispiel, ob sich die Seite in einem GEBLOCKTEN Status befindet oder nicht) zu verfolgen. Dies kann es einer sicheren Ladeoperation erlauben, das geblockte Bit in der EPCM in dem richtigen Status wiederherzustellen, zusätzlich zu anderen ladebezogenen Operationen bei dem Zurückladen der Seite in die Enklave. Diese und andere auf die Verwendung der UMDS-Architektur bezogene Operationen straffen die sichere Auslagerungsarchitektur und reduzieren somit Mehraufwand und Komplexität. Als nur ein Beispiel kann die erweiterte ELD (zum Beispiel ELDE) -Anweisung jetzt eine einzelne Anweisung sein, im Gegensatz zu herkömmlichen Systemen, wo eine andere ELD-Anweisung ausgeführt werden sollte, abhängig davon, ob die Seite geblockt (ELDB) oder freigegeben wurde (ELDU).

[0011] **Fig. 1A** ist ein Blockdiagramm von beispielhaftem System **100**, das verbessertes sicheres Auslagern ausführt, gemäß verschiedenen Implementierungen. Das System **100** beinhaltet Prozessor **110**, Systemspeicher **160** und Festplatte **170**. Der Prozessor **110** kann mehrere Kerne **112**, Speichersteuerung **140** und sichere Enklavenschaltung **120**, die optional innerhalb oder außerhalb der Kerne **112** des Prozessors **110** positioniert ist, beinhalten, aber nicht darauf beschränkt sein. Der Prozessor **110** kann zum Beispiel durch mindestens einen der mehreren Kerne **112** virtuellen Maschinenmonitor (VMM) **124** ausführen, der dazu verwendet werden kann, mehrere virtuelle Maschinen **122A** ... **122N** zu erzeugen, zu steuern und zu verwalten. Die Speichersteuerung **140** kann dazu verwendet werden, sich mit der Computerspeicherung **170** und dem Systemspeicher **160** zu verbinden und das Speichern von Daten zu und das Wiederabrufen von Daten von der Computerspeicherung **170** und dem Systemspeicher **160** zu steuern.

[0012] Das System **100** kann einen Typ eines Informationsverarbeitungssystems, wie zum Beispiel einen Server, einen Desktop-Computer, einen tragbaren Computer, eine Set-Top-Box, ein Handgerät oder ein eingebettetes Steuersystem, darstellen. Systeme zum Verkörpern der vorliegenden Implementierungen können jegliche Anzahl von jeder dieser Komponenten und jeglicher anderer Komponenten oder anderer Elemente, wie zum Beispiel Informationsspeicherungsgeräte, Peripheriegeräte und Eingabe-/Ausgabegeräte, beinhalten. Jegliche oder alle Komponenten oder andere Elemente in dieser oder jeglichen Systemausführungsform können miteinander durch jegliche Anzahl von Bussen, Punkt-zu-Punkt- oder anderen drahtgebundenen oder drahtlosen Schnittstellen oder Verbindungen verbunden sein, gekoppelt sein, oder auf andere Weise miteinander in Kommunikation stehen, sofern nicht anders spezifiziert.

[0013] Der Prozessor **110** kann einen oder mehrere auf einem einzelnen Substrat integrierte oder in einem einzelnen Package gepackte Prozessoren darstellen, von denen jeder mehrere Threads und/oder mehrere Ausführungskerne in jeglicher Kombination beinhalten kann. Jeder als Prozessor **110** dargestellte Prozessor kann jeglicher Typ von Prozessor sein, einschließlich eines Allzweck-Mikroprozessors, wie zum Beispiel eines Prozessors in der Intel® Core®-Prozessorfamilie, der Intel® Atom®-Prozessorfamilie oder einer anderen Prozessorfamilie der Intel® Corporation oder eines anderen Prozessors einer anderen Firma oder eines Spezialprozessors oder Mikrocontrollers.

[0014] Der Systemspeicher **160** kann Dynamic-Random-Access-Memory (DRAM) oder anderer Typ von Medium sein, das durch den Prozessor **110** gelesen werden kann. Der Systemspeicher **160** kann mehrere sichere Enklaven **162**, was hierin auch als Enclave-Page-Cache (EPC) bezeichnet wird, außerhalb der sicheren Enklaven **162** gespeicherte verschlüsselte Seiten **164**, zum Ausführen von Adressübersetzungen verwendete Seitentabellen **166** und erweiterte Seitentabellen (EPT) **167**, und Virtual-Machine-Control-Structure (VMCS) **168** beinhalten, aber nicht darauf beschränkt sein. Die VMCS **168** kann Zeiger auf die Seitentabellen **166** und die EPT **167** enthalten, und kann für die Interaktion mit und die Steuerung von einer der virtuellen Maschinen **122A** ... **122N** erzeugt werden. Die Festplatte **170** kann jeglichen Typ von beständigem oder nichtflüchtigem Speicher oder einer Speicherung beinhalten, wie zum Beispiel einen Flash-Speicher und/oder ein HDD-, SSD-, magnetisches oder optisches Festplattenlaufwerk.

[0015] Der Prozessor **110** kann ferner Anweisungseinheit **114**, Ausführungseinheit **115**, mehrere Register **116**, Schnittstellenschaltung **117**, Steuerschaltung **118**, Verarbeitungsspeicherung **119**, die sichere Enklavenschaltung **120**, Cache **130**, Translation-Lookaside-Buffer (TLB) **134** und Memory-Management-Unit (MMU) **136** beinhalten. Die Anweisungseinheit **114** kann eine Schaltung, Struktur oder andere Hardware, wie zum Beispiel einen Anweisungsdecoder für das Abrufen, Empfangen, Decodieren und/oder Planen von Anweisungen darstellen. Jegliches Anweisungsformat kann innerhalb des Umfangs der offenbarten Implementierungen verwendet werden; zum Beispiel kann eine Anweisung einen Opcode und einen oder mehrere Operanden beinhalten, wobei der Opcode in eine oder mehrere Mikroanweisungen oder Mikrooperationen für die Ausführung durch die Ausführungseinheit **115** decodiert werden kann. Die Ausführungseinheit **112** kann eine Schaltung, Struktur oder andere Hardware, wie zum Beispiel eine Arithmetikeinheit, Logikeinheit, Fließkommaeinheit, einen Verschieber, usw. beinhalten, um Daten zu verarbeiten und Anweisungen, Mikroanweisungen und/oder Mikrooperationen auszuführen.

[0016] Die Verarbeitungsspeicherung **119** kann einen Typ von Speicherung darstellen, der für einen lokalen Zweck innerhalb des Prozessors **110** verwendbar ist; zum Beispiel kann die Verarbeitungsspeicherung **119** Flash-Speicher, Static-Random-Access-Memory (SRAM) oder andere flüchtige Speicher oder Schnellzugriffsspeicherungsstrukturen beinhalten. Die Register **116** können zusätzliche lokale Speicherung für die Verwendung durch Architekturanweisungen bereitstellen, einschließlich der offenbarten sicheren Anweisungen, wie zum Beispiel General-Purpose-Register (GPR), Special-Purpose-Register (SPR) und Segmentregister, die in der Instruction-Set-Architecture (ISA) in der Intel® Core®-Prozessorfamilie oder einer anderen Prozessorfamilie der Intel® Corporation verwendet werden. Die Register **116** können ferner Datenregister, Anweisungsregister, Statusregister, Konfigurationsregister, Steuerregister oder andere programmierbare oder hartcodierte Register oder Registerdateien sein.

[0017] Die Schnittstellenschaltung **117** kann Hardware oder andere Strukturen beinhalten, wie zum Beispiel einen Bus, eine Nachrichtenschaltung oder andere Schaltung, einen Port oder eine Schnittstelle, um es dem Prozessor **110** zu erlauben, mit anderen Komponenten in dem System **100** durch jeglichen Typ von Bus, Punkt-zu-Punkt- oder andere Verbindung, direkt oder durch jegliche andere Komponente, wie zum Beispiel eine Speichersteuerung oder eine Busbrücke, zu kommunizieren.

[0018] Die Steuerschaltung **118** kann Logik, Mikrocode, Schaltung oder andere Hardware beinhalten, um die Operation der Einheiten und anderer Elemente des Prozessors **110** und die Übertragung von Daten innerhalb, in den und aus dem Prozessor **110** zu steuern. Die Steuerschaltung **118** kann den Prozessor **110** dazu veranlassen, Verfahren oder Prozesse, die sich auf die offenbarten Implementierungen beziehen, auszuführen oder daran teilzunehmen, zum Beispiel, indem er den Prozessor **110** dazu veranlasst, durch die Anweisungseinheit **114** empfangene Anweisungen und Mikroanweisungen oder Mikrooperationen, die von Anweisungen abgeleitet werden, die von der Anweisungseinheit **114** empfangen werden, auszuführen.

[0019] Der Cache **130** kann eine oder mehrere Ebenen von Cache-Speicher (zum Beispiel **L1**, **L2**, **L3**, **LLC**, ...) in einer Speicherhierarchie des Informationsverarbeitungssystems **100**, das in Static-Random-Access-Memory oder jeglicher anderen Speichertechnologie implementiert ist, darstellen. Der Cache **130** kann jegliche Kombination von Cache-Speichern, die einem oder mehreren der Kerne **112** innerhalb des Prozessors **110** gemäß verschiedenen Caching-Techniken zugeordnet sind oder zwischen diesen geteilt werden, beinhalten. Zum Beispiel kann einiger Cache für Daten sein, und anderer Cache kann Anweisungen sein.

[0020] Die TLB **134** können ebenfalls innerhalb mehrerer Ebenen von TLB existieren und können einige TLB für Anweisungen und separate TLB für Daten enthalten. Die TLB **134** können Adressübersetzungen für virtuelle Adressen von Seiten, die in der (den) sicheren Enklave(n) 162 ansässig sind, zwischenspeichern. Die Fähigkeit, Adressübersetzungen zwischenzuspeichern, spart, wie offensichtlich wird, erhebliche Verarbeitungsressourcen, die dazu erforderlich sind, jegliche gegebene virtuelle Adresse wiederholt zu übersetzen.

[0021] Die sichere Enklavenschaltung **120** kann Logik, Schaltungen, Hardware oder andere Strukturen für das Erzeugen und Aufrechterhalten einer gesicherten, geschützten oder isolierten Umgebung darstellen, wie zum Beispiel einer wie hierin beschriebenen sicheren Enklave, in der eine Anwendung oder andere Software laufen kann, ausgeführt werden kann, geladen werden kann, oder anderweitig in einem Informationsverarbeitungssystem, wie zum Beispiel dem System **100**, vorhanden sein kann. Für Zwecke dieser Beschreibung kann jede Instanz einer solchen Umgebung als sichere Enklave bezeichnet werden, obwohl offenbarte Implementierungen nicht auf diejenigen beschränkt sind, die eine sichere Enklave als die gesicherte, geschützte oder isolierte Umgebung verwenden. In einer Ausführungsform kann eine sichere Enklave durch Verwenden von Anweisungen in der ISA eines Prozessors erzeugt und aufrechterhalten werden. Die sichere Enklavenschal-

tung **120** wird unter Bezugnahme auf **Fig. 3** detailliert diskutiert. Der Prozessor **110** kann auch andere nicht in **Fig. 1** gezeigte Schaltungen, Strukturen oder Logik, und/oder Schaltungen, Strukturen oder Logik, die anderswo in den Figuren gezeigt oder beschrieben sind, beinhalten.

[0022] **Fig. 1B** ist ein Blockdiagramm einer beispielhaften sicheren Enklave **162** innerhalb des Systemspeichers **160** des Systems von **Fig. 1A**, gemäß einer Implementierung. Die sichere Enklave **162** kann Secure-Enclave-Control-Structure (SECS) **182** und Enklavenseite(n) **184**, auch bezeichnet als EPC-Seite(n) **184** beinhalten, ist aber nicht darauf beschränkt. Die sichere Enklave **162** kann ferner mindestens eine erweiterte Versions-Array-Seite (EVA) **186** beinhalten, um mehrere UMDS **188** zu speichern, wobei jede UMDS **188** einer Seite von sicherem Speicher zugeordnet ist. Die SECS **182** identifiziert und definiert die Enklave, zum Beispiel mit einer Enklaven-ID (EID), die nach der Erzeugung der Enklave nach Ausführung einer Enklavenerzeugungsanweisung (zum Beispiel ECREATE-Anweisung) erzeugt wird. Jede der Enklave zugeordnete Unter-EPC-Seite kann einen Back-Zeiger auf die SECS **182** beinhalten, um mit der Enklave identifiziert zu werden. Die EVA-Seite **186** kann mit einer Liste von Slots erzeugt werden, wobei jeder Slot die UMDS **188** einer in das Auslagern einbezogenen Seite speichern soll. Die EVA-Seite **186** kann auch ausgelagert werden (zum Beispiel wenn gefüllt), solange die EVA-Seite **186** logisch mit einer verbleibenden Parent-EVA-Seite **186** in dem sicheren Speicher verbunden bleibt. Die Strukturen der sicheren Enklave (EPC) **162** und ihre Rolle in der offenbarten UMDS-basierten Architektur werden detaillierter erläutert.

[0023] **Fig. 2** ist ein Blockdiagramm von Verfahren **200** für das Übersetzen von virtueller Adresse **202** für eine Seite letztendlich in physikalische Host-Adresse **222**, gemäß offenbarten Implementierungen. Aus einer Systemperspektive ist Adressübersetzung eine Ebene von Umleitung zwischen den virtuellen Adressen, die durch Speicherladung eines Programms und durch Speicheranweisungen verwendet werden, und den physikalischen Adressen, die auf den physikalischen Adressraum des Systemspeichers **160** verweisen. Die Abbildung zwischen virtuellen und physikalischen Adressen wird durch Seitentabellen definiert, die durch die Systemsoftware verwaltet werden.

[0024] Betriebssysteme verwenden Adressübersetzung, um virtuelle Speicherabstraktion zu implementieren. Die virtuelle Speicherabstraktion legt die gleiche Schnittstelle wie die Speicherabstraktion in ISA frei, jedoch kann jeder Prozess einen separaten virtuellen Adressraum verwenden, der nur auf den diesem Prozess zugeordneten Speicher verweist Von dem Standpunkt eines Anwendungsentwicklers kann virtueller Speicher modelliert werden, indem davon ausgegangen wird, dass jeder Prozess auf einem separaten Computer läuft und seinen eigenen DRAM hat.

[0025] Adressübersetzung kann durch das Betriebssystem dazu verwendet werden, DRAM zwischen mehreren Anwendungsprozessen zu vervielfachen, die Prozesse voneinander zu isolieren und Anwendungscode an direktem Zugreifen auf speicherabgebildete Geräte hindern. Die letzten beiden Schutzmaßnahmen hindern Fehler einer Anwendung an dem Beeinflussen anderer Anwendungen oder des OS-Kernel selbst. Der VMM **124** kann Adressübersetzung dazu verwenden, den DRAM zwischen Betriebssystemen, die gleichzeitig laufen, aufzuteilen, und um speicherabgebildete Geräte zu virtualisieren.

[0026] Das System **100** kann sich die Hardwarevirtualisierung mit Verwendung des VMM **124** (zum Beispiel Hypervisor) zu Nutze machen, um mehrere Guest-Betriebssysteme zur gleichen Zeit auszuführen. Dies erzeugt einige Spannung, da jedes Guest-Betriebssystem unter der Annahme geschrieben wurde, dass es den gesamten Systemspeicher des Computers (zum Beispiel DRAM) besitzt. Die Spannung kann durch eine zweite Ebene von Adressübersetzung gelöst werden, veranschaulicht in **Fig. 2**.

[0027] Wenn der VMM **124** aktiv ist, stellen die durch eine Betriebssystemabbildung eingerichteten Seitentabellen **166** erste **Abb. 210** zwischen den virtuellen Adressen **202** und physikalischen Guest-Adressen (GPA) **212** in einem physikalischen Guest-Adressraum bereit. Der VMM **124** vervielfacht den DRAM des Computers zwischen den physikalischen Guest-Adressräumen des Betriebssystems über die zweite Ebene von Adressübersetzungen, die die EPT **167** verwendet, um durch Verwenden von zweiter **Abb. 220** die physikalischen Guest-Adressen (GPA) **212** auf die physikalischen Host-Adressen (HPA) **222** abzubilden.

[0028] Die EPT **167** kann die gleiche Datenstruktur wie die Seitentabellen **166** verwenden, sodass der Prozess des Übersetzens von physikalischen Guest-Adressen in physikalische Adressen den gleichen Schritten wie die IA-32e-Adressübersetzung folgt. Der Hauptunterschied ist, dass die physikalische Adresse des Root-Knotens der Datenstruktur in einem Extended-Page-Table-Pointer (EPTP) -Feld der VMCS **168** für das Guest-OS gespeichert wird. Wie erwähnt, können, sobald Übersetzungen für eine Seite, die der VMM **124** im Voraus ausführen kann, erzeugt werden, die GPA **212** und die HPA **222** jeweils in einem TLB **134** gespeichert werden,

der jedem Kern **112** zugeordnet ist, der auf die Seite innerhalb der geteilten sicheren Enklave **162** zugegriffen hat. Das Caching der GPA **212** und der HPA **222** innerhalb des TLB **134** kann die Notwendigkeit verhindern, die virtuelle Adresse für wiederholten Zugriff auf Anweisungen oder Daten, die auf der HPA **222** platziert sind, kontinuierlich neu zu übersetzen.

[0029] **Fig. 3** ist ein Blockdiagramm einer sicheren Enklavenschaltung **120** des Prozessors **110** von **Fig. 1A**, gemäß Implementierungen. Die sichere Enklavenschaltung **120** kann innerhalb des Kerns **112** oder außerhalb des Kerns **112** ansässig sein, wie in **Fig. 1** gezeigt. Die sichere Enklavenschaltung **120** kann erweiterte Auslagerungsfähigkeiten für einen sicheren Enclave-Page-Cache (EPC) **162**, der allgemein in dem Systemspeicher **160** gespeichert ist, bereitstellen. In verschiedenen alternativen Implementierungen kann sich der EPC **162** auf eine oder mehrere verteilte Strukturen erstrecken, die durch mehrere Hardware-Threads, logische Prozessoren oder Verarbeitungskerne geteilt werden, um sichere Daten für eine Adresse einer geteilten EPC-Seite, die einer sicheren Enklave zugewiesen ist und durch die Hardware-Threads, logischen Prozessoren oder Verarbeitungskerne zugänglich ist, zu speichern. Zum Beispiel kann sich der EPC **162** auf den Cache **130** (oder ähnliche Speicherung) eines oder mehrerer Kerne **112** des Prozessors **110** erstrecken (mit den gestrichelten Linien angezeigt). Da die EPC-Seite eine Speicherseite ist, kann die EPC-Seite aus dem Cache geräumt werden. Aufgrund einer immer zunehmenden Verfügbarkeit von Systemspeicher und aufgrund dessen, dass der EPC **162** innerhalb prozessorreservierter Speicherbereiche flexibel konvertierbar sein kann, können moderne Prozessoren das Caching sicherer Daten in On-Chip-EPC unterlassen, obwohl diese Option immer noch innerhalb des Umfangs der offenbarten Implementierungen ist.

[0030] Die sichere Enklavenschaltung **120** kann als Reaktion auf eine Anzahl von Enklavenanweisungen **302** Operationen ausführen (in gestrichelten Linien angezeigt), von denen einige bereits erwähnt worden sind. Die Enklavenanweisungen **302** können Enklavenblock (EBLOCK), Enklavenfreigabe (EUNBLOCK), Enklavenverfolgen (ETRACK), Enklaven-Make-Extended-Version-Array (EMKEVA), Enklavenzurückschreiben (EWB), Enklavenladen (ELD), Enklavenverlassen (EEXIT), Enklaveneingabe (EENTER) und andere nicht gezeigte Enklavenanweisungen (zum Beispiel AEX-Anweisung, ERESUME-Anweisung, usw.) beinhalten, sind jedoch nicht darauf beschränkt. Darüber hinaus kann die ELD-Anweisung als eine erweiterte ELD (ELDE) -Anweisung bezeichnet werden, und die EWB-Anweisung kann als eine erweiterte EWB (EWBE) -Anweisung bezeichnet werden, wenn unter Verwendung von UMDSbezogenen Strukturen ausgeführt, wie unten detailliert beschrieben wird.

[0031] Der Systemspeicher **160** kann ferner UMDS-Informationsdatenstruktur (UMDSINFO) **380** speichern, die allgemeiner auch als eine Informationsdatenstruktur bezeichnet werden kann. Die UMDSINFO **380** kann Parameter, Adressen und ein der Verwendung der UMDS **188** innerhalb ausgewählter Enklavenanweisungen **302** zugeordnetes geblocktes Bit, wie zum Beispiel das ELDE und das EWBE, speichern. Die UMDSINFO **380** wird unter Bezugnahme auf mindestens die **Fig. 7A** und **Fig. 7B** in zusätzlichem Detail erklärt.

[0032] Die sichere Enklavenschaltung **120** kann ferner Enclave-Page-Cache-Map (EPCM) **304**, Eingaberegister **306**, Ausgaberegister **308**, zwei oder mehr Epochenzählerspeicherungsplätze beinhalten, einschließlich vorherigen Epochenzählers (PE) **307**, aktuellen Epochenzählers (CE) **309**, mehrere Bereichsregister **312**, Zugriffssteuereinheit **314** und Integritätsschutzeinheit **318**. Die Eingaberegister **306** und die Ausgaberegister **308** können GPR oder anderer Typ von Register sein. Die Eingaberegister **306** können eine virtuelle Adresse einer Seite (zum Beispiel in dem RCX), eine Adresse eines Versions-Array-Slots (zum Beispiel in dem RDX), die Adresse der UMDSINFO **380** (zum Beispiel in dem RBX) und andere, wie spezifiziert, bereitstellen, obwohl das bestimmte Register unterschiedlich als offenbart zugewiesen werden kann. Die Ausgaberegister **308** können primär dazu verwendet werden, Fehlercodes oder -Status unter Bezugnahme auf Ausführung von Aspekten der Enklavenanweisungen auszugeben. Die Bereichsregister **312** können für Systemsoftware (wie zum Beispiel das OS oder den VMM) Bereiche reservierten Speichers spezifizieren, die in den EPC **162** umgewandelt und als dieser verwendet werden können, die somit flexibel innerhalb des Systemspeichers **160** wachsen können, abhängig von der Anforderung durch Anwendungen, die zu der Verwendung der Enklavenanweisungen **302** aufrufen.

[0033] Die EPCM **304** ist eine sichere Struktur, die von dem Prozessor **110** dazu verwendet wird, Inhalte des EPC **162** zu verfolgen. Die EPCM **304** kann einen Eintrag für jede Seite halten, die aktuell in den EPC **162** geladen ist, auf die nicht durch Software zugegriffen werden kann, und das Layout der EPCM-Felder kann implementierungsspezifisch sein. Eines der Felder der EPCM kann ein geblocktes Bit speichern, um anzuzeigen, ob jegliche gegebene Seite durch die EBLOCK-Anweisung geblockt worden ist. Wenn die Seite in den EPC **162** aus diesem ausgelagert wird, kann ein aktueller Epochenzähler einen Wert liefern, der in

dem CE **309**-Speicherungsplatz gespeichert werden kann. Nach einem weiteren Taktzyklus kann der für eine bestimmte Epoche in dem CE **309** gespeicherte Wert an den PE **307**- Speicherungsplatz bewegt werden.

[0034] Der Prozessorkern **112** beinhaltet auch die TLB **138**, in denen Übersetzungen gepuffert werden können, um Zugriff auf bestimmte Seiten innerhalb des EPC **162** bereitzustellen. Der Prozessorkern **110** kann auch Decodierungsstufe **322**, Lesestufe **324**, eine oder mehrere Ausführungseinheiten (zum Beispiel Ausführungseinheit **326**) und Schreibstufe **328** beinhalten. Implementierungen des Prozessorkerns **110** können auch andere Pipelinestufen (zum Beispiel wie in Pipeline **1601** von **Fig. 16B** gezeigt) für die Ausführung der Enklavenanweisungen **302** umfassen, um erweiterte Auslagerungsfähigkeiten für den sicheren Enclave-Page-Cache (EPC) **162** bereitzustellen.

[0035] In einer Implementierung spezifiziert die EBLOCK-Anweisung eine geteilte Seitenadresse als einen Operanden. Eine oder mehrere Ausführungseinheiten (zum Beispiel die Ausführungseinheit **326**) markieren einen einer Enclave-Page-Cache-Mapping in EPCM **304** entsprechenden Eintrag für die Adresse der geteilten Seite, um die Erzeugung einer neuen TLB-Übersetzung (zum Beispiel in TLB **138** oder in jeglichem anderen TLB zu blocken) für Hardware-Threads, logische Prozessoren oder Verarbeitungskerne, um auf die geteilte Seite zuzugreifen. In einer Implementierung spezifiziert die ETRACK-Anweisung die sichere Enklave **162** als einen Operanden, und eine oder mehrere Ausführungseinheiten (zum Beispiel die Ausführungseinheit **326** oder die Zugriffssteuereinheit **314**) zeichnen die aktuell auf sichere Daten zugreifenden Hardware-Threads in dem der sicheren Enklave entsprechenden EPC **162** auf. Zum Beispiel kann in einer Implementierung die Enklave zwei oder mehr Epochenzähler (zum Beispiel in dem PE **307** und dem CE **309**) aufrechterhalten, um eine Anzahl von aktuell auf sichere Daten in der aktuellen Epoche der sicheren Enklave (zum Beispiel in dem CE **309**) zugreifende Hardware-Threads aufzuzeichnen. Die Enklave kann dann diese Anzahl in einen letzten vorherigen Epochenzähler (zum Beispiel in den PE **307**) kopieren und eine neue Epoche ohne Hardware-Threads als die aktuelle Epoche (zum Beispiel in dem CE **309**) initialisieren.

[0036] Die Systemsoftware (zum Beispiel OS, VMM) kann dann einen Inter-Processor-Interrupt (IPI) an beliebige Hardware-Threads, logische Prozessoren oder Verarbeitungskerne, die aktuell auf der sicheren Enklave entsprechende sichere Daten in der EPC **162** zugreifen, senden. Jeder Hardware-Thread, logische Prozessor oder Verarbeitungskern, der aktuell auf die der sicheren Enklave entsprechenden sicheren Daten zugreift, hätte die sichere Enklave mit einer die sichere Enklave spezifizierenden EENTER (oder ERESUME) -Anweisung betreten, und zu dieser Zeit wäre dem Hardware-Thread, logischen Prozessor oder Verarbeitungskern eine Epochennummer zugeordnet worden. Wenn die Hardware-Threads, logischen Prozessoren oder Verarbeitungskerne den Inter-Processor-Interrupt (IPI) bestätigen und die sichere Enklave verlassen, werden ihre TLB-Übersetzung oder -Übersetzungen geleert (zum Beispiel von dem TLB **138** oder anderen dem jeweiligen Thread, logischen Prozessor oder Verarbeitungskern zugeordneten TLB). Wann immer Hardware-Threads von der letzten vorherigen Epoche die sichere Enklave mit einer EEXIT (oder AEX) -Anweisung verlassen, wird die aufgezeichnete Anzahl von Hardware-Threads in dem letzten vorherigen Epochenzähler (zum Beispiel in dem PE **307**) verringert.

[0037] Wenn die aufgezeichnete Anzahl von Hardware-Threads (zum Beispiel in dem PE **307**) Null („0“) erreicht, wird es als sicher für das System angenommen, die Seite zu räumen, die Daten zu verschlüsseln und die verschlüsselte Seite in Speicher oder nichtflüchtige Speicherung zurückzuschreiben. In einer Implementierung kann die Systemsoftware die EWB-Anweisung, die die Adresse der geteilten Seite als einen Operanden spezifiziert, ausführen, um die Räumung abzuschließen, die gesicherten Daten zu verschlüsseln und die Seite in nichtflüchtige Speicherung zu schreiben. Da Enklavenschutz der sicheren Daten nicht in der Lage sein kann, der Systemsoftware zu vertrauen, kann eine Implementierung der EWB-Anweisung fehlschlagen, wenn die aufgezeichnete Anzahl von Hardware-Threads von der letzten vorherigen Epoche (zum Beispiel in dem PE **307**) nicht Null erreicht hat. In anderen alternativen Implementierungen kann die EWB-Anweisung warten, bis die aufgezeichnete Anzahl von Hardware-Threads (zum Beispiel in dem PE **307**) Null erreicht, um ausgeführt zu werden, oder EWB-Anweisung **1533** kann eine Ausnahme veranlassen.

[0038] Während Berechtigungen, physikalischer Speicher und/oder sich ändernde Abbildungen durch die Systemsoftware verwaltet werden können, wenn die Speicherinhalte, wie in einer sicheren Enklave, geschützt sind, kann die Systemsoftware nicht berechtigt oder verlässlich sein, auf die aktuellen geschützten Inhalte des privaten Speichers der Enklave zuzugreifen Ein abgestufter Ansatz kann dazu eingesetzt werden, die Sicherheit und/oder Integrität von Inhalten privaten Speichers zu garantieren und die technischen Einschränkungen begrenzter Menge physikalischen Speichers (zum Beispiel des EPC **162**) zu verwalten, um einen größeren, geschützten privaten Speicher der Enklave zu unterstützen, ohne in der Lage zu sein, Systemsoftware zu vertrauen. Zum Beispiel setzen die offenbarten Implementierungen Anweisungen und Verarbeitungslogik dazu

ein, erweiterte Auslagerungsfähigkeiten für sichere Enklave-Page-Caches bereitzustellen, ohne aufwendige Hardware-Unterstützung und/oder Design-Aufwand zu erfordern.

[0039] **Fig. 4** ist ein Flussdiagramm von Verfahren **400** für einen Auslagerungsfluss einer Seite in und aus sicherem Speicher, gemäß Implementierungen. Das Verfahren **400** beschreibt in abgekürzter Weise den Auslagerungsfluss einer Seite (die ausgelagert und dann zurück in sicheren Speicher eingelagert wird), wie unter Bezugnahme auf **Fig. 3** in einigen Details beschrieben, und wie der Auslagerungsfluss durch Ausführung der UNBLOCK-Anweisung beeinflusst wird. Das Verfahren **400** kann durch Verarbeitungslogik ausgeführt werden, die Hardware (zum Beispiel Schaltungen, zugeordnete Logik und/oder programmierbare Logik), Software (zum Beispiel auf einem Computersystem ausführbare Anweisungen, um Hardware-Simulation auszuführen), oder eine Kombination davon beinhalten kann. In einem veranschaulichenden Beispiel kann das Verfahren **400** durch den Prozessor **110** des Systems **100** von den **Fig. 1-3**, einschließlich durch die sichere Enklavenschaltung **120**, ausgeführt werden.

[0040] Das Verfahren **400** kann damit beginnen, dass die Verarbeitungslogik die Seite als nicht in den Seitentabellen **166** und den EPT **167** vorhanden markiert (**403**). Das Verfahren **400** kann damit fortfahren, dass die Verarbeitungslogik eine EBLOCK-Anweisung ausführt, um die Seite für Räumung zu markieren, sodass alle neuen Übersetzungen Fehler sein sollten (**405**). Ausführung der EBLOCK-Anweisung kann die Verarbeitungslogik dazu veranlassen, den Ablauf einer Zeitdauer zu erkennen, während derer nicht auf die Seite zugegriffen wird, und ein Bit in der Enklave-Page-Cache-Map für die virtuelle Adresse zu setzen, um Erzeugung weiterer Übersetzungen der virtuellen Adresse für Zugriff auf die Seite zu blocken. Das Verfahren **400** kann damit fortfahren, dass die Verarbeitungslogik eine ETRACK-Anweisung ausführt, um die Epochenzähler zu verfolgen, und jeglichen Hardware-Thread, logischen Prozessor oder Verarbeitungskern, der Zugriff auf die Seite hatte, der den sicheren Speicher verlassen hat, zu identifizieren (**407**). Wie diskutiert, kann die Verarbeitungslogik als Teil der Ausführung der ETRACK-Anweisung einen IPI an Kern(e), der/die den Hardware-Threads zugeordnet ist/sind, die Zugriff auf die Seite haben, senden, was die Hardware-Threads dazu veranlassen kann, die sichere Enklave und den Kern/die Kerne zu verlassen, um ihre TLB zu leeren (**409**).

[0041] Nachdem die Verarbeitungslogik die sicheren Anweisungen ausgeführt hat und die TLB-Leerung veranlasst hat, kann die Verarbeitungslogik in einigen Fällen einen Seitenfehler der Seite, die einen versuchten Zugriff auf die Seite durch einen System-Agent anzeigt, erkennen. Die offenbarten Implementierungen stellen die Ausführung einer neuen EUNBLOCK-Anweisung, zum Beispiel in Reaktion auf Erkennung des Seitenfehlers, bereit (**411**). Die EUNBLOCK-Anweisung kann die Erzeugung weiterer Übersetzungen der virtuellen Adresse für die Seite freigeben. Um dies zum Beispiel zu tun, kann die Verarbeitungslogik das Bit in der Enklave-Page-Cache-Map für die virtuelle Adresse löschen und die Seite als in den mehreren Übersetzungstabellen vorhanden markieren. Zusätzliche Schritte können bei dem Ausführen der EUNBLOCK-Anweisung verwendet werden, wie detaillierter unter Bezugnahme auf die **Fig. 5**, **Fig. 6A** und **Fig. 6B** diskutiert. Vorteilhafterweise kann die Verarbeitungslogik dann die Schritte **417**, **419**, **421** und **423**, die zum Beispiel mit einem erfolgreichen Abschluss der UNBLOCK-Anweisung für die Seite entfernt werden, überspringen (**412**). Das Verfahren **400** kann damit fortfahren, dass die Verarbeitungslogik die Seite wieder als in den Seitentabellen **166** und den EPT **167** vorhanden markiert (**413**).

[0042] Wenn jedoch kein Seitenfehler erkannt wird, bevor die Verarbeitungslogik die EWB-Anweisung ausführt, kann das Verfahren **400** damit fortfahren, dass die Verarbeitungslogik die Ausführung der EWB-Anweisung abschließt, um die Daten für die Seite zu verschlüsseln und die verschlüsselten Daten in Speicher zu schreiben (**417**). Das Verfahren **400** kann unter diesen Umständen damit fortfahren, dass die Verarbeitungslogik die verschlüsselten Daten auf Festplatte schreibt (**419**). Das Verfahren **400** kann damit fortfahren, dass die Verarbeitungslogik die verschlüsselten Daten von der Festplatte liest (**421**). Das Verfahren **400** kann damit fortfahren, dass die Verarbeitungslogik die verschlüsselten Daten entschlüsselt und die entschlüsselten Daten in einen freien Slot des EPC **162** des sicheren Speichers lädt (**423**). Wenn während der Verarbeitung der als **412** ausgewiesenen Schritte die Verarbeitungslogik einen Seitenfehler erkennt, kann es immer noch notwendig sein, dass der Prozessor **110** die Ausführung des Rests der als **412** ausgewiesenen Schritte abschließt, damit die Seite in die sichere Enklave wiederhergestellt wird.

[0043] **Fig. 5** ist ein Flussdiagramm von Verfahren **500** für das Freigeben der Erzeugung weiterer Übersetzungen einer Seite nach Empfang eines der Seite zugeordneten Seitenfehlers, gemäß Implementierungen. Das Verfahren **500** kann durch Verarbeitungslogik ausgeführt werden, die Hardware (zum Beispiel Schaltungen, zugeordnete Logik und/oder programmierbare Logik), Software (zum Beispiel auf einem Computersystem ausführbare Anweisungen, um Hardware-Simulation auszuführen), oder eine Kombination davon beinhalten

kann. In einem veranschaulichenden Beispiel kann das Verfahren **500** durch den Prozessor **110** des Systems **100** von den **Fig. 1-3**, einschließlich durch die sichere Enklavenschaltung **120**, ausgeführt werden.

[0044] Das Verfahren **500** kann damit beginnen, dass die Verarbeitungslogik sichere Daten einschließlich einer Seite an einer virtuellen Adresse in einer sicheren Enklave in Systemspeicher speichert (**510**). Das Verfahren **500** kann damit fortfahren, dass die Verarbeitungslogik die EPCM **304** für die Enklave in einer sicheren Speicherungsstruktur der sicheren Enklavenschaltung **120** speichert (**515**). Das Verfahren **500** kann damit fortfahren, dass die Verarbeitungslogik bestimmt, ob eine Anforderung, die Seite von dem sicheren Speicher zu räumen, empfangen worden ist (**520**). Wenn nicht empfangen, kann die Verarbeitungslogik warten (**520**). Wenn empfangen, kann das Verfahren **500** damit fortfahren, dass die Verarbeitungslogik die Seite als nicht in den Seitentabellen und erweiterten Seitentabellen vorhanden markiert (**525**). Das Verfahren **500** kann damit fortfahren, dass die Verarbeitungslogik den Ablauf von Zeit erkennt, während derer nicht auf die Seite zugegriffen wird (**530**). Das Verfahren **500** kann damit fortfahren, dass die Verarbeitungslogik ein Bit in der Enklave-Page-Cache-Map für die zu der virtuellen Adresse abgebildete Seite setzt, um die Erzeugung von Übersetzungen der virtuellen Adresse für Zugriff auf die Seite zu blocken, zum Beispiel, wenn die Seite von der sicheren Enklave entfernt werden soll (**535**).

[0045] In verschiedenen Implementierungen kann das Verfahren **500** damit fortfahren, dass die Verarbeitungslogik einen oder mehrere aktuell auf die sicheren Daten in der sicheren Enklave zugreifende Hardware-Threads aufzeichnet (**540**). Das Verfahren **500** kann damit fortfahren, dass die Verarbeitungslogik einen Inter-Processor-Interrupt (IPI) an einen oder mehrere dem einen oder den mehreren Hardware-Threads zugeordnete Kerne sendet, um den einen oder die mehreren Hardware-Threads dazu zu veranlassen, die sichere Enklave zu verlassen und Translation-Lookaside-Buffer des einen oder der mehreren Kerne zu leeren (**545**). Das Verfahren **500** kann damit fortfahren, dass die Verarbeitungslogik bestimmt, ob ein Seitenfehler auf der Seite vor Ausführung einer EWB/E-Anweisung erkannt worden ist (**550**). Wenn nicht, kann das Verfahren **500** damit fortfahren, dass die Verarbeitungslogik mit Ausführung der EWB/E-Anweisung fortfährt (**555**). Wenn ja, ein Fehler wird erkannt, kann das Verfahren **500** damit fortfahren, dass die Verarbeitungslogik das Bit in der EPCM für die zu der virtuellen Adresse abgebildete Seite löscht, um Erzeugung von Übersetzungen für Zugriff auf die Seite in der sicheren Enklave freizugeben (**560**). Das Verfahren **500** kann damit fortfahren, dass die Verarbeitungslogik die Seite als in den Seitentabellen und den EPT vorhanden markiert (**565**).

[0046] Die **Fig. 6A** und **Fig. 6B** sind ein Flussdiagramm von Verfahren **600** für das Freigeben der Erzeugung weiterer Übersetzungen der Seite nach Empfang eines der Seite zugeordneten Seitenfehlers, gemäß einer alternativen Implementierung. Das Verfahren **600** kann durch Verarbeitungslogik ausgeführt werden, die Hardware (zum Beispiel Schaltungen, zugeordnete Logik und/oder programmierbare Logik), Software (zum Beispiel auf einem Computersystem ausführbare Anweisungen, um Hardware-Simulation auszuführen), oder eine Kombination davon beinhalten kann. In einem veranschaulichenden Beispiel kann das Verfahren **600** durch den Prozessor **110** des Systems **100** von den **Fig. 1-3**, einschließlich durch die sichere Enklavenschaltung **120**, ausgeführt werden. Das Verfahren **600** soll zusätzliche Details bezüglich der Ausführung der EUNBLOCK-Anweisung, die ausgeführt werden soll, wenn die aktuelle Privilegstufe Null („0“) ist, erklären.

[0047] Die EUNBLOCK-Anweisung kann ausgeführt werden, um eine Seite von dem EPC (zum Beispiel EPC_PAGE) als nicht geblockt zu markieren. Das Universalregister RAX kann einen Fehlercode empfangen, der das Ergebnis der Ausführung der EUNBLOCK-Anweisung anzeigt. Das Universalregister RCX kann die virtuelle Adresse einer EPC-Seite enthalten. In einigen Implementierungen ist die Adresse in dem RCX-Register eine effektive Adresse, wobei das Datensegment (DS) dazu verwendet werden kann, die virtuelle Adresse zu erzeugen. Segmentüberschreibung kann nicht unterstützt werden. Die EUNBLOCK-Anweisung kann fehlschlagen, wenn der Operand nicht ordnungsgemäß ausgerichtet ist oder sich nicht auf eine EPC-Seite bezieht, oder die Seite durch einen anderen Thread in Gebrauch ist. Das RCX kann Lese-/Schreibberechtigungen für EPC PAGE haben und auf in Tabelle 1 aufgelistete temporäre Variablen verweisen, gemäß einer Implementierung. Andere Implementierungen sind vorgesehen, einschließlich Verwendung unterschiedlicher Register als derjenigen, die als verwendet erwähnt werden. Der Begriff „UINT“ in Tabelle 1 steht für vorzeichenlose Ganzzahl.

Tabelle 1

Variablen-Name	Typ	Größe (Byte)	Beschreibung
TMP_SECS	Physikalische Adresse	64	Physikalische Adresse der SECS der Seite, die entfernt wird
TMP_PBEPOCH	UINT64	8	Momentaufnahme des vorherigen Epochenwerts.
TMP_PBREFCOUNT	UINT64	8	Momentaufnahme des vorherigen Epochenzählerwerts.

[0048] Unter weiterer Bezugnahme auf die **Fig. 6A** und **Fig. 6B** kann das Verfahren **600** damit beginnen, dass die Verarbeitungslogik die Ausrichtung von DS:RCX, was ein Beispiel des Registers ist, das dazu verwendet werden kann, die virtuelle Adresse der freizugebenden Seite zu speichern, prüft (**610**). Das Verfahren **600** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass DS: RCX lese/schreibzugänglich ist (**615**) Das Verfahren **600** kann damit fortfahren, dass die Verarbeitungslogik bestimmte Flags (RFLAGS) in den 64-Bit-Spezialregistern (zum Beispiel genullt) sowie dem Universalregister RAX, das Fehlercodes empfangen soll, einrichtet.

[0049] Das Verfahren **600** kann damit fortfahren, dass die Verarbeitungslogik bestimmt, ob ein Ressourcenkonflikt für die Seite erkannt worden ist (**625**), zum Beispiel, um mögliche Enklaveneintrittswettbewerbszustände zu handhaben. Wenn ein Ressourcenkonflikt erkannt wird, kann das Verfahren **600** damit fortfahren, dass die Verarbeitungslogik einen Ressourcenkonfliktfehler (zum Beispiel RESOURCE_CONFLICT) in das RAX-Register schreibt (**630**). Die Konflikterkennung kann nach einem fehlgeschlagenen Versuch auftreten, den EPCM von dem Akzeptieren weiterer Einträge zu sperren. Ein ZF-Flag kann auch gesetzt werden infolge des Fehlschlags, eine Sperre auf der EPCM zu erfassen, oder wenn die Seite ungültig ist. Die Konflikterkennung kann auch unter Verwendung einer Transaktionsverarbeitung ausgeführt werden, bei der Ergebnisse bestimmter Verarbeitungsschritte aufgegeben werden, wenn ein Ressourcenkonflikt letztlich durch das Ende der Verarbeitungsschritte erkannt wird. Wenn es keinen Ressourcenkonflikt für die Seite gibt, kann das Verfahren **600** damit fortfahren, die virtuelle Adresse von dem RCS zu lesen (**635**).

[0050] Das Verfahren **600** kann damit fortfahren, dass die Verarbeitungslogik die virtuelle Adresse in ihre GPA und von der GPA in ihre HPA übersetzt (**640**). Das Verfahren **600** kann damit fortfahren, dass die Verarbeitungslogik bestimmt, ob sich die HPA zu einer EPC-Seite (zum Beispiel ist es eine Seite in einer sicheren Enklave) auflöst (**645**). Wenn sie das nicht tut, kann das Verfahren **600** damit fortfahren, dass die Verarbeitungslogik einen Nicht-EPC-Fehler in das RAX-Register schreibt (**650**). Wenn sie sich zu einer EPC-Seite auflöst, kann das Verfahren **600** damit fortfahren, dass die Verarbeitungslogik bestimmt, ob die Seite schon geblockt ist (**652**). Wenn die Seite schon geblockt ist, kann das Verfahren **600** damit fortfahren, dass die Verarbeitungslogik einen Page-Not-Blocked-Fehler in das RAX-Register schreibt (**655**). Ein CF-Flag kann auch gesetzt werden, wenn die Seite nicht geblockt ist, oder wenn die Seite aus einem erkennbaren Grund (zum Beispiel nicht der richtige Seitentyp) nicht blockbar ist.

[0051] Unter zusätzlicher Bezugnahme auf **Fig. 6B**, wenn die Seite nicht schon geblockt ist, kann das Verfahren **600** damit fortfahren, dass die Verarbeitungslogik die HPA der SECS unter Bezugnahme auf Eintrag in der EPCM bestimmt (**660**). Das Verfahren **600** kann damit fortfahren, dass die Verarbeitungslogik unter Verwendung der HPA der SECS auf eine Epochenverfolgungsstruktur für die Epoche der aktuellen Enklave zugreift (**665**). Das Verfahren **600** kann damit fortfahren, dass die Verarbeitungslogik von der Epochenverfolgungsstruktur einen vorherigen Epochenwert und einen vorherigen Epochenzählerwert in einen Satz lokaler Epochenregister liest (**670**). Dies ist das Protokoll für ETRACK- und EWB-Anweisung. Die ETRACK-Anweisung verfolgt den Status der Enklave mit diesen Epochenwerten und EWB verifiziert den Enklavenstatus durch Vergleichen der aktuellen und vorherigen Werte. Das Verfahren **600** kann damit fortfahren, dass die Verarbeitungslogik basierend auf diesen Epochenwerten verifiziert, dass Hardware-Threads die sichere Enklave verlassen haben (**675**). Das Verfahren **600** kann damit fortfahren, dass die Verarbeitungslogik den der sicheren Enklave zugeordneten vorherigen Epochenwert löscht (**680**). Das Verfahren **600** kann damit fortfahren, dass die Verarbeitungslogik das geblockte Bit in der EPCM für die virtuelle Adresse der Seite löscht (**685**).

[0052] **Fig. 7A** ist ein Diagramm von Unified-Metadata-Structure-Informationsdatenstruktur (UMDSINFO) **700** (zum Beispiel UMDSINFO-Datenstruktur), gemäß Implementierungen. **Fig. 7B** ist ein Diagramm eines Daten-

vektors, auch genannt Bitvektor (BITVEK) **710**, der ein Teil der UMDSINFO-Datenstruktur ist, gemäß Implementierungen. Die UMDSINFO **700** kann Felder beinhalten, die den BITVEC **710** als einen Satz von Bits umfassen, der an die EWBE- und ELDE-Anweisungen weitergegeben werden kann (siehe **Fig. 10**). Der BITVEC **710** kann auch das geblockte Bit von der EPCM und ein USE_UMBDS-Bit beinhalten, beide mit Auswirkung darauf, von wo das geblockte Bit durch die UMDS erhalten wird (siehe auch **Fig. 10**).

[0053] In Implementierungen hält der SRCPGE-Eintrag in der UMDSINFO **700** einen virtuellen Adresszeiger (oder in einigen Fällen einen effektiven Adresszeiger, der mit Segmentberechnungen in einen virtuellen Adresszeiger verwandelt werden kann) zu der Nicht-EPC-Seite, deren Inhalte in die neu zugewiesene EPC-Seite kopiert werden. Darüber hinaus kann der SECS-Eintrag ein virtueller Adresszeiger zu dem EPC-Slot sein, der aktuell eine Kopie der SECS enthält, die die Enklave mit einer nach Ausführung der ECREATE-Anweisung erzeugten Enklaven-ID identifiziert und definiert. Jede der Enklave zugeordnete Unter-EPC-Seite beinhaltet einen Back-Pointer zu der SECS, um mit dieser Enklave identifiziert zu werden.

[0054] **Fig. 8** ist ein Diagramm von erweiterter Versions-Array (EVA) -Seite **800**, gemäß Implementierungen. Das „erweitert“ in der EVA-Seite **800** stellt Bezug auf die Tatsache her, dass die EVA-Seite **800** jetzt größere (64 Byte) Unified-Metadata-Structures (UMDS) speichern kann, eine für jede in der sicheren Enklave gespeicherte Seite. Da die UMDS größer ist als der Versionswert, der alles war, was üblicherweise in einer Versions-Array-Seite gespeichert wurde, hat die EVA-Seite **800** weniger Slots (zum Beispiel 64) als ihr Vorgänger (zum Beispiel **512**), die VA-Seite. Wie in dem Feld „gesetzt durch“ angemerkt, können die EWBE- und ELDE-Anweisungen in eine in einem der Slots gespeicherte UMDS speichern und von dieser lesen. Darüber hinaus können in einer Implementierung Sperren verwendet werden, um zu gewährleisten, dass eine atomare Aktualisierung den Wert des Versionsfelds innerhalb der UMDS schreibt.

[0055] In Implementierungen kann die EVA-Seite **800** durch Verwenden der EMKEVA-Anweisung, die die virtuelle Adresse einer freien EPC-Seite nimmt und sie in ein Versions-Array mit leeren Slots umwandelt, zugewiesen werden. EVA-Seiten werden durch den Seitentyp EVA-Typ in ihren EPCM-Einträgen identifiziert. Wie VA-Seiten haben EVA-Seiten die ENCLAVEADDRESS-Felder in ihren EPCM-Einträgen auf Null gesetzt und können von keiner Software, einschließlich Enklaven, direkt aufgerufen werden.

[0056] Im Gegensatz zu einigen der anderen bisher diskutierten Seitentypen, sind EVA-Seiten nicht jeglicher Enklave zugeordnet. Das bedeutet, dass EVA-Seiten ohne Beschränkung über die EREMOVE-Anweisung freigegeben werden können. Das Freisetzen einer EVA-Seite, deren Slots in Gebrauch sind, verwirft effektiv die UMDS in diesen Slots, was zur Folge hat, dass die Möglichkeit verloren geht, die entsprechenden geräumten Seiten in den EPC zurückzuladen. Deshalb ist es unwahrscheinlich, dass eine korrekte OS-Implementierung jemals EREMOVE auf einer EVA-Seite mit nicht freien Slots aufrufen wird. Eine EVA-Seite kann jedoch auch aus sicherem Speicher entfernt (zum Beispiel ausgelagert) werden, solange sie (durch hierarchisches Verbinden von EVA-Seiten) mit einer EVA-Seite, die nicht aus dem sicheren Speicher ausgelagert worden ist, verbunden sein kann.

[0057] **Fig. 9** ist ein Diagramm von Unified-Metadata-Structure (UMDS) **900**, gemäß Implementierungen. Die UMDS **900** ist ein Beispiel der Datenstruktur, die in einem der Slots in der EVA-Seite **800** gespeichert werden kann. Jede UMDS **900** kann bei bestimmten Offsets eine Anzahl von Feldern speichern. Die UMDS **900** von **Fig. 9** ist ein Beispiel, und andere Implementierungen sind möglich, einschließlich derer, die mehr oder weniger Informationen beinhalten, oder die die Felder unterschiedlich ordnen. Die UMDS **900** kann deshalb einen Versionswert (VERSION), eine Zeilenadresse (LINADDR), die verschlüsselt werden soll, ein Flags-Feld (UMDSFLAGS), ein Sperrfeld (UMDSLock), das auf eine Sperre einer bestimmten Mikroarchitektur verweist (zum Beispiel um sicherzustellen, dass kein Ressourcenkonflikt besteht), eine Gruppe reservierter Bits (RESERVED) und einen für die Datenauthentifizierung der Seite verwendeten Message-Authentication-Code (**MAC**) beinhalten. Es ist zu beachten, dass Tabelle 2 eine Liste beispielhafter Flags für das UMDSFLAGS-Feld beinhaltet.

Tabelle 2

Flag-Name	Beschreibung
R	Bit-Wert von 1 zeigt an, dass die Seite von innerhalb der Enklave gelesen werden kann. Bit-Wert von 0 zeigt an, dass die Seite nicht von innerhalb der Enklave gelesen werden kann.
W	Bit-Wert von 1 zeigt an, dass die Seite von innerhalb der Enklave geschrieben werden kann. Bit-Wert von 0 zeigt an, dass die Seite nicht von innerhalb der Enklave geschrieben werden kann.
X	X: Bit-Wert von 1 zeigt an, dass die Seite von innerhalb der Enklave ausgeführt werden kann.
	Bit-Wert von 0 zeigt an, dass die Seite nicht von innerhalb der Enklave ausgeführt werden kann.
PENDING	Bit-Wert von 1 zeigt an, dass die Seite durch EAUG hinzugefügt worden ist. Bit-Wert von 0 zeigt entweder an, dass die Seite nicht durch EAUG hinzugefügt worden ist, oder dass das PENDING-Bit durch EACCEPT gelöscht wurde.
MODIFIED	Bit-Wert von 1 zeigt an, dass die Seite durch EMODT modifiziert worden ist. Bit-Wert von 0 zeigt entweder an, dass die Seite nicht modifiziert worden ist, oder dass das MODIFIED-Bit durch EACCEPT gelöscht wurde.
PR	Bit-Wert von 1 zeigt an, dass die Seite eine Berechtigungsbeschränkungsoperation erfährt. Bit-Wert von 0 zeigt entweder an, dass die Berechtigungen der Seite nicht beschränkt wurden, oder dass das PR-Bit durch eine andere Anweisung gelöscht worden ist.
B	Hält Wert des während EWBE gesetzten BITVEK B-Bits
PAGE_TYPE	Der Typ von Seite, der die UMDS zugeordnet ist.

[0058] Fig. 10 ist ein Flussdiagramm von Verfahren **1000** für das Weiterleiten eines geblockten Bits (B) zwischen der UMDSINFO und Flags der UMDS während eines Zurückschreibens, auf das später während eines Ladens verwiesen werden kann, um eine Seite für sicheren Speicher wiederherzustellen, gemäß Implementierungen. Das Verfahren **1000** kann durch Verarbeitungslogik ausgeführt werden, die Hardware (zum Beispiel Schaltungen, zugeordnete Logik und/oder programmierbare Logik), Software (zum Beispiel auf einem Computersystem ausführbare Anweisungen, um Hardware-Simulation auszuführen), oder eine Kombination davon beinhalten kann. In einem veranschaulichenden Beispiel kann das Verfahren **1000** durch den Prozessor **110** des Systems **100** von den **Fig. 1-3**, einschließlich des VMM **124**, ausgeführt werden.

[0059] Das Verfahren **1000** kann damit beginnen, dass die Verarbeitungslogik eine EBLOCK-Anweisung ausführt, um den Auslagerungsprozess einer Seite in die sichere Enklave **162** des Systemspeichers **160** zu beginnen (**1010**). Das Verfahren **1000** kann damit fortfahren, dass die Verarbeitungslogik ein Blockstatusbit erzeugt, das einen Status des ersten Blockbits anzeigt (**1020**). Das Verfahren **1000** kann damit fortfahren, dass die Verarbeitungslogik basierend auf dem Blockstatusbit ein zweites geblocktes Bit innerhalb eines ersten Felds einer in dem Systemspeicher gespeicherten Informationsdatenstruktur (zum Beispiel dem BITVEC der UMDSINFO oder Informationsdatenstruktur) setzt (**1030**).

[0060] Unter fortgesetzter Bezugnahme auf **Fig. 10** kann das Verfahren **1000** damit fortfahren, dass die Verarbeitungslogik eine Zurückschreibanweisung (zum Beispiel EBWE) ausführt, um einen Wert des zweiten geblockten Bits in ein geblocktes Bit-Flag einer UMDS zu kopieren, wobei die UMDS in einer Versions-Array-Seite (zum Beispiel einer EVA-Seite) der sicheren Enklave gespeichert werden soll (**1040**). Das Verfahren **1000** kann damit fortfahren, dass die Verarbeitungslogik die Seite von der sicheren Enklave des Systemspeichers zu ungesichertem Systemspeicher oder Festplatte entfernt (zum Beispiel auslagert) (**1050**). Das Verfahren **1000** kann damit fortfahren, dass die Verarbeitungslogik eine Ladeanweisung (zum Beispiel ELDE) ausführt, um die Seite von Festplatte wiederherzustellen (**1060**). Das Verfahren **1000** kann damit fortfahren, dass die Verarbeitungslogik innerhalb eines Eingaberegisters wie RCX auf eine Adresse des Orts der UMDSINFO **700** zugreift (**1070**).

[0061] Das Verfahren **1000** kann damit fortfahren, dass die Verarbeitungslogik bestimmt, ob das USE_UMDS-Bit in dem Bitvektor der UMDSINFO gesetzt ist (**1080**). Wenn das USE_UMDS-Bit gesetzt ist, kann das Verfahren **1000** damit fortfahren, dass die Verarbeitungslogik das erste geblockte Bit der EPCM mit einem Wert des geblockten Bit-Flags, der von der UMDS gelesen wird, lädt (**1085**). Wenn das USE_UMDS-Bit nicht gesetzt ist, (zum Beispiel es ist gelöscht), kann das Verfahren **1000** damit fortfahren, dass die Verarbeitungslogik das erste geblockte Bit der EPCM mit einem Wert des zweiten geblockten Bits, der von der UMDSINFO, zum Beispiel von dem BITVEK **710**, gelesen wird, lädt (**1090**).

[0062] **Fig. 11** ist ein Flussdiagramm von Verfahren **1100** für die Erzeugung einer Versions-Array-Seite (zum Beispiel einer EVA-Seite) vor sicheren Auslagerungsoperationen, gemäß verschiedenen Implementierungen. Das Verfahren **1100** kann durch Verarbeitungslogik ausgeführt werden, die Hardware (zum Beispiel Schaltungen, zugeordnete Logik und/oder programmierbare Logik), Software (zum Beispiel auf einem Computersystem ausführbare Anweisungen, um Hardware-Simulation auszuführen), oder eine Kombination davon beinhalten kann. In einem veranschaulichenden Beispiel kann das Verfahren **1100** durch den Prozessor **110** des Systems **100** von den **Fig. 1-3**, einschließlich durch die sichere Enklavenschaltung **120**, ausgeführt werden. Die Merkmale des Verfahrens **1100** können auf Ausführung einer Enclave-Make-Extended-Version-Array (EMKEVA) -Anweisung bezogen werden, die in der EPC-Seite (EPC_PAGE) einen leeren EVA-Typ erzeugen kann, dessen logische Adresse durch das DS:RCX-Register gegeben ist und die EPCM-Attribute für diese Seite einrichtet. Der Speicherparameter [RCX]EPCPAGE kann Schreibberechtigungen in Bezug auf EPC-Zugriff bereitstellen.

[0063] Das Verfahren **1100** kann damit beginnen, dass die Verarbeitungslogik die Seitenausrichtung der EPC_PAGE, zum Beispiel 4KB-Ausrichtung, in einer Implementierung verifiziert (**1110**). Das Verfahren **1100** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass DS:RCX lese/schreibzugänglich ist (**1120**). Das Verfahren **1100** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass DS:RCX zu einer EPC-Seite auflöst (**1130**). Das Verfahren **1100** kann damit fortfahren, dass die Verarbeitungslogik bestimmt, ob es einen Ressourcenkonflikt für den EPCM-Eintrag für die EPC PAGE gibt (**1140**). Wenn ja, es gibt einen Ressourcenkonflikt, kann das Verfahren **1100** damit fortfahren, dass die Verarbeitungslogik einen Ressourcenkonfliktfehler zurückgibt (**1145**). Wenn in einer Implementierung eine Sperre für den EPCM-Eintrag erfasst wird, und wenn die Verarbeitungslogik nicht dazu in der Lage ist, die Sperre zu erfassen, kann eine EPCM_LOCK_CONFLICT-Ausnahme resultieren.

[0064] Wenn nein, es gibt keinen Ressourcenkonflikt, kann das Verfahren **1100** damit fortfahren, dass die Verarbeitungslogik bestimmt, ob die EPC_PAGE leer ist (**1150**). Wenn die Seite leer ist, endet das Verfahren **1100**, da die EPC_PAGE darauf vorbereitet ist, als eine EVA-Seite zu handeln. Wenn die EPC_PAGE nicht leer ist, kann das Verfahren **1100** damit fortfahren, dass die Verarbeitungslogik die EPC_PAGE löscht, zum Beispiel Einträge der bei DS:RCX platzierten Seite nullt. Das Verfahren **1100** kann damit fortfahren, dass die Verarbeitungslogik eine nicht gültige Enklaven-ID in den EPCM-Eintrag für die EVA-Seite zum Beispiel schreibt, oder die Seite anderweitig als nicht zu jeglicher Enklave gehörend markiert (**1170**).

[0065] In verschiedenen Implementierungen kann, wenn jegliche der obigen Prüfungen fehlschlägt, eine allgemeine Schutzausnahme resultieren. Darüber hinaus kann, wenn ein Seitenfehler bei dem Zugreifen auf einen Speicheroperanden, einschließlich durch EPCM verursachten Fehlers, auftritt, ein Seitenfehlercode resultieren.

[0066] **Fig. 12A** ist ein Blockdiagramm zum Veranschaulichen sicherer Enklaven und Systemspeicherdatenstrukturen während eines sicheren Zurückschreibens, zum Beispiel Ausführens der Enclave-Write-Back-Extended (EWBE) -Anweisung, gemäß Implementierungen. In einer Implementierung kann der Prozessor **110** die EWBE-Anweisung ausführen, um eine sichere Seite in den EPC **162**, die inaktiv geworden ist, auszulagern (zum Beispiel entfernen), wie vorhergehend unter Bezugnahme auf die **Fig. 3-5** diskutiert. Die EWBE kann weniger Zeiger als vorhergehend in der EWB erledigt beinhalten, da sie nicht auf eine einzelne Platzierung in dem Systemspeicher **160** für einen Bereich der Metadaten zeigen muss. Jetzt kann die EVA-Seite **186** jede UMDS beinhalten, die Metadaten für die Auslagerungs-Krypto-Metadaten, die Version und den MAC für eine Seite konsolidieren kann. Zusammenfassend kann die EWBE auf die EVA-Seite **186**, die auszulagernde Subjekt-Enklavenseite **184** und einen Slot für die verschlüsselte Seite **164** in Systemspeicher zeigen.

[0067] **Fig. 12B** ist ein Flussdiagramm von Verfahren **1200** für das Ausführen eines sicheren Zurückschreibens unter Bezugnahme auf die Datenstrukturen von **Fig. 12A**, gemäß einer Implementierung. Das Verfahren **1200** detailliert die Ausführung der EWBE-Anweisung, gemäß einer Implementierung. Das Verfahren **1200** kann durch Verarbeitungslogik ausgeführt werden, die Hardware (zum Beispiel Schaltungen, zugeordnete Logik und/oder programmierbare Logik), Software (zum Beispiel auf einem Computersystem ausführbare An-

weisungen, um Hardware-Simulation auszuführen), oder eine Kombination davon beinhalten kann. In einem veranschaulichenden Beispiel kann das Verfahren **1200** durch den Prozessor **110** des Systems **100** von den **Fig. 1-3**, einschließlich durch die sichere Enklavenschaltung **120**, ausgeführt werden.

[0068] Das Verfahren **1200** kann damit beginnen, dass die Verarbeitungslogik in einem ersten Eingaberegister (zum Beispiel RCX) auf einen ersten Zeiger auf eine virtuelle Adresse einer Seite in der sicheren Enklave zugreift (**1202**). Das Verfahren **1200** kann damit fortfahren, dass die Verarbeitungslogik innerhalb eines zweiten Eingaberegisters (zum Beispiel RDX) auf einen zweiten Zeiger auf eine erweiterte Versions-Array (EVA) -Seite in dem EPC zugreift (**1208**). Das Verfahren **1200** kann damit fortfahren, dass die Verarbeitungslogik innerhalb eines dritten Eingaberegisters auf einen Speicherplatz außerhalb der sicheren Enklave zugreift (**1212**). Das Verfahren **1200** kann damit fortfahren, dass die Verarbeitungslogik einen Message-Authentication-Code (MAC) der Seite unter Verwendung einer zugewiesenen Versionsnummer erzeugt (**1214**).

[0069] Das Verfahren **1200** kann damit fortfahren, dass die Verarbeitungslogik Daten für die Seite unter Verwendung eines geheimen Schlüssels verschlüsselt, um verschlüsselte Daten für die Seite zu erzeugen (**1220**). Das Verfahren **1200** kann damit fortfahren, dass die Verarbeitungslogik die innerhalb der Versions-Array-Seite gespeicherte UMDS mit der Versionsnummer und dem MAC bestückt (**1224**). Das Verfahren **1200** kann damit fortfahren, dass die Verarbeitungslogik die verschlüsselten Daten in den Speicherplatz außerhalb der sicheren Enklave schreibt (**1228**). Das Verfahren **1200** kann damit fortfahren, dass die Verarbeitungslogik Auslagerungs-Krypto-Metadaten für die Seite zu der UMDS schreibt (**1232**). Das Verfahren **1200** kann damit fortfahren, dass die Verarbeitungslogik die Seite von der sicheren Enklave zurück zu ungesichertem Speicher oder der Festplatte entfernt (zum Beispiel auslagert) (**1236**). Das Verfahren **1200** kann damit fortfahren, dass die Verarbeitungslogik, falls erforderlich, die EVA-Seite innerhalb einer verbundenen Hierarchie, die mit mindestens einer zweiten EVA-Seite, die in der sicheren Enklave bleibt, verbunden ist, auslagert (**1240**). In einer Implementierung kann die EVA-Seite durch Verwenden einer VA-Seite ausgelagert werden. Natürlich kann das Verfahren **1200** zusätzliche oder weniger Schritte, die auch in einer unterschiedlichen Reihenfolge ausgeführt werden können, beinhalten.

[0070] Die **Fig. 13A**, **Fig. 13B** und **Fig. 13C** sind ein Flussdiagramm von Verfahren **1300** für das Ausführen eines sicheren Zurückschreibens, zum Beispiel über Ausführung der EWBE-Anweisung, gemäß verschiedenen anderen Implementierungen. Das Verfahren **1300** kann durch Verarbeitungslogik ausgeführt werden, die Hardware (zum Beispiel Schaltungen, zugeordnete Logik und/oder programmierbare Logik), Software (zum Beispiel auf einem Computersystem ausführbare Anweisungen, um Hardware-Simulation auszuführen), oder eine Kombination davon beinhalten kann. In einem veranschaulichenden Beispiel kann das Verfahren **1300** durch den Prozessor **110** des Systems **100** von den **Fig. 1-3**, einschließlich durch die sichere Enklavenschaltung **120**, ausgeführt werden.

[0071] Die EWBE-Anweisung kann eine Seite von einer EPC-Seite zu einer Nicht-EPC-Seite übertragen, einschließlich des Kopierprozesses, der die Seite kryptographisch schützt. Die EWBE-Anweisung kann ausgeführt werden, wenn die aktuelle Privilegstufe Null („0“) ist. die UMDSINFO soll mit der Adresse der SECS für die zu entfernende Seite bestückt werden. Es ist zu beachten dass unter Bezugnahme auf Ressourcen, die Ausführung der EWBE-Anweisung unterstützen, Tabelle 3 auf Speicherparameterinformationen verweist, Tabelle 4 auf temporäre Variablen verweist, die in lokalen Registern gespeichert werden können, und Tabelle 5 auf Fehlercodes verweist, die von verschiedenen Prüfungen oder Zugriffskonflikten resultieren können. In einer Implementierung kann das RAX-Register Fehlercodes (einschließlich GP-Ausnahmeseitenfehler und der Fehlercodes in Tabelle 5) empfangen, das RBX-Register kann die Adresse der UMDSINFO halten, das RCS kann die Adresse einer EPC-Seite halten, und das RDX kann die Adresse eines EVA-Slots für die EVA-Seite halten. Während auf spezifische Werte, Register und Codes verwiesen wird, sind diese beispielhaft und liefern nur eine Implementierung, bei der andere Werte, Register und Codes mit dem gleichen oder ähnlichem Ergebnis angewendet werden können, was allgemein die Ergebnisse des Verfahrens **1200** erzeugt.

Tabelle 3

Speicherparameter	Berechtigungen	Semantik
[RBX]UMDSINFO	RW	Nicht-EPC-Zugriff
[RBX]UMDSINFO.SRCPGE	RW	Nicht-EPC-Zugriff
[RBX]UMDSINFO.UMDS	RW	Nicht-EPC-Zugriff

Speicherparameter	Berechtigungen	Semantik
[RBX]UMDSINFO.SECS	RW	EPC-Zugriff
[RCX]EPCPAGE	RW	EPC-Zugriff
[RDX]EVASLOT	RW	EPC-Zugriff

Tabelle 4

Variablen-Name	Typ	Größe	Beschreibung
TMP_SRCPGE	Speicherseite	4KB	
TMP_SECS	SECS	4KB	
TMP_BPREFCOUNT	UINT64	8B	
TMP_HEADER	MAC-Header	128B	
TMP_VER	UINT64	8B	
TMP_PK	UINT128	16B	Seitenverschlüsselung/MAC-Schlüssel.

Tabelle 5

Fehlercode	Beschreibung
PAGE_NOT_BLOCKED	Wenn Seite nicht als geblockt gekennzeichnet ist
NOT_TRACKED	Wenn EWBE mit ETRACK im Wettbewerb steht
VA_SLOT_OCCUPIED	Versions-Array-Slot enthielt gültigen Eintrag
CHILD_PRESENT	Child-Seite vorhanden während des Versuchs, Enklave auszulagern
RESOURCE_CONFLICT	Ein Konflikt mit anderer Ressource trat auf

[0072] Unter Bezugnahme auf die **Fig. 13A**, **Fig. 13B** und **Fig. 13C** und auf die Tabellen 3-5 kann das Verfahren **1300** damit beginnen, dass die Verarbeitungslogik Ausrichtung der UMDSINFO (oder Informationsdatenstruktur), zum Beispiel für 32B-Ausrichtung, verifiziert und eine GP-Ausnahme erzeugt, wenn nicht (**1302**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass die UMDSINFO lesezugänglich ist (**1304**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass die UMDSINFO.SECS 4K-ausgerichtet ist, und eine GP-Ausnahme erzeugt, wenn nicht (**1306**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass die SECS lese-/schreibzugänglich ist (**1308**).

[0073] Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik Ausrichtung einer bestimmten EPC-Seite (nachfolgend EPC_PAGE) verifiziert, zum Beispiel ausgerichtet auf eine 4KB-Seite, und eine GP-Ausnahme erzeugt, wenn nicht so ausgerichtet (**1310**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass das DS:RCX lese-/schreibzugreifbar ist (**1312**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass die EPC_PAGE (bei DS:RCX) in eine physikalische Adresse, die eine EPC-Seite ist, auflöst, und einen Seitenfehler erzeugt, wenn sie das nicht tut (**1314**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass die UMDS 64-Byte-ausgerichtet ist, und eine GP-Ausnahme erzeugt, wenn nicht (**1316**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass DS:RDX lese-/schreibzugänglich ist (**1318**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass die UMDS in eine physikalische Adresse innerhalb des EPC auflöst, und einen Seitenfehler erzeugt, wenn nicht (**1320**).

[0074] Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik eine virtuelle Adresse in der UMDSINFO für Ausrichtungs- und Zugänglichkeitsprüfungen extrahiert (**1322**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik reservierte Bits in den BITVEK-Feldern der UMDSINFO verifiziert (**1324**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik Ausrichtung einer Quellenseite (TMP_SRCPGE), zum Beispiel 4KB-ausgerichtet, verifiziert, und in einer GP-Ausnahme resultiert, wenn nicht (**1326**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik Ausrichtung der Quellenseite

(DS:TMP_SRCPGE) verifiziert (**1328**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass DS:TMP_SCRCPGE lese-/schreibzugänglich ist (**1330**).

[0075] Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik bestimmt, ob es einen Ressourcenkonflikt mit dem EPCM-Eintrag für EPC_PAGE gibt (**1332**). Wenn es einen Konflikt gibt, kann das Verfahren **1300** damit fortfahren, dass die Verarbeitungslogik einen Ressourcenkonfliktfehler, der eine Sperrkonfliktausnahme beinhalten kann, wenn eine Sperre versucht wurde, zurückgibt (**1334**). Wenn es keinen Konflikt gibt, kann das Verfahren **1300** damit fortfahren, dass die Verarbeitungslogik bestimmt, ob es einen Lesekonflikt für den EPCM-Eintrag für die in einem Slot der EVA-Seite gespeicherte UMDS gibt (**1336**). Wenn dies eine Sperre ist, sollte der VMM die EVA nicht mit einen Guest teilen, sodass kein VMEXIT benötigt werden kann. Wenn es einen Lesekonflikt mit dem EPCM-Eintrag für die UMDS gibt, kann das Verfahren **1300** in einem Ressourcenkonfliktfehler, der in einer Implementierung ein EPC_PAGE_CONFLICT-Fehler ist, resultieren (**1338**).

[0076] Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass die EPC PAGE und die EVA-Seite gültige EPC-Seiten sind (**1340**), und dass DS:RDX EVA-Seitentyp ist (**1342**), und einen Seitenfehler zurückgeben, wenn nicht. Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik bestimmt, ob es einen Ressourcenkonflikt eines UMDS-Eintrags in der EVA-Seite gibt (**1344**). Wenn es eine Sperre ist, sollte der VMM die EVA-Seite nicht mit einen Guest teilen, sodass kein VMEXIT benötigt werden kann. Wenn es einen Ressourcenkonflikt mit dem UMDS-Eintrag gibt, kann das Verfahren **1300** damit fortfahren, dass die Verarbeitungslogik einen Ressourcenfehler, zum Beispiel einen EPC_PAGE_CONFLICT-Fehler, der das Setzen des ZF-Flags beinhalten kann, zurückgibt (**1346**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik eine seitentypspezifische Ausnahmeprüfung ausführt, die eine Prüfung beinhalten kann, dass die in der UMDINFO.SECS übergebene SECS mit dem Back-Pointer auf die SECS übereinstimmt, und eine GP-Schutzausnahme erzeugt, wenn nicht (**1348**).

[0077] Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik Epochenwerte (zum Beispiel TMP_PBEPOCH und TMP_PBREFCOUNT) in lokale Epochenregister liest (**1350**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik den TMP_HEADER von SECINFO initialisiert, und SECINFO in dem Feld lässt, um Validierung zu reduzieren (**1352**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik seitentypspezifische Prüfungen ausführt (**1354**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik bestimmt, ob die Seite geblockt ist und somit räumbar ist (**1356**). Wenn die Seite nicht geblockt ist, kann das Verfahren **1300** damit fortfahren, dass die Verarbeitungslogik einen PAGE_NOT_ BLOCKED-Fehler zurückgibt (**1358**). Wenn die Seite geblockt ist, kann das Verfahren **1300** damit fortfahren, dass die Verarbeitungslogik bestimmt, ob die Seite verfolgt wird und somit räumbar ist (**1360**). Wenn die Seite nicht verfolgt wird, kann das Verfahren **1300** damit fortfahren, dass die Verarbeitungslogik einen PAGE_NOT_ TRACKED-Fehler zurückgibt (**1362**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik eine Enklaven-ID erhält, um eine kryptographische Bindung zwischen ausgelagerter Seite und der Enklave einzurichten (**1364**).

[0078] Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik bestimmt, ob es innerhalb der Enklave noch jegliche Child-Seiten gibt (**1366**). Wenn es welche gibt, kann das Verfahren **1300** damit fortfahren, dass die Verarbeitungslogik einen CHILD_PRESENT-Fehler zurückgibt (**1368**). Wenn es keine weiteren Child-Seiten innerhalb der Enklave gibt, kann das Verfahren **1300** damit fortfahren, dass die Verarbeitungslogik die SECS als eine Child-Seite besitzend behandelt, wenn die virtuelle Child-Anzahl nicht Null ist (**1370**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik eine Version, zum Beispiel einen TMP_ VER-Wert, in die Seite schreibt (**1372**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik Daten der Seite verschlüsselt (**1374**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik die verschlüsselten Daten und den **MAC** in Bestimmungsortpuffer schreibt(**1376**).

[0079] Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik bestimmt, ob ein Fehler in dem Schreiben des Blocks **1376** erkannt wird (**1378**). Wenn ein Fehler erkannt wird, kann das Verfahren **1300** damit fortfahren, dass die Verarbeitungslogik Datenstrukturzugriffe freigibt und die EWBE-Anweisung abbricht (**1380**). Wenn kein Fehler erkannt wird, kann das Verfahren **1300** damit fortfahren, dass die Verarbeitungslogik den Rest des UMDS-Status (zum Beispiel Auslagerungs-Krypto-Metadaten) von der EPCM schreibt (**1382**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik bestimmt, ob ein Fehler in dem Schreiben des Blocks **1382** erkannt wird (**1384**). Wenn ein Fehler erkannt wird, kann das Verfahren **1300** damit fortfahren, dass die Verarbeitungslogik Datenstrukturzugriffe freigibt und die EWBE-Anweisung abbricht (**1386**). Wenn kein Fehler erkannt wird, kann das Verfahren **1300** damit fortfahren, dass die Verarbeitungslogik die Child-Anzahl in der SECS (**1388**) verringert. Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik einen vorherigen Versions (PREV _VER) -Wert wieder abruft und ihn in der UMDS(DS:RDX).VERSION

speichert (**1390**) Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik die Version in den EVA-Slot schreibt, zum Beispiel UMDS(DS:RDX).VERSION wird in TEMP_VER geschrieben (**1392**).

[0080] Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik bestimmt, ob der EVA-Slot belegt ist, bevor sie die Version schreibt (**1394**). Wenn der EVA-Slot belegt ist, kann das Verfahren **1300** damit fortfahren, dass die Verarbeitungslogik einen EVA_SLOT_OCCUPIED-Fehler zurückgibt (**1396**). Das Verfahren **1300** kann damit fortfahren, dass die Verarbeitungslogik einen EPC-Eintrag freigibt, zum Beispiel EPCM (DS: RCX).VALID wird genullt (**1398**). Jegliche verbleibenden Ressourcen können ebenfalls freigegeben werden. Darüber hinaus kann das ZF-Flag gesetzt werden, wenn die Seite nicht geblockt ist, die Seite nicht verfolgt wird, es einen Konflikt mit Status gibt, oder ein Child vorhanden ist; andernfalls wird das ZF-Flag gelöscht. Das CF-Flag kann darüber hinaus gesetzt werden, wenn der EVA-Slot vorhergehend belegt ist; ansonsten wird das CF-Flag gelöscht. Das RAX kann auf den Fehlercode gesetzt werden.

[0081] **Fig. 14A** ist ein Blockdiagramm zum Veranschaulichen sicherer Enklaven und Systemspeicherdatenstrukturen während eines sicheren Ladens, zum Beispiel Ausführung der **ELDE**-Anweisung, gemäß Implementierungen. Es ist zu beachten, dass erneut die Anzahl von Zeigern in der Fähigkeit, die **ELDE**-Anweisung auszuführen, verglichen mit Ausführung der ELD-Anweisung reduziert wurde. Hier gibt es keinen zusätzlichen Zeiger auf in dem Systemspeicher **160** gespeicherte Metadaten, um die Metadaten zu kombinieren, wenn eine Seite von Festplatte zu sicherem Speicher wiederhergestellt wird. Zusammenfassend kann die **ELDE**-Anweisung einen Zeiger auf die SECS **182**, einen Zeiger auf die EVA-Seite **186**, einen Zeiger auf einen Slot, in den die Enklavenseite **184** geladen werden soll, und einen Zeiger auf die verschlüsselte Seite **164** in dem Systemspeicher **160** wieder abrufen.

[0082] **Fig. 14B** ist Flussdiagramm von Verfahren **1400** für das Ausführen eines sicheren Ladens, zum Beispiel Ausführen der **ELDE**-Anweisung, unter Bezugnahme auf die Datenstrukturen von **Fig. 14A**, gemäß einer Implementierung. Das Verfahren **1400** kann durch Verarbeitungslogik ausgeführt werden, die Hardware (zum Beispiel Schaltungen, zugeordnete Logik und/oder programmierbare Logik), Software (zum Beispiel auf einem Computersystem ausführbare Anweisungen, um Hardware-Simulation auszuführen), oder eine Kombination davon beinhalten kann. In einem veranschaulichenden Beispiel kann das Verfahren **1400** durch den Prozessor **110** des Systems **100** von den **Fig. 1-3**, einschließlich durch die sichere Enklavenschaltung **120**, ausgeführt werden.

[0083] Das Verfahren **1400** kann damit beginnen, dass die Verarbeitungslogik in einem Ausgaberegister auf einen ersten Zeiger auf die verschlüsselte Seite an dem Speicherplatz zugreift (**1402**). Das Verfahren **1400** kann damit fortfahren, dass die Verarbeitungslogik in einem Eingaberegister (zum Beispiel RCX) auf einen zweiten Zeiger auf einen freien Seitenplatz in der sicheren Enklave (EPC) zugreift (**1408**). Das Verfahren **1400** kann damit fortfahren, dass die Verarbeitungslogik in einem zweiten Eingaberegister auf den dritten Zeiger auf eine virtuelle Adresse eines Versions-Array-Slots (für die EVA-Seite) in der sicheren Enklave zugreift (**1412**). Das Verfahren **1400** kann damit fortfahren, dass die Verarbeitungslogik die virtuelle Adresse in eine physikalische Guest-Adresse und die physikalische Guest-Adresse in eine physikalische Host-Adresse für die Versions-Array-Seite übersetzt (**1416**).

[0084] Das Verfahren **1400** kann damit fortfahren, dass die Verarbeitungslogik auf die UMDS innerhalb der EVA-Seite an der physikalischen Host-Adresse des Versions-Array-Slots zugreift (**1420**) Das Verfahren **1400** kann damit fortfahren, dass die Verarbeitungslogik die verschlüsselte Seite durch Verwenden des **MAC** und der Versionsnummer verifiziert (**1424**). Das Verfahren **1400** kann damit fortfahren, dass die Verarbeitungslogik die verschlüsselten Daten entschlüsselt, um entschlüsselte Daten für die Seite zu erzeugen (**1428**). Das Verfahren **1400** kann damit fortfahren, dass die Verarbeitungslogik die entschlüsselten Daten in der sicheren Enklave an dem freien Speicherplatz speichert (**1432**). Das Verfahren **1400** kann damit fortfahren, dass die Verarbeitungslogik einen Slot innerhalb der UMDS für einen nächsten Versionswert löscht (**1436**).

[0085] Die **Fig. 15A** und **Fig. 15B** sind ein Flussdiagramm von Verfahren **1500** für das Ausführen eines sicheren Ladens, zum Beispiel Ausführung des **ELDE**, gemäß verschiedenen anderen Implementierungen. Das Verfahren **1500** kann durch Verarbeitungslogik ausgeführt werden, die Hardware (zum Beispiel Schaltungen, zugeordnete Logik und/oder programmierbare Logik), Software (zum Beispiel auf einem Computersystem ausführbare Anweisungen, um Hardware-Simulation auszuführen), oder eine Kombination davon beinhalten kann. In einem veranschaulichenden Beispiel kann das Verfahren **1500** durch den Prozessor **110** des Systems **100** von den **Fig. 1-3**, einschließlich durch die sichere Enklavenschaltung **120**, ausgeführt werden.

[0086] Die ELDE-Anweisung kann eine Seite von ungeschütztem Speicher auf eine EPC-Seite kopieren. Als Teil des Kopierens kann die Seite kryptographisch authentifiziert und entschlüsselt werden. Diese Anweisung kann ausgeführt werden, wenn die aktuelle Privilegstufe Null („0“) ist. Es ist zu beachten dass unter Bezugnahme auf Ressourcen, die Ausführung der EWBE-Anweisung unterstützen, Tabelle 6 auf Speicherparameterinformationen verweist, Tabelle 7 auf temporäre Variablen verweist, die in lokalen Registern gespeichert werden können, und Tabelle 8 auf Fehlercodes verweist, die aus verschiedenen Prüfungen oder Zugriffskonflikten resultieren können. Darüber hinaus kann das Register RBX die virtuelle Adresse der UMDSINFO-Datenstruktur halten, das RCX kann die virtuelle Adresse einer EPC-Seite halten, und das RDX kann die logische Adresse eines EVA-Slots halten. Während auf spezifische Werte, Register und Codes verwiesen wird, sind diese beispielhaft und liefern nur eine Implementierung, wobei andere Werte, Register und Codes mit dem gleichen oder ähnlichem Ergebnis angewendet werden können, was allgemein die Ergebnisse des Verfahrens **1400** erzeugt.

Tabelle 6

Speicherparameter	Berechtigungen	Semantik
[RBX]UMDSINFO	R	Nicht-EPC-Zugriff
[RBX]UMDSINFO.SRCPGE	R	Nicht-EPC-Zugriff
[RBX]UMDSINFO.UMDS	R	Nicht-EPC-Zugriff
[RBX]UMDSINFO.SECS	RW	EPC-Zugriff
[RCX]EPCPAGE	RW	EPC-Zugriff
[RDX]EVASLOT	RW	EPC-Zugriff

Tabelle 7

Variablen-Name	Typ	Größe	Beschreibung
TMP_SRCPGE	Speicherseite	4KB	
TMP_SECS	Speicherseite	4KB	
TMP_HEADER	MACHEADER	128B	
TMP_VER	UNIT64	8B	
TMP_MAC	UINT128	16B	
TMP_PK	UINT128	16B	Seitenverschlüsselung/MAC-Schlüssel.

Tabelle 8

Fehlercode	Beschreibung
MAC_COMPARE_FAIL	Wenn die MAC-Prüfung fehlschlägt
RESOURCE_CONFLICT	Fehlschlag, eine der Sperren oder einen Ressourcenkonflikt zu erfassen (ELDBC/ELDUC)

[0087] Unter Bezugnahme auf die **Fig. 15A** und **Fig. 15B** kann das Verfahren **1500** damit beginnen, dass die Verarbeitungslogik Ausrichtung der UMDSINFO-Datenstruktur, zum Beispiel, dass sie 32B-ausgerichtet ist, verifiziert, ansonsten eine GP-Ausnahme zurückgibt (**1502**). Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass die UMDSINFO lesezugänglich ist (**1504**). Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik die Ausrichtung einer bestimmten EPC-Seite, nachfolgend „EPC_PAGE“, verifiziert (**1506**). Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass der Wert in dem DS:RCX-Register lese-/schreibzugänglich ist (**1508**). Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass die EPC PAGE in eine physikalische Host-Adresse innerhalb des EPC auflöst, ansonsten einen Seitenfehler zurückgibt (**1510**).

[0088] Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass der Slot der EVA-Seite in der EPC **162** 64-Byte-ausgerichtet ist, oder eine andere geeignete Ausrichtung hat (**1512**). Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass Register DS:RDX lese-/

schreibzugänglich ist (**1514**). Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass die EVA-Seite in eine physikalische Host-Adresse innerhalb des EPC auflöst, ansonsten einen Seitenfehler zurückgibt (**1516**). Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass die Ausrichtung von UMDSINFO-verbundenen Parametern in dem RBX-Register 4KB-ausgerichtet sind, ansonsten eine GP-Ausnahme zurückgibt (**1518**). Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass DS_TMP_SRCPGE lesezugänglich ist (**1520**). Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass die Quelle und der Bestimmungsort für die EPC_PAGE nicht überlappen (**1522**).

[0089] Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik bestimmt, ob es einen Ressourcenkonflikt eines EPCM-Eintrags für die EPC_PAGE gibt (**1524**). Wenn es einen Ressourcenkonflikt gibt, kann das Verfahren **1500** damit fortfahren, dass die Verarbeitungslogik einen Ressourcenkonfliktfehler, der zum Beispiel ein EPC_PAGE_CONFLICT-Fehler in einer Implementierung sein kann, zurückgibt (**1526**). Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik bestimmt, ob es einen Ressourcenkonflikt eines EPCM-Eintrags für die UMDS gibt (**1526**). Wenn es einen Ressourcenkonflikt gibt, kann das Verfahren **1500** damit fortfahren, dass die Verarbeitungslogik einen Ressourcenkonfliktfehler, der ein EPC_PAGE_CONFLICT-Fehler in einer Implementierung sein kann, zurückgibt (**1528**). Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik die EPCM-Attribute der EPC PAGE, der EVA-Seite und der DEST-Seite, zum Beispiel der Bestimmungsort-EPC-Seite für die Ladeoperation, verifiziert und einen Seitenfehler erzeugt, wenn ein Attribut nicht in der Lage ist, verifiziert zu werden (**1530**).

[0090] Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik bestimmt, ob es einen Ressourcenkonflikt des UMDS-Eintrags in der EVA-Seite gibt (**1532**). Wenn es einen Ressourcenkonflikt gibt, kann das Verfahren **1500** damit fortfahren, dass die Verarbeitungslogik einen Ressourcenkonfliktfehler, der ein EPC_PAGE_CONFLICT-Fehler in einer Implementierung sein kann, erzeugt (**1534**). Zusätzlich zu dem Fehler kann in einer Implementierung auch ein Bit in dem ZF-Flag gesetzt werden. Der VMM sollte die EVA-Seite nicht mit dem Guest teilen. Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik an dem TMP-Header zum Beispiel den TMP_HEADER-Wert nullt (**1535**). Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik Attribute des SECINFO-Parameters verifiziert, und eine GP-Ausnahme erzeugt, wenn nicht verifiziert (**1536**). Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass die DS_TMP_SECS lesezugänglich ist (**1538**).

[0091] Unter zusätzlicher Bezugnahme auf **Fig. 15B** kann das Verfahren **1500** damit fortfahren, dass die Verarbeitungslogik verifiziert, dass DS:RBX.SECS eine Adresse einer EPC-Seite ist (**1540**). Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik bestimmt, ob es einen Ressourcenkonflikt eines SECS-Eintrags in der EPCM gibt (**1542**). Wenn es einen Ressourcenkonflikt gibt, kann das Verfahren **1500** damit fortfahren, dass die Verarbeitungslogik einen Ressourcenkonfliktfehler, der ein EPC_PAGE_CONFLICT-Fehler sein kann, erzeugt, wenn eine Sperre des EPCM-Eintrags versucht wurde (**1544**). Zusätzlich zu dem Fehler kann in einer Implementierung auch ein Bit in dem ZF-Flag gesetzt werden. Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik verifiziert, dass die SECS gültig ist, zum Beispiel eine SECS-Seite ist, und einen Seitenfehler erzeugt, wenn nicht (**1546**). Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik Daten für die Seite durch Verwenden des MAC validiert (**1548**).

[0092] Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik sicherstellt, dass der UMDS-Eintrag in der EVA-Seite nicht wiederverwendet werden kann, was zum Beispiel durch Löschen eines Versionswerts der UMDS ausgeführt werden kann (**1550**). Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik Änderungen zu der EPCM vornimmt (**1552**). Solche Änderungen können über den in temporären Steuerregistern gespeicherten temporären (TMP) Wert gemacht werden (**1552**). Um dieses Vornehmen von Änderungen auszuführen, kann die Verarbeitungslogik der Seite DS:TMP_SECS durch den EPCM-Back-Pointer für die Seite zuordnen. Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik einen Back-Pointer zu der SECS in das ENCLAVESECS-Feld der EPCM schreibt (**1554**). Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik das geblockte Bit in der EPCM setzt (**1556**). Wenn der TMP_HEADER.SECINFO.FLAGS.PT kein SECS-, VA- oder EVA-Seitentyp ist, kann das geblockte Bit auf TMP_HEADER gesetzt werden. SECINFO.FLAGS.B; Andernfalls kann das geblockte Bit in der EPCM auf Null („0“) gesetzt werden.

[0093] Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik sicherstellt, dass der BEPOCH-Wert auf Null gesetzt ist (**1558**). Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik jegliche verbleibenden Sperren freigibt, in dem Fall, dass Sperren verwendet wurden, um Ressourcenkonflikte zu verhindern (**1560**). Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik RAX und das ZF-

Flag setzt, um anzuzeigen, dass eine erfolgreiche Ladeoperation abgeschlossen wurde (**1564**). Das Verfahren **1500** kann damit fortfahren, dass die Verarbeitungslogik andere Flags, die wegen des erfolgreichen Abschlusses nicht länger in Verwendung sind, löscht.

[0094] **Fig. 16A** ist ein Blockdiagramm zum Veranschaulichen einer Mikroarchitektur für Prozessor **1600**, das verbessertes sicheres Auslagern implementiert, gemäß einer Implementierung. Insbesondere stellt der Prozessor **1600** einen In-Order-Architekturkern und eine Registerumbenennungslogik, Out-of-Order-Ausgabe-/Ausführungslogik dar, die in einem Prozessor gemäß mindestens einer Implementierung der Offenbarung enthalten sein soll. In einer Implementierung ist der Prozess **1600** der gleiche wie der mit Bezug auf **Fig. 1** beschriebene Prozessor **100**.

[0095] Der Prozessor **1600** beinhaltet mit Ausführungs-Engine-Einheit **1650** gekoppelte Front-End-Einheit **1630**, und beide sind mit Speichereinheit **1670** gekoppelt. Der Prozessor **1600** kann einen Reduced-Instruction-Set-Computing (RISC) -Kern, einen Complex-Instruction-Set-Computing (CISC) -Kern, einen Very-Long-Instruction-Word (VLIW) -Kern oder einen hybriden oder alternativen Kerntyp beinhalten. Als noch eine weitere Option kann der Prozessor **1600** einen Spezialkern, wie zum Beispiel einen Netzwerk- oder Kommunikationskern, eine Komprimierungs-Engine, einen Grafikkern oder dergleichen beinhalten. In einer Implementierung kann der Prozessor **1600** ein Multicore-Prozessor sein, oder kann Teil eines Multiprozessorsystems sein.

[0096] Die Front-End-Einheit **1630** beinhaltet Sprungvorhersageeinheit **1632**, die mit Anweisungs-Cache-Einheit **1634** gekoppelt ist, die mit Translation-Lookaside-Buffer (TLB) **1636** gekoppelt ist, der mit Anweisungsabrufeinheit **1638** gekoppelt ist, die mit Decodierungseinheit **1640** gekoppelt ist. Die Decodierungseinheit **1640** (auch bekannt als ein Decoder) kann Anweisungen decodieren und als eine Ausgabe eine oder mehrere MikroOperationen, Mikrocode-Eintragungspunkte, Mikroanweisungen, andere Anweisungen oder andere Steuersignale erzeugen, die von den ursprünglichen Anweisungen decodiert werden oder diese auf andere Weise widerspiegeln oder davon abgeleitet werden. Der Decoder **1640** kann durch Verwenden verschiedener unterschiedlicher Mechanismen implementiert werden. Beispiele für geeignete Mechanismen beinhalten Nachschlagetabellen, Hardwareimplementierungen, Programmable-Logic-Arrays (PLA), Mikrocode-Read-Only-Memories (ROM), usw., sind jedoch nicht darauf beschränkt. Die Anweisungs-Cache-Einheit **1634** ist ferner mit der Speichereinheit **1670** gekoppelt. Die Decodierungseinheit **1640** ist mit Umbenennungs-/Zuordnungseinheit **1652** in der Ausführungs-Engine-Einheit **1650** gekoppelt.

[0097] Die Ausführungs-Engine-Einheit **1650** beinhaltet die Umbenennungs-/Zuordnungseinheit **1652**, die mit Rückzugseinheit **1654** gekoppelt ist, und einen Satz einer oder mehrerer Planereinheit(en) **1656**. Die Planereinheit(en) **1656** stellt/stellen jegliche Anzahl von verschiedenen Planerschaltungen dar, einschließlich Reservierungsstationen (RS), zentralem Anweisungsfenster, usw. Die Planereinheit(en) **1656** ist/sind mit physikalischer Registersatzeinheit(en) **1658** gekoppelt. Jede der physikalischen Registersatzeinheit(en) **1658** stellt/stellen einen oder mehrere physikalische Registersätze dar, von denen unterschiedliche einen oder mehrere unterschiedliche Datentypen, wie zum Beispiel skalare Ganzzahl, skalares Fließkomma, gepackte Ganzzahl, gepacktes Fließkomma, Vektor-Ganzzahl, Vektorfließkomma, usw., Status (zum Beispiel einen Anweisungszeiger, der die Adresse der nächsten auszuführenden Anweisung ist), usw., speichern. Die physikalische Registersatzeinheit(en) **1658** wird/werden von der Rückzugseinheit **1654** überlappt, um verschiedene Möglichkeiten zu veranschaulichen, in denen das Registerumbenennen und die Out-of-Order-Ausführung implementiert werden können (zum Beispiel durch Verwenden von Umordnungspuffer(n) und eines Satzes/Sätzen von Rückzugsregistern, durch Verwenden zukünftiger Datei(en), von Historienpuffer(n), und eines Satzes/Sätzen von Rückzugsregistern; durch Verwenden von Registerkarten und eines Pools von Registern; usw.).

[0098] Allgemein sind die architekturellen Register von außerhalb des Prozessors oder von einer Perspektive eines Programmierers sichtbar. Die Register sind nicht auf jeglichen bekannten bestimmten Typ von Schaltung beschränkt. Verschiedene unterschiedliche Typen von Registern sind geeignet, solange sie für das Speichern und Bereitstellen von Daten wie hierin beschrieben in der Lage sind. Beispiele geeigneter Register beinhalten zugeordnete physikalische Register, dynamisch zugewiesene physikalische Register durch Verwenden von Registerumbenennung, Kombinationen von zugeordneten und dynamisch zugewiesenen physikalischen Registern usw., sind jedoch nicht darauf beschränkt. Die Rückzugseinheit **1654** und die Einheit(en) des physikalischen Registersatzes (der physikalischen Registersätze) **1658** sind mit Ausführungs-Cluster(n) **1660** gekoppelt. Das/die Ausführungs-Cluster **1660** beinhalten einen Satz einer oder mehrerer Ausführungseinheiten **1662** und einen Satz einer oder mehrerer Speicherzugriffseinheiten **1664**. Die Ausführungseinheiten **1662** können verschiedene Operationen (zum Beispiel Verschiebungen, Addition, Subtraktion, Multiplikation) ausführen, und auf verschiedenen Typen von Daten (zum Beispiel skalares Fließkomma, gepackte Ganzzahl, gepacktes Fließkomma, Vektorganzzahl, Vektorfließkomma) operieren.

[0099] Während einige Implementierungen eine Anzahl von Ausführungseinheiten, die spezifischen Funktionen oder Sätzen von Funktionen zugeordnet sind, beinhalten können, können andere Implementierungen nur eine Ausführungseinheit oder mehrere Ausführungseinheiten, die alle alle Funktionen ausführen, beinhalten. Die Planereinheit(en) **1656**, die physikalischen Registersatzeinheit(en) **1658** und die Ausführungs-Cluster **1660** sind als möglicherweise eine Mehrzahl gezeigt, weil bestimmte Implementierungen separate Pipelines für bestimmte Typen von Daten/Operationen erzeugen (zum Beispiel eine skalare Ganzzahl-Pipeline, eine skalare Fließkomma-/gepackte Ganzzahl-/gepacktes Fließkomma-/Vektorganzzahl-/Vektorfließkomma-Pipeline und/oder eine Speicherzugriffs-Pipeline, die jeweils ihre eigene Planereinheit, physikalische Registersatzeinheit (en) und/oder Ausführungs-Cluster haben - und in dem Fall einer separaten Speicherzugriffs-Pipeline sind bestimmte Implementierungen implementiert, in denen nur das Ausführungs-Cluster dieser Pipeline die Speicherzugriffseinheit(en) **1664** hat). Es sollte auch klargestellt werden, dass, wo separate Pipelines verwendet werden, eine oder mehrere dieser Pipelines Out-of-Order-Ausgabe-/Ausführungs-Pipelines und der Rest In-Order-Pipelines sein können.

[0100] Der Satz von Speicherzugriffseinheiten **1664** ist mit der Speichereinheit **1670** gekoppelt, die Datenvorababholeinheit **1680**, TLB-Einheit **1672**, Data-Cache-Unit (DCU) **1674** und Level **2** (**L2**)-Cache-Einheit **1676** beinhalten kann, um einige Beispiele zu nennen. In einigen Implementierungen ist die DCU **1674** auch als ein Daten-Cache des ersten Levels (**L1** Cache) bekannt. Die DCU **1674** kann mehrere ausstehende Cache-Fehlschläge handhaben und damit fortfahren, ankommende Speicherungen und Ladungen zu bedienen. Sie unterstützt auch das Aufrechterhalten von Cache-Kohärenz. Die Daten-TLB-Einheit **1672** ist ein Cache, der dazu verwendet wird, die Übersetzungsgeschwindigkeit für virtuelle Adressen durch Abbilden virtueller und physikalischer Adressräume zu verbessern. In einer beispielhaften Implementierung können die Speicherzugriffseinheiten **1664** eine Ladeeinheit, eine Adressspeichereinheit und eine Datenspeichereinheit beinhalten, von denen jede mit der Daten-TLB-Einheit **1672** in der Speichereinheit **1670** gekoppelt ist. Die **L2**-Cache-Einheit **1676** kann mit einem oder mehreren anderen Cache-Levels und schließlich mit einem Hauptspeicher gekoppelt sein.

[0101] In einer Implementierung lädt der Datenvorababrufer **1680** Daten spekulativ in die DCU **1674**, bzw. ruft diese vorab ab, indem er automatisch vorhersagt, welche Daten ein Programm gerade verbraucht. Vorababrufen kann sich auf das Übertragen von Daten beziehen, die an einem Speicherplatz (zum Beispiel Position) einer Speicherhierarchie (zum Beispiel Caches oder Speicher niedrigerer Ebene) an einen Speicherplatz höherer Ebene gespeichert sind, der näher an dem Prozessor liegt (zum Beispiel niedrigere Zugriffslatenz erbringt), bevor die Daten tatsächlich durch den Prozessor angefordert werden. Insbesondere kann sich das Vorababrufen auf das frühe Wiederabrufen von Daten von einem der Caches/Speicher der niedrigeren Ebene in einen Datencache und/oder einen Vorababrufpuffer beziehen, bevor der Prozessor eine Anforderung für die spezifischen zurückzugebenden Daten ausgibt.

[0102] Der Prozessor **1600** kann einen oder mehrere Anweisungssätze unterstützen (zum Beispiel den x86-Anweisungssatz (mit einigen Erweiterungen, die mit neueren Versionen hinzugefügt worden sind); den MIPS-Anweisungssatz von Imagination Technologies in Kings Langley, Hertfordshire, UK; den ARM-Anweisungssatz (mit optional zusätzlichen Erweiterungen, wie zum Beispiel NEON) von ARM Holdings in Sunnyvale, CA).

[0103] Es sollte klargestellt werden, dass der Kern Multi-Threading (Ausführen von zwei oder mehr parallelen Sätzen von Operationen oder Threads) unterstützen kann und dies in einer Vielfalt von Weisen tun kann, einschließlich Zeitscheiben-Multi-Threading, gleichzeitiges Multi-Threading (wobei ein einzelner physikalischer Kern einen logischen Kern für jeden der Threads bereitstellt, für die dieser physikalische Kern ein gleichzeitiges Multi-Threading durchführt) oder einer Kombination davon (zum Beispiel Zeitscheiben-Abrufen und -Decodieren und gleichzeitiges Multi-Threading danach, wie zum Beispiel in der Intel® Hyperthreading-Technologie).

[0104] Während die Registerumbenennung in dem Zusammenhang der Out-of-Order-Ausführung beschrieben ist, sollte klargestellt werden, dass Registerumbenennung in einer In-Order-Architektur verwendet werden kann. Während die veranschaulichte Implementierung des Prozessors auch separate Anweisungs- und Data-Cache-Units und eine geteilte **L2**-Cache-Einheit beinhaltet, können alternative Implementierungen einen einzelnen internen Cache sowohl für Anweisungen, als auch für Daten haben, wie zum Beispiel einen internen Level **1** (**L1**) -Cache, oder mehrere Level interner Caches. In einigen Implementierungen kann das System eine Kombination von einem internen Cache und einem externen Cache, der außerhalb des Kerns und/oder des Prozessors ist, beinhalten. Alternativ kann der gesamte Cache außerhalb des Kerns und/oder des Prozessors sein.

[0105] **Fig. 16B** ist ein Blockdiagramm zum Veranschaulichen einer On-Order-Pipeline und einer Registerumbenennungsstufe, Out-of-Order-Ausgabe-/Ausführungs-Pipeline, die durch den Prozessor **1600** von **Fig. 16A** implementiert wird, der Hardware-Unterstützung für verbessertes sicheres Auslagern implementiert Die Kästchen mit durchgezogenen Linien in **Fig. 16B** veranschaulichen In-Order-Pipeline **1601**, während die Kästchen mit gestrichelten Linien Registerumbenennungs-, Out-of-Order-Ausgabe/Ausführungs-Pipeline **1603** veranschaulichen. In **Fig. 16B** beinhalten die Pipelines **1601** und **1603** Abrufstufe **1602**, Längendecodierungsstufe **1604**, Decodierungsstufe **1606**, Zuweisungsstufe **1608**, Umbenennungsstufe **1610**, Planungsstufe (auch bekannt als Sende- oder Ausgabe-Stufe) **1612**, Registerlese-/Speicherlese-Stufe **1614**, Ausführungsstufe **1616**, Zurückschreib-/Speicherschreib-Stufe **1618**, Ausnahmehandhabungsstufe **1622** und Festlegungsstufe **1624**. In einigen Implementierungen kann das Ordnen der Stufen **1602**-**1624** unterschiedlich als veranschaulicht sein und ist nicht auf das in **Fig. 16B** gezeigte spezifische Ordnen beschränkt.

[0106] **Fig. 17** veranschaulicht ein Blockdiagramm der Mikroarchitektur für Prozessor **1700**, der Logikschaltungen eines Prozessors oder eine integrierte Schaltung beinhaltet, die erweitertes sicheres Auslagern implementiert, gemäß einer Implementierung. In einigen Implementierungen kann eine Anweisung in Übereinstimmung mit einer Implementierung implementiert werden, um auf Datenelementen mit Größen von Byte, Wort, Double-Word, Quad-Word, usw., sowie Datentypen, wie zum Beispiel Ganzzahl- und Fließkomma-Datentypen mit einfacher und doppelter Genauigkeit, zu operieren. In einer Implementierung ist In-Order-Front-End **1701** der Teil des Prozessors **1700**, der die auszuführenden Anweisungen abruft und sie dazu vorbereitet, später in der Prozessor-Pipeline verwendet zu werden. Die Implementierungen der Seitenergänzungen und das Kopieren von Inhalt können in dem Prozessor **1700** implementiert werden.

[0107] Das Front-End **1701** kann einige Einheiten beinhalten. In einer Implementierung ruft Anweisungsvorababrufer **1726** Anweisungen von Speicher ab und führt sie Anweisungsdecoder **1718** zu, der sie wiederum decodiert oder interpretiert. Zum Beispiel decodiert in einer Implementierung der Decoder eine empfangene Anweisung in eine oder mehrere als „Mikroanweisungen“ oder „Mikrooperationen“ bezeichnete (auch bezeichnet als Mikro-Op oder U-Ops) Operationen, die die Maschine ausführen kann. In anderen Implementierungen analysiert der Decoder die Anweisung in einen Op-Code und entsprechende Daten- und Steuerfelder, die von der Mikroarchitektur dazu verwendet werden, Operationen in Übereinstimmung mit einer Implementierung auszuführen. In einer Implementierung nimmt Trace-Cache **1730** decodierte U-Ops und montiert sie für die Ausführung in programmgeordnete Sequenzen oder Traces in U-Op-Warteschlange **1734**. Wenn der Trace-Cache **1730** auf eine komplexe Anweisung trifft, stellt Mikrocode-ROM (oder -RAM) **1732** die für das Abschließen der Operation erforderlichen U-Ops bereit.

[0108] Einige Anweisungen werden in eine einzelne Mikro-Op umgewandelt, während andere einige Mikro-Ops benötigen, um die vollständige Operation abzuschließen. In einer Implementierung, wenn mehr als vier Mikro-Ops benötigt werden, um eine Anweisung abzuschließen, greift der Decoder **1718** auf den Mikrocode-ROM **1732** zu, um die Anweisung auszuführen. Für eine Implementierung kann eine Anweisung in eine kleine Anzahl von Mikro-Ops für das Verarbeiten in dem Anweisungsdecoder **1718** decodiert werden. In einer anderen Implementierung kann eine Anweisung innerhalb des Mikrocode-ROM **1732** gespeichert werden, sollte eine Anzahl von Mikro-Ops benötigt werden, um die Operation zu erfüllen. Der Trace-Cache **1730** bezieht sich auf ein Entry-Point-Programmable-Logik-Array (PLA), um einen korrekten Mikroanweisungszeiger für das Lesen der Mikrocodesequenzen zu bestimmen, um eine oder mehrere Anweisungen in Übereinstimmung mit einer Implementierung von dem Mikrocode-ROM **1732** abzuschließen. Nachdem der Mikrocode-ROM **1732** das Sequenzieren von Mikro-Ops für eine Anweisung beendet, nimmt das Front-End **1701** der Maschine das Abrufen von Mikro-Ops von dem Trace-Cache **1730** wieder auf.

[0109] Bei Out-of-Order-Ausführungs-Engine **1703** werden die Anweisungen für die Ausführung vorbereitet. Die Out-of-Order-Ausführungslogik hat eine Anzahl von Puffern, um den Fluss der Anweisungen zu glätten und neu zu ordnen, um Leistung zu optimieren, wenn sie die Pipeline durchlaufen und für die Ausführung geplant werden. Die Zuweisungslogik weist die Maschinenpuffer und Ressourcen zu, die jede U-Op benötigt, um ausgeführt zu werden. Die Registerumbenennungslogik benennt Logikregister in Einträge in einem Registersatz um. Der Zuweiser weist auch einen Eintrag für jede U-Op in einer der zwei U-Op-Warteschlangen zu, eine für Speicheroperationen und eine für Nichtspeicheroperationen, vor den Anweisungsplanern: Speicherplaner, schneller Planer **1702**, langsamer/allgemeiner Fließkommaplaner **1704** und einfacher Fließkommaplaner **1706**. Die U-Op-Planer **1702**, **1704**, **1706** bestimmen, wenn ein U-Op für die Ausführung bereit ist, basierend auf der Bereitschaft ihrer abhängigen Eingaberegisteroperandenquellen und der Verfügbarkeit der Ausführungsressourcen, die die U-Ops benötigen, um ihre Operation abzuschließen. Der schnelle Planer **1702** einer Implementierung kann auf jeder Hälfte des Haupttaktzyklus planen, während die anderen Planer nur

einmal pro Hauptprozessortaktzyklus planen können. Die Planer vermitteln, dass die Sende-Ports U-Ops für die Ausführung planen.

[0110] Registersätze **1708**, **1710** sitzen zwischen den Planern **1702**, **1704**, **1706** und Ausführungseinheiten **1712**, **1714**, **1716**, **1718**, **1720**, **1722**, **1724** in Ausführungsblock **1711**. Es gibt separaten Registersatz **1708**, **1710** für Ganzzahl- bzw. Fließkommaoperationen. Jeder Registersatz **1708**, **1710** einer Implementierung beinhaltet auch ein Umleitungsnetzwerk, das gerade abgeschlossene Ergebnisse, die noch nicht in den Registersatz geschrieben worden sind, an neue abhängige U-Ops umleiten oder weiterleiten kann. Der Ganzzahlregistersatz **1708** und der Fließkommaregistersatz **1710** sind auch dazu in der Lage, Daten mit dem anderen zu kommunizieren. Für eine Implementierung wird der Ganzzahlregistersatz **1708** in zwei separate Registersätze aufgeteilt, einen Registersatz für die niederwertigen 32 Bit von Daten und einen zweiten Registersatz für die hochwertigen 32 Bit von Daten. Der Fließkommaregistersatz **1710** einer Implementierung hat **128** Bit breite Einträge, da Fließkommaanweisungen typischerweise Operanden mit einer Breite von 64 bis **128** Bit haben.

[0111] Der Ausführungsblock **1711** enthält die Ausführungseinheiten **1712**, **1714**, **1716**, **1718**, **1720**, **1722**, **1724**, wo die Anweisungen tatsächlich ausgeführt werden. Dieser Abschnitt beinhaltet die Registersätze **1708**, **1710**, die die Ganzzahl- und Fließkomma-Datenoperandenwerte speichern, die die Mikroanweisungen benötigen, um auszuführen. Der Prozessor **1700** einer Implementierung setzt sich aus einer Anzahl von Ausführungseinheiten zusammen: Address-Generation-Unit (AGU) **1712**, AGU **1714**, schnelle ALU **1716**, schnelle ALU **1718**, langsame ALU **1720**, Fließkomma-ALU **1712**, Fließkommabewegungseinheit **1714**. Für eine Implementierung führen die Fließkommaausführungsblöcke **1712**, **1714** Fließkomma-, MMX-, SIMD- und SSE- oder andere Operationen aus. Die Fließkomma-ALU **1712** einer Implementierung beinhaltet einen 64-Bit-mal-64-Bit-Fließkommateiler, um Divisions-, Quadratwurzel- und Rest-Mikro-Ops auszuführen. Für Implementierungen der Offenbarung können einen Fließkommawert einbeziehende Anweisungen mit der Fließkommahardware gehandhabt werden.

[0112] In einer Implementierung gehen die ALU-Operationen zu den Hochgeschwindigkeits-ALU-Ausführungseinheiten **1716**, **1718**. Die schnellen ALU **1716**, **1718** einer Implementierung können schnelle Operationen mit einer effektiven Latenz von einem halben Taktzyklus ausführen. Für eine Implementierung gehen die meisten komplexen Ganzzahloperationen zu der langsamen ALU **1720**, da die langsame ALU **1720** Ganzzahl-Ausführungshardware für Operationen mit langer Latenz, wie zum Beispiel einen Vervielfacher, Verschiebungen, Flag-Logik und Zweigverarbeitung, beinhaltet. Speicherlade-/Speicheroperationen werden durch die AGU **1722**, **1724** ausgeführt. Für eine Implementierung werden die Ganzzahl-ALU **1716**, **1718**, **1720** in dem Zusammenhang mit dem Ausführen von Ganzzahloperationen auf 64-Bit-Datenoperanden beschrieben. In alternativen Implementierungen können die ALU **1716**, **1718**, **1720** implementiert werden, um eine Vielzahl von Datenbits einschließlich **16**, **32**, **128**, **256**, usw. zu unterstützen. In ähnlicher Weise können die Fließkommaeinheiten **1722**, **1724** implementiert sein, um einen Bereich von Operanden zu unterstützen, die Bits unterschiedlicher Breite haben. Für eine Implementierung können die Fließkommaeinheiten **1722**, **1724** auf **128** Bit breiten gepackten Datenoperanden in Verbindung mit SIMD- und Multimediaanweisungen operieren.

[0113] In einer Implementierung senden die U-Ops-Planer **1702**, **1704**, **1706** abhängige Operationen, bevor das Parent-Laden das Ausführen beendet hat. Da U-Ops in dem Prozessor **1700** spekulativ geplant und ausgeführt werden, beinhaltet der Prozessor **1700** auch eine Logik, um Speicherfehlschläge zu handhaben. Wenn ein Datenladen in dem Daten-Cache fehlschlägt, kann es abhängige Operationen in dem Ablauf in der Pipeline geben, die den Planer mit vorübergehend nicht korrekten Daten verlassen haben. Ein Wiederholmechanismus verfolgt Anweisungen, die nicht korrekte Daten verwenden, und führt sie erneut aus. Nur die abhängigen Operationen müssen wiederholt werden, und die unabhängigen Operationen dürfen abgeschlossen werden. Die Planer und der Wiederholmechanismus einer Implementierung eines Prozessors sind auch dazu ausgelegt, Anweisungssequenzen für Textkettenvergleichsoperationen abzufangen.

[0114] Der Begriff „Register“ kann sich auf die On-Board-Prozessorspeicherungsplätze beziehen, die als Teil von Anweisungen verwendet werden, um Operanden zu identifizieren. Mit anderen Worten können Register jene sein, die von außerhalb des Prozessors (aus einer Perspektive eines Programmierers) verwendbar sind. Jedoch sollten die Register einer Implementierung in der Bedeutung nicht auf einen bestimmten Typ von Schaltung beschränkt sein. Vielmehr ist ein Register einer Implementierung dazu in der Lage, Daten zu speichern und bereitzustellen und die hierin beschriebenen Funktionen auszuführen. Die hierin beschriebenen Register können durch Schaltung innerhalb eines Prozessors durch Verwenden jeglicher Anzahl unterschiedlicher Techniken, wie zum Beispiel zugeordnete physikalische Register, dynamisch zugeordnete physikalische Register durch Verwenden von Registerumbenennung, Kombinationen von zugeordneten und dynamisch zugeordneten physikalischen Registern, usw., implementiert werden. In einer Implementierung spei-

chern Ganzzahlregister 32-Bit-Ganzzahldaten. Ein Registersatz einer Implementierung enthält auch acht Multimedia-SIMD-Register für gepackte Daten.

[0115] Für die Diskussionen hierin werden die Register als Datenregister verstanden, die dazu ausgelegt sind, gepackte Daten zu halten, wie zum Beispiel 64 Bit breite MMX™ - Register (zum Teil auch bezeichnet als „mm"-Register) in Mikroprozessoren, die mit MMX-Technologie von Intel Corporation in Santa Clara, Kalifornien freigegeben sind. Diese MMX-Register, die sowohl in Ganzzahl-, als auch in Fließkommaformen verfügbar sind, können mit gepackten Datenelementen operieren, die SIMD- und SSE-Anweisungen begleiten. In ähnlicher Weise können 128 Bit breite XMM-Register, die sich auf **SSE2**-, **SSE3**-, **SSE4**- oder darüber hinausgehende (allgemein bezeichnet als „SSEx") Technologie beziehen, auch dazu verwendet werden, solche gepackten Datenoperanden zu halten. In einer Implementierung, bei dem Speichern gepackter Daten und Ganzzahldaten, müssen die Register nicht zwischen den zwei Datentypen unterscheiden. In einer Implementierung sind Ganzzahl und Fließkomma entweder in dem gleichen Registersatz oder in unterschiedlichen Registersätzen enthalten. Darüber hinaus können in einer Implementierung Fließkomma- und Ganzzahldaten in unterschiedlichen Registern oder den gleichen Registern gespeichert werden.

[0116] Implementierungen können in vielen unterschiedlichen Systemtypen verkörpert sein. Jetzt bezugnehmend auf **Fig. 18**, ist ein Blockdiagramm von Multiprozessorsystem **1800** gezeigt, das Hardware-Unterstützung für verbessertes sicheres Auslagern implementiert, in Übereinstimmung mit einer Implementierung. Wie in **Fig. 18** gezeigt, ist das Multiprozessorsystem **1800** ein Punkt-zu-Punkt-Verbindungssystem und beinhaltet ersten Prozessor **1870** und zweiten Prozessor **1880**, die über Punkt-zu-Punkt-Verbindung **1850** gekoppelt sind. Wie in **Fig. 18** gezeigt, kann jeder der Prozessoren **1870** und **1880** Multicore-Prozessoren sein, einschließlich erster und zweiter Prozessorkerne (das heißt Prozessorkerne **1874a** und **1874b** und Prozessorkerne **1884a** und **1884b)**, obwohl potenziell viel mehr Kerne in den Prozessoren vorhanden sein können. Während mit zwei Prozessoren **1870**, **1880** gezeigt, versteht es sich, dass der Umfang der Offenbarung nicht darauf beschränkt ist. In anderen Implementierungen können ein oder mehrere zusätzliche Prozessoren in einem gegebenen Prozessor vorhanden sein.

[0117] Die Prozessoren **1870** und **1880** sind einschließlich integrierter Speichersteuerungseinheiten **1872** bzw. **1882** gezeigt. Der Prozessor **1870** beinhaltet als Teil seiner Bus-Steuerungseinheiten auch Punkt-zu-Punkt (**P-P**) -Schnittstellen **1876** und **1888**; in ähnlicher Weise beinhaltet der zweite Prozessor **1880 P-P**-Schnittstellen **1886** und **1888**. Die Prozessoren **1870**, **1880** können Informationen über eine Punkt-zu-Punkt (**P-P**)-Schnittstelle **1850** durch Verwenden von **P-P**-Schnittstellenschaltungen **1878** bzw. **1888** austauschen. Wie in **Fig. 18** gezeigt, koppeln die IMC **1872** und **1882** die Prozessoren mit jeweiligen Speichern, nämlich mit Speicher **1832** und Speicher **1834**, die Abschnitte eines lokal an den jeweiligen Prozessoren befestigten Hauptspeichers sein können.

[0118] Die Prozessoren **1870**, **1880** können Informationen mit Chipsatz **1890** über individuelle **P-P**-Schnittstellen **1852**, **1854** durch Verwenden von Punkt-zu-Punkt-Schnittstellenschaltungen **1876**, **1894**, **1886**, **1898** austauschen. Der Chipsatz **1890** kann auch Informationen mit Hochleistungsgrafikschaltung **1838** über Hochleistungsgrafikschnittstelle **1839** austauschen.

[0119] Der Chipsatz **1890** kann über Schnittstelle **1896** mit erstem Bus **1816** gekoppelt sein. In einer Implementierung kann der erste Bus **1816** ein Peripheral-Component-Interconnect (PCI) -Bus oder ein Bus, wie zum Beispiel ein PCI Express-Bus oder Verbindungs-Bus, sein, obwohl der Umfang der Offenbarung nicht darauf beschränkt ist.

[0120] Jetzt bezugnehmend auf **Fig. 19**, ist ein Blockdiagramm von drittem System **1900** gezeigt, das Hardware-Unterstützung für verbessertes sicheres Auslagern implementiert, in Übereinstimmung mit einer Implementierung der Offenbarung. Gleiche Elemente in den **Fig. 18** und **Fig. 19** tragen gleiche Bezugszeichen, und bestimmte Aspekte von **Fig. 19** wurden in **Fig. 18** weggelassen, um ein Verschleiern anderer Aspekte von **Fig. 19** zu vermeiden.

[0121] **Fig. 19** veranschaulicht, dass Prozessoren **1970**, **1980** integrierte Speicher- und I/O-Steuerlogik („CL") **1972** bzw. **1992** beinhalten können. Für mindestens eine Implementierung kann die CL **1972**, **1982** integrierte Speichersteuerungseinheiten, wie zum Beispiel hierin beschrieben, beinhalten. Zusätzlich können CL **1972**, **1992** auch I/O-Steuerlogik beinhalten. **Fig. 19** veranschaulicht, dass Speicher **1932**, **1934** mit den CL **1972**, **1992** gekoppelt sind, und dass I/O-Geräte **1914** auch mit der Steuerlogik **1972**, **1992** gekoppelt sind. Etablierte I/O-Geräte **1915** sind mit Chipsatz **1990** gekoppelt.

[0122] **Fig. 20** ist beispielhaftes System-on-a-Chip (SoC) **2000**, das Hardware-Unterstützung für verbessertes sicheres Auslagern implementiert, gemäß einer Implementierung. Das SoC **2000** kann einen oder mehrere Kerne **2002A**...2002N beinhalten. Andere in dem Fachgebiet bekannte Systemauslegungen und Konfigurationen für Laptops, Desktops, Handheld-PC, Personal Digital Assistants, Engineering-Workstations, Server, Netzwerkgeräte, Netzwerk-Hubs, Schalter, eingebettete Prozessoren, Digital-Signal-Processors (DSP), Grafikgeräte, Videospielgeräte, Set-Top-Boxen, Mikrocontroller, Mobiltelefone, tragbare Medienwiedergabegeräte, Handheld-Geräte und verschiedene andere elektronische Geräte sind ebenfalls geeignet. Im Allgemeinen ist eine große Vielfalt von Systemen oder elektronischen Geräten, die zur Aufnahme eines Prozessors und/oder anderer Ausführungslogik als hierin offenbart geeignet sind, allgemein geeignet.

[0123] In dem beispielhaften SoC **2000** von **Fig. 20** sind Kästchen mit gestrichelten Linien Merkmale auf fortgeschrittenere SoC. Verbindungseinheit(en) **2002** können mit Folgendem gekoppelt sein: Anwendungsprozessor **2017**, der einen Satz von einem oder mehreren Kernen **2002A-N** und geteilte Cache-Einheit(en) **2006** beinhaltet; System-Agent-Einheit **2010**; Bussteuerungseinheit(en) **2016**; integrierte Speichersteuerungseinheit(en) **2014**; einen Satz von einem oder mehreren Mediaprozessoren **2020**, die integrierte Grafiklogik **2008**, Bildprozessor **2024** für das Bereitstellen von Standbild- und/oder Videokamerafunktionalität, Audioprozessor **2026** zum Bereitstellen von Hardware-Audiobeschleunigung und Videoprozessor **2028** zum Bereitstellen von Videocodierungs-/- Decodierungsbeschleunigung beinhalten können; Static-Random-Access-Memory (SRAM) -Einheit **2030**; Direct-Memory-Access (DMA) -Einheit **2032**; und Anzeigeeinheit **2040** zum Koppeln mit einer oder mehreren externen Anzeigen.

[0124] Als nächstes sich **Fig. 21** zuwendend, ist diese eine Implementierung eines System-on-Chip (SoC), das Hardware-Unterstützung für verbessertes sicheres Auslagern implementiert, in Übereinstimmung mit Implementierungen der Offenbarung. Als ein veranschaulichendes Beispiel ist SOC **2100** in einem User-Equipment (UE) enthalten. In einer Implementierung bezieht sich UE auf jegliches Gerät, das von einem Endverbraucher zum Kommunizieren verwendet werden soll, wie zum Beispiel ein tragbares Telefon, Smartphone, Tablet, ultradünnes Notebook, Notebook mit Breitbandadapter oder jegliches andere ähnliche Kommunikationsgerät. Ein UE kann mit einer Basisstation oder einem Knoten verbunden sein, der in seiner Natur einer Mobilstation (MS) in einem GSM-Netzwerk entsprechen kann. Die Implementierungen der Seitenergänzungen und das Kopieren von Inhalt können in dem SoC **2100** implementiert werden.

[0125] Hier beinhaltet das SoC **2100** 2 Kerne - **2106** und **2107**. Ähnlich der Diskussion oben können die Kerne **2106** und **2107** einer Anweisungssatzarchitektur entsprechen, wie zum Beispiel einem Prozessor mit Intel®-Architecture-Core™, einem Advanced-Micro-Devices, Inc. (AMD) -Prozessor, einem MIPS-basierten Prozessor, einem ARM-basierten Prozessor-Design oder einem Kunden davon, sowie ihren Lizenznehmern oder Anwendern. Die Kerne **2106** und **2107** sind mit Cache-Steuerung **2108** gekoppelt, die Busschnittstelleneinheit **2109** und **L2**-Cache **2110** zugeordnet ist, um mit anderen Teilen des Systems **2100** zu kommunizieren. Verbindung **2111** beinhaltet eine On-Chip-Verbindung, wie zum Beispiel eine IOSF, AMBA oder andere oben diskutierte Verbindungen, die einen oder mehrere Aspekte der beschriebenen Offenbarung implementieren können.

[0126] In einer Implementierung kann SDRAM-Steuerung **2140** über den Cache **2110** mit der Verbindung **2111** verbunden sein. Die Verbindung **2111** stellt Kommunikationskanäle zu den anderen Komponenten bereit, wie zum Beispiel Subscriber-Identity-Module (SIM) **2130** zum Verknüpfen mit einer SIM-Karte, Boot-ROM **2135** zum Halten von Boot-Code für die Ausführung durch die Kerne **2106** und **2107** zum Initialisieren und booten des SOC **2100**, SDRAM-Controller **2140** zum Verknüpfen mit externem Speicher (zum Beispiel DRAM **2160**), Flash-Controller **2145** zum Verbinden mit nichtflüchtigem Speicher (zum Beispiel Flash **2165**), periphere Steuerung **2150** (zum Beispiel Serial-Peripheral-Interface) zum Verknüpfen mit Peripheriegeräten, Video-Codecs **2120** und Videoschnittstelle **2125** zum Anzeigen und Empfangen von Eingaben (zum Beispiel berührungsaktivierte Eingaben), GPU **2115** zum Ausführen grafikbezogener Berechnungen, usw. Jegliche dieser Schnittstellen kann Aspekte der hierin beschriebenen Implementierungen enthalten.

[0127] Zusätzlich veranschaulicht das System Peripheriegeräte für die Kommunikation, wie zum Beispiel Bluetooth®-Modul **2170**, 3G-Modem **2175**, GPS **2180** und WiFi® **2185**. Es ist zu beachten, wie oben erwähnt, dass ein UE ein Funkgerät für die Kommunikation beinhaltet. Als ein Ergebnis können diese peripheren Kommunikationsmodule nicht alle enthalten sein. In einem UE sollte jedoch eine Form eines Funkgeräts für externe Kommunikation enthalten sein.

[0128] **Fig. 22** veranschaulicht eine schematische Darstellung einer Maschine in der beispielhaften Form von Computersystem **2200**, in dem ein Satz von Anweisungen für das Veranlassen der Maschine dazu, eine oder mehrere der hierin diskutierten Methodiken auszuführen, ausgeführt werden kann. Die eine oder die mehreren

Methodiken können jene beinhalten, die Hardware-Unterstützung für verbessertes sicheres Auslagern implementieren, gemäß einer Implementierung. In alternativen Implementierungen kann die Maschine mit anderen Maschinen in einem LAN, einem Intranet, einem Extranet oder dem Internet verbunden (zum Beispiel vernetzt), sein. Die Maschine kann in der Eigenschaft eines Servers oder eines Client-Geräts in einer Client-Server-Netzwerkumgebung oder als Peer-Maschine in einer Peer-to-Peer (oder verteilten) -Netzwerkumgebung operieren. Die Maschine kann ein Personal Computer (PC), ein Tablet-PC, eine Set-Top-Box (STB), ein Personal-Digital-Assistant (PDA), ein Mobiltelefon, eine Web-Appliance, ein Server, ein Netzwerk-Router, ein Schalter oder eine Brücke sein, oder jegliche Maschine, die dazu in der Lage ist, einen Satz von Anweisungen (sequentiell oder anderweitig) auszuführen, die die von dieser Maschine zu ergreifenden Maßnahmen spezifizieren. Darüber hinaus, während nur eine einzelne Maschine veranschaulicht ist, soll der Begriff „Maschine“ so verstanden werden, dass er jegliche Ansammlung von Maschinen beinhaltet, die einzeln oder gemeinsam einen Satz (oder mehrere Sätze) von Anweisungen ausführen, um eine oder mehrere der hierin diskutierten Methodiken auszuführen . Die Implementierungen der Seitenergänzungen und das Kopieren von Inhalt können in dem Computersystem **2200** implementiert werden.

[0129] Das Computersystem **2200** beinhaltet Verarbeitungsgerät **2202**, Hauptspeicher **2204** (zum Beispiel Flash-Speicher, Dynamic-Random-Access-Memory (DRAM) (wie zum Beispiel Synchronous-DRAM (SDRAM) oder DRAM (RDRAM), usw.), statischen Speicher **2206** (zum Beispiel Flash-Speicher, Static-Random-Access-Memory (SRAM), usw.) und Datenspeicherungsgerät **2216**, die über Bus **2208** miteinander kommunizieren.

[0130] Das Verarbeitungsgerät **2202** stellt ein oder mehrere Universalverarbeitungsgeräte dar, wie zum Beispiel einen Mikroprozessor, eine zentrale Verarbeitungseinheit oder dergleichen. Insbesondere kann das Verarbeitungsgerät ein Complex-Instruction-Set-Computing (CISC) -Mikroprozessor, Reduced-Instruction-Set-Computer (RISC) - Mikroprozessor, Very-Long-Instruction-Word (VLIW) -Mikroprozessor oder ein Prozessor, der andere Anweisungssätze implementiert, oder Prozessoren, die eine Kombination von Anweisungssätzen implementieren, sein. Das Verarbeitungsgerät **2202** kann auch ein oder mehrere Spezialverarbeitungsgeräte sein, wie zum Beispiel ein Application-Specific-Integrated-Circuit (ASIC), ein Field-Programmable-Gate-Array (FPGA), ein Digital-Signal-Processor (DSP), ein Netzwerkprozessor oder dergleichen. In einer Implementierung kann das Verarbeitungsgerät **2202** einen oder mehrere Prozessorkerne beinhalten. Das Verarbeitungsgerät **2202** ist dazu ausgelegt, Verarbeitungslogik **2226** zum Ausführen der hierin diskutierten Operationen auszuführen.

[0131] In einer Implementierung kann das Verarbeitungsgerät **2202** Teil eines Prozessors oder einer integrierten Schaltung sein, die die offenbarte LLC-Caching-Architektur beinhaltet. Alternativ kann das Computersystem **2200** andere Komponenten als hierin beschrieben beinhalten. Es sollte klargestellt werden, dass der Kern Multi-Threading (Ausführen von zwei oder mehr parallelen Sätzen von Operationen oder Threads) unterstützen und dies in einer Vielfalt von Weisen tun kann, einschließlich Zeitscheiben-Multi-Threading, gleichzeitiges Multi-Threading (wobei ein einzelner physikalischer Kern einen logischen Kern für alle Threads bereitstellt, für die dieser physikalische Kern ein gleichzeitiges Multi-Threading durchführt) oder einer Kombination davon (zum Beispiel Zeitscheiben-Abrufen und -Decodieren und gleichzeitiges Multi-Threading danach, wie zum Beispiel in der Intel® Hyperthreading-Technologie).

[0132] Das Computersystem **2200** kann ferner kommunizierend mit Netzwerk **2219** gekoppeltes Netzwerkschnittstellengerät **2218** beinhalten. Das Computersystem **2200** kann auch Videoanzeigegerät **2210** (zum Beispiel ein Liquid-Crystal-Display (LCD) oder eine Cathode-Ray-Tube (CRT)), alphanumerisches Eingabegerät **2212** (zum Beispiel eine Tastatur), Cursorsteuerungsgerät **2214** (zum Beispiel eine Maus), Signalerzeugungsgerät **2220** (zum Beispiel einen Lautsprecher) oder andere Peripheriegeräte beinhalten. Darüber hinaus kann das Computersystem **2200** Grafikverarbeitungseinheit **2222**, Videoverarbeitungseinheit **2228** und Audioverarbeitungseinheit **2232** beinhalten. In einer anderen Implementierung kann das Computersystem **2200** einen Chipsatz (nicht veranschaulicht) beinhalten, der sich auf eine Gruppe von integrierten Schaltungen oder Chips bezieht, die dazu ausgelegt sind, mit dem Verarbeitungsgerät **2202** zu arbeiten, und der die Kommunikationen zwischen dem Verarbeitungsgerät **2202** und externen Geräten steuern. Zum Beispiel kann der Chipsatz ein Satz von Chips auf einer Hauptplatine sein, die das Verarbeitungsgerät **2202** mit Hochgeschwindigkeitsgeräten, wie zum Beispiel dem Hauptspeicher **2204** und Grafiksteuerungen verbindet, sowie das Verarbeitungsgerät **2202** mit langsameren peripheren Bussen von Peripheriegeräten, wie zum Beispiel USB-, PCI- oder ISA-Bussen, verbindet.

[0133] Das Datenspeicherungsgerät **2216** kann computerlesbares Speicherungsmedium **2224** beinhalten, auf dem Software **2226** gespeichert ist, die eine oder mehrere der hierin beschriebenen Methodiken von Funktionen verkörpert. Die Software **2226** kann sich auch vollständig oder mindestens teilweise innerhalb des Haupt-

speichers **2204** als Anweisungen **2226** und/oder innerhalb des Verarbeitungsgeräts **2202** als Verarbeitungslogik während deren Ausführung durch das Computersystem **2200** befinden; der Hauptspeicher **2204** und das Verarbeitungsgerät **2202** bilden auch computerlesbare Speicherungsmedien.

[0134] Das computerlesbare Speicherungsmedium **2224** kann auch dazu verwendet werden, die Anweisungen **2226**, die das Verarbeitungsgerät **2202** verwenden, und/oder eine Softwarebibliothek, die Verfahren beinhaltet, die die obigen Anwendungen aufrufen, zu speichern. Während das computerlesbare Speicherungsmedium **2224** in einer beispielhaften Implementierung als ein einzelnes Medium gezeigt ist, sollte der Begriff „computerlesbares Speicherungsmedium“ so verstanden werden, dass er ein einzelnes Medium oder mehrere Medien (zum Beispiel eine zentralisierte oder verteilte Datenbank und/oder zugeordnete Caches und Server) beinhaltet, die den einen oder mehrere Sätze von Anweisungen speichern. Der Begriff „computerlesbares Speicherungsmedium“ soll auch so verstanden werden, dass er jedes Medium beinhaltet, das dazu geeignet ist, einen Satz von Anweisungen für die Ausführung durch die Maschine zu speichern, zu kodieren oder zu tragen, und der die Maschine dazu veranlasst, jegliche eine oder mehrere der Methodiken der offenbarten Implementierungen auszuführen. Der Begriff „computerlesbares Speicherungsmedium“ soll dementsprechend so verstanden werden, dass er Festkörperspeicher und optische und magnetische Medien beinhaltet, jedoch nicht darauf beschränkt ist.

[0135] Die folgenden Beispiele betreffen weitere Implementierungen.

[0136] Beispiel 1 ist ein Prozessor, der Folgendes umfasst: 1) eine sichere Enklavenschaltung, die eine sichere Speicherungsstruktur zum Speichern einer Enklave-Page-Cache-Map umfasst, wobei die Enklave-Page-Cache-Map Inhalte einer sicheren Enklave innerhalb von Systemspeicher verfolgen soll, wobei die sichere Enklave sichere Daten, die eine Seite mit einer virtuellen Adresse umfassen, speichern soll; und 2) eine operativ mit der sicheren Enklavenschaltung gekoppelte Ausführungseinheit, wobei die Ausführungseinheit als Reaktion auf eine Anforderung, die Seite von der sicheren Enklave zu räumen, mehrere decodierte Anweisungen ausführen soll, um: a) die Erzeugung von Übersetzungen der virtuellen Adresse zu blocken; b) einen oder mehrere aktuell auf die sicheren Daten in der sicheren Enklave zugreifende Hardware-Threads aufzuzeichnen; c) einen Inter-Processor-Interrupt an einen oder mehrere dem einen oder den mehreren Hardware-Threads zugeordnete Kerne zu senden, um den einen oder die mehreren Hardware-Threads dazu zu veranlassen, die sichere Enklave zu verlassen und Translation-Lookaside-Buffer des einen oder der mehreren Kerne zu leeren; und d) als Reaktion auf die Erkennung eines der virtuellen Adresse für die Seite in der sicheren Enklave zugeordneten Seitenfehlers die Erzeugung weiterer Übersetzungen der virtuellen Adresse freizugeben.

[0137] In Beispiel 2, dem Prozessor von Beispiel 1, wobei die Ausführungseinheit ferner den mit der Seite verbunden Seitenfehler erkennen soll, bevor die Ausführungseinheit eine Zurückschreibanweisung ausführen soll, um die Seite auf Festplatte zurückzuschreiben.

[0138] In Beispiel 3, dem Prozessor von Beispiel 1, wobei, um die Erzeugung der weiteren Übersetzungen der virtuellen Adresse zu blocken, die Ausführungseinheit ferner die mehreren decodierten Anweisungen ausführen soll, um: a) die Seite als nicht in mehreren Übersetzungsseitentabellen vorhanden zu kennzeichnen; b) den Ablauf einer Zeitdauer, während derer nicht auf die Seite zugegriffen wird, zu erkennen; und c) ein Bit in der Enklave-Page-Cache-Map für die zu der virtuellen Adresse abgebildete Seite zu setzen, um die Erzeugung weiterer Übersetzungen der virtuellen Adresse für Zugriff auf die Seite zu blocken.

[0139] In Beispiel 4, dem Prozessor von Beispiel 3, wobei, um die Erzeugung von Übersetzungen der virtuellen Adresse freizugeben, die Ausführungseinheit eine decodierte Freigabeanweisung ausführen soll, um: a) das Bit in der Enklave-Page-Cache-Map für die zu der virtuellen Adresse abgebildete Seite zu löschen; und b) die Seite als in den mehreren Übersetzungstabellen vorhanden zu markieren.

[0140] In Beispiel 5, dem Prozessor von Beispiel 1, wobei, um die Erzeugung von Übersetzungen der virtuellen Adresse freizugeben, die Ausführungseinheit eine decodierte Freigabeanweisung ausführen soll, um: a) zu verifizieren, dass es keinen dem Zugriff auf die Seite in der sicheren Enklave zugeordneten Ressourcenkonflikt gibt; b) von einem ersten Eingaberegister die virtuelle Adresse für die Seite in der sicheren Enklave zu lesen; c) die virtuelle Adresse in eine physikalische Guest-Adresse und die physikalische Guest-Adresse in eine physikalische Host-Adresse zu übersetzen; und d) über einen Zugriff auf die Enklave-Page-Cache-Map zu verifizieren, dass die physikalische Host-Adresse der Seite der sicheren Enklave in dem Systemspeicher zugeordnet ist.

[0141] In Beispiel 6, dem Prozessor von Beispiel 5, wobei die Ausführungseinheit ferner die decodierte Freigabeanweisung ausführen soll, um: a) eine physikalische Host-Adresse einer Secure-Enclave-Control-Structure (SECS), die der sicheren Enklave über Bezug zu einem Eintrag in der Enklave-Page-Cache-Map für die Seite zugeordnet ist, zu bestimmen; b) durch Verwenden der physikalischen Host-Adresse der SECS auf eine Epochenverfolgungsstruktur für die sichere Enklave zuzugreifen; c) von der Epochenverfolgungsstruktur einen vorherigen Epochenwert und einen vorherigen Epochenzählerwert in einen Satz von lokalen Epochenregistern zu lesen; d) basierend auf dem vorherigen Epochenwert und dem vorherigen Epochenzählerwert zu verifizieren, dass der eine oder die mehreren Hardware-Threads die sichere Enklave verlassen haben; und e) den der sicheren Enklave zugeordneten vorherigen Epochenwert zu löschen.

[0142] In Beispiel 7, dem Prozessor von Beispiel 6, wobei das Löschen des der sicheren Enklave zugeordneten vorherigen Epochenwerts vor dem Löschen eines geblockten Bits in der Enklave-Page-Cache-Map für die virtuelle Adresse ausgeführt wird.

[0143] Verschiedene Implementierungen können unterschiedliche Kombinationen der oben beschriebenen strukturellen Merkmale haben. Beispielsweise können alle optionalen Merkmale der oben beschriebenen Prozessoren und Verfahren auch in Bezug auf ein hierin beschriebenes System implementiert werden, und Besonderheiten in den Beispielen können irgendwo in einer oder mehreren Implementierungen verwendet werden.

[0144] Beispiel 8 ist ein System, das Folgendes umfasst: 1) mehrere Prozessorkerne, von denen jeder einen Translation-Lookaside-Buffer (TLB) umfasst; 2) eine sichere Enklavenschaltung, die eine sichere Speicherungsstruktur zum Speichern einer Enklave-Page-Cache-Map umfasst, wobei die Enklave-Page-Cache-Map Inhalte einer sicheren Enklave innerhalb von Systemspeicher verfolgen soll, wobei die sichere Enklave sichere Daten, die eine Seite mit einer virtuellen Adresse umfassen, speichern soll; und 3) eine operativ mit der sicheren Enklavenschaltung und den mehreren Prozessorkernen gekoppelte Ausführungseinheit, wobei die Ausführungseinheit als Reaktion auf eine Anforderung, die Seite von der sicheren Enklave zu räumen, mehrere decodierte Anweisungen ausführen soll, um: a) die Erzeugung von Übersetzungen der virtuellen Adresse zu blocken; b) einen oder mehrere aktuell auf die sicheren Daten in der sicheren Enklave zugreifende Hardware-Threads aufzuzeichnen; c) einen Inter-Processor-Interrupt an einen oder mehrere der mehreren dem einen oder den mehreren Hardware-Threads zugeordnete Prozessorkerne zu senden, um den einen oder die mehreren Hardware-Threads dazu zu veranlassen, die sichere Enklave zu verlassen und den TLB des einen oder der mehreren der mehreren Prozessorkerne zu leeren; und d) als Reaktion auf die Erkennung eines der virtuellen Adresse für die Seite in der sicheren Enklave zugeordneten Seitenfehlers die Erzeugung weiterer Übersetzungen der virtuellen Adresse freizugeben.

[0145] In Beispiel 9, dem System von Beispiel 8, wobei die Ausführungseinheit ferner den mit der Seite verbunden Seitenfehler erkennen soll, bevor die Ausführungseinheit eine Zurückschreibanweisung ausführen soll, um die Seite auf Festplatte zurückzuschreiben.

[0146] In Beispiel 10, dem System von Beispiel 8, wobei, um die Erzeugung der Übersetzungen der virtuellen Adresse zu blocken, die Ausführungseinheit ferner die mehreren decodierten Anweisungen ausführen soll, um: a) die Seite als nicht in mehreren Übersetzungsseitentabellen vorhanden zu kennzeichnen; b) den Ablauf einer Zeitdauer, während derer nicht auf die Seite zugegriffen wird, zu erkennen; und c) ein Bit in der Enklave-Page-Cache-Map für die zu der virtuellen Adresse abgebildete Seite zu setzen, um die Erzeugung weiterer Übersetzungen der virtuellen Adresse für Zugriff auf die Seite zu blocken.

[0147] In Beispiel 11, dem System von Beispiel 10, dem Prozessor von Anspruch 3, wobei, um die Erzeugung von Übersetzungen der virtuellen Adresse freizugeben, die Ausführungseinheit eine decodierte Freigabeanweisung ausführen soll, um: a) das Bit in der Enklave-Page-Cache-Map für die zu der virtuellen Adresse abgebildete Seite zu löschen; und b) die Seite als in den mehreren Übersetzungstabellen vorhanden zu markieren.

[0148] In Beispiel 12, dem System von Beispiel 8, wobei, um die Erzeugung von Übersetzungen der virtuellen Adresse freizugeben, die Ausführungseinheit eine decodierte Freigabeanweisung ausführen soll, um: a) zu verifizieren, dass es keinen dem Zugriff auf die Seite in der sicheren Enklave zugeordneten Ressourcenkonflikt gibt; b) von einem ersten Eingaberegister die virtuelle Adresse für die Seite in der sicheren Enklave zu lesen; c) die virtuelle Adresse in eine physikalische Guest-Adresse und die physikalische Guest-Adresse in eine physikalische Host-Adresse zu übersetzen; und d) über einen Zugriff auf die Enklave-Page-Cache-Map zu verifizieren, dass die physikalische Host-Adresse der Seite der sicheren Enklave in dem Systemspeicher zugeordnet ist.

[0149] In Beispiel 13, dem System von Beispiel 12, wobei die Ausführungseinheit ferner die decodierte Freigabeanweisung ausführen soll, um: a) eine physikalische Host-Adresse einer Secure-Enclave-Control-Structure (SECS), die der sicheren Enklave über Bezug zu einem Eintrag in der Enklave-Page-Cache-Map für die Seite zugeordnet ist, zu bestimmen; b) durch Verwenden der physikalischen Host-Adresse der SECS auf eine Epochenverfolgungsstruktur für die sichere Enklave zuzugreifen; c) von der Epochenverfolgungsstruktur einen vorherigen Epochenwert und einen vorherigen Epochenzählerwert in einen Satz von lokalen Epochenregistern zu lesen; d) basierend auf dem vorherigen Epochenwert und dem vorherigen Epochenzählerwert zu verifizieren, dass der eine oder die mehreren Hardware-Threads die sichere Enklave verlassen haben; und e) den der sicheren Enklave zugeordneten vorherigen Epochenwert zu löschen.

[0150] In Beispiel 14, dem System von Beispiel 13, wobei das Löschen des der sicheren Enklave zugeordneten vorherigen Epochenwerts vor dem Löschen eines geblockten Bits in der Enklave-Page-Cache-Map für die virtuelle Adresse ausgeführt wird.

[0151] Verschiedene Implementierungen können unterschiedliche Kombinationen der oben beschriebenen strukturellen Merkmale haben. Beispielsweise können alle optionalen Merkmale der oben beschriebenen Prozessoren und Verfahren auch mit Bezug auf ein hierin beschriebenes System implementiert werden, und die Besonderheiten in den Beispielen können irgendwo in einer oder mehreren Implementierungen verwendet werden.

[0152] Beispiel 15 ist ein Verfahren, das Folgendes umfasst: 1) das Speichern, in einer sicheren Enklave des Hauptspeichers eines Computersystems, sicherer Daten, die eine Seite mit einer virtuellen Adresse umfassen; 2) das Speichern, in einer sicheren Speicherungsstruktur einer sicheren Enklavenschaltung des Computersystems, einer Enklave-Page-Cache-Map, um Inhalte der sicheren Enklave innerhalb des Systemspeichers zu verfolgen; und als Reaktion auf eine Anforderung, die Seite von der sicheren Enklave zu räumen: 3) das Blocken der Erzeugung von Übersetzungen der virtuellen Adresse; 4) das Aufzeichnen eines oder mehrerer aktuell auf die sicheren Daten in der sicheren Enklave zugreifender Hardware-Threads; 5) das Senden eines Inter-Processor-Interrupt an einen oder mehrere dem einen oder den mehreren Hardware-Threads zugeordnete Kerne, um den einen oder die mehreren Hardware-Threads dazu zu veranlassen, die sichere Enklave zu verlassen und Translation-Lookaside-Buffer des einen oder der mehreren Kerne zu leeren; und 6) als Reaktion auf das Erkennen eines der virtuellen Adresse für die Seite in der sicheren Enklave zugeordneten Seitenfehlers, das Freigeben der Erzeugung von Übersetzungen der virtuellen Adresse.

[0153] In Beispiel 16, dem Verfahren von Beispiel 15, das ferner das Erkennen des mit der Seite verbundenen Seitenfehlers vor dem Ausführen einer Zurückschreibanweisung, um die Seite auf Festplatte zurückzuschreiben, umfasst.

[0154] In Beispiel 17, dem Verfahren von Beispiel 15, wobei das Blocken der Erzeugung von Übersetzungen der virtuellen Adresse Folgendes umfasst: 1) das Markieren der Seite als nicht in mehreren Übersetzungsseitentabellen vorhanden; 2) das Erkennen eines Ablaufs einer Zeitdauer, während derer nicht auf die Seite zugegriffen wird; und 3) das Setzen eines Bits in der Enklave-Page-Cache-Map für die zu der virtuellen Adresse abgebildete Seite, um die Erzeugung von Übersetzungen der virtuellen Adresse für Zugriff auf die Seite zu blocken.

[0155] In Beispiel 18, dem Verfahren von Beispiel 17, wobei das Freigeben der Erzeugung von Übersetzungen der virtuellen Adresse Folgendes umfasst: 1) das Löschen des Bits in der Enklave-Page-Cache-Map für die zu der virtuellen Adresse abgebildete Seite; und 2) das Markieren der Seite als in den mehreren Übersetzungstabellen vorhanden.

[0156] In Beispiel 19, dem Verfahren von Beispiel 15, wobei das Freigeben der Erzeugung von Übersetzungen der virtuellen Adresse Folgendes umfasst: 1) das Verifizieren, dass es keinen dem Zugriff auf die Seite in der sicheren Enklave zugeordneten Ressourcenkonflikt gibt; 2) das Lesen, von einem ersten Eingaberegister, der virtuellen Adresse für die Seite in der sicheren Enklave; 3) das Übersetzen der virtuellen Adresse in eine physikalische Guest-Adresse und der physikalischen Guest-Adresse in eine physikalische Host-Adresse; und 4) das Verifizieren, über einen Zugriff auf die Enklave-Page-Cache-Map, dass die physikalische Host-Adresse der Seite der sicheren Enklave in dem Systemspeicher zugeordnet ist.

[0157] In Beispiel 20, dem Verfahren von Beispiel 19, das ferner Folgendes umfasst: 1) das Bestimmen einer physikalischen Host-Adresse einer Secure-Enclave-Control-Structure (SECS), die der sicheren Enklave über Bezug zu einem Eintrag in der Enklave-Page-Cache-Map für die Seite zugeordnet ist; 2) das Zugreifen, durch

Verwenden der physikalischen Host-Adresse der SECS, auf eine Epochenverfolgungsstruktur für die sichere Enklave; 3) das Lesen, von der Epochenverfolgungsstruktur, eines vorherigen Epochenwerts und eines vorherigen Epochenzählerwerts in einen Satz von lokalen Epochenregistern; 4) das Verifizieren, basierend auf dem vorherigen Epochenwert und dem vorherigen Epochenzählerwert, dass der eine oder die mehreren Hardware-Threads die sichere Enklave verlassen haben; und 5) das Löschen des der sicheren Enklave zugeordneten vorherigen Epochenwerts vor der Löschung eines geblockten Bits in der Enklave-Page-Cache-Map für die virtuelle Adresse.

[0158] Verschiedene Implementierungen können unterschiedliche Kombinationen der oben beschriebenen strukturellen Merkmale haben. Beispielsweise können alle optionalen Merkmale der oben beschriebenen Prozessoren und Verfahren auch mit Bezug auf ein hierin beschriebenes System implementiert werden, und Besonderheiten in den Beispielen können irgendwo in einer oder mehreren Implementierungen verwendet werden.

[0159] Beispiel 21 ist ein Prozessor, der Folgendes umfasst: 1) eine sichere Enklavenschaltung, die eine sichere Speicherungsstruktur zum Speichern einer Enklave-Page-Cache-Map (EPCM) umfasst, wobei die EPCM Inhalte einer sicheren Enklave innerhalb des Systemspeichers verfolgen soll, wobei die sichere Enklave sichere Daten, die eine Seite an einer virtuellen Adresse umfassen, speichern soll, und wobei die EPCM ein erstes geblocktes Bit speichern soll, um anzuzeigen, ob die Seite geblockt ist; und 2) einen operativ mit der sicheren Enklavenschaltung gekoppelten Prozessorkern, wobei der Prozessorkern einen virtuellen Maschinenmonitor (VMM) ausführen soll, um das Auslagern innerhalb der sicheren Enklave für eine virtuelle Maschine (VM) zu verwalten, wobei der VMM, um die Seite von der sicheren Enklave zu räumen: a) das erste geblockte Bit in der EPCM setzen soll, um anzuzeigen, dass die Seite von weiterer Übersetzung der virtuellen Adresse für Zugriff auf die Seite geblockt ist; b) ein Blockstatusbit erzeugen soll, das einen Status des ersten geblockten Bits anzeigt, um ein zweites geblocktes Bit innerhalb eines ersten Feldes einer in dem Systemspeicher gespeicherten Informationsdatenstruktur zu setzen; c) eine Zurückschreibanweisung ausführen soll, um einen Wert des zweiten geblockten Bits in ein geblocktes Bit-Flag einer Unified-Metadata-Structure (UMDS) zu kopieren, wobei die UMDS in einer Versions-Array-Seite der sicheren Enklave gespeichert werden soll; und d) die Seite von der sicheren Enklave in eine des Systemspeichers, außerhalb der sicheren Enklave, oder auf Festplatte entfernen soll.

[0160] In Beispiel 22, dem Prozessor von Beispiel 21, wobei die Informationsdatenstruktur ferner ein zweites Feld umfasst, um ein USE_UMDS-Bit zu halten, wobei zum Wiederherstellen der Seite von der einen des Systemspeichers oder der Festplatte, auf die die Seite während der Ausführung der Zurückschreibanweisung geschrieben wurde, der VMM eine Ladeanweisung ausführen soll, um: a) innerhalb eines Eingaberegisters auf eine Adresse eines Platzes der Informationsdatenstruktur zuzugreifen; b) zu verifizieren, dass das USE_UMDS-Bit gesetzt ist; und c) das erste geblockte Bit der EPCM mit einem Wert des geblockten Bit-Flags, der von der UMDS gelesen wird, zu laden.

[0161] In Beispiel 23, dem Prozessor von Beispiel 21, wobei die Informationsdatenstruktur ferner ein zweites Feld umfasst, um ein USE_UMDS-Bit zu halten, wobei zum Wiederherstellen der Seite von der einen des Systemspeichers oder der Festplatte, auf die die Seite während der Ausführung der Zurückschreibanweisung geschrieben wurde, der VMM eine Ladeanweisung ausführen soll, um: a) innerhalb eines Eingaberegisters auf eine Adresse eines Platzes der Informationsdatenstruktur zuzugreifen; b) zu verifizieren, dass das USE_UMDS-Bit gelöscht ist; und c) das erste geblockte Bit der EPCM mit einem Wert des zweiten geblockten Bits, der von der Informationsdatenstruktur gelesen wird, zu laden.

[0162] In Beispiel 24, dem Prozessor von Beispiel 21, wobei der VMM ferner: a) in einem ersten Eingaberegister auf einen ersten Zeiger auf die virtuelle Adresse der Seite in der sicheren Enklave zugreifen soll; b) in einem zweiten Eingaberegister auf einen zweiten Zeiger auf die Versions-Array-Seite in der sicheren Enklave zugreifen soll; c) in einem dritten Eingaberegister auf einen dritten Zeiger auf einen Speicherplatz außerhalb der sicheren Enklave, auf den die Seite gespeichert werden soll, zugreifen soll; d) einen Message-Authentication-Code (MAC) der Seite unter Verwendung einer Versionsnummer erzeugen soll; e) Daten für die Seite unter Verwendung eines geheimen Schlüssels verschlüsseln soll, um verschlüsselte Daten für die Seite zu erzeugen; f) die innerhalb der Versions-Array-Seite gespeicherte UMDS mit der Versionsnummer und dem MAC bestücken soll; g) die verschlüsselten Daten in den Speicherplatz außerhalb der sicheren Enklave schreiben soll; h) Auslagerungs-Krypto-Metadaten für die Seite in die UMDS schreiben soll; und i) die Seite von der sicheren Enklave zurück auf die Festplatte entfernen soll.

[0163] In Beispiel 25, dem Prozessor von Beispiel 24, wobei der VMM, um die Seite wiederherzustellen, eine Ladeanweisung ausführen soll, um: a) in einem ersten Ausgaberegister auf einen vierten Zeiger auf die verschlüsselte Seite an dem Speicherplatz zuzugreifen; b) in einem vierten Eingaberegister auf einen fünften Zeiger auf einen freien Seitenplatz in der sicheren Enklave zuzugreifen; c) in dem zweiten Eingaberegister auf den zweiten Zeiger auf eine virtuelle Adresse eines Versions-Array-Slots in der sicheren Enklave zuzugreifen; d) die virtuelle Adresse in eine physikalische Guest-Adresse und die physikalische Guest-Adresse in eine physikalische Host-Adresse für die Versions-Array-Seite zu übersetzen; e) auf die UMDS innerhalb der Versions-Array-Seite an der physikalischen Host-Adresse des Versions-Array-Slots zuzugreifen; f) die verschlüsselte Seite durch Verwenden des MAC und der Versionsnummer zu verifizieren; g) die verschlüsselten Daten zu entschlüsseln, um entschlüsselte Daten für die Seite zu erzeugen; h) die entschlüsselten Daten in der sicheren Enklave an dem freien Seitenplatz zu speichern; und i) einen Slot innerhalb der UMDS für einen nächsten Versionswert zu löschen.

[0164] Verschiedene Implementierungen können unterschiedliche Kombinationen der oben beschriebenen strukturellen Merkmale haben. Beispielsweise können alle optionalen Merkmale der oben beschriebenen Prozessoren und Verfahren auch mit Bezug auf ein hierin beschriebenes System implementiert werden, und Besonderheiten in den Beispielen können irgendwo in einer oder mehreren Implementierungen verwendet werden.

[0165] Beispiel 26 ist ein nichtflüchtiges computerlesbares Speicherungsmedium zum Speichern von Anweisungen, die, wenn durch einen Prozessor ausgeführt, mehrere Operationen ausführen sollen, die Folgendes umfassen: 1) das Speichern, in einer sicheren Enklave des Hauptspeichers eines Computersystems, sicherer Daten, die eine Seite mit einer virtuellen Adresse umfassen; 2) das Speichern, in einer sicheren Speicherungsstruktur einer sicheren Enklavenschaltung des Computersystems, einer Enklave-Page-Cache-Map, um Inhalte der sicheren Enklave innerhalb des Systemspeichers zu verfolgen; und 3) als Reaktion auf eine Anforderung, die Seite von der sicheren Enklave zu räumen: a) das Blocken der Erzeugung von Übersetzungen der virtuellen Adresse; b) das Aufzeichnen eines oder mehrerer aktuell auf die sicheren Daten in der sicheren Enklave zugreifender Hardware-Threads; c) das Senden eines Inter-Processor-Interrupt an einen oder mehrere dem einen oder den mehreren Hardware-Threads zugeordnete Kerne, um den einen oder die mehreren Hardware-Threads dazu zu veranlassen, die sichere Enklave zu verlassen und Translation-Lookaside-Buffer des einen oder der mehreren Kerne zu leeren; und d) als Reaktion auf das Erkennen eines der virtuellen Adresse für die Seite in der sicheren Enklave zugeordneten Seitenfehlers, das Freigeben der Erzeugung von Übersetzungen der virtuellen Adresse.

[0166] In Beispiel 27, dem nichtflüchtigen computerlesbaren Speicherungsmedium von Anspruch 26, wobei die mehreren Operationen ferner das Erkennen des mit der Seite verbundenen Seitenfehlers vor der Ausführung einer Zurückschreibanweisung, um die Seite auf Festplatte zurückzuschreiben, umfassen.

[0167] In Beispiel 28, dem nichtflüchtigen computerlesbaren Speicherungsmedium von Beispiel 26, wobei das Blocken der Erzeugung von Übersetzungen der virtuellen Adresse Folgendes umfasst: 1) das Markieren der Seite als nicht in mehreren Übersetzungsseitentabellen vorhanden; 2) das Erkennen eines Ablaufs einer Zeitdauer, während derer nicht auf die Seite zugegriffen wird; und 3) das Setzen eines Bits in der Enklave-Page-Cache-Map für die zu der virtuellen Adresse abgebildete Seite, um die Erzeugung weiterer Übersetzungen der virtuellen Adresse für Zugriff auf die Seite zu blocken.

[0168] In Beispiel 29, dem nichtflüchtigen computerlesbaren Speicherungsmedium von Beispiel 28, wobei das Freigeben der Erzeugung von Übersetzungen der virtuellen Adresse Folgendes umfasst: 1) das Löschen des Bits in der Enklave-Page-Cache-Map für die zu der virtuellen Adresse abgebildete Seite; und 2) das Markieren der Seite als in den mehreren Übersetzungstabellen vorhanden.

[0169] In Beispiel 30, dem nichtflüchtigen computerlesbaren Speicherungsmedium von Beispiel 26, wobei das Freigeben der Erzeugung von Übersetzungen der virtuellen Adresse Folgendes umfasst: 1) das Verifizieren, dass es keinen dem Zugriff auf die Seite in der sicheren Enklave zugeordneten Ressourcenkonflikt gibt; 2) das Lesen, von einem ersten Eingaberegister, der virtuellen Adresse für die Seite in der sicheren Enklave; 3) das Übersetzen der virtuellen Adresse in eine physikalische Guest-Adresse und der physikalischen Guest-Adresse in eine physikalische Host-Adresse; und 4) das verifizieren, über einen Zugriff auf die Enklave-Page-Cache-Map, dass die physikalische Host-Adresse der Seite der sicheren Enklave in dem Systemspeicher zugeordnet ist.

[0170] In Beispiel 31, dem nichtflüchtigen computerlesbaren Speicherungsmedium von Beispiel 30, wobei die mehreren Operationen ferner Folgendes umfassen: 1) das Bestimmen einer physikalischen Host-Adresse einer Secure-Enclave-Control-Structure (SECS), die der sicheren Enklave über Bezug zu einem Eintrag in der Enklave-Page-Cache-Map für die Seite zugeordnet ist; 2) das Zugreifen, durch Verwenden der physikalischen Host-Adresse der SECS, auf eine Epochenverfolgungsstruktur für die sichere Enklave; 3) das Lesen, von der Epochenverfolgungsstruktur, eines vorherigen Epochenwerts und eines vorherigen Epochenzählerwerts in einen Satz von lokalen Epochenregistern; 4) das Verifizieren, basierend auf dem vorherigen Epochenwert und dem vorherigen Epochenzählerwert, dass der eine oder die mehreren Hardware-Threads die sichere Enklave verlassen haben; und 5) das Löschen des der sicheren Enklave zugeordneten vorherigen Epochenwert vor der Löschung eines geblockten Bits in der Enklave-Page-Cache-Map für die virtuelle Adresse.

[0171] Verschiedene Implementierungen können unterschiedliche Kombinationen der oben beschriebenen strukturellen Merkmale haben. Beispielsweise können alle optionalen Merkmale der oben beschriebenen Prozessoren und Verfahren auch in Bezug auf ein hierin beschriebenes System implementiert werden, und Besonderheiten in den Beispielen können irgendwo in einer oder mehreren Implementierungen verwendet werden.

[0172] Beispiel 32 ist eine Vorrichtung, die Folgendes umfasst: 1) Mittel für das Speichern, in einer sicheren Enklave des Hauptspeichers eines Computersystems, sicherer Daten, die eine Seite mit einer virtuellen Adresse umfassen; 2) Mittel für das Speichern, in einer sicheren Speicherungsstruktur einer sicheren Enklavenschaltung des Computersystems, einer Enklave-Page-Cache-Map, um Inhalte der sicheren Enklave innerhalb des Systemspeichers zu verfolgen; und als Reaktion auf eine Anforderung, die Seite von der sicheren Enklave zu räumen: 3) Mittel für das Blocken der Erzeugung von Übersetzungen der virtuellen Adresse; 4) Mittel für das Aufzeichnen eines oder mehrerer aktuell auf die sicheren Daten in der sicheren Enklave zugreifender Hardware-Threads; 5) Mittel für das Senden eines Inter-Processor-Interrupt an einen oder mehrere dem einen oder den mehreren Hardware-Threads zugeordnete Kerne, um den einen oder die mehreren Hardware-Threads dazu zu veranlassen, die sichere Enklave zu verlassen und Translation-Lookaside-Buffer des einen oder der mehreren Kerne zu leeren; und 6) als Reaktion auf das Erkennen eines der virtuellen Adresse für die Seite in der sicheren Enklave zugeordneten Seitenfehlers, Mittel für das Freigeben der Erzeugung von Übersetzungen der virtuellen Adresse.

[0173] In Beispiel 33, der Vorrichtung von Beispiel 32, die ferner Mittel für das Erkennen des mit der Seite verbundenen Seitenfehlers vor dem Ausführen einer Zurückschreibanweisung, um die Seite auf Festplatte zurückzuschreiben, umfasst.

[0174] In Beispiel 34, der Vorrichtung von Beispiel 32, wobei die Mittel für das Blocken der Erzeugung von Übersetzungen der virtuellen Adresse Folgendes umfassen: 1) Mittel für das Markieren der Seite als nicht in mehreren Übersetzungsseitentabellen vorhanden; 2) Mittel für das Erkennen eines Ablaufs einer Zeitdauer, während derer nicht auf die Seite zugegriffen wird; und 3) Mittel für das Setzen eines Bits in der Enklave-Page-Cache-Map für die zu der virtuellen Adresse abgebildete Seite, um die Erzeugung von Übersetzungen der virtuellen Adresse für Zugriff auf die Seite zu blocken.

[0175] In Beispiel 35, der Vorrichtung von Beispiel 34, wobei die Mittel für das Freigeben der Erzeugung von Übersetzungen der virtuellen Adresse Folgendes umfassen: 1) Mittel für das Löschen des Bits in der Enklave-Page-Cache-Map für die zu der virtuellen Adresse abgebildete Seite; und 2) Mittel für das Markieren der Seite als in den mehreren Übersetzungstabellen vorhanden.

[0176] In Beispiel 36, der Vorrichtung von Beispiel 32, wobei die Mittel für das Freigeben der Erzeugung von Übersetzungen der virtuellen Adresse Folgendes umfassen: 1) Mittel für das Verifizieren, dass es keinen dem Zugriff auf die Seite in der sicheren Enklave zugeordneten Ressourcenkonflikt gibt; 2) Mittel für das Lesen, von einem ersten Eingaberegister, der virtuellen Adresse für die Seite in der sicheren Enklave; 3) Mittel für das Übersetzen der virtuellen Adresse in eine physikalische Guest-Adresse und der physikalischen Guest-Adresse in eine physikalische Host-Adresse; und 4) Mittel für das Verifizieren, über einen Zugriff auf die Enklave-Page-Cache-Map, dass die physikalische Host-Adresse der Seite der sicheren Enklave in dem Systemspeicher zugeordnet ist.

[0177] In Beispiel 37, der Vorrichtung von Anspruch 36, die ferner Folgendes umfasst: 1) Mittel für das Bestimmen einer physikalischen Host-Adresse einer Secure-Enclave-Control-Structure (SECS), die der sicheren Enklave über Bezug zu einem Eintrag in der Enklave-Page-Cache-Map für die Seite zugeordnet ist; 2) Mittel für das Zugreifen, durch Verwenden der physikalischen Host-Adresse der SECS, auf eine Epochenverfolgungsstruktur für die sichere Enklave; 3) Mittel für das Lesen, von der Epochenverfolgungsstruktur, eines vorherigen

Epochenwerts und eines vorherigen Epochenzählerwerts in einen Satz von lokalen Epochenregistern; 4) Mittel für das Verifizieren, basierend auf dem vorherigen Epochenwert und dem vorherigen Epochenzählerwert, dass der eine oder die mehreren Hardware-Threads die sichere Enklave verlassen haben; und 5) Mittel für das Löschen des der sicheren Enklave zugeordneten vorherigen Epochenwerts vor der Löschung eines geblockten Bits in der Enklave-Page-Cache-Map für die virtuelle Adresse.

[0178] Während die Offenbarung mit Bezug auf eine beschränkte Anzahl von Implementierungen beschrieben wurde, werden Fachleute zahlreiche Modifikationen und Variationen davon anerkennen. Es ist beabsichtigt, dass die beigefügten Ansprüche alle solchen Modifikationen und Variationen, wie sie in den wahren Geist und Umfang dieser Offenbarung fallen, abdecken.

[0179] In der Beschreibung hierin werden zahlreiche spezifische Details dargelegt, wie zum Beispiel Beispiele spezifischer Prozessortypen und Systemkonfigurationen, spezifischer Hardware-Strukturen, spezifischer architektureller und mikroarchitektureller Details, spezifischer Registerkonfigurationen, spezifischer Anweisungstypen, spezifischer Systemkomponenten, spezifischer Messungen/Höhen, spezifischer Prozessor-Pipelinestufen und -Operationen, usw., um ein sorgfältiges Verständnis der Offenbarung bereitzustellen. Es wird jedoch für den Fachmann ersichtlich, dass diese spezifischen Details nicht eingesetzt werden müssen, um die Offenbarung zu praktizieren. In anderen Fällen wurden gut bekannte Komponenten oder Verfahren, wie zum Beispiel spezifische und alternative Prozessorarchitekturen, spezifische Logikschaltungen/-code für beschriebene Algorithmen, spezifischer Firmware-Code, spezifische Verbindungsoperationen, spezifische Logikkonfigurationen, spezifische Herstellungstechniken und Materialien, spezifische Compiler-Implementierungen, spezifische Ausdrücke von Algorithmen in Code, spezifische Abschalt- und Gattertechniken/-Logik und andere spezifische operative Details von Computersystemen nicht detailliert beschrieben, um unnötiges Verschleiern der Offenbarung zu vermeiden.

[0180] Die Implementierungen werden unter Bezugnahme auf das Bestimmen der Gültigkeit von Daten in Cache-Zeilen eines sektorbasierten Caches in spezifischen integrierten Schaltungen, wie zum Beispiel in Computerplattformen oder Mikroprozessoren, beschrieben. Die Implementierungen können auch auf andere Typen von integrierten Schaltungen und programmierbaren Logikgeräten anwendbar sein. Zum Beispiel sind die offenbarten Implementierungen nicht auf Desktop-Computersysteme oder tragbare Computer, wie zum Beispiel die Intel® Ultrabooks™ -Computer, beschränkt. Und können auch in anderen Geräten, wie zum Beispiel tragbaren Geräten, Tablets, anderen dünnen Notebooks, System-on-a-Chip (SoC) -Geräten und eingebetteten Anwendungen verwendet werden. Manche Beispiele von tragbaren Geräte beinhalten Mobiltelefone, Internetprotokollgeräte, Digitalkameras, Personal-Digital-Assistants (PDA) und tragbare PC. Eingebettete Anwendungen beinhalten typischerweise einen Mikro-Controller, einen Digital-Signal-Processor (DSP), ein System-on a-Chip, Network-Computer (NetPC), Set-Top-Boxen, Network-Hubs, Wide-Area-Network (WAN) -Schalter oder jegliches andere System, das die unten gelehrten Funktionen und Operationen ausführen kann. Es wird beschrieben, dass das System jegliche Art von Computer oder eingebettetem System sein kann. Die offenbarten Implementierungen können insbesondere für Low-End-Geräte, wie tragbare Geräte (zum Beispiel Uhren), elektronische Implantate, Sensor- und Steuerungsinfrastrukturgeräte, Steuerungen, Supervisory-Control-and-Data-Acquisition (SCADA) -Geräte, oder dergleichen verwendet werden. Außerdem sind die hierin beschriebenen Vorrichtungen, Verfahren und Systeme nicht auf physikalische Computergeräte beschränkt, sondern können sich auch auf Softwareoptimierungen für die Energieerhaltung und -Effizienz beziehen. Wie in der Beschreibung unten ohne Weiteres ersichtlich wird, sind die hierin beschriebenen Implementierungen von Verfahren, Vorrichtungen und Systemen (ob mit Bezug auf Hardware, Firmware, Software oder eine Kombination davon) für eine mit Leistungserwägungen ausgewogene Zukunft einer „grünen Technologie“ unverzichtbar.

[0181] Obwohl die Implementierungen hierin unter Bezugnahme auf einen Prozessor beschrieben werden, sind andere Implementierungen auf andere Typen von integrierten Schaltungen und Logikgeräten anwendbar. Ähnliche Techniken und Lehren von Implementierungen der Offenbarung können auf andere Typen von Schaltungen oder Halbleitergeräten, die von höherem Pipelinedurchsatz und verbesserter Leistung profitieren können, angewendet werden. Die Lehren von Implementierungen der Offenbarung sind auf jeglichen Prozessor oder jegliche Maschine anwendbar, der/die Datenmanipulationen ausführt. Jedoch ist die Offenbarung nicht auf Prozessoren oder Maschinen, die 512-Bit-, 256-Bit-, 128-Bit-, 64-Bit-, 32-Bit- oder 16-Bit-Datenoperationen ausführen, beschränkt, und kann auf jeglichen Prozessor und jegliche Maschine angewendet werden, in denen Manipulation oder Verwaltung von Daten ausgeführt wird. Zusätzlich stellt die Beschreibung hierin Beispiele bereit, und die begleitenden Zeichnungen zeigen verschiedene Beispiele für die Zwecke der Veranschaulichung. Jedoch sollten diese Beispiele nicht in einem einschränkenden Sinn ausgelegt werden, da sie lediglich dazu beabsichtigt sind, Beispiele von Implementierungen der Offenbarung bereitzustellen, anstatt eine erschöpfende Liste aller möglichen Implementierungen von Implementierungen der Offenbarung bereitzustellen.

[0182] Obwohl die obigen Beispiele Anweisungshandhabung und -Verteilung in dem Kontext von Ausführungseinheiten und Logikschaltungen beschreiben, können andere Implementierungen der Offenbarung mittels Daten oder Anweisungen erfüllt werden, die auf einem maschinenlesbaren, materiellen Medium gespeichert sind, die, wenn durch eine Maschine ausgeführt, die Maschine dazu veranlassen, mit mindestens einer Implementierung der Offenbarung übereinstimmende Funktionen auszuführen. In einer Implementierung werden Funktionen, die Implementierungen der Offenbarung zugeordnet sind, in maschinenausführbaren Anweisungen verkörpert. Die Anweisungen können dazu verwendet werden, einen Universal- oder Spezialprozessor, der mit den Anweisungen programmiert ist, dazu zu veranlassen, die Schritte der Offenbarung auszuführen. Implementierungen der Offenbarung können als ein Computerprogrammprodukt oder -Software bereitgestellt werden, die ein maschinen- oder computerlesbares Medium mit darauf gespeicherten Anweisungen beinhalten kann, die zum Programmieren eines Computers (oder anderer elektronischer Geräte) verwendet werden können, um eine oder mehrere Operationen gemäß Implementierungen der Offenbarung auszuführen. Alternativ könnten Operationen von Implementierungen der Offenbarung durch spezifische Hardwarekomponenten, die eine Logik mit fester Funktion für das Ausführen der Operationen enthalten, oder durch jegliche Kombination von programmierten Computerkomponenten und Hardwarekomponenten mit fester Funktion, ausgeführt werden.

[0183] Anweisungen, die dazu verwendet werden, Logik zu programmieren, um Implementierungen der Offenbarung auszuführen, können innerhalb eines Speichers in dem System, wie zum Beispiel DRAM, Cache, Flash-Speicher oder andere Speicherung, gespeichert werden. Darüber hinaus können die Anweisungen über ein Netzwerk oder mittels anderer computerlesbarer Medien verteilt werden. Somit kann ein maschinenlesbares Medium jeglichen Mechanismus für das Speichern oder Übertragen von Informationen in einer Form, die durch eine Maschine (zum Beispiel einen Computer) gelesen werden kann, Floppy Disks, optische Disks, Compact Disk, Read-Only-Memory (CD-ROM) und magneto-optische Disks, Read-Only-Memory (ROM), Random-Access-Memory (RAM), Erasable-Programmable-Read-Only-Memory(EPROM), Electrically-Erasable-Programmable-Read-Only-Memory (EEPROM), magnetische oder optische Karten, Flash-Speicher oder eine materielle, maschinenlesbare Speicherung, die bei der Informationsübertragung über das Internet über elektrische, optische, akustische oder andere Formen von propagierten Signalen (zum Beispiel Trägerwellen, Infrarotsignale, digitale Signale, usw.) verwendet werden, beinhalten, ist jedoch nicht darauf beschränkt. Dementsprechend beinhaltet das computerlesbare Medium jeglichen Typ von materiellem maschinenlesbarem Medium, das für die Speicherung oder das Übertragen elektronischer Anweisungen oder Informationen in einer durch eine Maschine (zum Beispiel einen Computer) lesbaren Form geeignet ist.

[0184] Ein Design kann verschiedene Stufen durchlaufen, von der Erzeugung über die Simulation bis zur Herstellung. Daten, die ein Design darstellen, können das Design auf mehrfache Weisen darstellen. Zuerst, wie in Simulationen nützlich ist, kann die Hardware durch Verwenden einer Hardwarebeschreibungssprache oder einer anderen funktionalen Beschreibungssprache dargestellt werden. Zusätzlich kann in einem gewissen Stadium des Designprozesses ein Schaltungsebenenmodell mit Logik- und/oder Transistor-Gates produziert werden. Darüber hinaus erreichen die meisten Designs in einem gewissen Stadium ein Datenniveau, das die physikalische Platzierung verschiedener Geräte in dem Hardware-Modell repräsentiert. In dem Fall, wo herkömmliche Halbleiterherstellungstechniken verwendet werden, können die das Hardware-Modell repräsentierenden Daten die Daten sein, die das Vorhandensein oder das Nichtvorhandensein verschiedener Merkmale auf unterschiedlichen Maskenschichten für Masken, die für das Produzieren der integrierten Schaltung verwendet werden, spezifizieren. In jeglicher Darstellung des Designs können die Daten in jeglicher Form eines maschinenlesbaren Mediums gespeichert werden. Ein Speicher oder eine magnetische oder optische Speicherung, wie zum Beispiel eine Festplatte, kann das maschinenlesbare Medium zum Speichern von Informationen sein, die über modulierte oder anderweitig erzeugte optische oder elektrische Wellen übertragen werden, um solche Informationen zu übertragen. Wenn eine elektrische Trägerwelle, die den Code oder das Design anzeigt oder trägt, in dem Ausmaß, in dem Kopieren, Puffern oder Neuübermittlung des elektrischen Signals ausgeführt wird, übertragen wird, wird eine neue Kopie gemacht. Somit kann ein Kommunikationsanbieter oder ein Netzwerkanbieter auf einem materiellen, maschinenlesbaren Medium mindestens zeitweise einen Gegenstand, wie zum Beispiel in eine Trägerwelle codierte Informationen, durch verkörpern von Techniken von Implementierungen der Offenbarung speichern.

[0185] Ein Modul, wie hierin verwendet, bezieht sich auf jegliche Kombination von Hardware, Software und/oder Firmware. Als ein Beispiel beinhaltet ein Modul Hardware, wie zum Beispiel einen Mikrocontroller, der einem nichtflüchtigen Medium zugeordnet ist, um Code, der dazu angepasst ist, durch den Mikrocontroller ausgeführt zu werden, zu speichern. Deshalb bezieht sich eine Bezugnahme auf ein Modul in einer Implementierung auf die Hardware, die spezifisch dazu ausgelegt ist, den Code, der auf einem nichtflüchtigen Medium gehalten werden soll, zu erkennen und/oder auszuführen. Darüber hinaus bezieht sich in einer anderen Imple-

mentierung die Verwendung eines Moduls auf das nichtflüchtige Medium, einschließlich des Codes, der spezifisch dazu angepasst ist, durch den Mikrocontroller ausgeführt zu werden, um die vorgegebenen Operationen auszuführen. Und wie gefolgert werden kann, kann sich in noch einer anderen Implementierung der Begriff Modul (in diesem Beispiel) auf die Kombination des Mikrocontrollers und des nichtflüchtigen Mediums beziehen. Modulgrenzen, die als getrennt veranschaulicht werden, variieren oft üblicherweise und überlappen sich möglicherweise. Zum Beispiel können sich ein erstes und ein zweites Modul Hardware, Software, Firmware oder eine Kombination davon teilen, während sie möglicherweise eine unabhängige Hardware, Software oder Firmware zurückhalten. In einer Implementierung beinhaltet die Verwendung des Begriffs Logik Hardware, wie zum Beispiel Transistoren, Register oder andere Hardware, wie zum Beispiel programmierbare Logikgeräte.

[0186] Die Verwendung der Phrase „dazu ausgelegt“ in einer Implementierung bezieht sich auf das Anordnen, Zusammenstellen, Herstellen, zum Verkauf anbieten, Importieren und/oder Designen einer Vorrichtung, Hardware, Logik oder eines Elements, um eine ausgewiesene oder bestimmte Aufgabe auszuführen. In diesem Beispiel ist eine Vorrichtung oder ein Element davon, die/das nicht operiert, immer noch „dazu ausgelegt“, eine ausgewiesene Aufgabe auszuführen, wenn sie/es designed, gekoppelt und/oder verbunden ist, um die besagte ausgewiesene Aufgabe auszuführen. Als ein rein veranschaulichendes Beispiel kann ein Logikgatter während der Operation eine 0 oder eine 1 bereitstellen. Aber ein Logikgatter, das „dazu ausgelegt“ ist, ein Freischaltsignal an einen Taktgeber bereitzustellen, beinhaltet nicht jedes potentielle Logikgatter, das eine 1 oder 0 bereitstellen kann. Stattdessen ist das Logikgatter eines, das auf eine Weise gekoppelt ist, dass während der Operation die Ausgabe von 1 oder 0 den Taktgeber freischalten soll. Es ist erneut zu beachten, dass Verwendung des Begriffs „dazu ausgelegt“ nicht eine Operation erfordert, sondern sich stattdessen auf den latenten Status einer Vorrichtung, Hardware und/oder eines Elements fokussiert, wobei in dem latenten Status die Vorrichtung, Hardware und/oder das Element dazu designed ist, eine bestimmte Aufgabe auszuführen, wenn die Vorrichtung, Hardware und/oder das Element operiert.

[0187] Weiterhin bezieht sich die Verwendung der Phrasen „soll“ „dazu geeignet“ und/oder „operierbar,um“ in einer Implementierung auf eine Vorrichtung, Logik, Hardware und/oder ein Element, die/das auf eine solche Weise designed ist/sind, die Verwendung der Vorrichtung, Logik, Hardware und/oder des Elements auf eine spezifizierte Weise zu ermöglichen. Wie oben ist zu beachten, dass sich eine Verwendung von „soll“, „dazu geeignet“ oder „operierbar, um“ in einer Implementierung auf den latenten Status einer Vorrichtung, Logik, Hardware und/oder eines Elements bezieht, wobei die Vorrichtung, Logik, Hardware und/oder das Element nicht operiert, aber auf eine solche Weise designed ist, dass sie/es die Verwendung einer Vorrichtung in einer spezifizierten Weise ermöglicht.

[0188] Ein Wert, wie hierin verwendet, beinhaltet jegliche bekannte Darstellung einer Zahl, eines Status, eines logischen Status oder eines binären logischen Status. Oft wird die Verwendung von Logikniveaus, Logikwerten oder logischen Werten auch als 1er und 0er bezeichnet, was einfach binäre Logikstatus darstellt. Zum Beispiel bezieht sich eine 1 auf ein hohes Logikniveau, und 0 bezieht sich auf ein niedriges Logikniveau. In einer Implementierung kann eine Speicherungszelle, wie zum Beispiel eine Transistor- oder Flash-Zelle, für das Halten eines einzelnen logischen Werts oder mehrerer logischer Werte geeignet sein. Jedoch sind andere Darstellungen von Werten in Computersystemen verwendet worden. Zum Beispiel kann die Dezimalzahl zehn auch als ein binärer Wert von **1010** und ein hexadezimaler Buchstabe A dargestellt werden. Deshalb beinhaltet ein Wert jegliche Darstellung von Informationen, die dazu geeignet sind in einem Computersystem gehalten zu werden.

[0189] Außerdem können Status durch Werte oder Bereiche von Werten dargestellt werden. Als ein Beispiel kann ein erster Wert, wie zum Beispiel eine logische Eins, einen Standard- oder Anfangsstatus darstellen, während ein zweiter Wert, wie zum Beispiel eine logische Null, einen Nicht-Standardstatus darstellen kann. Zusätzlich beziehen sich die Ausdrücke Zurücksetzen und Setzen in einer Ausführungsform auf einen Standard- bzw. einen aktualisierten Wert oder Status. Zum Beispiel beinhaltet ein Standardwert potentiell einen hohen logischen Wert, das heißt zurückgesetzt, während ein aktualisierter Wert potentiell einen niedrigen logischen Wert, das heißt gesetzt, beinhaltet. Es ist zu beachten, dass jegliche Kombination von Werten verwendet werden kann, um jegliche Anzahl von Status darzustellen.

[0190] Die oben dargelegten Implementierungen von Verfahren, Hardware, Software, Firmware oder Code können über Anweisungen oder Code implementiert werden, die auf einem maschinenzugreifbaren, maschinenlesbaren, computerzugreifbaren oder computerlesbaren Medium gespeichert sind, die durch ein Verarbeitungselement ausführbar sind. Ein nichtflüchtiges maschinenzugreifbares/-lesbares Medium beinhaltet jeglichen Mechanismus, der Informationen in einer durch eine Maschine, wie zum Beispiel einen Computer oder ein elektronisches System, lesbaren Form bereitstellt (das heißt speichert und/oder überträgt). Zum Beispiel

beinhaltet ein nichtflüchtiges, maschinenzugreifbares Medium Random-Access-Memory (RAM), wie zum Beispiel Static-RAM (SRAM) oder Dynamic-RAM (DRAM); ROM; magnetisches oder optisches Speicherungsmedium; Flash-Speichergeräte; elektrische Speicherungsgeräte; optische Speicherungsgeräte; akustische Speicherungsgeräte; andere Formen von Speicherungsgeräten für das Halten von Informationen, die von flüchtigen (propagierten) Signalen (zum Beispiel Trägerwellen, Infrarotsignalen, digitalen Signalen), usw., empfangen werden, die von den nichtflüchtigen Medien unterschieden werden müssen, die Informationen davon empfangen können.

[0191] Anweisungen, die dazu verwendet werden, Logik zu programmieren, um Implementierungen der Offenbarung auszuführen, können innerhalb eines Speichers in dem System, wie zum Beispiel DRAM, Cache, Flash-Speicher oder anderer Speicherung, gespeichert werden. Darüber hinaus können die Anweisungen über ein Netzwerk oder mittels anderer computerlesbarer Medien verteilt werden. Somit kann ein maschinenlesbares Medium jeglichen Mechanismus für das Speichern oder Übertragen von Informationen in einer Form, die durch eine Maschine (zum Beispiel einen Computer) gelesen werden kann, Floppy-Disketten, optische Disks, Compact Disk, Read-Only-Memory (CD-ROM) und magneto-optische Disks, Read-Only-Memory (ROM), Random-Access-Memory (RAM), Erasable-Programmable-Read-Only-Memory(EPROM), Electrically-Erasable-Programmable-Read-Only-Memory (EEPROM), magnetische oder optische Karten, Flash-Speicher oder eine materielle, maschinenlesbare Speicherung, die in der Informationsübertragung über das Internet über elektrische, optische, akustische oder andere Formen von propagierten Signalen (zum Beispiel Trägerwellen, Infrarotsignale, digitale Signale, usw.) verwendet werden, beinhalten, ist jedoch nicht darauf beschränkt. Dementsprechend beinhaltet das computerlesbare Medium jeglichen Typ von materiellem maschinenlesbarem Medium, das für das Speichern oder das Übertragen elektronischer Anweisungen oder Informationen in einer durch eine Maschine (zum Beispiel einen Computer) lesbaren Form geeignet ist.

[0192] Über diese gesamte Spezifikation hinweg bedeutet Bezugnahme auf „(genau) eine Implementierung“ oder „eine Implementierung“, dass ein in Verbindung mit der Implementierung beschriebenes bestimmtes Merkmal, eine bestimmte Struktur oder Eigenschaft in mindestens einer Implementierung der Offenbarung enthalten ist. Somit beziehen sich die Erscheinungen der Phrasen „in (genau) einer Implementierung“ oder „in einer Implementierung“ an verschiedenen Stellen über diese gesamte Spezifikation hinweg nicht alle notwendigerweise auf die gleiche Implementierung. Darüber hinaus können die bestimmten Merkmale, Strukturen oder Eigenschaften in einer oder mehreren Implementierungen in jeglicher geeigneten Weise kombiniert werden.

[0193] In der vorstehenden Spezifikation wurde eine detaillierte Beschreibung unter Bezugnahme auf spezifische beispielhafte Implementierungen gegeben. Es wird jedoch offensichtlich, dass verschiedene Modifikationen und Änderungen daran vorgenommen werden können, ohne von dem breiteren Geist und Umfang der Offenbarung wie in den beigefügten Ansprüchen dargelegt, abzuweichen. Die Spezifikation und die Zeichnungen sind dementsprechend in einem veranschaulichenden Sinne anstelle von einem beschränkenden Sinne anzusehen. Darüber hinaus bezieht sich die vorstehende Verwendung von Implementierung und anderer beispielhafter Sprache nicht notwendigerweise auf die gleiche Implementierung oder das gleiche Beispiel, sondern auf unterschiedliche und abgegrenzte Implementierungen, sowie auf potentiell die gleiche Implementierung.

[0194] Einige Bereiche der detaillierten Beschreibung werden hinsichtlich Algorithmen und symbolischen Darstellungen von Operationen an Datenbits innerhalb eines Computerspeichers präsentiert. Diese algorithmischen Beschreibungen und Darstellungen sind die Mittel, die von Fachleuten auf dem Gebiet der Datenverarbeitung verwendet werden, um den Inhalt ihrer Arbeit anderen Fachleuten am effektivsten zu vermitteln. Ein Algorithmus wird hier und allgemein als eine selbstkonsistente Sequenz von Operationen verstanden, die zu einem angestrebten Ergebnis führt. Die Operationen sind jene, die physikalische Manipulationen physikalischer Größen erfordern. Üblicherweise, jedoch nicht notwendigerweise, nehmen diese Größen die Form von elektrischen oder magnetischen Signalen an, die dazu geeignet sind, gespeichert, übertragen, kombiniert, verglichen und anderweitig manipuliert zu werden. Es hat sich bisweilen, hauptsächlich aus Gründen der üblichen Verwendung, als zweckmäßig erwiesen, diese Signale als Bits, Werte, Elemente, Symbole, Zeichen, Begriffe, Zahlen oder dergleichen zu bezeichnen. Die hierin beschriebenen Blöcke können Hardware, Software, Firmware oder eine Kombination davon sein.

[0195] Es sollte jedoch berücksichtigt werden, dass alle diese und ähnliche Begriffe den geeigneten physikalischen Größen zugeordnet werden müssen, und lediglich auf diese Größen angewendete zweckmäßige Etiketten sind. Sofern nicht ausdrücklich anders als offensichtlich von der obigen Diskussion erwähnt, wird es anerkannt, dass über die gesamte Beschreibung hinweg Diskussionen, die Begriffe wie „Definieren“, „Empfangen“, „Bestimmen“, „Ausgeben“, „Verbinden“, „Zuordnen“, „Erhalten“, „Authentifizieren“, „Verhindern“, „Ausführen“, „Anfordern“, „Kommunizieren“ oder dergleichen verwenden, sich auf die Aktionen und Prozesse eines

Computersystems oder ähnlichen elektronischen Computergeräts beziehen, das als physikalische (zum Beispiel elektronische) Größen dargestellte Daten innerhalb der Register und Speicher des Computersystems in andere Daten manipuliert und transformiert, die in ähnlicher Weise als physikalische Größen innerhalb der Speicher oder Register des Computersystems oder anderer solcher Informationsspeicherungs-, Übertragungs- oder Anzeigegeräte dargestellt werden.

[0196] Die Wörter „Beispiel“ oder „beispielhaft“ werden hierin so verwendet, dass sie bedeuten, als Beispiel, Fall oder Veranschaulichung zu dienen. Jeglicher hierin als „Beispiel“ oder „beispielhaft“ beschriebene Aspekt oder Design ist nicht notwendigerweise als bevorzugt oder vorteilhaft gegenüber anderen Aspekten oder Designs auszulegen. Vielmehr ist die Verwendung der Wörter „Beispiel“ oder „beispielhaft“ dazu beabsichtigt, Konzepte in einer konkreten Weise zu präsentieren. Wie in dieser Anmeldung verwendet, ist der Begriff „oder“ dazu beabsichtigt, ein einschließendes „oder“ anstelle eines ausschließenden „oder“ zu bedeuten. Das heißt, wenn nicht anderweitig spezifiziert oder aus dem Kontext klar hervorgehend, ist „**X** beinhaltet **A** oder **B**“ dazu beabsichtigt, jegliche der natürlichen einschließenden Permutationen zu bedeuten. Das heißt, wenn **X A** beinhaltet; **X B** beinhaltet; oder **X** sowohl **A**, als auch **B** beinhaltet, dann ist „**X** beinhaltet **A** oder **B**“ in jeglichem der vorstehenden Fälle erfüllt. Zusätzlich sollte der Artikel „ein“, wie in dieser Anmeldung und den beigefügten Ansprüchen verwendet, allgemein so ausgelegt werden, dass er „ein oder mehrere“ bedeutet, sofern nicht anders spezifiziert oder aus dem Kontext klar hervorgeht, dass er auf eine Singularform gerichtet ist. Außerdem ist die Verwendung des Begriffs „eine Implementierung“ oder „(genau) eine Implementierung“ durchgängig nicht dazu beabsichtigt, die gleiche Implementierung oder Implementierung zu bedeuten, sofern nicht als solche beschrieben. Auch sind die hierin verwendeten Begriffe „erstes“, „zweites“, „drittes“, „viertes“, usw. als Etiketten gemeint, um zwischen unterschiedlichen Elementen zu unterscheiden, und können nicht notwendigerweise eine ordinale Bedeutung gemäß ihrer numerischen Bezeichnung haben.

Patentansprüche

1. Prozessor, der Folgendes umfasst:
eine sichere Enklavenschaltung, die eine sichere Speicherungsstruktur zum Speichern einer Enklave-Page-Cache-Map umfasst, wobei die Enklave-Page-Cache-Map Inhalte einer sicheren Enklave innerhalb von Systemspeicher verfolgen soll, wobei die sichere Enklave sichere Daten, die eine Seite mit einer virtuellen Adresse umfassen, speichern soll; und
eine operativ mit der sicheren Enklavenschaltung gekoppelte Ausführungseinheit, wobei die Ausführungseinheit als Reaktion auf eine Anforderung, die Seite von der sicheren Enklave zu räumen, mehrere decodierte Anweisungen ausführen soll, um:
die Erzeugung von Übersetzungen der virtuellen Adresse zu blocken;
einen oder mehrere aktuell auf die sicheren Daten in der sicheren Enklave zugreifende Hardware-Threads aufzuzeichnen;
einen Inter-Processor-Interrupt an einen oder mehrere dem einen oder den mehreren Hardware-Threads zugeordnete Kerne zu senden, um den einen oder die mehreren Hardware-Threads dazu zu veranlassen, die sichere Enklave zu verlassen und Translation-Lookaside-Buffer des einen oder der mehreren Kerne zu leeren; und
als Reaktion auf die Erkennung eines der virtuellen Adresse für die Seite in der sicheren Enklave zugeordneten Seitenfehlers die Erzeugung weiterer Übersetzungen der virtuellen Adresse freizugeben.

2. Prozessor nach Anspruch 1, wobei die Ausführungseinheit ferner den der Seite zugeordneten Seitenfehler erkennen soll, bevor die Ausführungseinheit eine Zurückschreibanweisung ausführen soll, um die Seite auf Festplatte zurückzuschreiben.

3. Prozessor nach einem der Ansprüche 1-2, wobei, um die Erzeugung der weiteren Übersetzungen der virtuellen Adresse zu blocken, die Ausführungseinheit ferner die mehreren decodierten Anweisungen ausführen soll, um:
die Seite als nicht in mehreren Übersetzungsseitentabellen vorhanden zu kennzeichnen;
den Ablauf einer Zeitdauer, während derer nicht auf die Seite zugegriffen wird, zu erkennen; und
ein Bit in der Enklave-Page-Cache-Map für die zu der virtuellen Adresse abgebildete Seite zu setzen, um die Erzeugung weiterer Übersetzungen der virtuellen Adresse für Zugriff auf die Seite zu blocken.

4. Prozessor nach Anspruch 3, wobei, um die Erzeugung von Übersetzungen der virtuellen Adresse freizugeben, die Ausführungseinheit eine decodierte Freigabeanweisung ausführen soll, um:
das Bit in der Enklave-Page-Cache-Map für die zu der virtuellen Adresse abgebildete Seite zu löschen; und
die Seite als in den mehreren Übersetzungstabellen vorhanden zu markieren.

5. Prozessor nach einem der Ansprüche 1-4, wobei, um die Erzeugung von Übersetzungen der virtuellen Adresse freizugeben, die Ausführungseinheit eine decodierte Freigabeanweisung ausführen soll, um:
zu verifizieren, dass es keinen dem Zugriff auf die Seite in der sicheren Enklave zugeordneten Ressourcenkonflikt gibt;
von einem ersten Eingaberegister die virtuelle Adresse für die Seite in der sicheren Enklave zu lesen;
die virtuelle Adresse in eine physikalische Guest-Adresse und die physikalische Guest-Adresse in eine physikalische Host-Adresse zu übersetzen; und
über einen Zugriff auf die Enklave-Page-Cache-Map zu verifizieren, dass die physikalische Host-Adresse der Seite der sicheren Enklave in dem Systemspeicher zugeordnet ist.

6. Prozessor nach Anspruch 5, wobei die Ausführungseinheit ferner die decodierte Freigabeanweisung ausführen soll, um:
eine physikalische Host-Adresse einer Secure-Enclave-Control-Structure (SECS), die der sicheren Enklave über Bezug zu einem Eintrag in der Enklave-Page-Cache-Map für die Seite zugeordnet ist, zu bestimmen;
durch Verwenden der physikalischen Host-Adresse der SECS, auf eine Epochenverfolgungsstruktur für die sichere Enklave zuzugreifen;
von der Epochenverfolgungsstruktur einen vorherigen Epochenwert und einen vorherigen Epochenzählerwert in einen Satz von lokalen Epochenregistern zu lesen;
basierend auf dem vorherigen Epochenwert und dem vorherigen Epochenzählerwert zu verifizieren, dass der eine oder die mehreren Hardware-Threads die sichere Enklave verlassen haben; und
den der sicheren Enklave zugeordneten vorherigen Epochenwert zu löschen.

7. Prozessor nach Anspruch 6, wobei das Löschen des der sicheren Enklave zugeordneten vorherigen Epochenwerts vor dem Löschen eines geblockten Bits in der Enklave-Page-Cache-Map für die virtuelle Adresse ausgeführt wird.

8. System, das Folgendes umfasst:
mehrere Prozessorkerne, von denen jeder einen Translation-Lookaside-Buffer (TLB) umfasst;
eine sichere Enklavenschaltung, die eine sichere Speicherungsstruktur zum Speichern einer Enklave-Page-Cache-Map umfasst, wobei die Enklave-Page-Cache-Map Inhalte einer sicheren Enklave innerhalb von Systemspeicher verfolgen soll, wobei die sichere Enklave sichere Daten, die eine Seite mit einer virtuellen Adresse umfassen, speichern soll; und
eine operativ mit der sicheren Enklavenschaltung und den mehreren Prozessorkernen gekoppelte Ausführungseinheit, wobei die Ausführungseinheit als Reaktion auf eine Anforderung, die Seite von der sicheren Enklave zu räumen, mehrere decodierte Anweisungen ausführen soll, um:
die Erzeugung von Übersetzungen der virtuellen Adresse zu blocken;
einen oder mehrere aktuell auf die sicheren Daten in der sicheren Enklave zugreifende Hardware-Threads aufzuzeichnen;
einen Inter-Processor-Interrupt an einen oder mehrere der mehreren dem einen oder den mehreren Hardware-Threads zugeordneten Prozessorkerne zu senden, um den einen oder die mehreren Hardware-Threads dazu zu veranlassen, die sichere Enklave zu verlassen und den TLB des einen oder der mehreren der mehreren Prozessorkerne zu leeren; und
als Reaktion auf die Erkennung eines der virtuellen Adresse für die Seite in der sicheren Enklave zugeordneten Seitenfehlers die Erzeugung weiterer Übersetzungen der virtuellen Adresse freizugeben.

9. System nach Anspruch 8, wobei die Ausführungseinheit ferner den der Seite zugeordneten Seitenfehler erkennen soll, bevor die Ausführungseinheit eine Zurückschreibanweisung ausführen soll, um die Seite auf Festplatte zurückzuschreiben.

10. System nach einem der Ansprüche 8-9, wobei, um die Erzeugung der Übersetzungen der virtuellen Adresse zu blocken, die Ausführungseinheit ferner die mehreren decodierten Anweisungen ausführen soll, um:
die Seite als nicht in mehreren Übersetzungsseitentabellen vorhanden zu kennzeichnen;
den Ablauf einer Zeitdauer, während derer nicht auf die Seite zugegriffen wird, zu erkennen; und
ein Bit in der Enklave-Page-Cache-Map für die zu der virtuellen Adresse abgebildete Seite zu setzen, um die Erzeugung weiterer Übersetzungen der virtuellen Adresse für Zugriff auf die Seite zu blocken.

11. System nach Anspruch 10, Prozessor nach Anspruch 3, wobei, um die Erzeugung von Übersetzungen der virtuellen Adresse freizugeben, die Ausführungseinheit eine decodierte Freigabeanweisung ausführen soll, um:
das Bit in der Enklave-Page-Cache-Map für die zu der virtuellen Adresse abgebildete Seite zu löschen; und

die Seite als in den mehreren Übersetzungstabellen vorhanden zu markieren.

12. System nach einem der Ansprüche 8-11, wobei, um die Erzeugung von Übersetzungen der virtuellen Adresse freizugeben, die Ausführungseinheit eine decodierte Freigabeanweisung ausführen soll, um:
zu verifizieren, dass es keinen dem Zugriff auf die Seite in der sicheren Enklave zugeordneten Ressourcenkonflikt gibt;
von einem ersten Eingaberegister die virtuelle Adresse für die Seite in der sicheren Enklave zu lesen;
die virtuelle Adresse in eine physikalische Guest-Adresse und die physikalische Guest-Adresse in eine physikalische Host-Adresse zu übersetzen; und
über einen Zugriff auf die Enklave-Page-Cache-Map zu verifizieren, dass die physikalische Host-Adresse der Seite der sicheren Enklave in dem Systemspeicher zugeordnet ist.

13. System nach Anspruch 12, wobei die Ausführungseinheit ferner die decodierte Freigabeanweisung ausführen soll, um:
eine physikalische Host-Adresse einer Secure-Enclave-Control-Structure (SECS), die der sicheren Enklave über Bezug zu einem Eintrag in der Enklave-Page-Cache-Map für die Seite zugeordnet ist, zu bestimmen;
durch Verwenden der physikalischen Host-Adresse der SECS auf eine Epochenverfolgungsstruktur für die sichere Enklave zuzugreifen;
von der Epochenverfolgungsstruktur einen vorherigen Epochenwert und einen vorherigen Epochenzählerwert in einen Satz von lokalen Epochenregistern zu lesen;
basierend auf dem vorherigen Epochenwert und dem vorherigen Epochenzählerwert zu verifizieren, dass der eine oder die mehreren Hardware-Threads die sichere Enklave verlassen haben; und
den der sicheren Enklave zugeordneten vorherigen Epochenwert zu löschen.

14. System nach Anspruch 13, wobei das Löschen des der sicheren Enklave zugeordneten vorherigen Epochenwerts vor dem Löschen eines geblockten Bits in der Enklave-Page-Cache-Map für die virtuelle Adresse ausgeführt wird.

15. Verfahren, das Folgendes umfasst:
das Speichern, in einer sicheren Enklave des Hauptspeichers eines Computersystems, sicherer Daten, die eine Seite mit einer virtuellen Adresse umfassen;
das Speichern, in einer sicheren Speicherungsstruktur einer sicheren Enklavenschaltung des Computersystems, einer Enklave-Page-Cache-Map, um Inhalte der sicheren Enklave innerhalb des Systemspeichers zu verfolgen; und
als Reaktion auf eine Anforderung, die Seite von der sicheren Enklave zu räumen:
das Blocken der Erzeugung von Übersetzungen der virtuellen Adresse;
das Aufzeichnen eines oder mehrerer aktuell auf die sicheren Daten in der sicheren Enklave zugreifender Hardware-Threads;
das Senden eines Inter-Processor-Interrupt an einen oder mehrere dem einen oder den mehreren Hardware-Threads zugeordnete Kerne, um den einen oder die mehreren Hardware-Threads dazu zu veranlassen, die sichere Enklave zu verlassen und Translation-Lookaside-Buffer des einen oder der mehreren Kerne zu leeren; und
als Reaktion auf die Erkennung eines der virtuellen Adresse für die Seite in der sicheren Enklave zugeordneten Seitenfehlers das Freigeben der Erzeugung von Übersetzungen der virtuellen Adresse.

16. Verfahren nach Anspruch 15, das ferner das Erkennen des der Seite zugeordneten Seitenfehlers vor dem Ausführen einer Zurückschreibanweisung, um die Seite auf Festplatte zurückzuschreiben, umfasst.

17. Verfahren nach einem der Ansprüche 15-16, wobei das Blocken der Erzeugung von Übersetzungen der virtuellen Adresse Folgendes umfasst:
das Markieren der Seite als nicht in mehreren Übersetzungsseitentabellen vorhanden;
das Erkennen eines Ablaufs einer Zeitdauer, während derer nicht auf die Seite zugegriffen wird; und
das Setzen eines Bits in der Enklave-Page-Cache-Map für die zu der virtuellen Adresse abgebildete Seite, um die Erzeugung von Übersetzungen der virtuellen Adresse für Zugriff auf die Seite zu blocken.

18. Verfahren nach Anspruch 17, wobei das Freigeben der Erzeugung von Übersetzungen der virtuellen Adresse Folgendes umfasst:
das Löschen des Bits in der Enklave-Page-Cache-Map für die zu der virtuellen Adresse abgebildete Seite; und
das Markieren der Seite als in den mehreren Übersetzungstabellen vorhanden.

19. Verfahren nach einem der Ansprüche 15-18, wobei das Freigeben der Erzeugung von Übersetzungen der virtuellen Adresse Folgendes umfasst:
das Verifizieren, dass es keinen dem Zugriff auf die Seite in der sicheren Enklave zugeordneten Ressourcenkonflikt gibt;
das Lesen, von einem ersten Eingaberegister, der virtuellen Adresse für die Seite in der sicheren Enklave;
das Übersetzen der virtuellen Adresse in eine physikalische Guest-Adresse und der physikalischen Guest-Adresse in eine physikalische Host-Adresse; und
das Verifizieren, über einen Zugriff auf die Enklave-Page-Cache-Map, dass die physikalische Host-Adresse der Seite der sicheren Enklave in dem Systemspeicher zugeordnet ist.

20. Verfahren nach Anspruch 19, das ferner Folgendes umfasst:
das Bestimmen einer physikalischen Host-Adresse einer Secure-Enclave-Control-Structure (SECS), die der sicheren Enklave über Bezug zu einem Eintrag in der Enklave-Page-Cache-Map für die Seite zugeordnet ist;
das Zugreifen, durch Verwenden der physikalischen Host-Adresse der SECS, auf eine Epochenverfolgungsstruktur für die sichere Enklave;
das Lesen, von der Epochenverfolgungsstruktur, eines vorherigen Epochenwerts und eines vorherigen Epochenzählerwerts in einen Satz von lokalen Epochenregistern;
das Verifizieren, basierend auf dem vorherigen Epochenwert und dem vorherigen Epochenzählerwert, dass der eine oder die mehreren Hardware-Threads die sichere Enklave verlassen haben; und
das Löschen des der sicheren Enklave zugeordneten vorherigen Epochenwerts vor dem Löschen eines geblockten Bits in der Enklave-Page-Cache-Map für die virtuelle Adresse.

21. Prozessor, der Folgendes umfasst:
eine sichere Enklavenschaltung, die eine sichere Speicherungsstruktur zum Speichern einer Enklave-Page-Cache-Map (EPCM) umfasst, wobei die EPCM Inhalte einer sicheren Enklave innerhalb von Systemspeichers verfolgen soll, wobei die sichere Enklave sichere Daten, die eine Seite an einer virtuellen Adresse umfassen, speichern soll, und wobei die EPCM ein erstes geblocktes Bit speichern soll, um anzuzeigen, ob die Seite geblockt ist; und
einen operativ mit der sicheren Enklavenschaltung gekoppelten Prozessorkern, wobei der Prozessorkern einen virtuellen Maschinenmonitor (VMM) ausführen soll, um das Auslagern innerhalb der sicheren Enklave für eine virtuelle Maschine (VM) zu verwalten, wobei der VMM, um die Seite von der sicheren Enklave zu räumen:
das erste geblockte Bit in der EPCM setzen soll, um anzuzeigen, dass die Seite von weiterer Übersetzung der virtuellen Adresse für Zugriff auf die Seite geblockt ist;
ein Blockstatusbit erzeugen soll, das einen Status des ersten geblockten Bits anzeigt, um ein zweites geblocktes Bit innerhalb eines ersten Feldes einer in dem Systemspeicher gespeicherten Informationsdatenstruktur zu setzen;
eine Zurückschreibanweisung ausführen soll, um einen Wert des zweiten geblockten Bits in ein geblocktes Bit-Flag einer Unified-Metadata-Structure (UMDS) zu kopieren, wobei die UMDS in einer Versions-Array-Seite der sicheren Enklave gespeichert werden soll; und
die Seite von der sicheren Enklave in eine des Systemspeichers, außerhalb der sicheren Enklave, oder auf Festplatte entfernen soll.

22. Prozessor nach Anspruch 21, wobei die Informationsdatenstruktur ferner ein zweites Feld umfasst, um ein USE_UMDS-Bit zu halten, wobei zum Wiederherstellen der Seite von der einen des Systemspeichers oder der Festplatte, auf die die Seite während der Ausführung der Zurückschreibanweisung geschrieben wurde, der VMM eine Ladeanweisung ausführen soll, um:
innerhalb eines Eingaberegisters auf eine Adresse eines Platzes der Informationsdatenstruktur zuzugreifen;
zu verifizieren, dass das USE_UMDS-Bit gesetzt ist; und
das erste geblockte Bit der EPCM mit einem Wert des geblockten Bit-Flags, der von der UMDS gelesen wird, zu laden.

23. Prozessor nach Anspruch 21, wobei die Informationsdatenstruktur ferner ein zweites Feld umfasst, um ein USE_UMDS-Bit zu halten, wobei zum Wiederherstellen der Seite von der einen des Systemspeichers oder der Festplatte, auf die die Seite während der Ausführung der Zurückschreibanweisung geschrieben wurde, der VMM eine Ladeanweisung ausführen soll, um:
innerhalb eines Eingaberegisters auf eine Adresse eines Platzes der Informationsdatenstruktur zuzugreifen;
zu verifizieren, dass das USE_UMDS-Bit gelöscht ist; und
das erste geblockte Bit der EPCM mit einem Wert des zweiten geblockten Bits, der von der Informationsdatenstruktur gelesen wird, zu laden.

24. Prozessor nach einem der Ansprüche 21-23, wobei der VMM ferner:
in einem ersten Eingaberegister auf einen ersten Zeiger auf die virtuelle Adresse der Seite in der sicheren Enklave zugreifen soll;
in einem zweiten Eingaberegister auf einen zweiten Zeiger auf die Versions-Array-Seite in der sicheren Enklave zugreifen soll;
in einem dritten Eingaberegister auf einen dritten Zeiger auf einen Speicherplatz außerhalb der sicheren Enklave, auf den die Seite gespeichert werden soll, zugreifen soll;
einen Message-Authentication-Code (MAC) der Seite unter Verwendung einer Versionsnummer erzeugen soll;
Daten für die Seite unter Verwendung eines geheimen Schlüssels verschlüsseln soll, um verschlüsselte Daten für die Seite zu erzeugen;
die innerhalb der Versions-Array-Seite gespeicherte UMDS mit der Versionsnummer und dem MAC bestücken soll;
die verschlüsselten Daten in den Speicherplatz außerhalb der sicheren Enklave schreiben soll;
Auslagerungs-Krypto-Metadaten für die Seite in die UMDS schreiben soll; und
die Seite von der sicheren Enklave zurück auf die Festplatte entfernen soll.

25. Prozessor nach Anspruch 24, wobei der VMM, um die Seite wiederherzustellen, eine Ladeanweisung ausführen soll, um:
in einem ersten Ausgaberegister auf einen vierten Zeiger auf die verschlüsselte Seite an dem Speicherplatz zuzugreifen;
in einem vierten Eingaberegister auf einen fünften Zeiger auf einen freien Seitenplatz in der sicheren Enklave zuzugreifen;
in dem zweiten Eingaberegister auf den zweiten Zeiger auf eine virtuelle Adresse eines Versions-Array-Slots in der sicheren Enklave zuzugreifen;
die virtuelle Adresse in eine physikalische Guest-Adresse und die physikalische Guest-Adresse in eine physikalische Host-Adresse für die Versions-Array-Seite zu übersetzen;
auf die UMDS innerhalb der Versions-Array-Seite an der physikalischen Host-Adresse des Versions-Array-Slots zuzugreifen;
die verschlüsselte Seite durch Verwenden des MAC und der Versionsnummer zu verifizieren;
die verschlüsselten Daten zu entschlüsseln, um entschlüsselte Daten für die Seite zu erzeugen;
die entschlüsselten Daten in der sicheren Enklave an dem freien Seitenplatz zu speichern; und
einen Slot innerhalb der UMDS für einen nächsten Versionswert zu löschen.

26. Nichtflüchtiges computerlesbares Speicherungsmedium zum Speichern von Anweisungen, die, wenn durch einen Prozessor ausgeführt, mehrere Operationen ausführen sollen, die Folgendes umfassen:
das Speichern, in einer sicheren Enklave des Hauptspeichers eines Computersystems, sicherer Daten, die eine Seite mit einer virtuellen Adresse umfassen;
das Speichern, in einer sicheren Speicherungsstruktur einer sicheren Enklavenschaltung des Computersystems, einer Enklave-Page-Cache-Map, um Inhalte der sicheren Enklave innerhalb des Systemspeichers zu verfolgen; und
als Reaktion auf eine Anforderung die Seite von der sicheren Enklave zu räumen:
das Blocken der Erzeugung von Übersetzungen der virtuellen Adresse;
das Aufzeichnen eines oder mehrerer aktuell auf die sicheren Daten in der sicheren Enklave zugreifender Hardware-Threads;
das Senden eines Inter-Processor-Interrupt an einen oder mehrere dem einen oder den mehreren Hardware-Threads zugeordnete Kerne, um den einen oder die mehreren Hardware-Threads dazu zu veranlassen, die sichere Enklave zu verlassen und Translation-Lookaside-Buffer des einen oder der mehreren Kerne zu leeren; und als Reaktion auf die Erkennung eines der virtuellen Adresse für die Seite in der sicheren Enklave zugeordneten Seitenfehlers das Freigeben der Erzeugung von Übersetzungen der virtuellen Adresse.

27. Nichtflüchtiges computerlesbares Speicherungsmedium nach Anspruch 26, wobei die mehreren Operationen ferner das Erkennen des der Seite zugeordneten Seitenfehlers vor der Ausführung einer Zurückschreibanweisung, um die Seite auf Festplatte zurückzuschreiben, umfassen.

28. Nichtflüchtiges computerlesbares Speicherungsmedium nach einem der Ansprüche 26-27, wobei das Blocken der Erzeugung von Übersetzungen der virtuellen Adresse Folgendes umfasst:
das Markieren der Seite als nicht in mehreren Übersetzungsseitentabellen vorhanden;
das Erkennen eines Ablaufs einer Zeitdauer, während derer nicht auf die Seite zugegriffen wird; und
das Setzen eines Bits in der Enklave-Page-Cache-Map für die zu der virtuellen Adresse abgebildete Seite, um die Erzeugung von Übersetzungen der virtuellen Adresse für Zugriff auf die Seite zu blocken.

29. Nichtflüchtiges computerlesbares Speicherungsmedium nach Anspruch 28, wobei das Freigeben der Erzeugung von Übersetzungen der virtuellen Adresse Folgendes umfasst:
das Löschen des Bits in der Enklave-Page-Cache-Map für die zu der virtuellen Adresse abgebildete Seite; und
das Markieren der Seite als in den mehreren Übersetzungstabellen vorhanden.

30. Nichtflüchtiges computerlesbares Speicherungsmedium nach einem der Ansprüche 26-29, wobei das Freigeben der Erzeugung von Übersetzungen der virtuellen Adresse Folgendes umfasst:
das Verifizieren, dass es keinen dem Zugriff auf die Seite in der sicheren Enklave zugeordneten Ressourcenkonflikt gibt;
das Lesen, von einem ersten Eingaberegister, der virtuellen Adresse für die Seite in der sicheren Enklave;
das Übersetzen der virtuellen Adresse in eine physikalische Guest-Adresse und der physikalischen Guest-Adresse in eine physikalische Host-Adresse; und
das Verifizieren, über einen Zugriff auf die Enklave-Page-Cache-Map, dass die physikalische Host-Adresse der Seite der sicheren Enklave in dem Systemspeicher zugeordnet ist.

31. Nichtflüchtiges computerlesbares Speicherungsmedium nach Anspruch 30, wobei die mehreren Operationen ferner Folgendes umfassen:
das Bestimmen einer physikalischen Host-Adresse einer Secure-Enclave-Control-Structure (SECS), die der sicheren Enklave über Bezug zu einem Eintrag in der Enklave-Page-Cache-Map für die Seite zugeordnet ist;
das Zugreifen, durch Verwenden der physikalischen Host-Adresse der SECS, auf eine Epochenverfolgungsstruktur für die sichere Enklave;
das Lesen, von der Epochenverfolgungsstruktur, eines vorherigen Epochenwerts und eines vorherigen Epochenzählerwerts in einen Satz von lokalen Epochenregistern;
das Verifizieren, basierend auf dem vorherigen Epochenwert und dem vorherigen Epochenzählerwert, dass der eine oder die mehreren Hardware-Threads die sichere Enklave verlassen haben; und
das Löschen des der sicheren Enklave zugeordneten vorherigen Epochenwerts vor dem Löschen eines geblockten Bits in der Enklave-Page-Cache-Map für die virtuelle Adresse.

32. Vorrichtung, die Folgendes umfasst:
Mittel für das Speichern, in einer sicheren Enklave des Hauptspeichers eines Computersystems, sicherer Daten, die eine Seite mit einer virtuellen Adresse umfassen;
Mittel für das Speichern, in einer sicheren Speicherungsstruktur einer sicheren Enklavenschaltung des Computersystems, einer Enklave-Page-Cache-Map, um Inhalte der sicheren Enklave innerhalb des Systemspeichers zu verfolgen; und
als Reaktion auf eine Anforderung, die Seite von der sicheren Enklave zu räumen:
Mittel für das Blocken der Erzeugung von Übersetzungen der virtuellen Adresse;
Mittel für das Aufzeichnen eines oder mehrerer aktuell auf die sicheren Daten in der sicheren Enklave zugreifender Hardware-Threads;
Mittel für das Senden eines Inter-Processor-Interrupt an einen oder mehrere dem einen oder den mehreren Hardware-Threads zugeordnete Kerne, um den einen oder die mehreren Hardware-Threads dazu zu veranlassen, die sichere Enklave zu verlassen und Translation-Lookaside-Buffer des einen oder der mehreren Kerne zu leeren; und
als Reaktion auf die Erkennung eines der virtuellen Adresse für die Seite in der sicheren Enklave zugeordneten Seitenfehlers, Mittel für das Freigeben der Erzeugung von Übersetzungen der virtuellen Adresse.

33. Vorrichtung nach Anspruch 32, die ferner Mittel für das Erkennen des der Seite zugeordneten Seitenfehlers vor dem Ausführen einer Zurückschreibanweisung, um die Seite auf Festplatte zurückzuschreiben, umfasst.

34. Vorrichtung nach einem der Ansprüche 32-33, wobei die Mittel für das Blocken der Erzeugung von Übersetzungen der virtuellen Adresse Folgendes umfassen:
Mittel für das Markieren der Seite als nicht in mehreren Übersetzungsseitentabellen vorhanden;
Mittel für das Erkennen des Ablaufs einer Zeitdauer, während derer nicht auf die Seite zugegriffen wird; und
Mittel für das Setzen eines Bits in der Enklave-Page-Cache-Map für die zu der virtuellen Adresse abgebildete Seite, um die Erzeugung von Übersetzungen der virtuellen Adresse für Zugriff auf die Seite zu blocken.

35. Vorrichtung nach Anspruch 34, wobei die Mittel für das Freigeben der Erzeugung von Übersetzungen der virtuellen Adresse Folgendes umfassen:
Mittel für das Löschen des Bits in der Enklave-Page-Cache-Map für die zu der virtuellen Adresse abgebildete Seite; und

Mittel für das Markieren der Seite als in den mehreren Übersetzungstabellen vorhanden.

36. Vorrichtung nach einem der Ansprüche 32-35, wobei die Mittel für das Freigeben der Erzeugung von Übersetzungen der virtuellen Adresse Folgendes umfassen:
Mittel für das Verifizieren, dass es keinen dem Zugriff auf die Seite in der sicheren Enklave zugeordneten Ressourcenkonflikt gibt;
Mittel für das Lesen, von einem ersten Eingaberegister, der virtuellen Adresse für die Seite in der sicheren Enklave;
Mittel für das Übersetzen der virtuellen Adresse in eine physikalische Guest-Adresse und der physikalischen Guest-Adresse in eine physikalische Host-Adresse; und
Mittel für das Verifizieren, über einen Zugriff auf die Enklave-Page-Cache-Map, dass die physikalische Host-Adresse der Seite der sicheren Enklave in dem Systemspeicher zugeordnet ist.

37. Vorrichtung nach Anspruch 36, die ferner Folgendes umfasst:
Mittel für das Bestimmen einer physikalischen Host-Adresse einer Secure-Enclave-Control-Structure (SECS), die der sicheren Enklave über Bezug zu einem Eintrag in der Enklave-Page-Cache-Map für die Seite zugeordnet ist;
Mittel für das Zugreifen, durch Verwenden der physikalischen Host-Adresse der SECS, auf eine Epochenverfolgungsstruktur für die sichere Enklave;
Mittel für das Lesen, von der Epochenverfolgungsstruktur, eines vorherigen Epochenwerts und eines vorherigen Epochenzählerwerts in einen Satz von lokalen Epochenregistern;
Mittel für das Verifizieren, basierend auf dem vorherigen Epochenwert und dem vorherigen Epochenzählerwert, dass der eine oder die mehreren Hardware-Threads die sichere Enklave verlassen haben; und
Mittel für das Löschen des der sicheren Enklave zugeordneten vorherigen Epochenwerts vor dem Löschen eines geblockten Bits in der Enklave-Page-Cache-Map für die virtuelle Adresse.

Es folgen 27 Seiten Zeichnungen

Anhängende Zeichnungen

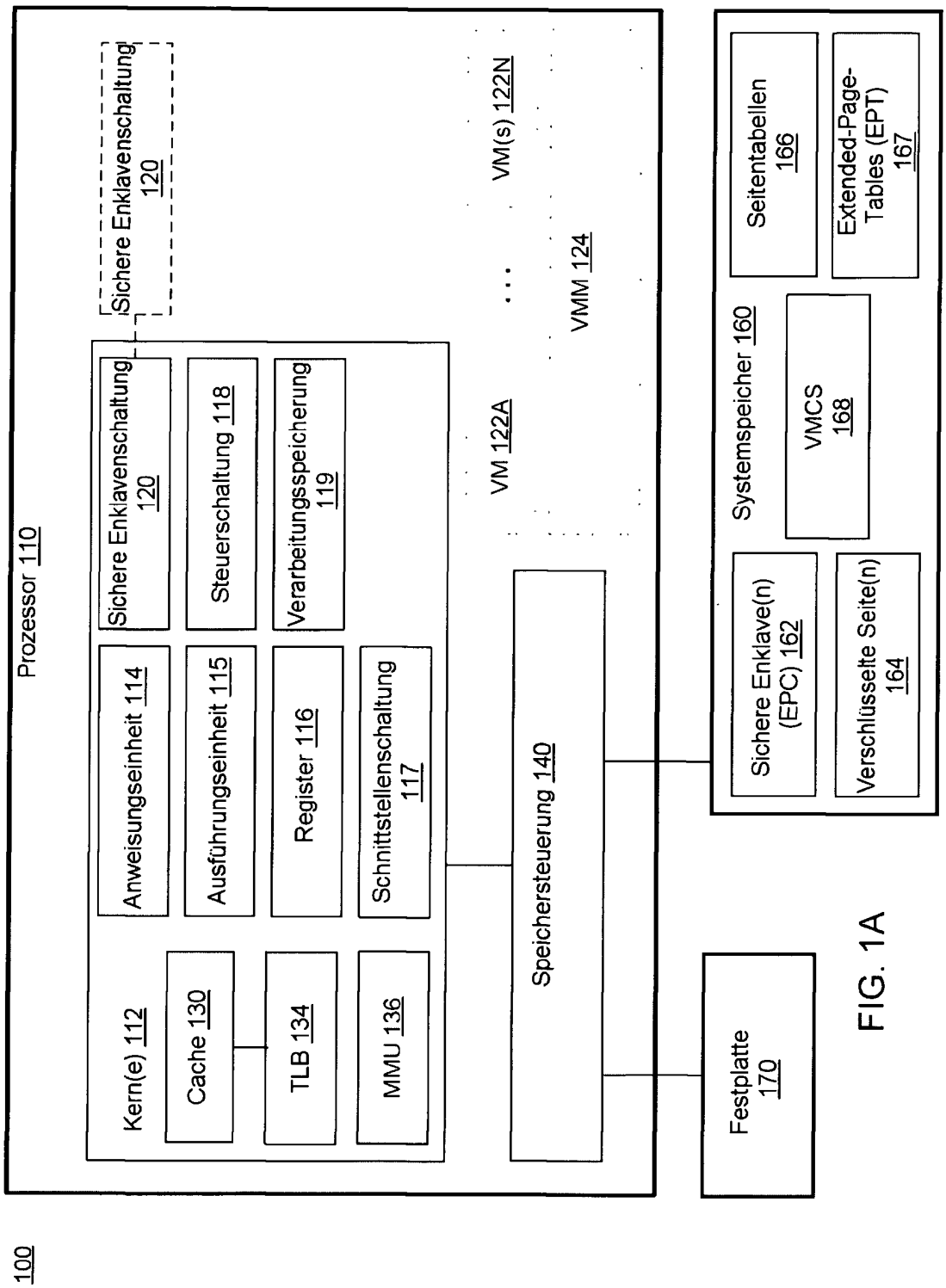


FIG. 1A

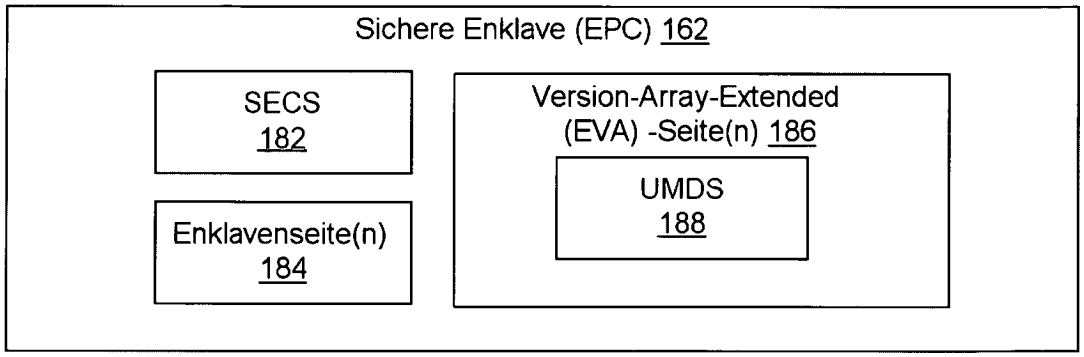

Sichere Enklave (EPC) 162
SECS
182
Enklavenseite(n)
184
Version-Array-Extended
(EVA) -Seite(n) 186
UMDS
188

FIG. 1B

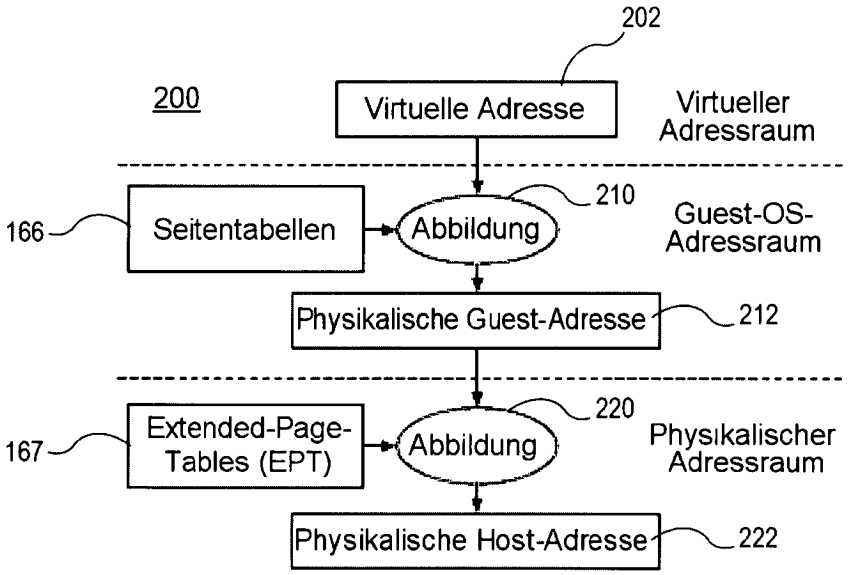

202
200
Virtuelle Adresse
Virtueller
Adressraum
166
Seitentabellen
Abbildung
210
Guest-OS-
Adressraum
Physikalische Guest-Adresse
212
167
Extended-Page-
Tables (EPT)
Abbildung
220
Physikalischer
Adressraum
Physikalische Host-Adresse
222

FIG. 2

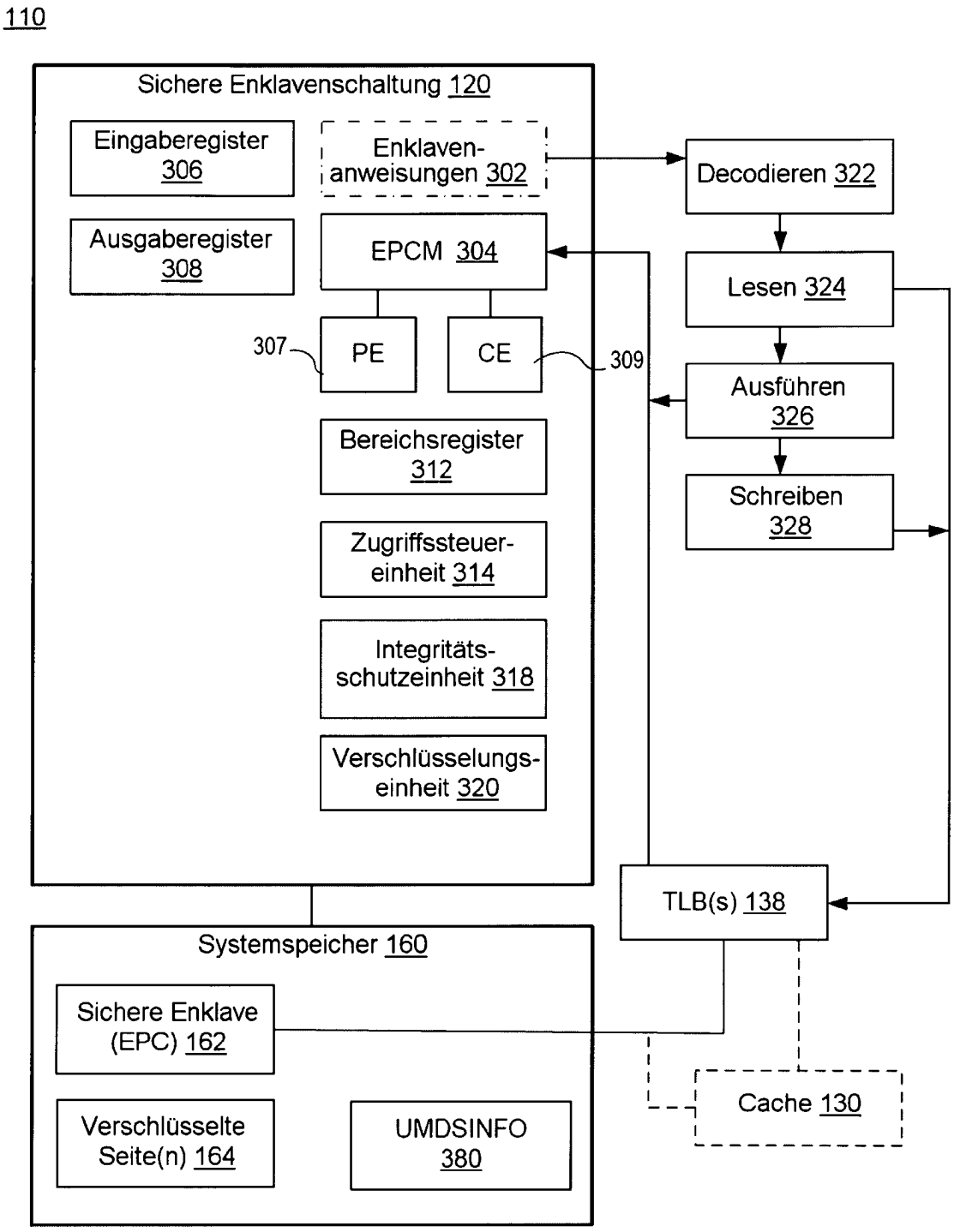

110
Sichere Enklavenschaltung 120
Eingaberegister 306
Enklaven-anweisungen 302
Ausgaberegister 308
EPCM 304
307
PE
CE
309
Bereichsregister 312
Zugriffssteuer-einheit 314
Integritäts-schutzeinheit 318
Verschlüsselungs-einheit 320
Decodieren 322
Lesen 324
Ausführen 326
Schreiben 328
TLB(s) 138
Systemspeicher 160
Sichere Enklave (EPC) 162
Verschlüsselte Seite(n) 164
UMDSINFO 380
Cache 130

FIG. 3

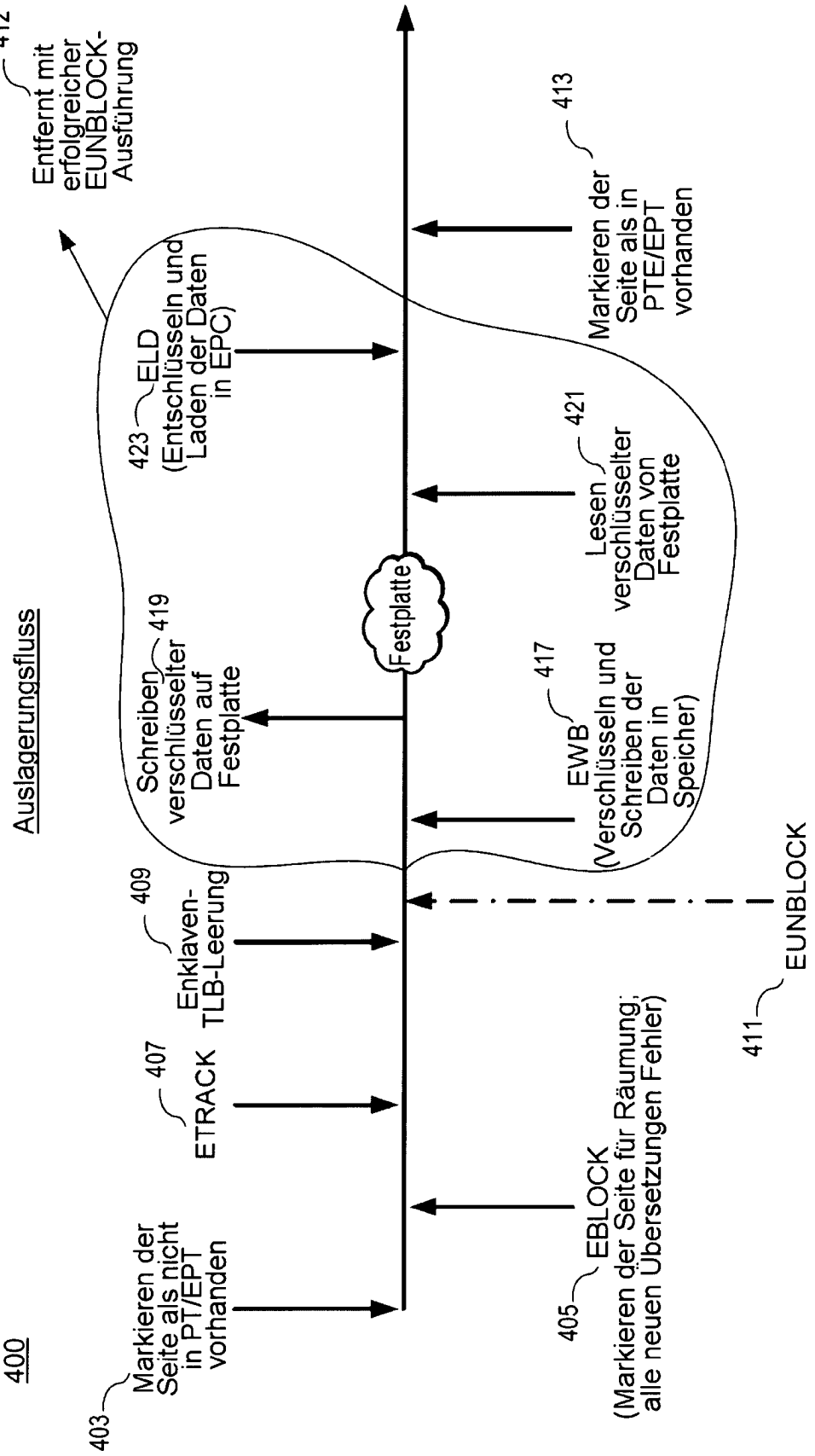

400
Auslagerungsfluss
403
Markieren der Seite als nicht in PT/EPT vorhanden
407
ETRACK
409
Enklaven-TLB-Leerung
405
EBLOCK
(Markieren der Seite für Räumung; alle neuen Übersetzungen Fehler)
411
EUNBLOCK
412
Entfernt mit erfolgreicher EUNBLOCK-Ausführung
413
Markieren der Seite als in PTE/EPT vorhanden
423
ELD
(Entschlüsseln und Laden der Daten in EPC)
421
Lesen verschlüsselter Daten von Festplatte
Festplatte
419
Schreiben verschlüsselter Daten auf Festplatte
417
EWB
(Verschlüsseln und Schreiben der Daten in Speicher)

FIG. 4

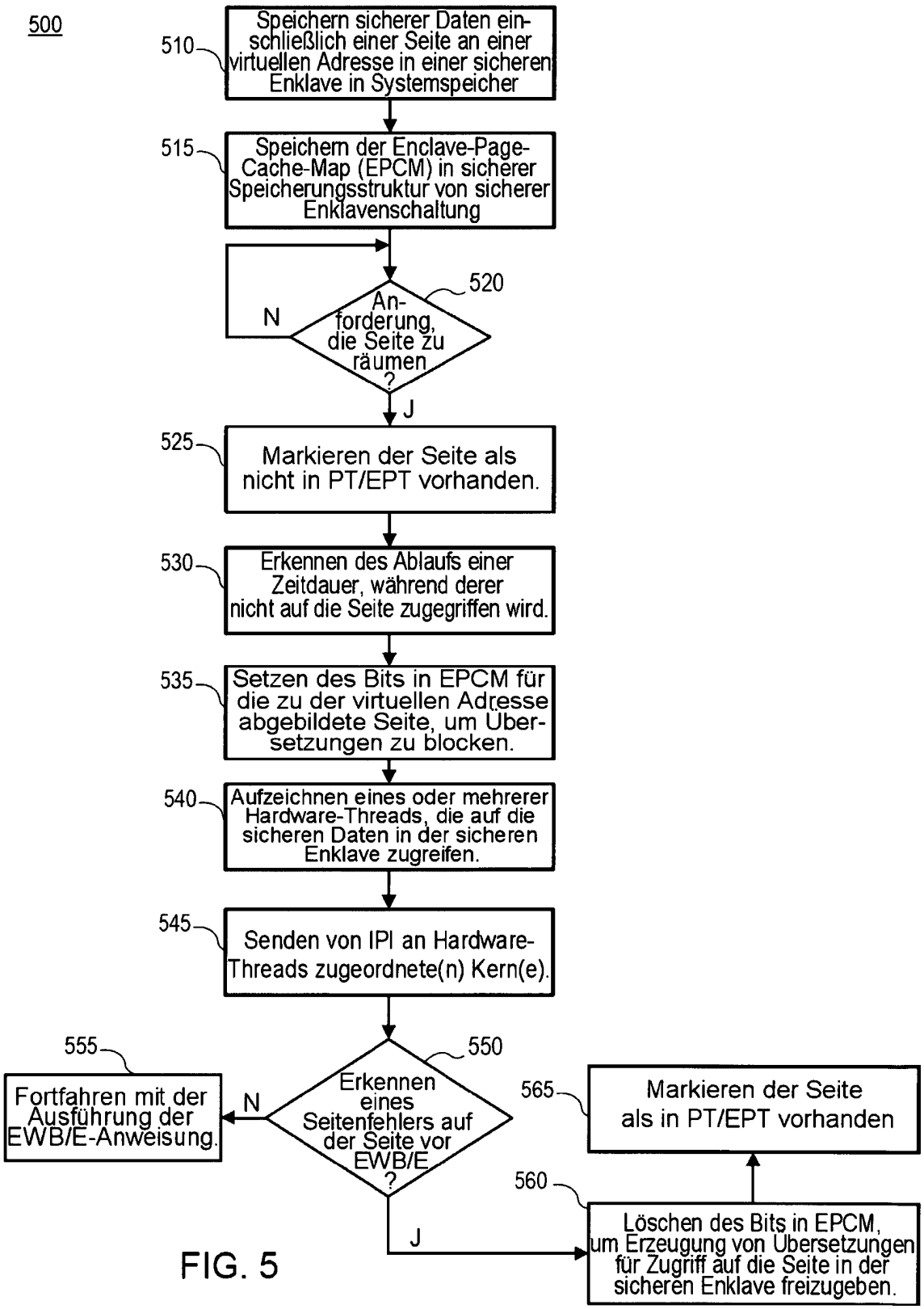

500
510
Speichern sicherer Daten einschließlich einer Seite an einer virtuellen Adresse in einer sicheren Enklave in Systemspeicher
515
Speichern der Enclave-Page-Cache-Map (EPCM) in sicherer Speicherungsstruktur von sicherer Enklavenschaltung
520
Anforderung, die Seite zu räumen ?
N
J
525
Markieren der Seite als nicht in PT/EPT vorhanden.
530
Erkennen des Ablaufs einer Zeitdauer, während derer nicht auf die Seite zugegriffen wird.
535
Setzen des Bits in EPCM für die zu der virtuellen Adresse abgebildete Seite, um Übersetzungen zu blocken.
540
Aufzeichnen eines oder mehrerer Hardware-Threads, die auf die sicheren Daten in der sicheren Enklave zugreifen.
545
Senden von IPI an Hardware-Threads zugeordnete(n) Kern(e).
550
Erkennen eines Seitenfehlers auf der Seite vor EWB/E ?
N
555
Fortfahren mit der Ausführung der EWB/E-Anweisung.
J
560
Löschen des Bits in EPCM, um Erzeugung von Übersetzungen für Zugriff auf die Seite in der sicheren Enklave freizugeben.
565
Markieren der Seite als in PT/EPT vorhanden

FIG. 5

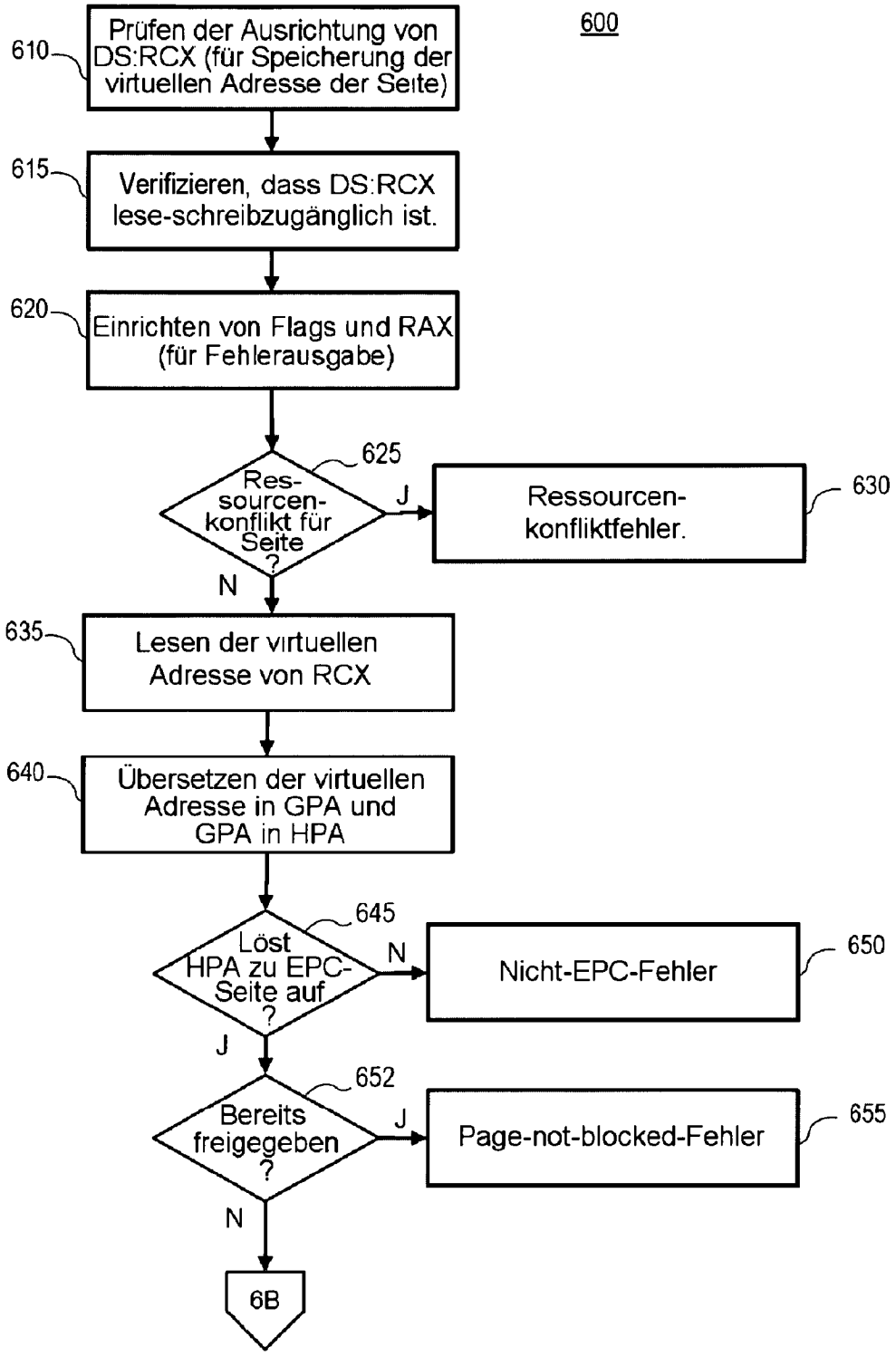

600
610
Prüfen der Ausrichtung von DS:RCX (für Speicherung der virtuellen Adresse der Seite)
615
Verifizieren, dass DS:RCX lese-schreibzugänglich ist.
620
Einrichten von Flags und RAX (für Fehlerausgabe)
625
Res-sourcen-konflikt für Seite ?
J
630
Ressourcen-konfliktfehler.
N
635
Lesen der virtuellen Adresse von RCX
640
Übersetzen der virtuellen Adresse in GPA und GPA in HPA
645
Löst HPA zu EPC-Seite auf ?
N
650
Nicht-EPC-Fehler
J
652
Bereits freigegeben ?
J
655
Page-not-blocked-Fehler
N
6B

FIG. 6A

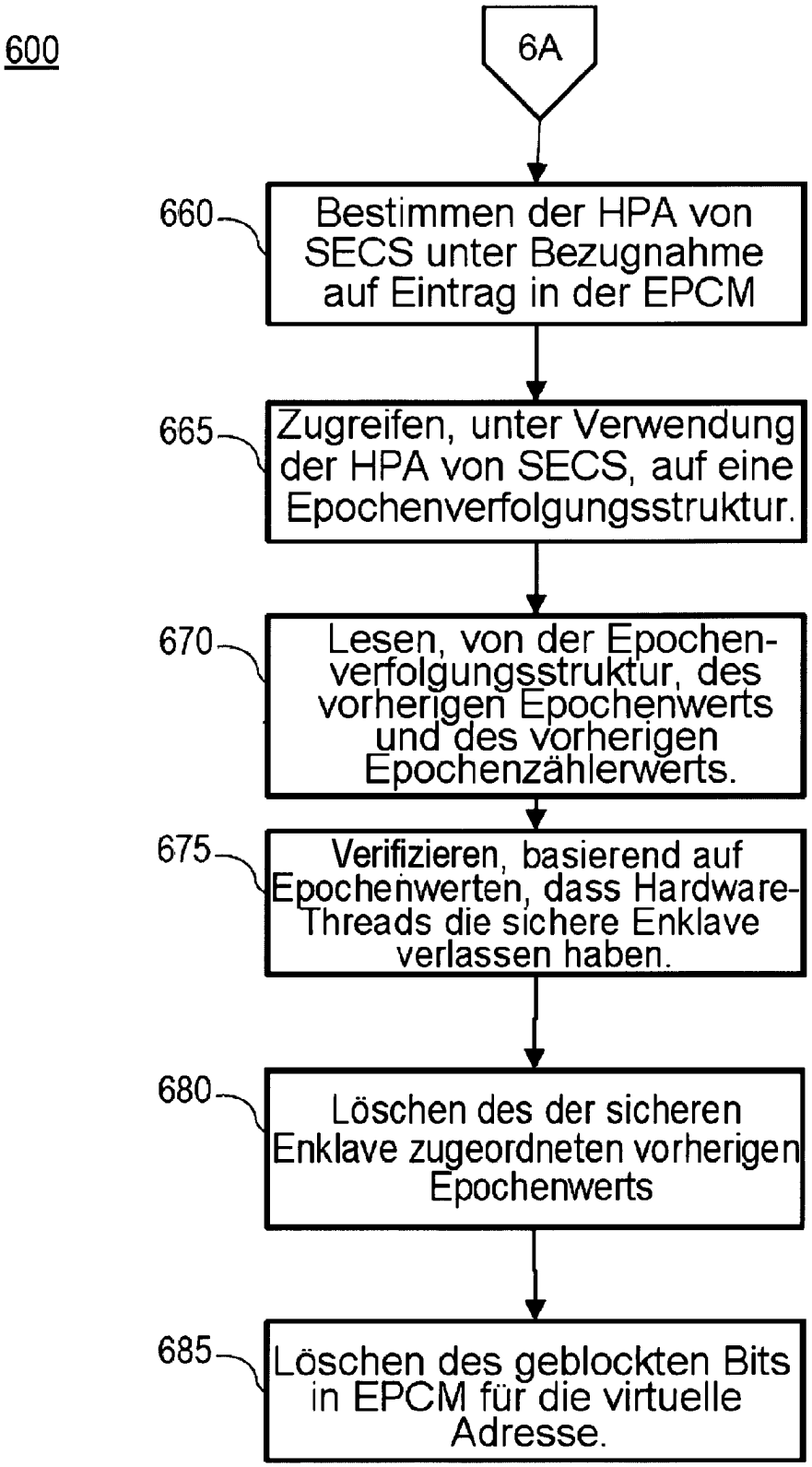

600
6A
660
Bestimmen der HPA von SECS unter Bezugnahme auf Eintrag in der EPCM
665
Zugreifen, unter Verwendung der HPA von SECS, auf eine Epochenverfolgungsstruktur.
670
Lesen, von der Epochen-verfolgungsstruktur, des vorherigen Epochenwerts und des vorherigen Epochenzählerwerts.
675
Verifizieren, basierend auf Epochenwerten, dass Hardware-Threads die sichere Enklave verlassen haben.
680
Löschen des der sicheren Enklave zugeordneten vorherigen Epochenwerts
685
Löschen des geblockten Bits in EPCM für die virtuelle Adresse.

FIG. 6B

Seiteninformationen (UMDSINFO) 700				
Versatz-Name	Versatz	Größe (Byte)	Beschreibung	Gesetzt durch:
BITVEC	0	8	Satz von Bits weitergegeben in EWBE und ELDE	Software
SRCPGE	8	8	Virtuelle Adresse der Seite, wo Seiteninhalte platziert sind.	Software
SECS	16	8	Virtuelle Adresse des EPC-Slots, der aktuell eine Kopie der SECS enthält.	Software

FIG. 7A

BITVEC 710				
Versatz-Name	Versatz	Größe (Byte)	Beschreibung	Gesetzt durch:
B	0	1	Geblocktes Bit, das durch Anweisungen verwendet werden soll.	Software
RESERVIERT	2	62	MBZ	Software

FIG. 7B

EVA 800					
Versatz-Name	Versatz	Größe (Byte)	Beschreibung	Gesetzt durch:	Sperren
Slot_0	0	64	UMDS Slot_0	EWBE/ ELDE-Anweisung	Atomare Aktualisierung, um Wert von VERSION-Feld zu schreiben
* * *					
Slot_63	4031	64	UMDS Slot_63	EWBE/ ELDE-Anweisung	

FIG. 8

UMDS 900		
Versatz-Name	Größe (Byte)	Beschreibung
VERSION	8	Einzige Version
LINADDR	6	Schützen Adressbits 56:12
UMDSFLAGS	2	Siehe Tabelle 2
UMDSLock	1	Mikroarchitektursperre
RESERVIERT	31	
MAC	16	MAC für Seite und reserviertes Feld

FIG. 9

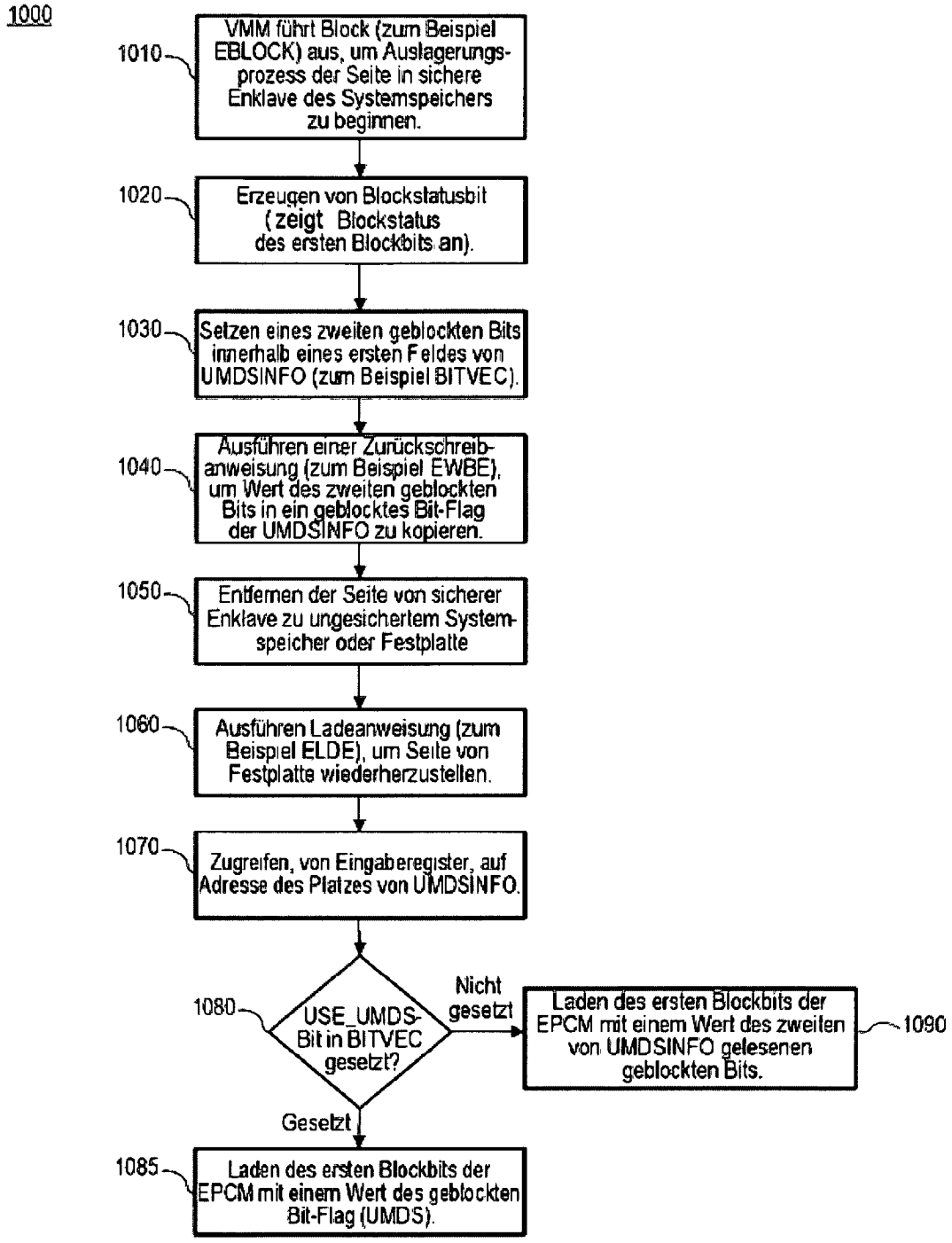

1000
1010
VMM führt Block (zum Beispiel EBLOCK) aus, um Auslagerungsprozess der Seite in sichere Enklave des Systemspeichers zu beginnen.
1020
Erzeugen von Blockstatusbit (zeigt Blockstatus des ersten Blockbits an).
1030
Setzen eines zweiten geblockten Bits innerhalb eines ersten Feldes von UMDSINFO (zum Beispiel BITVEC).
1040
Ausführen einer Zurückschreibanweisung (zum Beispiel EWBE), um Wert des zweiten geblockten Bits in ein geblocktes Bit-Flag der UMDSINFO zu kopieren.
1050
Entfernen der Seite von sicherer Enklave zu ungesichertem Systemspeicher oder Festplatte
1060
Ausführen Ladeanweisung (zum Beispiel ELDE), um Seite von Festplatte wiederherzustellen.
1070
Zugreifen, von Eingaberegister, auf Adresse des Platzes von UMDSINFO.
1080
USE_UMDS-Bit in BITVEC gesetzt?
Nicht gesetzt
Laden des ersten Blockbits der EPCM mit einem Wert des zweiten von UMDSINFO gelesenen geblockten Bits.
1090
Gesetzt
1085
Laden des ersten Blockbits der EPCM mit einem Wert des geblockten Bit-Flag (UMDS).

FIG. 10

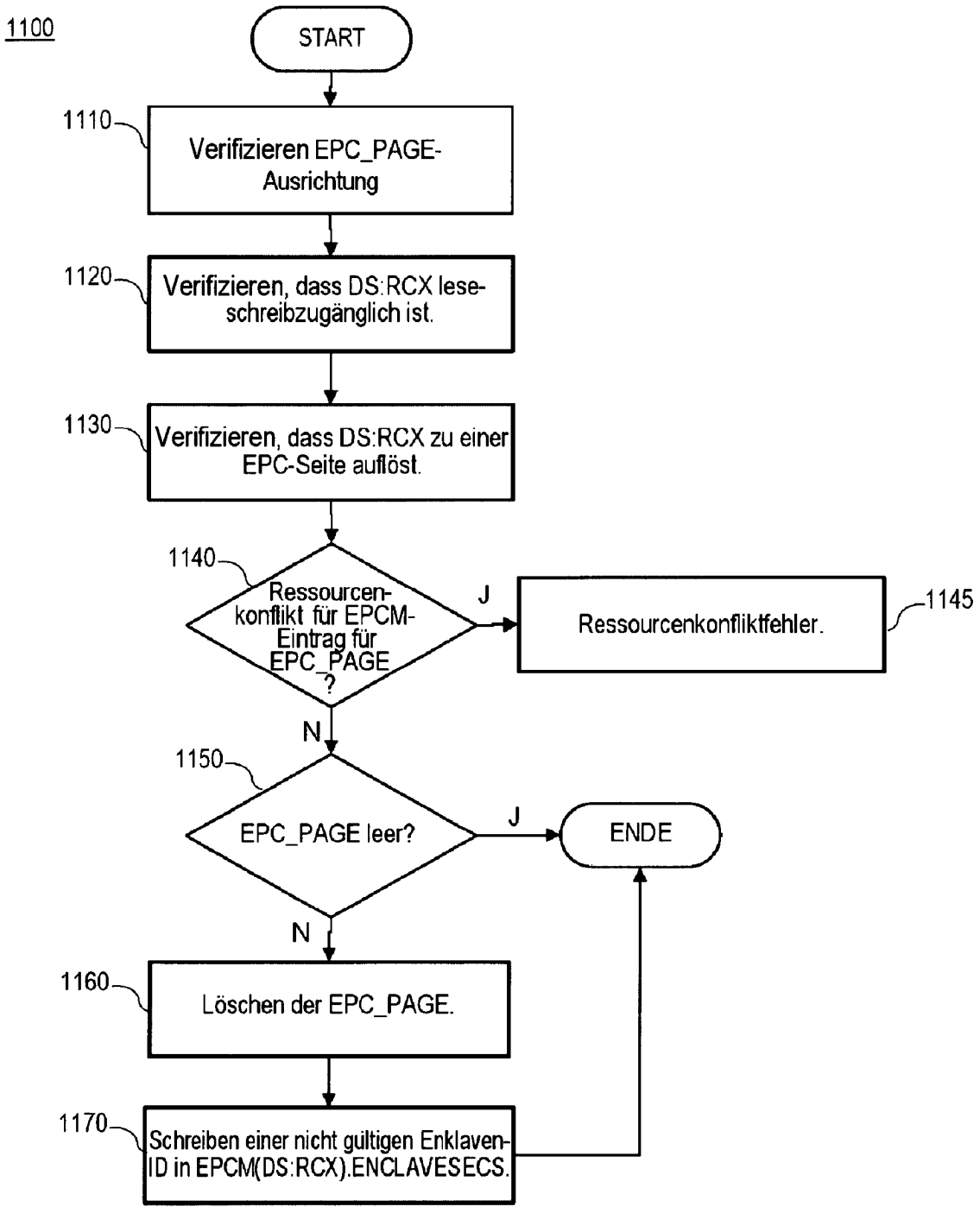

1100
START
1110
Verifizieren EPC_PAGE-Ausrichtung
1120
Verifizieren, dass DS:RCX lese-schreibzugänglich ist.
1130
Verifizieren, dass DS:RCX zu einer EPC-Seite auflöst.
1140
Ressourcen-konflikt für EPCM-Eintrag für EPC_PAGE ?
J
Ressourcenkonfliktfehler.
1145
N
1150
EPC_PAGE leer?
J
ENDE
N
1160
Löschen der EPC_PAGE.
1170
Schreiben einer nicht gültigen Enklaven-ID in EPCM(DS:RCX).ENCLAVESECS.

FIG. 11

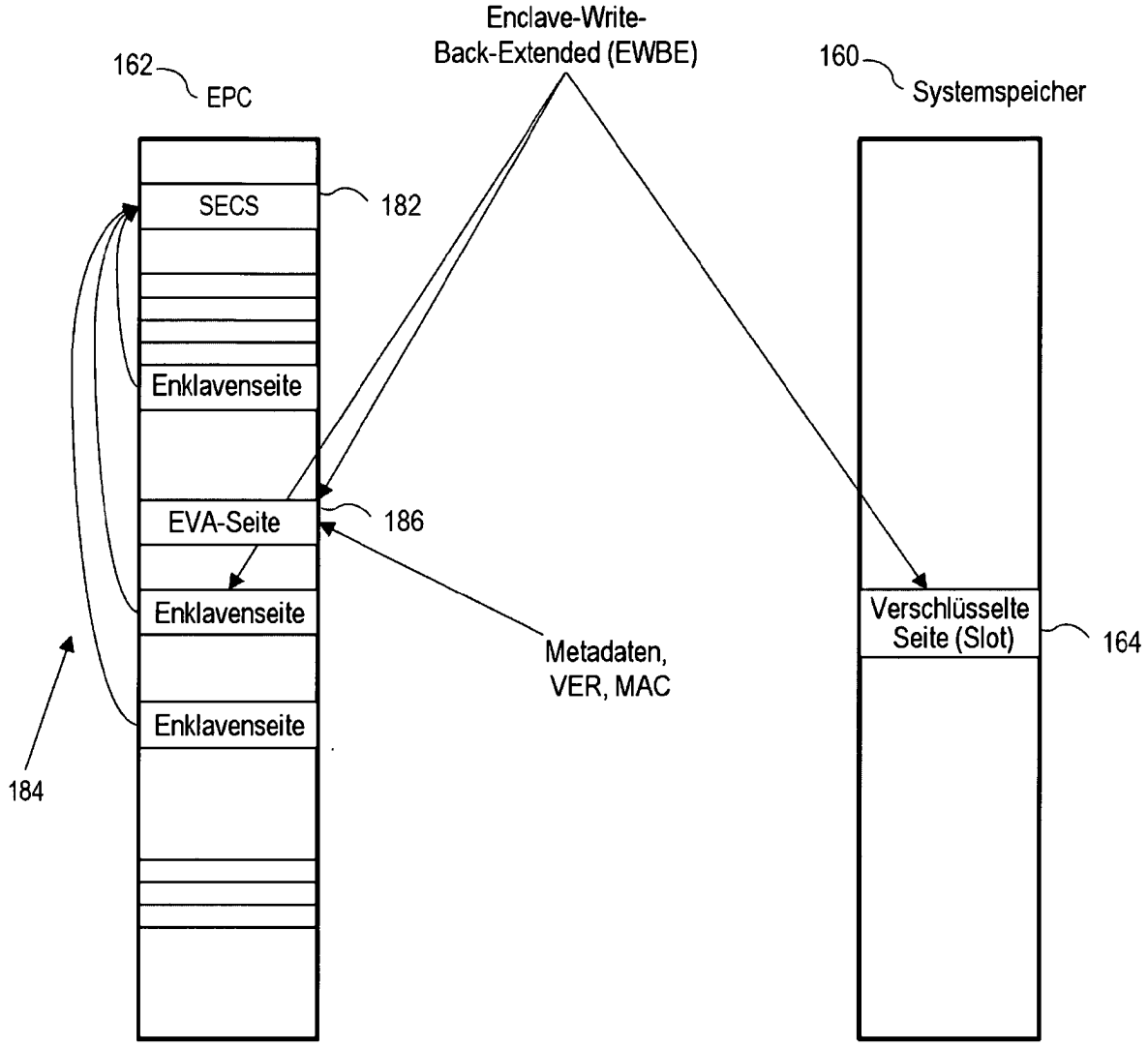

Enclave-Write-
Back-Extended (EWBE)
162
EPC
160
Systemspeicher
SECS
182
Enklavenseite
EVA-Seite
186
Enklavenseite
Metadaten,
VER, MAC
Verschlüsselte
Seite (Slot)
164
Enklavenseite
184

FIG. 12A

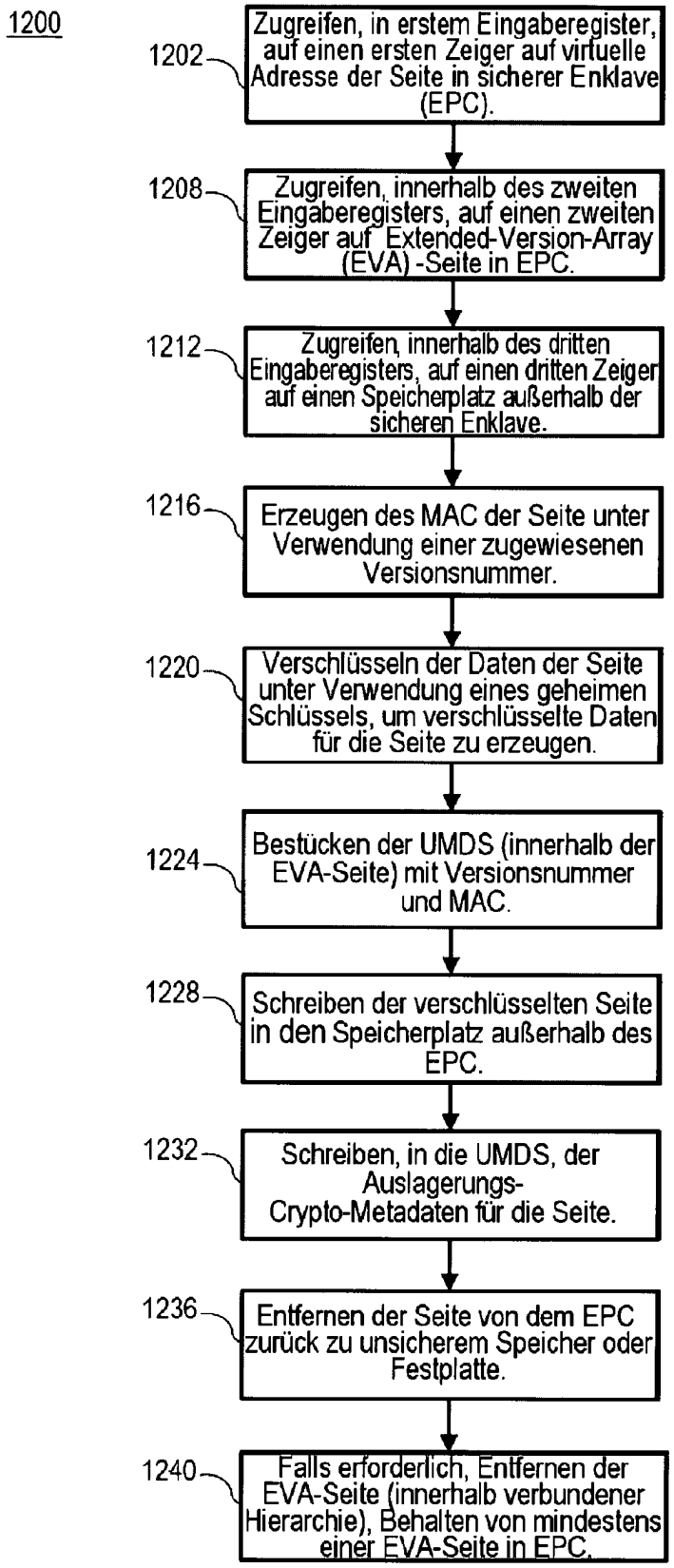

1200
1202
Zugreifen, in erstem Eingaberegister, auf einen ersten Zeiger auf virtuelle Adresse der Seite in sicherer Enklave (EPC).
1208
Zugreifen, innerhalb des zweiten Eingaberegisters, auf einen zweiten Zeiger auf Extended-Version-Array (EVA) -Seite in EPC.
1212
Zugreifen, innerhalb des dritten Eingaberegisters, auf einen dritten Zeiger auf einen Speicherplatz außerhalb der sicheren Enklave.
1216
Erzeugen des MAC der Seite unter Verwendung einer zugewiesenen Versionsnummer.
1220
Verschlüsseln der Daten der Seite unter Verwendung eines geheimen Schlüssels, um verschlüsselte Daten für die Seite zu erzeugen.
1224
Bestücken der UMDS (innerhalb der EVA-Seite) mit Versionsnummer und MAC.
1228
Schreiben der verschlüsselten Seite in den Speicherplatz außerhalb des EPC.
1232
Schreiben, in die UMDS, der Auslagerungs-Crypto-Metadaten für die Seite.
1236
Entfernen der Seite von dem EPC zurück zu unsicherem Speicher oder Festplatte.
1240
Falls erforderlich, Entfernen der EVA-Seite (innerhalb verbundener Hierarchie), Behalten von mindestens einer EVA-Seite in EPC.

FIG. 12B

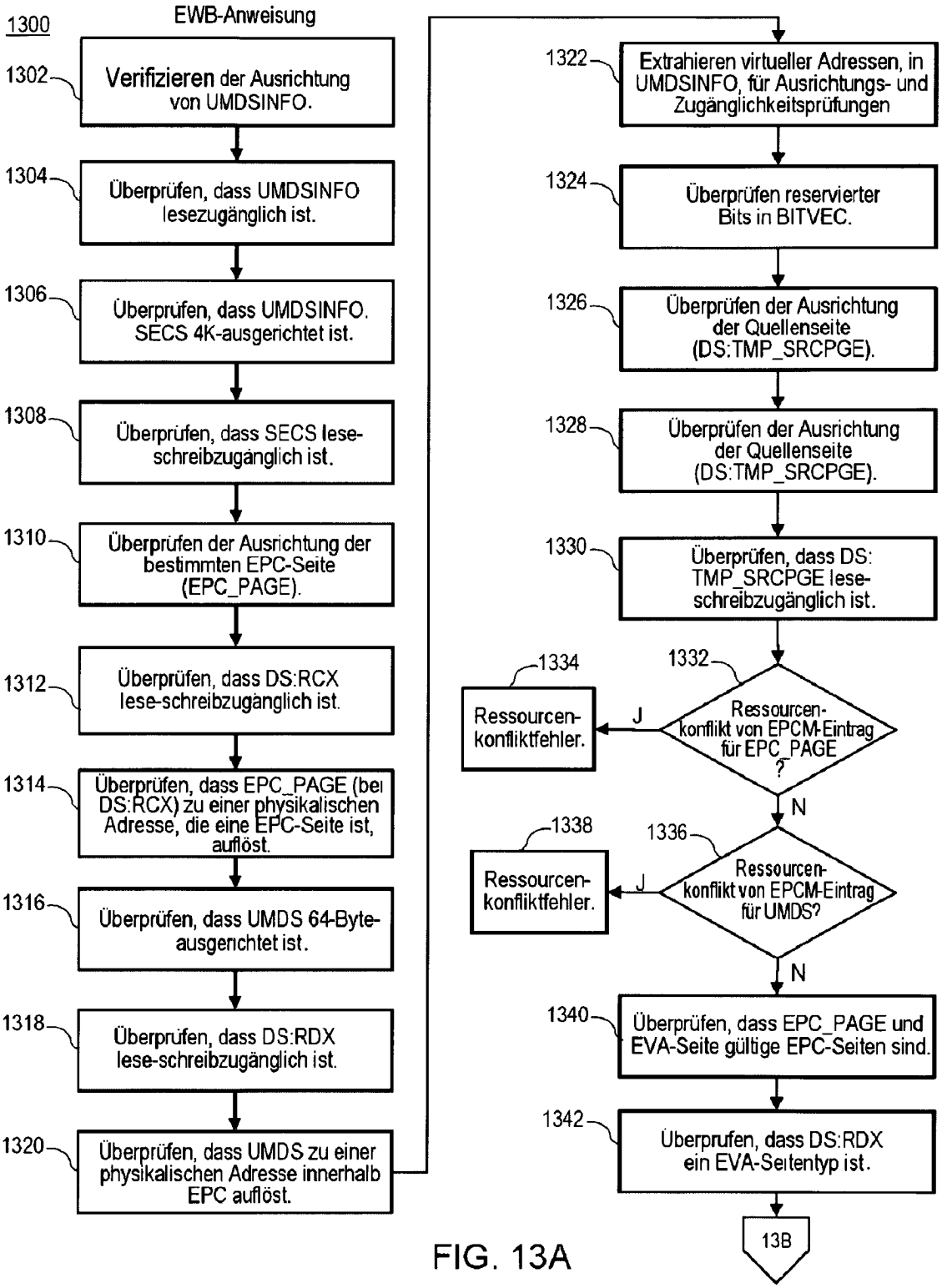

1300
EWB-Anweisung
1302
Verifizieren der Ausrichtung von UMDSINFO.
1304
Überprüfen, dass UMDSINFO lesezugänglich ist.
1306
Überprüfen, dass UMDSINFO. SECS 4K-ausgerichtet ist.
1308
Überprüfen, dass SECS lese-schreibzugänglich ist.
1310
Überprüfen der Ausrichtung der bestimmten EPC-Seite (EPC_PAGE).
1312
Überprüfen, dass DS:RCX lese-schreibzugänglich ist.
1314
Überprüfen, dass EPC_PAGE (bei DS:RCX) zu einer physikalischen Adresse, die eine EPC-Seite ist, auflöst.
1316
Überprüfen, dass UMDS 64-Byte-ausgerichtet ist.
1318
Überprüfen, dass DS:RDX lese-schreibzugänglich ist.
1320
Überprüfen, dass UMDS zu einer physikalischen Adresse innerhalb EPC auflöst.
1322
Extrahieren virtueller Adressen, in UMDSINFO, für Ausrichtungs- und Zugänglichkeitsprüfungen
1324
Überprüfen reservierter Bits in BITVEC.
1326
Überprüfen der Ausrichtung der Quellenseite (DS:TMP_SRCPGE).
1328
Überprüfen der Ausrichtung der Quellenseite (DS:TMP_SRCPGE).
1330
Überprüfen, dass DS: TMP_SRCPGE lese-schreibzugänglich ist.
1332
Ressourcen-konflikt von EPCM-Eintrag für EPC_PAGE ?
J
1334
Ressourcen-konfliktfehler.
N
1336
Ressourcen-konflikt von EPCM-Eintrag für UMDS?
J
1338
Ressourcen-konfliktfehler.
N
1340
Überprüfen, dass EPC_PAGE und EVA-Seite gültige EPC-Seiten sind.
1342
Überprüfen, dass DS:RDX ein EVA-Seitentyp ist.
13B

FIG. 13A

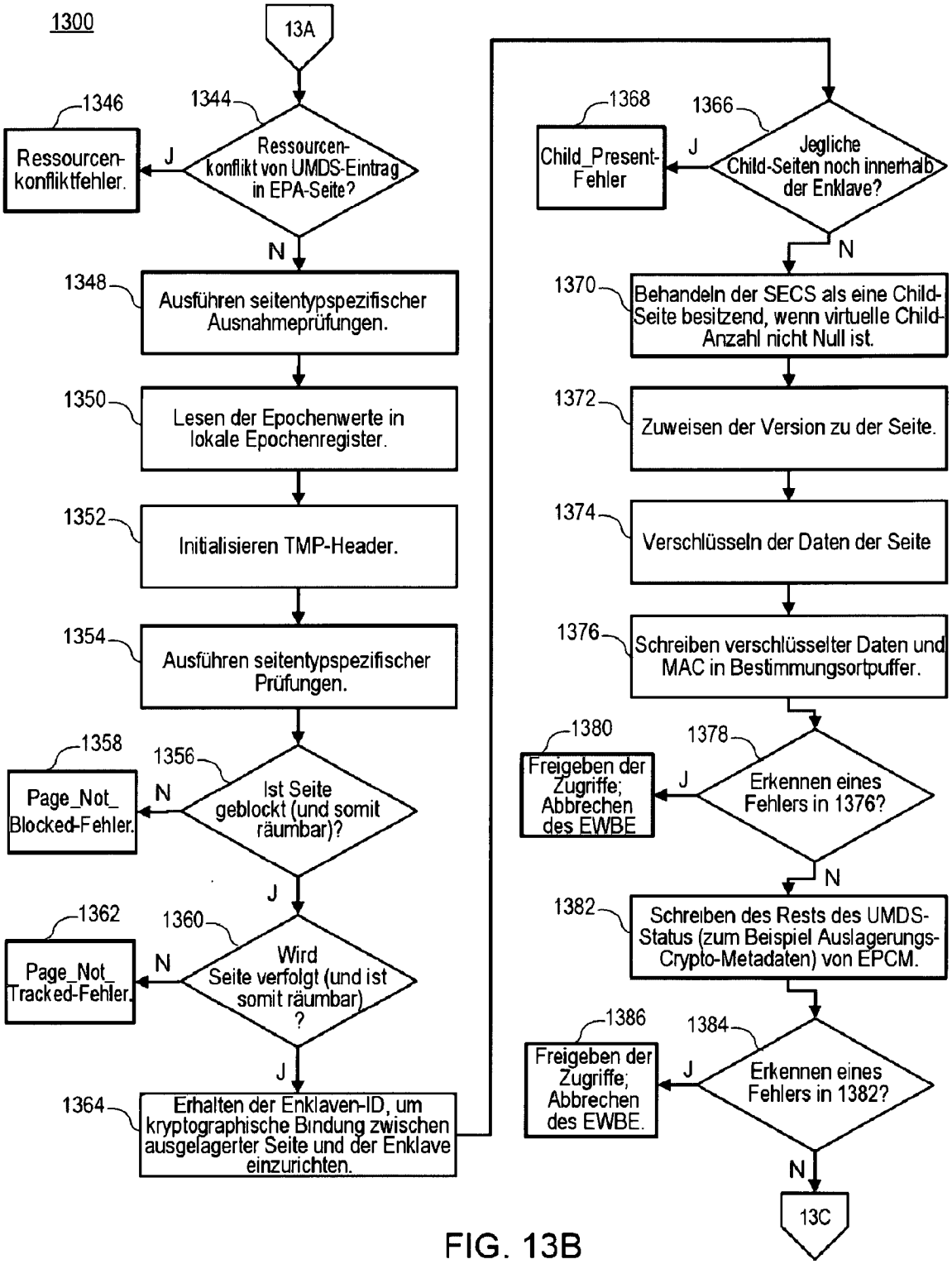

1300
13A
1344
Ressourcen-konflikt von UMDS-Eintrag in EPA-Seite?
J
1346
Ressourcen-konfliktfehler.
N
1348
Ausführen seitentypspezifischer Ausnahmeprüfungen.
1350
Lesen der Epochenwerte in lokale Epochenregister.
1352
Initialisieren TMP-Header.
1354
Ausführen seitentypspezifischer Prüfungen.
1356
Ist Seite geblockt (und somit räumbar)?
N
1358
Page_Not_Blocked-Fehler.
J
1360
Wird Seite verfolgt (und ist somit räumbar) ?
N
1362
Page_Not_Tracked-Fehler.
J
1364
Erhalten der Enklaven-ID, um kryptographische Bindung zwischen ausgelagerter Seite und der Enklave einzurichten.
1366
Jegliche Child-Seiten noch innerhalb der Enklave?
J
1368
Child_Present-Fehler
N
1370
Behandeln der SECS als eine Child-Seite besitzend, wenn virtuelle Child-Anzahl nicht Null ist.
1372
Zuweisen der Version zu der Seite.
1374
Verschlüsseln der Daten der Seite
1376
Schreiben verschlüsselter Daten und MAC in Bestimmungsortpuffer.
1378
Erkennen eines Fehlers in 1376?
J
1380
Freigeben der Zugriffe; Abbrechen des EWBE
N
1382
Schreiben des Rests des UMDS-Status (zum Beispiel Auslagerungs-Crypto-Metadaten) von EPCM.
1384
Erkennen eines Fehlers in 1382?
J
1386
Freigeben der Zugriffe; Abbrechen des EWBE.
N
13C

FIG. 13B

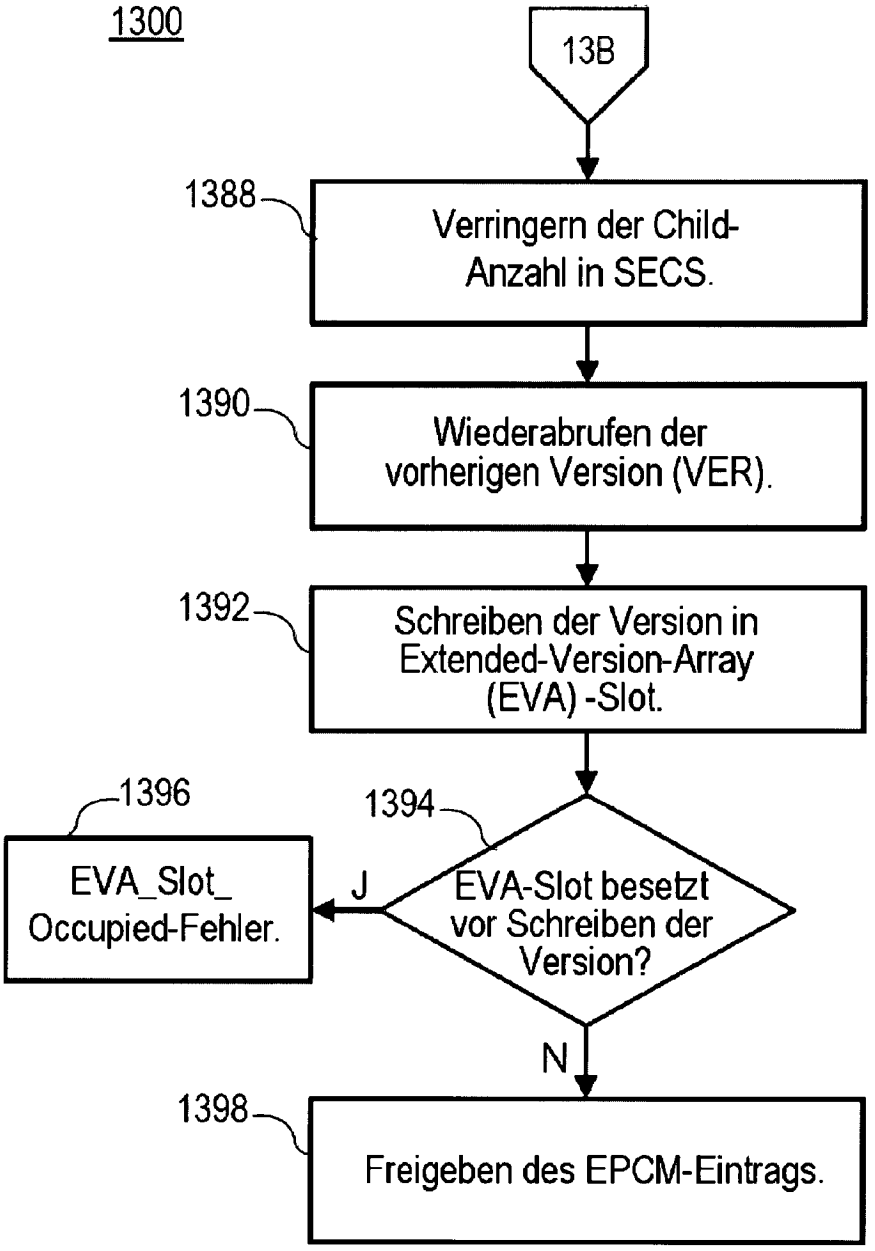

1300
13B
1388
Verringern der Child-Anzahl in SECS.
1390
Wiederabrufen der vorherigen Version (VER).
1392
Schreiben der Version in Extended-Version-Array (EVA) -Slot.
1394
EVA-Slot besetzt vor Schreiben der Version?
J
1396
EVA_Slot_ Occupied-Fehler.
N
1398
Freigeben des EPCM-Eintrags.

FIG. 13C

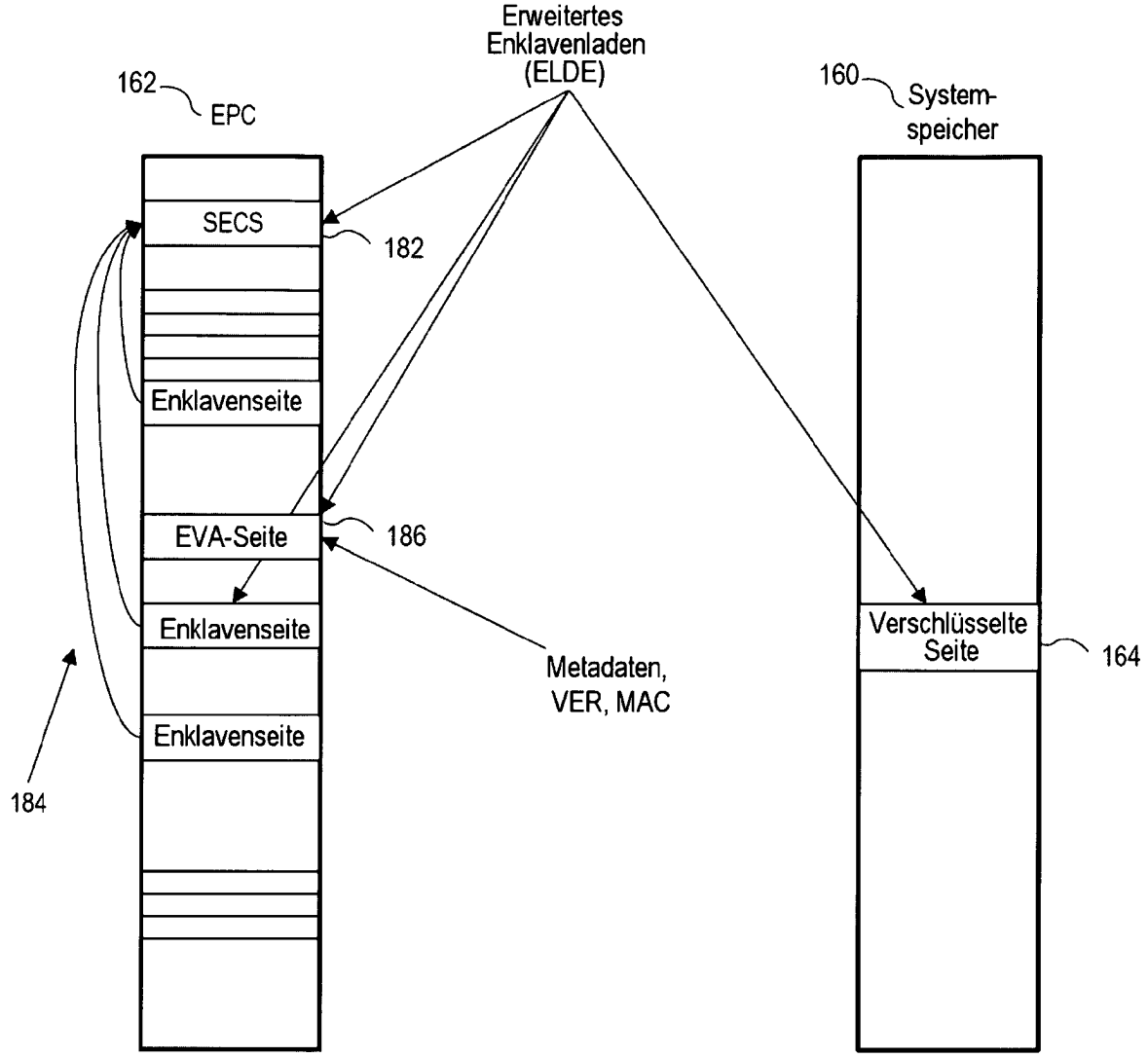

Erweitertes
Enklavenladen
(ELDE)
162
EPC
160
System-
speicher
SECS
182
Enklavenseite
EVA-Seite
186
Enklavenseite
Enklavenseite
Metadaten,
VER, MAC
Verschlüsselte
Seite
164
184

FIG. 14A

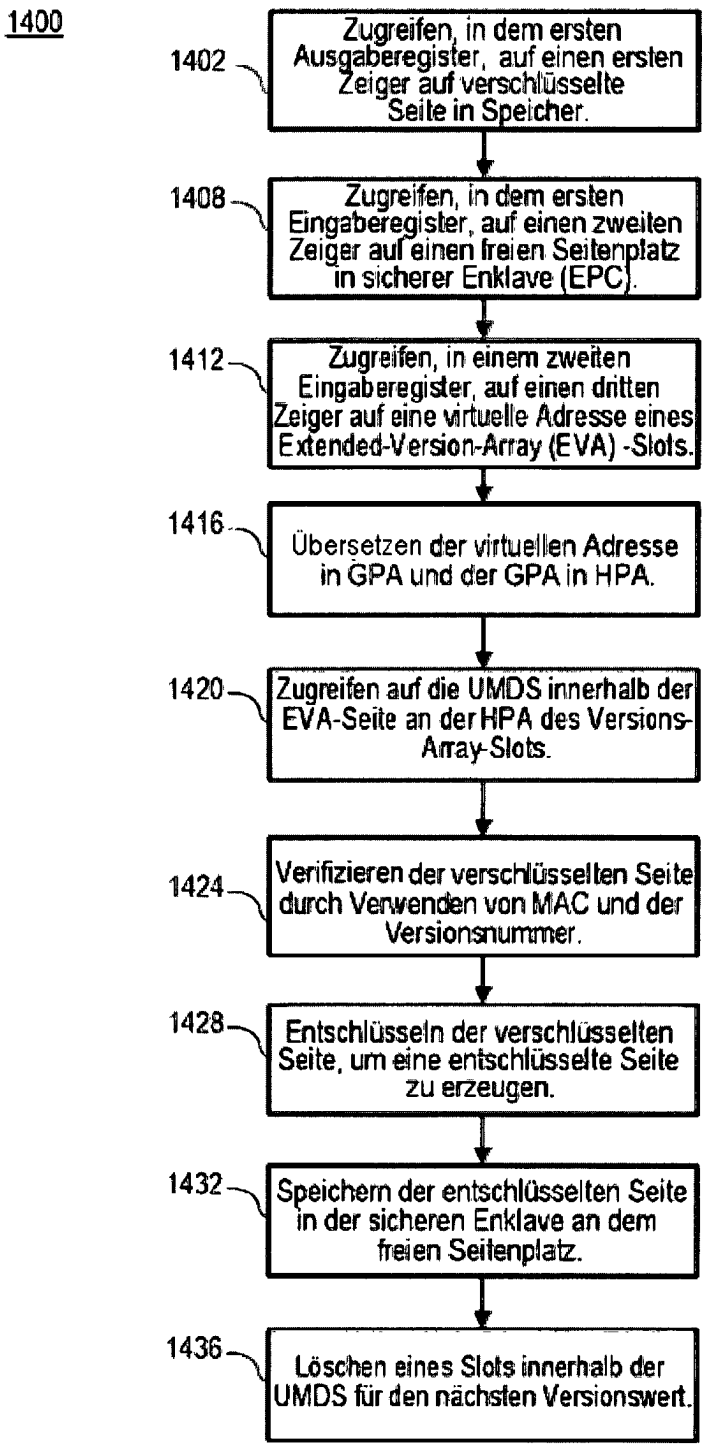

1400
1402
Zugreifen, in dem ersten Ausgaberegister, auf einen ersten Zeiger auf verschlüsselte Seite in Speicher.
1408
Zugreifen, in dem ersten Eingaberegister, auf einen zweiten Zeiger auf einen freien Seitenplatz in sicherer Enklave (EPC).
1412
Zugreifen, in einem zweiten Eingaberegister, auf einen dritten Zeiger auf eine virtuelle Adresse eines Extended-Version-Array (EVA) -Slots.
1416
Übersetzen der virtuellen Adresse in GPA und der GPA in HPA.
1420
Zugreifen auf die UMDS innerhalb der EVA-Seite an der HPA des Versions-Array-Slots.
1424
Verifizieren der verschlüsselten Seite durch Verwenden von MAC und der Versionsnummer.
1428
Entschlüsseln der verschlüsselten Seite, um eine entschlüsselte Seite zu erzeugen.
1432
Speichern der entschlüsselten Seite in der sicheren Enklave an dem freien Seitenplatz.
1436
Löschen eines Slots innerhalb der UMDS für den nächsten Versionswert.

FIG. 14B

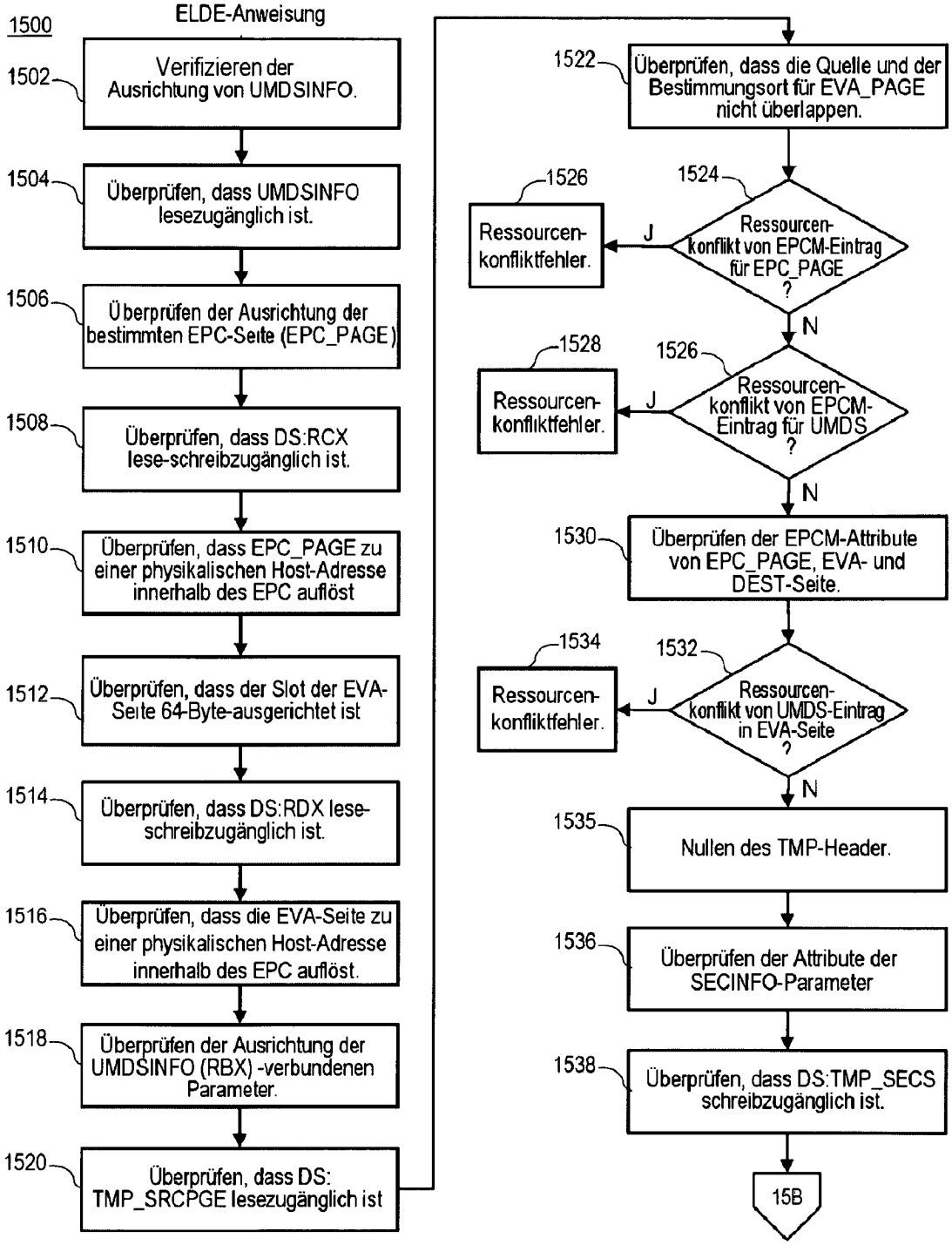

1500
ELDE-Anweisung
1502
Verifizieren der Ausrichtung von UMDSINFO.
1504
Überprüfen, dass UMDSINFO lesezugänglich ist.
1506
Überprüfen der Ausrichtung der bestimmten EPC-Seite (EPC_PAGE)
1508
Überprüfen, dass DS:RCX lese-schreibzugänglich ist.
1510
Überprüfen, dass EPC_PAGE zu einer physikalischen Host-Adresse innerhalb des EPC auflöst
1512
Überprüfen, dass der Slot der EVA-Seite 64-Byte-ausgerichtet ist
1514
Überprüfen, dass DS:RDX lese-schreibzugänglich ist.
1516
Überprüfen, dass die EVA-Seite zu einer physikalischen Host-Adresse innerhalb des EPC auflöst.
1518
Überprüfen der Ausrichtung der UMDSINFO (RBX) -verbundenen Parameter.
1520
Überprüfen, dass DS: TMP_SRCPGE lesezugänglich ist
1522
Überprüfen, dass die Quelle und der Bestimmungsort für EVA_PAGE nicht überlappen.
1524
Ressourcen-konflikt von EPCM-Eintrag für EPC_PAGE ?
J
1526
Ressourcen-konfliktfehler.
N
1526
Ressourcen-konflikt von EPCM-Eintrag für UMDS ?
J
1528
Ressourcen-konfliktfehler.
N
1530
Überprüfen der EPCM-Attribute von EPC_PAGE, EVA- und DEST-Seite.
1532
Ressourcen-konflikt von UMDS-Eintrag in EVA-Seite ?
J
1534
Ressourcen-konfliktfehler.
N
1535
Nullen des TMP-Header.
1536
Überprüfen der Attribute der SECINFO-Parameter
1538
Überprüfen, dass DS:TMP_SECS schreibzugänglich ist.
15B

FIG. 15A

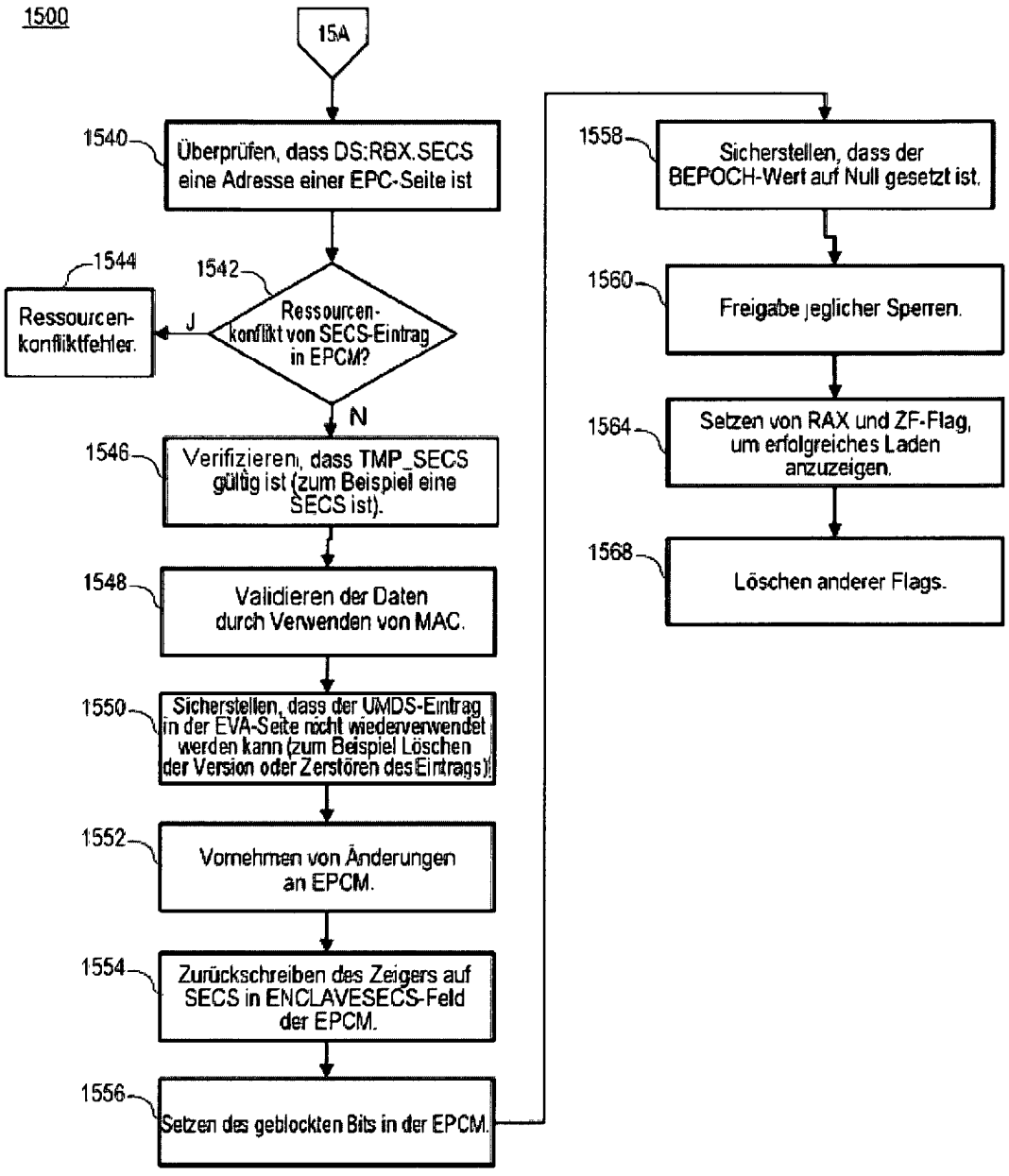

1500
15A
1540
Überprüfen, dass DS:RBX.SECS eine Adresse einer EPC-Seite ist
1542
Ressourcen-konflikt von SECS-Eintrag in EPCM?
J
1544
Ressourcen-konfliktfehler.
N
1546
Verifizieren, dass TMP_SECS gültig ist (zum Beispiel eine SECS ist).
1548
Validieren der Daten durch Verwenden von MAC.
1550
Sicherstellen, dass der UMDS-Eintrag in der EVA-Seite nicht wiederverwendet werden kann (zum Beispiel Löschen der Version oder Zerstören des Eintrags).
1552
Vornehmen von Änderungen an EPCM.
1554
Zurückschreiben des Zeigers auf SECS in ENCLAVESECS-Feld der EPCM.
1556
Setzen des geblockten Bits in der EPCM.
1558
Sicherstellen, dass der BEPOCH-Wert auf Null gesetzt ist.
1560
Freigabe jeglicher Sperren.
1564
Setzen von RAX und ZF-Flag, um erfolgreiches Laden anzuzeigen.
1568
Löschen anderer Flags.

FIG. 15B

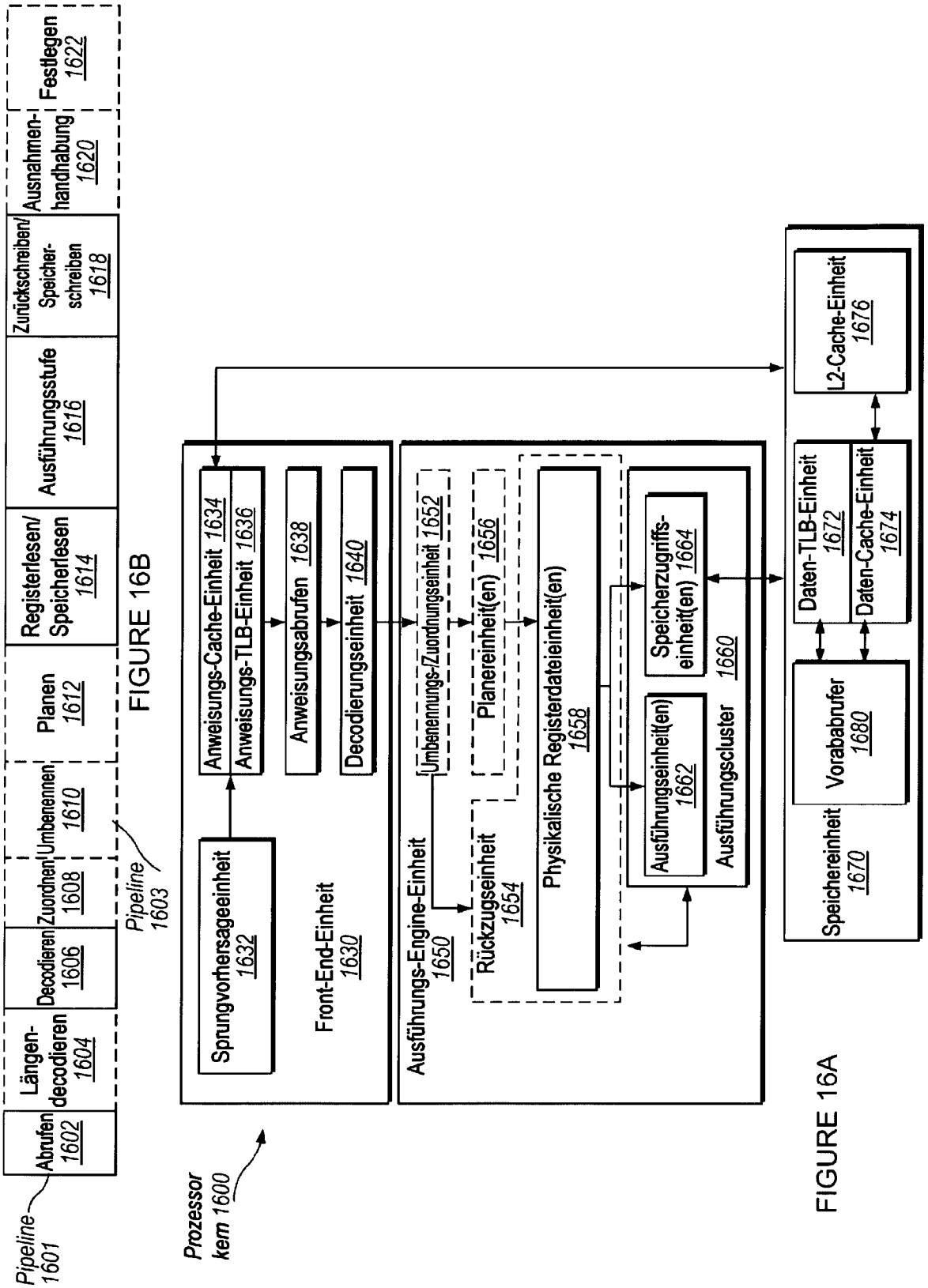


FIGURE 16A

FIGURE 16B

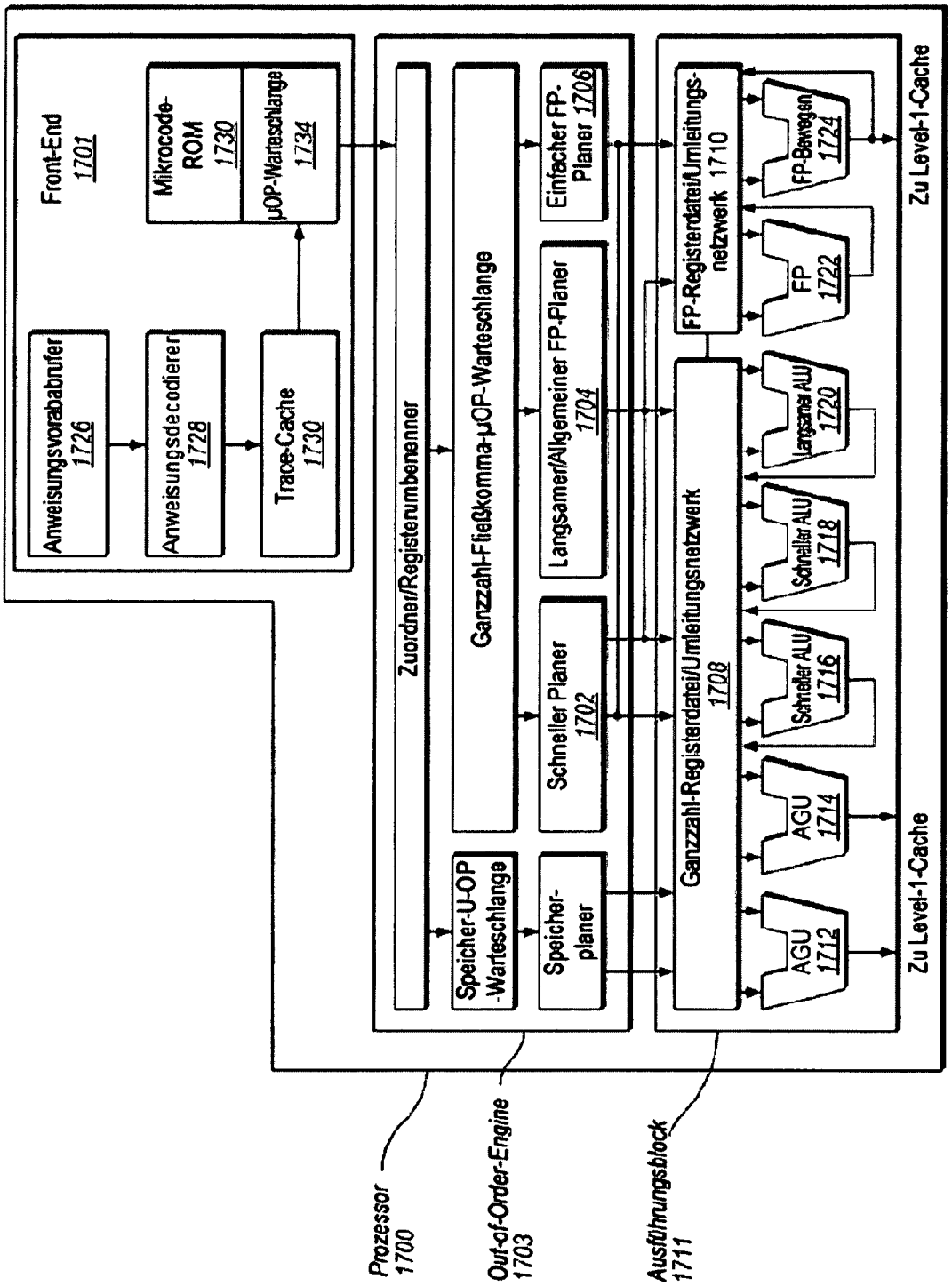

Front-End 1701
Anweisungsvorabrufer 1726
Anweisungsdecodierer 1728
Trace-Cache 1730
Mikrocode-ROM 1730
µOP-Warteschlange 1734
Zuordner/Registerumbenenner
Ganzzahl-Fließkomma-µOP-Warteschlange
Einfacher FP-Planer 1706
Schneller Planer 1702
Langsamer/Allgemeiner FP-Planer 1704
Speicher-U-OP-Warteschlange
Speicher-planer
Ganzzahl-Registerdatei/Umleitungsnetzwerk 1708
FP-Registerdatei/Umleitungsnetzwerk 1710
AGU 1712
AGU 1714
Schneller ALU 1716
Schneller ALU 1718
Langsame ALU 1720
FP 1722
FP-Bewegen 1724
Zu Level-1-Cache
Zu Level-1-Cache
Prozessor 1700
Out-of-Order-Engine 1703
Ausführungsblock 1711

FIG. 17

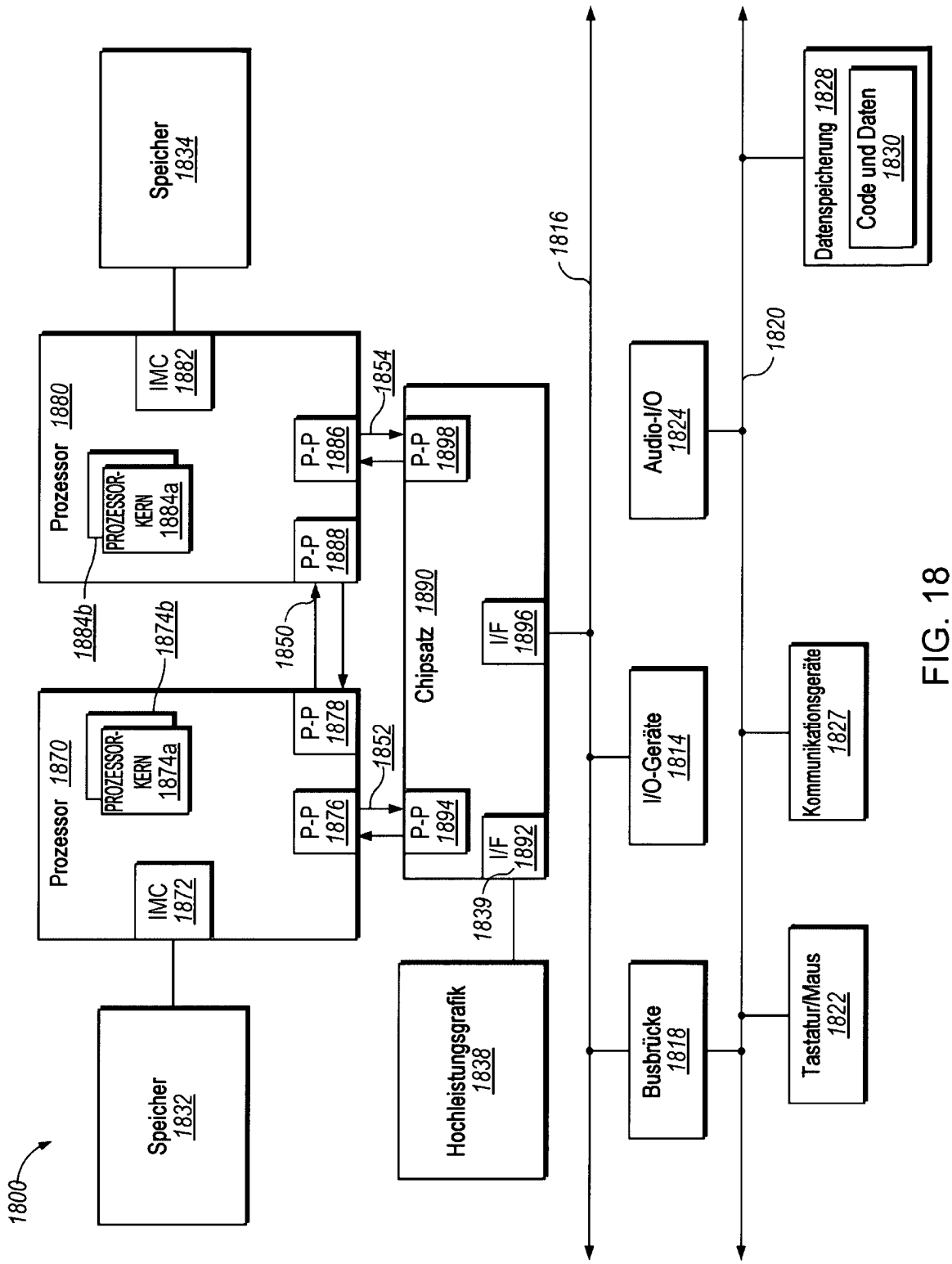

1800
Speicher 1832
Prozessor 1870
IMC 1872
PROZESSOR-KERN 1874a
1874b
P-P 1876
P-P 1878
1852
1850
1884b
Prozessor 1880
IMC 1882
PROZESSOR-KERN 1884a
P-P 1886
P-P 1888
1854
Speicher 1834
Chipsatz 1890
P-P 1894
P-P 1898
I/F 1892
I/F 1896
1839
Hochleistungsgrafik 1838
1816
Busbrücke 1818
I/O-Geräte 1814
Audio-I/O 1824
1820
Tastatur/Maus 1822
Kommunikationsgeräte 1827
Datenspeicherung 1828
Code und Daten 1830

FIG. 18

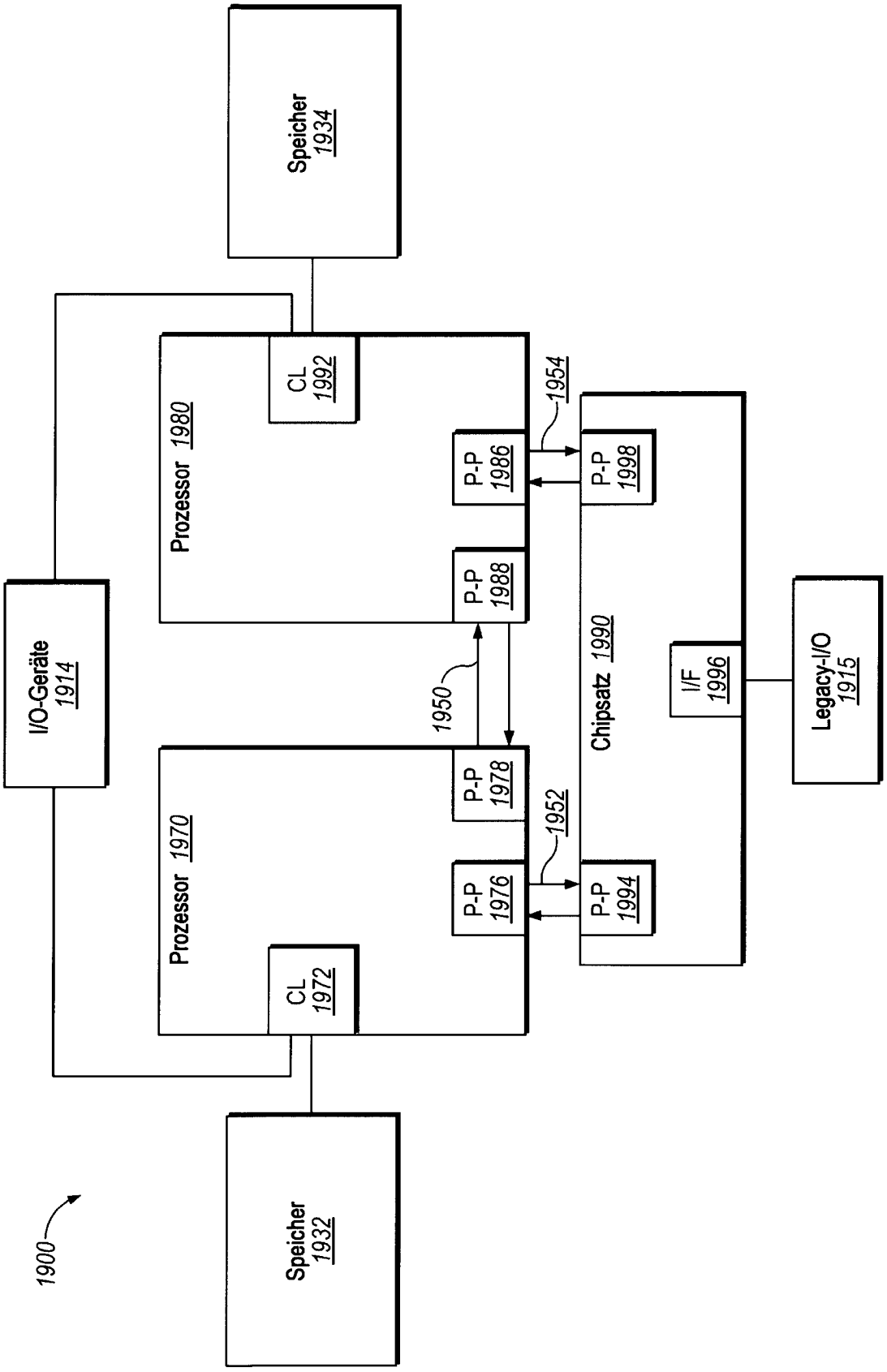

Speicher 1934
CL 1992
Prozessor 1980
P-P 1986
P-P 1988
1954
P-P 1998
I/O-Geräte 1914
1950
Chipsatz 1990
I/F 1996
Legacy-I/O 1915
P-P 1978
1952
P-P 1976
P-P 1994
Prozessor 1970
CL 1972
Speicher 1932
1900

FIG. 19

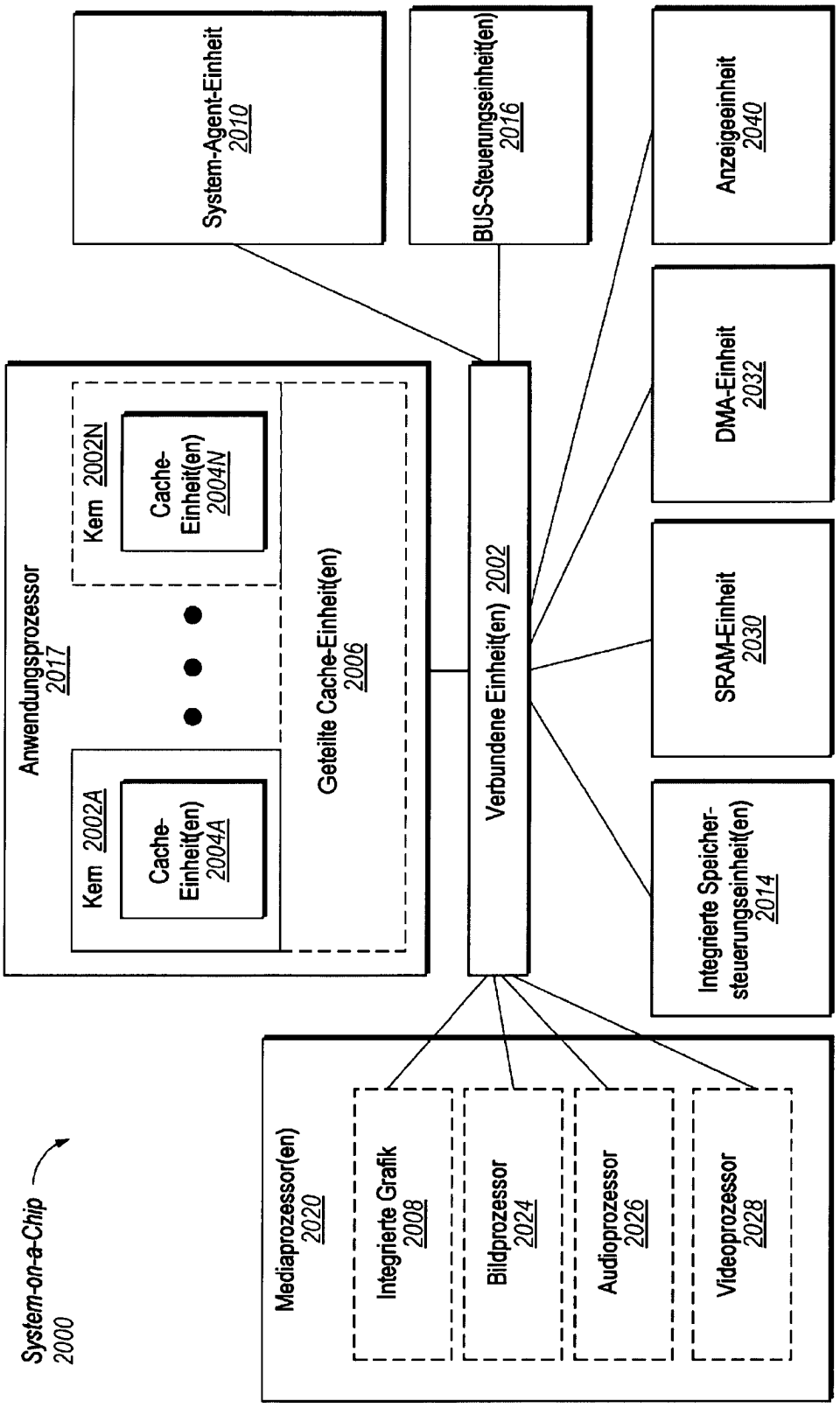

System-on-a-Chip
2000
Anwendungsprozessor
2017
Kern 2002A
Cache-Einheit(en)
2004A
Kern 2002N
Cache-Einheit(en)
2004N
Geteilte Cache-Einheit(en)
2006
System-Agent-Einheit
2010
BUS-Steuerungseinheit(en)
2016
Verbundene Einheit(en) 2002
Anzeigeeinheit
2040
DMA-Einheit
2032
SRAM-Einheit
2030
Integrierte Speicher-steuerungseinheit(en)
2014
Mediaprozessor(en)
2020
Integrierte Grafik
2008
Bildprozessor
2024
Audioprozessor
2026
Videoprozessor
2028

FIG. 20

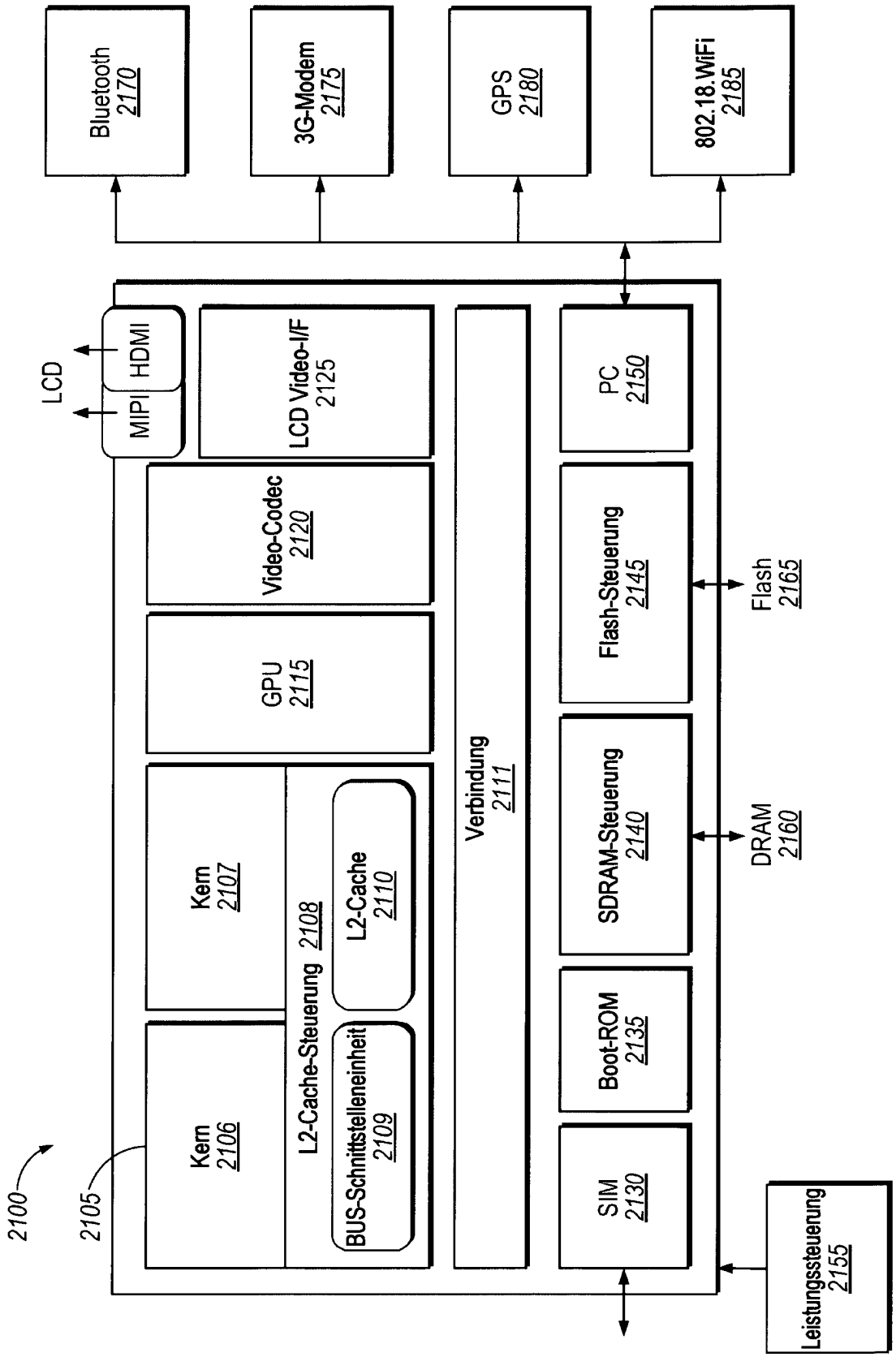

2100
2105
Kern 2106
L2-Cache-Steuerung 2108
BUS-Schnittstelleneinheit 2109
Kern 2107
L2-Cache 2110
GPU 2115
Video-Codec 2120
LCD Video-I/F 2125
MIPI
HDMI
LCD
Verbindung 2111
SIM 2130
Boot-ROM 2135
SDRAM-Steuerung 2140
Flash-Steuerung 2145
PC 2150
Leistungssteuerung 2155
DRAM 2160
Flash 2165
Bluetooth 2170
3G-Modem 2175
GPS 2180
802.18.WiFi 2185

FIG. 21

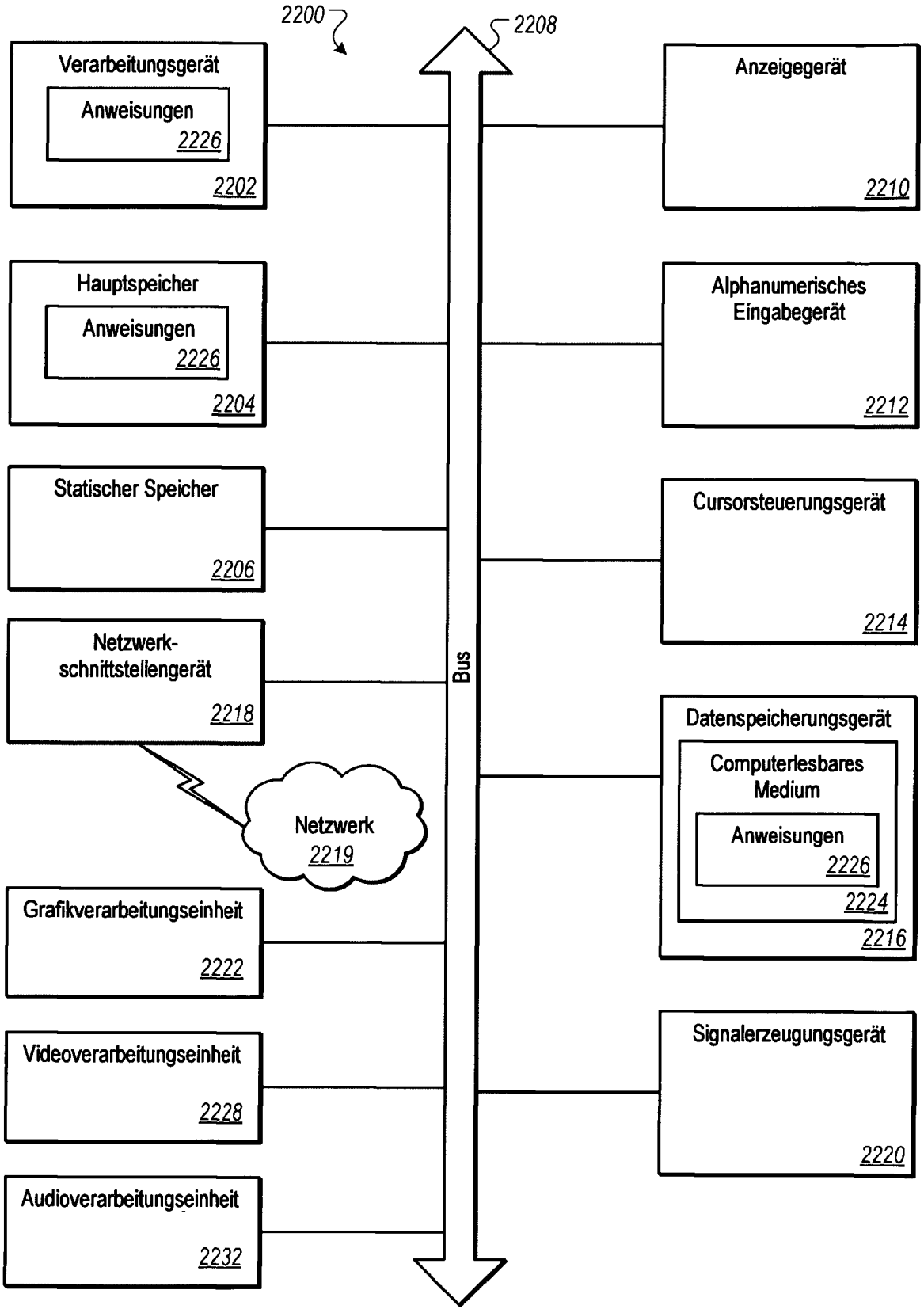

2200
2208
Verarbeitungsgerät
Anweisungen
2226
2202
Hauptspeicher
Anweisungen
2226
2204
Statischer Speicher
2206
Netzwerk-
schnittstellengerät
2218
Netzwerk
2219
Grafikverarbeitungseinheit
2222
Videoverarbeitungseinheit
2228
Audioverarbeitungseinheit
2232
Bus
Anzeigegerät
2210
Alphanumerisches
Eingabegerät
2212
Cursorsteuerungsgerät
2214
Datenspeicherungsgerät
Computerlesbares
Medium
Anweisungen
2226
2224
2216
Signalerzeugungsgerät
2220

FIG. 22