



(10) **DE 10 2017 125 180 A1** 2018.10.04

(12)

Offenlegungsschrift

(21) Aktenzeichen: **10 2017 125 180.4**

(22) Anmeldetag: **26.10.2017**

(43) Offenlegungstag: **04.10.2018**

(51) Int Cl.: **G06F 11/30 (2006.01)**

(30) Unionspriorität:

15/472,932

29.03.2017

US

(71) Anmelder:

GOOGLE LLC, Mountain View, Calif., US

(74) Vertreter:

**Betten & Resch Patent- und Rechtsanwälte
PartGmbH, 80333 München, DE**

(72) Erfinder:

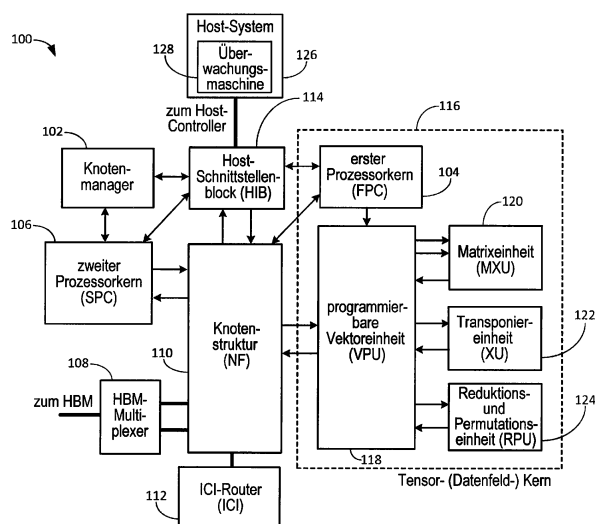
**Norrie, Thomas, Mountain View, Calif., US; Kumar,
Naveen, Mountain View, Calif., US**

Prüfungsantrag gemäß § 44 PatG ist gestellt.

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen.

(54) Bezeichnung: **Synchrone Hardware-Ereignis-Sammlung**

(57) Zusammenfassung: Ein computerimplementiertes Verfahren, das das Überwachen der Ausführung von Programmcode durch eine erste und eine zweite Prozessorkomponente enthält. Ein Computersystem detektiert, dass eine Auslösebedingung erfüllt ist, durch: i) Identifizieren eines Operanden in einem Abschnitt des Programmcodes; oder ii) Bestimmen, dass ein aktueller Zeitpunkt eines Takts des Computersystems einen vorgegebenen Zeitwert angibt. Der Operand und der vorgegebene Zeitwert werden verwendet, um Verfolgungsereignisse einzuleiten. Wenn die Auslösebedingung erfüllt ist, leitet das System Verfolgungsereignisse ein, die Verfolgungsdaten erzeugen, die jeweilige Hardware-Ereignisse identifizieren, die über das Computersystem auftreten. Das System verwendet die Verfolgungsdaten, um einen korrelierten Satz von Verfolgungsdaten zu erzeugen. Die korrelierten Verfolgungsdaten geben einen zeitlich geordneten Ablauf der jeweiligen Hardware-Ereignisse an. Das System verwendet den korrelierten Satz von Verfolgungsdaten, um die Leistung des ausgeführten Programmcodes zu analysieren.



Beschreibung**HINTERGRUND**

[0001] Diese Beschreibung bezieht sich auf das Analysieren der Ausführung von Programmcode.

[0002] Die effektive Leistungsanalyse verteilter Software, die innerhalb verteilter Hardware-Komponenten ausgeführt wird, kann eine komplexe Aufgabe sein. Die verteilten Hardware-Komponenten können die jeweiligen Prozessorkerne von zwei oder mehr Zentraleinheiten (CPUs) (oder Graphikverarbeitungseinheiten (GPUs)) sein, die zusammenarbeiten und wechselwirken, um Abschnitte eines größeren Software-Programms oder Programmcodes auszuführen.

[0003] Aus der Hardware-Perspektive (z. B. innerhalb der CPUs oder GPUs) gibt es im Allgemeinen zwei Typen von Informationen oder Merkmalen, die für die Leistungsanalyse verfügbar sind: 1) Hardware-Leistungs-Zähler und 2) Hardware-Ereignis-Verfolgungen.

ZUSAMMENFASSUNG

[0004] Im Allgemeinen kann ein erfinderischer Aspekt des in dieser Beschreibung beschriebenen Gegenstands in einem computerimplementierten Verfahren verkörpert sein, das durch einen oder mehrere Prozessoren ausgeführt wird, wobei das Verfahren das Überwachen der Ausführung des Programmcodes durch eine erste Prozessorkomponente, wobei die erste Prozessorkomponente konfiguriert ist, um wenigstens einen ersten Abschnitt des Programmcodes auszuführen; und das Überwachen der Ausführung des Programmcodes durch eine zweite Prozessorkomponente, wobei die zweite Prozessorkomponente konfiguriert ist, um wenigstens einen zweiten Abschnitt des Programmcodes auszuführen, enthält.

[0005] Das Verfahren enthält ferner das Detektieren durch ein Computersystem, das eine Auslösebedingung erfüllt ist, basierend wenigstens auf einem des Folgenden: i) Identifizieren eines Auftretens eines Operanden in wenigstens dem ersten Abschnitt des Programmcodes oder dem zweiten Abschnitt des Programmcodes, wobei der Operand einen ersten Parameterwert enthält, der verwendet wird, um ein oder mehrere Verfolgungsereignisse einzuleiten, oder ii) Bestimmen, dass ein aktueller Zeitpunkt wenigstens eines Takts des Computersystems einen vorgegebenen Zeitwert angibt, der verwendet wird, um ein oder mehrere Verfolgungsereignisse einzuleiten.

[0006] In Reaktion auf das Detektieren, dass die Auslösebedingung erfüllt ist, enthält das Verfahren das Einleiten durch das Computersystem wenig-

tens eines ersten Verfolgungsereignisses, das Verfolungsdaten erzeugt, wobei die Verfolungsdaten jeweilige Hardware-Ereignisse identifizieren, die über die verteilten Prozessoreinheiten auftreten, die wenigstens die erste Prozessorkomponente und die zweite Prozessorkomponente enthalten. Für jedes der jeweiligen Hardware-Ereignisse umfassen die Verfolungsdaten wenigstens einen Hardware-Ereignis-Zeitstempel. Das Verfahren enthält ferner das Verwenden durch das Computersystem der Verfolungsdaten, um einen korrelierten Satz von Verfolungsdaten zu erzeugen, die wenigstens eine zeitlich geordnete Folge der jeweiligen Hardware-Ereignisse angeben, die erzeugt werden, wenn die Auslösebedingung erfüllt ist.

[0007] Diese und andere Implementierungen können jede optional eines oder mehrere der folgenden Merkmale enthalten. In einigen Implementierungen ist z. B. das erste Verfolgungsereignis ein synchronisiertes Verfolgungsereignis, das Verfolungsdaten erzeugt, die die über die verteilten Prozessoreinheiten auftretenden Hardware-Ereignisse identifizieren, wobei die Verfolungsdaten außerdem eine eindeutige Verfolgungskennung für die jeweiligen Hardware-Ereignisse identifizieren, und wobei die Hardware-Ereignisse mehrere synchronisierte Hardware-Ereignisse enthalten, wobei zwei Hardware-Ereignisse synchronisiert sind, wenn die Ereignisse einen globalen Hardware-Ereignis-Zeitstempel gemeinsam benutzen.

[0008] In einigen Implementierungen umfasst das Detektieren, dass die Auslösebedingung erfüllt ist: Detektieren durch das Computersystem, dass eines des Folgenden : i) der erste Parameterwert des Operanden übersteigt einen ersten Grenzwert eines Registers oder ii) der durch den aktuellen Zeitpunkt angegebene vorgegebene Zeitwert übersteigt einen ersten Grenzwert eines Registers; und in Reaktion auf das Detektieren, dass die Auslösebedingung erfüllt ist, Einleiten durch das Computersystem eines zweiten Verfolgungsereignisses, das Verfolungsdaten erzeugt, wobei die Verfolungsdaten wenigstens ein Attribut identifizieren, das unter den jeweiligen Hardware-Ereignissen, die über die verteilten Prozessoreinheiten auftreten, gemeinsam benutzt wird.

[0009] In einigen Implementierungen enthält das Verfahren ferner: Detektieren durch das Computersystem, dass eines des Folgenden: ein zweiter Parameterwert des Operanden übersteigt einen zweiten Grenzwert des Registers oder ein zweiter vorgegebener Zeitwert, der durch den aktuellen Zeitpunkt angegeben wird, übersteigt einen zweiten Grenzwert des Registers; und in Reaktion auf das Detektieren Stoppen durch das Computersystem des zweiten Verfolgungsereignisses, wenn der zweite Parameterwert des Operanden den zweiten Grenzwert über-

steigt oder wenn der zweite vorgegebene Zeitwert den zweiten Grenzwert übersteigt.

[0010] In einigen Implementierungen enthält der Operand ferner wenigstens einen der Folgenden: einen Parameter, der einen speziellen Ablaufschritt des Programmcodes angibt; oder einen globalen Steuerparameter, der einen speziellen Leistungszustand der verteilten Prozessoreinheiten angibt; und enthält der vorgegebene Zeitwert wenigstens einen der Folgenden: einen speziellen Zeitwert, der durch einen globalen Zeittakt der verteilten Prozessoreinheiten angegeben wird; oder einen speziellen Zeitwert eines vorgegebenen Zeitfensters, das dem globalen Zeittakt zugeordnet ist.

[0011] In einigen Implementierungen weist der Operand eine erste binäre Datenstruktur auf und entspricht der erste Parameterwert des ersten Operanden einer Verfolgungsmarkierung, weist der vorgegebene Zeitwert eine zweite binäre Datenstruktur auf und wird der aktuelle Zeitpunkt durch einen globalen Zeittakt angegeben, wobei der globale Zeittakt durch die verteilten Prozessoreinheiten verwendet wird, um einen oder mehrere Hardware-Ereignis-Zeitstempel zu erzeugen.

[0012] In einigen Implementierungen enthält das Verfahren ferner: Einfügen durch einen Kompilierer des Computersystems des Operanden der Auslösebedingung in wenigstens den ersten Abschnitt des durch die erste Prozessorkomponente ausgeführten Programmcodes; und Einfügen durch den Kompilierer des Computersystems wenigstens eines vorgegebenen Zeitwerts der Auslösebedingung in wenigstens den zweiten Abschnitt des durch die zweite Prozessorkomponente ausgeführten Programmcodes.

[0013] In einigen Implementierungen enthält das Einleiten wenigstens eines des ersten Verfolgungsereignisses oder des zweiten Verfolgungsereignisses: Erzeugen durch das Computersystem eines ersten Steuersignals, das durch ein erstes Zählregister des ersten Prozessorkerns empfangen wird, wobei das erste Steuersignal verursacht, dass die einem ersten Hardware-Ereignis zugeordneten Daten in dem ersten Zählregister gespeichert werden; und Erzeugen durch das Computersystem eines zweiten Steuersignals, das durch ein zweites Zählregister des zweiten Prozessorkerns empfangen wird, wobei das zweite Steuersignal verursacht, dass die einem zweiten Hardware-Ereignis zugeordneten Daten in dem zweiten Zählregister gespeichert werden.

[0014] In einigen Implementierungen umfassen die Daten, die entweder dem ersten Hardware-Ereignis oder dem zweiten Hardware-Ereignis zugeordnet sind, wenigstens eines des Folgenden: eine Anzahl von Bytes, die in einen speziellen Speicherpuffer eines speziellen Prozessorkerns der verteilten Pro-

zessoreinheiten geschrieben sind; oder eine Anzahl von Anweisungen, die durch einen speziellen Prozessorkern der verteilten Prozessoreinheiten ausgeführt werden.

[0015] In einigen Implementierungen enthält das Verfahren ferner: Identifizieren durch das Computersystem eines Auftretens eines zweiten Operanden in wenigstens einem Abschnitt des durch die erste oder die zweite Prozessorkomponente ausgeführten Programmcodes, wobei der zweite Operand einen zweiten Parameterwert enthält; Bestimmen durch das Computersystem, das eine Filterbedingung erfüllt ist, basierend auf dem zweiten Parameterwert des zweiten Operanden, der entweder: einen speziellen Grenzwert eines Registers übersteigt oder sich unter einem speziellen Grenzwert des Registers befindet; und in Reaktion auf das Bestimmen, dass die Filterbedingung erfüllt ist, Filtern durch das Computersystem eines oder mehrerer Verfolgungsereignisse, wobei das Filtern des einen oder der mehreren Verfolgungsereignisse das Ausschließen der Speicherung der einem oder mehreren Hardware-Ereignissen zugeordneten Verfolgungsdaten umfasst.

[0016] Ein weiterer erfinderischer Aspekt des in dieser Beschreibung beschriebenen Gegenstands kann in einem Hardware-Ereignis-Sammlungssystem verkörpert sein, das enthält: einen oder mehrere Prozessoren, die einen oder mehrere Prozessorkerne enthalten; eine oder mehrere maschinenlesbare Speichereinheiten zum Speichern von Anweisungen, die durch den einen oder die mehreren Prozessoren ausführbar sind, um Operationen auszuführen, die umfassen: Überwachen der Ausführung des Programmcodes durch eine erste Prozessorkomponente, wobei die erste Prozessorkomponente konfiguriert ist, um wenigstens einen ersten Abschnitt des Programmcodes auszuführen; und Überwachen der Ausführung des Programmcodes durch eine zweite Prozessorkomponente, wobei die zweite Prozessorkomponente konfiguriert ist, um wenigstens einen zweiten Abschnitt des Programmcodes auszuführen.

[0017] Das Verfahren enthält ferner das Detektieren durch ein Computersystem, das eine Auslösebedingung erfüllt ist, basierend auf wenigstens einem des Folgenden: i) Identifizieren eines Auftretens eines Operanden in wenigstens dem ersten Abschnitt des Programmcodes oder dem zweiten Abschnitt des Programmcodes, wobei der Operand einen ersten Parameterwert enthält, der verwendet wird, um ein oder mehrere Verfolgungsereignisse einzuleiten, oder ii) Bestimmen, dass ein aktueller Zeitpunkt wenigstens eines Takts des Computersystems einen vorgegebenen Zeitwert angibt, der verwendet wird, um ein oder mehrere Verfolgungsereignisse einzuleiten.

[0018] In Reaktion auf das Detektieren, dass die Auslösebedingung erfüllt ist, enthält das Verfahren das Einleiten durch das Computersystem wenigstens eines ersten Verfolgungsereignisses, das Verfolungsdaten erzeugt, wobei die Verfolungsdaten jeweilige Hardware-Ereignisse identifizieren, die über die verteilten Prozessoreinheiten auftreten, die wenigstens die erste Prozessorkomponente und die zweite Prozessorkomponente enthalten. Für jedes der jeweiligen Hardware-Ereignisse umfassen die Verfolungsdaten wenigstens einen Hardware-Ereignis-Zeitstempel. Das Verfahren enthält ferner das Verwenden durch das Computersystem der Verfolungsdaten, um einen korrelierten Satz von Verfolungsdaten zu erzeugen, die wenigstens eine zeitlich geordnete Folge der jeweiligen Hardware-Ereignisse angeben, die erzeugt werden, wenn die Auslösebedingung erfüllt ist.

[0019] Im Allgemeinen kann ein erfinderischer Aspekt des in dieser Beschreibung beschriebenen Gegenstands in einem computerimplementierten Verfahren verkörpert sein, das durch einen oder mehrere Prozessoren ausgeführt wird, wobei das Verfahren das Überwachen der Ausführung des Programmcodes durch eine Prozessorkomponente enthält, wobei die Prozessorkomponente konfiguriert ist, um wenigstens einen ersten Abschnitt des Programmcodes auszuführen.

[0020] Das Verfahren enthält ferner das Detektieren durch ein Computersystem, das eine Auslösebedingung erfüllt ist, basierend auf wenigstens einem des Folgenden: i) Identifizieren eines Auftretens eines Operanden in wenigstens dem ersten Abschnitt des Programmcodes, wobei der Operand einen ersten Parameterwert enthält, der verwendet wird, um ein oder mehrere Verfolgungsereignisse einzuleiten, oder ii) Bestimmen, dass ein aktueller Zeitpunkt wenigstens eines Takts des Computersystems einen vorgegebenen Zeitwert angibt, der verwendet wird, um das eine oder die mehreren Verfolgungsereignisse einzuleiten.

[0021] In Reaktion auf das Detektieren, dass die Auslösebedingung erfüllt ist, enthält das Verfahren ferner das Erzeugen durch das Computersystem eines Steuersignals, das durch ein Zählregister der Prozessorkomponente empfangen wird, wobei das Steuersignal verursacht, dass die einem Hardware-Ereignis zugeordneten Zählraten in dem Zählregister gespeichert werden; und Erzeugen durch das Computersystem einer Datenstruktur, die ein oder mehrere Leistungsattribute angibt, die dem ausgeführten Programmcodes zugeordnet sind, wobei die Datenstruktur basierend auf einem oder mehreren Zählparametern der gespeicherten Zählraten erzeugt wird.

[0022] Diese und andere Implementierungen können optional ein oder mehrere der folgenden Merk-

male enthalten. In einigen Implementierungen ist das Zählregister z. B. einer von mehreren Leistungszählern, die konfiguriert sind, um die Zählraten über die Leistung einer oder mehrerer Prozessorkomponenten des Computersystems zu speichern, wobei wenigstens ein Leistungszähler einen der Folgenden enthält: einen Aktivitätszähler, einen Stehenbleiben-Zähler, einen statistischen Zähler oder einen Abtastzähler.

[0023] In einigen Implementierungen geben der eine oder die mehreren Zählparameter eine der Folgenden an: i) eine Anzahl der durch eine spezielle Prozessorkomponente empfangenen Anweisungen; ii) eine Anzahl der durch die spezielle Prozessorkomponente verarbeiteten Anweisungen; iii) eine Anzahl der durch die spezielle Prozessorkomponente ausgeführten Anweisungen; oder iv) eine Anzahl der Speicherlesevorgänge oder eine Anzahl der Speicherschreibvorgänge, die durch die spezielle Prozessorkomponente ausgeführt werden.

[0024] In einigen Implementierungen umfasst ein Leistungsattribut, das dem ausgeführten Programmcodes zugeordnet ist, eines des Folgenden: i) eine Stehenbleiben-Frequenz einer speziellen Prozessorkomponente, die den Programmcodes ausführt; ii) eine Angabe, dass sich die Verwendung der speziellen Prozessorkomponente unter einer Schwellenverwendung befindet; oder iii) eine Angabe, dass sich eine durch die spezielle Prozessorkomponente verwendete Datenspeicher-Warteschlange auf oder unter einem Warteschlangenbelegungsschwellenwert befindet.

[0025] Andere Implementierungen dieser und anderer Aspekte enthalten entsprechende Systeme, Vorrichtungen und Computerprogramme, die konfiguriert sind, um die Handlungen der Verfahren auszuführen, die in Computer-Speichervorrichtungen codiert sind. Ein System aus einem oder mehreren Computern kann daher aufgrund Software, Firmware, Hardware oder einer Kombination aus diesen, die in einem System installiert ist, konfiguriert sein, die in Betrieb das System veranlassen, die Handlungen auszuführen. Ein oder mehrere Computerprogramme können daher aufgrund dessen konfiguriert sein, dass sie Anweisungen aufweisen, die, wenn sie durch eine Datenverarbeitungsvorrichtung ausgeführt werden, die Vorrichtung veranlassen, die Handlungen auszuführen.

[0026] Der in dieser Beschreibung beschriebene Gegenstand kann in speziellen Ausführungsformen implementiert sein, um einen oder mehrere der folgenden Vorteile zu verwirklichen. Das beschriebene Hardware-Verfolgungssystem verbessert den Rechenwirkungsgrad durch die Verwendung dynamischer Auslöser, die durch Hardware-Knöpfe/Merkmale ausgeführt werden. Die Auslöser ermöglichen

die synchronisierte Erfassung von Ereignissen teilweise basierend auf einem globalen Zeitgeber, eingebetteten Logikoperanden und Hardware-Registern anstatt rein durch codierte Folgen, die typischerweise Systembetriebsmittel erfordern, um die Ereigniserfassung zu überwachen und auszuführen.

[0027] Gleichmaßen ist die Verwendung des Systemspeichers optimiert, wenn die Hardware-Auslöser abgestimmt sind, um Ereignisse mit kurzer Dauer und synchronisierte globale Erfassungen, die gleichzeitig auftreten, anstatt als nicht kontrollierte ineffiziente Ereigniserfassungen zu erfassen. Die Steuerungen für die synchronisierte Erfassung von Ereignissen mit kürzerer Dauer schwächen die Informationsüberlastung ab und ermöglichen dadurch, dass einige Speicherbetriebsmittel unbenutzt und für andere Systemprozesse verfügbar bleiben.

[0028] Die Einzelheiten einer oder mehrerer Implementierungen des in dieser Beschreibung beschriebenen Gegenstands sind in den beigefügten Zeichnungen und in der Beschreibung im Folgenden dargestellt. Andere potentielle Merkmale, Aspekte und Vorteile des Gegenstands werden aus der Beschreibung, den Zeichnungen und den Ansprüchen offensichtlich werden.

Figurenliste

Fig. 1 veranschaulicht einen Blockschaltplan eines beispielhaften Computersystems zur verteilten Hardware-Verfolgung.

Fig. 2 veranschaulicht einen Blockschaltplan der Verfolgungsketten und jeweiligen Knoten eines beispielhaften Computersystems zur verteilten Hardware-Verfolgung.

Fig. 3 veranschaulicht einen Blockschaltplan einer beispielhaften Verfolgungsmultiplexer-Konfigurationsarchitektur und eine beispielhafte Datenstruktur.

Fig. 4 ist ein Blockschaltplan, der eine Verfolgungsaktivität für ein Speicherdirektzugriffs-Verfolgungsereignis angibt, das durch ein beispielhaftes Computersystem zur verteilten Hardware-Verfolgung ausgeführt wird.

Fig. 5 veranschaulicht eine beispielhafte Datenstruktur für einen globalen Zeitzähler (GTC) und Tabellen, die die Verwendungsszenarios des GTC durch ein beispielhaftes Computersystem zur verteilten Hardware-Verfolgung angeben.

Fig. 6 veranschaulicht einen Blockschaltplan beispielhafter Zeitzähler und Hardware-Merkmale, die den Komponenten eines beispielhaf-

ten Computersystems zur verteilten Hardware-Verfolgung zugeordnet sind.

Fig. 7 ist ein Prozessablaufplan eines beispielhaften Prozesses zur verteilten Hardware-Verfolgung.

[0029] Gleiche Bezugszeichen und Bezeichnungen in den verschiedenen Zeichnungen geben gleiche Elemente an.

AUSFÜHRLICHE BESCHREIBUNG

[0030] Der in dieser Beschreibung beschriebene Gegenstand bezieht sich im Allgemeinen auf die verteilte Hardware-Verfolgung. Insbesondere überwacht ein Computersystem die Ausführung von Programmcode, der durch einen ersten Prozessorkern ausgeführt wird, und die Ausführung von Programmcode, der durch einen zweiten Prozessorkern ausgeführt wird. Das Computersystem speichert eine Zeitachse der Hardware-Ereignisse in einem Speicherpuffer. Die gespeicherten Ereignisse treten über die verteilten Prozessoreinheiten auf, die wenigstens den ersten und den zweiten Prozessorkern enthalten.

[0031] Die Zeitachse enthält für jedes der Hardware-Ereignisse einen Ereignis-Zeitstempel und Metadaten, die das Hardware-Ereignis charakterisieren. Das System erzeugt eine Datenstruktur, die die Hardware-Ereignisse von der Zeitachse enthält. Das System speichert die Datenstruktur in einer Speicherebene einer Host-Vorrichtung und verwendet die Datenstruktur, um die Leistung des durch den ersten oder den zweiten Prozessorkern ausgeführten Programmcodes zu bewerten. Innerhalb dieses Kontexts der Ereignisverfolgung beschreibt diese Beschreibung die Herangehensweisen für die Hardware-Ereignis-Verfolgung in einem verteilten Computersystem, die in den **Fig. 1-4** gezeigt sind.

[0032] Diese Beschreibung beschreibt ferner die Herangehensweisen für eine synchronisierte Hardware-Ereignis-Sammlung basierend auf einem oder mehreren Auslösemechanismen, die in den **Fig. 5-7** gezeigt sind. Wie im Folgenden ausführlicher erörtert wird, beziehen sich die Aspekte der in dieser Beschreibung beschriebenen Computersysteme wenigstens auf die koordinierte/synchrone Sammlung von Verfolgungsdaten und die Ereigniszählung. Insbesondere enthält wenigstens ein Aspekt Systeme und Verfahren zur synchronisierten Sammlung von Hardware-Leistungszählerdaten und Verfolgungsereignisdaten sowohl innerhalb eines selbstständigen als auch innerhalb eines verteilten Systems. Die synchronisierte Ereignissammlung verbessert die Analyse sowohl der Leistungsdaten als auch der Fehlerbehebungsdaten des verteilten Programmcodes. Die verbesserte Analyse wird teilweise durch die Korrelation der Ereignisse erreicht, die in Reaktion auf die Ausführung der durch das System **100** analysierten

verbundenen Software-Komponenten/Module auftreten.

[0033] Fig. 1 veranschaulicht einen Blockschaltplan eines beispielhaften Computersystems 100 für die verteilte Hardware-Verfolgung. Die verteilte Hardware-System-Verfolgung, wie sie in dieser Beschreibung verwendet wird, entspricht der Speicherung von Ereignissen, die innerhalb der Komponenten und Unterkomponenten eines beispielhaften Prozessor-Mikrochips auftreten. Ferner entspricht ein verteiltes Hardware-System (oder Verfolgungssystem), wie es hier verwendet wird, einer Sammlung von Prozessor-Mikrochips, die zusammenarbeiten, um die jeweiligen Abschnitte einer Software/eines Programmcodes auszuführen, die/der für die verteilte Ausführung zwischen der Sammlung von Prozessor-Mikrochips konfiguriert ist. In einigen Implementierungen können verschiedene Prozessor-Chips des Systems 100 jeweilige Knoten des verteilten Hardware-Systems bilden. In alternativen Implementierungen kann ein einziger Prozessor-Chip einen oder mehrere Prozessorkerne und die Hardware-Merkmale, die jeweils die entsprechenden Knoten des Prozessor-Chips bilden können, enthalten.

[0034] Im Kontext einer Zentraleinheit (CPU) kann ein Prozessor-Chip z. B. wenigstens zwei Knoten enthalten, wobei jeder Knoten ein jeweiliger Kern der CPU sein kann. Alternativ kann im Kontext einer Graphikprozessoreinheit (GPU) ein Prozessor-Chip wenigstens zwei Knoten enthalten, wobei jeder Knoten ein jeweiliger Streaming-Multiprozessor der GPU sein kann. Das Computersystem 100 kann mehrere Prozessorkomponenten enthalten. In einigen Implementierungen können die Prozessorkomponenten ein Prozessor-Chip und/oder ein Prozessorkern und/oder eine Speicherzugriffsmaschine und/oder wenigstens eine Hardware-Komponente des Gesamt-Computersystems 100 sein.

[0035] In einigen Fällen kann eine Prozessorkomponente, wie z. B. ein Prozessorkern, eine Komponente mit fester Funktion sein, die konfiguriert ist, um basierend auf wenigstens einer ausgegebenen Anweisung des ausgeführten Programmcodes wenigstens eine spezifische Operation auszuführen. In anderen Fällen kann eine Prozessorkomponente, wie z. B. eine Speicherzugriffsmaschine (MAE), konfiguriert sein, um Programmcode auf einer tieferen Ebene der Einzelheiten oder der Granularität als der durch die anderen Prozessorkomponenten des Systems 100 ausgeführte Programmcode auszuführen.

[0036] Der durch einen Prozessorkern ausgeführte Programmcode kann z. B. verursachen, dass ein MAE-Deskriptor erzeugt und an die MAE übertragen/gesendet wird. Nach dem Empfang des Deskriptors kann die MAE eine Datenübertragungsoperation basierend auf dem MAE-Deskriptor ausführen.

In einigen Implementierungen können die durch die MAE ausgeführten Datenübertragungen z. B. das Bewegen von Daten über bestimmte Datenwege oder Schnittstellenkomponenten des Systems zu und von bestimmten Komponenten des Systems 100 oder das Ausgeben von Datenanforderungen auf einem beispielhaften Konfigurationsbus des Systems 100 enthalten.

[0037] In einigen Implementierungen kann jeder Tensorknoten eines beispielhaften Prozessor-Chips des Systems 100 wenigstens zwei „Front-Ends“ aufweisen, die Hardware-Blöcke/Merkmale sein können, die Programmanweisungen verarbeiten. Wie im Folgenden ausführlicher erörtert wird, kann ein erstes Front-End dem ersten Prozessorkern 104 entsprechen, während ein zweites Front-End dem zweiten Prozessorkern 106 entsprechen kann. Folglich können der erste und der zweite Prozessorkern hier außerdem als das erste Front-End 104 und das zweite Front-End 106 beschrieben werden.

[0038] Eine Verfolgungskette, wie sie in dieser Beschreibung verwendet wird, kann ein spezifischer physischer Datenübertragungsbus sein, auf den die Verfolgungseinträge für die Übertragung zu einem beispielhaften Chip-Manager innerhalb des Systems 100 gesetzt werden können. Die empfangenen Dateneinträge können Datenwörter/-strukturen sein, die mehrere Bytes und mehrere binäre Werte oder Ziffern enthalten. Folglich gibt der Deskriptor „Wort“ ein Stück mit fester Größe der binären Daten an, das durch die Hardware-Vorrichtungen eines beispielhaften Prozessorkerns als eine Einheit behandelt werden kann.

[0039] In einigen Implementierungen sind die Prozessor-Chips des verteilten Hardware-Verfolgungssystems Mehrkernprozessoren (d. h., sie weisen mehrere Kerne auf), die jeder Abschnitte des Programmcodes in den jeweiligen Kernen des Chips ausführen. In einigen Implementierungen können die Abschnitte des Programmcodes vektorisierten Berechnungen für Schlussfolgerungs-Arbeitsbelastungen eines beispielhaften mehrschichtigen neuronalen Netzes entsprechen. Während in alternativen Implementierungen die Abschnitte des Programmcodes im Allgemeinen Software-Modulen, die herkömmlichen Programmiersprachen zugeordnet sind, entsprechen können.

[0040] Das Computersystem 100 enthält im Allgemeinen einen Knoten-Manager 102, einen ersten Prozessorkern (FPC) 104, einen zweiten Prozessorkern (SPC) 106, eine Knotenstruktur (NF) 110, einen Daten-Router 112 und einen Host-Schnittstellenblock (HIB) 114. In einigen Implementierungen kann das System 100 einen Speichermultiplexer 108 enthalten, der konfiguriert ist, um die Funktionen der Signalvermittlung, -multiplexierung und -demultiplexierung

xierung auszuführen. Das System 100 enthält ferner einen Tensor Kern **116**, der den darin angeordneten FPC 104 enthält. Der Tensor Kern kann eine beispielhafte Rechenvorrichtung sein, die konfiguriert ist, um vektorbasierte Berechnungen an mehrdimensionalen Datenfeldern auszuführen. Der Tensor Kern **116** kann eine Vektorverarbeitungseinheit (VPU) **118** enthalten, die mit einer Matrixeinheit (MXU) **120**, einer Transponiereinheit (XU) **122** und einer Reduktions- und Permutationseinheit (RPU) **124** in Wechselwirkung tritt. In einigen Implementierungen kann das Computersystem **100** eine oder mehrere Ausführungseinheiten einer herkömmlichen CPU oder GPU, wie z. B. Lade-/Speichereinheiten, Arithmetik-Logik-Einheiten (ALUs) und Vektoreinheiten, enthalten.

[0041] Die Komponenten des Systems **100** enthalten gemeinsam sowohl einen großen Satz von Hardware-Leistungs-Zählern als auch Unterstützungs-Hardware, die den Abschluss der Verfolgungsaktivität innerhalb der Komponenten fördert. Wie im Folgenden ausführlicher beschrieben wird, kann der durch die jeweiligen Prozessorkerne des Systems **100** ausgeführte Programmcode eingebettete Auslöser enthalten, die verwendet werden, um während der Codeausführung mehrere Leistungszähler gleichzeitig freizugeben. Im Allgemeinen verursachen die detektierten Auslöser, dass für ein oder mehrere Verfolgungsereignisse Verfolungsdaten erzeugt werden. Die Verfolungsdaten können den inkrementalen Parameterzählerständen entsprechen, die in den Zählern gespeichert sind und die analysiert werden können, um die Leistungseigenschaften des Programmcodes zu erkennen. Die Daten für die jeweiligen Verfolgungsereignisse können in einem beispielhaften Speichermedium (z. B. einem Hardware-Puffer) gespeichert sein und können einen Zeitstempel enthalten, der in Reaktion auf die Detektion des Auslösers erzeugt wird.

[0042] Ferner können die Verfolungsdaten für verschiedene Ereignisse, die innerhalb der Hardware-Komponenten des Systems **100** auftreten, erzeugt werden. Beispielhafte Ereignisse können Zwischenknoten- und knotenübergreifende Kommunikationsoperationen, wie z. B. Speicherdirektzugriffsoperationen (DMA-Operationen) und Synchronisationsmerkeraktualisierungen (die jeweils im Folgenden ausführlicher beschrieben werden), enthalten. In einigen Implementierungen kann das System **100** einen globalen synchronen Zeitstempel-Zähler enthalten, der im Allgemeinen als ein globaler Zeitgeber („GTC“) bezeichnet wird. In anderen Implementierungen kann das System **100** andere Typen globaler Takte, wie z. B. einen Lamport-Takt, enthalten.

[0043] Der GTC kann für die genaue Korrelation der Programmcodeausführung und der Leistung der Software/des Programmcodes, die/der in einer verteilten Verarbeitungsumgebung ausgeführt wird, ver-

wendet werden. Zusätzlich und teilweise bezüglich des GTC kann das System **100** in einigen Implementierungen einen oder mehrere Auslösemechanismen enthalten, die durch verteilte Software-Programme verwendet werden, um die Datenverfolgung in einem verteilten System in einer im hohen Grade koordinierten Weise zu starten und zu stoppen.

[0044] In einigen Implementierungen kompiliert ein Host-System **126** den Programmcode, der eingebettete Operanden enthalten kann, die bei der Detektion auslösen, um die Erfassung und die Speicherung der Verfolungsdaten, die den Hardware-Ereignissen zugeordnet sind, zu veranlassen. In einigen Implementierungen stellt das Host-System **126** den kompilierten Programmcode einem oder mehreren Prozessor-Chips des Systems **100** bereit. In alternativen Implementierungen kann der Programmcode (mit den eingebetteten Auslösern) durch einen beispielhaften externen Kompilierer kompiliert und in den einen oder die mehreren Prozessor-Chips des Systems **100** geladen werden. In einigen Fällen kann der Kompilierer ein oder mehrere (im Folgenden erörterte) Verfolgungsbits setzen, die bestimmten Auslösern zugeordnet sind, die in den Abschnitten der Software-Anweisungen eingebettet sind. Der kompilierte Programmcode kann ein verteiltes Software-Programm sein, das durch die eine oder die mehreren Komponenten des Systems **100** ausgeführt wird.

[0045] Das Host-System **126** kann eine Überwachungsmaschine **128** enthalten, die konfiguriert ist, um die Ausführung des Programmcodes durch eine oder mehrere Komponenten des Systems **100** zu überwachen. In einigen Implementierungen ermöglicht es die Überwachungsmaschine **128** dem Host-System **126**, die Ausführung des Programmcodes zu überwachen, der wenigstens durch den FPC **104** und den SPC **106** ausgeführt wird. Während der Codeausführung kann das Host-System **126** z. B. über die Überwachungsmaschine **128** die Leistung des Ausführens des Codes wenigstens durch das Empfangen periodischer Zeitachsen der Hardware-Ereignisse basierend auf den erzeugten Verfolungsdaten überwachen. Obwohl für das Host-System **126** ein einziger Block gezeigt ist, kann das System **126** in einigen Implementierungen mehrerer Hosts (oder Host-Untersysteme) enthalten, die mehreren Prozessor-Chips oder Chip-Kernen des Systems **100** zugeordnet sind.

[0046] In anderen Implementierungen können die knotenübergreifenden Kommunikationen, die wenigstens drei Prozessorkerne einbeziehen, das Host-System **126** veranlassen, den Datenverkehr an einem oder mehreren dazwischenliegenden „Sprüngen“ zu überwachen, wenn der Datenverkehr einen Kommunikationsweg zwischen dem FPC **104** und einem beispielhaften dritten Prozessorkern/-knoten durchläuft. Der FPC **104** und der dritte Prozessor-

kern können z. B. die einzigen Kerne sein, die in einem gegebenen Zeitraum Programmcode ausführen. Folglich kann die Datenübertragung von dem FPC **104** zu dem dritten Prozessorkern Verfolgungsdaten für einen dazwischenliegenden Sprung an dem SPC **106** erzeugen, wenn die Daten von dem FPC **104** zu dem dritten Prozessorkern übertragen werden. Anders dargelegt, während der Datenleitung in dem System **100** kann es sein, dass die Daten von einem ersten Prozessor-Chip, die zu einem dritten Prozessor-Chip gehen, einen zweiten Prozessor-Chip durchlaufen müssen, wobei daher die Ausführung der Datenleitungsoperation verursachen kann, dass für die Leitungsaktivität in dem zweiten Chip Verfolgungseinträge erzeugt werden.

[0047] Bei der Ausführung des kompilierten Programmcodes können die Komponenten des Systems **100** in Wechselwirkung treten, um die Zeitachsen der Hardware-Ereignisse zu erzeugen, die in einem verteilten Computersystem auftreten. Die Hardware-Ereignisse können Innerknoten- und kontenübergreifende Kommunikationsereignisse enthalten. Beispielhafte Knoten eines verteilten Hardware-Systems und ihre zugeordneten Kommunikationen werden im Folgenden bezüglich **Fig. 2** ausführlicher beschrieben. In einigen Implementierungen wird eine Datenstruktur erzeugt, die eine Sammlung von Hardware-Ereignissen für wenigstens eine Hardware-Ereignis-Zeitachse identifiziert. Die Zeitachse ermöglicht die Rekonstruktion der Ereignisse, die in dem verteilten System auftreten. In einigen Implementierungen kann die Ereignisrekonstruktion eine richtige Ereignisordnung basierend auf der Analyse der während des Auftretens eines speziellen Ereignisses erzeugten Zeitstempel enthalten.

[0048] Im Allgemeinen kann ein beispielhaftes verteiltes Hardware-Verfolgungssystem sowohl die oben beschriebenen Komponenten des Systems **100** als auch wenigstens einen Host-Controller, der einem Host-System **126** zugeordnet ist, enthalten. Die Leistung oder das Austesten der von einem verteilten Verfolgungssystem erhaltenen Daten kann nützlich sein, wenn die Ereignisdaten z. B. in einer zeitlich geordneten oder aufeinanderfolgenden Weise korreliert sind. In einigen Implementierungen kann die Datenkorrelation auftreten, wenn mehrere gespeicherte Hardware-Ereignisse, die verbundenen Software-Modulen entsprechen, gespeichert und dann für die strukturierte Analyse durch das Host-System **126** sequentiell geordnet werden. Für die Implementierungen, die mehrere Host-Systeme enthalten, kann eine Korrelation der über die verschiedenen Hosts erhaltenen Daten z. B. durch den Host-Controller ausgeführt werden.

[0049] In einigen Implementierungen sind sowohl der FPC **104** als auch der SPC **106** selbständige Kerne eines Mehrkern-Prozessor-Chips; während in an-

deren Implementierungen der FPC und der SPC **104**, **106** jeweilige Kerne selbständiger Mehrkern-Prozessor-Chips sind. Wie oben angegeben worden ist, kann das System **100** verteilte Prozessoreinheiten enthalten, die wenigstens den FPC **104** und den SPC **106** aufweisen. In einigen Implementierungen können die verteilten Prozessoreinheiten des Systems **100** eine oder mehrere Hardware- oder Software-Komponenten enthalten, die konfiguriert sind, um wenigstens einen Abschnitt eines größeren verteilten Software-Programms oder Programmcodes auszuführen.

[0050] Ein Daten-Router **112** ist eine Zwischen-Chip-Zusammenschaltung (ICI), die Datenkommunikationswege zwischen den Komponenten des Systems **100** bereitstellt. Insbesondere kann der Router **112** eine Kommunikationskopplung oder Kommunikationsverbindungen zwischen dem FPC **104** und dem SPC **106** und zwischen den jeweiligen Komponenten, die den Kernen **104**, **106** zugeordnet sind, bereitstellen. Eine Knotenstruktur **110** wechselwirkt mit dem Daten-Router **112**, um Datenpakete innerhalb der verteilten Hardware-Komponenten und -Unterkomponenten des Systems **100** zu bewegen.

[0051] Der Knotenmanager **102** ist eine Vorrichtung auf hoher Ebene, die die Knotenfunktionen auf tiefer Ebene in den Mehrknoten-Prozessor-Chips managt. Wie im Folgenden ausführlicher erörtert wird, können ein oder mehrere Knoten eines Prozessor-Chips Chip-Manager enthalten, die durch den Knotenmanager **102** gesteuert sind, um die Hardware-Ereignisdaten in lokalen Eintragsprotokollen zu managen und zu speichern. Der Speichermultiplexer **108** ist eine Multiplexierungsvorrichtung, die Vermittlungs-, Multiplexierungs- und Demultiplexierungsoperationen an den einem beispielhaften externen Speicher mit hoher Bandbreite (HBM) bereitgestellten Datensignalen oder den von dem externen HBM empfangenen Datensignalen ausführen kann.

[0052] In einigen Implementierungen kann ein (im Folgenden beschriebener) beispielhafter Verfolgungseintrag durch den Multiplexer **108** erzeugt werden, wenn der Multiplexer **108** zwischen dem FPC **104** und dem SPC **106** wechselt. Der Speichermultiplexer **108** kann die Leistung eines speziellen Prozessorkerns **104**, **106**, der nicht auf den Multiplexer **108** zugreifen kann, potentiell beeinflussen. Folglich können die durch den Multiplexer **108** erzeugten Verfolgungseintragsdaten das Verständnis der resultierenden Spitzen der Latenzzeiten bestimmter Systemaktivitäten, die den jeweiligen Kernen **104**, **106** zugeordnet sind, unterstützen. In einigen Implementierungen können die Hardware-Ereignisdaten (z. B. die im Folgenden erörterten Verfolgungspunkte), die innerhalb des Multiplexers **108** entstehen, zusammen mit den Ereignisdaten für die Knotenstruktur **110** in einer beispielhaften Hardware-Er-

ereignis-Zeitachse gruppiert werden. Die Ereignisgruppierung kann auftreten, wenn eine bestimmte Verfolgungsaktivität veranlasst, dass die Ereignisdaten für mehrere Hardware-Komponenten in einem beispielhaften Hardware-Puffer (z. B. dem im Folgenden erörterten Verfolgungseintragsprotokoll **218**) gespeichert werden.

[0053] Im System **100** umfasst die Leistungsanalyse-Hardware den FPC **104**, den SPC **106**, den Multiplexer **108**, die Knotenstruktur **110**, den Daten-Router **112** und den HIB **114**. Jede dieser Hardware-Komponenten oder -Einheiten enthält sowohl Hardware-Leistungszähler als auch Hardware-Ereignisverfolgungs-Einrichtungen und -Funktionen. In einigen Implementierungen enthalten die VPU **118**, die MXU **120**, die XU **122** und die RPU **124** nicht ihre eigene dedizierte Leistungs-Hardware. Stattdessen kann in derartigen Implementierungen der FPC **104** konfiguriert sein, um die notwendigen Zähler für die VPU **118**, die MXU **120**, die XU **122** und die RPU **124** bereitzustellen.

[0054] Die VPU **118** kann eine interne Entwurfsarchitektur enthalten, die eine lokalisierte Datenverarbeitung mit hoher Bandbreite und Arithmetikoperationen, die den Vektorelementen eines beispielhaften Matrix-Vektor-Prozessors zugeordnet sind, unterstützt. Die MXU **120** ist eine Matrixmultiplikationseinheit, die konfiguriert ist, um z. B. bis zu 128×128 -Matrixmultiplikationen an Vektordatensätzen von Multiplikatoren auszuführen.

[0055] Die XU **122** ist eine Transponiereinheit, die konfiguriert ist, um z. B. bis zu 128×128 -Matrixtransponieroperationen an Vektordaten, die den Matrixmultiplikationsoperationen zugeordnet sind, auszuführen. Die RPU **124** kann eine Sigma-Einheit und eine Permutationseinheit enthalten. Die Sigma-Einheit führt sequentielle Reduktionen an den Vektordaten, die den Matrixmultiplikationsoperationen zugeordnet sind, aus. Die Reduktionen können Summen und verschiedene Typen von Vergleichsoperationen enthalten. Die Permutationseinheit kann alle Elemente der Vektordaten, die den Matrixmultiplikationsoperationen zugeordnet sind, vollständig permutierten oder kopieren.

[0056] In einigen Implementierungen kann der durch die Komponenten des Systems **100** ausgeführte Programmcode für das maschinelle Lernen, die Schlussfolgerungsberechnungen neuronaler Netze und/oder eine oder mehrere Speicherdirektzugriffsfunktionen repräsentativ sein. Die Komponenten des Systems **100** können konfiguriert sein, um ein oder mehrere Software-Programme auszuführen, die Anweisungen enthalten, die eine Verarbeitungseinheit(en) oder -vorrichtung(en) des Systems veranlassen, eine oder mehrere Funktionen auszuführen. Es ist vorgesehen, dass der Begriff „Komponente“ irgendeine Daten-

verarbeitungsvorrichtung oder Speichervorrichtung, wie z. B. Steuerstatusregister oder irgendeine andere Vorrichtung, die Daten verarbeiten und speichern kann, enthält.

[0057] Das System **100** kann im Allgemeinen mehrere Verarbeitungseinheiten oder -vorrichtungen enthalten, die einen oder mehrere Prozessoren (z. B. Mikroprozessoren oder Zentraleinheiten (CPUs)), Graphikverarbeitungseinheiten (GPUs), anwendungsspezifische integrierte Schaltungen (ASICs) oder eine Kombination verschiedener Prozessoren enthalten können. In alternativen Ausführungsformen kann das System **100** jeweils andere Computer-Betriebsmittel/Vorrichtungen (z. B. cloud-basierte Server) enthalten, die zusätzliche Verarbeitungsoptionen zum Ausführen der auf die in dieser Beschreibung beschriebenen Hardware-Verfolgungsinformationen bezogenen Berechnungen bereitstellen.

[0058] Die Verarbeitungseinheiten oder -vorrichtungen können ferner eine oder mehrere Speichereinheiten oder Speicherbänke (z. B. Register/Zähler) enthalten. In einigen Implementierungen führen die Verarbeitungseinheiten die programmierten Anweisungen aus, die in dem Speicher der Vorrichtungen des Systems **100** gespeichert sind, um eine oder mehrere der in dieser Beschreibung beschriebenen Funktionen auszuführen. Die Speichereinheiten/-bänke können ein oder mehrere nichttransitorische maschinenlesbare Speichermedien enthalten. Das nichttransitorische maschinenlesbare Speichermedium kann einen Halbleiterspeicher, eine Magnetplatte und eine optische Platte, einen Schreib-Lese-Speicher (RAM), einen Festwertspeicher (ROM), einen löschbaren programmierbaren Festwertspeicher (z. B. einen EPROM, einen EEPROM oder einen Flash-Speicher) oder irgendein anderes greifbares Medium, das Informationen speichern kann, enthalten.

[0059] Fig. 2 veranschaulicht einen Blockschaltplan beispielhafter Verfolgungsketten und jeweiliger beispielhafter Knoten **200**, **201**, die für die durch das System **100** ausgeführte verteilte Hardware-Verfolgung verwendet werden. In einigen Implementierungen können die Knoten **200**, **201** des Systems **100** verschiedene Knoten innerhalb eines einzigen Mehrkernprozessors sein. In anderen Implementierungen kann der Knoten **200** ein erster Knoten in einem ersten Mehrkern-Prozessor-Chip sein, während der Knoten **201** ein zweiter Knoten in einem zweiten Mehrkern-Prozessor-Chip sein kann.

[0060] Obwohl in der Implementierung nach Fig. 2 zwei Knoten dargestellt sind, kann das System **100** in alternativen Implementierungen mehrere Knoten enthalten. Für die Implementierungen, die mehrere Knoten umfassen, können die knotenübergreifenden Datenübertragungen Verfolgungsdaten an dazwischen-

liegenden Sprüngen entlang einem beispielhaften Datenweg, der mehrere Knoten durchläuft, erzeugen. Die dazwischenliegenden Sprünge können z. B. den Datenübertragungen entsprechen, die durch verschiedene Knoten in einem speziellen Datenübertragungsweg hindurchgehen. In einigen Fällen können die Verfolgungsdaten, die ICI-Verfolgungen/Hardware-Ereignissen zugeordnet sind, für einen oder mehrere dazwischenliegenden Sprünge erzeugt werden, die während der knotenübergreifenden Datenübertragungen auftreten, die durch einen oder mehrere Knoten hindurchgehen.

[0061] In einigen Implementierungen sind der Knoten **0** und der Knoten **1** Tensorknoten, die für vektorisierte Berechnungen verwendet werden, die Abschnitten des Programmcodes für Schlussfolgerungs-Arbeitsbelastungen zugeordnet sind. Ein Tensor, wie er in dieser Beschreibung verwendet wird, ist ein mehrdimensionales geometrisches Objekt, wobei beispielhafte mehrdimensionale geometrische Objekte Matrizen und Datenfelder enthalten.

[0062] Wie in der Implementierung nach **Fig. 2** gezeigt ist, enthält der Knoten **200** eine Verfolgungskette **203**, die mit wenigstens einer Teilmenge der Komponenten des Systems **100** in Wechselwirkung tritt. Gleichermäßen enthält der Knoten **201** eine Verfolgungskette **205**, die mit wenigstens einer Teilmenge der Komponenten des Systems **100** in Wechselwirkung tritt. In einigen Implementierungen sind die Knoten **200**, **201** beispielhafte Knoten derselben Teilmenge von Komponenten, während in anderen Implementierungen die Knoten **200**, **201** jeweilige Knoten verschiedener Komponententeilmengen sind. Der Daten-Router/die ICI **112** enthält eine Verfolgungskette **207**, die im Allgemeinen mit den Verfolgungsketten **203** und **205** zusammenläuft, um die Verfolgungsdaten dem Chip-Manager **216** bereitzustellen.

[0063] In der Implementierung nach **Fig. 2** kann jeder der Knoten **200**, **201** jeweilige Komponententeilmengen enthalten, die wenigstens den FPC **104**, den SPC **106**, die Knotenstruktur **110** und den HIB **114** aufweisen. Jede Komponente der Knoten **200**, **201** enthält einen oder mehrere Verfolgungsmultiplexer, die konfiguriert sind, um die (im Folgenden beschriebenen) Verfolgungspunkte, die durch eine spezielle Komponente des Knotens erzeugt werden, zu gruppieren. Der FPC **104** enthält einen Verfolgungsmultiplexer **204**, die Knotenstruktur **110** enthält die Verfolgungsmultiplexer **210a/b**, der SPC **106** enthält die Verfolgungsmultiplexer **206a/b/c/d**, der HIB **214** enthält den Verfolgungsmultiplexer **214** und die ICI **212** enthält den Verfolgungsmultiplexer **212**. In einigen Implementierungen ermöglicht ein Verfolgungssteuerregister für jeden Verfolgungsmultiplexer, dass einzelne Verfolgungspunkte freigegeben und gesperrt werden. In einigen Fällen können für

einen oder mehrere Verfolgungsmultiplexer ihre entsprechenden Verfolgungssteuerregister sowohl einzelne Freigabebits als auch breitere Verfolgungsmultiplexer-Steuerungen enthalten.

[0064] Im Allgemeinen können die Verfolgungssteuerregister herkömmliche Steuerstatusregister (CSR) sein, die die Verfolgungsanweisungsdaten empfangen und speichern. Hinsichtlich der breiteren Verfolgungsmultiplexer-Steuerungen kann in einigen Implementierungen die Verfolgung basierend auf den durch das System **100** ausgeführten CSR-Schreibvorgängen freigegeben und gesperrt werden. In einigen Implementierungen kann die Verfolgung durch das System **100** basierend auf dem Wert eines globalen Zeitzählers (GTC), basierend auf dem Wert eines beispielhaften Verfolgungsmarkierungsregisters in dem FPC **104** (oder dem Kern **116**) oder basierend auf dem Wert eines beispielhaften Verfolgungsmarkierungsregisters in dem SPC **106** dynamisch gestartet und gestoppt werden.

[0065] Zusätzliche Einzelheiten und Beschreibungen bezüglich der Systeme und Verfahren sowohl zum dynamischen Starten und Stoppen der Verfolgungsaktivität als auch für eine synchronisierte Hardware-Ereignis-Sammlung werden bezüglich der Implementierungen nach den **Fig. 5-7** ausführlicher beschrieben.

[0066] In einigen Implementierungen kann der FPC **104** für den Kern **116** Verfolgungssteuerparameter verwenden, um ein Verfolgungsfenster zu erzeugen, das der Ereignisaktivität zugeordnet ist, die innerhalb des Kerns **116** auftritt. Der Verfolgungssteuerparameter ermöglicht, dass das Verfolgungsfenster hinsichtlich sowohl unterer und oberer Grenzen für den GTC als auch unterer und oberer Grenzen für das Verfolgungsmarkierungsregister definiert wird.

[0067] In einigen Implementierungen kann das System **100** Funktionen, die die Verringerung der Anzahl der Verfolgungseinträge, die erzeugt werden, ermöglichen, wie z. B. Verfolgungsereignis-Filterungsmerkmale, enthalten. Der FPC **104** und der SPC **106** können z. B. jeder Filterungsmerkmale enthalten, die die Rate begrenzen, mit der jeder Kern ein Verfolgungsbit in einem (im Folgenden beschriebenen) beispielhaften erzeugten Verfolgungsdeskriptor setzt. Der HIB **114** kann ähnliche Filterungsmerkmale enthalten, wie z. B. einen beispielhaften DMA-Ratenbegrenzer, der die Verfolgungsbits begrenzt, die der Erfassung bestimmter DMA-Verfolgungsereignisse zugeordnet sind. Zusätzlich kann der HIB **114** Steuerungen (z. B. über ein Freigabebit) zum Begrenzen enthalten, welche Warteschlangen die DMA-Verfolgungseinträge hervorbringen.

[0068] In einigen Implementierungen kann ein Deskriptor für eine DMA-Operation ein Verfolgungsbit

aufweisen, das durch einen beispielhaften Kompilierer des Host-Systems 126 gesetzt wird. Wenn das Verfolgungsbit gesetzt ist, werden Hardware-Merkmale/Knöpfe, die die Verfolgungsdaten bestimmen und erzeugen, verwendet, um ein beispielhaftes Verfolgungsereignis abzuschließen. In einigen Fällen kann ein endgültiges Verfolgungsbit in der DMA eine logische ODER-Operation zwischen einem Verfolgungsbit, das durch den Kompilierer statisch eingefügt wird, und einem Verfolgungsbit, das durch eine spezielle Hardware-Komponente dynamisch bestimmt wird, sein. Folglich kann das vom Kompilierer erzeugte Verfolgungsbit in einigen Fällen einen Mechanismus abgesehen von der Filterung schaffen, um eine Gesamtmenge der Verfolgungsdaten, die erzeugt werden, zu verringern.

[0069] Ein Kompilierer des Host-Systems **126** kann z. B. entscheiden, nur die Verfolgungsbits für eine oder mehrere Fern-DMA-Operationen (z. B. einen DMA über wenigstens zwei Knoten) zu setzen und die Verfolgungsbits für eine oder mehrere lokale DMA-Operationen (z. B. einen DMA innerhalb eines speziellen Tensorknotens, wie z. B. des Knotens **200**) zu löschen. In dieser Weise kann eine Menge der Verfolgungsdaten, die erzeugt werden, anstatt basierend auf einer Verfolgungsaktivität, die sowohl knotenübergreifende als auch lokale DMA-Operationen enthält, basierend auf einer Verfolgungsaktivität, die auf knotenübergreifende DMA-Operationen (d. h., Fern-DMA-Operationen) eingeschränkt ist, verringert werden.

[0070] In einigen Implementierungen kann wenigstens ein durch das System **100** eingeleitetes Verfolgungsereignis einer Speicherzugriffsoperation zugeordnet sein, die mehrere Zwischenoperationen, die über das System **100** auftreten, enthält. Ein Deskriptor (z. B. ein MAE-Deskriptor) für die Speicherzugriffsoperation kann ein Verfolgungsbit enthalten, das veranlasst, dass die Daten, die mehreren Zwischenoperationen zugeordnet sind, in einem oder mehreren Speicherpuffern gespeichert werden. Folglich kann das Verfolgungsbit verwendet werden, um dazwischenliegende Speicheroperationen zu „kennzeichnen“ und mehrere Verfolgungsereignisse an dazwischenliegenden Sprüngen der DMA-Operation zu erzeugen, wenn die Datenpakete das System **100** durchqueren.

[0071] In einigen Implementierungen kann die ICI **112** einen Satz von Freigabebits und einen Satz von Paketfiltern enthalten, die eine Steuerfunktionalität für jeden Eintritts- und Austritts-Port einer speziellen Komponente des Knotens **200**, **201** bereitstellen. Diese Freigabebits und Paketfilter ermöglichen es der ICI **112**, die Verfolgungspunkte, die speziellen Komponenten der Knoten **200**, **201** zugeordnet sind, freizugeben und zu sperren. Zusätzlich zu dem Freigeben und Sperren der Verfolgungspunkte kann die ICI

112 konfiguriert sein, um die Verfolgungsdaten basierend auf der Ereignisquelle, dem Ereignisziel und dem Verfolgungsereignis-Pakettyp zu filtern.

[0072] In einigen Implementierungen kann zusätzlich zur Verwendung des GTC oder der Verfolgungsmarkierungen jedes Verfolgungssteuerregister für die Prozessorkerne **104**, **106** und den HIB **114** außerdem einen „Alles“-Verfolgungsmodus enthalten. Dieser „Alles“-Verfolgungsmodus kann die Verfolgung über einen gesamten zu steuernden Prozessor-Chip entweder durch den Verfolgungsmultiplexer **204** oder den Verfolgungsmultiplexer **206a** ermöglichen. Während sich die Verfolgungsmultiplexer **204** und **206a** im Alles-Verfolgungsmodus befinden, können sie ein „Im-Fenster“-Verfolgungssteuersignal senden, das spezifiziert, ob sich der spezielle Verfolgungsmultiplexer, entweder der Multiplexer **204** oder der Multiplexer **206a**, in einem Verfolgungsfenster befindet.

[0073] Das Im-Fenster-Verfolgungssteuersignal kann rundgesendet oder universell an alle anderen Verfolgungsmultiplexer, z. B. innerhalb eines Prozessor-Chips oder über mehrere Prozessor-Chips, übertragen werden. Die Rundsendung an die anderen Verfolgungsmultiplexer kann verursachen, dass alle Verfolgung freigegeben ist, wenn entweder der Multiplexer **204** oder der Multiplexer **206a** eine Verfolgungsaktivität ausführt. In einigen Implementierungen enthält jeder der Verfolgungsmultiplexer, die den Prozessorkernen **104**, **106** und dem HIB **114** zugeordnet sind, ein Verfolgungsfenster-Steuerregister, das spezifiziert, wann und/oder wie das „Alles-Verfolgungs“-Steuersignal erzeugt wird.

[0074] In einigen Implementierungen wird die Verfolgungsaktivität in den Verfolgungsmultiplexern **210a/b** und im Verfolgungsmultiplexer **212** im Allgemeinen darauf basierend freigegeben, ob in den Datenwörtern für die DMA-Operationen oder in den Steuernachrichten, die die ICI/den Daten-Router **112** durchqueren, ein Verfolgungsbit gesetzt ist. Die DMA-Operationen oder die Steuernachrichten können binäre Datenstrukturen mit fester Größe sein, die ein Verfolgungsbit innerhalb der binären Datenpakete aufweisen, das basierend auf bestimmten Umständen oder Software-Bedingungen gesetzt wird.

[0075] Wenn z. B. in dem FPC **104** (oder dem SPC **106**) eine DMA-Operation mit einer Verfolgungstyp-DMA-Anweisung eingeleitet wird und sich der Initiator (die Prozessorkerne **104** oder **106**) in einem Verfolgungsfenster befindet, wird das Verfolgungsbit in diesem speziellen DMA gesetzt. In einem weiteren Beispiel weisen für den FPC **104** die Steuernachrichten für die Datenschreibvorgänge zu einer weiteren Komponente innerhalb des Systems **100** das gesetzte Verfolgungsbit auf, falls sich der FPC **104** in einem Verfolgungsfenster befindet und ein Verfol-

gungspunkt, der veranlasst, dass die Verfolgungsdaten gespeichert werden, freigegeben ist.

[0076] In einigen Implementierungen stellen die DMA-Operationen mit einer Länge von null ein Beispiel einer breiteren DMA-Implementierung innerhalb des Systems 100 dar. Einige DMA-Operationen können z. B. eine Nicht-DMA-Aktivität innerhalb des Systems 100 erzeugen. Die Ausführung der Nicht-DMA-Aktivität kann außerdem verfolgt werden (z. B. Verfolgungsdaten erzeugen), als ob die Nicht-DMA-Aktivität eine DMA-Operation (z. B. eine DMA-Aktivität, die Operationen mit einer von null verschiedenen Länge enthält) wäre. Eine DMA-Operation, die z. B. an einem Quellort eingeleitet wird, aber ohne irgendwelche Daten (z. B. mit einer Länge von null), die zu senden oder zu übertragen sind, könnte stattdessen eine Steuernachricht an den Zielort senden. Die Steuernachricht gibt an, dass es am Ziel keine Daten gibt, die zu empfangen sind oder mit denen zu arbeiten ist, wobei die Steuernachricht selbst durch das System 100 verfolgt würde, wie eine DMA-Operation mit von null verschiedener Länge verfolgt würde.

[0077] In einigen Fällen können die DMA-Operationen mit einer Länge von null für den SPC 106 eine Steuernachricht erzeugen, wobei ein der Nachricht zugeordnetes Verfolgungsbit nur gesetzt wird, falls der DMA das Verfolgungsbit gesetzt aufweisen würde, d. h., die Steuernachricht aufweist, die keine Länge von null aufweist. Im Allgemeinen weisen die von dem Host-System 126 eingeleiteten DMA-Operationen das gesetzte Verfolgungsbit auf, falls sich der HIB 114 in einem Verfolgungsfenster befindet.

[0078] In der Implementierung nach Fig. 2 empfängt die Verfolgungskette 203 die Verfolgungseintragsdaten für die Komponententeilmenge, die auf den Knoten 0 ausgerichtet ist, während die Verfolgungskette 205 die Verfolgungseintragsdaten für die Komponententeilmenge empfängt, die auf den Knoten 1 ausgerichtet ist. Jede Verfolgungskette 203, 205, 207 sind verschiedene Datenkommunikationswege, die durch die jeweiligen Knoten 200, 201 und die ICI 112 verwendet werden, um die Verfolgungseintragsdaten einem beispielhaften Verfolgungseintragsdatenprotokoll 218 eines Chip-Managers 216 bereitzustellen. Folglich ist der Endpunkt der Verfolgungsketten 203, 205, 207 der Chip-Manager 216, wo die Verfolgungsereignisse in beispielhaften Speichereinheiten gespeichert sein können.

[0079] In einigen Implementierungen kann wenigstens eine Speichereinheit des Chip-Managers 216 128 Bits breit sein und kann eine Speichertiefe von wenigstens 20.000 Verfolgungseinträgen aufweisen. In alternativen Implementierungen kann wenigstens eine Speichereinheit eine größere oder kleinere Bitbreite aufweisen und kann eine Speichertiefe aufweisen, die mehr oder weniger Einträge speichern kann.

[0080] In einigen Implementierungen kann der Chip-Manager 216 wenigstens eine Verarbeitungsvorrichtung enthalten, die Anweisungen ausführt, um die empfangenen Verfolgungseintragsdaten zu managen. Der Chip-Manager 216 kann z. B. Anweisungen ausführen, um die Zeitstempeldaten für die jeweiligen Hardware-Ereignisse der über die Verfolgungsketten 203, 205, 207 empfangenen Verfolgungsdaten abzutasten/zu analysieren. Basierend auf der Analyse kann der Chip-Manager 216 das Verfolgungseintragsprotokoll 218 befüllen, so dass es Daten enthält, die verwendet werden können, um eine zeitlich geordnete Folge von Hardware-Verfolgungsereignissen zu identifizieren (oder zu erzeugen). Die Hardware-Verfolgungsereignisse können der Bewegung der Datenpakete entsprechen, die auf der Komponenten- und Unterkomponentenebene auftritt, wenn die Verarbeitungseinheiten des Systems 100 ein beispielhaftes verteiltes Software-Programm ausführen.

[0081] In einigen Implementierungen können die Hardware-Einheiten des Systems 100 Verfolgungseinträge (und entsprechende Zeitstempel) erzeugen, die einen beispielhaften Hardware-Verfolgungspuffer in einer nicht zeitlich geordneten Weise (d. h., in Unordnung) bevölkern. Der Chip-Manager 216 kann z. B. veranlassen, dass mehrere Verfolgungseinträge, die erzeugte Zeitstempel aufweisen, in das Eintragsprotokoll 218 eingefügt werden. Die jeweiligen Verfolgungseinträge der mehreren eingefügten Verfolgungseinträge können in Bezug aufeinander nicht zeitlich geordnet sein. In dieser Implementierung können die nicht zeitlich geordneten Verfolgungseinträge durch einen beispielhaften Host-Puffer des Host-Systems 126 empfangen werden. Beim Empfang durch den Host-Puffer kann das Host-System 126 Anweisungen bezüglich der Leistungs-Analyse/Überwachungs-Software ausführen, um die Zeitstempeldaten für die jeweiligen Verfolgungseinträge abzutasten/zu analysieren. Die ausgeführten Anweisungen können verwendet werden, um die Verfolgungseinträge zu sortieren und/oder um eine Zeitachse der Hardware-Verfolgungsereignisse zu konstruieren/zu erzeugen.

[0082] In einigen Implementierungen können die Verfolgungseinträge während einer Verfolgungssitzung über eine Host-DMA-Operation aus dem Eintragsprotokoll 218 entfernt werden. In einigen Fällen kann das Host-System 126 DMA-Einträge nicht so schnell aus dem Verfolgungseintragsprotokoll 218 entfernen, wie sie zu dem Protokoll hinzugefügt werden. In anderen Implementierungen kann das Eintragsprotokoll 218 eine vorgegebene Speichertiefe enthalten. Falls die Speichertiefengrenze des Eintragsprotokolls 218 erreicht wird, können zusätzliche Verfolgungseinträge verloren werden. Um zu steuern, welche Verfolgungseinträge verloren werden, kann das Eintragsprotokoll 218 in einem First-in-first-

out-Modus (FIFO-Modus) oder alternativ in einem Überschreib-Aufzeichnungsmodus arbeiten.

[0083] In einigen Implementierungen kann der Überschreib-Aufzeichnungsmodus durch das System **100** verwendet werden, um die Leistungsanalyse zu unterstützen, die einer Postmortem-Fehlersuche zugeordnet ist. Der Programmcode kann z. B. während eines bestimmten Zeitraums ausgeführt werden, wobei die Verfolgungsaktivität freigegeben ist und der Überschreib-Aufzeichnungsmodus freigegeben ist. In Reaktion auf ein Postmortem-Software-Ereignis (z. B. einen Programmabsturz) innerhalb des Systems **100** kann die durch das Host-System **126** ausgeführte Überwachungs-Software die Dateninhalte eines beispielhaften Hardware-Verfolgungspuffers analysieren, um Einblick in die Hardware-Ereignisse zu erlangen, die vor dem Programmabsturz aufgetreten sind. Die Postmortem-Fehlersuche, wie sie in dieser Beschreibung verwendet wird, bezieht sich auf die Analyse oder die Fehlersuche des Programmcodes, nachdem der Code abgestürzt ist oder im Allgemeinen versagt hat, so ausgeführt zu werden/zu arbeiten, wie vorgesehen ist.

[0084] Falls im FIFO-Modus das Eintragsprotokoll **218** voll ist und falls das Host-System **126** die gespeicherten Protokolleinträge innerhalb eines bestimmten Zeitrahmens entfernt, um Speicherbetriebsmittel einzusparen, können keine neuen Verfolgungseinträge in einer Speichereinheit des Chip-Managers **216** gespeichert werden. Während des Überschreib-Aufzeichnungsmodus können, falls das Eintragsprotokoll **218** voll ist, weil das Host-System **126** innerhalb eines bestimmten Zeitrahmens gespeicherte Protokolleinträge entfernt, um Speicherbetriebsmittel einzusparen, die neuen Verfolgungseinträge den ältesten Verfolgungseintrag, der innerhalb des Eintragsprotokolls **218** gespeichert ist, überschreiben. In einigen Implementierungen werden die Verfolgungseinträge in Reaktion auf eine DMA-Operation unter Verwendung der Verarbeitungsmerkmale des HIB **114** zu einem Speicher des Host-Systems **126** bewegt.

[0085] Ein Verfolgungspunkt, wie er in dieser Beschreibung verwendet wird, ist der Generator eines Verfolgungseintrags und der Daten, die dem Verfolgungseintrag zugeordnet sind, der durch den Chip-Manager **216** empfangen und in dem Verfolgungseintragsprotokoll **218** gespeichert wird. In einigen Implementierungen kann ein Mehrkern-Mehrnoten-Prozessor-Mikrochip drei Verfolgungsketten innerhalb des Chips enthalten, so dass eine erste Verfolgungskette die Verfolgungseinträge von einem Chip-Knoten **0** empfängt, eine zweite Verfolgungskette die Verfolgungseinträge von einem Chip-Knoten **1** empfängt und eine dritte Verfolgungskette die Verfolgungseinträge von einem ICI-Router des Chips empfängt.

[0086] Jeder Verfolgungspunkt weist eine eindeutige Verfolgungsidentifikationsnummer innerhalb seiner Verfolgungskette auf, die er in den Kopf jedes Verfolgungseintrags einfügt. In einigen Implementierungen identifiziert jeder Verfolgungseintrag die Verfolgungskette, von der er ausgegangen ist, in einem Kopf, der durch ein oder mehrere Bytes/Bits des Datenworts angegeben ist. Jeder Verfolgungseintrag kann z. B. eine Datenstruktur enthalten, die definierte Feldformate (z. B. Kopf, Nutzdaten usw.) aufweist, die die Informationen über ein spezielles Verfolgungsereignis befördern. Jedes Feld in einem Verfolgungseintrag entspricht nützlichen Daten, die auf den Verfolgungspunkt anwendbar sind, der den Verfolgungseintrag erzeugt hat.

[0087] Wie oben angegeben worden ist, kann jeder Verfolgungseintrag in eine Speichereinheit des Chip-Managers **216**, der dem Verfolgungseintragsprotokoll **218** zugeordnet ist, geschrieben oder innerhalb einer Speichereinheit des Chip-Managers **216**, der dem Verfolgungseintragsprotokoll **218** zugeordnet ist, gespeichert werden. In einigen Implementierungen können die Verfolgungspunkte einzeln freigegeben oder gesperrt werden und können mehrere Verfolgungspunkte den gleichen Typ von Verfolgungseintrag, obgleich mit unterschiedlichen Verfolgungspunktkennungen, erzeugen.

[0088] In einigen Implementierungen kann jeder Verfolgungseintragstyp einen Verfolgungsnamen, eine Verfolgungsbeschreibung und einen Kopf, der die Codierungen für spezielle Felder und/oder eine Sammlung von Feldern innerhalb des Verfolgungseintrags identifiziert, enthalten. Der Name, die Beschreibung und der Kopf stellen gemeinsam eine Beschreibung dessen bereit, was der Verfolgungseintrag repräsentiert. Aus der Perspektive des Chip-Managers **216** kann diese Beschreibung außerdem die spezielle Verfolgungskette **203**, **205**, **207** identifizieren, in der ein spezifischer Verfolgungseintrag in einen speziellen Prozessor-Chip hereinkam. Folglich repräsentieren die Felder innerhalb eines Verfolgungseintrags die Stücke der Daten (z. B. in Bytes/Bits), die für die Beschreibung relevant sind, wobei sie eine Verfolgungseintragskennung sein können, die verwendet wird, um zu bestimmen, welcher Verfolgungspunkt einen speziellen Verfolgungseintrag erzeugt hat.

[0089] In einigen Implementierungen können die Verfolgungseintragsdaten, die einem oder mehreren der gespeicherten Hardware-Ereignissen zugeordnet sind, teilweise den Datenkommunikationen entsprechen, die auftreten: a) zwischen wenigstens einem Knoten **0** und einem Knoten **1**; b) zwischen wenigstens Komponenten innerhalb des Knotens **0**; und c) zwischen wenigstens Komponenten innerhalb des Knotens **1**. Die gespeicherten Hardware-Ereignisse können z. B. teilweise den Datenkommunikationen

entsprechen, die zwischen wenigstens einem des Folgenden auftreten: 1) dem FPC **104** des Knotens **0** und dem FPC **104** des Knotens **1**; dem FPC **104** des Knotens **0** und dem SPC **106** des Knotens **0**; 2) dem SPC **106** des Knotens **1** und dem SPC **106** des Knotens **1**.

[0090] Fig. 3 veranschaulicht einen Blockschaltplan einer beispielhaften Verfolgungsmultiplexer-Konfigurationsarchitektur 300 und eine beispielhafte Datenstruktur 320. Die Konfiguration **300** des Verfolgungsmultiplexers enthält im Allgemeinen einen Verfolgungsbuseingang **302**, einen Bus-Arbitrator **304** und einen lokalen Verfolgungspunkt-Arbitrator **306**, einen Bus-FIFO **308**, wenigstens eine lokale Verfolgungsereignis-Warteschlange **310**, einen gemeinsam benutzten Verfolgungsereignis-FIFO **312** und einen Verfolgungsbusausgang **314**.

[0091] Die Multiplexerkonfiguration **300** entspricht einem beispielhaften Verfolgungsmultiplexer, der innerhalb einer Komponente des Systems **100** angeordnet ist. Die Multiplexerkonfiguration **300** kann die folgende Funktionalität enthalten. Der Buseingang **302** kann sich auf lokale Verfolgungspunkt-daten beziehen, die bis zu einem derartigen Zeitpunkt, zu dem die Arbitrierungslogik (z. B. der Arbitrator **304**) veranlassen kann, dass die Verfolgungsdaten in einer beispielhaften Verfolgungskette angeordnet werden, vorübergehend innerhalb des Bus-FIFO **308** gespeichert sind. Ein oder mehrere Verfolgungspunkte für eine Komponente können Verfolgungsereignisdaten in wenigstens eine lokale Verfolgungsereignis-Warteschlange **310** einfügen. Der Arbitrator **306** schafft eine Arbitrierung einer ersten Ebene und ermöglicht die Auswahl von Ereignissen aus den innerhalb der Warteschlange **310** gespeicherten lokalen Verfolgungsereignissen. Die ausgewählten Ereignisse werden in dem gemeinsam benutzten Verfolgungsereignis-FIFO **312** angeordnet, der außerdem als eine Speicherwarteschlange arbeitet.

[0092] Der Arbitrator **304** schafft eine Arbitrierung einer zweiten Ebene, die lokale Verfolgungsereignisse von der FIFO-Warteschlange **312** empfängt und die lokalen Verfolgungsereignisse auf einer speziellen Verfolgungskette **203**, **205**, **207** über den Verfolgungsbusausgang **314** verschmilzt. In einigen Implementierungen können die Verfolgungseinträge schneller in die lokalen Warteschlangen **310** geschoben werden, als sie in dem gemeinsam benutzten FIFO **312** verschmolzen werden können, oder alternativ können die Verfolgungseinträge schneller in den gemeinsam benutzten FIFO **312** geschoben werden, als sie auf dem Verfolgungsbus **314** verschmolzen werden können. Wenn diese Szenarios auftreten, werden die jeweiligen Warteschlangen **310** und **312** voll mit Verfolgungsdaten.

[0093] Wenn entweder die Warteschlange **310** oder **312** voll mit Verfolgungsdaten wird, kann das System **100** in einigen Implementierungen konfiguriert sein, so dass die neuesten Verfolgungseinträge fallen gelassen werden und nicht in einer speziellen Warteschlange gespeichert oder in einer speziellen Warteschlange verschmolzen werden. In anderen Implementierungen kann das System **100** konfiguriert sein, um eine beispielhafte Verarbeitungspipeline anzuhalten, bis die Warteschlangen, die gefüllt sind, abermals verfügbaren Warteschlangenraum aufweisen, um Einträge zu empfangen, anstatt die Verfolgungseinträge fallen zu lassen, wenn sich bestimmte Warteschlangen (z. B. die Warteschlangen **310**, **312**) füllen.

[0094] Eine Verarbeitungspipeline, die die Warteschlangen **310**, **312** verwendet, kann z. B. angehalten werden, bis eine ausreichende oder Schwellenanzahl von Verfolgungseinträgen auf dem Verfolgungsbus **314** verschmolzen ist. Die ausreichende oder Schwellenanzahl kann einer speziellen Anzahl verschmolzener Verfolgungseinträge entsprechen, die zu einem verfügbaren Warteschlangenraum führen, damit ein oder mehrere Verfolgungseinträge durch die Warteschlangen **310**, **312** empfangen werden. Die Implementierungen, in denen die Verarbeitungspipelines angehalten werden, bis stromabwärts gelegener Warteschlangenraum verfügbar wird, können basierend auf bestimmten Verfolgungseinträgen, die beibehalten werden, anstatt dass sie fallengelassen werden, Verfolgungsdaten in höherer Wiedergabetreue bereitstellen.

[0095] In einigen Implementierungen sind die lokalen Verfolgungswarteschlangen so breit, wie durch den Verfolgungseintrag erforderlich ist, so dass jeder Verfolgungseintrag nur eine Stelle in der lokalen Warteschlange **310** einnimmt. Die gemeinsam benutzte Verfolgungs-FIFO-Warteschlange **312** kann jedoch eine eindeutige Verfolgungseintrags-Zeilencodierung verwenden, so dass einige Verfolgungseinträge zwei Orte in der gemeinsam benutzten Warteschlange **312** belegen können. Wenn in einigen Implementierungen irgendwelche Daten eines Verfolgungspakets fallengelassen werden, wird das volle Paket fallengelassen, so dass keine Teilpakete in dem Verfolgungseintragsprotokoll **218** erscheinen.

[0096] Im Allgemeinen ist eine Verfolgung eine Zeitachse der Aktivitäten oder Hardware-Ereignisse, die einer speziellen Komponente des Systems **100** zugeordnet sind. Ungleich der (im Folgenden beschriebenen) Leistungszähler, die Daten ansammeln, enthalten die Verfolgungen ausführliche Ereignisdaten, die einen Einblick in die Hardware-Aktivität bereitstellen, die während eines spezifizierten Verfolgungsfensters auftritt. Das beschriebene Hardware-System ermöglicht eine ausgedehnte Unterstützung für die verteilte Hardware-Verfolgung einschließlich der Erzeu-

gung von Verfolgungseinträgen, der vorübergehenden Speicherung der Verfolgungseinträge in einem hardware-gemanagten Puffer, des statischen und dynamischen Freigebens eines oder mehrerer Verfolgungstypen und des Streamings der Verfolgungseintragsdaten zu dem Host-System **126**.

[0097] In einigen Implementierungen können die Verfolgungen für die Hardware-Ereignisse, die durch die Komponenten des Systems **100** ausgeführt werden, wie z. B. das Erzeugen einer DMA-Operation, das Ausführen einer DMA-Operation, das Ausgeben/ die Ausführung bestimmter Anweisungen oder das Aktualisieren von Synchronisationsmerkern, erzeugt werden. In einigen Fällen kann die Verfolgungsaktivität verwendet werden, um die DMAs durch das System zu verfolgen oder um die in einem speziellen Prozessorkern ausgeführten Anweisungen zu verfolgen.

[0098] Das System **100** kann konfiguriert sein, um wenigstens eine Datenstruktur **320** zu erzeugen, die ein oder mehrere Hardware-Ereignisse **322**, **324** aus einer Zeitachse der Hardware-Ereignisse identifiziert. In einigen Implementierungen ordnet die Datenstruktur **320** ein oder mehrere Hardware-Ereignisse **322**, **324** in einer zeitlich geordneten Folge von Ereignissen an, die wenigstens dem FPC **104** und dem SPC **106** zugeordnet sind. In einigen Fällen kann das System **100** die Datenstruktur **320** in einer Speicherbank einer Host-Steuervorrichtung des Host-Systems **126** speichern. Die Datenstruktur **320** kann verwendet werden, um die Leistung des Programmcodes zu bewerten, der wenigstens durch die Prozessorkerne **104** und **106** ausgeführt wird.

[0099] Wie durch die Hardware-Ereignisse **324** gezeigt ist, kann in einigen Implementierungen eine spezielle Verfolgungsidentifikationsnummer (Verfolgungs-ID-Nummer) (z. B. die Verfolgungsbits-ID '003) mehreren Hardware-Ereignissen zugeordnet sein, die über die verteilten Prozessoreinheiten auftreten. Die mehreren Hardware-Ereignisse können einer speziellen Speicherzugriffsoperation (z. B. einem DMA) entsprechen, wobei die spezielle Verfolgungs-ID-Nummer verwendet wird, um ein oder mehrere Hardware-Ereignisse zu korrelieren.

[0100] Wie z. B. durch das Ereignis **324** angegeben ist, kann eine einzige Verfolgungs-ID für eine DMA-Operation mehrere Zeitschritte enthalten, die mehreren verschiedenen Punkten in dem DMA entsprechen. In einigen Fällen kann die Verfolgungs-ID '003 ein „ausgegebenes“ Ereignis, ein „ausgeführtes“ Ereignis und ein „abgeschlossenes“ Ereignis aufweisen, die als in Bezug aufeinander etwas zeitlich getrennt identifiziert sind. Folglich kann in dieser Hinsicht die Verfolgungs-ID ferner zum Bestimmen eines Latenzzeitattributs der Speicherzugriffsoperation basierend auf der Korrelation und unter Bezugnahme auf die Zeitschritte verwendet werden.

[0101] In einigen Implementierungen kann das Erzeugen der Datenstruktur **320** z. B. enthalten, dass das System **100** die Ereigniszeitstempel der jeweiligen Ereignisse in einer ersten Teilmenge von Hardware-Ereignissen mit den Ereigniszeitstempeln der jeweiligen Ereignisse in einer zweiten Teilmenge von Hardware-Ereignissen vergleicht. Das Erzeugen der Datenstruktur **320** kann ferner enthalten, dass das System **100** für die Darstellung in der Datenstruktur einen korrelierten Satz von Hardware-Ereignissen teilweise basierend auf dem Vergleich zwischen der ersten Teilmenge von Ereignissen und der zweiten Teilmenge von Ereignissen bereitstellt.

[0102] Wie in **Fig. 3** gezeigt ist, kann die Datenstruktur **320** wenigstens einen Parameter identifizieren, der ein Latenzzeitattribut eines speziellen Hardware-Ereignisses **322**, **324** angibt. Das Latenzzeitattribut kann wenigstens eine Dauer des speziellen Hardware-Ereignisses angeben. In einigen Implementierungen wird die Datenstruktur **320** durch Software-Anweisungen erzeugt, die durch eine Steuervorrichtung des Host-Systems **126** ausgeführt werden. In einigen Fällen kann die Struktur **320** in Reaktion auf die Steuervorrichtung erzeugt werden, die die Verfolgungseintragsdaten in einer Speicherplatte/-einheit des Host-Systems **126** speichert.

[0103] **Fig. 4** ist ein Blockschaltplan **400**, der eine beispielhafte Verfolgungsaktivität für ein durch das System **100** ausgeführtes Speicherdirektzugriffs-Verfolgungsereignis (DMA-Verfolgungsereignis) angibt. Für die DMA-Verfolgung können sich die Daten für eine beispielhafte DMA-Operation, die von einem ersten Prozessorknoten zu einem zweiten Prozessorknoten ausgeht, über die ICI **112** bewegen und können unmittelbare ICI-/Router-Sprünge entlang dem Datenweg erzeugen. Die DMA-Operation erzeugt Verfolgungseinträge an jedem Knoten innerhalb eines Prozessor-Chips und entlang jedes Sprungs, wenn die DMA-Operation die ICI **112** durchquert. Die Informationen werden durch jeden dieser erzeugten Verfolgungseinträge erfasst, um einen zeitlichen Fortschritt der DMA-Operationen entlang den Knoten und Sprüngen zu rekonstruieren.

[0104] Eine beispielhafte DMA-Operation kann den Prozessschritten zugeordnet sein, die in der Implementierung nach **Fig. 4** dargestellt sind. Für diese Operationen überträgt ein lokaler DMA Daten von einem virtuellen Speicher **402** (vmem 402), der wenigstens einem der Prozessorkerne **104**, **106** zugeordnet ist, zu dem HBM **108**. Die in der graphischen Darstellung **400** dargestellte Nummerierung entspricht den Schritten der Tabelle **404** und repräsentiert im Allgemeinen die Aktivitäten in einer Knotenstruktur **110** oder die durch die Knotenstruktur **110** eingeleiteten Aktivitäten.

[0105] Die Schritte der Tabelle 404 beschreiben im Allgemeinen die zugeordneten Verfolgungspunkte. Der beispielhafte Betrieb erzeugt für diesen DMA sechs Verfolgungseinträge. Der Schritt eins enthält die anfängliche DMA-Anforderung von dem Prozessorkern an die Knotenstruktur **110**, die einen Verfolgungspunkt in der Knotenstruktur erzeugt. Der Schritt zwei enthält einen Lesebefehl, in dem die Knotenstruktur **110** den Prozessorkern auffordert, die Daten zu übertragen, was einen weiteren Verfolgungspunkt in der Knotenstruktur **110** erzeugt. Die beispielhafte Operation weist keinen Verfolgungseintrag für den Schritt drei auf, wenn der vmem 402 ein Lesen der Knotenstruktur **110** abschließt.

[0106] Der Schritt vier enthält, dass die Knotenstruktur **110** eine Lesebetriebsmittelaktualisierung ausführt, um eine Synchronisationsmerkeraktualisierung in dem Prozessorkern zu verursachen, was einen Verfolgungspunkt in dem Prozessorkern erzeugt. Der Schritt fünf enthält einen Schreibbefehl, bei dem die Knotenstruktur **110** den Speichermultiplexer **108** von den bevorstehenden Daten, die in den HBM zu schreiben sind, benachrichtigt. Die Benachrichtigung über den Schreibbefehl erzeugt einen Verfolgungspunkt in der Knotenstruktur **110**, während der Schritt sechs, der Abschluss des Schreibens in den HBM, außerdem einen Verfolgungspunkt in der Knotenstruktur **110** erzeugt. Im Schritt sieben führt die Knotenstruktur **110** eine Schreibbetriebsmittelaktualisierung aus, um eine Synchronisationsmerkeraktualisierung in dem Prozessorkern zu verursachen, was einen Verfolgungspunkt in dem Prozessorkern (z. B. im FPC **104**) erzeugt. Zusätzlich zu der Schreibbetriebsmittelaktualisierung kann die Knotenstruktur **110** eine Quittungsaktualisierung („ack“-Aktualisierung) ausführen, bei der der Datenabschluss für die DMA-Operation zurück zu dem Prozessorkern signalisiert wird. Die ack-Aktualisierung kann Verfolgungseinträge erzeugen, die zu den durch die Schreibbetriebsmittelaktualisierung erzeugten Verfolgungseinträgen ähnlich sind.

[0107] In einer weiteren beispielhaften DMA-Operation wird ein erster Verfolgungseintrag erzeugt, wenn in einer Knotenstruktur **110** eines Ursprungsknotens eine DMA-Anweisung ausgegeben wird. In der Knotenstruktur **110** können zusätzliche Verfolgungseinträge erzeugt werden, um den Zeitraum zu erfassen, der verwendet wird, um die Daten für den DMA zu lesen und die Daten in die abgehenden Warteschlangen zu schreiben. In einigen Implementierungen kann die Knotenstruktur **110** die DMA-Daten in kleinere Blöcke der Daten pakettieren. Für die in kleinere Blöcke pakettierten Daten können die Lese- und Schreibverfolgungseinträge für einen ersten Datenblock und einen letzten Datenblock erzeugt werden. Optional kann festgelegt sein, dass zusätzlich zu dem ersten und dem letzten Datenblock alle Datenblöcke Verfolgungseinträge erzeugen.

[0108] Für Fern-/nicht lokale DMA-Operationen, die ICI-Sprünge erfordern können, können der erste und der letzte Datenblock zusätzliche Verfolgungseinträge an den Eintritts- und Austrittspunkten in jedem Zwischensprung entlang der I-CI/des Routers **112** erzeugen. Wenn die DMA-Daten an einem Zielknoten ankommen, werden Verfolgungseinträge, die zu den vorhergehenden Einträgen der Knotenstruktur **110** ähnlich sind, (z. B. das Lesen/Schreiben des ersten und des letzten Datenblocks) an dem Zielknoten erzeugt. In einigen Implementierungen kann ein letzter Schritt der DMA-Operation ausgeführte Anweisungen enthalten, die dem DMA zugeordnet sind, die eine Aktualisierung eines Synchronisationsmerkers an dem Zielknoten verursachen. Wenn der Synchronisationsmerker aktualisiert wird, kann ein Verfolgungseintrag erzeugt werden, der den Abschluss der DMA-Operation angibt.

[0109] In einigen Implementierungen wird die DMA-Verfolgung durch den FPC **104**, den SPC **106** oder den HIB **114** eingeleitet, wenn sich jede Komponente in dem Verfolgungsmodus befindet, so dass die Verfolgungspunkte ausgeführt werden können. Die Komponenten des Systems **100** können basierend auf globalen Steuerungen in dem FPC **104** oder dem SPC **106** über einen Auslösemechanismus in den Verfolgungsmodus eintreten. Die Verfolgungspunkte lösen in Reaktion auf das Auftreten einer spezifischen Handlung oder Bedingung, die der Ausführung des Programmcodes durch die Komponenten des Systems **100** zugeordnet ist, aus. Die Abschnitte des Programmcodes können z. B. eingebettete Auslösefunktionen enthalten, die durch wenigstens eine Hardware-Komponente des Systems **100** detektierbar sind.

[0110] Die Komponenten des Systems **100** können konfiguriert sein, um eine Auslösefunktion zu detektieren, die den Abschnitten des Programmcodes zugeordnet ist, der durch wenigstens einen des FPC **104** oder des SPC **106** ausgeführt wird. In einigen Fällen kann die Auslösefunktion wenigstens einem des Folgenden entsprechen: 1) einem speziellen Ablaufschritt in einem Abschnitt oder Modul des ausgeführten Programmcodes; oder 2) einem speziellen Zeitparameter, der durch den GTC angegeben wird, der durch die verteilten Prozessoreinheiten des Systems **100** verwendet wird.

[0111] In Reaktion auf das Detektieren der Auslösefunktion kann eine spezielle Komponente des Systems **100** wenigstens einen Verfolgungspunkt (z. B. ein Verfolgungsereignis) einleiten, auslösen oder ausführen, der verursacht, dass die Verfolgungseintragsdaten, die einem oder mehreren Hardware-Ereignissen zugeordnet sind, in wenigstens einem Speicherpuffer der Hardware-Komponente gespeichert werden. Wie oben angegeben worden ist, können dann die gespeicherten Verfolgungsdaten

dem Chip-Manager **216** über wenigstens eine Verfolgungskette **203**, **205**, **207** bereitgestellt werden.

[0112] Wie oben angegeben worden ist, haben die **Fig. 1-4** die Herangehensweisen für eine Hardware-Ereignis-Verfolgung in einem verteilten Computersystem veranschaulicht. Die verbleibenden **Fig. 5-7** beschreiben die Herangehensweisen zur synchronisierten Hardware-Ereignis-Sammlung in einem verteilten Computersystem. Die synchronisierte Sammlung von Hardware-Ereignissen kann wenigstens teilweise auf einem oder mehreren Auslösemechanismen basieren.

[0113] **Fig. 5** veranschaulicht eine beispielhafte Datenstruktur für einen globalen Zeitzähler (GTC) **502** und die Tabellen 504 und 506, die die Verwendungsszenarios des GTC **502** durch ein beispielhaftes Computersystem für die verteilte Hardware-Verfolgung (z. B. das System **100**) angeben. In der Implementierung nach **Fig. 5** ist der GTC **502** ein 64-Bit-Wert, der einen 60-Bit-Zähler und einen 4-Bit-Versatz enthält. In alternativen Implementierungen kann der GTC **502** ein Stück binärer Daten mit fester Größe sein, das eine Größe in Bits aufweist die von weniger als 64 Bits bis mehr als 64 Bits reicht.

[0114] Die dem GTC **502** zugeordnete Datenstruktur wendet sowohl einen globalen Master-Zähler als auch lokale Zeitzähler (LTCs), die im Folgenden bezüglich der Implementierung nach **Fig. 6** beschrieben werden, an. Die oberen 60 Bits des GTC **502** entsprechen einem tatsächlichen Zeitzähler, der um eins pro Zyklus inkrementiert wird, mit Ausnahme wenn er „einholt“ oder „eingeholt wird“, um die (im Folgenden beschriebenen) GTC-Phasenvariationen zu behandeln. Während einer Aktualisierung eines lokalen GTC **502** (d. h., eines LTC) ist es z. B. möglich, dass der lokale GTC **502** während mehrerer Zyklen nicht „tickt“, um von einem globalen Master-GTC **502** „eingeholt“ zu werden. In diesen Szenarios werden die unteren vier „Versatz“-Bits verwendet, um die GTC-Phasenvariationen zu kompensieren, die während der Programmcodeausführung auftreten können.

[0115] Wenn die GTC-Phasenvariationen behandelt werden, können die letzten vier Bits unter diesen Umständen inkrementiert werden, um teilweise die Unterscheidung zwischen den Zeitstempeln für die lokalen Hardware-Ereignisaktivitäten aufrechtzuerhalten, deren GTCs andernfalls völlig gleich sein würden. In den meisten anderen Fällen ist der 4-Bit-Versatz des GTC **502** frei und folglich unbenutzt. In einigen Implementierungen können die Versatzbits bis 15 hochzählen und dann stoppen. Während in alternativen Implementierungen die Versatzbits bis zu irgendeinem ganzzahligen Wert zählen können, wie es in einer speziellen Zählerkonfiguration benötigt wird. In einigen Fällen werden die Versatzbits gelöscht, wenn

die oberen 60 Bits des GTC **502** abermals das Ticken beginnen.

[0116] Bezüglich der Konfiguration und des Managements des (im Folgenden beschriebenen) GTC **502**, **602** kann in einigen Implementierungen der ausgeführte Programmcode einen GTC durch das Konfigurieren eines oder mehrerer Prozessor-Chips des Systems **100** konfigurieren oder aufbauen. Das Konfigurieren der Prozessor-Chips kann sowohl das Bestimmen eines globalen Master-GTC für das System als auch das Bestimmen lokaler Master-GTCs (d. h., LTCs) für die jeweiligen Knoten oder Komponenten des Systems **100** enthalten.

[0117] Die Komponenten können Hardware-Knöpfe/Merkmale enthalten, die der ausgeführte Programmcode/die ausgeführte Software verwendet, um die Taktlatenzzeiten zu berechnen und zu kompensieren und um eine manuelle Abstimmung eines oder mehrerer lokaler GTCs (d. h., der LTCs nach **Fig. 6**) auszuführen. Die LTC-Kompensation für die Taktlatenzzeiten und die manuelle Abstimmung werden bezüglich des Master-GTC ausgeführt, um die Phasenvariationen zu minimieren oder abzuschwächen. In einigen Implementierungen kann das System **100** einen Autopilotmodus für die Latenzzeitkompensation enthalten, bei dem die Komponenten-Hardware-Merkmale die Verbindungslatenzzeiten automatisch über einen „Ping-und-Echo“-Mechanismus bestimmen.

[0118] In einigen Implementierungen kann das System **100** einen Verbindungs-/Software-Stapel enthalten, der einen Abschnitt oder eine Teilmenge des Programmcodes enthält, der verwendet wird, um die Parameterwerte eines beispielhaften Steuer-/Statusregisters (CSR) einzustellen. In einigen Fällen verursachen die in Reaktion auf die Ausführung des Verbindungsstapel-Programmcodes erzeugten Steuersignale, dass ein oder mehrere Parameterwerte in das CSR geschrieben werden. Diese CSR-Parameterwerte können verwendet werden, um die verschiedenen Aspekte des GTC **502**, **602** oder der LTCs zu konfigurieren.

[0119] Der eine oder die mehreren Parameter können z. B. enthalten: 1) einen GTC_config-Parameter, der die Auswahl eines globalen Master-GTC, eines lokalen Master-GTC oder eines lokalen Slave-GTC und die Auswahl einer Aktualisierungsprobengröße verursacht; 2) einen GTC_sync-Parameter, der verwendet wird, um einen Master-GTC rückzusetzen oder eine „Ping“-Messung zu löschen; 3) einen GTC_latency_compensation_control-Parameter, der die manuelle Latenzzeitkompensationen konfiguriert; 4) einen GTC_link_ping-Parameter, der eine minimale Ping-Latenzzeit angibt; 5) einen GTC_count-Parameter, der einen tatsächlichen GTC-Wert angibt; und/oder 6) einen GTC_max_diff-Parameter, der ei-

nen maximalen beobachteten Unterschied zwischen einem lokalen GTC und dem Master-GTC angibt.

[0120] Wie oben angegeben worden ist, können ein globaler Master-GTC **502** und ein oder mehrere lokale GTCs **502** durch das System **100** verwendet werden, um Ereignis-Zeitstempel zu erhalten und um eine sequentiell geordnete Anordnung unter den Hardware-Ereignissen, die innerhalb des Systems auftreten, zu bestimmen. Zusätzlich zu den Aktivitäten des Ordners kann der GTC **502** verwendet werden, um die Latenzzeiten der Aktivitäten zu bestimmen. In einigen Implementierungen kann ein spezieller GTC **502** Phasenvariationen zeigen, wobei folglich die bestimmten Latenzzeiten um ein oder mehrere Zyklen ungenau sein können.

[0121] Wie oben angegeben worden ist, kann die Rate, mit der ein bestimmter GTC **502** inkrementiert wird, in Abhängigkeit davon variieren, ob der lokale GTC von dem globalen Master verschieden ist, d. h., wenn es Phasenvariationen gibt. In einem stationären Zustand, in dem es keine Phasenvariationen gibt, wird ein globaler Master-GTC **502** einmal pro Zyklus inkrementiert, wobei er genau so als ein LTC spezieller Komponenten verwendet werden kann, um die Latenzzeiten und die Reihenfolge der lokalen Aktivitäten (z. B. einer bestimmten Komponente) zu zählen.

[0122] In einigen Implementierungen kann ein Master-GTC **502** langsamer als ein bestimmter LTC zählen. In diesen Fällen werden die Versatzbits des GTC **502** weiterhin einmal pro lokalen Taktzyklus inkrementiert. Wie durch die Tabellen 504, 506 angegeben ist, kann in Anbetracht der eindeutigen Funktionalität der GTC-Versatzbits ein erweiterter GTC („GTC-ext“) für die Verwendung durch das System **100** definiert werden. In einigen Implementierungen kann der GTC-ext durch das Hinzufügen der oberen 60 Bits (GTC:obere) und der unteren vier Versatzbits (GTC:Versatz) konstruiert werden, während in anderen Implementierungen der Programmcode, der in einem speziellen Prozessor ausgeführt wird, den GTC-ext gleich dem GTC:obere, der mit einem Teilwert des GTC:Versatz summiert wird, berechnen kann.

[0123] In einigen Implementierungen und in Reaktion auf das Inkrementieren der Versatzbits kann ein Parameter des GTC-ext verwendet werden, um sowohl eine Ereignisordnung als auch die Latenzzeiten der lokalen Operationen zu bestimmen. Die Tabellen 504, 506 geben Verwendungsszenarios an, in denen sich die Parameter des GTC-ext als nützlich erweisen können, wenn die Hardware-Ereignisordnung und die Latenzzeit der Ereignisoperationen bestimmt werden.

[0124] Die Tabelle 504 beschreibt eine beispielhafte GTC-Verwendung zum Bestimmen der Reihenfolge

zwischen den Aktivitäten oder den Hardware-Ereignissen. In einigen Implementierungen kann die Rate, mit der ein GTC **502** während des Verlaufs einer Aktivität, die analysiert wird, inkrementiert wird, nicht relevant sein, wenn der GTC **502** verwendet wird, um eine globale Reihenfolge, z. B. die Reihenfolge einer oder mehrere Aktivitäten zwischen wenigstens zwei verschiedenen Computerknoten (z. B. verschiedenen Knoten, von denen jeder einen Master-GTC aufweist), zu bestimmen. Wenn jedoch ein lokaler GTC **502** an einem speziellen Knoten langsamer als der Master-GTC der Knoten ist, kann der GTC-ext verwendet werden, um zwischen zwei lokalen Aktivitäten oder dazwischen, wann zwei lokale Aktivitäten Schreibvorgänge zu entfernten Betriebsmitteln einleiten, zu unterscheiden, wie durch das Merkmal **508** der Tabelle 504 gezeigt ist.

[0125] Die Tabelle 506 beschreibt eine beispielhafte GTC-Verwendung zum Berechnen der Latenzzeit lokaler Aktivitäten/Ereignisse oder der Aktivitäten, die zwei oder mehr Knoten überspannen. In einigen Implementierungen können die Systemparameter nicht angeben, ob ein GTC **502** in einem speziellen Chip mit einer normalen Rate tickt (d. h., in Synchronisation mit einem bezeichneten globalen Master-Takt) oder mit einer Rate tickt, die zu dem bezeichneten globalen Master phasenverschoben ist. Folglich kann in einigen Fällen der LTC für die Bestimmung der Ereignislatenzzeit der lokalen Aktivitäten verwendet werden, wie durch das Merkmal 510 der Tabelle 506 gezeigt ist. Während in anderen Implementierungen der GTC oder der GTC-ext verwendet werden kann, um die Latenzzeiten einer oder mehrerer Operationen zu messen.

[0126] Im Allgemeinen können die Hardware-Verfolgungsereignisse, die in verschiedenen Knoten oder Komponenten des Systems **100** auftreten, einen GTC-Parameter enthalten. In einigen Ausführungsformen sollten, wenn die verfolgten Hardware-Ereignisse/Aktivitäten mehrere Knoten des Systems **100** überspannen, die den lokalen Aktivitäten zugeordneten Latenzzeiten anstatt über einen LTC über den GTC berechnet werden. In diesen Implementierungen kann die Verwendung des GTC oder des GTC-ext die bevorzugte Zähler-/Taktlösung sein, weil die LTCs über zwei oder mehr verschiedene Knoten nicht synchronisiert sein können oder nicht mit der gleichen Rate takten/ticken können.

[0127] Fig. 6 veranschaulicht einen Blockschaltplan beispielhafter Zeitähler und eines oder mehrerer Zähler, die jeweiligen Komponenten eines beispielhaften Computersystems zur verteilten Hardware-Verfolgung (z. B. dem System **100**) zugeordnet sind. Die Zeitähler können einen globalen Zeitähler (GTC) **602** und mehrerer lokale Zeitähler (LTC) enthalten, die jeder jeweiligen Komponenten des Systems **100** zugeordnet sind.

[0128] Obwohl Fig. 6 für die Klarheit einen einzigen LTC-Block, der an spezielle Komponenten des Systems **100** gekoppelt ist, darstellt, kann in alternativen Implementierungen jede Komponente einen oder mehrere lokale Zeitähler/LTCs enthalten. Obwohl **Fig. 6** einen einzigen Master-GTC **602** darstellt, kann ferner dieses Merkmal außerdem einem speziellen LTC entsprechen oder für einen speziellen LTC repräsentativ sein, der ausgewählt ist, um als der globale Master zu arbeiten. In einigen Implementierungen entspricht der GTC **602** z. B. einer durch das Host-System **126** implementierten Zählerauswahllogik, um einen speziellen LTC auszuwählen, um als ein globaler Master-Takt/Zähler zu arbeiten.

[0129] In einigen Fällen kann das System **100** einen beispielhaften Knoten der ICI **112** auswählen, um als ein globaler Master-Takt zu arbeiten, wobei der LTC des beispielhaften Knotens um eins pro Taktzyklus tickt. Während in alternativen Implementierungen der GTC **602** ein globaler Zeitähler ist, der als ein globaler Master-Takt arbeitet und einen speziellen Zeitparameter an die Komponenten des Systems **100** rundsendet. In jeder Implementierung kann der GTC **602** ein globaler synchroner Takt sein, der verwendet wird, um die Hardware-Ereignisse zu korrelieren, die über das System **100** auftreten.

[0130] In einigen Implementierungen kann der GTC **602** kleine Phasenvariationen über das System **100** aufweisen, wie oben beschrieben worden ist, wobei er aber konfiguriert sein kann, um eine langfristige Zeitdrift zu vermeiden. Der GTC **602** kann z. B. basierend auf einem beispielhaften Oszillator eines speziellen Prozessor-Chips (z. B. eines Oszillators, der ausgewählt ist, um der globale Master zu sein) ticken oder zählen. Im Allgemeinen gibt es keine langfristige Drift oder Phasenvariationen, die für den ausgewählten Oszillator eines Prozessor-Chips in dem System **100** spezifisch sind, wobei jedoch bezüglich der Oszillatoren zwischen verschiedenen Prozessor-Chips im Lauf der Zeit eine Drift oder Variationen auftreten können. Folglich kann der GTC **602** schneller oder langsamer als ein lokaler Takt (LTC) in einem Prozessor-Chip, der nicht der globale Master ist, inkrementieren (d. h., ticken).

[0131] Als der globale Master kann ein aktueller Master-Wert des GTC **602** an alle Knoten der ICI **112** rundgesendet werden. Die beispielhaften ICI-Knoten können der einen oder den mehreren Komponenten des Systems **100** entsprechen. Die empfangenden Knoten können einen Unterschied zwischen ihrem lokalen Wert eines beispielhaften GTC, der für lokale Operationen verwendet wird, die für die spezielle Komponente/den speziellen Knoten spezifisch sind, berechnen. Ein minimaler Unterschied zwischen dem empfangenen Master-Wert und dem lokalen GTC-Wert über einen beispielhaften Abtastzeitraum kann verwendet werden, um den lokalen GTC zu aktua-

lisieren. Im Lauf der Zeit können die Phasenvariationen zwischen dem Master-Wert und den lokalen GTC-Werten, wenn überhaupt, basierend auf den Zählereinstellungen, die unter Verwendung des einen oder der mehreren oben erörterten Versatzbits implementiert sind, „eingeholt“ werden.

[0132] Wie in der Implementierung nach **Fig. 6** gezeigt ist, kann jede der Komponenten des Systems **100** einen oder mehrere Leistungszähler enthalten, wobei gemeinsame Mengen oder Teilmengen von Leistungszählern innerhalb des Systems **100** hier als mehrere Zähler **600** beschrieben werden. Insbesondere und spezifisch für jede Komponente kann der FPC **104** mehrere Zähler **604** enthalten, kann der SPC **106** mehrere Zähler **606** enthalten, kann der Multiplexerspeicher (HBM) **108** mehrere Zähler **608** enthalten, kann die Knotenstruktur **110** mehrere Zähler **610** enthalten, kann die ICI **112** mehrere Zähler **612** enthalten und kann der HIB **114** mehrere Zähler **614** enthalten.

[0133] Im System **100** kann die Leistungsanalyse des Programmcodes, der durch einen oder mehrere Prozessoren ausgeführt wird, basierend auf der Verwendung eines oder mehrerer Hardware-Leistungszähler, die speziellen Komponenten des Systems zugeordnet sind, freigegeben/gesperrt werden. Diese Leistungszähler können den mehreren jeweiligen Zählern nach **Fig. 6** entsprechen und können wenigstens einen der Folgenden enthalten: 1) Aktivitätszähler; 2) Stehenbleiben-Zähler; 3) statistische Zähler; und 4) Abtastzähler.

[0134] In einigen Implementierungen enthält das System **100** verschiedene andere Typen von Leistungszählern, die programmiert sein können, um spezifische Zählraten, die den technischen Aspekten des Ausführens des Programmcodes zugeordnet sind, zu inkrementieren und zu speichern. Durch die Verwendung des einen oder der mehreren Zähler kann die Hardware-Unterstützung für die Verfolgungsaktivität ferner Funktionen enthalten, die z. B. mit Folgendem in Beziehung stehen: der Erfassung von Anweisungsverfolgungs-Ereignisdaten; der Erfassung von Speichermultiplexerverfolgungs-Ereignisdaten; der Erfassung von DMA-Verfolgungs-Ereignisdaten; und der Erfassung von Ereignisdaten in Verfolgungspuffern.

[0135] In einigen Implementierungen kann wenigstens eine Teilmenge der mehreren Zähler **600** CSRs oder Zählregister sein, die über den HIB **114** durch das Host-System **126** zugänglich sind. Diese Zählregister/CSRs können Speichervorrichtungen sein, die konfiguriert sind, um sowohl verschiedene Typen von Zählraten/-informationen, die die Anweisungen identifizieren, die durch eine spezielle Hardware-Komponente empfangen, verarbeitet oder ausgeführt werden, als auch verschiedene andere Typen von Da-

ten/Informationen, die den technischen Aspekten des Ausführens des Programmcodes zugeordnet sind, zu speichern. In einigen Fällen sind die Daten in der Form von Zählerständen oder Inkrementen, die speziellen Anweisungen oder Prozessoroperationen zugeordnet sind, (z. B. eines Zählparameters) gespeichert. In einigen Implementierungen können mehrere Zähler 600 des Systems **100** hunderten oder tausenden von Leistungszähler/Zählregistern entsprechen.

[0136] Hinsichtlich der oben angegebenen wenigstens vier Typen von Leistungszählern können in einigen Implementierungen die Aktivitätszähler verwendet werden, um die Daten zu speichern und zu analysieren, die den Verwendungsparametern für die verschiedenen Hardware-Einheiten zugeordnet sind. Die Aktivitätszähler können außerdem verwendet werden, um Abschnitte des ausgeführten Programmcodes zu analysieren oder um die Mengen oder Typen der Daten, die während der Codeausführung übertragen werden, zu analysieren. Einige Prozessorkerne können z. B. Aktivitätszähler enthalten, die verwendet werden, um eine Anweisungsmischung in verschiedenen Strömen von Anweisungen abzuleiten. Ein Aktivitätszähler kann z. B. konfiguriert sein, um die der Ausführung von speziellen Anweisungstypen in einem beispielhaften Anweisungsstrom zugeordneten Zählerstände zu speichern.

[0137] In einigen Implementierungen können die Aktivitätszähler Teilmengen von Zählern enthalten, wie z. B.: Problemzähler, die bei der Ausgabe bestimmter Anweisungstypen inkrementiert werden; Speicheraktivitätszähler, die in den Speicherwegen verfügbar sind und die in Reaktion z. B. auf Lesevorgänge/Schreibvorgänge, die den Speicherübertragungen dazwischen (z. B. zwischen dem Multiplexer **108** und einem VMEM **402**) zugeordnet sind, inkrementiert werden; und ein oder mehrere andere allgemeine Aktivitätszähler, die in Reaktion auf das Auftreten von Anweisungen, wie z. B. Unterbrechungen, Synchronisationsmerkern und Alarmzählern, inkrementiert werden.

[0138] Hinsichtlich der Stehenbleiben-Zähler können in einigen Implementierungen diese Zähler Stehenbleiben-Zählerstände bereitstellen, die eine Anzahl von Zyklen angeben, die eine bestimmte Hardware-Einheit/Komponente aufgrund eines speziellen Grundes des Stehenbleibens stehengeblieben war, anstatt nützliche Arbeit auszuführen, die der Anweisungsverarbeitung oder den Datenberechnungen zugeordnet ist. Wenn sich z. B. die Verwendung einer speziellen Hardware-Einheit unter einer Schwellenverwendung befindet oder der Programmcodes nicht mit einem Soll-Schwellenleistungspegel ausgeführt wird, können die Stehenbleiben-Zähler durch das System **100** verwendet werden, um relevante Informationen über das Stehenbleiben zum Ausführen einer Analyse der Hauptursache der Verwendungs-

und Leistungsprobleme (z. B. der Verwendung/Leistung unterhalb eines Schwellenwerts) zu erhalten.

[0139] In einigen Implementierungen können die Stehenbleiben-Zähler relevante Informationen bezüglich des Folgenden bereitstellen: Stehenbleiben basierend auf der Detektion bestimmter normal verfügbarer Speicherwege, die als nicht verfügbar angegeben werden; Stehenbleiben basierend auf einer verzögerten Ausführung einer oder mehrerer Synchronisationsmerkeranweisungen; und/oder Stehenbleiben aufgrund einer oder mehrerer normalerweise verfügbarer Ausführungseinheiten (z. B. der XU **122** oder der RPU **124**), die als nicht verfügbar angegeben werden.

[0140] In einigen Implementierungen kann der FPC **104** wenigstens eine Skalareinheit enthalten, die skalare Größen für die durch die VPU **118** ausgeführten vektorisierten Berechnungen bereitstellt. Die Stehenbleiben-Zähler, die der Skalareinheit zugeordnet sind, können relevante Informationen für die Stehenbleiben bereitstellen, die auf ein oder mehrere Risiken zurückzuführen sind, die innerhalb der Skalareinheit auftreten können. In einigen Fällen können verschiedene Risikotypen auftreten und kann der FPC **104** Stehenbleiben-Zähler für jeden Risikotyp enthalten. Beispielhafte Risikotypen können DMA-Verzögerungsrisiken, Skalarhindernisverzögerungsrisiken und/oder Skalarverzögerungsanweisungsrisiken enthalten.

[0141] Hinsichtlich der statistischen Zähler können in einigen Implementierungen die Komponenten des Systems **100** eine oder mehrere Datenwarteschlangen (z. B. Speichermedien) enthalten, wobei die Aktivitätszähler konfiguriert sein können, um die auf die Warteschlangenverwendung bezogenen Informationen bereitzustellen. In einigen Fällen können jedoch die Aktivitätszähler für die Warteschlangen nicht ausreichend sein, um alle Informationen darüber bereitzustellen, wie gut die Warteschlangen durch das Ausführen des Programmcodes verwendet werden. In diesen Fällen können statistische Zähler in einer oder mehreren Komponenten verfügbar sein, um bestimmte Datenzählerstände zu erfassen, so dass die Berechnungen bezüglich der durchschnittlichen Warteschlangenbelegung und der in bestimmten Warteschlangen zugebrachten Zeit bestimmt werden können.

[0142] In einigen Implementierungen können die statistischen Warteschlangenzähler statistische Zählerstände bezüglich der Warteschlangenbelegung, des Warteschlangen-Einfügungsstatus, des Warteschlangen-voll-Status oder des Warteschlange-beider-Schwellenbelegung-Status bereitstellen. In einigen Fällen, wenn der statistische Warteschlangenzähler aktiv ist, wird ein Belegungszähler, der einer Komponentenwarteschlange zugeordnet ist, basie-

rend auf der aktuellen Belegung der Warteschlange in jedem Prozessor-/Anweisungszyklus inkrementiert.

[0143] In anderen Fällen, in denen ein aktiver statistischer Warteschlangenzähler Zählerstände bezüglich des Warteschlangen-voll-Status bereitstellt, wird der Zähler in den Zyklen, in denen eine Warteschlange voll belegt ist (z. B. um eins) inkrementiert. Wenn ein aktiver statistischer Warteschlangenzähler Zählerstände bezüglich des Warteschlangen-Einfügungsstatus bereitstellt, wird der Zähler in Reaktion auf die Datenblöcke, die in die Warteschlange gelegt werden, (d. h., die Warteschlangeneinfügung) inkrementiert. Noch weiter kann für den Warteschlange-bei-der-Schwellenbelegung-Status der Zähler in jedem Zyklus, in dem die aktuelle Warteschlangenbelegung größer als eine spezielle Warteschlangenschwellenbelegung ist, (z. B. um eins) inkrementiert werden.

[0144] In einigen Implementierungen können die Komponenten des Systems **100** Sätze von Abtastzählern enthalten, die periodisch von dem Host-System **126** gelesen werden können, um Datenstrukturen (z. B. die Strukturen **320** oder **620**) zu erzeugen. Die Datenstrukturen können die abgetasteten Statistiken über die Hardware-Ereignisse, die Verfolgungsaktivität oder ausführliche Datenverarbeitungsinformationen für Aktivitäten/Ereignisse, die in den verteilten Prozessoreinheiten des Systems **100** auftreten, enthalten. In einigen Implementierungen können die basierend auf den abgetasteten Statistiken erzeugten Datenstrukturen z. B. die (oben beschriebene) Struktur **320** oder die (im Folgenden beschriebene) Struktur **620** enthalten. In einigen Fällen werden die erzeugten Datenstrukturen verwendet, um ein oder mehrere Überwachungswerkzeuge aufzubauen, die durch die Überwachungsmaschine **128** des Host-Systems **126** verwendet werden, um die Leistung des durch die Komponenten des Systems **100** ausgeführten Programmcodes zu analysieren.

[0145] Wie oben angegeben worden ist, können die Komponenten des Systems **100** eine Ereignisverfolgung basierend auf globalen Steuerungen, die einem oder mehreren Auslösemechanismen zugeordnet sind, einleiten. In einigen Implementierungen können mehrere Zähler **600** freigegeben, gesperrt oder anderweitig gesteuert werden, um Auslösemechanismen einzuleiten, die veranlassen, dass die Verfolgungspunkte ausführen und die Verfolgungseinträge, die spezifische Daten enthalten, die auf ein oder mehrere Hardware-Ereignisse bezogen sind, erzeugen. In einigen Implementierungen werden die Verfolgungseinträge bezüglich der Hardware-Ereignisse mit den Ereignisdaten wenigstens teilweise basierend auf den Zählern befüllt, die durch einen oder mehrere der oben beschriebenen mehreren Zähler **600** angesammelt werden.

[0146] Die globalen Steuerungen enthaltenden im Allgemeinen die Verwendung sowohl von CSR-Steuerungen (d. h., spezifischer CSR-Hardware-Merkmale/Knöpfe) als auch von Auslösemechanismen, die es ermöglichen, dass die Zähler inkrementiert werden, und die es ermöglichen, dass die Verfolgungen in einer koordinierten und synchronisierten Weise erzeugt werden. In einigen Implementierungen kann eine koordinierte und synchronisierte Steuerung über die verteilten Prozessoreinheiten des Systems **100** stattfinden. Wie oben erörtert worden ist, kann eine synchronisierte Verfolgung teilweise durch die Verwendung eines globalen synchronen Zählers freigegeben werden, der als ein globaler Master-Takt (z. B. der GTC **602**) verwendet wird.

[0147] In einigen Implementierungen löst das System **100** eine synchronisierte Ereignisverfolgung basierend auf einem speziellen Zeitparameter, der durch einen globalen Zeittakt der verteilten Prozessoreinheiten angegeben ist, (z. B. einem Zeitparameter 09:01.13) aus. Das System **100** kann z. B. die Werte des globalen Master-Takts des GTC **602** als einen Auslöser verwenden, um alle Zähler zu starten und zu stoppen und während der Dauer einer bestimmten Operation genau zu verfolgen.

[0148] In einigen Fällen kann ein beispielhafter Software-Kompilierer des Systems **100** innerhalb des ausgeführten Programmcodes Anweisungen einbetten oder einfügen, die Operanden oder Parameterwerte für einen Auslösemechanismus enthalten, der auf einem spezifischen vorgegebenen Zeitfenster basiert, das dem GTC **602** zugeordnet ist. Das vorgegebene Zeitfenster kann eine beispielhafte Verfolgungsdauer enthalten, die einen Verfolgungsereignis-Startzeitpunkt (z. B. einen ersten Zeitparameter des GTC **602**) und einen Verfolgungsereignis-Endzeitpunkt (z. B. einen zweiten Zeitparameter des GTC **602**, der zeitlich später als der erste Zeitparameter des GTC ist) enthalten.

[0149] Wie in der Implementierung nach **Fig. 6** gezeigt ist, können die Komponenten des Systems **100** mehrere Leistungszähler **600** enthalten. Wenn es sich auf die Auslösemechanismen bezieht, können die Leistungszähler-Steuerungen Mechanismen enthalten, um sowohl die Zähler freizugeben und zu sperren als auch die Zähler zu löschen und die Zähler zu unterbrechen. Ferner können zusätzlich zu den oben beschriebenen statistischen Zählerfunktionen einige Hardware-Einheiten statistische Zähler enthalten, die zugeordnete Selektoren (z. B. CSRs) aufweisen, die verwendet werden, um eine Teilmenge von Hardware-Blöcken innerhalb der Hardware-Einheit, deren Ereignisse gezählt werden, auszuwählen.

[0150] Wie oben bezüglich der Verfolgungsmultiplexersteuerungen erörtert worden ist, kann in einigen Implementierungen die Verfolgung basierend auf den

durch das System **100** ausgeführten CSR-Schreibvorgängen freigegeben und gesperrt werden. In einigen Fällen kann die Verfolgung durch das System **100** basierend auf dem Wert eines beispielhaften Verfolgungsmarkierungsregisters (TM-Registers) in dem FPC **104** (oder dem Kern **116**) oder basierend auf dem Wert eines beispielhaften TM-Registers in dem SPC **106** dynamisch gestartet und gestoppt werden.

[0151] In einigen Implementierungen kann ein beispielhafter Verfolgungssteuerparameter verwendet werden, um ein spezielles Verfolgungsfenster zu definieren. Der Verfolgungssteuerparameter kann z. B. erlauben, dass hinsichtlich der unteren und oberen Grenzen des GTC **602** (z. B. der Taktwerte der Startzeitpunkts u. des Stoppzeitpunkts) und der unteren und oberen Grenzen eines beispielhaften Verfolgungsmarkierungsregisters in dem FPC **104** oder eines beispielhaften Verfolgungsmarkierungsregisters in dem SPC **106** die Verfolgungsfenster definiert werden.

[0152] In einigen Implementierungen kann der innerhalb des Systems **100** ausgeführte Programmcode eine Auslöserverarbeitungslogik zum Auslösen der Verfolgungsereignisse jedes Prozessorzyklus innerhalb eines des FPC **104** oder des SPC **106** enthalten. In einigen Fällen ist die Auslöselogik einer speziellen „Setzeine-Verfolgungsmarkierung“-Anweisung zugeordnet, die durch die ausgeführte Software/den ausgeführten Programmcode in einen oder mehrere Anweisungsströme eingefügt werden kann.

[0153] Beispielhaft und hinsichtlich der Auslöselogik für den FPC **104** kann die Auslöselogik, wenn eine beispielhafte Anweisung, um eine Verfolgungsmarkierung zu setzen, innerhalb des FPC **104** ausgegeben wird, das Analysieren eines Verfolgungsmarkierungsoperanden der Anweisung und das Vergleichen eines Parameterwerts des Operanden mit den Werten eines beispielhaften Verfolgungsmarkierungsfensters enthalten. Diese Analyse und dieser Vergleich werden verwendet, um zu bestimmen, ob der Wert des Operandenparameters entweder gleich einem unteren Grenzwert ist oder innerhalb des unteren Grenzwerts und eines oberen Grenzwerts eines speziellen Verfolgungsmarkierungsfensters liegt.

[0154] Falls ein Parameterwert des Operanden in die Grenzen des Verfolgungsmarkierungsfensters fällt, kann ein Auslöser aktiviert werden (z. B. eine Auslösebedingung erfüllt sein), wobei der FPC **104** in einen Verfolgungs- oder Zählmodus eintreten und einen oder mehrere Zähler der mehreren Zähler **604** freigeben kann. In einigen Implementierungen verlässt der FPC **104** den Verfolgungs-/Zählmodus, wenn eine anschließende „Setzeine-Verfolgungsmarkierung“-Anweisung einen Wert des Operanden-

parameters enthält, der verursacht, dass ein Stoppe-das-Zählen-Auslöser basierend auf dem Operandenwert, der entweder gleich der oberen Grenze eines beispielhaften Verfolgungsmarkierungsfensters ist oder sich außerhalb der oberen Grenze eines beispielhaften Verfolgungsmarkierungsfensters befindet, aktiviert wird.

[0155] In einigen Implementierungen kann das System **100** basierend auf wenigstens einem des Folgenden detektieren, dass eine Auslösebedingung erfüllt ist: i) dem Identifizieren eines Auftretens eines beispielhaften Verfolgungsmarkierungsoperanden in wenigstens einem ersten Abschnitt des ausgeführten Programmcodes. Der Verfolgungsmarkierungsoperand kann einen Parameterwert (z. B. einen Codeschrittablauf/-wert) enthalten, der verwendet wird, um ein oder mehrere Verfolgungsereignisse und/oder das Leistungszählen einzuleiten. Das System **100** kann außerdem basierend auf einer Bestimmung, dass ein aktueller Zeit-GTC **602** einen vorgegebenen Zeitwert angibt, der verwendet wird, um ein oder mehrere Verfolgungsereignisse und/oder das Leistungszählen einzuleiten, detektieren, dass eine Auslösebedingung erfüllt ist.

[0156] In Reaktion auf das Detektieren, dass die Auslösebedingung erfüllt ist, kann eine Prozessor-komponente des Systems **100** ein Steuersignal erzeugen, das durch ein Zählregister der Prozessor-komponente empfangen wird. Das Steuersignal kann veranlassen, das verschiedene Zählraten, die einem oder mehreren Hardware-Ereignissen zugeordnet sind, in dem Zählregister gespeichert werden. In einigen Implementierungen kann das System **100** konfiguriert oder programmiert sein, um z. B. über das Host-System **126** eine Datenstruktur zu erzeugen, die ein oder mehrere Leistungsattribute angibt, die dem ausgeführten Programmcode zugeordnet sind. Ferner kann die Datenstruktur basierend auf einem oder mehreren Zählparametern der gespeicherten Zählraten erzeugt werden. In einigen Implementierungen ist das Zählregister einer der mehreren Leistungszähler **600**, die konfiguriert sind, um die Zählraten über die Leistung einer oder mehrerer Prozessorkomponenten des Systems **100** zu speichern.

[0157] Um den Verfolgungsmarkierungsauslöser weiter zu veranschaulichen, kann in einer beispielhaften Codefolge die gesetzte Verfolgungsmarkierungsanweisung einen 32-Bit-Auslöseroperanden enthalten, der ein verteiltes Software-Konstrukt (z. B. einen Rechenschritt in einer beispielhaften Schlussfolgerungs-Arbeitsbelastung in einem neuronalen Netz) identifiziert. Der FPC **104** kann ein Spezial-CSR enthalten, das in dem Programmcode als „tracemark_limits“ identifiziert ist. Obwohl in dieser beispielhaften Codefolge 32-Bit-Operanden beschrieben werden, können in alternativen Implementierungen die Auslöseoperanden oder die Registergrenzen-Ope-

randen/Parameterwerte binäre Datenstrukturen sein, die weniger als 32 Bits oder mehr als 32 Bits aufweisen.

[0158] Dieses Spezial-CSR kann wenigstens zwei 32-Bit-Werte decodieren. Ein erster codierter 32-Bit-Wert kann einer unteren Verfolgungsmarkierungsgrenze entsprechen, die verwendet wird, um das Verfolgen/Zählen zu starten, während ein zweiter codierter 32-Bit-Wert einer oberen Verfolgungsmarkierungsgrenze entsprechen kann, die verwendet wird, um das Verfolgen/Zählen zu stoppen. In einigen Implementierungen entsprechen der erste und der zweite codierte Wert den Rechenschritten in einem verteilten Software-Programm/Konstrukt, wie z. B. einer beispielhaften Schlussfolgerungs-Arbeitsbelastung in einem neuronalen Netz oder irgendeinem anderen verteilten Programmcode.

[0159] Wenn abermals in der beispielhaften Codefolge eine „Setze-eine-Verfolgungsmarkierung“-Anweisung ausgeführt wird, wird der 32-Bit-Operandenwert mit dem ersten und dem zweiten codierten Wert in dem trace-mark_limits-Register verglichen. Falls basierend auf diesen Vergleich der Auslöseoperand dem Startwert (d. h., dem ersten codierten Wert) gleicht/entspricht oder einen höheren Wert als der Startwert (d. h., der erste codierte Wert) aufweist, wird die Leistungsanalyseaktivität ausgelöst. Falls gleichermaßen der Auslöseoperand mit dem Stoppwert (d. h., dem zweiten codierten Wert) übereinstimmt oder einen höheren Wert als der Stoppwert (d. h., der zweite codierte Wert) aufweist, wird eine andauernde Leistungsanalyseaktivität gestoppt.

[0160] Ganz wie der Verfolgungsmarkierungsauslöser können, wenn der GTC **602** als ein Auslöser (z. B. für den FPC **104** oder den SPC **106**) verwendet wird, die oberen und die unteren Grenzen der beispielhaften GTC/LTC-Auslöseregister für eine spezielle Komponente gegenüber einem aktuellen Wert des GTC **602** bewertet werden. Folglich kann ein aktueller Zeitparameter oder Wert des GTC **602** mit einer unteren Grenze eines beispielhaften vorgegebenen Zeitfensters verglichen werden, um die Verfolgungsaktivität unter Verwendung eines GTC-Auslösers auszuführen. Ähnlich kann nach der Ausführung des GTC-Auslösers, um die Zählaktivität zu beginnen, ein aktueller Zeitparameter oder Wert des GTC **602** mit der oberen Grenze des GTC-Auslöseregisters verglichen werden, um zu bestimmen, ob der GTC **602** mit dem oberen Grenzwert übereinstimmt oder außerhalb des oberen Grenzwerts liegt. In Reaktion auf den GTC **602**, der wenigstens mit dem oberen Grenzwert übereinstimmt, wird ein Stoppe-das-Zählen-Auslöser aktiviert, um zu veranlassen, dass die Komponente den Zählmodus verlässt.

[0161] In einigen Implementierungen enthält der SPC **106** ein Gegenstück zu der Setze-eine-Verfol-

gungsmarkierung-Anweisung des FPC **104**. Folglich kann die Auslöserverarbeitungslogik für den SPC **106** in der gleichen oder im Wesentlichen ähnlichen Weise wie in dem FPC **104** arbeiten. Ein Verfolgungsmarkierungsparameter kann z. B. als ein Operand in einer oder mehreren durch den SPC **106** ausgegebene Anweisungen enthalten sein. In einigen Fällen findet die Auslöserauswertung des Operanden bei jeder Anweisungsausgabe statt, wobei die Auswertungslogik im Wesentlichen zu der Weise ähnlich sein kann, in der das Zählen in dem FPC **104** (z. B. unter Verwendung eines Verfolgungsmarkierungsoperanden) freigegeben wird. In dem SPC **106** wird z. B. die Verfolgung freigegeben, wenn sich ein in der Verfolgungsmarkierungsanweisung enthaltener Verfolgungsmarkierungsoperand innerhalb der unteren und oberen der Grenze eines beispielhaften Verfolgungsfensters befindet. Wenn sich gleichermaßen der in der Verfolgungsmarkierungsanweisung enthaltene Verfolgungsmarkierungsoperand außerhalb der unteren und der oberen Grenze des Verfolgungsfensters befindet, wird die Verfolgung gesperrt.

[0162] Wie oben erörtert worden ist, kann das System **100** zusätzlich zu der Verwendung des GTC **602** oder der Verfolgungsmarkierungen einen „Alles“-Verfolgungsmodus enthalten. Im Allgemeinen kann dieser Alles-Verfolgungsmodus veranlassen, dass die Verfolgung über einen gesamten Prozessor-Chip z. B. durch ein Steuersignal gesteuert wird, das durch eine Komponente des FPC **104** oder des SPC **106** erzeugt wird. In einigen Implementierungen kann das Steuersignal ein globaler Steuerparameter sein, der einen speziellen Leistungszustand der verteilten Prozessoreinheiten des Systems **100** angibt. In einigen Fällen entspricht der spezielle Leistungszustand dem „Alles“-Verfolgungsmodus.

[0163] In einigen Implementierungen können eine oder mehrere Hardware-Komponenten (z. B. der HIB **114** oder die ICI **112**) programmiert oder konfiguriert sein, um eine Ereignisverfolgung in Reaktion auf die Aktivierung des „Alles“-Verfolgungsmodus einzuleiten oder um einfach ein sich ausbreitendes Steuersignals, das den Alles-Verfolgungsmodus angibt, zu ignorieren. Obwohl das System **100** einen Signalisierungsmechanismus enthalten kann, um ein Alles-Verfolgungs-Steuersignal auszubreiten, kann folglich das System **100** außerdem Mechanismen enthalten, um wenigstens eine Hardware-Komponente zu veranlassen, das Steuersignal zu ignorieren.

[0164] Ein Anwender (z. B. ein Leistungsanalytiker) des Systems **100** kann z. B. wünschen, den HIB **114** zu verwenden, um die Datenpakete zu verfolgen, die alle Host-Kommunikationen angeben. Der Anwender kann z. B. über einen externen Kompilierer eine beispielhafte Programmanweisung einfügen, um den HIB **114** zu veranlassen in einen „Immer-verfolgen“-Modus einzutreten, während außerdem der Alles-

Verfolgungsmodus verwendet wird, um bestimmte einzelne Schrittfolgen der Daten von anderen Hardware-Komponenten des Systems **100** zu erfassen. Wenn der HIB **114** programmiert ist, um im Immerverfolgen-Modus ausgeführt zu werden, kann der HIB **114** das Alles-Verfolgungs-Steuersignal ignorieren, das verwendet wird, um einzelne Schrittfolgen-Verfolgungsdaten von anderen Hardware-Komponenten zu erzeugen.

[0165] In einigen Implementierungen können die Verfolgungspunkte in Reaktion auf das Auftreten einer spezifischen Handlung oder Bedingung bezüglich der Ausführung des Programmcodes durch die Komponenten des Systems **100** auslösen. Ein verteiltes Software-Programm kann z. B. mehrere Software-Module enthalten, wobei jedes gemeinsam benutzte oder überlappende Codefolgen oder -schritte enthält. Die Prozessorkerne **104**, **106** können z. B. jeder Abschnitte eines verteilten Software-Programms empfangen und ausführen, wobei jeder Programmabschnitt eine beispielhafte Codefolge **1-100** enthalten kann.

[0166] Wie oben angegeben worden ist, kann in einigen Implementierungen das System **100** konfiguriert sein, so dass es eine/einen oder mehrere Auslösebedingungen/-mechanismen enthält, die einen speziellen Ablaufschritt des verteilten Programmcodes aufweisen oder auf einem speziellen Ablaufschritt des verteilten Programmcodes basieren. Folglich kann das System **100** konfiguriert sein, um einen Verfolgungspunkt in Reaktion auf jeden der Prozessorkerne **104**, **106**, der an einem bestimmten codierten Ablauf/Schritt (z. B. dem Codeablaufschritt **33**) während der Codeausführung ankommt, auszulösen. In einigen Fällen können die Abschnitte des ausgeführten verteilten Programmcodes eingebettete Auslösefunktionen enthalten, die durch wenigstens eine Hardware-Komponente des Systems **100** detektierbar sind.

[0167] Wie oben angegeben worden ist, kann das System **100** z. B. konfiguriert sein, um basierend auf dem Identifizieren der Vorkommen von Operanden in wenigstens einem Abschnitt des Programmcodes, der entweder durch den FPC **104** oder den SPC **106** ausgeführt wird, zu detektieren, das eine oder mehrere Auslösebedingungen erfüllt sind. In Reaktion auf das Detektieren, das wenigstens eine Auslösebedingung erfüllt ist, kann das System **100** ein oder mehrere Verfolgungsereignisse einleiten, die Verfolgungseinträge/Verfolgungsdaten oder Zählraten erzeugen. In einigen Implementierungen können die erzeugten Verfolgungseinträge wenigstens ein Attribut enthalten, das unter den jeweiligen Hardware-Ereignissen, die über die verteilten Prozesseinheiten auftreten, gemeinsam benutzt wird (z. B. ein gemeinsam benutzter Zeitstempel oder Codeablaufschritt).

[0168] Beispielhaft und wie durch die Datenstruktur **620** gezeigt ist, können die erzeugten Verfolgungseinträge **622** einem synchronisierten Verfolgungsereignis entsprechen, wobei die Verfolgungsdaten eindeutige Verfolgungskennungen für die jeweiligen Hardware-Ereignisse und einen globalen Hardware-Ereignis-Zeitstempel (z.B. 09:01.13), der unter wenigstens zwei jeweiligen Hardware-Ereignissen gemeinsam benutzt wird, enthalten können. Folglich kann für die Verfolgungseinträge **622** wenigstens ein Attribut, das unter den jeweiligen Hardware-Ereignissen gemeinsam benutzt wird, der Hardware-Ereignis-Zeitstempel sein. Wenn das System **100** die Verfolgungspunkte basierend auf einer speziellen Codefolge auslöst, kann gleichermaßen für die Verfolgungseinträge **624** wenigstens ein Attribut, das unter den jeweiligen Hardware-Ereignissen gemeinsam benutzt wird, der Codefolgen- oder Verfolgungsmarkierungsschritt sein.

[0169] In einigen Implementierungen können die Verfolgungseinträge einen Zeitstempel-Kopf oder ein Zeitstempel-Feld enthalten. Der Zeitstempel-Kopf kann einen 48-Bit-Zeitstempel enthalten, der den Zeitpunkt identifiziert, zu dem der Verfolgungseintrag erzeugt wurde. In einigen Fällen kann der Zeitstempel-Kopf den unteren 48 Bits des GTC **502** entsprechen.

[0170] Wie oben angegeben worden ist, kann in einigen Implementierungen ein Kompilierer des Systems **100** einen oder mehrere Operanden für mehrere Auslösebedingungen in Abschnitte des durch die beispielhaften Prozessorkerne des Systems **100** ausgeführten Programmcodes einfügen. Ein dem FPC **104** zugeordneter Kompilierer kann z. B. einen oder mehrere Operanden für die Auslösebedingungen in Abschnitte des durch den FPC **104** ausgeführten Programmcodes einfügen oder einbetten. Gleichermaßen kann der dem SPC **106** zugeordnete Kompilierer einen oder mehrere Operanden für die Auslösebedingungen in Abschnitte des durch den SPC **106** ausgeführten Programmcodes einfügen oder einbetten.

[0171] Im Allgemeinen und wie oben angegeben worden ist, ist das System **100** konfiguriert, um die eine oder die mehreren Auslösebedingungen zu detektieren und in Reaktion auf das Detektieren der Bedingungen wenigstens ein Verfolgungsereignis einzuleiten. In einigen Implementierungen kann das Einleiten des wenigstens einen Verfolgungsereignisses enthalten, dass das System **100** z. B. ein erstes Steuersignal einem ersten Leistungszähler/-register (z. B. einem CSR, wie z. B. einem Aktivitätszähler) des FPC **104** bereitstellt. Dieses erste Steuersignal kann verursachen, dass die wenigstens einem Hardware-Ereignis zugeordneten Daten in dem ersten Leistungszähler gespeichert werden.

[0172] Gleichermaßen kann in einigen Implementierungen des Einleiten des wenigstens einen Verfolgungsereignisses ferner enthalten, dass das System **100** z. B. ein zweites Steuersignal einem zweiten Leistungszähler/-register (z. B. einem CSR, wie z. B. einem statistischen Zähler) des SPC **106** bereitstellt. Dieses zweite Steuersignal kann verursachen, dass die wenigstens einem Hardware-Ereignis zugeordneten Daten in dem zweiten Leistungszähler gespeichert werden.

[0173] In einigen Implementierungen können die dem wenigstens einem Hardware-Ereignis zugeordneten Daten eines des Folgenden enthalten: 1) eine Anzahl von Bytes, die in einem speziellen Speicherpuffer eines speziellen Prozessorkerns der verteilten Prozessoreinheiten des Systems **100** geschrieben sind; oder 2) eine Anzahl von Anweisungen, die durch einen bestimmten Prozessorkern der verteilten Prozessoreinheiten des Systems **100** ausgeführt werden.

[0174] Im Allgemeinen enthält das beschriebene Verfolgungssystem hardwarebasierte Auslösevorgangsmechanismen. Die Auslöser können frühzeitig konfiguriert werden, um basierend auf einer Software-Semantik aktiviert zu werden. Die Operanden für die Auslösebedingungen können z. B. in die Prologe und Epiloge der Software-Komponenten (z. B. in Funktionen, Schleifenköpfe oder andere günstige Orte in einem verteilten Software-System) eingefügt oder eingebettet sein. Wie oben angegeben worden ist, wird in einigen Fällen die Einfügung der Operanden für die Auslösebedingungen durch einen Kompilierer in dem System **100** ausgeführt, während in anderen Fällen die Auslöser und die Auslöseoperanden durch die Programmierer vor der Codeausführung durch das System **100** eingebettet/eingefügt werden können oder während der Echtzeit-Codeausführung durch das Host-System **126** eingebettet werden können.

[0175] In einigen Implementierungen kann z. B. eine Bedingung, die angibt, wann der Auslöser aktiviert werden soll, über ein verteiltes Hardware-System (z. B. in einem Chip/Knoten des Systems **100**) durch die in dem Host-System **126** ausgeführte Leistungsanalyse-Software eingerichtet sein. Wenn der Auslöser aktiviert wird, werden die Hardware-Verfolgung und die Leistungszählung in einer speziellen Hardware-Einheit ausgeführt, wobei ähnliche Auslösebedingungen eingerichtet sind, um die Sammlung der Verfolungsdaten zu stoppen.

[0176] In anderen Implementierungen kann ein Auslöser in dem System **100** ein monoton zunehmender Wert sein, der ein Operand für eine Anweisung ist. Die Anweisung kann durch einen beispielhaften Kompilierer eines Prozessorkerns in die Prologe und Epiloge wichtiger Software-Konstrukte eingefügt wer-

den. Im Allgemeinen gibt es einen vernachlässigbaren Leistungsnachteil für den ausgeführten Programmcode, weil der Auslöseoperand in der Anweisung immer vorhanden ist, weil die Leistungsdaten nicht gesammelt werden, bis ein Auslöser aktiviert wird.

[0177] Eine Auslösebedingung kann z. B. durch das Schreiben in ein beispielhaftes Register in einer Hardware-Komponente eingerichtet werden. Durch das Auswerten oder Detektieren des Auslösers, jedes Mal, wenn einer Auslöseanweisung begegnet wird, kann die Hardware-Komponente den Wert in diesem Register mit dem Operandenwert der Auslösebedingung in der Anweisung vergleichen. Im Allgemeinen sollte die Auslösebedingung gut im Voraus bestimmter codierter Ablaufschritte eingerichtet sein, so dass keiner der Prozessor-Chips/Knoten in dem verteilten System vorzeitig an der Auslösebedingung ankommt.

[0178] In einigen Fällen erfordert die Hardware-Auslöser-Implementierung das Ausbreiten eines Auslösesignals innerhalb unterschiedlicher Komponenten eines beispielhaften Prozessor-Chips/Knotens. Wenigstens ein Vorteil des Auslösevorgangsmechanismus ermöglicht es einem verteilten Software-Programm, eine Aktivität (z. B. einen Codeablaufschritt **33**) in einem ersten Knoten zu einem Zeitpunkt auszuführen, der von einer in Beziehung stehenden Aktivität (z. B. dem Codeablaufschritt **33**) in einem weiteren (zweiten) Knoten sehr verschieden ist. In dieser Implementierung kann das System **100** eine genaue Sammlung der Hardware-Leistungsdaten für völlig gleiche oder im Wesentlichen in Beziehung stehende Aktivitäten aufrechterhalten, die durch verschiedene Hardware-Komponenten oder Prozessorknoten in verschiedenen Zeiträumen ausgeführt werden.

[0179] In einigen Implementierungen kann ein beispielhafter Auslöserverwendungsfall enthalten, dass das System **100** mehrere Sätze von Verfolungsdaten ohne Umprogrammieren, Rücksetzen oder erneutes Einfügen einer speziellen Auslösebedingung sammelt. Der ausgeführte Programmcode kann z. B. Software-Konstrukte enthalten, die eine Anweisungsfolge erzeugen, in der der gleiche Operand, um eine Verfolgungsmarkierung zu setzen, mehrmals erscheint. In einigen Fällen kann der gleiche gesetzte Verfolgungsmarkierungsoperand mehrmals erscheinen, wenn ein Prozessor des Systems **100** einen bestimmten Abschnitt des Programmcodes in einer Schleife ausführt. Während der Ausführung einer Programmschleife kann eine Ereignisverfolgung in Übereinstimmung mit den Iterationen der Programmschleife beginnen und enden.

[0180] Eine Auslösebedingung kann z. B. basierend auf einem Operandenwert, der am Anfang einer Schleifeniteration erscheint, erfüllt sein, wobei der

Auslöser z. B. aktiviert wird und die Ereignisverfolgung beginnt. Gleichmaßen kann eine Auslösebedingung basierend auf einem Operandenwert, der am Ende der Schleifeniteration erscheint, nicht länger erfüllt sein, wobei die Ereignisverfolgung z. B. stoppt. Im Allgemeinen können der Anfang einer Schleifeniteration und das Ende einer Schleifeniteration den Grenzen einer Programmschleife für ein gegebenes Software-Konstrukt entsprechen. Folglich kann eine Verfolgungssitzung bei jeder Ausführung der Software-Konstrukts (z. B. jeder Schleifeniteration) beginnen und enden.

[0181] In einigen Implementierungen können geeignete Auslöseroperandenwerte (z. B. ein GTC oder eine Verfolgungsmarkierung) in der folgenden Weise bestimmt werden. Wie oben angegeben worden ist, kann ein verteiltes Software-Programm Software-Module enthalten, von denen jedes gemeinsam benutzte oder überlappende Codeabläufe oder -schritte enthält. Während der Ausführung des verteilten Programmcodes innerhalb des Systems **100** kann ein Leistungsanalytiker über eine Computerkonsole des Systems **100** einen CSR-Wert eines Verfolgungsmarkierungsregisters ablesen oder analysieren, um z. B. eine aktuelle Nummer des Codeablaufs oder -schritts zu bestimmen.

[0182] In einigen Fällen liest der Analytiker den CSR-Wert wiederholt, um eine durchschnittliche Rate der Zunahme des Ablaufs oder der Schritte zu bestimmen. Die bestimmte Rate der Zunahme kann verwendet werden, um eine oder mehrere Auslösebedingungen basierend auf den Vorkommen z. B. der Verfolgungsmarkierungs-Operandenwerte vorausschauend festzulegen. In einigen Implementierungen kann ein Leistungsanalytiker, wenn ein GTC verwendet wird, die Taktgeschwindigkeit des Systems/der Maschine (oder einer Hardware-Komponente) ablesen oder analysieren, um eine Rate vorherzusagen oder zu bestimmen, mit der der GTC zunimmt. Der Analytiker kann dann die bestimmte Rate verwenden, um basierend auf einem Auftreten eines speziellen GTC-Wertes eine oder mehrere Auslösebedingungen festzulegen.

[0183] Fig. 7 ist ein Prozessablaufplan eines beispielhaften Prozesses **700** zur verteilten Hardware-Verfolgung unter Verwendung des Computersystems **100** und eines oder mehrerer Knoten **200**, **201** des Systems **100**. Folglich kann der Prozess 700 unter Verwendung eines oder mehrerer der oben erwähnten Computerbetriebmittel des Systems **100** einschließlich der Betriebmittel der Knoten **200**, **201** implementiert sein.

[0184] Der Prozess **700** beginnt im Block **702** und enthält, dass das Computersystem 100 die Ausführung eines durch einen ersten Prozessorkern ausgeführten Programmcodes überwacht. In einigen Im-

plementierungen ist der erste Prozessorkern konfiguriert, um wenigstens einen ersten Abschnitt des Programmcodes, der überwacht wird, auszuführen. Im Block **704** des Prozesses **700** überwacht das System 100 die Ausführung des durch einen zweiten Prozessorkern ausgeführten Programmcodes. In einigen Implementierungen ist der zweite Prozessorkern konfiguriert, um wenigstens einen zweiten Abschnitt des Programmcodes, der überwacht wird, auszuführen.

[0185] Im Block **706** detektiert das System **100**, dass wenigstens eine Auslösebedingung erfüllt ist. In einigen Implementierungen detektiert das System **100**, dass die Auslösebedingung erfüllt ist, basierend auf dem Identifizieren eines Auftretens eines Operanden in wenigstens einem Abschnitt des Programmcodes, der entweder durch den ersten Prozessorkern oder durch den zweiten Prozessorkern ausgeführt wird. In einigen Implementierungen enthält der innerhalb des Systems **100** ausgeführte Programmcode eine Auslöserverarbeitungslogik zum Auslösen von Verfolgungsereignissen.

[0186] In einigen Fällen ist der Auslöselogik eine „Setze-eine-Verfolgungsmarkierung“-Anweisung zugeordnet, die in einen oder mehrere Anweisungsströme eingefügt ist. Die Auslöselogik kann das Identifizieren des Auftretens eines Verfolgungsmarkierungsoperanden der Anweisung, das Analysieren des Verfolgungsmarkierungsoperanden und das Vergleichen des Operanden mit den Werten eines beispielhaften Verfolgungsmarkierungsfensters enthalten. In einigen Fällen werden diese Analyse und dieser Vergleich verwendet, um zu detektieren, dass die erste Auslösebedingung erfüllt ist.

[0187] In anderen Fällen ist die Auslöselogik einem vorgegebenen Zeitwert zugeordnet, der durch einen externen Anwender des Systems **100** oder durch einen beispielhaften Kompilierer des Systems in einen oder mehrere Anweisungsströme eingefügt werden kann. Die Auslöselogik kann das Bestimmen enthalten, dass ein aktueller Zeitpunkt wenigstens eines Takts des Computersystems einen vorgegebenen Zeitwert angibt, der verwendet wird, um ein oder mehrere Verfolgungsereignisse einzuleiten. In einigen Implementierungen kann das Bestimmen, dass der aktuelle Zeitpunkt einen vorgegebenen Zeitwert angibt, das Empfangen des vorgegebenen Zeitwerts und das Vergleichen des vorgegebenen Zeitwerts mit den Zeitwerten eines beispielhaften vorgegebenen Zeitfensters enthalten.

[0188] Im Block **708** leitet das System **100** in Reaktion auf das Detektieren, dass die Auslösebedingung erfüllt ist, wenigstens ein erstes Verfolgungsereignis ein, das Verfolgungsdaten erzeugt. Die erzeugten Verfolgungsdaten identifizieren jeweilige Hardware-Ereignisse, die über die verteilten Prozesseinheiten

auftreten, die wenigstens den ersten Prozessorkern und den zweiten Prozessorkern enthalten.

[0189] In Reaktion auf das Detektieren, dass die Auslösebedingung erfüllt ist, kann das System **100** z. B. ein synchronisiertes Verfolgungsereignis einleiten, das die Verfolgungsdaten erzeugt, die die über den FPC **104** und den SPC **106** auftretenden Hardware-Ereignisse identifizieren. In einigen Implementierungen identifizieren die Verfolgungsdaten eine eindeutige Verfolgungskennung für die jeweiligen Hardware-Ereignisse. In einigen Fällen enthalten die Hardware-Ereignisse mehrere synchronisierte Hardware-Ereignisse, wobei wenigstens zwei Hardware-Ereignisse der mehreren synchronisierten Ereignisse synchronisiert sind, wenn die Ereignisse einen globalen Hardware-Ereignis-Zeitstempel gemeinsam benutzen.

[0190] In einigen Implementierungen können dem ausgeführten Programmcode mehrere Auslösebedingungen zugeordnet sein. In einigen Fällen kann eine spezielle Auslösebedingung basierend auf dem Identifizieren eines Auftretens eines speziellen Operanden in wenigstens einem Abschnitt des Programmcodes, der durch wenigstens den ersten oder den zweiten Prozessorkern ausgeführt wird, erfüllt sein. Der spezielle Operand kann einen von mehreren Parameterwerten enthalten, von denen jeder verwendet werden kann, um ein zweites Verfolgungsereignis des einen oder der mehreren Verfolgungsereignisse einzuleiten. In dieser Implementierung kann das Detektieren, dass die spezielle Auslösebedingung erfüllt ist, enthalten, dass das System **100** detektiert, dass ein spezieller Parameterwert einen ersten Grenzwert eines Registers übersteigt.

[0191] In einigen Implementierungen leitet das System **100** in Reaktion auf das Detektieren, dass die spezielle Auslösebedingung erfüllt ist, ein Verfolgungsereignis ein, das Verfolgungsdaten erzeugt, die wenigstens ein Attribut identifizieren, das unter den jeweiligen Hardware-Ereignissen, die über die verteilten Prozessoreinheiten auftreten, gemeinsam benutzt wird. In einigen Implementierungen kann ein Attribut der jeweiligen Hardware-Ereignisse wenigstens eine Quellspeicheradresse, eine Zielspeicheradresse oder einen Ablauf-/Programmrechenschritt enthalten. Wohingegen wenigstens ein gemeinsam benutztes Attribut einen speziellen Ablaufschritt oder Programmschritt des Programmcodes enthalten kann.

[0192] In einigen Implementierungen können eine oder mehrere Auslösebedingungen wenigstens einem des Folgenden zugeordnet sein: 1) einem speziellen Ablaufschritt des Programmcodes; 2) einem globalen Steuerparameter, der einen speziellen Leistungszustand der verteilten Prozessoreinheiten angibt; 3) einem aktuellen Zeitpunkt oder einem speziellen Zeitparameter, der durch einen globalen Zeittakt/Zähler der verteilten Prozessoreinheiten angegeben

wird; oder 4) einem vorgegebenen Zeitfenster, das dem globalen Zeittakt zugeordnet ist.

[0193] In einigen Implementierungen kann das System **100** außerdem detektieren, dass der spezielle Parameterwert des speziellen Operanden oder ein spezieller vorgegebener Zeitwert einen zweiten Grenzwert des Registers übersteigt. In einigen Fällen kann das System **100** in Reaktion auf diese Detektion das zweite Verfolgungsereignis stoppen, wenn der spezielle Parameterwert eines speziellen Operanden oder ein bestimmter vorgegebener Zeitwert den zweiten Grenzwert des Registers übersteigt.

[0194] Im Block **710** des Prozesses **700** stellt das System **100** einer Host-Vorrichtung einen korrelierten Satz von Verfolgungsdaten bereit, der die jeweiligen Hardware-Ereignisse enthält. In einigen Implementierungen gibt der korrelierte Satz von Verfolgungsdaten wenigstens eine zeitlich geordnete Folge der jeweiligen Hardware-Ereignisse an, die erzeugt werden, wenn die erste Auslösebedingung erfüllt ist. Für jedes der jeweiligen Hardware-Ereignisse umfassen die Verfolgungsdaten wenigstens einen Hardware-Ereignis-Zeitstempel für das Hardware-Ereignis und die Metadaten, die das Hardware-Ereignis charakterisieren. In einigen Fällen entspricht der korrelierte Satz von Verfolgungsdaten wenigstens teilweise der Datenstruktur **620**.

[0195] Im Block **712** verwendet das Computersystem **100** die korrelierte Menge von Verfolgungsdaten, um die Leistung des Programmcodes zu analysieren, der durch wenigstens den ersten Prozessorkern und den zweiten Prozessorkern ausgeführt wird. In einigen Implementierungen wird die Datenstruktur **620** (d. h., die den korrelierten Verfolgungsdaten entspricht) durch das Host-System **126** verwendet, um die Leistung des Programmcodes, der durch wenigstens die Prozessorkerne **104** und **106** ausgeführt wird, zu analysieren. Gleichermäßen kann die Datenstruktur **620** durch das Host-System **126** verwendet werden, um die Leistung wenigstens einer Komponente des Systems **100** zu analysieren.

[0196] Das Host-System **126** oder ein beispielhafter Anwender kann z. B. die Datenstruktur **620** analysieren, um zu detektieren oder zu bestimmen, ob es ein Leistungsproblem gibt, das der Ausführung eines speziellen Software-Moduls des Programmcodes zugeordnet ist. Ein beispielhaftes Problem kann enthalten, dass das Software-Modul die Ausführung bestimmter Anweisungsströme innerhalb eines zugewiesenen Zeitfensters oder unter einer Schwellenlatenzzeit nicht abschließen kann.

[0197] Ferner kann der Anwender oder die Host-Vorrichtung/das Host-System **126** detektieren oder bestimmen, ob eine spezielle Komponente des Systems **100** über oder unter einem Schwellenleistungs-

niveau arbeitet. Ein beispielhaftes Problem bezüglich der Komponentenleistung kann eine spezielle Hardware-Komponente enthalten, die Ergebnisdaten erzeugt, die sich außerhalb annehmbarer Schwellenbereiche der Ergebnisparameter befinden. In einigen Implementierungen können die erzeugten Ergebnisdaten mit den Ergebnisdaten, die durch andere in Beziehung stehende Komponenten des Systems **100** erzeugt werden, die im Wesentlichen ähnliche Anweisungen/Operationen ausführen, inkonsistent sein.

[0198] Während der Ausführung des Programmcodes kann es z. B. erforderlich sein, dass eine erste Komponente des Systems **100** eine Operation abschließt und ein Ergebnis erzeugt. Gleichmaßen kann es erforderlich sein, dass eine zweite Komponente des Systems **100** eine im Wesentlichen ähnliche Operation abschließt und ein im Wesentlichen ähnliches Ergebnis erzeugt. Die Analyse eines korrelierten Satzes von Verfolgungsdaten kann angeben, dass die zweite Komponente ein Ergebnis erzeugt hat, das von dem durch die erste Komponente erzeugten Ergebnis beträchtlich verschieden ist. Gleichmaßen kann eine beispielhafte Datenstruktur korrelierter Verfolgungseinträge Ergebnisparameterwerte der zweiten Komponente angeben, die sich außerhalb annehmbarer Parameterbereiche befinden. Diese Ergebnisse können ein potentiellcs Leistungsproblem der zweiten Komponente angeben.

[0199] Die Ausführungsformen des Gegenstands und die funktionalen Operationen, die in dieser Beschreibung beschrieben sind, können in einer digitalen elektronischen Schaltungsanordnung, in greifbar verkörperter Computer-Software oder -Firmware, in Computer-Hardware einschließlich der in dieser Beschreibung offenbarten Strukturen und ihrer strukturellen Äquivalente oder in Kombinationen aus einem oder mehreren von diesen implementiert sein. Die Ausführungsformen des in dieser Beschreibung beschriebenen Gegenstands können als ein oder mehrere Computerprogramme, d. h., ein oder mehrere Module von Computerprogrammanweisungen, die in einem greifbaren nichttransitorischen Programmträger codiert sind, für die Ausführung durch eine Datenverarbeitungsvorrichtung oder zum Steuern des Betriebs einer Datenverarbeitungsvorrichtung implementiert sein. Alternativ oder zusätzlich können die Programmanweisungen auf einem künstlich erzeugten ausgebreiteten Signal, z. B. einem maschinen-erzeugten elektrischen, optischen oder elektromagnetischen Signal, codiert sein, das erzeugt wird, um die Informationen für die Übertragung zu einer geeigneten Empfängervorrichtung zur Ausführung durch eine Datenverarbeitungsvorrichtung zu codieren. Das Computerspeichermedium kann eine maschinenlesbare Speichervorrichtung, ein maschinenlesbares Speichersubstrat, eine Schreib-Lese-Speichervorrichtung oder eine Speichervorrichtung mit

seriellem Zugriff oder eine Kombination aus einem oder mehreren von diesen sein.

[0200] Die Prozesse und Logikflüsse, die in dieser Beschreibung beschrieben sind, können durch einen oder mehrere programmierbare Computer, die ein oder mehrere Computerprogramme ausführen, um die Funktionen durch das Arbeiten an den Eingangsdaten und das Erzeugen der Ausgabe(n) auszuführen, ausgeführt werden. Die Prozesse und Logikflüsse können außerdem durch eine Speziallogikschaltungsanordnung, z. B. eine FPGA (eine feldprogrammierbare Gatteranordnung), eine ASIC (eine anwendungsspezifische integrierte Schaltung) oder eine GPGPU (eine Universalgraphikverarbeitungseinheit), ausgeführt werden, wobei die Vorrichtung außerdem als eine Speziallogikschaltungsanordnung, z. B. eine FPGA (eine feldprogrammierbare Gatteranordnung), eine ASIC (eine anwendungsspezifische integrierte Schaltung) oder eine GPGPU (eine Universalgraphikverarbeitungseinheit), implementiert sein kann.

[0201] Die für die Ausführung eines Computerprogramms geeigneten Computer enthalten beispielhaft sowohl Universal- als auch Spezialmikroprozessoren oder beides und irgendeine andere Art von Zentraleinheit und können auf diesen basieren. Im Allgemeinen empfängt eine Zentraleinheit Anweisungen und Daten von einem Festwertspeicher oder einem Schreib-Lese-Speicher oder beiden. Die wesentlichen Elemente eines Computers sind eine Zentraleinheit zum Ausführen oder Abarbeiten von Anweisungen und eine oder mehrere Speichervorrichtungen zum Speichern von Anweisungen und Daten. Im Allgemeinen enthält ein Computer außerdem eine oder mehrere Massenspeichervorrichtungen zum Speichern von Daten, z. B. magnetische, magneto-optische Platten oder optische Platten, oder ist ein Computer außerdem betriebstechnisch an eine oder mehrere Massenspeichervorrichtungen zum Speichern von Daten, z. B. magnetische, magneto-optische Platten oder optische Platten, gekoppelt, um Daten von ihnen zu empfangen oder Daten zu ihnen übertragen, oder beides. Ein Computer muss jedoch derartige Vorrichtungen nicht aufweisen.

[0202] Die zum Speichern von Computerprogrammanweisungen und Daten geeigneten computerlesbaren Medien enthalten alle Formen von nichtflüchtigem Speicher, nichtflüchtigen Medien und nichtflüchtigen Speichervorrichtungen, einschließlich beispielhafter Halbleiterspeichervorrichtungen, z. B. EPROM, EEPROM und Flash-Speichervorrichtungen; Magnetplatten, z. B. interner Festplatten oder abnehmbarer Platten. Der Prozessor und der Speicher können außerdem durch eine logische Spezialschaltungsanordnung ergänzt oder in einer logischen Spezialschaltungsanordnung enthalten sein.

[0203] Während diese Beschreibung viele spezifische Implementierungseinzelheiten enthält, sollten diese nicht als Einschränkungen an den Schutzzumfang irgendeiner Erfindung oder dessen, was beansprucht werden kann, sondern stattdessen als Beschreibungen der Merkmale, die für spezielle Ausführungsformen spezieller Erfindungen spezifisch sein können, ausgelegt werden. Bestimmte Merkmale, die in dieser Beschreibung im Kontext separater Ausführungsformen beschrieben sind, können außerdem in Kombination in einer einzigen Ausführungsform implementiert sein. Umgekehrt können verschiedene Merkmale, die im Kontext einer einzigen Ausführungsform beschrieben sind, außerdem in mehreren Ausführungsformen getrennt oder in irgendeiner geeigneten Unterkombination implementiert sein. Obwohl die Merkmale oben als in bestimmten Kombinationen wirkend beschrieben und sogar anfangs als solche beansprucht sein können, können überdies ein oder mehrere Merkmale aus einer beanspruchten Kombination in einigen Fällen aus der Kombination entfernt werden und kann die beanspruchte Kombination auf eine Unterkombination oder eine Variation einer Unterkombination gerichtet sein.

[0204] Während die Operationen in den Zeichnungen in einer speziellen Reihenfolge dargestellt sind, sollte dies ähnlich nicht so verstanden werden, dass es erforderlich ist, dass derartige Operationen in der gezeigten speziellen Reihenfolge oder in einer sequentiellen Reihenfolge ausgeführt werden oder dass alle veranschaulichten Operationen ausgeführt werden, um die erwünschten Ergebnisse zu erreichen. Unter bestimmten Umständen können Multitasking und Parallelverarbeitung vorteilhaft sein. Überdies sollte die Trennung der verschiedenen Systemmodule und -komponenten in den oben beschriebenen Ausführungsformen nicht so verstanden werden, dass eine derartige Trennung in allen Ausführungsformen erforderlich ist, wobei sie so verstanden werden sollte, dass die beschriebenen Programmkomponenten und Systeme im Allgemeinen in einem einzigen Software-Produkt zusammen integriert sein können oder in mehrere Pakete von Software-Produkten verpackt sein können.

[0205] Es sind spezielle Ausführungsformen des Gegenstands beschrieben worden. Andere Ausführungsformen befinden sich innerhalb des Schutzzumfangs der folgenden Ansprüche. Die in den Ansprüchen dargestellten Vorgänge können z. B. in einer anderen Reihenfolge ausgeführt werden und dennoch die erwünschten Ergebnisse erreichen. Als ein Beispiel erfordern die in den beigefügten Figuren dargestellten Prozesse nicht notwendigerweise die gezeigte spezielle Reihenfolge oder eine sequentielle Reihenfolge, um die erwünschten Ergebnisse zu erreichen. In bestimmten Implementierungen können Multitasking und Parallelverarbeitung vorteilhaft sein.

[0206] Gemäß den Implementierungen in dieser Offenbarung wird ein computerimplementiertes Verfahren geschaffen, das das Überwachen der Ausführung von Programmcode durch eine erste und eine zweite Prozessorkomponente enthält. Ein Computersystem detektiert, dass eine Auslösebedingung erfüllt ist, durch: i) Identifizieren eines Operanden in einem Abschnitt des Programmcodes; oder ii) Bestimmen, dass ein aktueller Zeitpunkt eines Takts des Computersystems einen vorgegebenen Zeitwert angibt. Der Operand und der vorgegebene Zeitwert werden verwendet, um Verfolgungsereignisse einzuleiten. Wenn die Auslösebedingung erfüllt ist, leitet das System Verfolgungsereignisse ein, die Verfolgungsdaten erzeugen, die jeweilige Hardware-Ereignisse identifizieren, die über das Computersystem auftreten. Das System verwendet die Verfolgungsdaten, um einen korrelierten Satz von Verfolgungsdaten zu erzeugen. Die korrelierten Verfolgungsdaten geben einen zeitlich geordneten Ablauf der jeweiligen Hardware-Ereignisse an. Das System verwendet den korrelierten Satz von Verfolgungsdaten, um die Leistung des Ausführens des Programmcodes zu analysieren.

Patentansprüche

1. Computerimplementiertes Verfahren, wobei das Verfahren umfasst:

Überwachen der Ausführung des Programmcodes durch eine erste Prozessorkomponente, wobei die erste Prozessorkomponente konfiguriert ist, um wenigstens einen ersten Abschnitt des Programmcodes auszuführen; und

Überwachen der Ausführung des Programmcodes durch eine zweite Prozessorkomponente, wobei die zweite Prozessorkomponente konfiguriert ist, um wenigstens einen zweiten Abschnitt des Programmcodes auszuführen;

Detektieren durch ein Computersystem, das eine Auslösebedingung erfüllt ist, basierend wenigstens auf einem des Folgenden:

i) Identifizieren eines Auftretens eines Operanden in wenigstens dem ersten Abschnitt des Programmcodes oder dem zweiten Abschnitt des Programmcodes, wobei der Operand einen ersten Parameterwert enthält, der verwendet wird, um ein oder mehrere Verfolgungsereignisse einzuleiten, oder

ii) Bestimmen, dass ein aktueller Zeitpunkt wenigstens eines Takts des Computersystems einen vorgegebenen Zeitwert angibt, der verwendet wird, um ein oder mehrere Verfolgungsereignisse einzuleiten; in Reaktion auf das Detektieren, dass die Auslösebedingung erfüllt ist, Einleiten durch das Computersystem wenigstens eines ersten Verfolgungsereignisses, das Verfolgungsdaten erzeugt, wobei die Verfolgungsdaten jeweilige Hardware-Ereignisse identifizieren, die über die verteilten Prozessoreinheiten auftreten, die wenigstens die erste Prozessorkomponente und die zweite Prozessorkomponente enthalten, wobei für jedes der jeweiligen Hardware-Ereig-

nisse die Verfolungsdaten wenigstens einen Hardware-Ereignis-Zeitstempel umfassen; und
Verwenden durch das Computersystem der Verfolungsdaten, um einen korrelierten Satz von Verfolungsdaten zu erzeugen, die wenigstens eine zeitlich geordnete Folge der jeweiligen Hardware-Ereignisse angeben, die erzeugt werden, wenn die Auslösebedingung erfüllt ist.

2. Verfahren nach Anspruch 1, wobei das erste Verfolgungsereignis ein synchronisiertes Verfolgungsereignis ist, das Verfolungsdaten erzeugt, die die über die verteilten Prozessoreinheiten auftretenden Hardware-Ereignisse identifizieren, wobei die Verfolungsdaten außerdem eine eindeutige Verfolgungskennung für die jeweiligen Hardware-Ereignisse identifizieren und wobei die Hardware-Ereignisse mehrere synchronisierte Hardware-Ereignisse enthalten, wobei zwei Hardware-Ereignisse synchronisiert sind, wenn die Ereignisse einen globalen Hardware-Ereignis-Zeitstempel gemeinsam benutzen.

3. Verfahren nach Anspruch 1 oder 2, wobei das Detektieren, dass die Auslösebedingung erfüllt ist, umfasst:

Detektieren durch das Computersystem, dass eines des Folgenden:

- i) der erste Parameterwert des Operanden übersteigt einen ersten Grenzwert eines Registers oder
- ii) der durch den aktuellen Zeitpunkt angegebene vorgegebene Zeitwert übersteigt einen ersten Grenzwert eines Registers; und

in Reaktion auf das Detektieren, dass die Auslösebedingung erfüllt ist, Einleiten durch das Computersystem eines zweiten Verfolgungsereignisses, das Verfolungsdaten erzeugt, wobei die Verfolungsdaten wenigstens ein Attribut identifizieren, das unter den jeweiligen Hardware-Ereignissen, die über die verteilten Prozessoreinheiten auftreten, gemeinsam benutzt wird.

4. Verfahren nach Anspruch 3, das ferner umfasst: Detektieren durch das Computersystem, dass eines des Folgenden:

ein zweiter Parameterwert des Operanden übersteigt einen zweiten Grenzwert des Registers oder
ein zweiter vorgegebener Zeitwert, der durch den aktuellen Zeitpunkt angegeben wird, übersteigt einen zweiten Grenzwert des Registers; und
in Reaktion auf das Detektieren Stoppen durch das Computersystem des zweiten Verfolgungsereignisses, wenn der zweite Parameterwert des Operanden den zweiten Grenzwert übersteigt oder wenn der zweite vorgegebene Zeitwert den zweiten Grenzwert übersteigt.

5. Verfahren nach Anspruch 3 oder 4, wobei: der Operand ferner wenigstens einen der Folgenden enthält:

einen Parameter, der einen speziellen Ablaufschritt des Programmcodes angibt; oder
einen globalen Steuerparameter, der einen speziellen Leistungszustand der verteilten Prozessoreinheiten angibt; und

wobei der vorgegebene Zeitwert wenigstens einen der Folgenden enthält:

einen speziellen Zeitwert, der durch einen globalen Zeittakt der verteilten Prozessoreinheiten angegeben wird; oder

einen speziellen Zeitwert eines vorgegebenen Zeitfensters, das dem globalen Zeittakt zugeordnet ist.

6. Verfahren nach einem der Ansprüche 3 bis 5, wobei:

der Operand eine erste binäre Datenstruktur aufweist und der erste Parameterwert des ersten Operanden einer Verfolgungsmarkierung entspricht,
der vorgegebene Zeitwert eine zweite binäre Datenstruktur aufweist und der aktuelle Zeitpunkt durch einen globalen Zeittakt angegeben wird,
und wobei der globale Zeittakt durch die verteilten Prozessoreinheiten verwendet wird, um einen oder mehrere Hardware-Ereignis-Zeitstempel zu erzeugen.

7. Verfahren nach einem der Ansprüche 3 bis 6, das ferner umfasst:

Einfügen durch einen Kompilierer des Computersystems des Operanden der Auslösebedingung in wenigstens den ersten Abschnitt des durch die erste Prozessorkomponente ausgeführten Programmcodes; und

Einfügen durch den Kompilierer des Computersystems wenigstens eines vorgegebenen Zeitwerts der Auslösebedingung in wenigstens den zweiten Abschnitt des durch die zweite Prozessorkomponente ausgeführten Programmcodes.

8. Verfahren nach einem der Ansprüche 3 bis 7, wobei das Einleiten wenigstens eines des ersten Verfolgungsereignisses oder des zweiten Verfolgungsereignisses umfasst:

Erzeugen durch das Computersystem eines ersten Steuersignals, das durch ein erstes Zählregister der ersten Prozessorkomponente empfangen wird, wobei das erste Steuersignal verursacht, dass die einem ersten Hardware-Ereignis zugeordneten Daten in dem ersten Zählregister gespeichert werden; und
Erzeugen durch das Computersystem eines zweiten Steuersignals, das durch ein zweites Zählregister der zweiten Prozessorkomponente empfangen wird, wobei das zweite Steuersignal verursacht, dass die einem zweiten Hardware-Ereignis zugeordneten Daten in dem zweiten Zählregister gespeichert werden.

9. Verfahren nach Anspruch 8, wobei die Daten, die entweder dem ersten Hardware-Ereignis oder dem zweiten Hardware-Ereignis zugeordnet sind, wenigstens eine der Folgenden umfassen:

eine Anzahl von Bytes, die in einen speziellen Speicherpuffer eines speziellen Prozessorkerns der verteilten Prozessoreinheiten geschrieben sind; oder eine Anzahl von Anweisungen, die durch eine spezielle Prozessorkomponente der verteilten Prozessoreinheiten ausgeführt werden.

10. Verfahren nach einem der Ansprüche 1 bis 9, das ferner umfasst:

Identifizieren durch das Computersystem eines Auftretens eines zweiten Operanden in wenigstens einem Abschnitt des durch die erste oder die zweite Prozessorkomponente ausgeführten Programmcodes, wobei der zweite Operand einen zweiten Parameterwert enthält;

Bestimmen durch das Computersystem, das eine Filterbedingung erfüllt ist, basierend auf dem zweiten Parameterwert des zweiten Operanden, der entweder: einen speziellen Grenzwert eines Registers übersteigt oder sich unter einem speziellen Grenzwert des Registers befindet; und

in Reaktion auf das Bestimmen, dass die Filterbedingung erfüllt ist, Filtern durch das Computersystem eines oder mehrerer Verfolgungsereignisse, wobei das Filtern des eines oder der mehreren Verfolgungsereignisse das Ausschließen der Speicherung der einem oder mehreren Hardware-Ereignissen zugeordneten Verfolungsdaten umfasst.

11. Hardware-Ereignis-Sammelsystem, das umfasst:

einen oder mehrere Prozessoren, die einen oder mehrere Prozessorkerne enthalten;

eine oder mehrere maschinenlesbare Speichereinheiten zum Speichern von Anweisungen, die durch den einen oder die mehreren Prozessoren ausführbar sind, um Operationen auszuführen, die umfassen:

Überwachen der Ausführung des Programmcodes durch eine erste Prozessorkomponente, wobei die Prozessorkomponente konfiguriert ist, um wenigstens einen ersten Abschnitt des Programmcodes auszuführen;

Überwachen der Ausführung des Programmcodes durch eine zweite Prozessorkomponente, wobei die zweite Prozessorkomponente konfiguriert ist, um wenigstens einen zweiten Abschnitt des Programmcodes auszuführen;

Detektieren durch ein Computersystem, das eine Auslösebedingung erfüllt ist, basierend auf wenigstens einem des Folgenden:

i) Identifizieren eines Auftretens eines Operanden in wenigstens dem ersten Abschnitt des Programmcodes oder dem zweiten Abschnitt des Programmcodes, wobei der Operand einen ersten Parameterwert enthält, der verwendet wird, um ein oder mehrere Verfolgungsereignisse einzuleiten, oder

ii) Bestimmen, dass ein aktueller Zeitpunkt wenigstens eines Takts des Computersystems einen vorge-

gebenen Zeitwert angibt, der verwendet wird, um ein oder mehrere Verfolgungsereignisse einzuleiten; in Reaktion auf das Detektieren, dass die Auslösebedingung erfüllt ist, Einleiten durch das Computersystem wenigstens eines ersten Verfolgungsereignisses, das Verfolungsdaten erzeugt, wobei die Verfolungsdaten jeweilige Hardware-Ereignisse identifizieren, die über die verteilten Prozessoreinheiten auftreten, die wenigstens die erste Prozessorkomponente und die zweite Prozessorkomponente enthalten, wobei die Verfolungsdaten für jedes der jeweiligen Hardware-Ereignisse wenigstens einen Hardware-Ereignis-Zeitstempel umfassen; und Verwenden durch das Computersystem der Verfolungsdaten, um einen korrelierten Satz von Verfolungsdaten zu erzeugen, die wenigstens eine zeitlich geordnete Folge der jeweiligen Hardware-Ereignisse angeben, die erzeugt werden, wenn die Auslösebedingung erfüllt ist.

12. Hardware-Ereignis-Sammelsystem nach Anspruch 11, wobei das erste Verfolgungsereignis ein synchronisiertes Verfolgungsereignis ist, das Verfolungsdaten erzeugt, die die über die verteilten Prozessoreinheiten auftretenden Hardware-Ereignisse identifizieren, wobei die Verfolungsdaten außerdem eine eindeutige Verfolgungskennung für die jeweiligen Hardware-Ereignisse identifizieren und wobei die Hardware-Ereignisse mehrere synchronisierte Hardware-Ereignisse enthalten, wobei zwei Hardware-Ereignisse synchronisiert sind, wenn die Ereignisse einen globalen Hardware-Ereignis-Zeitstempel gemeinsam benutzen.

13. Hardware-Ereignis-Sammelsystem nach Anspruch 11 oder 12, wobei das Detektieren, dass die Auslösebedingung erfüllt ist, umfasst:

Detektieren durch das Computersystem, dass eines des Folgenden:

i) der erste Parameterwert des Operanden übersteigt einen ersten Grenzwert eines Registers oder

ii) der durch den aktuellen Zeitpunkt angegebene vorgegebene Zeitwert übersteigt einen ersten Grenzwert eines Registers; und

in Reaktion auf das Detektieren, dass die Auslösebedingung erfüllt ist, Einleiten durch das Computersystem eines zweiten Verfolgungsereignisses, das Verfolungsdaten erzeugt, wobei die Verfolungsdaten wenigstens ein Attribut identifizieren, das unter den jeweiligen Hardware-Ereignissen, die über die verteilten Prozessoreinheiten auftreten, gemeinsam benutzt wird.

14. Hardware-Ereignis-Sammelsystem nach Anspruch 13, wobei die Operationen ferner umfassen: Detektieren durch das Computersystem, dass eines des Folgenden:

ein zweiter Parameterwert des Operanden übersteigt einen zweiten Grenzwert des Registers oder

ein zweiter vorgegebener Zeitwert, der durch den aktuellen Zeitpunkt angegeben wird, übersteigt einen zweiten Grenzwert des Registers; und
in Reaktion auf das Detektieren Stoppen durch das Computersystem des zweiten Verfolgungsereignisses, wenn der zweite Parameterwert des Operanden den zweiten Grenzwert übersteigt oder wenn der zweite vorgegebene Zeitwert den zweiten Grenzwert übersteigt.

15. Hardware-Ereignis-Sammelsystem nach Anspruch 13 oder 14, wobei:
der Operand ferner wenigstens eines des Folgenden enthält:

einen Parameter, der einen speziellen Ablaufschritt des Programmcodes angibt; oder
einen globalen Steuerparameter, der einen speziellen Leistungszustand der verteilten Prozessoreinheiten angibt; und
wobei der vorgegebene Zeitwert wenigstens einen der Folgenden enthält:
einen speziellen Zeitwert, der durch einen globalen Zeittakt der verteilten Prozessoreinheiten angegeben wird; oder
einen speziellen Zeitwert eines vorgegebenen Zeitfensters, das dem globalen Zeittakt zugeordnet ist.

16. Hardware-Ereignis-Sammelsystem nach einem der Ansprüche 13 bis 15, wobei:
der Operand eine erste binäre Datenstruktur aufweist und der erste Parameterwert des ersten Operanden einer Verfolgungsmarkierung entspricht,
der vorgegebene Zeitwert eine zweite binäre Datenstruktur aufweist und der aktuelle Zeitpunkt durch einen globalen Zeittakt angegeben wird,
und wobei der globale Zeittakt durch die verteilten Prozessoreinheiten verwendet wird, um einen oder mehrere Hardware-Ereignis-Zeitstempel zu erzeugen.

17. Computerimplementiertes Verfahren, wobei das Verfahren umfasst:

Überwachen der Ausführung des Programmcodes durch eine Prozessorkomponente, wobei die Prozessorkomponente konfiguriert ist, um wenigstens einen ersten Abschnitt des Programmcodes auszuführen;
Detektieren durch ein Computersystem, das eine Auslösebedingung erfüllt ist, basierend auf wenigstens einem des Folgenden:

i) Identifizieren eines Auftretens eines Operanden in wenigstens dem ersten Abschnitt des Programmcodes, wobei der Operand einen ersten Parameterwert enthält, der verwendet wird, um ein oder mehrere Verfolgungsereignisse einzuleiten, oder
ii) Bestimmen, dass ein aktueller Zeitpunkt wenigstens eines Takts des Computersystems einen vorgegebenen Zeitwert angibt, der verwendet wird, um das eine oder die mehreren Verfolgungsereignisse einzuleiten;

in Reaktion auf das Detektieren, dass die Auslösebedingung erfüllt ist, Erzeugen durch das Computersystem eines Steuersignals, das durch ein Zählregister der Prozessorkomponente empfangen wird, wobei das Steuersignal verursacht, dass die einem Hardware-Ereignis zugeordneten Zähldaten in dem Zählregister gespeichert werden; und
Erzeugen durch das Computersystem einer Datenstruktur, die ein oder mehrere Leistungsattribute angibt, die dem ausgeführten Programmcode zugeordnet sind, wobei die Datenstruktur basierend auf einem oder mehreren Zählparametern der gespeicherten Zähldaten erzeugt wird.

18. Verfahren nach Anspruch 17, wobei das Zählregister einer von mehreren Leistungszählern ist, die konfiguriert sind, um die Zähldaten über die Leistung einer oder mehrerer Prozessorkomponenten des Computersystems zu speichern, und wobei wenigstens ein Leistungszähler einen der Folgenden enthält: einen Aktivitätszähler, einen Stehenbleiben-Zähler, einen statistischen Zähler oder einen Abtastzähler.

19. Verfahren nach Anspruch 18, wobei der eine oder die mehreren Zählparameter eine der Folgenden angeben:

i) eine Anzahl der durch eine spezielle Prozessorkomponente empfangenen Anweisungen;
ii) eine Anzahl der durch die spezielle Prozessorkomponente verarbeiteten Anweisungen;
iii) eine Anzahl der durch die spezielle Prozessorkomponente ausgeführten Anweisungen; oder
iv) eine Anzahl der Speicherlesevorgänge oder eine Anzahl der Speicherschreibvorgänge, die durch die spezielle Prozessorkomponente ausgeführt werden.

20. Verfahren nach einem der Ansprüche 17 bis 19, wobei ein Leistungsattribut, das dem ausgeführten Programmcode zugeordnet ist, eines des Folgenden umfasst:

i) eine Stehenbleiben-Frequenz einer speziellen Prozessorkomponente, die den Programmcode ausführt;
ii) eine Angabe, dass sich die Verwendung der speziellen Prozessorkomponente unter einer Schwellenverwendung befindet; oder
iii) eine Angabe, dass sich eine durch die spezielle Prozessorkomponente verwendete Datenspeicher-Warteschlange auf oder unter einem Warteschlangenbelegungsschwellenwert befindet.

Es folgen 7 Seiten Zeichnungen

Anhängende Zeichnungen

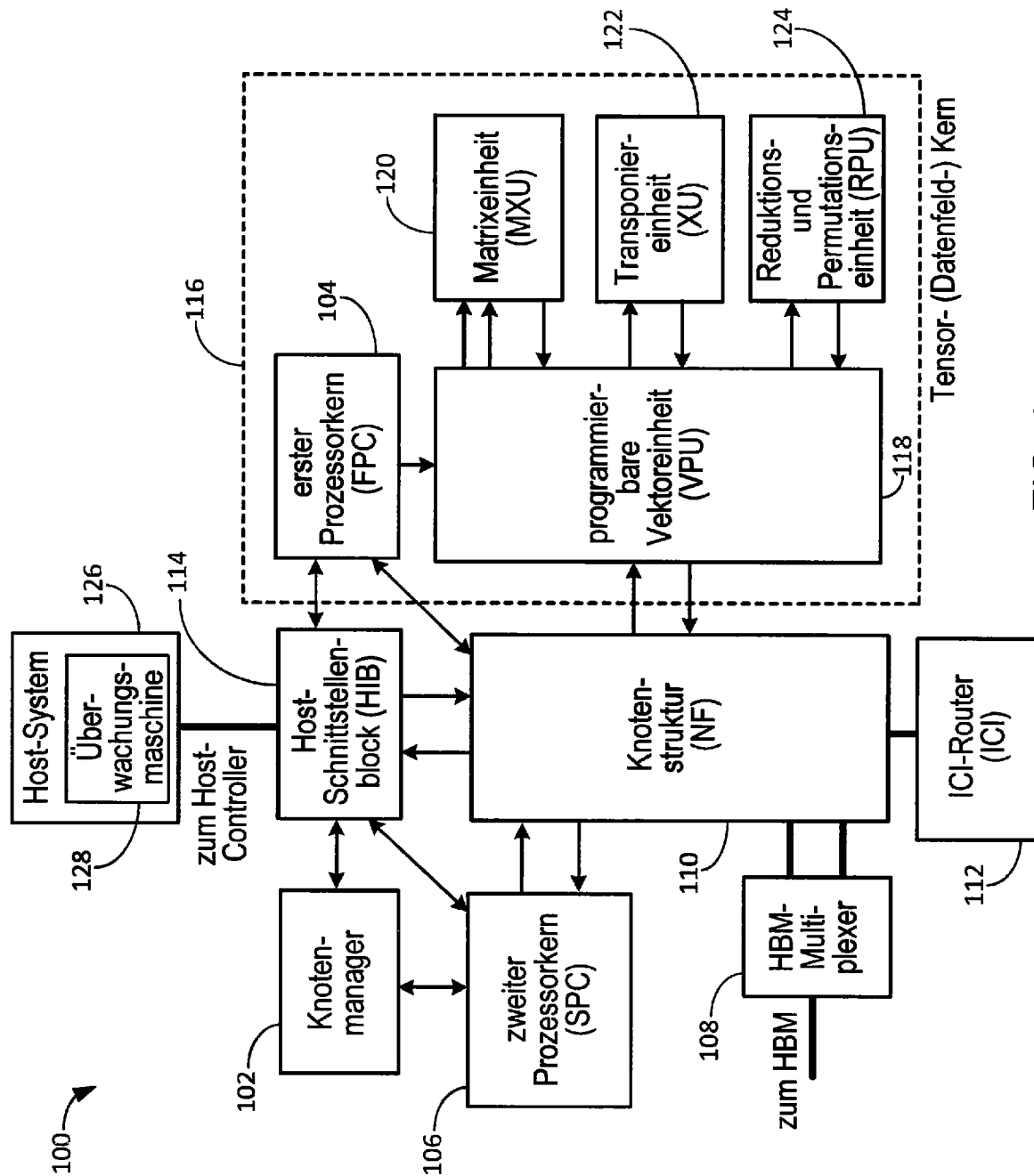


FIG. 1

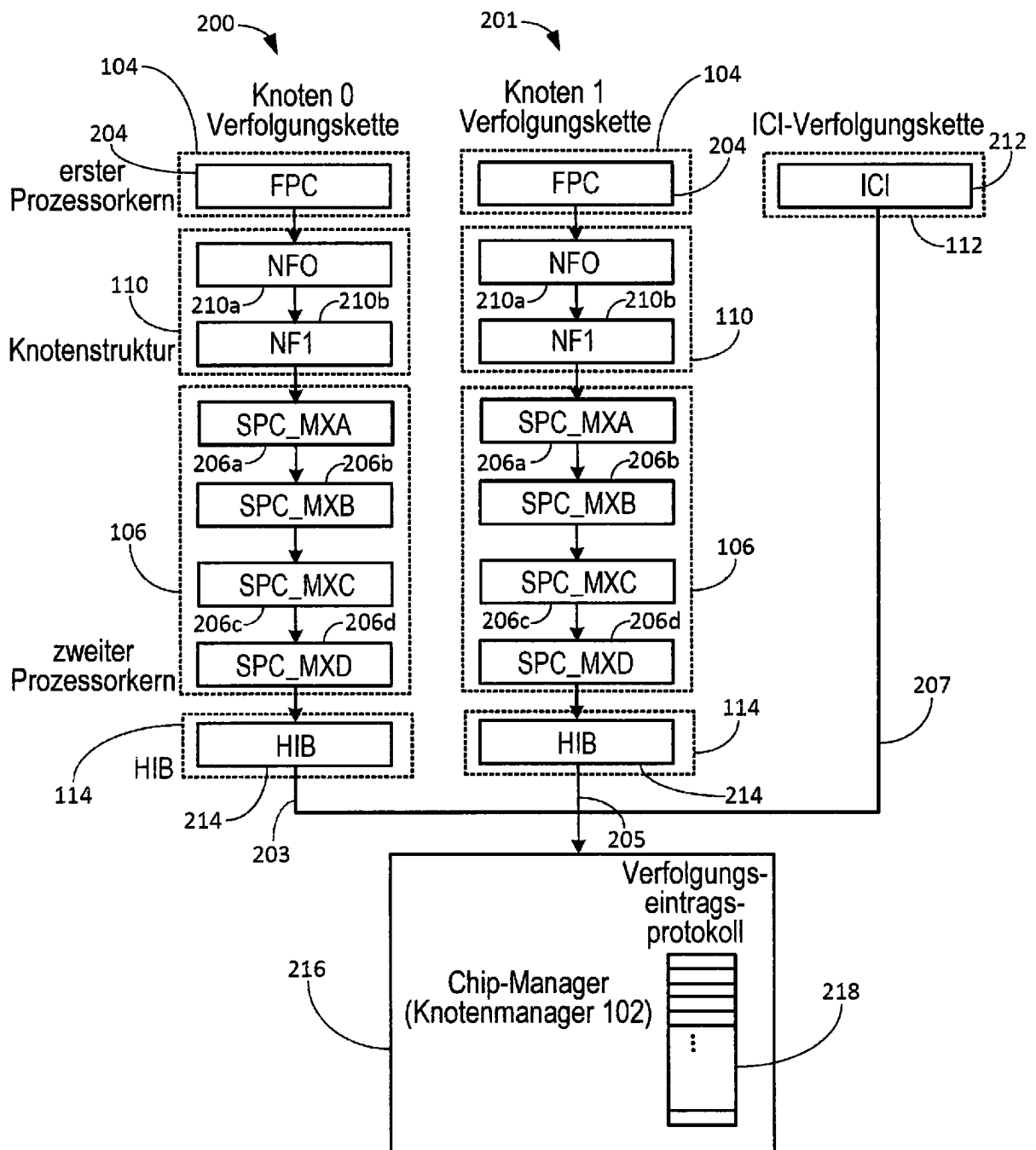
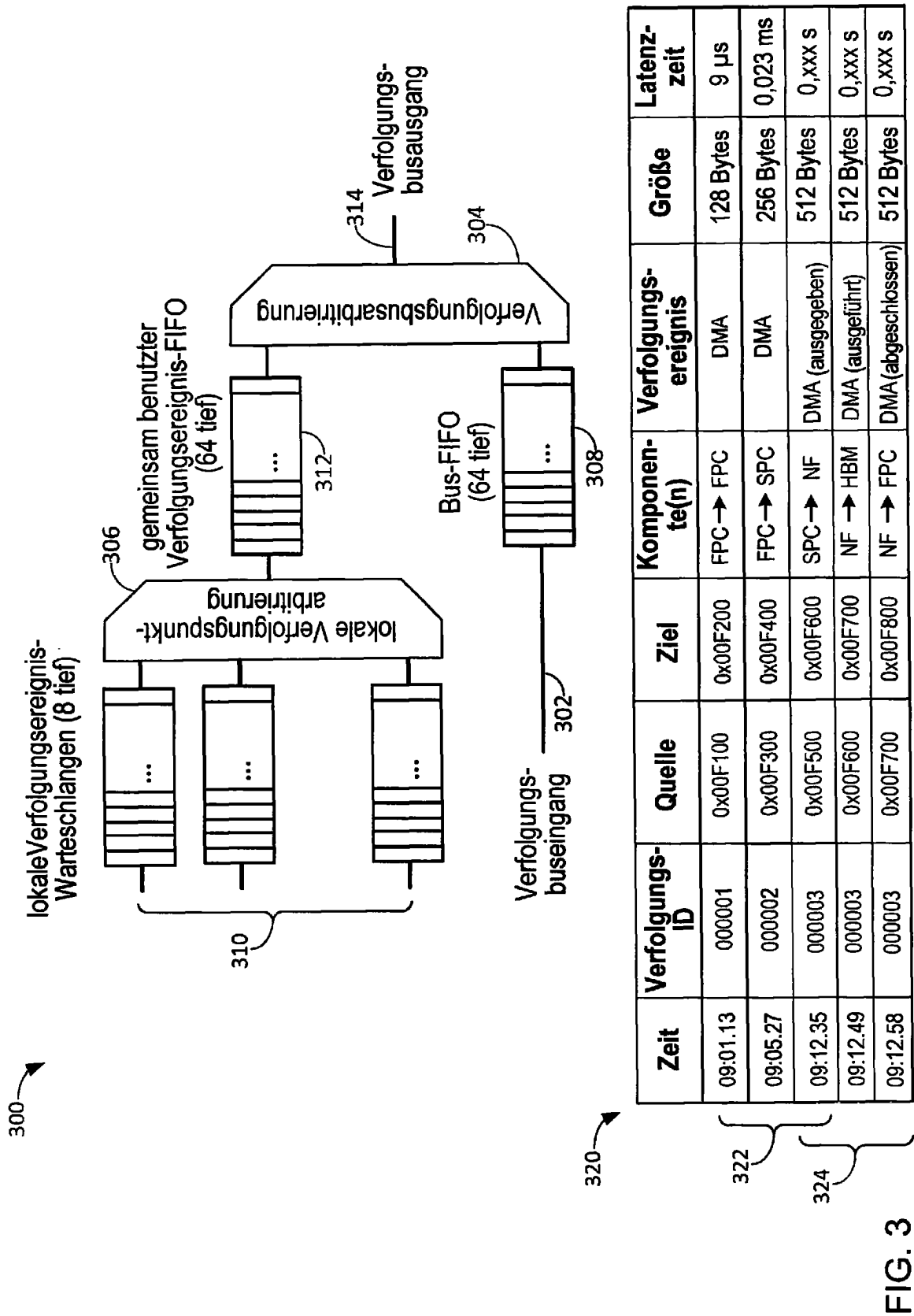
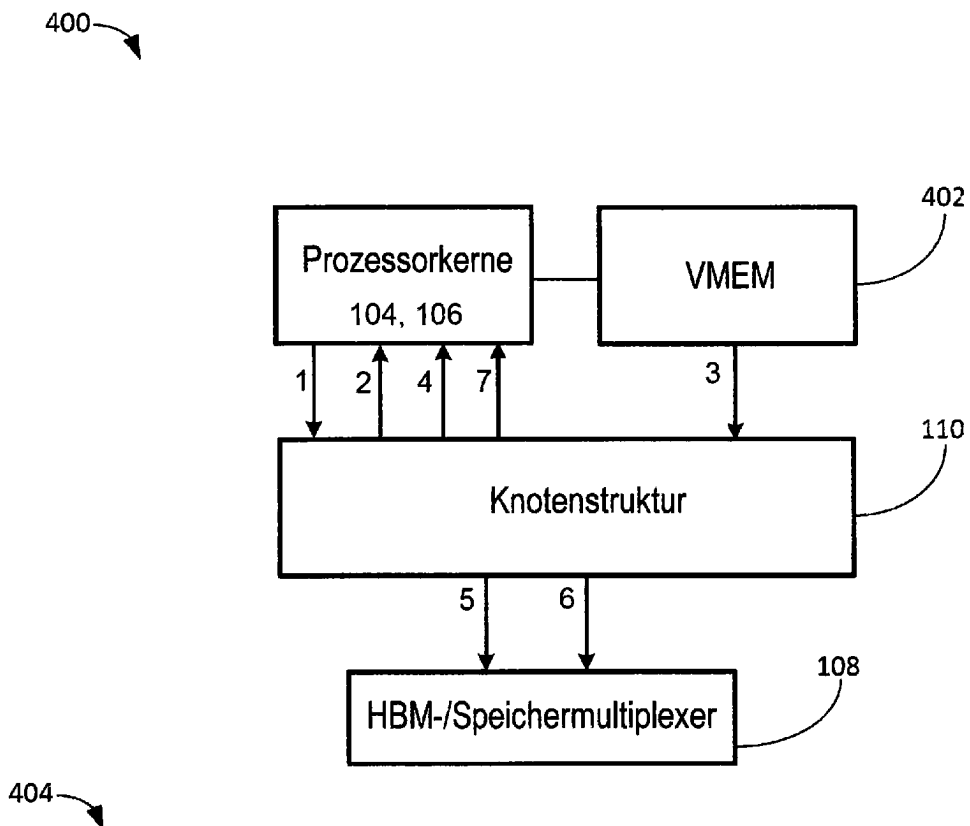


FIG. 2





Schritt	Operation
1	Anfängliche DMA-Anforderung; Verfolgungspunkt in der Knotenstruktur
2	Lesebefehl: Die NF fordert den Kern auf, Daten zu übertragen; Verfolgungspunkt in der NF
3	Lesen abgeschlossen: Hier keinen Verfolgungspunkt in der NF!
4	Lesebetriebsmittelaktualisierung: Synchronisationsmerkeraktualisierung im Kern; Verfolgungspunkt im FPC
5	Schreibbefehl: Die NF benachrichtigt den HBM; Verfolgungspunkt in der NF
6	Schreiben abgeschlossen: Verfolgungspunkt in der NF
7	Schreibbetriebsmittelaktualisierung: Synchronisationsmerkeraktualisierungen im FPC; Verfolgungspunkt im FPC

FIG. 4

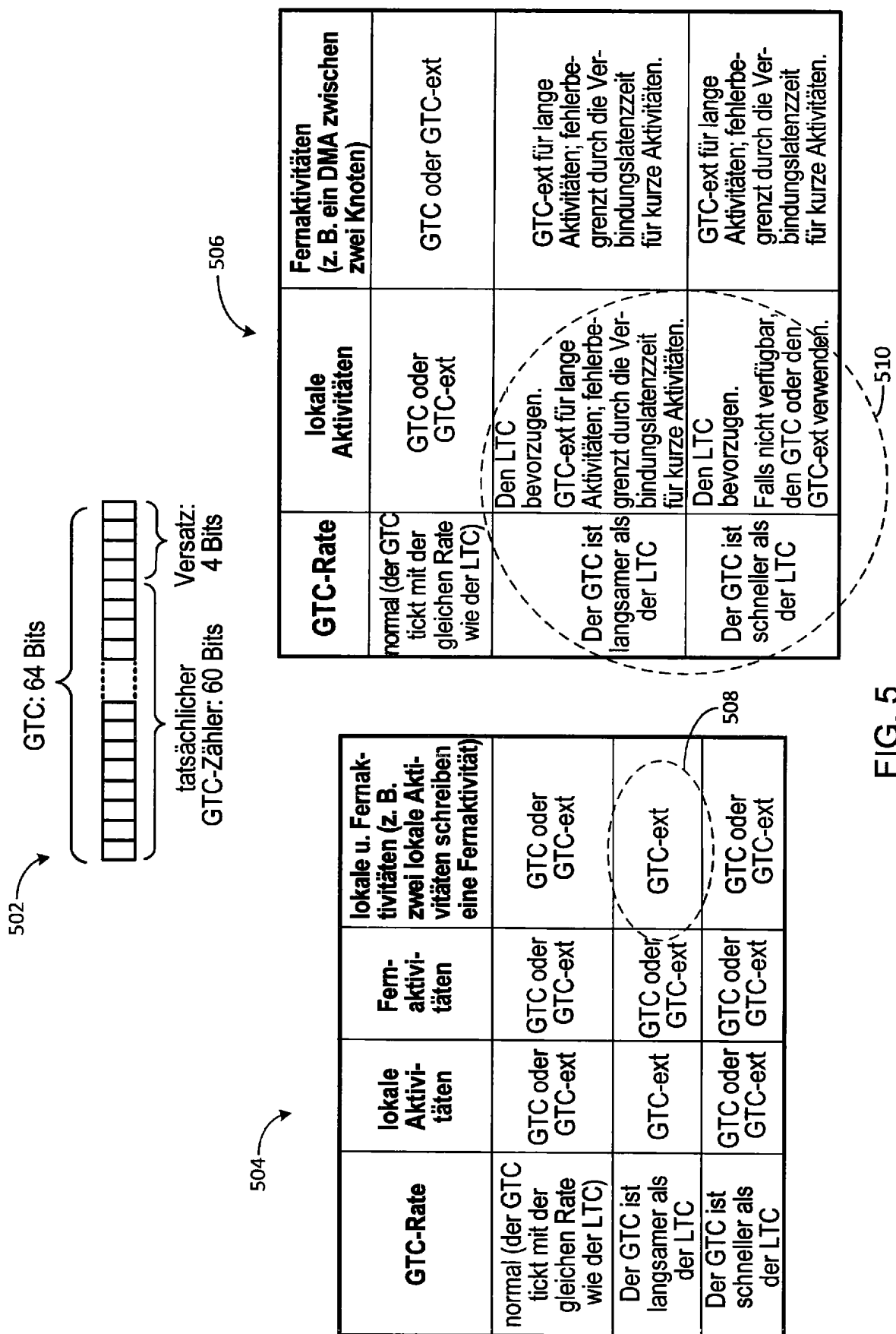
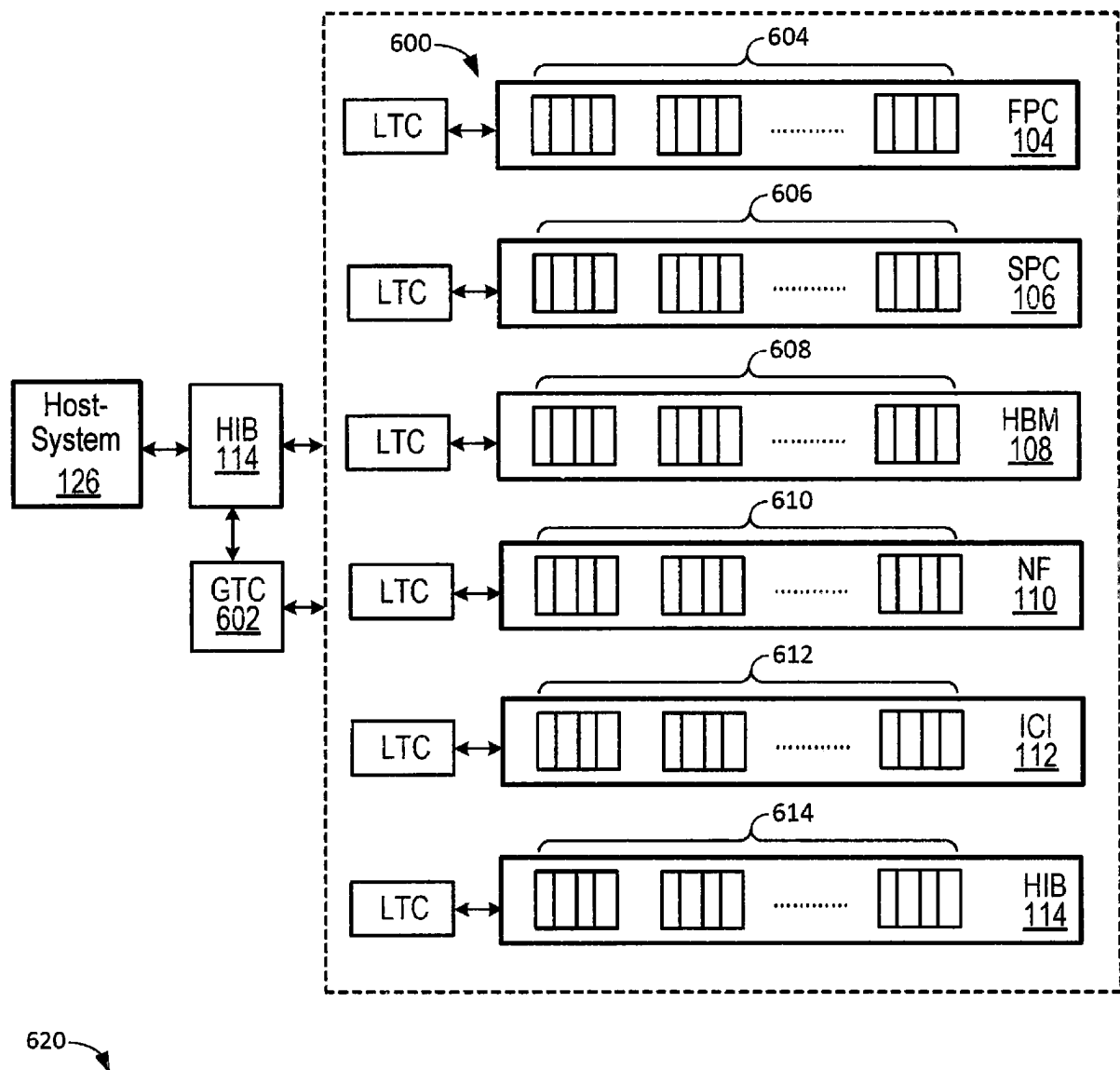


FIG. 5



620

	Zeit	Verfolgungs-ID	Quelle	Ziel	Komponente(n)	Verfolgungsereignis	Ablauf oder TM	Latenzzeit
622	09:01.13	000001	0x00F100	0x00F200	FPC → FPC	DMA	Codeablauf 56	9 μs
	09:01.13	000002	0x00F300	0x00F400	FPC → SPC	DMA	Codeablauf 75	0,023 ms
624	09:15.36	000003	0x00F500	0x00F600	FPC	DMA/Anweisung	Codeablauf 33	0,041 ms
	09:26.19	000004	0x00F700	0x00F800	SPC	DMA/Anweisung	Codeablauf 33	0,062 ms

FIG. 6

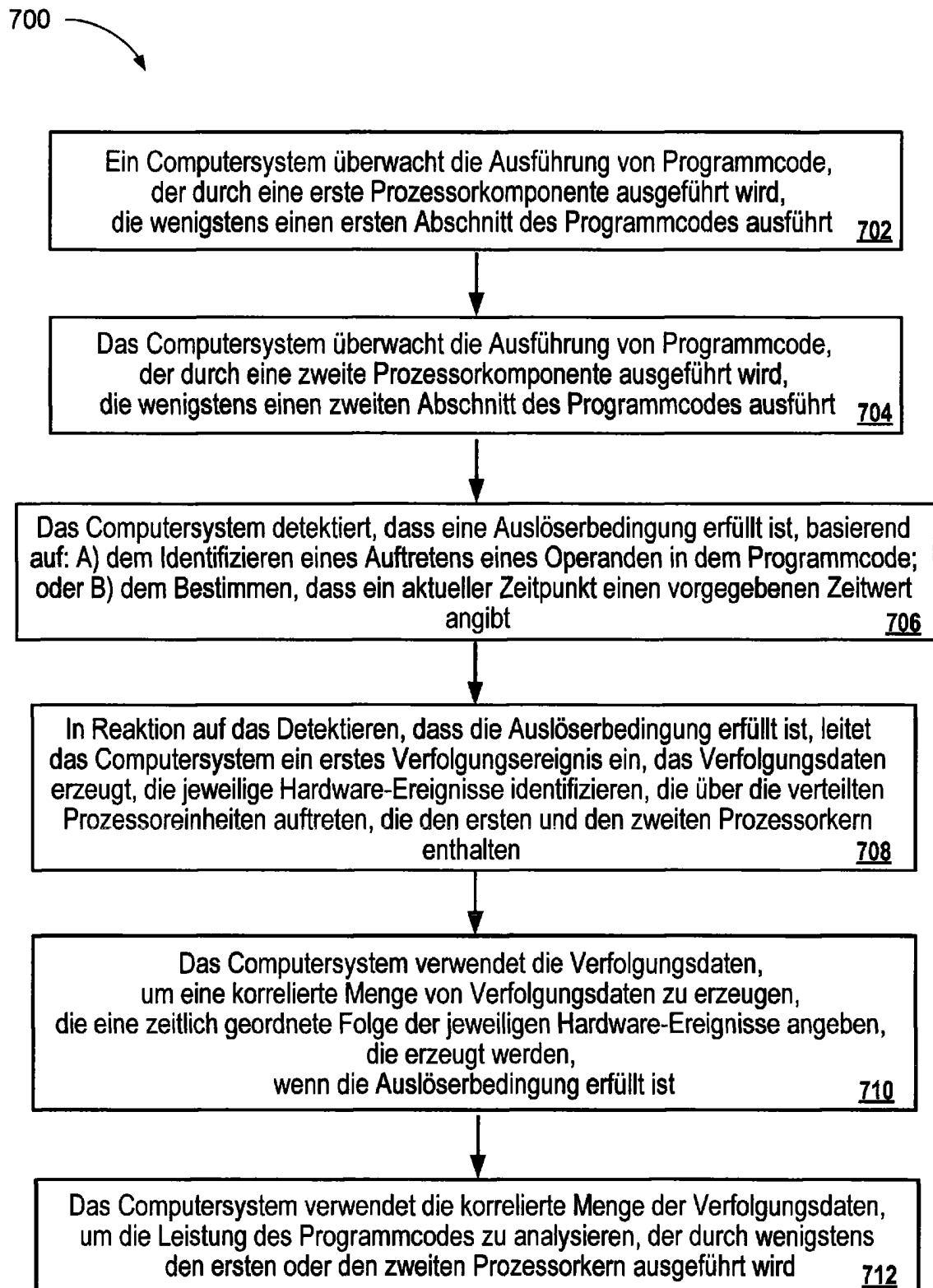


FIG. 7