(19) Deutsches Patent- und Markenamt

(10) **DE 20 2017 105 708 U1** 2018.02.08

(12) # Gebrauchsmusterschrift

(21) Aktenzeichen: **20 2017 105 708.9**
(22) Anmeldetag: **20.09.2017**
(47) Eintragungstag: **03.01.2018**
(45) Bekanntmachungstag im Patentblatt: **08.02.2018**

(51) Int Cl.: ***G06F 17/10*** (2006.01)
G06N 3/04 (2006.01)

(30) Unionspriorität:
15/335,769 **27.10.2016** **US**

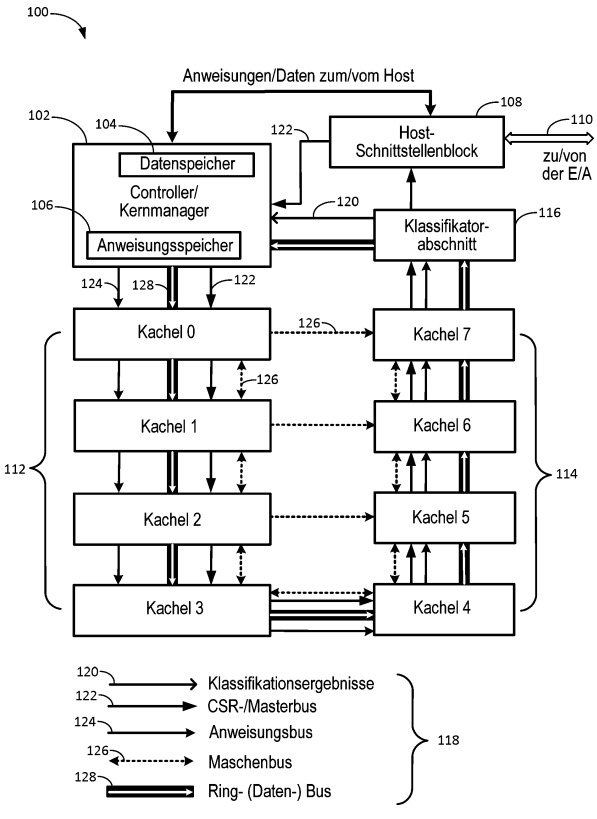
(73) Name und Wohnsitz des Inhabers:
GOOGLE INC., Mountain View, Calif., US

(74) Name und Wohnsitz des Vertreters:
Betten & Resch Patent- und Rechtsanwälte PartGmbB, 80333 München, DE

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen.

(54) Bezeichnung: **Rechenkachel für neuronale Netze**

(57) Hauptanspruch: Computereinheit zum Beschleunigen von Tensorberechnungen, die umfasst:
eine erste Speicherbank, die eine erste Datenbreite aufweist, zum Speichern wenigstens einer der Eingangsaktivierungen oder der Ausgangsaktivierungen;
eine zweite Speicherbank, die eine zweite Datenbreite, die größer als die erste Datenbreite ist, aufweist, zum Speichern eines oder mehrerer Parameter, die beim Ausführen der Berechnungen verwendet werden;
wenigstens eine Zelle, die wenigstens einen Multiplikations-Akkumulations-Operator ("MAC"-Operator) umfasst, der Parameter von der zweiten Speicherbank empfängt und Berechnungen ausführt;
eine erste Traversierungseinheit, die mit wenigstens der ersten Speicherbank in Datenverbindung steht, wobei die erste Traversierungseinheit konfiguriert ist, um der ersten Speicherbank ein Steuersignal bereitzustellen, um zu veranlassen, das eine Eingangsaktivierung einem Datenbus, der durch den MAC-Operator zugänglich ist, bereitgestellt wird; und
wobei die Computereinheit eine oder mehrere Berechnungen ausführt, die wenigstens einem Element eines Datenfeldes zugeordnet sind, wobei die eine oder die mehreren Berechnungen durch den MAC-Operator ausgeführt werden und teilweise eine Multiplikationsoperation der von dem Datenbus empfangenen Eingangsaktivierung und eines von der zweiten Speicherbank empfangenen Parameters umfassen.



Beschreibung

HINTERGRUND

[0001] Diese Beschreibung bezieht sich im Allgemeinen auf eine Rechenkachel für neuronale Netze für die Berechnung von Schichten tiefer neuronaler Netze ("DNN"), die eine verringerte Anweisungsbandbreite und einen verringerten Anweisungsspeicher ermöglicht.

[0002] Unter Schutz gestellt werden und Gegenstand des Gebrauchsmusters sind dabei, entsprechend den Vorschriften des Gebrauchsmustergesetzes, lediglich Vorrichtungen wie in den beigefügten Schutzansprüchen definiert, jedoch keine Verfahren. Soweit nachfolgend in der Beschreibung gegebenenfalls auf Verfahren Bezug genommen wird, dienen diese Bezugnahmen lediglich der beispielhaften Erläuterung der in den beigefügten Schutzansprüchen unter Schutz gestellten Vorrichtung oder Vorrichtungen.

ZUSAMMENFASSUNG

[0003] Im Allgemeinen kann ein innovativer Aspekt des in dieser Beschreibung beschriebenen Gegenstands in einer Computereinheit zum Beschleunigen von Tensorberechnungen verkörpert sein. Die Computereinheit umfasst eine erste Speicherbank, die eine erste Datenbreite zum Speichern wenigstens einer der Eingangsaktivierungen oder der Ausgangsaktivierungen aufweist, und eine zweite Speicherbank, die eine zweite Datenbreite, die größer als die erste Datenbreite ist, zum Speichern eines oder mehrerer Parameter, die beim Ausführen der Berechnungen verwendet werden, aufweist. Die Computereinheit kann ferner wenigstens eine Zelle enthalten, die wenigstens einen Multiplikations-Akkumulations-Operator ("MAC"-Operator) umfasst, der die Parameter von der zweiten Speicherbank empfängt und Berechnungen ausführt. Die Computereinheit kann noch weiter eine erste Traversierungseinheit enthalten, die mit wenigstens der ersten Speicherbank in Datenverbindung steht, wobei die erste Traversierungseinheit konfiguriert ist, um der ersten Speicherbank ein Steuersignal bereitzustellen, um zu verursachen, dass einem Datenbus, der durch den MAC-Operator zugänglich ist, eine Eingangsaktivierung bereitgestellt wird. Die Computereinheit führt eine oder mehrere Berechnungen aus, die wenigstens einem Element eines Datenfeldes zugeordnet sind, wobei die eine oder die mehreren Berechnungen durch den MAC-Operator ausgeführt werden und teilweise eine Multiplikationsoperation der von dem Datenbus empfangenen Eingangsaktivierung und eines von der zweiten Speicherbank empfangenen Parameters umfasst.

[0004] Ein weiterer innovativer Aspekt des in dieser Beschreibung beschriebenen Gegenstands kann in einem computerimplementierten Verfahren zum Beschleunigen von Tensorberechnungen verkörpert sein. Das computerimplementierte Verfahren enthält das Senden durch eine erste Speicherbank, die eine erste Datenbreite aufweist, einer ersten Eingangsaktivierung in Reaktion auf die erste Speicherbank, die ein Steuersignal von einer ersten Traversierungseinheit empfängt, wobei die erste Speicherbank in einer Computereinheit angeordnet ist und wobei die erste Eingangsaktivierung durch einen Datenbus bereitgestellt wird, der durch wenigstens eine Zelle der Computereinheit zugänglich ist. Das Verfahren kann ferner das Empfangen durch die wenigstens eine Zelle eines oder mehrerer Parameter von einer zweiten Speicherbank, die eine zweite Datenbreite, die größer als die erste Datenbreite ist, aufweist, enthalten, wobei die wenigstens eine Zelle wenigstens einen Multiplikations-Akkumulations-Operator ("MAC"-Operator) umfasst. Das Verfahren kann noch weiter das Ausführen durch den MAC-Operator einer oder mehrerer Berechnungen enthalten, die wenigstens einem Element eines Datenfeldes zugeordnet sind, wobei die eine oder die mehreren Berechnungen teilweise eine Multiplikationsoperation wenigstens einer ersten Eingangsaktivierung, auf die von dem Datenbus zugegriffen wird, und wenigstens eines Parameters, der von der zweiten Speicherbank empfangen wird, umfassen.

[0005] Ein weiterer innovativer Aspekt des in dieser Beschreibung beschriebenen Gegenstands kann in einem nicht transitorischen computerlesbaren Speichermedium verkörpert sein. Das nicht transitorische computerlesbare Speichermedium umfasst Anweisungen, die durch einen oder mehrere Prozessoren ausführbar sind, die bei einer derartigen Ausführung den einen oder die mehreren Prozessoren veranlassen, Operationen auszuführen, die das Senden durch eine erste Speicherbank, die eine erste Datenbreite aufweist, einer ersten Eingangsaktivierung in Reaktion auf die erste Speicherbank, die ein Steuersignal von einer ersten Traversierungseinheit empfängt, umfassen, wobei die erste Speicherbank in einer Computereinheit angeordnet ist und wobei die erste Eingangsaktivierung durch einen Datenbus bereitgestellt wird, der durch wenigstens eine Zelle der Computereinheit zugänglich ist. Die ausgeführten Operationen können außerdem das Empfangen durch die wenigstens eine Zelle eines oder mehrerer Parameter von einer zweiten Speicherbank, die eine zweite Datenbreite, die größer als die erste Datenbreite ist, aufweist, enthalten, wobei die wenigstens eine Zelle wenigstens einen Multiplikations-Akkumulations-Operator ("MAC"-Operator) umfasst. Die ausgeführten Operationen können außerdem ferner das Ausführen durch den MAC-Operator einer oder mehrerer Berechnungen enthalten, die wenigstens einem Element eines Datenfeldes zugeordnet sind, wobei

die eine oder die mehreren Berechnungen teilweise eine Multiplikationsoperation wenigstens der ersten Eingangsaktivierung, auf die von dem Datenbus zugegriffen wird, und wenigstens eines Parameters, der von der zweiten Speicherbank empfangen wird, umfassen.

[0006] Der in dieser Beschreibung beschriebene Gegenstand kann in speziellen Ausführungsformen implementiert sein, um einen oder mehrere der folgenden Vorteile zu verwirklichen. Die Verwendung von Registern, um die Speicheradressenwerte zu verfolgen, erlaubt es einem Programm, tief verschachtelte Schleifen mit einer Anweisung zu iterieren. Ein von schmalen Speichereinheiten und breiten Speichereinheiten zugänglicher Tensor wird basierend auf den von den Registern wiedergewonnenen Speicheradressenwerten in einer einzigen Rechenkachel traversiert. Die Speicheradressenwerte entsprechen den Elementen des Tensors. Die Tensorberechnungen finden in einzelnen Rechenkacheln basierend auf der Ausführung tiefer Schleifenschachtelungen statt. Die Berechnungen können über mehrere Kacheln verteilt sein. Die Recheneffizienz wird basierend auf dem Verteilen der Tensorberechnungen für ein mehrschichtiges neuronales Netz über mehrere Rechenkacheln verbessert und beschleunigt. Die Tensoren können traversiert werden und die Tensorberechnungen können mit einer verringerten Anzahl von Anweisungen ausgeführt werden.

[0007] Der in dieser Beschreibung beschriebene Gegenstand kann in speziellen Ausführungsformen implementiert sein, um andere Vorteile zu verwirklichen. Durch das Verwenden einer Speicherhierarchie, die einen Speicher mit schmaler geringer Bandbreite, der eine Adressierungsflexibilität erlaubt, um ein mehrdimensionales Feld in irgendeiner Reihenfolge zu traversieren, mit einem Speicher mit hoher breiter Bandbreite koppelt, kann z. B. eine hohe Verwendung der MAC-Operatoren für DNN-Schichten mit sehr verschiedenen Dimensionen erreicht werden und kann eine Lokalität bei der Berechnung maximal ausgenutzt werden.

[0008] Andere Implementierungen dieser und anderer Aspekte enthalten entsprechende Systeme, Vorrichtungen und Computerprogramme, die konfiguriert sind, die Vorgänge der Verfahren auszuführen, die in Computer-Speichervorrichtungen codiert sind. Ein System aus einem oder mehreren Computern kann aufgrund von Software, Firmware, Hardware oder einer Kombination aus ihnen, die in dem System installiert sind und die in Betrieb das System veranlassen, die Vorgänge auszuführen, so konfiguriert sein. Ein oder mehrere Computerprogramme können aufgrund dessen, dass sie Anweisungen aufweisen, die, wenn sie durch eine Datenverarbeitungsvorrichtung ausgeführt werden, die Vorrichtung veranlassen, die Vorgänge auszuführen, so konfiguriert sein.

[0009] Der in dieser Beschreibung beschriebene Gegenstand bezieht sich außerdem auf ein Bilderkennungs- und/oder Klassifikationsverfahren/-system. Das System kann (die Systeme können) unter Verwendung der offenbarten Techniken und des beschriebenen Hardware-Computersystems, das die Hardware-Computereinheiten oder die Hardware-Rechenkacheln aufweist, implementiert sein. Die Computereinheiten verarbeiten Tensorberechnungen zum Berechnen von Folgerungen unter Verwendung eines neuronalen Netzes, das mehrere Schichten des neuronalen Netzes aufweist.

[0010] Die Einzelheiten einer oder mehrerer Implementierungen des in dieser Beschreibung beschriebenen Gegenstands sind in den beigefügten Zeichnungen und der Beschreibung im Folgenden dargelegt. Andere potentielle Merkmale, Aspekte und Vorteile des Gegenstands werden aus der Beschreibung, den Zeichnungen und den Ansprüchen ersichtlich.

KURZBESCHREIBUNG DER ZEICHNUNGEN

[0011] **Fig.** 1 ist ein Blockschaltplan eines beispielhaften Computersystems.

[0012] **Fig.** 2 veranschaulicht eine beispielhafte Rechenkachel eines neuronalen Netzes.

[0013] **Fig.** 3 veranschaulicht eine beispielhafte Struktur einer Tensortraversierungseinheit (TTU).

[0014] **Fig.** 4 veranschaulicht eine beispielhafte Architektur, die eine schmale Speichereinheit enthält, die einem oder mehreren Multiplikations-Akkumulations-Operatoren (MAC-Operatoren) Eingangsaktivierungen bereitstellt.

[0015] **Fig.** 5 veranschaulicht eine beispielhafte Architektur, die einen Ausgangsbus enthält, der der schmalen Speichereinheit nach **Fig.** 2 und **Fig.** 4 Ausgangsaktivierungen bereitstellt.

[0016] **Fig.** 6 ist ein beispielhafter Ablaufplan eines Prozesses zum Ausführen von Tensorberechnungen unter Verwendung der Rechenkachel eines neuronalen Netzes nach **Fig.** 2.

[0017] Gleiche Bezugszeichen und Bezeichnungen in den verschiedenen Zeichnungen geben gleiche Elemente an.

AUSFÜHRLICHE BESCHREIBUNG

[0018] Der in dieser Beschreibung beschriebene Gegenstand bezieht sich auf ein Hardware-Computersystem, das mehrere Computereinheiten enthält, die konfiguriert sind, die Arbeitsbelastungen der Folgerungen des maschinellen Lernens einer

Schicht eines neuronalen Netzes zu beschleunigen. Jede Computereinheit des Hardware-Computersystems ist in sich geschlossen und kann die durch eine gegebene Schicht eines mehrschichtigen neuronalen Netzes benötigten Berechnungen unabhängig ausführen.

[0019] Ein neuronales Netz mit mehreren Schichten kann verwendet werden, um Folgerungen zu berechnen. Für eine gegebene Eingabe kann das neuronale Netz z. B. eine Folgerung für die Eingabe berechnen. Das neuronale Netz berechnet diese Folgerung durch das Verarbeiten der Eingabe durch jede der Schichten des neuronalen Netzes. Insbesondere weist jede der Schichten des neuronalen Netzes einen jeweiligen Satz von Gewichten auf. Jede Schicht empfängt eine Eingabe und verarbeitet die Eingabe in Übereinstimmung mit den Satz von Gewichten für die Schicht, um eine Ausgabe zu erzeugen.

[0020] Um eine Folgerung aus einer empfangenen Eingabe zu berechnen, empfängt deshalb das neuronale Netz die Eingabe, wobei es sie durch jede der Schichten des neuronalen Netzes verarbeitet, um die Folgerung zu erzeugen, wobei die Ausgabe von einer Schicht des neuronalen Netzes als die Eingabe in die nächste Schicht des neuronalen Netzes bereitgestellt wird. Die Dateneingaben in eine Schicht des neuronalen Netzes, z. B. entweder die Eingabe in das neuronale Netz oder die Ausgaben der Schicht unter der Schicht in der Folge in eine Schicht des neuronalen Netzes, können als die Aktivierungseingaben in die Schicht bezeichnet werden.

[0021] In einigen Implementierungen sind die Schichten des neuronalen Netzes in einer Folge angeordnet. In anderen Implementierungen sind die Schichten in einem gerichteten Graph angeordnet. Das heißt, jede spezielle Schicht kann mehrere Eingaben, mehrere Ausgaben oder beides empfangen. Die Schichten des neuronalen Netzes können außerdem so angeordnet sein, dass eine Ausgabe einer Schicht als eine Eingabe zu einer vorhergehenden Schicht zurückgeschickt werden kann.

[0022] Das in dieser Beschreibung beschriebene Hardware-Computersystem kann die Berechnung einer Schicht des neuronalen Netzes durch das Verteilen der Tensorberechnungen über mehrere Rechenkacheln ausführen. Ein innerhalb einer Schicht des neuronalen Netzes ausgeführter Rechenprozess kann eine Multiplikation eines Eingangstensors, der Eingangsaktivierungen enthält, mit einem Parametertensor, der Gewichte enthält, enthalten. Die Berechnung enthält das Multiplizieren einer Eingangsaktivierung mit einem Gewicht in einem oder mehreren Zyklen und das Ausführen einer Akkumulation der Produkte über viele Zyklen.

[0023] Ein Tensor ist ein mehrdimensionales geometrisches Objekt, wobei beispielhafte mehrdimensionale geometrische Objekte Matrizen und Datenfelder enthalten. Im Allgemeinen wird ein Software-Algorithmus durch eine Rechenkachel ausgeführt, um Tensorberechnungen durch das Verarbeiten einer verschachtelten Schleife, um einen N-dimensionalen Tensor zu traverisieren, auszuführen. In einem beispielhaften Rechenprozess kann jede Schleife für das Traversieren einer speziellen Dimension des N-dimensionalen Tensors verantwortlich sein. Für ein gegebenes Tensorkonstrukt kann eine Rechenkachel Zugriff auf ein Element eines speziellen Tensors erfordern, um mehrere Skalarproduktberechnungen, die dem Tensor zugeordnet sind, auszuführen. Die Berechnung findet statt, wenn eine durch eine schmale Speicherstruktur bereitgestellte Eingangsaktivierung mit einem durch eine breite Speicherstruktur bereitgestellten Parameter oder Gewicht multipliziert wird. Weil der Tensor in einem Speicher gespeichert ist, kann ein Satz von Tensorindizes eine Übersetzung in einen Satz von Speicheradressen erfordern. Im Allgemeinen führt eine Tensortraversierungseinheit einer Rechenkachel Steueroperationen aus, die den Index jeder dem Tensor zugeordneten Dimension und die Reihenfolge, in der Indexelemente traversiert werden, um die Berechnungen auszuführen, bereitstellen. Die Tensorberechnungen enden, wenn die Multiplikationsergebnisse auf einen Ausgangsbus geschrieben und im Speicher gespeichert sind.

[0024] **Fig.** 1 zeigt einen Blockschaltplan eines beispielhaften Computersystems **100** zum Beschleunigen von Tensorberechnungen, die tiefen neuronalen Netzen (DNNs) zugeordnet sind. Das System **100** enthält im Allgemeinen einen Controller **102**, eine Host-Schnittstelle **108**, eine Eingabe-/Ausgabeverbindung (E/A-Verbindung) **110**, mehrere Kacheln, die einen ersten Kachelsatz **112** und einen zweiten Kachelsatz **114** enthalten, einen Klassifikatorabschnitt **116** und die Datenbusse, die in einer Busabbildung **118** (die für die Klarheit gezeigt ist, aber in dem System **100** nicht enthalten ist) identifiziert sind. Der Controller **102** enthält im Allgemeinen einen Datenspeicher **104**, einen Anweisungsspeicher **106** und wenigstens einen Prozessor, der konfiguriert ist, um eine oder mehrere Anweisungen, die in einem computerlesbaren Speichermedium codiert sind, auszuführen. Der Anweisungsspeicher **106** kann eine oder mehrere maschinenlesbare Anweisungen speichern, die durch den einen oder die mehreren Prozessoren des Controllers **102** ausführbar sind. Der Datenspeicher **104** kann irgendeines von verschiedenen Datenspeichermedien zum Speichern und anschließenden Zugreifen auf verschiedene Daten bezüglich der Berechnungen, die innerhalb des Systems **100** stattfinden, sein.

[0025] Der Controller **102** ist konfiguriert, um eine oder mehrere Anweisungen bezüglich der Tensorberechnungen innerhalb des Systems **100** einschließlich der im Anweisungsspeicher **106** gespeicherten Anweisungen auszuführen. In einigen Implementierungen sind der Datenspeicher **104** und der Anweisungsspeicher **106** eine flüchtige Speichereinheit oder flüchtige Speichereinheiten. In einigen anderen Implementierungen sind der Datenspeicher **104** und der Anweisungsspeicher **106** eine nichtflüchtige Einheit oder nichtflüchtige Einheiten. Der Datenspeicher **104** und der Anweisungsspeicher **106** können eine weitere Form eines computerlesbaren Mediums, wie z. B. eine Diskettenvorrichtung, eine Festplattenvorrichtung, eine Vorrichtung optischer Platten oder eine Bandvorrichtung, ein Flash-Speicher oder eine andere ähnliche Festkörper-Speichervorrichtung oder eine Anordnung von Vorrichtungen einschließlich Vorrichtungen in einem Speicherbereichsnetz oder anderer Konfigurationen, sein. In verschiedenen Implementierungen kann außerdem auf den Controller **102** als Kernmanager **102** Bezug genommen werden oder der Controller **102** als Kernmanager **102** bezeichnet werden.

[0026] Wie dargestellt ist, ist die Host-Schnittstelle **108** an die E/A-Verbindung **110**, den Controller **102** und den Klassifikatorabschnitt **116** gekoppelt. Die Host-Schnittstelle **108** empfängt Anweisungen und Datenparameter von der E/A-Verbindung **110** und stellt die Anweisungen und Parameter dem Controller **102** bereit. Im Allgemeinen können die Anweisungen durch den (im Folgenden beschriebenen) Anweisungsbus **124** einer oder mehreren Vorrichtungen in dem System **100** bereitgestellt werden und können die Parameter durch einen (im Folgenden beschriebenen) Ringbus **128** einer oder mehreren Vorrichtungen in dem System **100** bereitgestellt werden. In einigen Implementierungen werden die Anweisungen zu einem Anfangszeitpunkt durch den Controller **102** von der Host-Schnittstelle **118** empfangen und für die Ausführung durch den Controller **102** zu einem späteren Zeitpunkt im Anweisungsspeicher **106** gespeichert.

[0027] Der Klassifikatorabschnitt **116** ist gleichermaßen an den Controller **102** und die Kachel **7** des zweiten Kachelsatzes **114** gekoppelt. In einigen Implementierungen ist der Klassifikatorabschnitt **116** als eine separate Kachel innerhalb des Systems **100** implementiert. In alternativen Implementierungen ist der Klassifikatorabschnitt **116** innerhalb des Controllers **102** als eine Unterschaltung oder eine Untervorrichtung des Controllers **102** angeordnet oder befindlich. Der Klassifikatorabschnitt **116** ist im Allgemeinen konfiguriert, eine oder mehrere Funktionen an akkumulierten Voraktivierungswerten auszuführen, die als die Ausgaben der vollständig verbundenen Schichten empfangen werden. Die vollständig verbundenen Schichten können über die Kacheln in den Kachelsätzen **112** und **114** partitioniert sein. Folglich ist jede Kachel konfiguriert, um eine Teilmenge der Voraktivierungswerte (d. h., der linearen Ausgaben) zu erzeugen, die in einer Speichereinheit(en) der Kachel gespeichert sein können. Der Klassifikationsergebnisbus **120** schafft einen Datenweg von dem Klassifikatorabschnitt **116** bis zum Controller **102**. Die Daten, die Nachfunktionswerte (d. h., Ergebnisse) enthalten, werden von dem Klassifikatorabschnitt **116** über den Klassifikationsergebnisbus **120** dem Controller **102** bereitgestellt.

[0028] Die Busabbildung **118** zeigt die Datenbusse, die einen oder mehrere miteinander verbundene Datenkommunikationswege zwischen den Kacheln des ersten Kachelsatzes **112** und des zweiten Kachelnsatzes **114** schaffen. Die Busabbildung **118** stellt eine Legende zum Identifizieren eines Klassifikationsergebnisbusses **120**, eines CSR-/Masterbusses **122**, eines Anweisungsbusses **124**, eines Maschenbusses **126** und eines Ringbusses **128** bereit, wie in **Fig.** 1 dargestellt ist. Im Allgemeinen ist eine Kachel eine Kernkomponente innerhalb der Beschleunigerarchitektur des Systems **100**, wobei sie ein Brennpunkt für die Tensorberechnungen ist, die in dem System stattfinden. Jede Kachel ist eine einzelne Computereinheit, wobei mehrere Kacheln mit anderen Kacheln in dem System wechselwirken können, um die Berechnungen (z. B. die Tensorberechnungen) über eine oder mehrere Schichten eines mehrschichtigen neuronalen Netzes zu beschleunigen. Die Berechnungen können z. B. über mehrere Kacheln verteilt sein. Die Recheneffizienz kann basierend auf dem Verteilen der Tensorberechnungen für ein mehrschichtiges neuronales Netz über mehrere Rechenkacheln verbessert und beschleunigt werden. Obwohl die Kacheln in den Kachelsätzen **112**, **114** die Ausführung der einer gegebenen Anweisung zugeordneten Tensorberechnungen teilen können, ist eine einzelne Recheneinheit eine in sich geschlossene Rechenkomponente, die konfiguriert ist, um eine Teilmenge der Tensorberechnungen unabhängig bezüglich anderer entsprechender Kacheln innerhalb der Kachelsätze **112**, **114** auszuführen.

[0029] Der Steuer- u. Statusregisterbus (CSR-Bus) **122** ist ein Einzelner-Mastermehrere-Slaves-Bus, der es dem Controller **102** ermöglicht, eine oder mehrere Anweisungen zu senden, die die Programmkonfigurationen und die Lesestatusregister, die einer oder mehreren Kacheln zugeordnet sind, setzen. Der CSR-Bus **122** kann mit einem Master-Bus-Segment und mehreren Slave-Bus-Segmenten in einer Daisy-Chain-Konfiguration verbunden sein. Wie in **Fig.** 1 gezeigt ist, schafft der CSR-Bus **122** eine Kommunikationskopplung durch einen Busdatenweg, der die Kacheln in den Kachelsätzen **112**, **114** und den Controller **102** in einem Ring mit der Host-Schnittstelle **110** verbindet. In einigen Implementierungen ist die Host-Schnittstelle **110** der einzige Master des CSR-

Busrings und ist der gesamte CSR-Bus-Adressenraum auf einen Speicherraum in der Host-Schnittstelle **110** speicherabgebildet.

[0030] Der CSR-Bus **122** kann durch die Host-Schnittstelle **110** verwendet werden, um eine oder mehrere Operationen auszuführen, die z. B. das Programmieren von Speicherpufferzeigern im Controller **102**, um es dem Controller **102** zu ermöglichen, das Holen von Anweisungen aus dem Anweisungsspeicher **106** zu beginnen, das Aktualisieren/Programmieren verschiedener Kacheleinstellungen (z. B. der Koeffiziententabellen für die Berechnungen der Polynomialapproximationen), die während einer oder mehreren Berechnungen statisch bleiben, und/oder das Laden/erneute Laden der Firmware in den Klassifikationsabschnitt **116** enthalten. In einem Beispiel können die Neueinspielungen neue Funktionen enthalten, die auf die linearen Ausgaben (d. h., die Voraktivierungswerte) angewendet werden. Entsprechend weist jeder Slave, der Zugriff auf den CSR-Bus **122** aufweist, eine charakteristische Knotenkennung (Knoten-ID) auf, die an den Slave gebunden ist und ihn identifiziert. Die Knoten-ID ist ein Teil einer Anweisungsadresse und wird durch die CSR-Slaves (d. h., den Controller **102**, die Kacheln **112**, **114** und den Klassifikator **116**) verwendet, geprüft oder anderweitig untersucht, um zu bestimmen, ob das CSR-Paket an den Slave adressiert ist.

[0031] In einigen Implementierungen können eine oder mehrere Anweisungen durch die Host-Schnittstelle **102** durch den Controller **102** gesendet werden. Die Anweisungen können z. B. 32 Bits breit sein, wobei die ersten 7 Bits die Kopfinformationen enthalten, die die Adresse/das Ziel der Anweisung angeben, das die Anweisungen empfangen und ausführen soll. Die ersten 7 Bits des Kopfs können Datenparameter enthalten, die eine spezielle Knoten-ID repräsentieren. Die Slaves (z. B. jede Kachel) auf dem CSR-Busring können deshalb den Kopf der Anweisung prüfen, um zu bestimmen, ob die Anforderung durch den Master (die Host-Schnittstelle **110**) an die Kachel adressiert war, die den Kopf prüft. Falls die Knoten-ID des Kopfs nicht angibt, dass das Ziel die prüfende Kachel ist, kopiert die prüfende Kachel das Eingangs-CSR-Anweisungspaket zu dem CSR-Buseingang, der mit der nächsten Kachel verbunden ist, für die Prüfung durch die nächste Kachel.

[0032] Der Anweisungsbus **124** geht von dem Controller **102** aus und schafft außerdem ähnlich zu dem CSR-Bus **122** eine Kommunikationskopplung durch einen Busdatenweg, der die Kacheln in den Kachelsätzen **112**, **114** in einem Ring zurück zum Controller **102** verbindet. In einer Implementierung sendet der Controller **102** eine oder mehrere Anweisungen über den Anweisungsbus **124**. Die Anweisungen, die durch den Controller **102** rundgesendet werden, können sich von den über den CSR-Bus **122** bereitgestellten Anweisungen unterscheiden. Die Weise, in der eine Kachel die über den Bus **124** empfangene Anweisung empfängt und/oder verbraucht oder ausführt, kann jedoch zu dem Prozess zum Ausführen der über den CSR-Bus **122** empfangenen Anweisungen ähnlich sein.

[0033] In einem Beispiel gibt ein Kopf (d. h., ein Bitmuster) der Anweisung einer empfangenden Kachel an, dass die empfangende Kachel eine spezielle Anweisung basierend auf einem Bitmuster, das der Anweisung zugeordnet ist, verbrauchen muss. Das Bitmuster kann eine spezielle Breite aufweisen, die in Form von Bits definiert ist. Die Anweisung wird typischerweise basierend auf den Parametern der Anweisung von einer Kachel auf die nächste Kachel weitergeleitet. In einer Implementierung kann die Breite des Anweisungsbusses **124** konfiguriert sein, so dass sie kleiner als die Größe/Breite der Anweisung ist. Folglich geschieht in einer derartigen Konfiguration die Übertragung der Anweisungen über mehrere Zyklen, wobei die Busstopps des Anweisungsbusses **124** Decodierer aufweisen, um die an der Kachel empfangenen Anweisungen in den geeigneten Zielanweisungspuffer, der dieser Kachel zugeordnet ist, zu legen.

[0034] Wie im Folgenden weiter beschrieben wird, sind die Kacheln in den Kachelsätzen **112**, **114** im Allgemeinen konfiguriert, um zwei umfassende Kategorien von Anweisungen zu unterstützen. Die beiden umfassenden Kategorien können außerdem als Anweisungstypen bezeichnet werden. Die Anweisungstypen enthalten eine Tensoroperationsanweisung (TensorOp-Anweisung) und eine Speicherdirektzugriffsanweisung (DMAOp-Anweisung). In einigen Implementierungen weisen die DMAOp-Anweisungen eine oder mehrere Spezialisierungen auf, für die es zulässig ist, dass sie gleichzeitig sind. Die eine oder die mehreren Spezialisierungen können als Subtypen oder Opcodes der DMAOp-Anweisung bezeichnet werden. In einigen Fällen weist jedes eindeutige und/oder gültige Tupel des Typs/Untertyps der DMAOp-Anweisung einen separaten Anweisungspuffer innerhalb einer speziellen Kachel auf.

[0035] An einer speziellen Kachel der Kacheln **112**, **114** untersucht der dem Anweisungsbus **124** zugeordnete Busstopp das Kopf-Bitmuster, um den Typ/Untertyp der Anweisung zu bestimmen. Die Anweisung kann durch die Kachel empfangen und anschließend vor der Ausführung der Anweisung durch die Kachel in einen Anweisungspuffer der Kachel geschrieben werden. Der Anweisungspuffer der Kachel, in den die Anweisung geschrieben wird, kann durch den Indikator/das Feld des Typs und des Untertyps der Anweisung bestimmt sein. Die Anweisungspuffer können ein erstes First-in-first-out-Steuerschema (FIFO-Steuerschema) enthalten, das den Verbrauch einer oder mehrerer in Beziehung stehender Anwei-

sungen priorisiert. Folglich werden gemäß diesem FIFO-Steuerschema die Anwendungen des gleichen Typs/Untertyps immer in der Reihenfolge ausgeführt, in der die Anweisung an dem Anweisungsbus angekommen ist.

[0036] Die verschiedenen Anweisungspuffer innerhalb einer Kachel sind die TensorOp-Anweisungspuffer und die DMAOp-Anweisungspuffer. Wie oben angegeben worden ist, enthalten die Anweisungstypen die TensorOp-Anweisung und die DMAOp-Anweisung. Bezüglich der DMAOp-Anweisungen enthalten die Anweisungsuntertypen (die einen 'Schreibe-in'-Pufferort angeben) die Folgenden: 1) einen Maschenpuffer für eingehende Anweisungen; 2) einen Maschenpuffer für abgehende Anweisungen; 3) einen Puffer für Schmal-breit-DMA-Anweisungen; 4) einen Puffer für Breit-schmal-DMA-Anweisungen; und 5) einen Puffer für Ringbus-DMA-Anweisungen. Diese Pufferorte werden im Folgenden bezüglich **Fig.** 2 ausführlicher beschrieben. Die Bezeichnungen breit und schmal werden überall in der Beschreibung verwendet und beziehen sich im Allgemeinen auf eine annähernde Größe in der Breite (Bits/Bytes) einer oder mehrerer Speichereinheiten. Wie "schmal" hier verwendet wird, kann es sich auf eine oder mehrere Speichereinheiten beziehen, von denen jede eine Größe oder Breite von weniger als 16 Bits aufweist, während sich "breit" auf eine oder mehrere Speichereinheiten beziehen kann, von denen jede eine Größe oder Breite von weniger als 64 Bits aufweist.

[0037] Der (im Folgenden beschriebene) Maschenbus **126** schafft einen Datenkommunikationsweg, der von dem CSR-Bus **122**, dem Anweisungsbus **124** und dem Ringbus **128** verschieden ist. Wie in **Fig.** 1 dargestellt ist, schafft der Maschenbus **126** einen Kommunikationsweg, der jede Kachel sowohl in der X- als auch in der Y-Dimension mit ihrer entsprechenden Nachbarkachel koppelt oder verbindet. In verschiedenen Implementierungen kann der Maschenbus **126** verwendet werden, um Eingangsaktivierungsgrößen zwischen einer oder mehreren schmalen Speichereinheiten in benachbarten Kacheln zu transportieren. Wie gezeigt ist, erlaubt der Maschenbus **126** keine direkte Weiterleitung von Eingangsaktivierungsdaten zu nicht benachbarten Kacheln.

[0038] In verschiedenen Implementierungen können der Maschenbus **126** und die über den Maschenbus **126** verbundenen verschiedenen Kacheln die folgende Konfiguration aufweisen. Die vier Eckkacheln der Masche weisen zwei abgehende Anschlüsse und zwei eingehende Anschlüsse auf. Die vier Randkacheln der Masche weisen drei eingehende Anschlüsse und drei abgehende Anschlüsse auf. Alle Nicht-Rand-nicht-Ecken-Kacheln weisen vier eingehende Anschlüsse und vier abgehende Anschlüsse auf. Im Allgemeinen sind in einer gegebenen beispielhaften N × N-Kachelanordnung die Randkacheln Kacheln mit nur drei Nachbarkacheln, während die Eckkacheln Kacheln mit zwei Nachbarkacheln sind. Hinsichtlich der Datenflussmethodologie über den Maschenbus **126** muss im Allgemeinen jede Eingangsaktivierung, die über den Maschenbus **126** für eine spezielle Kachel ankommt, zu einer oder mehreren schmalen Speichereinheiten der Kachel übergeben werden. Überdies können für die Kachelkonfigurationen, die weniger als vier eingehende Anschlüsse aufweisen, die DMAOp-Anweisungen Nullwerte an die Orte in dem schmalen Speicher der Kachel schreiben, anstatt an einem fehlenden Eingangsanschluss auf Daten zu warten. Gleichermaßen führen für die Kachelkonfigurationen, die weniger als vier abgehende Anschlüsse aufweisen, die DMAOp-Anweisungen die Lesevorgänge der schmalen Speicher und die Anschluss-Schreibvorgänge, die auf die Übertragungen für irgendwelche fehlenden Anschlüsse bezogen sind, nicht aus.

[0039] In einigen Implementierungen werden ein Ort oder eine Adresse einer schmalen Speichereinheit (en), in die eine spezielle Eingangsaktivierung geschrieben wird oder aus der eine spezielle Eingangsaktivierung gelesen wird, durch eine Tensortraversierungseinheit (im Folgenden "TTU") basierend auf einer über den Maschenbus **126** bereitgestellten eingehenden/abgehenden DMAOp erzeugt. Eine eingehende DMAOp und eine abgehende DMAOp können gleichzeitig ausgeführt werden, wobei irgendeine erforderliche Synchronisation durch Synchronisationsmerkersteuerschemata, die durch den Controller **102** gemanagt werden, gemanagt werden. Die TTUs werden im Folgenden bezüglich **Fig.** 2 und **Fig.** 3 ausführlicher beschrieben.

[0040] Der Ringbus **128** geht von dem Controller **102** aus und schafft ähnlich zu dem CSR-Bus **122** und dem Anweisungsbus **124** außerdem eine Kommunikationskopplung durch einen Busdatenweg, der die Kacheln **112**, **114** in einem Ring zurück zum Controller **102** verbindet. In verschiedenen Implementierungen verbindet oder koppelt der Ringbus **128** im Allgemeinen alle breiten Speichereinheiten (die im Folgenden bezüglich **Fig.** 2 ausführlicher beschrieben werden) in allen Kacheln **112**, **114**. Folglich entspricht eine Nutzdatenbreite des Ringbusses **128** der Breite der breiten Speichereinheiten, die innerhalb jeder Kachel der Kachelsätze **112**, **114** angeordnet sind. Wie oben erörtert worden ist, enthält der Ringbus **128** außerdem einen Bitmuster-Kopf, der die Kacheln angibt, die die über den Ringbus **128** übertragenen Nutzdaten verbrauchen müssen, die die Anweisungen oder die Parameter umfassen.

[0041] Hinsichtlich der über den Ringbus **128** an einer speziellen Kachel empfangenen Daten (d. h., der Nutzdaten) setzt (d. h., räumt) jede Kachel in Reaktion auf das Empfangen der Informationen vor dem Weiterleiten der Daten zu einer weiteren Kachel die

in dem Bitmuster-Kopf angegebenen Positionsdaten, die für die empfangende Kachel eindeutig sind, auf null (aus). Wenn das Kopf-Bitmuster keine verbleibenden Bitsatzdaten, die eine spezielle Kachel angeben, die die Nutzdaten empfangen soll, aufweist, stoppt folglich das Weiterleiten der Nutzdaten zu einer weiteren Kachel. Die Nutzdaten beziehen sich im Allgemeinen auf die Aktivierungen und die Gewichte, die während der basierend auf der Ausführung tief verschachtelter Schleifen ausgeführten Tensorberechnungen durch die eine oder die mehreren Kacheln verwendet werden.

[0042] In einigen Implementierungen kann der Controller **102** als ein Teil des Ringbusses **128** beschrieben werden. In einem Beispiel kann der Controller **102** für die innerhalb einer speziellen Kachel ausgeführten DMAOp-Anweisungen verwendet werden, um die Daten/Nutzdaten aus den Ringbusstopps herauszuholen und die Nutzdaten zu einem Ringbusstopp in einer nächsten Kachel in dem Ring weiterzuleiten. Der Controller **102** kann außerdem verursachen, dass die Nutzdaten zu einer oder mehreren breiten Speichereinheiten der Kachel übergeben werden, falls ein derartiger Vorgang durch die Anweisungen in dem Bitmuster-Kopf erforderlich ist. Die Adresse der einen oder der mehreren breiten Speichereinheiten, zu denen die Daten geschrieben werden müssen, können durch die DMAOp-Anweisungen innerhalb der speziellen Kachel erzeugt werden.

[0043] In verschiedenen Implementierungen kann jede Kachel des Kachelsatzes **112**, **114** entweder ein Erzeuger der Nutzdaten oder ein Verbraucher der Nutzdaten sein. Wenn eine Kachel ein Erzeuger der Nutzdaten ist, liest die Kachel die Daten aus einer oder mehreren ihrer breiten Speichereinheiten und sendet die Daten als Multicast über den Ringbus **128** für den Verbrauch durch eine oder mehrere andere Kacheln. Wenn eine Kachel ein Verbraucher der Nutzdaten ist, empfängt die Kachel die Daten und schreibt sie in eine oder mehrere breite Speichereinheiten innerhalb der Kachel, wobei sie die Nutzdaten für den Verbrauch durch eine oder mehrere andere Kacheln weiterleitet. Bezüglich der Bewegung der Nutzdaten über den Ringbus **128** gibt es zu irgendeinem gegebenen Zeitpunkt typischerweise nur einen Erzeuger/Master der Daten auf dem Ringbus **128**. Die Ausführungsreihenfolge der DMAOp-Anweisungen (z. B. das FIFO-Steuerschema) in allen Kacheln stellt sicher, dass es zu einem gegebenen Zeitpunkt nur einen Erzeuger/Master der Daten auf den Ringbus **128** gibt.

[0044] In einigen Implementierungen verwendet der Controller **102** eine Synchronisationsmerkersteuerarchitektur, um sicherzustellen, dass es zu einem gegebenen Zeitpunkt nur einen Erzeuger/Master der Nutzdaten auf dem Ringbus **128** gibt. In einem Beispiel löst jeder Schreibvorgang durch eine Kachel zu einem Ringausgang ein Inkrement des entsprechenden Synchronisationsmerkerzählerstands aus. Der Controller **102** kann die Nutzdaten untersuchen, um die Anzahl der Datenbatzen oder -segmente, die die Nutzdaten umfassen, zu bestimmen. Der Controller **102** überwacht dann die Ausführung durch die Kachel, um sicherzustellen, dass die erwartete Anzahl der Datensegmente durch die Kachel weitergeleitet und/oder verbraucht wird, bevor eine weitere Kachel im Master-Modus ausgeführt wird.

[0045] Eine Ausnahme zum Sicherstellen, dass es zu einem gegebenen Zeitpunkt nur einen Erzeuger/Master der Daten auf dem Ringbus **128** gibt, tritt auf, wenn es lokale Multicast-Gruppen gibt, die über den Ringbus **128** verbunden sind, die keinen überlappenden Bereich auf dem Ringbus aufweisen. Die Kachel 0 (der Master) kann z. B. einen Multicast an eine Kachel in der Kachel-0-Kachel-3-Gruppierung ausführen (d. h., Daten erzeugen), während die Kachel 4 (der Master) das Gleiche zu einer Kachel in der Kachel-4-Kachel-7-Gruppierung ausführen kann. Eine wichtige Anforderung an diese doppelte Master-Multicast-Methodologie ist, dass es den verschiedenen Multicast-Gruppen nicht erlaubt sein muss, die Datenpakete jeder anderen zu sehen, weil eine Paketüberlappung auftreten kann und zu einem oder mehreren Datenberechnungsfehlern führen kann.

[0046] Wie in **Fig.** 1 gezeigt ist, schafft der Controller **102** einen Kommunikationsdatenweg, der die Kacheln in den Kachelsätzen **112**, **114** mit der E/A **110** koppelt oder verbindet, wobei er mehrere Kernfunktionen enthält. Die Kernfunktionen des Controllers **102** enthalten im Allgemeinen das Zuführen einer oder mehrerer E/A-Eingangsaktivierungen zu den Kacheln in den Kachelsätzen **112**, **114**, das Zuführen einer oder mehrerer Eingangsaktivierungen und Parameter, die von der E/A **110** empfangen werden, zu den Kacheln, das Zuführen einer oder mehrerer von der E/A **110** empfangenen Anweisungen zu den Kacheln, das Senden von E/A-Ausgangsaktivierungen an die Host-Schnittstelle **108** und das Dienen als ein Ringstopp sowohl für den CSR-Bus **122** als auch für den Ringbus **128**. Wie im Folgenden ausführlicher beschrieben wird, enthalten sowohl der erste Kachelsatz **112** als auch der zweite Kachelsatz **114** jeder mehrerer Kacheln, die verwendet werden, um eine oder mehrere Tensorberechnungen auszuführen, die basierend auf einer tiefen Schleifenschachtelung ausgeführt werden, die aus inneren und äußeren Schleifen besteht.

[0047] Das System **100** arbeitet im Allgemeinen wie folgt. Die Host-Schnittstelle **108** erzeugt eine oder mehrere Anweisungen für den Controller **102**, die Speicherdirektzugriffsoperationen (DMAOp) definieren, die für eine gegebene Berechnung auftreten. Die den dem Controller **102** zugeführten Anweisungen zugeordneten Deskriptoren enthalten Informationen,

die durch den Controller benötigt werden, um große Skalarproduktberechnungen zu fördern, die den mehrdimensionalen Datenfeldern (Tensoren) zugeordnet sind. Im Allgemeinen empfängt der Controller **102** von der Host-Schnittstelle **108** Eingangsaktivierungen, Kachelanweisungen und Modellparameter (d. h., Gewichte) zum Ausführen der Tensorberechnungen für eine gegebene Schicht eines neuronalen Netzes. Der Controller **102** kann dann verursachen, dass die Anweisungen in der durch die Anweisung(en) definierten Weise eines Datenflusses durch Multicast an die Kacheln **112**, **114** gesendet werden. Wie oben erörtert worden ist, können die eine Anweisung verbrauchenden Kacheln dann eine Rundsendung einer neuen/anschließenden Anweisung an eine weitere Kachel basierend auf den Bitmuster-Daten in dem Anweisungskopf einleiten.

[0048] Bezüglich des Datenflusses werden die Eingangsaktivierungen und die Parameter über den Ringbus **128** an die Kacheln der Kachelsätze **112**, **114** gesendet. Jede der Kacheln **112**, **114** speichert eine Teilmenge der Eingangsaktivierungen, die benötigt werden, um eine Teilmenge der Ausgangsaktivierungen zu berechnen, die dieser speziellen Kachel zugewiesen sind. Die DMAOp-Anweisungen für eine Kachel verursachen, dass die Eingangsaktivierung von dem breiten Speicher zu dem schmalen Speicher bewegt wird. Die Berechnung innerhalb einer Kachel beginnt, wenn die erforderlichen Eingangsaktivierungen, Parameter/Gewichte und Rechenanweisungen (TTU-Operationen, Speicheradressen usw.) in der Kachel verfügbar sind. Die innerhalb einer Kachel stattfindenden Berechnungen enden, wenn die (im Folgenden beschriebenen) MAC-Operatoren innerhalb einer Kachel alle durch den Anweisungssatz definierten Skalarproduktoperationen abschließen und die Voraktivierungsfunktionen auf die Ergebnisse (d. h., die Ausgangsaktivierungen) der Multiplikationsoperation angewendet werden.

[0049] Die Ergebnisse der einen oder der mehreren Tensorberechnungen enthalten das Schreiben von Ausgangsaktivierungen einer Rechenschicht in eine schmale Speichereinheit(en) der Kacheln, die die Berechnung ausführt. Für bestimmte Tensorberechnungen gibt es eine Übertragung von Ausgangsrandaktivierungen über den Maschenbus **126** zu benachbarten Kacheln. Die Übertragung der Ausgangsrandaktivierungen zu den benachbarten Kacheln ist erforderlich, um die Ausgangsaktivierungen für eine anschließende Schicht zu berechnen, wenn die Berechnungen mehrere Schichten überspannen. Wenn die Berechnungen für alle Schichten abgeschlossen sind, bewegt eine DMAOp die endgültigen Aktivierungen durch den Ringbus **128** zu der Klassifiziererkachel **116**. Der Controller **102** liest dann die endgültigen Aktivierungen aus der Klassifiziererkachel **116** und führt eine DMAOp aus, um die endgültigen Aktivierungen zu der Host-Schnittstelle **108** zu bewegen. In einigen Implementierungen führt der Klassifiziererabschnitt **116** die Berechnungen einer Ausgangsschicht (d. h., der letzten Schicht) des NN aus. In anderen Implementierungen ist die Ausgangsschicht des NN eine von der Klassifiziererschicht, einer Regressionsschicht oder einem anderen Schichttyp, der im Allgemeinen neuronalen Netzen zugeordnet ist.

[0050] **Fig.** 2 veranschaulicht eine beispielhafte Rechenkachel **200** eines neuronalen Netzes (NN). Im Allgemeinen kann die beispielhafte Kachel **200** irgendeiner der Kacheln innerhalb des ersten Kachelsatzes **112** und des zweiten Kachelsatzes **114**, die oben bezüglich **Fig.** 1 erörtert worden sind, entsprechen. In verschiedenen Implementierungen kann auf die Rechenkachel **200** außerdem als eine Computereinheit **200** Bezug genommen werden oder kann die Rechenkachel **200** außerdem als eine Computereinheit **200** bezeichnet werden. Jede Rechenkachel **200** ist eine in sich geschlossene Recheneinheit, die konfiguriert ist, um Anweisungen unabhängig bezüglich anderer entsprechender Kacheln innerhalb der Kachelsätze **112**, **114** auszuführen. Wie oben kurz erörtert worden ist, führt jede Rechenkachel **200** zwei Typen von Anweisungen, eine TensorOp-Anweisung und eine DMAOp-Anweisung, aus. Im Allgemeinen enthält jeder Anweisungstyp Rechenoperationen, die tiefen Schleifenschachtelungen zugeordnet sind, wobei folglich jeder Anweisungstyp im Allgemeinen über mehrere Zeiträume ausgeführt wird, um den Abschluss aller Schleifeniterationen sicherzustellen.

[0051] Wie im Folgenden ausführlicher erörtert wird, werden die verschiedenen Anweisungstypen durch unabhängige Steuereinheiten innerhalb der Rechenkachel **200** ausgeführt, die die Daten durch Synchronisationsmerkersteuerungen synchronisieren, die innerhalb der Rechenkachel **200** gemanagt werden. Die Synchronisationsmerkersteuerungen managen gleichzeitig zwischen den Ausführungen verschiedener Anweisungstypen innerhalb der Rechenkachel **200**. Jede Rechenoperation, die jedem Anweisungstypen zugeordnet ist, wird in einer strengen Reihenfolge der Ausgabe (d. h., First-In-First-Out) ausgeführt. Bezüglich der beiden Anweisungstypen, TensorOP und DMAOp, gibt es keine Ordnungsgarantien zwischen diesen verschiedenen Anweisungstypen, wobei jeder Typ durch die Rechenkachel **200** als ein separater Thread der Steuerung behandelt wird.

[0052] Bezüglich der Datenflusskonstrukte enthält die Rechenkachel **200** im Allgemeinen einen Datenweg **202** und einen Datenweg **205**, von denen jeder einen Kommunikationsweg für den Datenfluss in die und aus der Rechenkachel **200** schafft. Wie oben beschrieben worden ist, enthält das System **100** drei unterschiedene Datenbusstrukturen, die in einer Ringkonfiguration ausgelegt sind – den CSR-Bus **122**, den Anweisungsbus **124** und den Ringbus **128**.

In **Fig.** 2 entspricht der Datenweg **205** dem Anweisungsbus **124**, während der Datenweg **202** im Allgemeinen entweder dem CSR-Bus **122** oder dem Ringbus **128** entspricht. Wie gezeigt ist, enthält der Datenweg **202** einen Ringausgang **203**, der einen Ausgangsweg für die Daten, die die Rechenkachel **200** verlassen, schafft, und einen Ringeingang **204**, der einen Eingangsweg für die Daten, die in die Rechenkachel **200** eintreten, schafft.

[0053] Die Rechenkachel **200** enthält ferner eine TensorOp-Steuerung **206**, die eine TensorOp-Tensortraversierungseinheit (TensorOp-TTU) **226** und eine DMA-Op-Steuerung **208**, die eine DMAOp-TTU **228** enthält, enthält. Die TensorOp-Steuerung **206** managt im Allgemeinen die Schreibvorgänge in ein und die Lesevorgänge aus einem TensorOp-TTU-Register **232** und managt die Traversierungsoperationen für die Ausführung durch die TensorOp-TTU **226**. Gleichermaßen managt die DMAOp-Steuerung **208** im Allgemeinen die Schreibvorgänge in ein und die Lesevorgänge aus einem DMAOp-TTU-Register **234**, wobei sie die Traversierungsoperationen für die Ausführung durch die DMAOp-TTU **228** managt. Das TTU-Register **232** enthält Anweisungspuffer zum Speichern einer oder mehrerer Anweisungen, die die Operationen umfassen, die durch die TensorOp-TTU **226** bei der Ausführung der Anweisungen durch die TensorOp-Steuerung **206** ausgeführt werden. Gleichermaßen enthält das TTU-Register **234** Anweisungspuffer zum Speichern einer oder mehrerer Anweisungen, die Operationen umfassen, die durch die TTU **228** bei der Ausführung der Anweisungen durch die DMAOp-Steuerung **208** ausgeführt werden.

[0054] Wie im Folgenden weiter beschrieben wird, werden die TTUs **226** und/oder **228** durch die Rechenkachel **200** verwendet, um die Feldelemente eines oder mehrerer Tensoren zu traversieren, die im Allgemeinen in einem schmalen Speicher **210** und einem breiten Speicher **212** gespeichert sind. In einigen Implementierungen wird die TTU **226** durch die TensorOp-Steuerung **206** verwendet, um Tensoroperationen zum Traversieren der Dimensionen eines mehrdimensionalen Tensors basierend auf der Ausführung einer tiefen Schleifenschachtelung bereitzustellen.

[0055] In einigen Implementierungen kommen bestimmte Anweisungen für die Ausführung durch die Rechenkachel **200** über den Datenweg **205** (d. h., einen Abschnitt des Anweisungsbusses **124**) an der Kachel an. Die Rechenkachel **200** untersucht das Kopf-Bitmuster, um den Anweisungstyp (TensorOp oder DMAOp) und den Anweisungsuntertyp (Leseoperation oder Schreiboperation) zu bestimmen. Die durch die Rechenkachel **200** empfangene Anweisung wird (empfangenen Anweisungen werden) anschließend in Abhängigkeit von dem Anweisungstyp in einen speziellen Anweisungspuffer geschrieben. Im Allgemeinen werden die Anweisungen vor der Ausführung der Anweisung durch eine Komponente der Rechenkachel **200** empfangen und gespeichert (d. h., in den Puffer geschrieben). Wie in **Fig.** 2 gezeigt ist, können die Anweisungspuffer (d. h., das TensorOp-TTU-Register **232** und das DMAOp-TTU-Register **234**) jeder ein First-in-first-out-Steuerschema (FIFO-Steuerschema) enthalten, das den Verbrauch (die Ausführung) einer oder mehrerer in Beziehung stehender Anweisungen priorisiert.

[0056] Wie oben kurz erörtert worden ist, ist ein Tensor ein mehrdimensionales geometrisches Objekt, wobei beispielhafte mehrdimensionale geometrische Objekte Matrizen und Datenfelder enthalten. Ein Algorithmus, der tief verschachtelte Schleifen enthält, kann durch die Rechenkachel **200** ausgeführt werden, um Tensorberechnungen durch das Iterieren einer oder mehrerer verschachtelter Schleifen auszuführen, um einen N-dimensionalen Tensor zu traversieren. In einem beispielhaften Rechenprozess kann jede Schleife der Schleifenschachtelung für das Traversieren einer speziellen Dimension des N-dimensionalen Tensors verantwortlich sein. Wie hier beschrieben ist, managt die TensorOp-Steuerung **206** im Allgemeinen eine oder mehrere Tensoroperationen, die die Folge ansteuern, in der die Dimensionselemente eines speziellen Tensorkonstrukts traversiert werden und auf die Dimensionselemente eines speziellen Tensorkonstrukts zugegriffen wird, um die durch die tief verschachtelten Schleifen definierten Berechnungen abzuschließen.

[0057] Die Rechenkachel **200** enthält ferner einen schmalen Speicher **210** und einen breiten Speicher **212**. Die Bezeichnungen schmal und breit beziehen sich im Allgemeinen auf eine Größe in der Breite (Bits/Bytes) der Speichereinheiten des schmalen Speichers **210** und des breiten Speichers **212**. In einigen Implementierungen enthält der schmale Speicher **210** Speichereinheiten, die jede eine Größe oder Breite von weniger als 16 Bits aufweisen, und enthält der breite Speicher **212** Speichereinheiten, die jede eine Größe oder Breite von weniger als 32 Bits aufweisen. Im Allgemeinen empfängt die Rechenkachel **200** die Eingangsaktivierungen über den Datenweg **205**, wobei die DMA-Steuerung **208** eine Operation ausführt, um die Eingangsaktivierungen in den schmalen Speicher **210** zu schreiben. Gleichermaßen empfängt die Rechenkachel **200** die Parameter (die Gewichte) über den Datenweg **202**, wobei die DMA-Steuerung **208** eine Operation ausführt, um die Parameter in den breiten Speicher **212** zu schreiben. In einigen Implementierungen kann der schmale Speicher **210** einen Speicher-Arbiter enthalten, der typischerweise in gemeinsam benutzten Speichersystemen verwendet wird, um für jeden Speicherzyklus zu entscheiden, welcher Steuervorrichtung (z. B. der TensorOp-Steuerung **206** oder der DMAOp-

Steuerung **208**) erlaubt wird, auf die gemeinsam benutzten Speichereinheiten des schmalen Speichers **210** zuzugreifen.

[0058] Die Rechenkachel **200** enthält ferner einen Eingangsaktivierungsbus **216** und ein MAC-Feld **214**, das mehrere Zellen enthält, die jede einen MAC-Operator **215** und ein Summenregister **220** enthalten. Im Allgemeinen führt das MAC-Feld **214** unter Verwendung der MAC-Operatoren **215** und der Summenregister **220** über mehrere Zellen die Tensorberechnungen aus, die arithmetische Operationen bezüglich der Skalarproduktberechnungen enthalten. Der Eingangsaktivierungsbus **216** schafft einen Datenweg, in dem die Eingangsaktivierungen durch den schmalen Speicher **210** nacheinander für den jeweiligen Zugriff durch jeden MAC-Operator **215** des MAC-Feldes **214** bereitgestellt werden. Folglich empfängt ein einziger MAC-Operator **215** einer speziellen Zelle basierend auf der Rundsendung der Reihe nach einer Eingangsaktivierung jeweils eine Eingangsaktivierung. Die durch die MAC-Operatoren des MAC-Feldes **214** ausgeführten arithmetischen Operationen enthalten im Allgemeinen das Multiplizieren einer durch den schmalen Speicher **210** bereitgestellten Eingangsaktivierung mit einem Parameter, auf den von dem breiten Speicher **212** zugegriffen wird, um einen einzigen Ausgangsaktivierungswert zu erzeugen.

[0059] Während der arithmetischen Operationen können die Partialsummen akkumuliert und in einem entsprechenden z. B. Summenregister **220** gespeichert werden oder in den breiten Speicher **212** geschrieben werden, wobei durch eine spezielle Zelle des MAC-Feldes **214** erneut auf sie zugegriffen werden kann, um die folgenden Multiplikationsoperationen abzuschließen. Die Tensorberechnungen können als einen ersten Abschnitt und einen zweiten Abschnitt aufweisend beschrieben werden. Der erste Abschnitt ist abgeschlossen, wenn die Multiplikationsoperationen eine Ausgangsaktivierung, z. B. durch das Abschließen einer Multiplikation einer Eingangsaktivierung und eines Parameters, um die Ausgangsaktivierung zu erzeugen, erzeugen. Der zweite Abschnitt enthält die Anwendung einer nichtlinearen Funktion auf eine Ausgangsaktivierung, wobei der zweite Abschnitt abgeschlossen ist, wenn die Ausgangsaktivierung nach der Anwendung der Funktion in den schmalen Speicher **210** geschrieben wird.

[0060] Die Rechenkachel **200** enthält ferner einen Ausgangsaktivierungsbus **218**, eine Nichtlinear-Einheit (NLU) **222**, die eine Ausgangsaktivierungspipeline **224** umfasst, eine NLU-Steuerung **238** und eine Bezugsabbildung **230**, die ein Kernattribut einer Komponente in der Rechenkachel **200** angibt. Die Bezugsabbildung **230** ist für die Klarheit gezeigt, wobei sie aber nicht in der Rechenkachel **200** enthalten ist. Die Kernattribute enthalten, ob eine spezielle Komponente eine Einheit, eine Speichervorrichtung, ein Operator, eine Steuervorrichtung oder ein Datenweg ist. Im Allgemeinen werden beim Abschluss des ersten Abschnitts der Tensorberechnungen die Ausgangsaktivierungen von dem MAC-Feld **214** über den Ausgangsaktivierungsbus **218** der NLU **222** bereitgestellt. Nach der Ankunft an der NLU **222** werden die Daten, die eine Aktivierungsfunktion spezifizieren und die über die Aktivierungspipeline **224** empfangen werden, auf die Ausgangsaktivierungen angewendet, wobei die Ausgangsaktivierungen dann in den schmalen Speicher **210** geschrieben werden. In einigen Implementierungen enthält der Ausgangsaktivierungsbus **218** wenigstens ein Pipeline-Schieberegister **236**, wobei das Abschließen des zweiten Abschnitts der Tensorberechnungen das Verwenden eines Schieberegisters **236** des Aktivierungsbusses **218** enthält, um die Ausgangsaktivierungen zu dem schmalen Speicher **210** zu schieben.

[0061] Bezüglich der Skalarproduktberechnungen von z. B. zwei mehrdimensionalen Datenfeldern für eine einzige Rechenkachel **200** stellt das MAC-Feld **214** z. B. eine robuste Einzelbefehl-Mehrfachdaten-Funktionalität (SIMD-Funktionalität) bereit. Die SIMD bedeutet im Allgemeinen, dass alle parallelen Einheiten (mehrere MAC-Operatoren **215**) die gleiche Anweisung (basierend auf der tiefen Schleifenschachtelung) gemeinsam benutzen, wobei aber jeder MAC-Operator **215** die Anweisung an verschiedenen Datenelementen ausführt. In einem grundlegenden Beispiel erfordert das elementweise Addieren der Felder [1, 2, 3, 4] und [5, 6, 7, 8], um das Feld [6, 8, 10, 12] in einem Zyklus zu erhalten, typischerweise vier Arithmetikeinheiten, um die Operation an jedem Element auszuführen. Unter Verwendung der SIMD können die vier Einheiten die gleiche Anweisung (z. B. "Addieren") gemeinsam benutzen und die Berechnungen parallel ausführen. Weil die Anweisungen gemeinsam benutzt werden, sind die Anforderungen für die Anweisungsbandbreite und den Anweisungsspeicher verringert, wobei folglich die Effizienz erhöht wird. Folglich schaffen das System **100** und die Rechenkachel **200** eine verbesserte Beschleunigung und eine verbesserte Parallelität der Tensorberechnungen gegenüber früheren Verfahren.

[0062] In einem Beispiel, und wie im Folgenden ausführlicher beschrieben wird, kann eine einzige Anweisung durch den Controller **102** mehreren Rechenkacheln **200** (siehe die Kachelsätze **112**, **114** nach **Fig.** 1) für den Verbrauch durch mehrere MAC-Felder **214** bereitgestellt werden. Im Allgemeinen können die Schichten des neuronalen Netzes mehrere Ausgangsneuronen enthalten, wobei die Ausgangsneuronen partitioniert sein können, so dass die einer Teilmenge der Ausgangsneuronen zugeordneten Tensorberechnungen einer speziellen Kachel der Kachelsätze **112**, **114** zugewiesen sein können. Jede Kachel der Kachelsätze **112**, **114** kann dann die in Beziehung stehenden Tensorberechnungen an

verschiedenen Gruppen von Neuronen für eine gegebene Schicht ausführen. Die Rechenkachel **200** kann deshalb wenigstens zwei Formen der Parallelität schaffen: 1) eine Form enthält das Partitionieren der Ausgangsaktivierungen (die der Teilmenge von Ausgangsneuronen entsprechen) unter den mehreren Kacheln des Kachelsatzes **112**, **114**; und 2) eine weitere Form enthält die gleichzeitige Berechnung (mit einer einzigen Anweisung) mehrerer Teilmengen der Ausgangsneuronen basierend auf dem Partitionieren unter den Kacheln der Kachelsätze **112**, **114**.

[0063] Fig. 3 veranschaulicht eine beispielhafte Tensortraversierungseinheit-Struktur (TTU-Struktur) **300**, die vier zu verfolgende Tensoren umfasst, wobei jeder eine Tiefe von acht aufweist. Die TTU **300** enthält im Allgemeinen einen Zählertensor **302**, einen Schritttensor **304**, einen Anfangstensor **306** und einen Grenztensor **308**. Die TTU **300** enthält ferner eine Addiererbank **310** und einen Tensoradressenindex **312**. Wie oben beschrieben worden ist, ist ein Tensor ein mehrdimensionales geometrisches Objekt, wobei, um auf ein Element des Tensors zuzugreifen, ein Index jeder Dimension bereitgestellt werden muss. Weil der Tensor im schmalen Speicher **210** und im breiten Speicher **212** gespeichert ist, muss ein Satz von Tensorindizes in einen Satz von Speicheradressen übersetzt werden. In einigen Implementierungen wird die Übersetzung der Indizes in die Speicheradressen ausgeführt, indem die Speicheradressen zu einer Linearkombination der Indizes gemacht werden und die Adressen über den Tensoradressenindex **212** widergespiegelt werden.

[0064] Es gibt eine TTU pro Steuer-Thread und es gibt einen Steuer-Thread pro Anweisungstyp (TensorOp und DMAOp) in der Rechenkachel **200**. Entsprechend gibt es, wie oben erörtert worden ist, zwei Sätze von TTUs in der Rechenkachel **200**: 1) die TensorOp-TTU **226**; und 2) die DMAOp-TTU **228**. In verschiedenen Implementierungen veranlasst die TensorOp-Steuerung **206** die TTU **300**, am Anfang einer speziellen Tensoroperation die Werte des Zählers **302**, der Grenze **308** und des Schrittes **304** der TensorOp-TTU zu laden, wobei sie die Registerwerte nicht ändert, bevor die Anweisung stillgelegt ist. Jede der beiden TTUs muss eine Adresse für die folgenden Speicheradressenanschlüsse in der Rechenkachel **200** erzeugen: 1) die Adressenanschlüsse des breiten Speichers **212** und 2) des schmalen Speichers **210**, der vier unabhängig arbitrierte Bänke aufweist, die als vier Adressenanschlüsse dargestellt sind.

[0065] Wie oben erörtert worden ist, kann in einigen Implementierungen der schmale Speicher **210** einen Speicher-Arbiter enthalten, der typischerweise in gemeinsam benutzten Speichersystemen verwendet wird, um für jeden Speicherzyklus zu entscheiden, welcher Steuervorrichtung (z. B. der TensorOp-Steuerung **206** oder der DMAOp-Steuerung **208**) erlaubt wird, auf die gemeinsam benutzten Speicherbetriebsmittel des schmalen Speichers **210** zuzugreifen. In einem Beispiel sind die verschiedenen Anweisungstypen (TensorOp und DMAOp) unabhängige Steuer-Threads, die einen Speicherzugriff anfordern, der arbitriert werden muss. Wenn ein spezieller Steuer-Thread ein Tensorelement an den Speicher übergibt, inkrementiert der Steuer-Thread die Zähler **302** der Tensorbezugnahme, die an den Speicher übergeben wurde.

[0066] In einem Beispiel kann die TTU **300**, wenn die TensorOp-Steuerung **202** eine Anweisung zum Zugreifen auf ein spezielles Element eines Tensors ausführt, die Adresse des speziellen Elements des Tensors bestimmen, so dass die Steuerung **206** auf den Speicher, z. B. den schmalen Speicher **210**, zugreifen kann, um die Daten zu lesen, die einen Aktivierungswert des speziellen Elements repräsentieren. In einigen Implementierungen kann ein Programm eine verschachtelte Schleife enthalten und kann die Steuerung **206** eine Anweisung ausführen, um auf ein Element einer zweidimensionalen Feldvariable innerhalb der verschachtelten Schleife gemäß den aktuellen Werten der Indexvariable, die der verschachtelten Schleife zugeordnet ist, zuzugreifen.

[0067] Die TTU **300** kann einen Traversierungszustand für bis zu einer Anzahl von X TTU-Zeilen für einen gegebenen Tensor (gegebene Tensoren) gleichzeitig halten. Jeder Tensor, der gleichzeitig in der TTU **300** gespeichert ist, belegt einen dedizierten Hardware-Tensorsteuerungsdeskriptor. Der Hardware-Steuerungsdeskriptor kann aus einer Anzahl X von TTU-Zählern **302** pro Zeilenposition, dem Schritt- **304** und dem Grenz- **308** Register bestehen, die Tensoren unterstützen, die bis zu einer Anzahl von X TTU-Zählern pro Zeilendimensionen aufweisen. In einigen Implementierungen kann die Anzahl der Zeilen und die Anzahl der Zähler pro Zeile verschieden sein.

[0068] Für ein gegebenes Positionsregister wird die endgültige Speicheradresse aus einer Additionsoperation berechnet, die das Addieren der Positionsregister enthält. Die Basisadresse wird in den Zähler **302** aufgenommen. Ein oder mehrere Addierer werden für die Tensorbezugnahmen, die in demselben Speicher gespeichert sind, gemeinsam benutzt. Weil es an einem gegebenen Anschluss in einem Zyklus nur ein einziges Laden/Speichern geben kann, gibt es in einer Implementierung eine Funktion der Steuerung der Schleifenschachtelung, um sicherzustellen, dass die Zähler mehrerer Tensorbezugnahmen, die in demselben schmalen Speicher oder breiten Speicher gespeichert sind, in irgendeinem gegebenen Zyklus nicht inkrementiert werden. Die Verwendung der Register zum Berechnen der Speicherzugriffs-Adressenwerte einschließlich der Bestimmung der Versatz-

werte sind in der Patentanmeldung, laufende Nr. 15/014.265, mit dem Titel "Matrix Processing Apparatus", eingereicht am 3. Februar 2016, ausführlicher beschrieben, deren gesamte Offenbarung hierdurch durch Bezugnahme in ihrer Gesamtheit hier ausdrücklich aufgenommen ist.

[0069] Wenn z. B. ein Software-Algorithmus einen N-dimensionalen Tensor verarbeitet, kann eine verschachtelte Schleife verwendet werden, wobei jede Schleife für das Traversieren jeder Dimension des N-dimensionalen Tensors verantwortlich ist. Ein mehrdimensionaler Tensor kann eine Matrix oder mehrdimensionale Matrizen sein. Jede Dimension des N-dimensionalen Tensors kann ein oder mehrere Elemente enthalten, wobei jedes Element einen jeweiligen Datenwert speichern kann. Ein Tensor kann z. B. eine Variable in einem Programm sein, wobei die Variable drei Dimensionen aufweisen kann. Die erste Dimension kann eine Länge von dreihundert Elementen aufweisen, die zweite Dimension kann eine Länge von eintausend Elementen aufweisen und die dritte Dimension kann eine Länge von zwanzig Elementen aufweisen.

[0070] Das Traversieren des Tensors in einer verschachtelten Schleife kann eine Berechnung eines Speicheradressenwertes eines Elements erfordern, um den entsprechenden Datenwert des Elements zu laden oder zu speichern. Eine For-Schleife ist z. B. eine verschachtelte Schleife, wobei die drei Schleifen, die durch die drei Schleifenindexvariable verfolgt werden, verschachtelt sein können, um durch den dreidimensionalen Tensor zu traversieren. In einigen Fällen kann es sein, dass ein Prozessor eine Schleifengrenzbedingung ausführen muss, wie z. B. das Festlegen einer Schleifengrenze einer inneren Schleife mit einer Indexvariable einer äußeren Schleife. Beim Bestimmen, ob die innerste Schleife einer verschachtelten Schleife zu verlassen ist, kann das Programm z. B. den aktuellen Wert der Schleifenindexvariable der innersten Schleife mit dem aktuellen Wert der Schleifenindexvariable der äußersten Schleife der verschachtelten Schleife vergleichen.

[0071] Wenn die Verarbeitungseinheit einer Rechenkachel eine Anweisung zum Zugreifen auf ein spezielles Element eines Tensors ausführt, bestimmt eine Tensortraversierungseinheit z. B. die Adresse des speziellen Elements des Tensors, so dass die Verarbeitungseinheit auf das Speichermedium (den Speicher) zugreifen kann, um die Daten zu lesen, die den Wert des speziellen Elements repräsentieren. Ein Programm kann z. B. eine verschachtelte Schleife enthalten und die Verarbeitungseinheit kann eine Anweisung ausführen, um auf ein Element einer zweidimensionalen Feldvariable innerhalb der verschachtelten Schleife gemäß den aktuellen Werten der Indexvariable, die der verschachtelten Schleife zugeordnet sind, zuzugreifen. Basierend auf den aktuellen Werten der Indexvariable, die der verschachtelten Schleife zugeordnet sind, kann die Tensortraversierungseinheit einen Versatzwert bestimmen, der einen Versatz von einem ersten Element einer zweidimensionalen Feldvariable repräsentiert. Die Verarbeitungseinheit kann dann unter Verwendung des Versatzwertes und von dem Speicher auf das spezielle Element der zweidimensionalen Feldvariable zugreifen.

[0072] Das Folgende stellt Schablonenparameter bereit, die verwendet werden können, um eine spezialisierte TTU **300** zu instanziieren: 1) eine Anzahl X von TTU-Zeilen; 2) eine Anzahl X von TTU-Zählern pro Zeile; 3) eine Anzahl X von TTU-Addiereinheiten; 4) pro TTU-Zeile eine gemeinsam benutzte Addiererbezugnahme angeben; und 5) pro Zähler eine Zählergröße [TTU][Zeile][Tiefe] X angeben. Alle TTU-Register sind architektonisch sichtbar. Eine Adresse eines speziellen Tensorelements (d. h., die Tensoradresse **312**), auf die für die Berechnung zugegriffen werden muss, ist das Ergebnis der Addition der Zähler. Wenn von dem Steuer-Thread ein Inkrementsignal an eine Zeile der TTU ausgegeben wird, führt die TTU **300** eine Einzelzyklusoperation aus, wobei sie die innerste Dimension um einen Schritt **304** dieser Dimension inkrementiert und die überlappende Eingabe durch alle Tiefen ausbreitet.

[0073] Im Allgemeinen bestimmt die TTU **300** einen Status, der einem oder mehreren Tensoren zugeordnet ist. Der Status kann die Schleifengrenzwerte, die aktuellen Werte der Schleifenindexvariable, die Dimensionsmultiplikatoren zum Berechnen eines Speicheradressenwertes und/oder die Programmzählerwerte zur Handhabung der Verzweigungsschleifengrenzen enthalten. Die TTU **300** kann ein oder mehrere Tensorstatuselemente und eine Arithmetiklogikeinheit enthalten. Jedes der Tensorstatuselemente kann ein Speicherelement, z. B. ein Register oder irgendeine andere geeignete Speicherschaltungsanordnung, sein. In einigen Implementierungen können die Tensorstatuselemente physisch oder logisch in verschiedenen Gruppen angeordnet sein, wie in der Patentanmeldung, laufende Nr. 15/014.265, ausführlicher beschrieben ist.

[0074] **Fig.** 4 veranschaulicht eine beispielhafte Architektur, die einen schmalen Speicher **210** enthält, der die Aktivierungen **404** über den Eingangsbus **216** an einen oder mehrere Multiplikations-Akkumulations-Operatoren (MAC-Operatoren) rundsendet. Ein Schieberegister **404** schafft eine Schiebefunktionalität, wodurch die Aktivierungen **404** eine auf einmal auf den Eingabebus **216** für den Empfang durch einen oder mehrere MAC-Operatoren **215** in einer MAC-Zelle **410** ausgesendet werden. Im Allgemeinen können die MAC-Zellen **410**, die die MAC-Operatoren **215** enthalten, als Rechenzellen definiert sein, die eine Partialsumme berechnen, und die in einigen

Implementierungen konfiguriert sind, um ein Partialsummen-Datenelement an den Ausgangsbus **218** zu schreiben. Wie gezeigt ist, können die Zellen **410** einen oder mehrere MAC-Operatoren umfassen. In einer Implementierung wird die Anzahl der MAC-Operatoren **215** in einer MAC-Zelle **410** als die Ausgabebreite der Zelle bezeichnet. Als ein Beispiel bezieht sich eine Doppelausgabezelle auf eine Zelle mit zwei MAC-Operatoren, die die Multiplikation von zwei Aktivierungswerten (aus dem schmalen Speicher **210**) mit zwei Parametern (aus dem breiten Speicher **212**) berechnen und eine Addition zwischen den Ergebnissen der zwei Multiplizierer und der aktuellen Partialsumme ausführen können.

[0075] Wie oben beschrieben worden ist, ist der Eingangsbus **216** ein Rundsendebus, der die Eingangsaktivierungen den MAC-Operatoren **215** der Lineareinheit (d. h., dem MAC-Feld **214**) bereitstellt. In einigen Implementierungen wird die gleiche Eingabe zwischen allen MAC-Operatoren **215** gemeinsam benutzt. Die Breite des Eingangsbusses **216** muss breit genug sein, um die Rundsendeeingaben der entsprechenden Anzahl von Zellen für ein gegebenes MAC-Feld **214** zuzuführen. Es wird das folgende Beispiel betrachtet, um die Struktur des Eingangsbusses **216** zu veranschaulichen. Wenn die Anzahl der Zellen in der Lineareinheit gleich vier ist und die Aktivierungsbreite gleich acht Bits ist, kann der Eingangsbus **216** konfiguriert sein, um bis zu vier Eingangsaktivierungen in jedem Zyklus bereitzustellen. In diesem Beispiel greift jede Zelle in dem MAC-Feld **214** nur auf eine aus den vier Aktivierungen zu, die rundgesendet werden.

[0076] Basierend auf den Einstellungen des TensorOp-Feldes der durch die Rechenkachel **200** empfangenen Anweisung kann es sein, dass die Zellen des MAC-Felds **214** Berechnungen unter Verwendung derselben Eingangsaktivierung ausführen müssen. Dies kann als eine Zout-Partitionierung innerhalb einer Zelle des MAC-Feldes **214** bezeichnet werden. Gleichermaßen tritt eine Zin-Partitionierung innerhalb einer Zelle auf, wenn die Zellen eines MAC-Feldes **214** verschiedene Aktivierungen benötigen, um die Berechnungen auszuführen. In dem ersteren Fall wird die einzige Eingangsaktivierung viermal wiederholt, wobei die aus dem schmalen Speicher **210** gelesenen vier Aktivierungen während vier Zyklen rundgesendet werden. In dem letzteren Fall ist für jeden Zyklus ein Lesen des schmalen Speichers **210** erforderlich. Für das obererwähnte Beispiel baut die TensorOp-Steuerung **206** diese Rundsendemethodologie basierend auf der Ausführung der von dem Controller **102** empfangenen Anweisungen auf.

[0077] **Fig.** 5 veranschaulicht eine beispielhafte Architektur, die einen Ausgangsbus **218** zum Bereitstellen von Ausgangsaktivierungen für eine schmale Speichereinheit **210** nach **Fig.** 2 und **Fig.** 4 enthält. Im Allgemeinen berechnet jede MAC-Zelle **215** des MAC-Feldes **214** in der Rechenkachel **200** eine andere Ausgangsaktivierung. Bezüglich eines Ausgangsmerkmalsfeldes können jedoch in den Fällen, in denen die Ausgangsmerkmalstiefe kleiner als die Anzahl der MAC-Zellen **215** in einer Rechenkachel **200** ist, die Zellen gruppiert werden, um eine oder mehrere Zellengruppen zu bilden. Alle MAC-Zellen **215** in einer Zellengruppe berechnen die gleiche Ausgabe (d. h., für eine Ausgangsmerkmalsabbildung), wobei jedoch jede Zelle nur eine Teilmenge der Ausgaben berechnet, die einer Teilmenge der Zin-Dimension entspricht. Im Ergebnis ist die Ausgabe einer MAC-Zelle **215** nun eine Partialsumme und nicht die endgültige lineare Ausgabe. In einigen Implementierungen vereinigt die NLU **222** diese Partialsummen basierend auf einem durch die NLU-Steuerung **238** der NLU **222** bereitgestellten Steuersignal in die endgültige lineare Ausgabe.

[0078] Wie oben erörtert worden ist, ist der Ausgangsbus **218** ein Pipeline-Schieberegister. Wenn in verschiedenen Implementierungen ein erster Abschnitt der Tensorberechnungen endet und die TensorOp-Steuerung **206** (durch das Ausführen einer Anweisung) angibt, dass eine Partialsumme ausgeschrieben werden muss, gibt es ein paralleles Laden von Partialsummen, die an dem Ausgangsbus **218** bereitgestellt werden. Die Anzahl der parallelen Ladevorgänge entspricht der Anzahl der MAC-Zellen in der Rechenkachel **200**. Die TensorOp-Steuerung **206** veranlasst dann, dass die Partialsummengrößen herausgeschoben werden und durch die nichtlineare Pipeline gesendet werden. In einigen Implementierungen kann es Umstände geben, in denen nicht alle MAC-Zellen in einer Kachel tatsächlich verwendet werden, um die Berechnungen auszuführen. Unter einem derartigen Umstand sind nicht alle auf den Ausgangsbus geschobenen Partialsummen gültig. In diesem Beispiel kann die TensorOp-Steuerung **206** dem MAC-Feld **214** ein Steuersignal bereitstellen, um die Anzahl der gültigen Zellen anzugeben, die herausgeschoben werden sollten. Die Parallelladegrößen, die auf den Ausgangsbus **118** geladen sind, entsprechen immer noch der Anzahl der MAC-Zellen in der Rechenkachel, wobei jedoch nur die gültigen Werte herausgeschoben und dem schmalen Speicher **210** übergeben werden.

[0079] **Fig.** 6 ist ein beispielhafter Ablaufplan eines Prozesses **600** zum Ausführen der Tensorberechnungen unter Verwendung der Rechenkachel **200** nach **Fig.** 2 eines neuronalen Netzes. Der Prozess **600** beginnt im Block **602**, wobei der schmale Speicher **210** der Rechenkachel **200** die Aktivierungen nacheinander an den Eingangsaktivierungs-Datenbus **260** sendet (d. h., rundsendet). Die Aktivierungswerte werden in einem schmalen Speicher **210** gespeichert. Der schmale Speicher **210** kann eine Sammlung von Bänken statischen Schreib-Le-

se-Speichers (SRAM-Bänken) sein, die die Adressierung für spezielle Speicherstellen zum Zugreifen auf die Eingangsgrößen erlauben. Die aus dem schmalen Speicher **210** gelesenen Aktivierungen werden über den Eingangsaktivierungsbus **216** an die linearen Zellen des MAC-Feldes **214** (d. h., die Lineareinheit) rundgesendet, die mehrere MAC-Operatoren **215** und Summenregister **220** umfassen. Im Block **604** des Prozesses **600** empfangen die MAC-Operatoren **215** der Rechenkachel **200** jeder zwei Eingaben – eine Eingabe (eine Aktivierung) wird von dem Eingangsaktivierungsbus **216** empfangen; während eine weitere Eingabe (ein Parameter) von dem breiten Speicher **212** empfangen wird. Entsprechend erhalten die einem der Eingänge jedes MAC-Operators **215** und jedes MAC-Operators **215** in den Zellen des MAC-Feldes **214** zugeführten Aktivierungen ihre zweite Multipliziereingabe von dem breiten Speicher **212**.

[0080] Im Block **606** des Prozesses **600** führt das MAC-Feld **214** der Rechenkachel **200** die Tensorberechnungen, die Skalarproduktberechnungen umfassen, basierend auf den Elementen einer Datenfeldstruktur, auf die vom Speicher zugegriffen wird, aus. Der breite Speicher **212** weist eine Breite in Bits auf, die gleich der Breite der Lineareinheit (32 Bits) ist. Die Lineareinheit (LU) ist folglich eine SIMD-Vektorarithmetiklogikeinheit (SIMD-ALU-Einheit), die Daten von einem Vektorspeicher (d. h., dem breiten Speicher **212** empfängt). In einigen Implementierungen können die MAC-Operatoren **215** außerdem ebenso die Akkumulatoreingaben (die Partialsummen) von dem breiten Speicher **212** erhalten. In einigen Implementierungen gibt es ein Zeitanteilsverfahren bezüglich der Anschlüsse des breiten Speichers **212** für die Lesevorgänge und/oder Schreibvorgänge bezüglich der beiden verschiedenen Operanden (der Parameter und der Partialsumme). Um den Bereich zu optimieren, kann der breite Speicher **212** eine begrenzte Anzahl von Anschlüssen aufweisen. Wenn es eine Notwendigkeit gibt, einen Operanden (z. B. einen Parameter) aus dem breiten Speicher **212** zu lesen und gleichzeitig einen Operanden (z. B. eine Partialsumme) in den breiten Speicher **212** zu schreiben, kann im Ergebnis eine einem speziellen Operanden zugeordnete Pipeline stehenbleiben.

[0081] Im Block **608** erzeugt eine Rechenzelle (die einen MAC-Operator **215** und ein Summenregister **220** aufweist) der Rechenkachel **200** wenigstens eine Ausgangsaktivierung basierend auf den durch die MAC-/Rechenzelle ausgeführten Multiplikationsoperationen. Das Ergebnis der Operationen der MAC-Zelle enthält entweder die Partialsummen, die (während der Partialsummen-Arithmetikoperationen) zurück zu dem breiten Speicher geschrieben werden, oder die Ausgangsaktivierungen, die an den Ausgangsbus **218** gesendet werden. Im Block **610** wendet die NLU **222** der Rechenkachel **200** eine nichtlineare Aktivierungsfunktion auf die Ausgangsaktivierungen an, wobei sie dann die Aktivierungen in den schmalen Speicher **210** schreibt. In einigen Implementierungen ist der Ausgangsbus **218** ein Schieberegister, wobei er ein paralleles Laden der Ergebnisse/Ausgangsaktivierungen von dem MAC-Operator **215** akkumulierten kann, wobei er sie eine auf einmal für die Anwendung der nichtlinearen Funktion und der Schreiboperation in den schmalen Speicher **210** desselben Kachel herausschiebt.

[0082] 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 einer oder mehreren von ihnen verkörpert 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 nicht transitorischen Programmträger für die Ausführung durch eine oder die Steuerung des Betriebs einer Datenverarbeitungsvorrichtung codiert sind, implementiert sein. Alternativ oder zusätzlich können die Programmanweisungen in einem künstlich erzeugten ausgebreiteten Signal, z. B. einem maschinenerzeugten elektrischen, optischen oder elektromagnetischen Signal, codiert sein, das erzeugt wird, um die Informationen für die Übertragung zu einer geeigneten Empfängervorrichtung für die 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 einer oder mehreren von ihnen sein.

[0083] Die in dieser Beschreibung beschriebenen Prozesse und Logikflüsse können durch einen oder mehrere programmierbare Computer ausgeführt werden, die ein oder mehrere Computerprogramme ausführen, um die Funktionen durch das Einwirken auf die Eingangsdaten und das Erzeugen einer Ausgabe (von Ausgaben) auszuführen. Die Prozesse und die Logikflüsse können außerdem durch eine Spezial-Logikschaltungsanordnung, z. B. eine FPGA (eine feldprogrammierbare Anordnung), eine ASIC (eine anwendungsspezifische integrierte Schaltung) oder eine GPGPU (eine Universal-Graphikverarbeitungseinheit), ausgeführt werden und die Vorrichtung kann außerdem als eine Spezial-Logikschaltungsanordnung, z. B. eine FPGA (eine feldprogrammierbare Anordnung), eine ASIC (eine anwendungsspezifische integrierte Schaltung) oder eine GPGPU

(eine Universal-Graphikverarbeitungseinheit), implementiert sein.

[0084] Die für die Ausführung eines Computerprogramms geeigneten Computer enthalten beispielhaft Universal- oder Spezialmikroprozessoren oder beides oder irgendeine andere Art einer Zentraleinheit oder können auf Universal- oder Spezialmikroprozessoren oder beidem oder irgendeiner anderen Art einer Zentraleinheit basieren. Im Allgemeinen empfängt die 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 oder eine oder mehrere Speichervorrichtungen zum Speichern von Anweisungen und Daten. Im Allgemeinen enthält ein Computer außerdem eine oder mehrere Massenspeichervorrichtungen zum Speichern von Daten oder kann ein Computer außerdem betriebstechnisch angeschlossen sein, um Daten von einer oder mehreren Massenspeichervorrichtungen zum Speichern von Daten zu empfangen oder Daten zu einer oder mehreren Massenspeichervorrichtungen zum Speichern von Daten zu senden oder beides, z. B. magnetischen, magnetooptischen Platten oder optischen Platten. Ein Computer muss jedoch derartige Vorrichtungen nicht aufweisen.

[0085] Die computerlesbaren Medien, die zum Speichern von Computerprogrammanweisungen und Daten geeignet sind, enthalten alle Formen nichtflüchtigen Speichers, nichtflüchtiger Medien und nichtflüchtiger Speichervorrichtungen, beispielhaft einschließlich Halbleiterspeichervorrichtungen, z. B. EPROM-, EEPROM- und Flash-Speichervorrichtungen; Magnetplatten, z. B. interner Festplatten oder abnehmbarer Platten. Der Prozessor und der Speicher können durch eine Spezial-Logikschaltungsanordnung ergänzt oder in eine Spezial-Logikschaltungsanordnung aufgenommen sein.

[0086] Während diese Beschreibung viele spezifische Implementierungseinzelheiten enthält, sollten diese nicht als Einschränkungen an den Schutzumfang irgendeiner Erfindung oder an das, was beansprucht sein kann, ausgelegt werden, sondern stattdessen als Beschreibungen der Merkmale, die für spezielle Ausführungsformen der speziellen Erfindungen spezifisch sein können. 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 separat oder in irgendeiner geeigneten Unterkombination implementiert sein. Obwohl die Merkmale oben als in bestimmten Kombinationen wirkend beschrieben sein können 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.

[0087] 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 fortlaufenden 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 oder in mehrere Pakete von Software-Produkten verpackt sein können.

[0088] In den folgenden Beispielen sind weitere Implementierungen zusammengefasst:
Beispiel 1: Eine Computereinheit zum Beschleunigen von Tensorberechnungen, die umfasst: eine erste Speicherbank, die eine erste Datenbreite aufweist, zum Speichern wenigstens einer der Eingangsaktivierungen oder der Ausgangsaktivierungen; eine zweite Speicherbank, die eine zweite Datenbreite, die größer als die erste Datenbreite ist, aufweist, zum Speichern eines oder mehrerer Parameter, die beim Ausführen der Berechnungen verwendet werden; wenigstens eine Zelle, die wenigstens einen Multiplikations-Akkumulations-Operator ("MAC"-Operator) umfasst, der Parameter von der zweiten Speicherbank empfängt und Berechnungen ausführt; eine erste Traversierungseinheit, die mit wenigstens der ersten Speicherbank in Datenverbindung steht, wobei die erste Traversierungseinheit konfiguriert ist, um der ersten Speicherbank ein Steuersignal bereitstellen, um zu veranlassen, das eine Eingangsaktivierung einem Datenbus, der durch den MAC-Operator zugänglich ist, bereitgestellt wird; und wobei die Computereinheit eine oder mehrere Berechnungen ausführt, die wenigstens einem Element eines Datenfeldes zugeordnet sind, wobei die eine oder die mehreren Berechnungen durch den MAC-Operator ausgeführt werden und teilweise eine Multiplikationsoperation der von dem Datenbus empfangenen Eingangsaktivierung und eines von der zweiten Speicherbank empfangenen Parameters umfassen.

[0089] Beispiel 2: Die Computereinheit nach Beispiel 1, wobei die Computereinheit eine oder mehrere der Berechnungen durch das Ausführen einer Schleifenschachtelung ausführt, die mehrere Schleifen umfasst, wobei eine Struktur der Schleifenschachtelung jeweilige Schleifen enthält, die durch die erste Traversierungseinheit verwendet werden, um eine oder mehrere Dimensionen des Datenfeldes zu traversieren.

[0090] Beispiel 3: Die Computereinheit nach Beispiel 2, wobei die eine oder die mehreren Berechnungen teilweise basierend auf einer Tensoroperation ausgeführt werden, die durch die erste Traversierungseinheit bereitgestellt wird, wobei die Tensoroperation eine Schleifenschachtelungsstruktur zum Zugreifen auf ein oder mehrere Elemente des Datenfeldes enthält.

[0091] Beispiel 4: Die Computereinheit nach einem der Beispiele 1 bis 3, die ferner eine zweite Traversierungseinheit umfasst, die konfiguriert ist, um auf wenigstens eine Speicherstelle der ersten Speicherbank und wenigstens eine Speicherstelle der zweiten Speicherbank basierend auf den von einer Quelle, die sich außerhalb der Computereinheit befindet, empfangenen Anweisungen zuzugreifen.

[0092] Beispiel 5: Die Computereinheit nach Beispiel 4, wobei die erste Traversierungseinheit eine Tensoroperations-Traversierungseinheit ist und die zweite Traversierungseinheit eine Speicherdirektzugriffs-Traversierungseinheit ist und wobei das Datenfeld einem Tensor entspricht, der mehrere Elemente umfasst.

[0093] Beispiel 6: Die Computereinheit nach einem der Beispiele 1 bis 5, wobei die Computereinheit eine Nichtlinear-Einheit enthält und ein erster Abschnitt der Berechnungen das Erzeugen einer oder mehrerer Ausgangsaktivierungen basierend auf der Multiplikationsoperation umfasst und ein zweiter Abschnitt der Berechnungen das Anwenden durch die Nichtlinear-Einheit einer nichtlinearen Funktion auf die eine oder die mehreren Ausgangsaktivierungen umfasst.

[0094] Beispiel 7: Die Computereinheit nach Beispiel 6, wobei die eine oder die mehreren durch die Computereinheit ausgeführten Berechnungen die Verwendung eines Schieberegisters umfassen, um die Ausgangsaktivierungen zu der ersten Speicherbank zu schieben.

[0095] Beispiel 8: Die Computereinheit nach einem der Beispiele 1 bis 8, die ferner einen Abschnitt eines Ringbusses umfasst, der außerhalb der Computereinheit verläuft, wobei der Ringbus einen Datenweg zwischen der ersten Speicherbank und einer Speicherbank einer weiteren benachbarten Computereinheit und zwischen der zweiten Speicherbank und einer Speicherbank einer weiteren benachbarten Computereinheit schafft.

[0096] Beispiel 9: Die Computereinheit nach einem der Beispiele 1 bis 8, wobei die zweite Speicherbank konfiguriert ist, um wenigstens eine der Partialsummen oder eine oder mehrere der Eingaben der Vereinigungsschicht zu speichern.

[0097] Beispiel 10: Ein computerimplementiertes Verfahren zum Beschleunigen von Tensorberechnungen, das umfasst: Senden, durch eine erste Speicherbank, die eine erste Datenbreite aufweist, einer ersten Eingangsaktivierung in Reaktion auf die erste Speicherbank, die ein Steuersignal von einer ersten Traversierungseinheit empfängt, wobei die erste Speicherbank in einer Computereinheit angeordnet ist und wobei die erste Eingangsaktivierung durch einen Datenbus bereitgestellt wird, der durch wenigstens eine Zelle der Computereinheit zugänglich ist; Empfangen durch die wenigstens eine Zelle eines oder mehrerer Parameter von einer zweiten Speicherbank, die eine zweite Datenbreite, die größer als die erste Datenbreite ist, aufweist, und wobei die wenigstens eine Zelle wenigstens einen Multiplikations-Akkumulations-Operator ("MAC"-Operator) umfasst; und Ausführen durch den MAC-Operator einer oder mehrerer Berechnungen, die wenigstens einem Element eines Datenfeldes zugeordnet sind, wobei die eine oder die mehreren Berechnungen teilweise eine Multiplikationsoperation wenigstens der ersten Eingangsaktivierung, auf die von dem Datenbus zugegriffen wird, und wenigstens eines Parameters, der von der zweiten Speicherbank empfangen wird, umfassen.

[0098] Beispiel 11: Das computerimplementierte Verfahren nach Beispiel 10, wobei die eine oder die mehreren Berechnungen teilweise basierend auf der Computereinheit ausgeführt werden, die eine Schleifenschachtelung ausführt, die mehrere Schleifen umfasst, wobei eine Struktur der Schleifenschachtelung jeweilige Schleifen enthält, die durch die erste Traversierungseinheit verwendet werden, um eine oder mehrere Dimensionen des Datenfeldes zu traversieren.

[0099] Beispiel 12: Das computerimplementierte Verfahren nach Beispiel 11, das ferner das Bereitstellen durch die erste Traversierungseinheit einer Tensoroperation umfasst, die eine Schleifenschachtelungsstruktur zum Zugreifen auf ein oder mehrere Elemente des Datenfeldes enthält.

[0100] Beispiel 13: Das computerimplementierte Verfahren nach einem der Beispiele 10 bis 12, wobei die erste Traversierungseinheit eine Tensoroperations-Traversierungseinheit ist und die zweite Traversierungseinheit eine Speicherdirektzugriffs-Traversierungseinheit ist und wobei das Datenfeld ei-

nem Tensor entspricht, der mehrere Elemente umfasst.

[0101] Beispiel 14: Das computerimplementierte Verfahren nach einem der Beispiele 10 bis 13, das ferner das Ausführen eines ersten Abschnitts der einen oder der mehreren Berechnungen durch das Erzeugen wenigstens einer Ausgangsaktivierung basierend auf der Multiplikationsoperation umfasst.

[0102] Beispiel 15: Das computerimplementierte Verfahren nach Beispiel 14, das ferner das Ausführen eines zweiten Abschnitts der einen oder der mehreren Berechnungen durch das Anwenden einer nichtlinearen Funktion auf die eine oder die mehreren Ausgangsaktivierungen umfasst.

[0103] Beispiel 16: Nicht transitorisches computerlesbares Speichermedium, das Anweisungen umfasst, die durch einen oder mehrere Prozessoren ausführbar sind, die bei einer derartigen Ausführung den einen oder die mehreren Prozessoren veranlassen, Operationen auszuführen, die umfassen: Senden, durch eine erste Speicherbank, die eine erste Datenbreite aufweist, einer ersten Eingangsaktivierung in Reaktion auf die erste Speicherbank, die ein Steuersignal von einer ersten Traversierungseinheit empfängt, wobei die erste Speicherbank in einer Computereinheit angeordnet ist und wobei die erste Eingangsaktivierung durch einen Datenbus bereitgestellt wird, der durch wenigstens eine Zelle der Computereinheit zugänglich ist; Empfangen durch die wenigstens eine Zelle eines oder mehrerer Parameter von einer zweiten Speicherbank, die eine zweite Datenbreite, die größer als die erste Datenbreite ist, aufweist, und wobei die wenigstens eine Zelle wenigstens einen Multiplikations-Akkumulations-Operator ("MAC"-Operator) umfasst; und Ausführen durch den MAC-Operator einer oder mehrerer Berechnungen, die wenigstens einem Element eines Datenfeldes zugeordnet sind, wobei die eine oder die mehreren Berechnungen teilweise eine Multiplikationsoperation wenigstens einer ersten Eingangsaktivierung, auf die von dem Datenbus zugegriffen wird, und wenigstens eines Parameters, der von der zweiten Speicherbank empfangen wird, umfassen.

[0104] Beispiel 17: Das nicht transitorische computerlesbare Speichermedium nach Beispiel 16, wobei die eine oder die mehreren Berechnungen teilweise basierend auf der Computereinheit ausgeführt werden, die eine Schleifenschachtelung ausführt, die mehrere Schleifen umfasst, wobei eine Struktur der Schleifenschachtelung jeweilige Schleifen enthält, die durch die erste Traversierungseinheit verwendet werden, um eine oder mehrere Dimensionen des Datenfeldes zu traversieren.

[0105] Beispiel 18: Das nicht transitorische computerlesbare Speichermedium nach Beispiel 17, das ferner das Bereitstellen durch die erste Traversierungseinheit einer Tensoroperation umfasst, die eine Schleifenschachtelungsstruktur zum Zugreifen auf ein oder mehrere Elemente des Datenfeldes enthält.

[0106] Beispiel 19: Das nicht transitorische computerlesbare Speichermedium nach einem der Beispiele 16 bis 18, das ferner das Ausführen eines ersten Abschnitts der einen oder der mehreren Berechnungen durch das Erzeugen wenigstens einer Ausgangsaktivierung basierend auf der Multiplikationsoperation umfasst.

[0107] Beispiel 20: Das nicht transitorische computerlesbare Speichermedium nach Beispiel 19, das ferner das Ausführen eines zweiten Abschnitts der einen oder der mehreren Berechnungen durch das Anwenden einer nichtlinearen Funktion auf die eine oder die mehreren Ausgangsaktivierungen umfasst.

[0108] Es sind spezielle Ausführungsformen des Gegenstands beschrieben worden. Weitere Ausführungsformen befinden sich innerhalb des Schutzumfangs der folgenden Ansprüche. Die in den Ansprüchen dargestellten Vorgänge können 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 aufeinanderfolgende Reihenfolge, um die erwünschten Ergebnisse zu erreichen. In bestimmten Implementierungen können Multitasking und Parallelverarbeitung vorteilhaft sein.

Schutzansprüche

1. Computereinheit zum Beschleunigen von Tensorberechnungen, die umfasst:
eine erste Speicherbank, die eine erste Datenbreite aufweist, zum Speichern wenigstens einer der Eingangsaktivierungen oder der Ausgangsaktivierungen;
eine zweite Speicherbank, die eine zweite Datenbreite, die größer als die erste Datenbreite ist, aufweist, zum Speichern eines oder mehrerer Parameter, die beim Ausführen der Berechnungen verwendet werden;
wenigstens eine Zelle, die wenigstens einen Multiplikations-Akkumulations-Operator ("MAC"-Operator) umfasst, der Parameter von der zweiten Speicherbank empfängt und Berechnungen ausführt;
eine erste Traversierungseinheit, die mit wenigstens der ersten Speicherbank in Datenverbindung steht, wobei die erste Traversierungseinheit konfiguriert ist, um der ersten Speicherbank ein Steuersignal bereitzustellen, um zu veranlassen, das eine Eingangsaktivierung einem Datenbus, der durch den MAC-Operator zugänglich ist, bereitgestellt wird; und
wobei die Computereinheit eine oder mehrere Berechnungen ausführt, die wenigstens einem Element

eines Datenfeldes zugeordnet sind, wobei die eine oder die mehreren Berechnungen durch den MAC-Operator ausgeführt werden und teilweise eine Multiplikationsoperation der von dem Datenbus empfangenen Eingangsaktivierung und eines von der zweiten Speicherbank empfangenen Parameters umfassen.

2. Computereinheit nach Anspruch 1, wobei die Computereinheit eine oder mehrere der Berechnungen durch das Ausführen einer Schleifenschachtelung ausführt, die mehrere Schleifen umfasst, wobei eine Struktur der Schleifenschachtelung jeweilige Schleifen enthält, die durch die erste Traversierungseinheit verwendet werden, um eine oder mehrere Dimensionen des Datenfeldes zu traversieren.

3. Computereinheit nach Anspruch 2, wobei die eine oder die mehreren Berechnungen teilweise basierend auf einer Tensoroperation ausgeführt werden, die durch die erste Traversierungseinheit bereitgestellt wird, wobei die Tensoroperation eine Schleifenschachtelungsstruktur zum Zugreifen auf ein oder mehrere Elemente des Datenfeldes enthält.

4. Computereinheit nach einem der Ansprüche 1 bis 3, die ferner eine zweite Traversierungseinheit umfasst, die konfiguriert ist, um auf wenigstens eine Speicherstelle der ersten Speicherbank und wenigstens eine Speicherstelle der zweiten Speicherbank basierend auf den von einer Quelle, die sich außerhalb der Computereinheit befindet, empfangenen Anweisungen zuzugreifen.

5. Computereinheit nach Anspruch 4, wobei die erste Traversierungseinheit eine Tensoroperations-Traversierungseinheit ist und die zweite Traversierungseinheit eine Speicherdirektzugriffs-Traversierungseinheit ist und wobei das Datenfeld einem Tensor entspricht, der mehrere Elemente umfasst.

6. Computereinheit nach einem der Ansprüche 1 bis 5, wobei die Computereinheit eine Nichtlinear-Einheit enthält und ein erster Abschnitt der Berechnungen das Erzeugen einer oder mehrerer Ausgangsaktivierungen basierend auf der Multiplikationsoperation umfasst und ein zweiter Abschnitt der Berechnungen das Anwenden durch die Nichtlinear-Einheit einer nichtlinearen Funktion auf die eine oder die mehreren Ausgangsaktivierungen umfasst.

7. Computereinheit nach Anspruch 6, wobei die eine oder die mehreren durch die Computereinheit ausgeführten Berechnungen die Verwendung eines Schieberegisters umfassen, um die Ausgangsaktivierungen zu der ersten Speicherbank zu schieben.

8. Computereinheit nach einem der Ansprüche 1 bis 7, die ferner einen Abschnitt eines Ringbusses umfasst, der außerhalb der Computereinheit verläuft, wobei der Ringbus einen Datenweg zwischen der ersten Speicherbank und einer Speicherbank einer weiteren benachbarten Computereinheit und zwischen der zweiten Speicherbank und einer Speicherbank einer weiteren benachbarten Computereinheit schafft.

9. Computereinheit nach einem der Ansprüche 1 bis 8, wobei die zweite Speicherbank konfiguriert ist, um wenigstens eine der Partialsummen oder eine oder mehrere der Eingaben der Vereinigungsschicht zu speichern.

10. Nicht transitorisches computerlesbares Speichermedium, das Anweisungen umfasst, die durch einen oder mehrere Prozessoren ausführbar sind, die bei einer derartigen Ausführung den einen oder die mehreren Prozessoren veranlassen, Operationen auszuführen, die umfassen:
Senden, durch eine erste Speicherbank, die eine erste Datenbreite aufweist, einer ersten Eingangsaktivierung in Reaktion auf die erste Speicherbank, die ein Steuersignal von einer ersten Traversierungseinheit empfängt, wobei die erste Speicherbank in einer Computereinheit angeordnet ist und wobei die erste Eingangsaktivierung durch einen Datenbus bereitgestellt wird, der durch wenigstens eine Zelle der Computereinheit zugänglich ist;
Empfangen durch die wenigstens eine Zelle eines oder mehrerer Parameter von einer zweiten Speicherbank, die eine zweite Datenbreite, die größer als die erste Datenbreite ist, aufweist, wobei die wenigstens eine Zelle wenigstens einen Multiplikations-Akkumulations-Operator ("MAC"-Operator) umfasst; und
Ausführen durch den MAC-Operator einer oder mehrerer Berechnungen, die wenigstens einem Element eines Datenfeldes zugeordnet sind, wobei die eine oder die mehreren Berechnungen teilweise eine Multiplikationsoperation wenigstens einer ersten Eingangsaktivierung, auf die von dem Datenbus zugegriffen wird, und wenigstens eines Parameters, der von der zweiten Speicherbank empfangen wird, umfassen.

11. Nicht transitorisches computerlesbares Speichermedium nach Anspruch 10, wobei die eine oder die mehreren Berechnungen teilweise basierend auf der Computereinheit ausgeführt werden, die eine Schleifenschachtelung ausführt, die mehrere Schleifen umfasst, wobei eine Struktur der Schleifenschachtelung die Weise angibt, in der die erste Traversierungseinheit eine oder mehrere Dimensionen des Datenfeldes traversiert.

12. Nicht transitorisches computerlesbares Speichermedium nach Anspruch 11, das ferner das Bereitstellen durch die erste Traversierungseinheit einer Tensoroperation umfasst, die eine Schleifenschach-

telungsstruktur zum Zugreifen auf ein oder mehrere Elemente des Datenfeldes enthält.

13. Nicht transitorisches computerlesbares Speichermedium nach einem der Ansprüche 10 bis 12, wobei die erste Traversierungseinheit eine Tensoroperations-Traversierungseinheit ist und die zweite Traversierungseinheit eine Speicherdirektzugriffs-Traversierungseinheit ist und wobei das Datenfeld einem Tensor entspricht, der mehrere Elemente umfasst.

14. Nicht transitorisches computerlesbares Speichermedium nach einem der Ansprüche 10 bis 13, das ferner das Ausführen eines ersten Abschnitts der einen oder der mehreren Berechnungen durch das Erzeugen wenigstens einer Ausgangsaktivierung basierend auf der Multiplikationsoperation umfasst.

15. Nicht transitorisches computerlesbares Speichermedium nach Anspruch 14, das ferner das Ausführen eines zweiten Abschnitts der einen oder der mehreren Berechnungen durch das Anwenden einer nichtlinearen Funktion auf die eine oder die mehreren Ausgangsaktivierungen umfasst.

Es folgen 6 Seiten Zeichnungen

Anhängende Zeichnungen

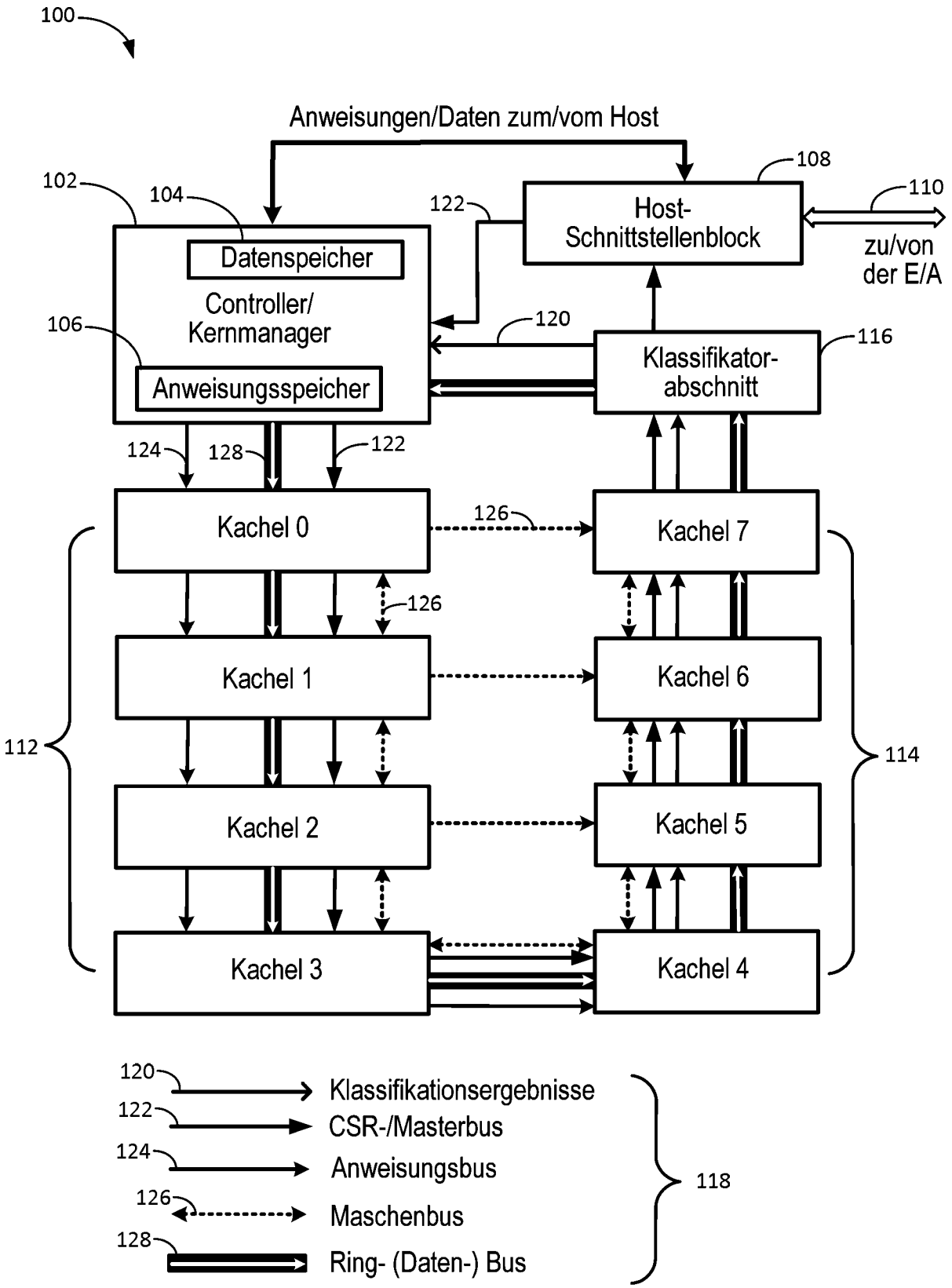


FIG. 1

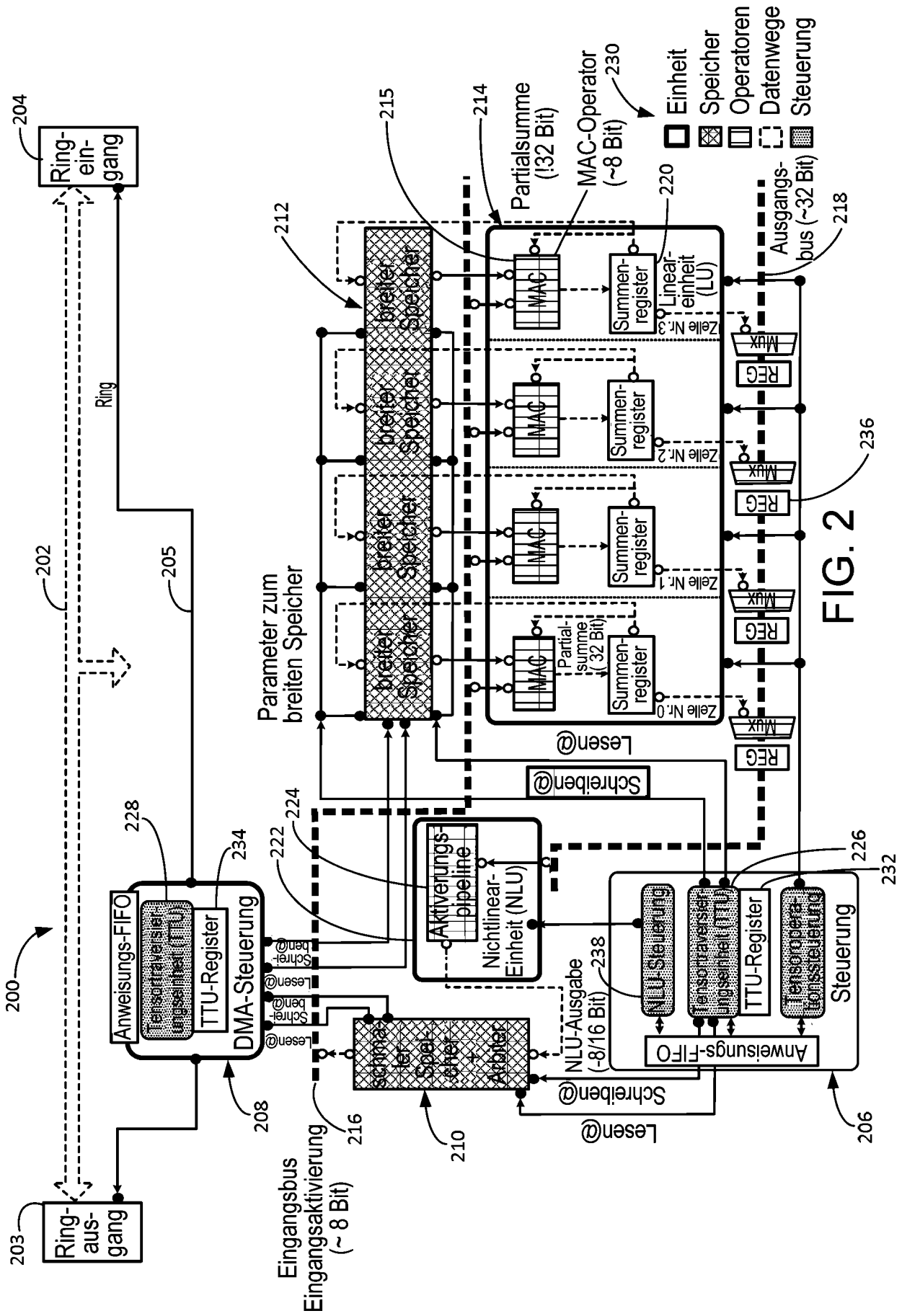

Einheit
Speicher
Operatoren
Datenwege
Steuerung
MAC-Operator (~8 Bit)
Partialsumme (!32 Bit)
Ausgangs-bus (~32 Bit)
Ring-ein-gang
Ring-aus-gang
Ring
breiter Speicher
Parameter zum breiten Speicher
MAC
Summen-register
Linear-einheit (LU)
Partial-summe (~32 Bit)
Zelle Nr. 3
Zelle Nr. 2
Zelle Nr. 1
Zelle Nr. 0
Mux
REG
Lesen@
Schreiben@
Aktivierungs-pipeline
Nichtlinear-Einheit (NLU)
NLU-Ausgabe (~8/16 Bit)
NLU Steuerung
Tensortraverser (TTU)
TTU-Register
Tensoroperation konstellation
Steuerung
Anweisungs-FIFO
Anweisungs-FIFO
DMA-Steuerung
Schreiber
Leser
Arbiter
Eingangsbus Eingangsaktivierung (~ 8 Bit)
200
202
203
204
205
206
208
210
212
214
215
216
218
220
222
224
226
228
230
232
234
236
238

FIG. 2

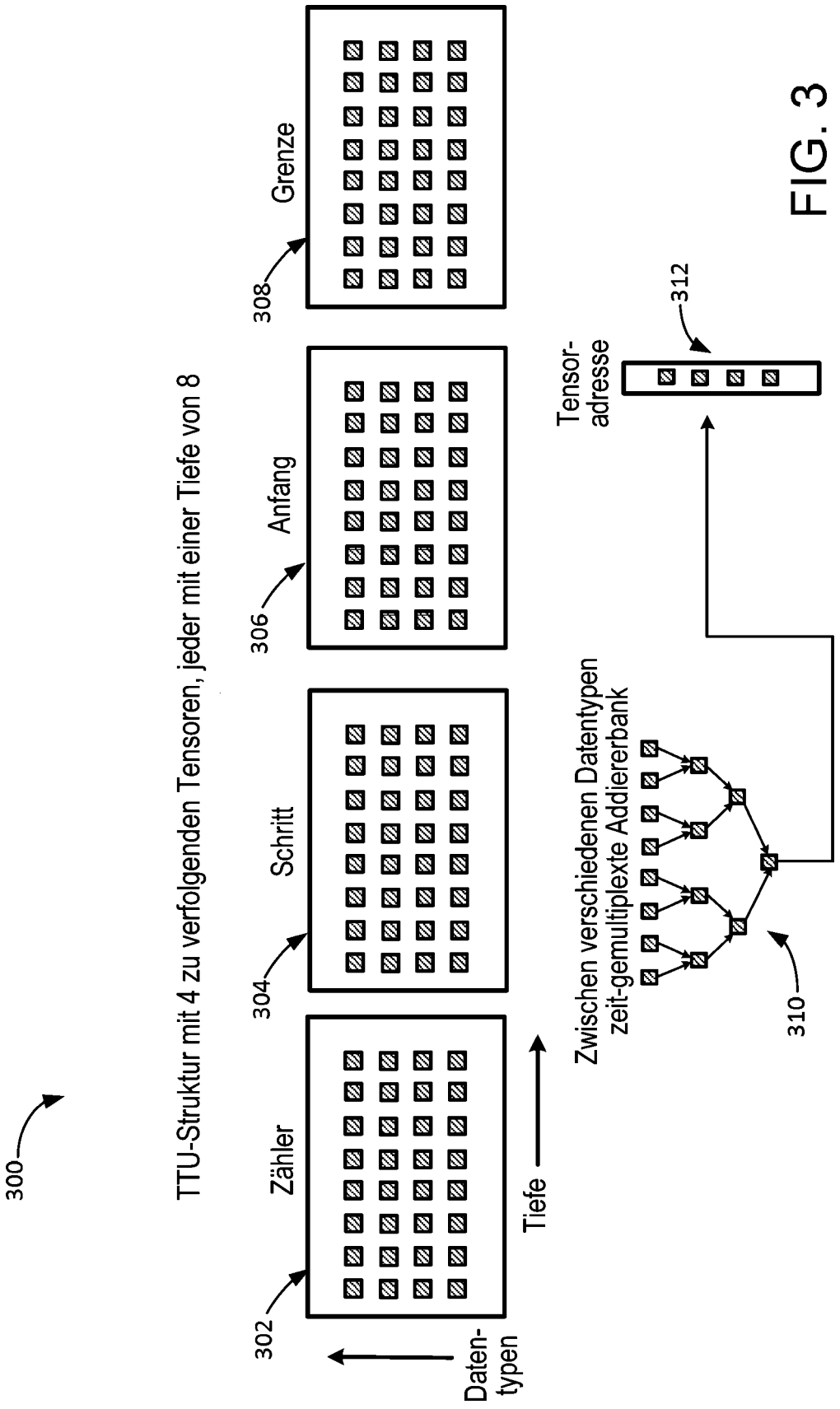

300
TTU-Struktur mit 4 zu verfolgenden Tensoren, jeder mit einer Tiefe von 8
Zähler
Schritt
Anfang
Grenze
302
304
306
308
Daten-typen
Tiefe
Zwischen verschiedenen Datentypen zeit-gemultiplexte Addiererbank
310
Tensor-adresse
312

FIG. 3

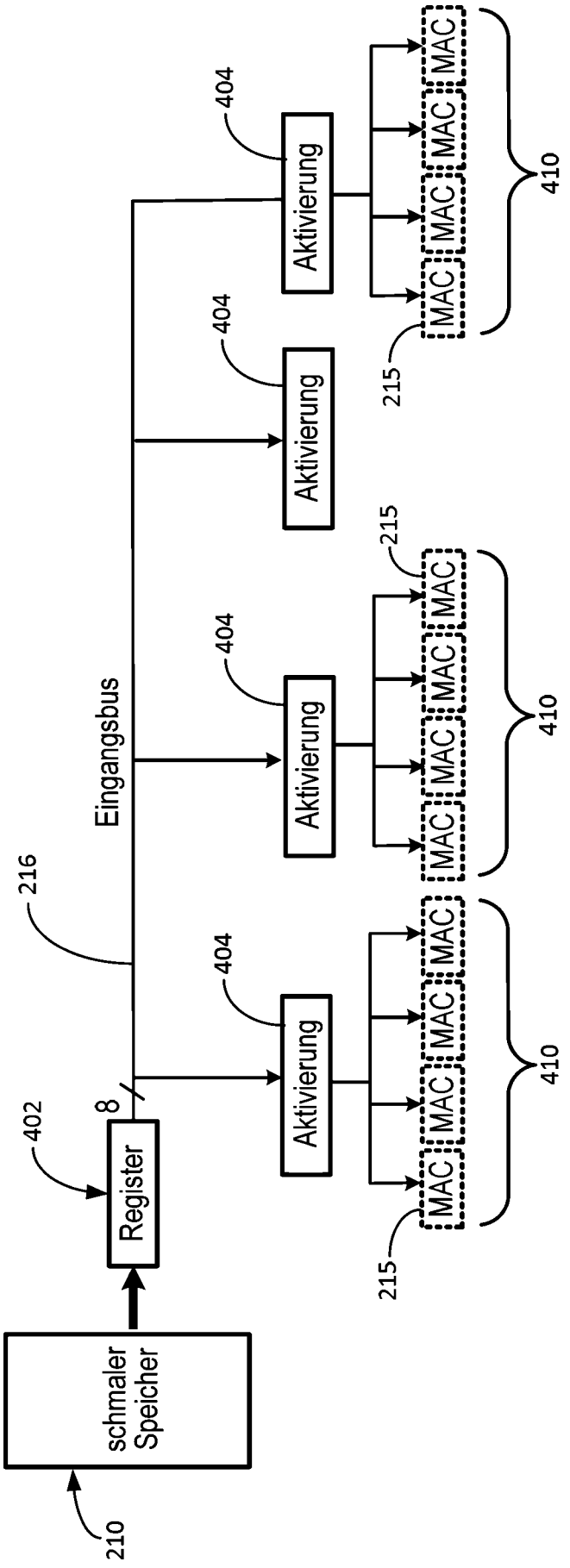

schmaler Speicher
210
Register
402
8
Eingangsbus
216
Aktivierung
404
MAC
215
410

FIG. 4

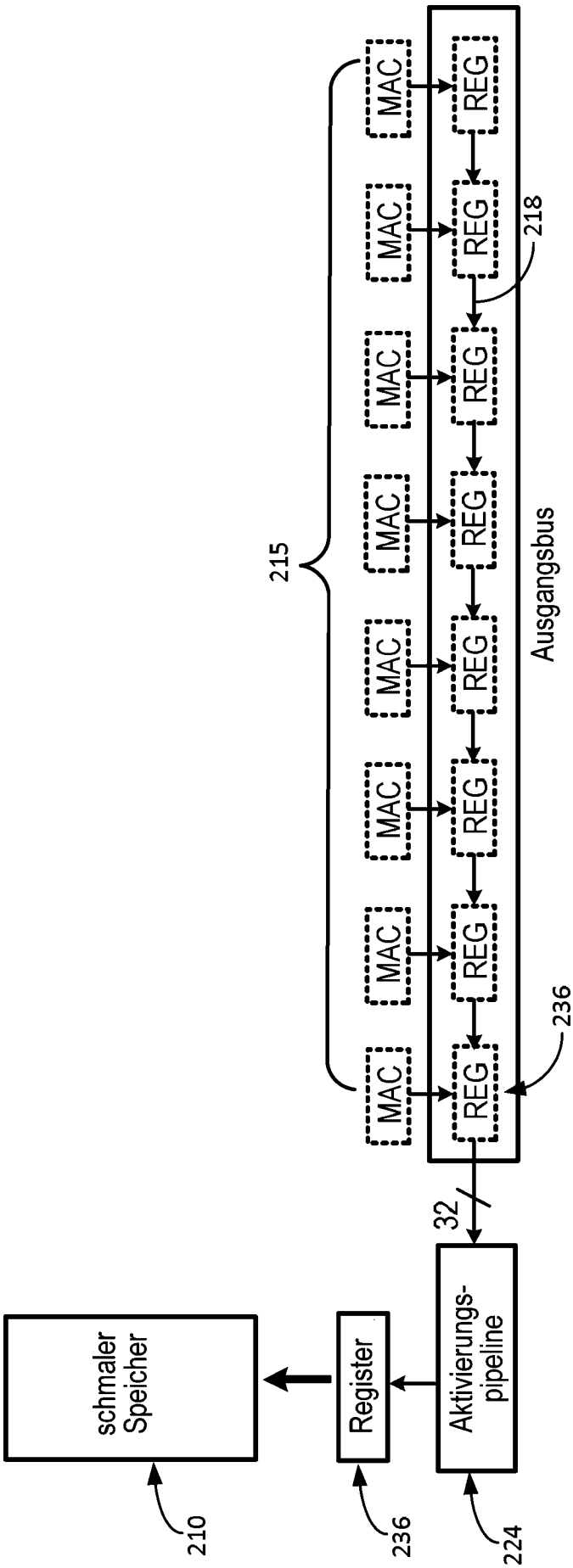

MAC
REG
215
218
236
Ausgangsbus
32
Aktivierungs-pipeline
Register
schmaler Speicher
210
236
224

FIG. 5

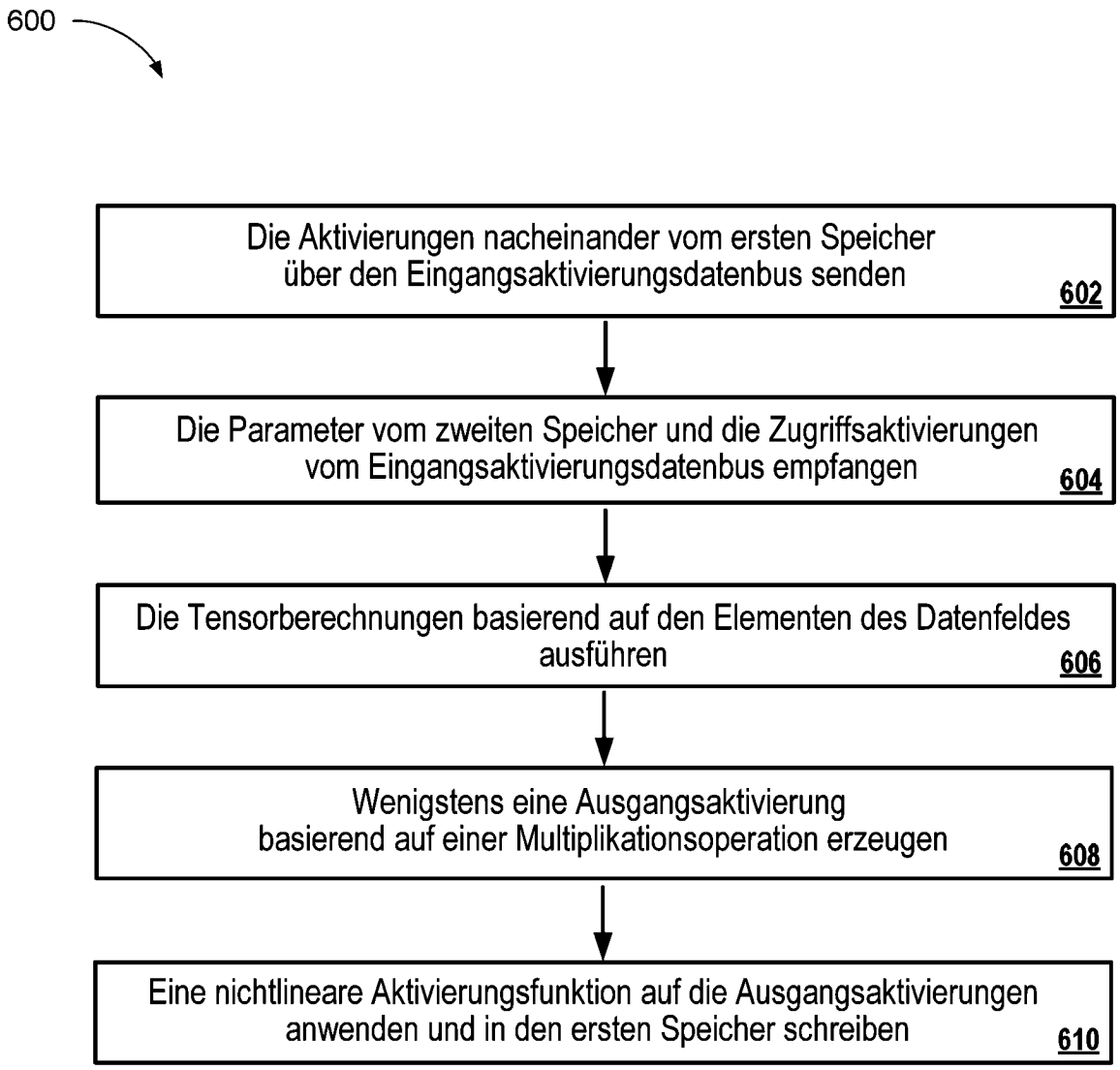

600
Die Aktivierungen nacheinander vom ersten Speicher über den Eingangsaktivierungsdatenbus senden
602
Die Parameter vom zweiten Speicher und die Zugriffsaktivierungen vom Eingangsaktivierungsdatenbus empfangen
604
Die Tensorberechnungen basierend auf den Elementen des Datenfeldes ausführen
606
Wenigstens eine Ausgangsaktivierung basierend auf einer Multiplikationsoperation erzeugen
608
Eine nichtlineare Aktivierungsfunktion auf die Ausgangsaktivierungen anwenden und in den ersten Speicher schreiben
610

FIG. 6