(19) Deutsches Patent- und Markenamt

(10) **DE 10 2013 018 139 A1** 2014.06.26

(12) # Offenlegungsschrift

(21) Aktenzeichen: **10 2013 018 139.9**
(22) Anmeldetag: **04.12.2013**
(43) Offenlegungstag: **26.06.2014**

(51) Int Cl.: ***G06T 1/60*** (2006.01)
G06T 11/00 (2006.01)

(30) Unionspriorität:
13/723,015 **20.12.2012** **US**

(71) Anmelder:
Nvidia Corp., Santa Clara, Calif., US

(74) Vertreter:
Verscht, Thomas K., Dipl.-Phys.(Univ.), 81673, München, DE

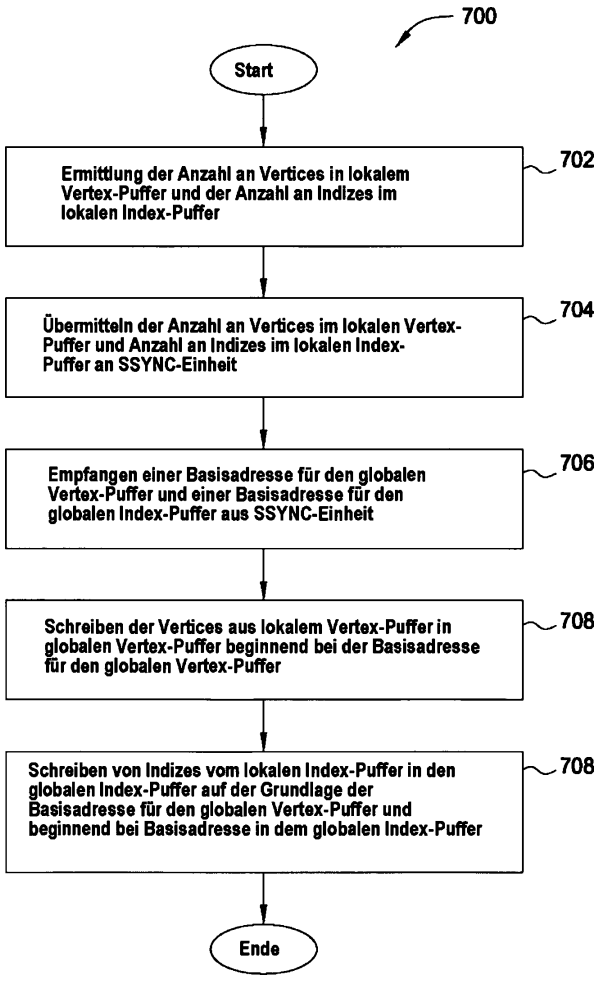
(72) Erfinder:
Duluk jun., Jerome F., Palo Alto, Calif., US; Hakura, Ziyad S., Gilroy, Calif., US; Moreton, Henry Packard, Woodside, Calif., US

Prüfungsantrag gemäß § 44 PatG ist gestellt.

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

(54) Bezeichnung: **Technik zur Speicherung gemeinsamer Vertices**

(57) Zusammenfassung: Eine grafische Verarbeitungseinheit umfasst eine Gruppe aus Geometrie-Verarbeitungseinheiten, wovon jede ausgebildet ist, grafische Grundelemente parallel zueinander zu verarbeiten. Eine gegebene Geometrie-Verarbeitungseinheit erzeugt einen oder mehrere grafische Grundelemente oder geometrische Objekte und zwischenspeichert die zugehörigen Vertex-Daten lokal. Die Geometrie-Verarbeitungseinheit zwischenspeichert ferner unterschiedliche Gruppen an Indizes für diese Vertices, wobei jede derartige Gruppe ein anderes grafisches Grundelement oder geometrisches Objekt repräsentiert. Die Geometrie-Verarbeitungseinheit kann dann die zwischengespeicherten Vertices und Indizes an globale Puffer parallel zu anderen als Datenstrom übermitteln. Eine Datenstromausgabesynchronisier-Einheit koordiniert die parallele Zuleitung als Datenstrom von Vertices und Indizes, indem jede Geometrie-Verarbeitungseinheit mit einer anderen Basisadresse für einen globalen Vertex-Puffer versehen wird, an der Vertices geschrieben werden können. Die Datenstromausgabesynchronisier-Einheit stellt ferner für jede Geometrie-Verarbeitungseinheit eine andere Basisadresse für einen globalen Index-Puffer bereit, an der Indizes geschrieben werden können.



Beschreibung

HINTERGRUND DER ERFINDUNG

Gebiet der Erfindung

[0001] Die vorliegende Erfindung betrifft allgemein die Verarbeitung von Grafikdaten und insbesondere eine Technik zur Speicherung gemeinsamer Vertices bzw. Eckpunkte.

Beschreibung des Stands der Technik

[0002] Eine konventionelle grafische Verarbeitungseinheit (GPU) realisiert typischerweise eine Grafikverarbeitungs-Pipeline, die eine Abfolge von grafischen Verarbeitungsstufen enthält. In jeder Stufe in der Grafikverarbeitungs-Pipeline kann die GPU eine oder mehrere unterschiedliche Grafik bezogene Verarbeitungsoperationen ausführen. Beispielsweise kann in einer Stufe die GPU eine Gruppe aus primitiven Elementen bzw. Grundelementen zusammenstellen, die eine Grafikszene repräsentieren, und in einer nachfolgenden Stufe kann die GPU Schattierungsoperationen an Vertices ausführen, die mit dieser Gruppe aus Grundelementen verknüpft sind. Schließlich kann die GPU diese Vertices in Pixel als Raster einteilen, die die Grafikszene repräsentieren.

[0003] Eine GPU, die eine konventionelle Grafikverarbeitungs-Pipeline realisiert, etwa wie sie in dem vorhergehenden Beispiel beschrieben ist, enthält typischerweise eine Geometrie-Schattierungseinheit, die ausgebildet ist, Schattierungsoperationen an Vertices und einer Geometrie-basierten Information auszuführen und dann ein oder mehrere grafische Grundelemente oder ein oder mehrere geometrische Objekte mit relativ größerer Komplexität an eine nachfolgende Einheit für die Rastereinteilung auszugeben. Für jedes erzeugte grafische Grundelement oder geometrische Objekt gibt die Geometrie-Schattierungseinheit Vertex-Daten aus, die jedem Vertex entsprechen, der mit diesem grafischen Grundelement oder geometrischen Objekt verknüpft ist. Wenn beispielsweise ein Dreieck verarbeitet wird, würde die Geometrie-Schattierungseinheit Vertex-Daten für jeden der drei Vertices bzw. Eckpunkte dieses Dreiecks ausgeben. Vertex-Daten für einen gegebenen Vertex können die Position des Vertex innerhalb der Szene, Abdeckungsdaten, die zu dem Vertex gehören, oder eine Gruppe aus Attributen beschreiben, die zu dem Vertex gehören, um nur einige Dinge auszuführen. Wenn grafische Grundelemente oder geometrische Objekte erzeugt werden, dann speichert die Geometrie-Schattierungseinheit typischerweise jedes erzeugte grafische Grundelement oder jedes grafische Grundelement, das einen Teil oder das gesamte erzeugte geometrische Objekt ausmacht, als die Gruppe an Vertices, die mit diesem Grundelement und den Vertex-Daten, die jedem Vertex in dieser Gruppe entsprechen, verknüpft ist.

[0004] Wiederum kann in einigen Situationen die Geometrie-Schattierungseinheit ein geometrisches Objekt erzeugen, das eine Ansammlung miteinander verbundener grafischer Grundelemente enthält, die gemeinsame Vertices bzw. Eckpunkte besitzen. Das geometrische Objekt könnte beispielsweise ein fächerartiges, ein streifenartiges oder ein gitterartiges geometrisches Objekt sein. Beispielsweise könnte eine gegebene Grafikszene zahlreiche einzelne grafische Grundelemente enthalten, die in einem Fächer, einem Streifen oder einem Gitter miteinander verbunden sind, um die Erscheinung einer Oberfläche mit einer beliebigen Form zu erzeugen. Jedes grafische Grundelement in der Oberfläche könnte mit einem benachbarten grafischen Grundelement durch einen oder mehrere Eckpunkte verbunden sein, die gemeinsam für die beiden grafischen Grundelemente sind. In anderen Situationen können mehrere geometrische Objekte, etwa Dreiecke oder Streifen, die einen oder mehrere Eckpunkte gemeinsam haben, von der Geometrie-Schattierungseinheit erzeugt werden.

[0005] In diesen unterschiedlichen Situationen speichert die Geometrie-Schattierungseinheit typischerweise redundante Kopien der Vertex-Daten, die mit jedem Eckpunkt verknüpft sind, der in grafischen Grundelementen oder geometrischen Objekten gemeinsam auftritt. Jedoch ist diese Vorgehensweise problematisch, da eine typische Grafikszene Millionen an gemeinsamen Eckpunkten enthalten kann. Folglich speichert eine konventionelle Geometrie-Schattierungseinheit unter Umständen Millionen an Kopieren redundanter Daten. Die Verarbeitung dieser redundanten Daten verbraucht GPU-Ressourcen in ineffizienter Weise und kann die Geschwindigkeit verringern, mit der eine Grafikszene erzeugt wird.

[0006] Was daher auf diesem Gebiet der Technik benötigt wird, ist eine verbesserte Technik zur Verarbeitung von Eckpunkten, die in grafischen Grundelementen oder geometrischen Objekten in einer Grafikszene gemeinsam auftreten.

ÜBERBLICK ÜBER DIE ERFINDUNG

[0007] Eine Ausführungsform der vorliegenden Erfindung gibt ein computerimplementiertes Verfahren zum Auffüllen mehrerer globaler Puffer an, mit: Epfangen von Daten aus einer ersten Verarbeitungseinheit, die eine Anzahl an Einträgen in einem ersten lokalen Index-Puffer angeben, und die von der ersten Verarbeitungseinheit erzeugt wurden, wobei die erste Verarbeitungseinheit eine von mehreren Verarbeitungseinheiten ist, und Senden einer ersten Basisadresse für einen globalen Index-Puffer an die erste Verarbeitungseinheit, die eine erste Position in dem globalen Index-Puffer angibt, an der die Einträge aus

dem ersten lokalen Index-Puffer geschrieben werden sollten, wobei die erste Basisadresse für den ersten globalen Index-Puffer auf einer Anzahl an Einträgen in dem globalen Index-Puffer beruht, die bereits mindestens einer anderen Verarbeitungseinheit, aus den mehreren Verarbeitungseinheiten, zugewiesen ist.

[0008] Ein Vorteil der offenbarten Vorgehensweise besteht darin, dass redundante Kopien von Vertex-Daten weder in den Vertex-Puffern, die lokal für die unterschiedlichen Geometrie-Schattierungseinheiten vorgesehen sind, oder in dem globalen Vertex-Puffer gespeichert werden, da die Vertex-Daten indiziert sind, wodurch eine Einsparung von Ressourcen von Verarbeitungseinheiten und eine Erhöhung der gesamten Verarbeitungseffizienz erreicht werden.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

[0009] Um die Art und Weise, in der die oben genannten Merkmale der vorliegenden Erfindung detailliert verstanden werden können, anzugeben, wird eine speziellere Beschreibung der Erfindung, die zuvor kurz zusammengefasst ist, mit Bezug zu Ausführungsformen angegeben, wovon einige in den angefügten Zeichnungen dargestellt sind. Es ist jedoch zu beachten, dass die angefügten Zeichnungen nur typische Ausführungsformen dieser Erfindung darstellen und daher nicht beabsichtigen, ihren Schutzbereich zu beschränken, da die Erfindung andere gleichermaßen wirksame Ausführungsformen zulässt.

[0010] **Fig.** 1 ist eine Blockansicht, die ein Computersystem darstellt, das ausgebildet ist, einen oder mehrere Aspekte der vorliegenden Erfindung zu realisieren;

[0011] **Fig.** 2 ist eine Blockansicht eines Parallelverarbeitungssubsystems für das Computersystem aus **Fig.** 1 gemäß einer Ausführungsform der vorliegenden Erfindung;

[0012] **Fig.** 3 ist eine Blockansicht eines Teils eines Datenstrom-Multiprozessors in dem allgemeinen Verarbeitungs-Cluster aus **Fig.** 2 gemäß einer Ausführungsform der vorliegenden Erfindung;

[0013] **Fig.** 4 ist eine Konzeptansicht einer Grafikverarbeitungs-Pipeline, die eine oder mehrere Parallelverarbeitungseinheiten zeigt, in die zur Realisierung eine oder mehrere der Parallelverarbeitungseinheiten aus **Fig.** 2 gemäß einer Ausführungsform der vorliegenden Erfindung konfiguriert werden können;

[0014] **Fig.** 5 ist eine Konzeptansicht einer Ansammlung an Geometrie-Verarbeitungseinheiten gemäß einer Ausführungsform der vorliegenden Erfindung;

[0015] **Fig.** 6 ist ein Flussdiagramm von Verfahrensschritten zur Speicherung von Vertex-Daten und Index-Daten in mehreren lokalen Puffern gemäß einer Ausführungsform der vorliegenden Erfindung;

[0016] **Fig.** 7 ist ein Flussdiagramm von Verfahrensschritten zur Zuleitung von Vertices und in Indizes als Datenstrom zu den mehreren lokalen Puffern gemäß einer Ausführungsform der vorliegenden Erfindung;

[0017] **Fig.** 8 ist ein Flussdiagramm von Verfahrensschritten zur Anreicherung mehrerer globaler Puffer gemäß einer Ausführungsform der vorliegenden Erfindung;

[0018] **Fig.** 9 ist eine Konzeptansicht, die anschauliche Geometrie-Verarbeitungseinheiten zeigt, die ausgebildet sind, Indizes und Vertices bzw. Eckpunkte lokal gemäß einer Ausführungsform der vorliegenden Erfindung zu speichern; und

[0019] **Fig.** 10 ist eine Konzeptansicht, die anschauliche globale Puffer zeigt, die ausgebildet sind, Indizes oder Vertices gemäß einer Ausführungsform der vorliegenden Erfindung zu speichern.

DETAILLIERTE BESCHREIBUNG

[0020] In der folgenden Beschreibung werden zahlreiche spezielle Details angegeben, um ein gründlicheres Verständnis der vorliegenden Erfindung zu ermöglichen. Jedoch erkennt der Fachmann, dass die vorliegende Erfindung auch ohne eines oder mehrerer dieser speziellen Details in die Praxis umgesetzt werden kann.

Systemüberblick

[0021] **Fig.** 1 ist eine Blockansicht, die ein Computersystem **100** darstellt, das ausgebildet ist, einen oder mehrere Aspekte der vorliegenden Erfindung zu realisieren. Das Computersystem **100** umfasst eine zentrale Recheneinheit (CPU) **102** und einen Systemspeicher **104**, die über einen Verbindungspfad, der eine Speicherbrücke **105** enthalten kann, miteinander in Verbindung stehen. Die Speicherbrücke **105**, die beispielsweise ein Nordbrücken-Chip sein kann, ist über einen Bus oder einen anderen Kommunikationspfad **106** (beispielsweise eine HyperTransport-Verbindung) mit einer I/O-(Eingabe/Ausgabe-)Brücke **107** verbunden. Die I/O-Brücke **107**, die beispielsweise ein Südbrücken-Chip sein kann, empfängt eine Anwendereingabe aus einem oder mehreren Anwender-Eingabegeräten **108** (beispielsweise Tastatur, Maus) und leitet die Eingabe an die CPU **102** über den Kommunikationspfad **106** und die Speicherbrücke **105** weiter. Ein Parallelverarbeitungssubsystem **112** ist mit der Speicherbrücke **105** über einen Bus oder einen zweiten Kommunikationspfad **113** (beispielsweise ein peripherer Komponenten-Verbindung-(PCI)Express, ein beschleunigter Graphikport oder eine HyperTrans-

port-Verbindung) verbunden. In einer Ausführungsform ist das Parallelverarbeitungssubsystem **112** ein Grafiksubsystem, das Pixel einem Anzeigegerät **110** zuleitet, das eine konventionelle Kathodenstrahlröhre, eine Flüssigkristallanzeige, eine Anzeige mit lichtemittierenden Dioden oder dergleichen sein kann. Eine Systemdiskette **114** ist ebenfalls mit der I/O-Brücke **107** verbunden und kann ausgebildet sein, Inhalt und Anwendungen und Daten zur Verwendung durch die CPU **102** und das Parallelverarbeitungssubsystem **112** zu speichern. Die Systemdiskette **114** stellt nicht-flüchtigen Speicherplatz für Anwendungen und Daten bereit und kann fest installierte oder entfernbare Festplattenlaufwerke, Flash-Speichereinrichtungen und CD-(Kompaktdisketten-Nur-Lese-Speicher), DVD-(digitale Vielseitigkeitsdisketten-ROM), Blu-ray, HD-DVD (hochauflösende DVD) oder andere magnetische, optische Speichereinrichtungen oder Halbleiterspeichereinrichtungen umfassen.

[0022] Ein Schalter bzw. eine Schalteinrichtung **116** stellt Verbindungen zwischen der I/O-Brücke **107** und anderen Komponenten, etwa einem Netzwerkadapter **118** und diversen Zusatzkarten **120** und **121** bereit. Andere Komponenten (nicht explizit gezeigt) einschließlich eines universellen seriellen Busses (USB) oder andere Portverbindungen, Kompaktdisketten-(CD-)Laufwerke, digitale Vielseitigkeitsdisketten-(DVD-)Laufwerke, Filmaufzeichnungsgeräte und dergleichen, können ebenfalls mit der I/O-Brücke **107** verbunden sein. Die diversen Kommunikationspfade, die in **Fig.** 1 gezeigt sind, einschließlich der speziell genannten Kommunikationspfade **106** und **113**, können unter Anwendung beliebiger geeigneter Protokolle realisiert werden, etwa durch PCI-Express, AGP (beschleunigter Graphikport), HyperTransport oder durch ein oder mehrere andere Bus- oder Punkt-Zu-Punkt-Kommunikationsprotokolle, und Verbindungen zwischen unterschiedlichen Geräten können unterschiedliche Protokolle benutzen, wie dies im Stand der Technik bekannt ist.

[0023] In einer Ausführungsform enthält das Parallelverarbeitungssubsystem **112** eine Schaltung, die für Grafikverarbeitung und Videoverarbeitung optimiert ist, wozu beispielsweise eine Videoausgabeschaltung gehört, und das System bildet eine grafische Verarbeitungseinheit (GPU). In einer weiteren Ausführungsform enthält das Parallelverarbeitungssubsystem **112** eine Schaltung, die für die Verarbeitung für Allgemeinzwecke optimiert ist, wobei dennoch die zu Grunde liegende Rechenarchitektur, die nachfolgend detaillierter beschrieben ist, beibehalten ist. In einer noch weiteren Ausführungsform kann das Parallelverarbeitungssubsystem **112** mit einem oder mehreren anderen Systemelementen zu einem einzelnen Subsystem zusammengefasst sein, etwa durch Vereinigen der Speicherbrücke **105**, der CPU **102** und der I/O-Brücke **107**, um ein System auf einem Chip (SoC) zu bilden.

[0024] Zu beachten ist, dass das hierin gezeigte System anschaulicher Natur ist und dass Variationen und Modifizierungen möglich sind. Die Verbindungstopologie, einschließlich der Anzahl und Anordnung von Brücken, die Anzahl an CPUs **102** und die Anzahl an Parallelverarbeitungssubsystemen **112** kann nach Bedarf modifiziert werden. Beispielsweise ist in einigen Ausführungsformen der Systemspeicher **104** direkt mit der CPU **102** anstatt über eine Brücke verbunden, und andere Geräte kommunizieren mit dem Systemspeicher **104** über die Speicherbrücke **105** und die CPU **102**. In anderen alternativen Topologien ist das Parallelverarbeitungssubsystem **112** mit der I/O-Brücke **107** oder direkt mit der CPU **102** anstatt mit der Speicherbrücke **105** verbunden. In noch anderen Ausführungsformen können die I/O-Brücke **107** und die Speicherbrücke **105** in einem einzelnen Chip integriert sein, anstatt dass sie als ein oder mehrere diskrete Bauelemente vorhanden sind. Große Ausführungsformen können zwei oder mehr CPUs **102** und zwei oder mehr Parallelverarbeitungssubsysteme **112** aufweisen. Die speziellen Komponenten, die hierin gezeigt sind, sind optional; beispielsweise kann eine beliebige Zahl an Zusatzkarten oder peripheren Geräten unterstützt werden. In einigen Ausführungsformen ist der Schalter **116** weggelassen, und der Netzwerkadapter **118** und die Zusatzkarten **120**, **121** sind direkt mit der I/O-Brücke **107** verbunden.

[0025] **Fig.** 2 zeigt ein Parallelverarbeitungssubsystem **112** gemäß einer Ausführungsform der vorliegenden Erfindung. Wie gezeigt, umfasst das Parallelverarbeitungssubsystem **112** eine oder mehrere Parallelverarbeitungseinheiten (PPUs) **202**, wovon jede mit einem lokalen Parallelverarbeitungs-(PP)Speicher **204** verbunden ist. Im Allgemeinen enthält ein Parallelverarbeitungssubsystem eine Anzahl U an PPUs, wobei $U \geq 1$ ist. (Hierin werden mehrere Instanzen gleicher Objekte mit Bezugszeichen belegt, die das Objekt kennzeichnen, und Zahlen in Klammern geben bei Bedarf die Instanz an.) Die PPUs **202** und die Parallelverarbeitungsspeicher **204** können unter Anwendung einer oder mehrerer integrierter Schaltungseinrichtungen, etwa durch programmierbare Prozessoren, anwendungsspezifische integrierte Schaltungen (ASIC), oder Speichereinrichtungen oder durch eine andere technisch machbare Weise realisiert werden.

[0026] Es sei wieder auf **Fig.** 1 sowie auf **Fig.** 2 verwiesen; in einigen Ausführungsformen sind einige oder alle der PPUs **202** in dem Parallelverarbeitungssubsystem **112** Grafikprozessoren mit Bilderzeugungs-Pipelines, die konfiguriert werden können, um diverse Operationen auszuführen, die betreffen: die Erzeugung von Pixeldaten aus Grafikdaten, die von der CPU **102** und/oder dem System-

speicher **104** über die Speicherbrücke **105** und den zweiten Kommunikationspfad **113** bereitgestellt werden, die Wechselwirkung mit dem lokalen Parallelverarbeitungsspeicher **204** (der als Grafikspeicher verwendbar ist und beispielsweise einen konventionellen Blockpuffer enthält), um Pixeldaten zu speichern und zu aktualisieren, die Zuleitung von Pixeldaten an das Anzeigegerät **110**, und dergleichen. In einigen Ausführungsformen kann das Parallelverarbeitungssubsystem **112** eine oder mehrere PPUs **202** aufweisen, die als Grafikprozessoren arbeiten, und kann eine oder mehrere andere PPUs **202** aufweisen, die für Berechnungen für Allgemeinzwecke verwendet werden. Die PPUs **202** können identisch oder unterschiedlich sein, und jede PPU **202** kann einen oder mehrere spezielle Parallelverarbeitungsspeichereinrichtungen oder keinen speziellen Parallelverarbeitungsspeicher aufweisen. Eine oder mehrere PPUs **202** in dem Parallelverarbeitungssubsystem **112** können Daten an das Anzeigegerät **110** ausgeben, oder jede PPU **202** in dem Parallelverarbeitungssubsystem **112** kann Daten an ein oder mehrere Anzeigegeräte **110** ausgeben.

[0027] Während des Betriebs ist die CPU **102** der übergeordnete Prozessor des Computersystems **100** und steuert und koordiniert den Betrieb anderer Systemkomponenten. Insbesondere gibt die CPU **102** Befehle aus, die den Betrieb der PPUs **202** steuern. In einigen Ausführungsformen schreibt die CPU **102** einen Strom aus Befehlen für jede PPU **202** in eine Datenstruktur (die in **Fig.** 1 oder **Fig.** 2 nicht explizit gezeigt ist), die in dem Systemspeicher **104**, dem Parallelverarbeitungsspeicher **204** oder an einer weiteren Speicherstelle liegen kann, auf die sowohl die CPU **102** als auch die PPU **202** zugreifen kann. Ein Zeiger auf jede Datenstruktur wird in einen Schiebepuffer geschrieben, um die Verarbeitung des Stroms aus Befehlen in der Datenstruktur zu initiieren. Die PPU **202** liest Befehlsströme aus einem oder mehreren Schiebepuffern aus und führt die Befehle asynchron relativ zu der Arbeitsweise der CPU **102** aus. Es können für jeden Schiebepuffer durch ein Anwendungsprogramm über den Gerätetreiber **103** Prioritäten für die Ausführung angegeben werden, um die Disponierung der unterschiedlichen Schiebepuffer zu steuern.

[0028] Es sei nun wieder auf **Fig.** 2 sowie auf **Fig.** 1 verwiesen; jede PPU **202** enthält eine I/O-(Eingabe/Ausgabe-)Einheit **205**, die mit dem Rest des Computersystems **100** über den Kommunikationspfad **113** in Verbindung steht, der mit der Speicherbrücke **105** (oder in einer alternativen Ausführungsform direkt mit der CPU **102**) in Verbindung steht. Die Verbindung der PPU **202** mit dem Rest des Computersystems **100** kann auch anders sein. In einigen Ausführungsformen ist das Parallelverarbeitungssubsystem **112** als eine Zusatzkarte realisiert, die in einen Erweiterungssteckplatz des Computersystems **100** eingeführt werden kann. In anderen Ausführungsformen kann eine PPU **202** in einem einzelnen Chip zusammen mit einer Busbrücke, etwa der Speicherbrücke **105** oder der I/O-Brücke **107** integriert sein. In noch anderen Ausführungsformen können einige oder alle Elemente der PPU **202** zusammen mit der CPU **102** in einem einzelnen Chip integriert sein.

[0029] In einer Ausführungsform ist der Kommunikationspfad **113** eine PCI-Expressverbindung, in der jeder PPU **202** spezielle Bahnen zugeordnet sind, wie dies im Stand der Technik bekannt ist. Es können auch andere Kommunikationspfade verwendet werden. Eine I/O-Einheit **205** erzeugt Pakete (oder andere Signale) zur Übertragung auf dem Kommunikationspfad **113** und empfängt ferner alle eintreffenden Pakete (oder andere Signale) aus dem Kommunikationspfad **113**, wodurch die eintreffenden Pakete zu geeigneten Komponenten der PPU **202** weitergeleitet werden. Beispielsweise können Befehle, die Verarbeitungsaufgaben betreffen, an eine Hauptschnittstelle **206** geleitet werden, während Befehle, die Speicheroperationen (beispielsweise das Lesen aus dem oder Schreiben in den Parallelverarbeitungsspeicher **204**) betreffen, an eine Speicherkreuzungseinheit **210** geleitet werden können. Die Hauptschnittstelle **206** liest jeden Schiebepuffer aus und gibt den in dem Schiebepuffer gespeicherten Befehlsstrom an einen Frontbereich **212** aus.

[0030] Jede PPU **202** realisiert vorteilhafterweise eine äußerst parallele Verarbeitungsarchitektur. Wie detailliert gezeigt ist, enthält die PPU **202(0)** ein Verarbeitung-Cluster-Array **230**, das eine Anzahl C an allgemeinen Verarbeitungs-Clustern (GPCs) **208** enthält, wobei C ≥ 1 ist. Jeder GPC **208** ist in der Lage, eine große Anzahl (beispielsweise hunderte oder tausende) Stränge gleichzeitig auszuführen, wobei jeder Strang eine Instanz eines Programms ist. In diversen Anwendungen können unterschiedliche GPCs **208** zur Verarbeitung unterschiedlicher Arten von Programmen oder zur Ausführung unterschiedlicher Arten von Berechnungen reserviert werden. Die Reservierung bzw. Zuweisung von GPCs **208** kann in Abhängigkeit von der Arbeitslast unterschiedlich sein, die sich für jede Art von Programm oder Berechnung ergibt.

[0031] Die GPCs **208** empfangen zu verarbeitende Ausführungsaufgabe aus einer Arbeitsverteilungseinheit in einer Aufgaben/Arbeitseinheit **207**. Die Arbeitsverteilungseinheit empfängt Zeiger auf Verarbeitungsaufgaben, die als Aufgaben-Metadaten (TMD) (nicht gezeigt) codiert und im Speicher abgelegt sind. Die Zeiger auf die TMD sind in dem Befehlsstrom enthalten, der als ein Schiebepuffer gespeichert ist und von der Frontbereichseinheit **212** aus der Hauptschnittstelle **206** empfangen wird. Verarbeitungsaufgaben, die als TMD codiert sein können, enthalten Indizes von zu verarbeitenden Daten, sowie Zustands-

parameter und Befehle, die definieren, wie die Daten zu verarbeiten sind (beispielsweise welches Programm auszuführen ist). Die Aufgaben/Arbeitseinheit **207** empfängt Aufgaben aus dem Frontbereich **212** und stellt sicher, dass die GPCs **208** in einen zulässigen Zustand konfiguriert werden, bevor die Verarbeitung, wie sie durch jeden Satz der TMD spezifiziert ist, initiiert wird. Es kann eine Priorität für jeden Satz an TMD angegeben werden, die verwendet wird, um die Ausführung der Verarbeitungsaufgaben zu disponieren. Verarbeitungsaufgaben können auch aus dem Verarbeitungs-Cluster-Array **230** empfangen werden. Optional können die TMD einen Parameter enthalten, der steuert, ob die TMD dem Anfang oder dem Ende einer Liste von Verarbeitungsaufgaben (oder einer Liste aus Zeigern auf die Verarbeitungsaufgaben) hinzuzufügen sind, wodurch eine weitere Ebene einer Steuerung zusätzlich zur Priorität bereitgestellt wird.

[0032] Eine Speicherschnittstelle **214** enthält eine Anzahl D an Partitionseinheiten **215**, die jeweils direkt mit einem Teil des Parallelverarbeitungsspeichers **204** verbunden sind, wobei D ≥ 1 ist. Wie gezeigt, ist generell die Anzahl an Partitionseinheiten **215** gleich der Anzahl an dynamischen Speichern mit wahlfreiem Zugriff (DRAM) **220**. In anderen Ausführungsformen ist die Anzahl an Partitionseinheiten **215** gegebenenfalls nicht gleich der Anzahl an Speichereinrichtungen. Der Fachmann auf dem Gebiet erkennt, dass die DRAM **220** durch andere geeignete Speichereinrichtungen ersetzt werden können und dass sie von allgemein konventioneller Gestaltung sein können. Eine detaillierte Beschreibung wird daher weggelassen. Bilderzeugungsziele, etwa Blockpuffer oder Texturzuordnungen, können in den DRAMs **220** gespeichert sein, wodurch es den Partitionseinheiten **215** möglich ist, Bereiche jedes Bilderzeugungsziels parallel zu beschreiben, um in effizienter Weise die verfügbare Bandbreite des Parallelverarbeitungsspeichers **204** auszunutzen.

[0033] Jeder der GPCs **208** kann Daten verarbeiten, dass Sie in die DRAMs **220** in dem Parallelverarbeitungsspeicher **204** geschrieben werden. Die Kreuzungseinheit **210** ist ausgebildet, die Ausgabe jedes GPC **208** dem Eingang einer Partitionseinheit **215** oder einem weiteren GPC **208** für die weitere Verarbeitung zuzuleiten. Die GPCs **208** kommunizieren mit der Speicherschnittstelle **214** über die Kreuzungseinheit **210**, um diverse externe Speichereinrichtungen auszulesen oder diese zu beschreiben. In einer Ausführungsform hat die Kreuzungseinheit **210** eine Verbindung zu der Speicherschnittstelle **214**, um mit der I/O-Einheit **205** zu kommunizieren, und hat auch eine Verbindung zu dem lokalen Parallelverarbeitungsspeicher **204**, wodurch es den Verarbeitungskernen in den unterschiedlichen GPCs **208** ermöglicht wird, mit dem Systemspeicher **104** oder einem anderen Speicher zu kommunizieren, der nicht lokal für die PPU **202** ist. In der in **Fig.** 2 gezeigten Ausführungsform ist die Kreuzungseinheit **210** direkt mit der I/O-Einheit **205** verbunden. Die Kreuzungseinheit **210** kann virtuelle Kanäle verwenden, um Verkehrsströme zwischen den GPCs **208** und den Partitionseinheiten **215** zu trennen.

[0034] Die GPCs **208** können wiederum so programmiert sein, dass sie Verarbeitungsaufgaben, die eine Fülle von Anwendungen betreffen, ausführen, wozu gehören, ohne einschränkend zu sein, lineare und nicht-lineare Datentransformationen, die Filterung von Video- und/oder Audiodaten, Modellierungsoperationen (beispielsweise die Anwendung physikalischer Gesetze zur Bestimmung der Position, Geschwindigkeit und anderer Attribute von Objekten), Bilderzeugungsoperationen (beispielsweise Programme für die Parkettierungs-Schattierung, Vertex-Schattierung, Geometrie-Schattierung und/oder Pixel Schattierung) usw. Die PPUs **202** können Daten von dem Systemspeicher **104** und/oder dem lokalen Parallelverarbeitungsspeicher **204** in einen internen (Chip internen) Speicher übertragen, die Daten verarbeiten und die Ergebnisdaten zurück in den Systemspeicher **104** und/oder die lokalen Parallelverarbeitungsspeicher **204** schreiben, wo auf derartige Daten von anderen Systemkomponenten zugegriffen werden kann, wozu die CPU **102** oder ein weiteres Parallelverarbeitungssubsystem **112** gehören.

[0035] Eine PPU **202** kann mit einer beliebigen Größe an lokalem Parallelverarbeitungsspeicher **204** versehen sein, wobei auch kein lokaler Speicher mit eingeschlossen ist, und kann den lokalen Speicher und einen Systemspeicher in beliebiger Kombination verwenden. Beispielsweise kann eine PPU **202** ein Grafikprozessor in einer Ausführungsform mit vereinheitlichter Speicherarchitektur (UMA) sein. In derartigen Ausführungsformen wird wenig oder kein spezieller graphischer (Parallelverarbeitungs-) Speicher bereitgestellt, und die PPU **202** verwendet ausschließlich oder nahezu ausschließlich den Systemspeicher. In UMA-Ausführungsformen kann eine PPU **202** in einem Brückenchip oder einem Prozessorchip integriert sein, oder kann als ein diskreter Chip mit einer Hochgeschwindigkeitsverbindung vorgesehen sein (beispielsweise PCI-Expressverbindung), die die PPU **202** mit dem Systemspeicher über einen Brückenchip oder eine andere Kommunikationseinrichtungen verbindet.

[0036] Wie zuvor angegeben ist, kann eine beliebige Anzahl an PPUs **202** in einem Parallelverarbeitungssubsystem **112** enthalten sein. Beispielsweise können mehrere PPUs **202** auf einer einzelnen Zusatzkarte bereitgestellt werden, oder es können mehrere Zusatzkarten mit dem Kommunikationspfad **112** verbunden sein, oder eine oder mehrere PPUs **202** können in einem Brückenchip integriert sein. Die PPUs **202** in einem Multi-PPU-System können identisch

oder unterschiedlich zueinander sein. Beispielsweise können verschiedene PPUs **202** eine verschiedene Anzahl an Verarbeitungskernen, eine verschiedene Größe an lokalem Parallelverarbeitungsspeicher usw. aufweisen. Wenn mehrere PPUs **202** vorhanden sind, können diese PPUs parallel betrieben werden, so dass Daten mit höherem Durchsatz verarbeitet werden, als dies mit einer einzelnen PPU **202** möglich wäre. Systeme, die eine oder mehrere PPUs **202** enthalten, können in einer Vielzahl von Konfigurationen und Formfaktoren eingerichtet werden, wozu Tischrechner, mobile Rechner oder Hand-Personalcomputer, Dienstleister-Rechner, Arbeitsplatzrechner, Spielekonsolen, eingebettete Systeme und dergleichen gehören.

[0037] Es können mehrere Verarbeitungsaufgaben gleichzeitig in den GPCs **208** ausgeführt werden, und eine Verarbeitungsaufgabe kann eine oder mehrere „Kind-"Verarbeitungsaufgaben während der Ausführung erzeugen. Die Aufgaben/Arbeitseinheit **207** empfängt die Aufgaben und disponiert dynamisch die Verarbeitungsaufgaben und die Kind-Verarbeitungsaufgaben zur Ausführung durch die GPCs **208**.

[0038] Fig. 3 ist eine Blockansicht eines Datenstrom-Multiprozessors (SM) **310** in einem GPC **208** aus **Fig.** 2 gemäß einer Ausführungsform der vorliegenden Erfindung. Jeder GPC **208** kann so gestaltet sein, dass er eine große Anzahl an Strängen parallel ausführen kann, wobei der Begriff „Strang" eine Instanz eines speziellen Programms bezeichnet, das mit einem speziellen Satz an Eingangsdaten ausgeführt wird. In einigen Ausführungsformen werden Einzelbefehl-Mehrfach-Daten-(SIMD-) Befehlsausgabetechniken eingesetzt, um eine parallele Ausführung einer großen Anzahl an Strängen zu ermöglichen, ohne dass mehrere unabhängige Befehlseinheiten bereitgestellt werden. In anderen Ausführungsformen werden Einzelbefehl-Multi-Strang-(SIMT-)Techniken angewendet, um die parallele Ausführung einer großen Anzahl an im allgemeinen synchronisierten Strängen zu ermöglichen, wobei eine gemeinsame Befehlseinheit verwendet wird, die ausgebildet ist, Befehle an eine Gruppe aus Verarbeitungseinheiten innerhalb jedes GPCs **208** auszugeben. Anders als ein SIMD-Ausführungsregime, in welchem alle Verarbeitungseinheiten typischerweise identische Befehle ausführen, ermöglicht die SIMT-Ausführung, das unterschiedliche Stränge effizienter divergenten Ausführungspfaden durch ein gegebenes Strangprogramm hindurch folgen. Der Fachmann erkennt, dass ein SIMD-Verarbeitungsregime eine funktionale Untergruppe eines SIMT-Verarbeitungsregimes darstellt.

[0039] Der Betrieb des GPC **208** wird vorteilhafterweise über einen Pipeline-Verwalter (nicht gezeigt) gesteuert, der Verarbeitungsaufgaben an einen oder mehrere Datenstrom-Multiprozessoren (SM) **310** verteilt, wobei jeder SM **310** ausgebildet ist, eine oder mehrere Stranggruppen zu verarbeiten. Jeder SM **310** enthält einen Befehls-L1-Cache-Speicher **370**, der ausgebildet ist, Befehle und Konstanten aus dem Speicher über einen L1.5-Cache-Speicher (nicht gezeigt) in dem GPC **208** zu empfangen. Eine Kettendisponier-und-Befehlseinheit **312** empfängt Befehle und Konstanten aus dem Befehls-L1-Cache-Speicher **370** und steuert eine lokale Registerdatei **304** und Funktionseinheiten des SM **310** entsprechend den Befehlen und den Konstanten. Die Funktionseinheiten des SM **310** enthalten N Exec-(Ausführungs- oder Verarbeitungs-)Einheiten **302** und P Lade-Speichereinheiten (LSU) **303**. Die Funktionseinheiten des SM **310** werden als Pipeline bzw. parallel betrieben werden, wodurch es möglich ist, dass ein neuer Befehl ausgegeben wird, bevor ein vorhergehender Befehl beendet ist, wie dies im Stand der Technik bekannt ist. Es kann eine beliebige Kombination aus Funktionsausführungseinheiten bereitgestellt werden. In einer Ausführungsform unterstützen die Funktionseinheiten eine Fülle von Operationen, wozu Ganzzahl- und Gleitkommaarithmetik (beispielsweise Addition und Multiplikation), Vergleichsoperationen, Bool'sche Operationen (UND, ODER, EXKLUSIV ODER), Bit-Verschiebung und Berechnung diverser algebraischer Funktionen gehören (beispielsweise Ebeneninterpolation, trigonometrische, exponentielle und logarithmische Funktionen usw.); und die gleiche Hardware der Funktionseinheiten kann vorteilhafterweise zum Ausführen unterschiedlicher Operationen verwendet werden.

[0040] Die Reihe von Befehlen, die an einen speziellen GPC **208** ausgegeben wird, bildet einen Strang, wie dies zuvor hierin definiert ist, und die Ansammlung einer gewissen Anzahl an gleichzeitig ausgeführten Strängen in den Parallelverarbeitungseinheiten (nicht gezeigt) innerhalb eines SM **310** wird hierin als eine „Kette bzw. Wölbung" oder „Stranggruppe" bezeichnet. Wie hierin verwendet ist, bezeichnet eine „Stranggruppe" eine Gruppe aus Strängen, die gleichzeitig in dem gleichen Programm mit unterschiedlichen Eingangsdaten ausgeführt werden, wobei ein Strang der Gruppe einer unterschiedlichen Verarbeitungseinheit innerhalb eines SM **310** zugewiesen ist. Eine Stranggruppe kann weniger Stränge als die Anzahl an Verarbeitungseinheiten innerhalb des SM **310** aufweisen, in welchem Falle gewisse Verarbeitungseinheiten während Arbeitszyklen untätig sind, wenn diese Stranggruppe gerade verarbeitet wird. Eine Stranggruppe kann auch mehr Stränge als die Anzahl an Verarbeitungseinheiten innerhalb des SM **310** aufweisen, in welchem Falle die Verarbeitung über aufeinanderfolgende Taktzyklen erfolgt. Da jeder SM **310** bis zu G Stranggruppen gleichzeitig unterstützen kann, ergibt sich, dass ein System in einem GPC **208**, das $\underline{M}$ Datenstrom-Multiprozessoren **310** aufweist, bis zu $\overline{G}\cdot M$ Stranggruppen in dem GPC **208** zu jeder Zeit ausgeführt werden können.

[0041] Ferner können mehrere in Beziehung stehende Stranggruppen (in unterschiedlichen Phasen der Ausführung) gleichzeitig in einen SM **310** aktiv sein. Diese Ansammlung an Stranggruppen wird hierin als ein „kooperatives Strang-Array" („CTA") oder „Strang-Array" bezeichnet. Die Größe eines speziellen CTA ist gleich m·k, wobei k die Anzahl an gleichzeitig ausgeführten Strängen in einer Stranggruppe ist und typischerweise ein ganzzahliges Vielfaches der Anzahl an Parallelverarbeitungseinheiten innerhalb des SM **310** ist, und wobei m die Anzahl an Stranggruppen ist, die gleichzeitig in dem SM **310** aktiv ist. Die Größe eines CTA wird generell von dem Programmierer und der Menge an Hardware-Ressourcen bestimmt, etwa von Speichern und Registern, die für das CTA verfügbar sind.

[0042] In Ausführungsformen der vorliegenden Erfindung ist es wünschenswert, die PPU **202** oder einen oder mehrere andere Prozessoren eines Rechensystems zu verwenden, um Berechnungen für Allgemeinzwecke unter Verwendung von Strang-Arrays auszuführen. Jedem Strang in dem Strang-Array ist eine eindeutige Strangkennung („Strang-ID") zugewiesen, auf die der Strang während der Ausführung des Strangs zugreifen kann. Die Strang-ID, die als ein eindimensionaler oder mehrdimensionaler numerischer Wert definiert sein kann, steuert diverse Aspekte des Verarbeitungsverhaltens des Strangs. Beispielsweise kann eine Strang-ID verwendet werden, um zu bestimmen, welchen Teil eines Eingangsdatensatzes ein Strang zu verarbeiten hat und/oder um zu bestimmen, welchen Teil eines Ausgangsdatensatzes ein Strang zu erzeugen oder zu schreiben hat.

[0043] Eine Sequenz aus Befehlen pro Strang kann mindestens einen Befehl enthalten, der ein kooperatives Verhalten zwischen dem repräsentativen Strang und einem oder mehreren anderen Strängen des Strang-Arrays angibt. Beispielsweise kann die Sequenz aus Befehlen pro Strang enthalten: einen Befehl, um die Ausführung von Operationen für den repräsentativen Strang an einem speziellen Punkt in der Sequenz zu unterbrechen, bis zu einer Zeit, bei der eine oder mehrere der anderen Stränge diesen speziellen Punkt erreicht haben, einen Befehl für den repräsentativen Strang, Daten in einen gemeinsam benutzten Speicher zu speichern, auf den eine oder mehrere der anderen Stränge Zugriff haben, einen Befehl für den repräsentativen Strang, in atomarer Weise bzw. ungeteilter Weise Daten zu lesen und zu aktualisieren, die in einem gemeinsam benutzten Speicher gespeichert sind, auf den einer oder mehrere der anderen Stränge Zugriff haben auf Grundlage ihrer Strang-ID, oder dergleichen. Das CTA-Programm kann ferner einen Befehl enthalten, um eine Adresse in dem gemeinsam benutzten Speicher zu berechnen, aus der Daten auszulesen sind, wobei die Adresse eine Funktion der Strang-ID ist. Durch die Definition geeigneter Funktionen und durch die Bereitstellung von Synchronisiertechniken können Daten in eine gegebene Speicherstelle in einem gemeinsam benutzten Speicher durch einen Strang eines CTA geschrieben werden, und können aus dieser Speicherstelle von einem anderen Strang des gleichen CTA in vorhersagbarer Weise ausgelesen werden. Folglich kann ein beliebiges gewünschtes Schema an gemeinsamer Datennutzung zwischen Strängen unterstützt werden, und ein beliebiger Strang in einem CTA kann Daten mit einem beliebigen anderen Strang in dem gleichen CTA gemeinsam benutzen. Der Grad, falls verhandeln, einer gemeinsamen Datennutzung zwischen Strängen eines CTA ist durch das CTA-Programm bestimmt; es ist somit zu beachten, dass in einer speziellen Anwendung, die CTAs verwendet, die Stränge eines CTA Daten untereinander gemeinsam nutzen können oder auch nicht, wobei dies von dem CTA-Programm bestimmt ist, und die Begriffe „CTA" und „Strang-Array" werden hierin als Synonym verwendet.

[0044] Der SM **310** stellt einen chipinternen (internen) Datenspeicherplatz mit unterschiedlichen Ebenen an Zugänglichkeit bereit. Spezialregister (nicht gezeigt) sind von der LSU **303** lesbar aber nicht beschreibbar und können verwendet werden, um Parameter zu speichern, die die „Position" jedes Strangs definieren. In einer Ausführungsform enthalten die Spezialregister ein Register pro Strang (oder pro Exec-Einheit **302** in dem SM **310**), das eine Strang-ID speichert. Jedes Register einer Strang-ID ist nur von einer entsprechenden Exec-Einheit **302** ansprechbar. Spezialregister können auch zusätzliche Register enthalten, die von allen Strängen lesbar sind, die die gleiche Verarbeitungsaufgabe ausführen, die durch einem Satz an TMD **322** repräsentiert ist (oder von allen LSU **303**), die eine CTA-Kennung, die CTA-Dimensionen, die Dimensionen eines Gitters, zu welchem das CTA gehört (eine Warteschlangenposition, wenn die TMD **322** eine Warteschlangenaufgabe anstelle einer Gitteraufgabe kodieren), und eine Kennung der TMD **322**, denen das CTA zugeordnet ist, speichern.

[0045] Wenn die TMD **322** ein Gitter-TMD sind, bewirkt die Ausführung der TMD **322**, dass eine festgelegte Anzahl an CTAs gestartet und ausgeführt wird, um die festgelegte Menge an Daten zu verarbeiten, die in der Warteschlange gespeichert sind. Die Anzahl an CTAs ist als das Produkt der Gitterbreite, Höhe und Tiefe angegeben. Die festgelegte Menge an Daten kann in den TMD **322** gespeichert sein, oder die TMD **322** können einen Zeiger auf die Daten speichern, die von den CTAs verarbeitet werden. Die TMD **322** können ferner eine Startadresse des Programms enthalten, das von den CTAs ausgeführt wird.

[0046] Wenn die TMD **322** Warteschlange-TMD sind, dann kann eine Warteschlangeneigenschaft der TMD **322** verwendet werden, was bedeutet, dass die zu verarbeitende Datenmenge nicht notwendigerweise festgelegt ist. Warteschlangeneinträge speichern Daten zur Verarbeitung durch die CTAs, die den TMD **322** zugeordnet sind. Die Warteschlangeneinträge können auch eine Kind-Aufgabe repräsentieren, die von weiteren TMD **322** während der Ausführung eines Strangs erzeugt wird, wodurch eine eingebettete Parallelität geschaffen wird. Typischerweise wird die Ausführung des Strangs oder des CTA, das den Strang enthält, unterbrochen, bis die Ausführung der Kind-Aufgabe abgeschlossen ist. Die Warteschlange kann in den TMD **322** oder separat zu den TMD **322** gespeichert werden, in welchem Falle die TMD **322** einen Warteschlangenzeiger auf die Warteschlange speichern. Vorteilhafterweise können die von der Kind-Aufgabe erzeugten Daten in die Warteschlange geschrieben werden, während die TMD **322**, die die Kind-Aufgabe repräsentieren, ausgeführt werden. Die Warteschlange kann als eine Ringwarteschlange realisiert werden, so dass die gesamte Datenmenge nicht auf die Größe der Warteschlange beschränkt ist.

[0047] CTAs, die einem Gitter angehören, haben implizite Parameter für die Gitterbreite, Höhe und Tiefe, die die Position des jeweiligen CTA innerhalb des Gitters angegeben. Spezialregister werden während der Initialisierung in Reaktion auf Befehle beschrieben, die über den Frontbereich **212** von dem Gerätetreiber **103** empfangen werden, und die Register ändern sich während der Ausführung einer Verarbeitungsaufgabe nicht. Der Frontbereich **212** disponiert jede Verarbeitungsaufgabe für die Ausführung. Jedes CTA ist einem speziellen Satz an TMD **322** für die gleichzeitige Ausführung einer oder mehrerer Aufgaben zugeordnet. Ferner kann ein einzelner GPC **208** mehrere Aufgaben gleichzeitig ausführen.

[0048] Ein Parameterspeicher (nicht gezeigt) speichert Laufzeitparameter (Konstanten), die von einem beliebigen Strang innerhalb des gleichen CTA (oder einer LSU **303**) gelesen aber nicht beschrieben werden können. In einer Ausführungsform leitet der Gerätetreiber **103** Parameter dem Parameterspeicher zu, bevor der SM **310** angewiesen wird, die Ausführung einer Aufgabe zu beginnen, die diese Parameter verwendet. Ein beliebiger Strang innerhalb eines CTA (oder eine Exec-Einheit **302** innerhalb des SM **310**) kann auf den globalen Speicher mittels einer Speicherschnittstelle **214** zugreifen. Teile des globalen Speichers können in dem L1-Cache-Speicher **320** liegen.

[0049] Eine lokale Registerdatei **304** wird von jedem Strang als Arbeitsbereich verwendet; jedes Register ist für die ausschließliche Nutzung durch einen einzigen Strang reserviert, und Daten in einem Register der lokalen Registerdatei **304** sind nur für den Strang verfügbar, dem das Register zugewiesen ist. Die lokale Registerdatei **304** kann als eine Registerdatei realisiert sein, die physikalisch oder logisch in P Bahnen unterteilt ist, wobei jede eine gewisse Anzahl an Einträgen (wobei jeder Eintrag beispielsweise ein 32-Bit-Wort speichern kann) aufweist. Jeder der N Exec-Einheiten **302** und jeder der P Lade-Speichereinheiten LSU **303** ist eine einzelne Bahn zugewiesen, und entsprechende Einträge in unterschiedlichen Bahnen können mit Daten für unterschiedliche Stränge, die in dem gleichen Programm ausgeführt werden, angereichert werden, um eine SIMD-Ausführung zu ermöglichen. Unterschiedliche Teile der Bahnen können unterschiedlichen Gruppen der G gleichzeitigen Stranggruppen zugewiesen sein, so dass auf einen gegebenen Eintrag in der lokalen Registerdatei **304** nur von einem speziellen Strang zugegriffen werden kann. In einer Ausführungsform sind gewisse Einträge innerhalb der lokalen Registerdatei **304** für die Speicherung von Strangkennungen reserviert, wodurch eines der Spezialregister realisiert wird. Ferner speichert ein gleichförmiger L1-Cache-Speicher **320** gleichförmige oder konstante Werte für jede Bahn der N Exec-Einheiten **302** und der P Lade-Speichereinheiten LSU **303**.

[0050] Der gemeinsam genutzte Speicher **306** ist für Stränge innerhalb eines einzelnen CTA zugänglich; anders ausgedrückt, jede Stelle in dem gemeinsam benutzten Speicher **306** ist für einen beliebigen Strang innerhalb des gleichen CTA (oder eine Verarbeitungseinheit innerhalb des SM **310**) zugänglich. Der gemeinsam genutzte Speicher **306** kann als eine gemeinsame benutzte Registerdatei oder ein gemeinsam benutzter chipinterner Cache-Speicher mit einer Verbindung realisiert werden, die es einer beliebigen Verarbeitungseinheit ermöglicht, jede Stelle in dem gemeinsam benutzten Speicher zu lesen oder zu beschreiben. In anderen Ausführungsformen kann der gemeinsam benutzte Zustandsraum auf ein Gebiet pro CTA eines chipexternen Speichers abgebildet sein, und kann in dem L1-Cache-Speicher **320** zwischengespeichert sein. Der Parameterspeicher kann als ein spezieller Abschnitt innerhalb der gleichen gemeinsamen benutzten Registerdatei oder als ein gemeinsam benutzter Crash-Speicher realisiert sein, der den gemeinsam benutzten Speicher **306** realisiert, oder kann als eine separate gemeinsame benutzte Registerdatei oder ein chipinterner Cache-Speicher realisiert sein, auf den die LSUs **303** nur einen lesenden Zugriff haben. In einer Ausführungsform wird der Bereich, der den Parameterspeicher bildet, auch verwendet, um die CTA-ID und die Aufgaben-ID sowie die CTA- und Gitter-Abmessungen oder Warteschlangenposition zu speichern, wodurch Teile der Spezialregister realisiert werden. Jede LSU **303** in dem SM **310** ist mit einer vereinheitlichten Adressenzuordnungseinheit **352** verbunden, die eine Adresse, die für Lade- und Speicherbefehle be-

reitgestellt wird, die in einem vereinheitlichten Speicherraum angegeben sind, eine Adresse in jedem einzelnen Speicherraum umwandelt. Folglich kann ein Befehl verwendet werden, um auf den lokalen, den gemeinsam benutzten oder den globalen Speicherraum zuzugreifen, indem eine Adresse in dem vereinheitlichten Speicherraum angegeben wird.

[0051] Der L1-Cache-Speicher **320** in jedem SM **310** kann verwendet werden, um private lokale Daten pro Strang und auch globale Daten pro Anwendung zwischenzuspeichern. In einigen Ausführungsformen können die gemeinsamen benutzten Daten pro CTA in dem L1-Cache-Speicher **320** zwischengespeichert werden. Die LSU **303** sind mit dem gemeinsam benutzten Speicher **306** und dem L1-Cache-Speicher **320** über eine Speicher- und Cache-Verbindung **318** verbunden.

[0052] Es ist ersichtlich, dass die hierin beschriebene Kernarchitektur anschaulicher Natur ist und dass Variationen und Modifizierungen möglich sind. Es kann eine beliebige Anzahl an Verarbeitungseinheiten, beispielsweise SM **310**, in einem GPC **208** enthalten sein. Wie ferner in **Fig.** 2 gezeigt ist, kann eine PPU **202** eine beliebige Anzahl an GPC **208** enthalten, die vorteilhafterweise in funktionaler Hinsicht ähnlich zueinander sind, so dass das Ausführungsverhalten nicht davon abhängt, welcher GPC **208** welche spezielle Verarbeitungsaufgabe empfängt. Ferner kann jeder GPC **208** vorteilhafterweise unabhängig von anderen GPCs **208** unter Verwendung separater und unterscheidbarer Verarbeitungseinheiten, L1-Cache-Speicher arbeiten, um Aufgaben für ein oder mehrere Anwendungsprogramme auszuführen.

[0053] Der Fachmann auf dem Gebiet erkennt, dass die in den **Fig.** 1–**Fig.** 3 beschriebene Architektur in keiner Weise den Schutzbereich der vorliegenden Erfindung einschränkt und dass die hierin gelehrten Techniken in einer beliebigen geeignet ausgebildeten Verarbeitungseinheit realisiert werden können, wozu gehören, ohne Einschränkung, eine oder mehrere CPUs, eine oder mehrere Mehrkern-CPUs, eine oder mehrere PPUs **202**, ein oder mehrere GPCs **208**, eine oder mehrere grafische Verarbeitungseinheiten oder Verarbeitungseinheiten für Allgemeinzwecke, oder dergleichen, ohne von dem Schutzbereich der vorliegenden Erfindung abzuweichen.

Architektur der Grafik-Pipeline

[0054] **Fig.** 4 ist eine Konzeptansicht einer Grafikverarbeitungs-Pipeline **400**, die in eine oder mehrere der PPUs **202** aus **Fig.** 2 zur Realisierung konfiguriert werden können gemäß einer Ausführungsform der vorliegenden Erfindung. Beispielsweise kann einer der SM **310** so gestaltet sein, dass er die Funktionen einer oder mehrerer Vertex-Verarbeitungseinheiten **415**, einer Geometrie-Verarbeitungseinheit **425** und einer Fragment-Verarbeitungseinheit **460** ausführt. Die Funktionen eines Daten-Assemblers **410**, eines Grundelemente-Assemblers **420**, einer Rastereinheit **455** und einer Rasteroperationseinheit **465** können ebenfalls durch andere Verarbeitungseinheiten innerhalb eines GPC **208** und einer entsprechenden Partitionseinheit **215** ausgeführt werden. Andernfalls kann die Grafikverarbeitungs-Pipeline **400** unter Verwendung spezieller Verarbeitungseinheiten für eine oder mehrere Funktionen realisiert werden.

[0055] Der Daten-Assembler **410** sammelt Vertex-Daten für Oberflächen höherer Ordnung, Grundelemente und dergleichen und gibt die Vertex-Daten, wozu die Vertex-Attribute gehören, an die Vertex-Verarbeitungseinheit **415** aus. Die Vertex-Verarbeitungseinheit **415** ist eine programmierbare Ausführungseinheit, die ausgebildet ist, Vertex-Schattierungsprogramme auszuführen, um Beleuchtung und die Transformation von Vertex-Daten anzugeben, wie sie durch die Vertex-Schattierungsprogramme spezifiziert sind. Beispielsweise kann die Vertex-Verarbeitungseinheit **415** programmiert sein, um die Vertex-Daten von einer objektbasierten Koordinatendarstellung (Objektraum) in ein Koordinatensystem mit alternativer Basis zu transformieren, etwa in einen Welt-Raum oder in einen normierten Gerätekoordinaten-(NDC)Raum. Die Vertex-Verarbeitungseinheit **415** kann Daten auslesen, die in dem L1-Cache-Speicher **320**, in dem Parallelverarbeitungsspeicher **204** oder in dem Systemspeicher **104** von dem Daten-Assemblers **410** zur Verwendung bei der Verarbeitung der Vertex-Daten gespeichert sind.

[0056] Der Grundelemente-Assembler **420** empfängt Vertex-Attribute aus der Vertex-Verarbeitungseinheit **415**, wobei gespeicherte Vertex-Attribute nach Bedarf ausgelesen werden, und baut grafische Grundelemente für die Verarbeitung durch die Geometrie-Verarbeitungseinheit **425** auf. Zu grafischen Grundelementen gehören Dreiecke, Liniensegmente, Punkte und dergleichen. Die Geometrie-Verarbeitungseinheit **425** ist eine programmierbare Ausführungseinheit, die ausgebildet ist, Geometrie-Schattierungsprogramme auszuführen, wobei grafische Grundelemente, die von dem Assembler für Grundelemente **420** empfangen werden, transformiert werden, wie dies durch die Geometrie-Schattierungsprogramme angegeben ist. Beispielsweise kann die Geometrie-Verarbeitungseinheit **425** so programmiert sein, dass sie die grafischen Grundelemente in ein oder mehrere neue grafische Grundelemente unterteilt und Parameter berechnet, etwa Koeffizienten für die Ebenengleichung, die verwendet werden, um die neuen grafischen Grundelemente in Raster einzuteilen. Die Geometrie-Verarbeitungseinheit **425** kann auch programmiert werden, um weitere grafische Grundelemente oder ein oder mehrere geometrische Objekte, die aus einem oder meh-

reren grafischen Grundelemente aufgebaut sind, auf der Grundlage der grafischen Grundelemente, die aus dem Grundelemente-Assembler **420** empfangen wurden, zu erzeugen.

[0057] In einigen Ausführungsformen kann die Geometrie-Verarbeitungseinheit **425** auch Elemente dem Geometrie-Strom hinzufügen oder aus diesem löschen. Die Geometrie-Verarbeitungseinheit **425** gibt die Parameter und Vertices bzw. Eckpunkte, die die neuen grafischen Grundelemente angeben, an eine Darstellungsfeld-Skalierungs-, Auswahl-, und Schneideeinheit **450** aus. Die Geometrie-Verarbeitungseinheit **425** kann Daten lesen, die in dem Parallelverarbeitungsspeicher **204** oder in den Systemspeicher **104** gespeichert sind und zur Weiterverarbeitung der Geometriedaten verwendet werden. Die Darstellungsfeld-Skalier-, Auswahl-, und Schneideeinheit **450** führt einen Schneiden, Auswählen und eine Darstellungsfeldskalierung aus und gibt die verarbeiteten grafischen Grundelemente an eine Rastereinheit **455** aus.

[0058] Die Rastereinheit **455** konvertiert in abtastender Weise die neuen grafischen Grundelemente und gibt Fragmente und Abdeckungsdaten an die Fragment-Verarbeitungseinheit **460** aus. Ferner kann die Rastereinheit **455** ausgebildet sein, eine z-Auswahl und andere z-basierte Optimierungen durchzuführen.

[0059] Die Fragment-Verarbeitungseinheit **460** ist eine programmierbare Ausführungseinheit, die ausgebildet ist, Fragment-Schattierungsprogramme auszuführen, wobei Fragmente, die aus der Rastereinheit **455** erhalten werden, transformiert werden in der Art, wie dies durch die Fragment-Schattierungsprogramme angegeben ist. Beispielsweise kann die Fragment-Verarbeitungseinheit **460** so programmiert sein, dass Operationen ausgeführt werden, etwa eine perspektivische Korrektur, eine Texturzuordnungen, eine Schattierung, eine Mischung und dergleichen, um schattierte Fragmente zu erzeugen, die an die Rasteroperationseinheit **465** ausgegeben werden. Die Fragment-Verarbeitungseinheit **460** kann Daten lesen, die in dem Parallelverarbeitungsspeicher **204** oder in den Systemspeicher **104** zur Verwendung bei der Verarbeitung der Fragmentdaten gespeichert sind. Fragmente können schattiert sein auf Ebene von Pixel, Abtastwerten oder entsprechend einer anderen Auflösung, wobei dies von der programmierten Abtastrate abhängig ist.

[0060] Die Rasteroperationseinheit **465** ist eine Verarbeitungseinheit, die Rasteroperationen, etwa Schablonenbildung, z-Test, Mischung und dergleichen ausführt, und die Pixeldaten als verarbeitete Grafikdaten zur Speicherung im Grafikspeicher ausgibt. Die verarbeiteten Grafikdaten können in dem Grafikspeicher, beispielsweise dem Parallelverarbeitungsspeicher **204** und/oder in dem Systemspeicher **104** zur Anzeige auf dem Anzeigegerät **110** oder für die Weiterverarbeitung durch die CPU **102** oder das Parallelverarbeitungssubsystem **112** gespeichert werden. In einigen Ausführungsformen der vorliegenden Erfindung ist die Rasteroperationseinheit **465** ausgebildet, z- oder Farbdaten zu komprimieren, die in den Speicher geschrieben werden, und um z- oder Farbdaten zu dekomprimieren, die aus dem Speicher ausgewiesen werden.

Verarbeitung von Vertices, die in grafischen Grundelemente gemeinsam vorhanden sind

[0061] **Fig.** 5 ist eine Konzeptansicht, die ein Subsystem **500** zeigt, das Geometrie-Verarbeitungseinheiten **550-0** bis **550-N** gemäß einer Ausführungsform der vorliegenden Erfindung aufweist. Wie gezeigt, ist jede der Geometrie-Verarbeitungseinheiten **550-0** bis **550-N** mit einer Datenstromausgabesynchronisier-(SSYNC)Einheit **514**, einem globalen Index-Puffer **516** und einen globalen Vertex-Puffer **518** verbunden. Jede Geometrie-Verarbeitungseinheit **550** enthält eine Geometrie-Schattierungseinheit **504**, einen Puffer **506**, der einen lokalen Index-Puffer **508** und einen lokalen Vertex-Puffer **510** enthält, und eine Datenstromausgabeeinheit **512**.

[0062] Wie gezeigt ist, enthält die Geometrie-Schattierungseinheit **504-0** die Geometrie-Schattierungseinheit **504-0**, den Puffer **506-0**, der den lokalen Index-Puffer **508-0** und den lokalen Vertex-Puffer **510-0** enthält, und die Datenstromausgabeeinheit **512-0**. In ähnlicher Weise enthält die Geometrie-Schattierungseinheit **504-1** die Geometrie-Schattierungseinheit **504-1**, den Puffer **506-1**, der den lokalen Index-Puffer **508-1** und den lokalen Vertex-Puffer **510-1** enthält, und die Datenstromausgabeeinheit **512-1**, und die Geometrie-Schattierungseinheit **504** umfasst die Geometrie-Schattierungseinheit **504-N**, den Puffer **506-N**, der den lokalen Index-Puffer **508-N** und den lokalen Vertex-Puffer **510-0** enthält, und die Datenstromausgabeeinheit **512-N**. In der folgenden Beschreibung sind mehrere Instanzen gleicher Objekte mit Bezugszeichen bezeichnet, die das Objekt angeben, und Bezugszahlen in Klammern geben die Instanz an, falls dies erforderlich ist.

[0063] Die Geometrie-Verarbeitungseinheit **550** ist ausgebildet, grafische Grundelemente oder geometrische Objekte **502** zu verarbeiten und ein oder mehrere grafische Grundelemente oder ein oder mehrere grafische Objekte, die aus einem oder mehreren grafischen Grundelemente aufgebaut sind, zu erzeugen. Die graphische Verarbeitungseinheit **550** ist ferner ausgebildet, Vertex-Daten und Index-Information, die mit dem erzeugten grafischen Grundelementen oder geometrischen Objekten verknüpft sind, dem globalen Vertex-Puffer **518** und entsprechend dem globalen Index-Puffer **516** als Datenstrom zuzuleiten. Die SSYNC-Einheit **514** ist ausgebildet, die Zuleitung als

Datenstrom dieser Daten über die unterschiedlichen Geometrie-Verarbeitungseinheiten **550** zu koordinieren.

[0064] Die Geometrie-Verarbeitungseinheit **550** kann durch den SM **310**, der in **Fig.** 3 gezeigt ist, realisiert werden und kann eine Verarbeitungsstufe in der Grafikverarbeitungs-Pipeline **400**, die in **Fig.** 4 gezeigt ist, repräsentieren. In einer Ausführungsform ist die Geometrie-Verarbeitungseinheit **550** ähnlich zu der in **Fig.** 4 gezeigten Geometrie-Verarbeitungseinheit **425**. Die Geometrie-Schattierungseinheit **504** in der Geometrie-Verarbeitungseinheit **550** ist ausgebildet, grafische Grundelemente oder geometrische Objekte **502** aus einer vorgeordneten Verarbeitungseinheit zu empfangen, etwa dem Grundelemente-Assembler **420**, der in **Fig.** 4 gezeigt ist. Die grafischen Grundelemente können beispielsweise Dreiecke, Rechtecke, Liniensegmente, Punkte oder andere Arten von grafischen Grundelementen repräsentieren. Geometrische Objekte können grafische Konstrukte auf höherer Ebene repräsentieren, die aus einem einzelnen grafischen Grundelement aufgebaut sein können, oder die in eine Ansammlung grafischer Grundelemente zerlegt werden können, wobei diese Ansammlung ein streifenartiges, fächerartiges oder gitterartiges geometrisches Objekt repräsentieren kann. In diversen Ausführungsformen können die grafischen Grundelemente oder geometrischen Objekte **502** einen Teil einer Grafikszene repräsentieren oder können einem speziellen Gebiet eines Anzeigebildschirms entsprechen, der mit der Geometrie-Verarbeitungseinheit **550** verbunden ist.

[0065] Wenn die Geometrie-Verarbeitungseinheit **550** ein grafisches Grundelement oder ein geometrisches Objekt **502** empfängt, ist die Geometrie-Schattierungseinheit **504** ausgebildet, eine oder mehrere Geometrie-Schattierungsoperationen an Eckpunkten und anderer Informationen, die mit diesem grafischen Grundelement oder geometrischen Objekt **502** verknüpft ist, auszuführen. Die Vertices und die andere Information, die mit einem gegebenen grafischen Grundelement oder geometrischen Objekt **502** verknüpft sind, können beispielsweise die Ecken eines Dreiecks oder eines anderen Polygons repräsentieren. Diese Vertices bzw. Eckpunkte und andere Informationen können auch Vertex-Attribute enthalten, die mit dem grafischen Grundelement oder geometrischen Objekt **502** verknüpft sind, und können auch andere Arten von Vertex-Daten beinhalten. Die Geometrie-Schattierungseinheit **504** ist ausgebildet, die Vertex-Daten, die zu dem einen oder den mehreren grafischen Grundelementen oder dem einen oder den mehreren geometrischen Objekten gehören, die von der Geometrie-Schattierungseinheit **504** erzeugt wurden, in den lokalen Vertex-Puffer **510** zu speichern.

[0066] In Situationen, in denen die Geometrie-Schattierungseinheit **504** unterschiedliche grafische Grundelemente oder geometrische Objekte erzeugt, die einen Eckpunkt gemeinsam haben, etwa, wenn ein geometrisches Objekt **502** in kleinere grafische Grundelemente unterteilt wird, ist die Geometrie-Schattierungseinheit **504** ausgebildet, den gemeinsamen Eckpunkt und die zugehörigen Vertex-Daten nur einmal in dem lokalen Vertex-Puffer **510** zu speichern. Mit dieser Vorgehensweise vermeidet die Geometrie-Schattierungseinheit **504** vorteilhafterweise die Zwischenspeicherung von redundanten Kopien von Eckpunkten und den zugehörigen Vertex-Daten. Beispielsweise kann die Geometrie-Schattierungseinheit **504** ein einzelnes grafisches Grundelement **502** empfangen und dann eine Ansammlung grafischer Grundelemente auf der Grundlage des grafischen Grundelements **502** erzeugen, die gemeinsame Eckpunkte besitzen. Die Ansammlung der erzeugten grafischen Grundelemente könnte ein „Streifen-", ein „Fächer-" oder ein „Gitter-"Konstrukt sein. In dieser Situation speichert die Geometrie-Schattierungseinheit **504** jeden einzigartigen Eckpunkt nur einmal in dem lokalen Vertex-Puffer **510**.

[0067] Die Geometrie-Schattierungseinheit **504** ist ferner ausgebildet, Verbindungsinformation für ein grafisches Grundelement **502** beizubehalten, indem eine Gruppe aus Indizes für den lokalen Vertex-Puffer **510** erzeugt wird, die die Vertices angibt, die mit dem grafischen Grundelementen oder grafischen Objekten, die von der Geometrie-Schattierungseinheit **504** erzeugt werden, verknüpft sind. In einer Ausführungsform kann die Geometrie-Schattierungseinheit **504** ermitteln, dass ein gegebener Vertex bereits in dem lokalen Vertex-Puffer **510** liegt, und kann dann die Gruppe aus Indizes so erzeugen, dass sie einen Index enthält, der den gegebenen Vertex bezeichnet, das heißt ohne dass der Vertex in den lokalen Vertex-Puffer **510** erneut gespeichert wird. Die Geometrie-Schattierungseinheit **504** ist ausgebildet, die Gruppe an Indizes für jedes erzeugte grafische Grundelement oder geometrische Objekt in einem Eintrag in dem lokalen Index-Puffer **508** zu speichern. Generell kann jeder Index in einem Eintrag in dem lokalen Index-Puffer **508** einem unterschiedlichen Vertex entsprechen, der in dem lokalen Vertex-Puffer **510** gespeichert ist, und eine Gruppe an Indizes, die in einem Eintrag in dem lokalen Index-Puffer **508** gespeichert sind, kann einem speziellen grafischen Grundelement, einem speziellen geometrischen Objekt oder einer beliebigen Ansammlung an Vertices entsprechen, die ein Objekt in der Grafikszene repräsentieren. Beispielsweise kann die Geometrie-Schattierungseinheit **504** auch Gruppen an Indizes entsprechen, die geometrische Objekte auf höherer Ebene darstellen, etwa als „große" Polygone, die in mehrere miteinander verbundene grafische Grundelemente aufgebrochen werden können, wozu strei-

fenartige, fächerartige und gitterartige Objekte gehören.

[0068] Eine Gruppe an Indizes in einem lokalen Index-Puffer **508** kann unterschiedliche Vertices in dem lokalen Vertex-Puffer **510** direkt bezeichnen, d. h. durch Angabe diverser Adressen innerhalb des lokalen Vertex-Puffers **510**. Alternativ kann die Gruppe an Indizes auch unterschiedliche Vertices kennzeichnen, indem ein lokaler Offset innerhalb des lokalen Vertex-Puffers **510** oder ein lokaler Index innerhalb des lokalen Vertex-Puffers **510** angegeben wird.

[0069] Zu diversen Zeiten kann die Datenstromausgabeeinheit **512**, die mit dem Puffer **506** verbunden ist, die in dem lokalen Vertex-Puffer **510** gespeicherten Vertices und die in dem lokalen Index-Puffer **508** gespeicherten Indizes an den globalen Vertex-Puffer **518** und entsprechend an den globalen Index-Puffer **516** als Datenstrom übertragen. Bevor dies bewerkstelligt wird, ist jedoch die Datenstromausgabeeinheit **512** ausgebildet, zunächst die Anzahl an Vertices zu ermitteln, die in dem lokalen Vertex-Puffer **510** gespeichert ist, sowie die Anzahl an Indizes zu ermitteln, die in dem lokalen Index-Puffer **508** gespeichert ist. Die Datenstromausgabeeinheit **512** übermittelt dann diese Zahlen an die SSYNC-Einheit **514**.

[0070] Die SSYNC-Einheit **514** antwortet der Datenstromausgabeeinheit **512** mit einer Basisadresse in dem globalen Vertex-Puffer **518** und einer Basisadresse in dem globalen Index-Puffer **516**. Die Basisadresse für den globalen Vertex-Puffer **518** repräsentiert eine Position in diesem Puffer, an welcher die Datenstromausgabeeinheit **512** die Anzahl an Vertices, die der SSYNC-Einheit **514** von der Datenstromausgabeeinheit **512** übermittelt wurde, sicher speichern kann. In ähnlicher Weise repräsentiert die Basisadresse für den globalen Index-Puffer **516** eine Position in diesem Puffer, an welcher die Datenstromausgabeeinheit **512** die Anzahl an Indizes, die der SSYNC-Einheit **514** von der Datenstromausgabeeinheit **512** übermittelt wurden, sicher speichern kann.

[0071] Die SSYNC-Einheit **514** ist ausgebildet, diese Basisadressen unter Anwendung einer Technik zu erzeugen, die nachfolgend detaillierter beschrieben ist. Bei Empfang der Basisadresse für den globalen Vertex-Puffer kann dann die Datenstromausgabeeinheit **512** die Vertices innerhalb des lokalen Vertex-Puffers **510** beginnend bei dieser Basisadresse in den globalen Vertex-Puffer **518** kopieren. Ferner kann bei Empfang der Basisadresse für den globalen Index-Puffer die Datenstromausgabeeinheit **512** dann die Indizes in den lokalen Index-Puffer **508** beginnend bei dieser Basisadresse in den globalen Index-Puffer **516** kopieren.

[0072] Wenn Indizes aus dem lokalen Index-Puffer **508** in den globalen Index-Puffer **516** kopiert werden, dann ist die Datenstromausgabeeinheit **512** ausgebildet, jene Indizes zu aktualisieren, die die neuen Positionen der angegebenen Vertices in dem globalen Vertex-Puffer **518** wiedergeben. In einer Ausführungsform inkrementiert die Datenstromausgabeeinheit **512** den Index um einen Wert, der gleich der Basisadresse in dem globalen Vertex-Puffer **518** ist.

[0073] Bei der vorhergehenden Vorgehensweise ist jede der Geometrie-Verarbeitungseinheiten **550-0** bis **550-N** ausgebildet, grafische Grundelemente oder geometrische Objekte parallel zu verarbeiten und dann Ergebnisse dieser Verarbeitung sowie die zugehörigen Indizes in den lokalen Puffern zu speichern. Die lokal gespeicherten Vertex-Daten und Index-Daten werden dann als Datenstrom an die globalen Puffer geleitet.

[0074] Wie zuvor angegeben ist, ist die SSYNC-Einheit **514** ausgebildet, den Datenstrom von Vertices und Indizes zu dem globalen Vertex-Puffer **518** und zu dem globalen Index-Puffer **516** für die unterschiedlichen Geometrie-Verarbeitungseinheiten **550** zu koordinieren. In der Praxis ist die SSYNC-Einheit **514** ausgebildet, jede der Datenstromausgabeeinheiten **512-0** bis **512-N** entsprechend einer Reihenfolge zu bedienen. Wenn dies bewerkstelligt wird, übermittelt die SSYNC-Einheit **514** eine Basisadresse für den globalen Vertex-Puffer **518** und eine Basisadresse für den globalen Index-Puffer **516** an jede Datenstromausgabeeinheit **512-0** bis **512-N** entsprechend dieser Reihenfolge. In einer Ausführungsform ist die Reihenfolge die Reihenfolge einer Anwendungsprogrammierschnittstelle (API). In einer weiteren Ausführungsform ist die Reihenfolge durch eine Softwareanwendung definiert, die in der Geometrie-Verarbeitungseinheit **550** ausgeführt wird, und ein Programmierer dieser Softwareanwendung legt die Reihenfolge fest.

[0075] Die SSYNC-Einheit **514** ist ausgebildet, jeder Datenstromausgabeeinheit **512** eine andere Basisadresse für den globalen Index-Puffer **516** und eine andere Basisadresse für den lokalen Indexpuffer **518** zuzuleiten, wenn diese Datenstromausgabeeinheiten **512** der Reihe nach bedient werden. Folglich ist jede unterschiedliche Datenstromausgabeeinheit **512** in der Lage, Vertices und Indizes entsprechend in einen anderen Bereich des globalen Vertex-Puffers **518** und des globalen Index-Puffers **516** zu schreiben. In einer Ausführungsform ist jede Datenstromausgabeeinheit **512** in der Lage, Vertices und Indizes entsprechend in den globalen Vertex-Puffer **518** und den globalen Index-Puffer **516** parallel mit anderen Datenstromausgabeeinheit **512** zu schreiben, die Vertices und Indizes in diese Puffer schreiben.

[0076] Die SSYNC-Einheit **514** ermittelt eine Basisadresse für den globalen Vertex-Puffer **518** für eine gegebene Datenstromausgabeeinheit **512** in der

Reihenfolge auf der Grundlage der Anzahl an Vertices, die in den globalen Vertex-Puffer **518** von einer vorhergehenden Datenstromausgabeeinheit **512** in der Reihenfolge geschrieben wurde. Insbesondere wird in der SSYNC-Einheit **514** eine „aktuelle" Basisadresse für den globalen Vertex-Puffer **518** beibehalten, die eine Position in dem globalen Vertex-Puffer **518** angibt, an der Vertices sicher gespeichert werden können. Bei Erhalt von Daten, die die Anzahl an Vertices angeben, die von einer speziellen Datenstromausgabeeinheit **512** in den globalen Vertex-Puffer **518** zu schreiben sind, sendet die SSYNC-Einheit **514** die „aktuelle" Basisadresse für den globalen Vertex-Puffer **518** zu dieser Datenstromausgabeeinheit **512** zur Verwendung, wenn Vertices geschrieben werden. Die SSYNC-Einheit **514** aktualisiert dann die „aktuelle" Basisadresse für den globalen Vertex-Puffer **518** auf der Grundlage dieser Anzahl an Vertices und auf der Grundlage der Größe dieser Vertices. Danach repräsentiert die aktualisierte Basisadresse für den globalen Vertex-Puffer **518** eine Position innerhalb dieses Puffers, an der eine nachfolgende Datenstromausgabeeinheit **512** in der Reihenfolge der Datenstromausgabeeinheiten **512** sicher Vertex-Daten speichern kann.

[0077] Die SSYNC-Einheit **514** ermittelt ferner eine Basisadresse für den globalen Index-Puffer **516** für eine gegebene Datenstromausgabeeinheit **512** in der Reihenfolge auf der Grundlage der Anzahl an Indizes, die in den globalen Index-Puffer **516** von einer vorhergehenden Datenstromausgabeeinheit **512** in der Reihenfolge geschrieben wurden. Insbesondere wird von der SSYNC-Einheit **514** eine „aktuelle" Basisadresse für den globalen Index-Puffer **516** beibehalten, die eine Position in dem globalen Index-Puffer **516** angibt, an der Indizes sicher gespeichert werden können. Bei Empfang von Daten, die die Anzahl an Indizes angeben, die von einer speziellen Datenstromausgabeeinheit **512** in den globalen Index-Puffer **516** zu schreiben ist, sendet die SSYNC-Einheit **514** die „aktuelle" Basisadresse für den globalen Index-Puffer **516** zu dieser Datenstromausgabeeinheit **512** zur Verwendung beim Schreiben von Indizes. Die SSYNC-Einheit **514** aktualisiert dann die „aktuelle" Basisadresse in dem globalen Index-Puffer **516** auf der Grundlage dieser Anzahl an Indizes und auf der Grundlage der Größe dieser Indizes. Danach repräsentiert die aktualisierte Basisadresse für den globalen Index-Puffer **516** eine Position innerhalb dieses Puffers, an der eine nachfolgende Datenstromausgabeeinheit **512** in der Reihenfolge aus Datenstromausgabeeinheiten **512** Index-Daten sicher speichern kann.

[0078] Durch die Einrichtung der zuvor beschriebenen Vorgehensweise ist die SSYNC-Einheit **514** ausgebildet, eine „aktuelle" Basisadresse für den globalen Vertex-Puffer **518** und eine „aktuelle" Basisadresse für den globalen Index-Puffer **516** beizubehalten, die einer gegebenen Datenstromausgabeeinheit **512** zugeleitet werden können. Die SSYNC-Einheit **514** ist ferner ausgebildet, um dann diese „aktuellen" Basisadressen zu aktualisieren, um eine nachfolgende Datenstromausgabeeinheit **512** zu handhaben, die versucht, Vertices und Indizes in den globalen Vertex-Puffer **518** und den globalen Index-Puffer **516** als Datenstrom zu senden.

[0079] Diverse Vorgehensweisen zur Realisierung der hierin beschriebenen Funktionen sind nachfolgend detaillierter in Verbindung mit den **Fig.** 6–**Fig.** 8 mit Bezug zu unterschiedlichen Flussdiagrammen angegeben. Die hierin beschriebenen Funktionen sind auch nachfolgend beispielhaft in Verbindung mit den **Fig.** 9–**Fig.** 10 dargestellt.

[0080] **Fig.** 6 ist ein Flussdiagramm von Verfahrensschritten zur Speicherung von Vertex-Daten und Index-Daten entsprechend in einen lokalen Vertex-Puffer **510** und einen lokalen Index-Puffer **508** gemäß einer Ausführungsform der vorliegenden Erfindung. Obwohl die Verfahrensschritte in Verbindung mit den Systemen der **Fig.** 1–**Fig.** 3 geschrieben sind, erkennt der Fachmann, dass ein beliebiges System, das zur Ausführung der Verfahrensschritte in beliebiger Reihenfolge geeignet ist, innerhalb des Schutzbereichs der Erfindung liegt.

[0081] Wie gezeigt, beginnt ein Verfahren **600** in einem Schritt **602**, in welchem die Geometrie-Verarbeitungseinheit **550** ein grafisches Grundelement oder geometrisches Objekt **502** empfängt. Im Schritt **604** führt die Geometrie-Schattierungseinheit **504** in der Geometrie-Verarbeitungseinheit **550** eine oder mehrere Geometrie-Schattierungsoperationen an dem grafischen Grundelement oder geometrischen Objekt **502** aus. Dabei kann die Geometrie-Schattierungseinheit **504** programmiert sein, weitere grafische Grundelemente oder ein oder mehrere geometrische Objekte zu erzeugen, die aus einem oder mehreren grafischen Grundelementen auf der Basis des empfangenen grafischen Grundelements oder geometrischen Objekts **502** erzeugt werden. In einer Ausführungsform kann beispielsweise die Geometrie-Verarbeitungseinheit **550** ein geometrisches Objekt empfangen und dann mehrere grafische Grundelemente erzeugen, indem das geometrische Objekt in eine Ansammlung von miteinander verbundenen grafischen Grundelementen unterteilt wird, die einen oder mehrere Vertices gemeinsam haben.

[0082] Im Schritt **606** speichert die Geometrie-Schattierungseinheit **504** die Vertices, die mit den von der Geometrie-Schattierungseinheit **504** erzeugten grafischen Grundelementen oder geometrischen Objekten verknüpft sind, in dem lokalen Vertex-Puffer **510**. In Situationen, in denen ein spezielles erzeugtes grafisches Grundelement oder geometrisches Objekt mit einem Vertex verknüpft ist, der bereits in dem lo-

kalen Vertex-Puffer **510** gespeichert ist (beispielsweise dieser Vertex ist auch in einem weiteren erzeugten grafischen Grundelement oder geometrischen Objekt enthalten), kann die Geometrie-Schattierungseinheit **504** den Schritt **606** in Bezug auf diesen Vertex überspringen. Mit dieser Vorgehensweise vermeidet die Geometrie-Schattierungseinheit **504** vorteilhafterweise die Speicherung redundanter Kopien der erzeugten Vertices und ihrer zugehörige Vertex-Daten.

[0083] Im Schritt **608** speichert die Geometrie-Schattierungseinheit **504** Indizes in dem lokalen Index-Puffer **508**, die Vertices in dem lokalen Vertex-Puffer **510** kennzeichnen. Die Indizes in dem lokalen Index-Puffer **508** können unterschiedliche Vertices in dem lokalen Vertex-Puffer **510** direkt angeben, d. h. durch Angabe diverser Adressen in dem lokalen Vertex-Puffer **510**. Alternativ können die Indizes auch die unterschiedlichen Vertices angeben, indem ein lokaler Offset in dem lokalen Vertex-Puffer **510** oder ein lokaler Index in dem lokalen Vertex-Puffer **510** angegeben werden. Im Allgemeinen repräsentieren die in dem lokalen Index-Puffer **508** im Schritt **608** gespeicherten Indizes die Vertices, die den grafischen Grundelementen oder geometrischen Objekte entsprechen, die im Schritt **604** erzeugt wurden. Dann endet das Verfahren **600**.

[0084] Durch die Realisierung des zuvor beschriebenen Ansatzes können Vertices, die von einer Geometrie-Verarbeitungseinheit **550** erzeugt werden, die mit erzeugten Grundelementen oder geometrischen Objekten verknüpft ist, lokal zwischengespeichert und lokal indiziert werden, wodurch eine Situation verhindert wird, in der mehrere Kopien von Vertices und zugehörigen Vertex-Daten wiederholt gespeichert werden. Wenn ferner ein System mehrere Geometrie-Verarbeitungseinheiten **550** enthält, kann jede dieser Geometrie-Verarbeitungseinheiten **550** grafische Grundelemente oder geometrische Objekte erzeugen und dann die Vertices und Indizes, die mit diesen grafischen Grundelementen oder geometrischen Objekten verknüpft sind, lokal gleichzeitig mit anderen Geometrie-Verarbeitungseinheiten **550** speichern. Der Fachmann erkennt, dass das Verfahren **600** auch auf die Verarbeitung eines geometrischen Objekts oder anderer grafischer Konstrukte auf höherer Ebene, die eine Ansammlung von Vertices enthalten, angewendet werden kann. Beispielsweise kann das Verfahren **600** angewendet werden, um Vertices und zugehörige Indizes für ein Polygon zu speichern, wobei ein derartiges Polygon in eine Ansammlung von miteinander verbundenen grafischen Grundelemente aufgebrochen werden kann, die einen oder mehrere Vertices gemeinsam haben.

[0085] Jede der Geometrie-Verarbeitungseinheiten **550** ist ausgebildet, mit der SSYNC-Einheit **514** zu kommunizieren, um die Zuleitung von Vertices und Indizes zu dem globalen Vertex-Puffer **518** und dem globalen Index-Puffer **516** als Datenstrom zu koordinieren, wie dies in Verbindung mit **Fig.** 7 detaillierter erläutert ist.

[0086] **Fig.** 7 ist ein Flussdiagramm von Verfahrensschritten zur Zuleitung von Vertices und Indizes zu mehreren globalen Puffern in Form eines Datenstroms gemäß einer Ausführungsform der vorliegenden Erfindung. Obwohl die Verfahrensschritte in Verbindung mit den Systemen der **Fig.** 1–**Fig.** 3 beschrieben sind, erkennt der Fachmann, dass ein beliebiges System, das zur Ausführung der Verfahrensschritte in beliebiger Reihenfolge geeignet ist, innerhalb des Schutzbereichs der Erfindung liegt.

[0087] Wie gezeigt, beginnt ein Verfahren **700** im Schritt **700**, in welchem die Datenstromausgabeeinheit **512** in der Geometrie-Verarbeitungseinheit **550** die Anzahl an Vertices in den lokalen Vertex-Puffer **510** und die Anzahl an Indizes in den lokalen Index-Puffer **508** ermittelt. Die Vertices und Indizes in dem lokalen Vertex-Puffer **510** und dem lokalen Index-Puffer **508** können in diese Puffer eingeführt werden, indem das Verfahren **600** realisiert wird, das in Verbindung mit **Fig.** 6 erläutert ist.

[0088] Im Schritt **704** übermittelt die Datenstromausgabeeinheit **512** die Anzahl an Vertices in dem lokalen Vertex-Puffer **510** und die Anzahl an Indizes in dem lokalen Index-Puffer **508** an die SSYNC-Einheit **514**. Im Schritt **706** empfängt die Datenstromausgabeeinheit **512** eine Basisadresse für den globalen Vertex-Puffer **518** und eine Basisadresse für den globalen Index-Puffer **516** von der SSYNC-Einheit **514**. Die Basisadresse in dem globalen Vertex-Puffer **518** repräsentiert eine Position in dem Puffer, an welcher die Datenstromausgabeeinheit **512** die Anzahl an Vertices sicher speichern kann, die der SSYNC-Einheit **514** von der Datenstromausgabeeinheit **512** übermittelt wurde. In ähnlicher Weise repräsentiert die Basisadresse in dem globalen Index-Puffer **516** eine Position innerhalb dieses Puffers, an welcher die Datenstromausgabeeinheit **512** die Anzahl an Indizes sicher speichern kann, die von der Datenstromausgabeeinheit **512** an die SSYNC-Einheit **514** übermittelt wurde. Die SSYNC-Einheit **514** ist ausgebildet, diese Basisadressen zu erzeugen, indem die zuvor in Verbindung mit **Fig.** 5 beschriebene Technik realisiert wird, wie dies auch nachfolgend in Verbindung mit **Fig.** 8 beschrieben ist.

[0089] Im Schritt **708** leitet die Datenstromausgabeeinheit **512** Vertices aus dem lokalen Vertex-Puffer **510** zu dem globalen Vertex-Puffer **518** beginnend bei der Basisadresse in dem globalen Vertex-Puffer **518**, die von der SSYNC-Einheit **514** bereitgestellt wurde, als Datenstrom zu. Im Schritt **712** leitet die Datenstromausgabeeinheit **512** Indizes aus dem lokalen Index-Puffer **508** an den globalen Index-Puffer **516** beginnend bei der Basisadresse in dem globa-

len Index-Puffer **516** als Datenstrom zu, die von der SSYNC-Einheit **514** bereitgestellt wird. In diesem Falle ist die Datenstromausgabeeinheit **512** ausgebildet, jene Indizes zu aktualisieren, die die neuen Positionen der angesprochenen Vertices innerhalb des globalen Vertex-Puffers **518** wiedergeben. In einer Ausführungsform inkrementiert die Datenstromausgabeeinheit **512** den Index um einen Wert, der gleich der Basisadresse in dem globalen Vertex-Puffer **518** ist, die von der SSYNC-Einheit **514** im Schritt **706** bereitgestellt wird. Das Verfahren endet dann.

[0090] Durch Realisierung des zuvor beschriebenen Ansatzes ist jede der Geometrie-Verarbeitungseinheiten **550-0** bis **550-N** ausgebildet, lokal zwischengespeicherte Vertices und Indizes entsprechend dem globalen Vertex-Puffer **518** und den globalen Index-Puffer **516** als Datenstrom zuzuleiten. Ferner kann jede derartige Geometrie-Verarbeitungseinheit **550** Vertices und Indizes entsprechend dem globalen Vertex-Puffer **518** und dem globalen Index-Puffer **516** als Datenstrom in paralleler Weise in Bezug zu anderen Geometrie-Verarbeitungseinheiten **550** zuführen. Eine Technik, die von der SSYNC-Einheit **514** zur Bereitstellung der Basisadressen in diesen Puffern für die Geometrie-Verarbeitungseinheiten **550** realisiert werden kann, ist nachfolgend in Verbindung mit **Fig.** 8 beschrieben.

[0091] **Fig.** 8 ist ein Flussdiagramm von Verfahrensschritten zur Auffüllung mehrerer globaler Puffer gemäß einer Ausführungsform der vorliegenden Erfindung. Obwohl die Verfahrensschritte in Verbindung mit den Systemen der **Fig.** 1–**Fig.** 3 beschrieben sind, erkennt der Fachmann, dass ein beliebiges System, das zur Ausführung der Verfahrensschritte in beliebiger Reihenfolge geeignet ist, innerhalb des Schutzbereichs der Erfindung liegt.

[0092] Wie gezeigt, beginnt ein Verfahren **800** in einem Schritt **802**, in welchem die SSYNC-Einheit **514** Daten aus der Datenstromausgabeeinheit **512** in der Geometrie-Verarbeitungseinheit **550** empfängt, die die Anzahl an Vertices spezifizieren, die in dem lokalen Vertex-Puffer **510** gespeichert sind, und die die Anzahl an Indizes angeben, die in dem lokalen Index-Puffer **508** gespeichert sind. Im Schritt **804** sendet die SSYNC-Einheit **514** die aktuelle Basisadresse in dem globalen Vertex-Puffer **518** und die aktuelle Basisadresse in dem globalen Index-Puffer **560** an die Datenstromausgabeeinheit **514**. Die Basisadresse in dem globalen Vertex-Puffer **518** repräsentiert eine Position in diesem Puffer, an der die Datenstromausgabeeinheit **512** die Anzahl an Vertices sicher speichern kann, die von der Datenstromausgabeeinheit **512** an die SSYNC-Einheit **514** übermittelt wurde. In ähnlicher Weise repräsentiert die Basisadresse in dem globalen Index-Puffer **516** eine Position in diesem Puffer, an welcher die Datenstromausgabeeinheit **512** die Anzahl an Indizes sicher speichern kann, die der SSYNC-Einheit **514** von der Datenstromausgabeeinheit **512** übermittelt wurde.

[0093] Im Schritt **806** aktualisiert die SSYNC-Einheit **514** die aktuelle Basisadresse in dem globalen Vertex-Puffer **518** auf der Grundlage der Anzahl an Vertices, die von der Datenstromausgabeeinheit **514** angegeben wird. Die SSYNC-Einheit **514** kann auch die aktuelle Basisadresse in dem globalen Vertex-Puffer **518** auf der Grundlage der Größe dieser Vertices aktualisieren. Im Schritt **808** aktualisiert die SSYNC-Einheit **514** die aktuelle Basisadresse in dem globalen Index-Puffer **516** auf der Grundlage der Anzahl an Indizes, die von der Datenstromausgabeeinheit **514** angegeben wird. Die SSYNC-Einheit **514** kann auch die aktuelle Basisadresse in dem globalen Index-Puffer **516** auf der Grundlage der Größe dieser Indizes aktualisieren. Das Verfahren **800** endet dann.

[0094] Durch die Realisierung des zuvor beschriebenen Ansatzes ist die SSYNC-Einheit **514** in der Lage, Basisadressen für den globalen Vertex-Puffer **518** und den globalen Index-Puffer **516** zu bewahren, die Adressen innerhalb dieser Puffer repräsentieren, an denen Daten sicher gespeichert werden können. Wenn die Geometrie-Verarbeitungseinheiten **550** der Reihe nach bedient werden, ist somit die SSYNC-Einheit **514** in der Lage, unterschiedliche Basisadressen in diesem Puffern jeder Geometrie-Verarbeitungseinheit **550** in der Reihenfolge zuzuleiten.

[0095] Die diversen Techniken, die zuvor in Verbindung mit den **Fig.** 5–**Fig.** 8 beschrieben wurden, werden nunmehr beispielhaft in Verbindung mit den **Fig.** 9–**Fig.** 10 dargestellt.

[0096] **Fig.** 9 ist eine Konzeptansicht, die anschauliche Geometrie-Verarbeitungseinheiten **550-0** und **550-1** darstellt, die ausgebildet sind, Indizes und Vertices gemäß einer Ausführungsform der vorliegenden Erfindung zu speichern. Wie gezeigt, umfasst die Geometrie-Verarbeitungseinheit **550-0** die Geometrie-Schattierungseinheit **504-0**, den lokalen Index-Puffer **508-0** und den lokalen Vertex-Puffer **510-0**. In ähnlicher Weise enthält die Geometrie-Verarbeitungseinheit **502-1** die Geometrie-Schattierungseinheit **504-1**, den lokalen Index-Puffer **508-1** und den lokalen Vertex-Puffer **510-1**. Die Geometrie-Verarbeitungseinheiten **550-0** und **550-1** sind auch in **Fig.** 5 gezeigt, obwohl in diesem Beispiel gewisse Elemente dieser Geometrie-Verarbeitungseinheiten **550** zum Zwecke der Klarheit weggelassen sind.

[0097] Die Geometrie-Verarbeitungseinheit **550-0** ist ausgebildet, Vertex-Daten und zugehörige Geometrie-Informationen, die mit einem grafischen Grundelement oder geometrischen Objekt **502** verknüpft sind, zu empfangen. Die Geometrie-Schattierungseinheit **504-0** erzeugt dann das geometrische Objekt **502-0**, das einen Streifen aus Dreiecken repräs-

sentiert, wobei Vertices bzw. Eckpunkte A, B, C, D und E Vertices sind, die zu diesen Dreiecken gehören. Die Geometrie-Schattierungseinheit **504-0** ist ferner ausgebildet, diese Vertices und zugehörige Vertex-Daten in dem lokalen Vertex-Puffer **510-0** zu speichern. Da unterschiedliche Dreiecke, die zu dem geometrischen Objekt **502-0** gehören, Vertices gemeinsam haben, werden diese gemeinsamen Vertices gegebenenfalls in dem lokalen Vertex-Puffer **510-0** nur einmal eingeführt. Die Geometrie-Schattierungseinheit **504-0** ist ferner ausgebildet, Indizes, die diese Vertices kennzeichnen, in dem lokalen Index-Puffer **508-0** zu speichern, wie dies gezeigt ist. In Situationen, in denen ein gegebener Vertex bereits in dem lokalen Vertex-Puffer **510-0** vorhanden ist, fügt die Geometrie-Schattierungseinheit **504-0** einen Index auf diesen Vertex in den lokalen Index-Puffer **508-0** ein, ohne dass dieser Vertex in dem lokalen Vertex-Puffer **510-0** erneut gespeichert wird, wodurch eine redundante Kopie von Vertex-Daten vermieden wird. In dem anschaulichen hierin erläuterten Szenario erzeugt die Geometrie-Schattierungseinheit **504-0** Dreiecke aus dem geometrischen Objekt **502-0** auf der Grundlage einer Umlaufrichtung im Uhrzeigersinn oder im Gegenuhrzeigersinn. Der Fachmann erkennt, dass die Geometrie-Schattierungseinheit **504-0** Dreiecke und/oder andere grafische Grundelemente unter Anwendung einer beliebigen speziellen Umlaufrichtung oder einer Kombination aus Umlaufrichtungen erzeugen kann.

[0098] Ferner kann die Geometrie-Schattierungseinheit **504-0** auch eine Gruppe an Indizes einfügen, die ein Dreieck repräsentieren, das nicht in dem geometrischen Objekt **502-0** enthalten ist (beispielsweise das Dreieck ACD, das den Indizes 0, 2 und 3 entspricht). In einer Ausführungsform ist die Geometrie-Schattierungseinheit **504-0** ausgebildet, die unterschiedlichen Dreiecke, die aus den Vertices A, B, C, D und E gebildet sind, zu erzeugen, indem ein komplexes geometrisches Objekt **502** in diese unterschiedlichen Dreiecke unterteilt wird. In einer weiteren Ausführungsform kann die Geometrie-Schattierungseinheit **504-0** ausgebildet sein, die aus den Vertices A, B, C, D und E gebildeten unterschiedlichen Dreiecke zu erzeugen, indem ein einfaches geometrisches Objekt **502**, etwa ein einfaches Dreieck, dupliziert wird. Die Geometrie-Schattierungseinheit **504-0** kann auch Indizes in den lokalen Index-Puffer **508-0** speichern, die das geometrische Objekt **502-0** als Ganzes repräsentieren, das heißt Indizes, die alle Vertices A, B, C, D und E repräsentieren.

[0099] Analog zu der Geometrie-Verarbeitungseinheit **550-0** ist die Geometrie-Verarbeitungseinheit **550-1** ausgebildet, Vertex-Daten und zugehörige Geometrie-Informationen, die mit einem grafischen Grundelement oder geometrischen Objekt **502** verknüpft sind, zu empfangen. Die Geometrie-Schattierungseinheit **504-1** erzeugt dann das geometrische Objekt **502-1**, das einen Streifen aus Dreiecken repräsentiert, wobei Vertices J, K, L, M, N und O Vertices sind, die mit diesen Dreiecken verknüpft sind. Die Geometrie-Schattierungseinheit **504-1** ist ferner ausgebildet, dann diese Vertices und zugehörige Vertex-Daten in dem lokalen Vertex-Puffer **510-1** zu speichern. Da unterschiedliche Dreiecke, die mit dem geometrischen Objekt **502-1** verknüpft sind, Vertices gemeinsam haben, werden diese gemeinsamen Vertices gegebenenfalls nur einmal in dem lokalen Vertex-Puffer **510-1** abgelegt. Die Geometrie-Schattierungseinheit **504-1** ist ferner ausgebildet, Indizes, die diese Vertices kennzeichnen, in dem lokalen Index-Puffer **508-1** zu speichern, wie dies gezeigt ist. In Situationen, in denen ein gegebener Vertex bereits in den lokalen Vertex-Puffer **510-1** vorhanden ist, kann gegebenenfalls die Geometrie-Schattierungseinheit **504-1** einen Index auf diesem Vertex in den lokalen Index-Puffer **508-1** einfügen, ohne dass dieser Vertex in dem lokalen Vertex-Puffer **510-1** erneut gespeichert wird, wodurch redundante Kopien von Vertex-Daten vermieden werden. In dem hierin erläuterten anschaulichen Szenario erzeugt die Geometrie-Schattierungseinheit **504-1** Dreiecke aus dem geometrischen Objekt **502-1** auf der Grundlage einer Umlaufrichtung im Uhrzeigersinn oder im Gegenuhrzeigersinn. Der Fachmann erkennt, dass die Geometrie-Schattierungseinheit **504-1** Dreiecke und/oder andere grafische Grundelemente unter Anwendung einer speziellen Umlaufrichtung oder eine Kombination von Umlaufrichtungen erzeugen kann.

[0100] In einer Ausführungsform ist die Geometrie-Schattierungseinheit **504-1** ausgebildet, die unterschiedlichen Dreiecke aus den Vertices J, K, L, M, N und O zu erzeugen, indem ein komplexes geometrisches Objekt **502** in diese unterschiedlichen Dreiecke zerlegt wird. In einer weiteren Ausführungsform kann die Geometrie-Schattierungseinheit **504-1** ausgebildet sein, die unterschiedlichen Dreiecke, die aus den Vertices J, K, L, M, N und O gebildet sind, zu erzeugen, indem ein einfaches geometrisches Objekt **502**, etwa ein einfaches Dreieck, wiederholt erzeugt wird. Die Geometrie-Schattierungseinheit **504-1** kann ferner auch Indizes in dem lokalen Index-Puffer **508-1** speichern, die das geometrische Objekt **502-1** als Ganzes repräsentieren, das heißt Indizes, die alle Vertices J, K, L, N, N und O repräsentieren.

[0101] Die Datenstromausgabeeinheiten **512-0** und **512-1** (in **Fig.** 5 gezeigt) können dann die Vertices und Indizes, die entsprechend in dem lokalen Vertex-Puffer **510** und dem entsprechenden lokalen Index-Puffer **508** gespeichert sind, als Datenstrom entsprechend an den globalen Vertex-Puffer **518** und den globalen Index-Puffer **516** auf der Grundlage und Basisadressen leiten, die von der SSYNC-Einheit **514** bereitgestellt werden. Ein anschaulicher globaler Vertex-Puffer **518** und ein anschaulicher globaler Index-Puffer **516** sind in **Fig.** 10 gezeigt.

[0102] **Fig.** 10 ist eine Konzeptansicht, die einen anschaulichen globalen Vertex-Puffer **518** und einen anschaulichen globalen Index-Puffer **516** zeigt, die ausgebildet sind, entsprechend Vertices und Indizes gemäß einer Ausführungsform der vorliegenden Erfindung zu speichern.

[0103] Wie gezeigt, umfasst der globale Vertex-Puffer **518** entsprechend unterschiedliche Vertices, die mit dem grafischen Grundelementen **502-0** und **502-1** verknüpft sind, die in **Fig.** 9 gezeigt sind. Insbesondere enthält der globale Vertex-Puffer **518** die Vertices A, B, C, D, E, die den grafischen Grundelementen **502-0** entsprechen, sowie die Vertices J, K, L, N, N und O, die den grafischen Grundelementen **502-1** entsprechen. Die Geometrie-Verarbeitungseinheit **550-0** ist ausgebildet, die Vertices A–E in den globalen Index-Puffer **518** auf der Grundlage einer Basisadresse zu schreiben, die von der SSYNC-Einheit **514** empfangen wird.

[0104] Analog dazu ist die Geometrie-Verarbeitungseinheit **550-1** ausgebildet, die Vertices J–O in den globalen Index-Puffer **518** auf der Grundlage einer anderen Basisadresse zu schreiben, die von der SSYNC-Einheit **514** empfangen wird. Wie ferner gezeigt ist, enthält der globale Index-Puffer **516** Indizes auf die Vertices, die in dem globalen Vertex-Puffer **518** gespeichert sind. Die Geometrie-Verarbeitungseinheiten **550-0** und **550-1** sind ausgebildet, diese Indizes in den globalen Index-Puffer **516** auf der Grundlage der Indizes zuschreiben, die in dem lokalen Index-Puffer **508-0** und **508-1** gespeichert sind, und auf der Grundlage von Basisadressen, die von der SSYNC-Einheit **514** empfangen werden.

[0105] In diesem Beispiel bedient die SSYNC-Einheit **514** die Geometrie-Verarbeitungseinheiten **550-0** und **550-1** der Reihe nach, wobei mit der Geometrie-Verarbeitungseinheit **550-0** begonnen wird. Die SSYNC-Einheit **514** empfängt Daten aus der Geometrie-Verarbeitungseinheit **550-0**, die die Anzahl an Vertices A–E angeben, die in den globalen Vertex-Puffer **518** zu schreiben sind (diese Anzahl ist in dem vorliegenden Beispiel 5). Die SSYNC-Einheit **514** antwortet der Geometrie-Verarbeitungseinheit **550-0** mit der aktuellen Basisadresse für den globalen Vertex-Puffer **518**. Anfänglich bewahrt die SSYNC-Einheit **514** eine anfängliche Basisadresse in dem globalen Vertex-Puffer **518** von „0”. Die SSYNC-Einheit **514** aktualisiert diese aktuelle Basisadresse auf der Grundlage der Anzahl an Vertices, die die Geometrie-Verarbeitungseinheit **550-0** in den globalen Vertex-Speicher **518** schreiben wird, um eine neue Basisadresse für den globalen Vertex-Puffer **518** anzugeben, an der weitere Vertices und zugehörige Daten sicher geschrieben werden können (in diesem Beispiel eine Basisadresse von „5”).

[0106] Nach dem Erhalt der Daten aus der Geometrie-Verarbeitungseinheit **550-0**, die die Anzahl an Vertices A–E angeben, kann dann die SSYNC-Einheit **514** weitere Daten aus der Geometrie-Verarbeitungseinheit **550-0** erhalten, die die Anzahl an unterschiedlichen Gruppen aus Indizes bezeichnen, die in den globalen Index-Puffer **516** zu schreiben sind (in diesem Beispiel ist diese Anzahl 4). Wiederum kann jede Gruppe an Indizes einem anderen Dreieck innerhalb der grafischen Grundelemente **502-0** entsprechen. Die SSYNC-Einheit **514** antwortet der Geometrie-Verarbeitungseinheit **550-0** mit der aktuellen Basisadresse in dem globalen Index-Puffer **516**. Zunächst bewahrt die SSYNC-Einheit **514** eine anfängliche Basisadresse in dem globalen Index-Puffer **516** von „0”. Die SSYNC-Einheit **514** kann dann diese aktuelle Basisadresse auf der Grundlage der Anzahl an Indizes aktualisieren, die die Geometrie-Verarbeitungseinheit **550-0** in den globalen Index-Puffer **516** schreiben wird, um eine neue Basisadresse in dem globalen Index-Puffer **516** anzugeben, an der weitere Indizes sicher gespeichert werden können (in diesem Beispiel eine Basisadresse von „4”).

[0107] Danach kann die SSYNC-Einheit **514** die Geometrie-Verarbeitungseinheit **550-1** bedienen. Die SSYNC-Einheit **514** empfängt Daten aus der Geometrie-Verarbeitungseinheit **550-1**, die die Anzahl an Vertices J–O angibt, die in den globalen Vertex-Puffer **518** zu schreiben ist (in diesem Beispiel ist diese Anzahl 6). Die SSYNC-Einheit **514** antwortet der Geometrie-Verarbeitungseinheit **550-1** mit der aktuellen Basisadresse in dem globalen Vertex-Puffer **518** von „6”. Die SSYNC-Einheit **514** aktualisiert dann die aktuelle Basisadresse auf der Grundlage der Anzahl an Vertices, die die Geometrie-Verarbeitungseinheit **550-1** in den globalen Vertex-Puffer **518** schreiben wird, um eine neue Basisadresse in dem globalen Vertex-Puffer **518** anzugeben, an der Vertices und zugehörige Daten sicher gespeichert werden können (in diesem Beispiel eine Basisadresse von „11”).

[0108] Nach dem Empfang der Daten aus der Geometrie-Verarbeitungseinheit **550-1**, die die Anzahl an Vertices J angeben, kann dann die SSYNC-Einheit **514** weitere Daten aus der Geometrie-Verarbeitungseinheit **550-1** empfangen, die die Anzahl unterschiedlicher Gruppen von Indizes angeben, die in den globalen Index-Puffer **516** zu schreiben sind (in diesem Beispiel ist die Anzahl 4). Wiederum entspricht jede Gruppe an Indizes einem unterschiedlichen Dreieck in den grafischen Grundelementen **502-1**. Die SSYNC-Einheit **514** antwortet der Geometrie-Verarbeitungseinheit **550-1** mit der aktuellen Basisadresse in dem globalen Index-Puffer **516** von „4”. Die SSYNC-Einheit **514** aktualisiert dann diese aktuelle Basisadresse auf der Grundlage der Anzahl an Indizes, die die Geometrie-Verarbeitungseinheit **550-1** in den globalen Index-Puffer **516** schreiben wird, um eine neue Basisadresse in dem globalen Index-Puffer

516 anzugeben, an der weitere Indizes sicher gespeichert werden können (in diesem Beispiel eine Basisadresse von „8").

[0109] Wenn die Geometrie-Verarbeitungseinheiten **550-0** oder **550-1** Indizes in den globalen Index-Puffer **516** gemäß der zuvor beschriebenen Technik schreiben, ist jede dieser Geometrie-Verarbeitungseinheiten **550** ausgebildet, die Indizes zu aktualisieren auf der Grundlage der Basisadresse in dem globalen Vertex-Puffer **518**, die aus der SSYNC-Einheit **514** empfangen wurde. Folglich kann die Geometrie-Verarbeitungseinheit **550-0** den Index um „0" erhöhen, d. h. der Basisadresse in dem globalen Vertex-Puffer **518**, die von der SSYNC-Einheit **514** bereitgestellt wird, wenn sie die Geometrie-Verarbeitungseinheit **550-0** bedient. In ähnlicher Weise kann die Geometrie-Verarbeitungseinheit **550-1** den Index um „5" erhöhen, d. h., die Basisadresse in dem globalen Vertex-Puffer **518**, die von der SSYNC-Einheit **514** bereitgestellt wird, wenn diese die Geometrie-Verarbeitungseinheit **550-1** bedient. Mit dieser Vorgehensweise aktualisiert jede Geometrie-Verarbeitungseinheit **550** Indizes, die dem globalen Index-Puffer **516** zugeleitet werden, so dass korrekte Vertices wiedergegeben werden, die in dem globalen Vertex-Puffer **518** gespeichert sind.

[0110] Der Fachmann erkennt, dass das in Verbindung mit den **Fig.** 9 und **Fig.** 10 beschriebene Beispiel nur eine mögliche Situation darstellt, in der die Funktion der vorliegenden Erfindung eingerichtet werden kann, und dass die vorliegende Erfindung auch in einer breiten Fülle anderer Situationen realisierbar ist.

[0111] Zusammengefasst gilt: eine grafische Verarbeitungseinheit umfasst eine Gruppe aus Geometrie-Verarbeitungseinheiten, die jeweils ausgebildet sind, grafische Grundelemente oder geometrische Objekte parallel miteinander zu verarbeiten. Eine gegebene Geometrie-Verarbeitungseinheit erzeugt ein oder mehrere grafische Grundelemente oder ein oder mehrere geometrische Objekte und speichert Vertex-Daten, die mit dem bzw. den grafischen Grundelement(en) oder dem bzw. den die grafischen Objekt(en) in Bezug stehen, in einem lokalen Puffer. Die Geometrie-Verarbeitungseinheit erzeugt ebenfalls unterschiedliche Gruppen an Indizes für diese Vertices, wobei jede derartige Gruppe ein anderes grafisches Grundelement oder geometrisches Objekt repräsentiert. Die Geometrie-Verarbeitungseinheiten leiten dann die zwischengespeicherten Vertices und Indizes den globalen Puffern als Datenstrom zu. Eine Datenstromausgabesynchronisier-Einheit koordiniert den Datenstrom von Vertices und Indizes in den unterschiedlichen Geometrie-Verarbeitungseinheiten, indem jeder Geometrie-Verarbeitungseinheit eine andere Basisadresse in einem globalen Vertex-Puffer gegeben wird, an der Vertices geschrieben werden können. Die Datenstromausgabesynchronisier-Einheit leitet jeder Geometrie-Verarbeitungseinheit auch eine andere Basisadresse in einen globalen Index-Puffer zu, an der Indizes gespeichert werden können.

[0112] Vorteilhafterweise speichert mit der offenbarten Vorgehensweise die Geometrie-Verarbeitungseinheit keine redundanten Kopien von Vertex-Daten, da die Vertex-Daten lokal indiziert werden, wodurch GPU-Ressourcen eingespart werden. Ferner kann jede derartige Geometrie-Verarbeitungseinheit lokal erzeugte Vertex-Daten in einem globalen Vertex-Puffer speichern, der ebenfalls indiziert ist. Die Indizes für den globalen Index-Puffer werden über alle Geometrie-Verarbeitungseinheiten so verteilt, dass die Größe des Vertex-Puffers für das Gesamtsystem optimiert ist. Da der indizierte globale Vertex-Puffer indiziert wird, kann dieser Puffer wesentlich kleiner sein als konventionelle nicht-indizierte globale Vertex-Puffer. Mit einem kleineren globalen Vertex-Puffer kann die Speisung des globalen Vertex-Puffers zurück in die Stufen der Grafikverarbeitungs-Pipeline, die den Geometrie-Verarbeitungseinheiten vorgelagert ist, daher eine wesentlich effizientere Angelegenheit werden im Vergleich zu bekannten Architekturen, wodurch die Verarbeitungseffizienz des Gesamtsystems verbessert wird

[0113] Eine Ausführungsform der Erfindung kann als ein Programmprodukt zur Verwendung in einem Computersystem realisiert werden. Das bzw. die Programme des Programmprodukts definieren Funktionen der Ausführungsformen (einschließlich der hierin beschriebenen Verfahren) und können in einer Vielzahl von computerlesbaren Speichermedien enthalten sein. Zu anschaulichen computerlesbaren Speichermedien gehören, ohne Einschränkung: (i) nichtbeschreibbare Speichermedien (beispielsweise Nur-Lese-Speichereinrichtungen in einem Computer, etwa CD-ROM-Disketten, die von einem CD-ROM-Laufwerk lesbar sind, Flash-Speicher, ROM-Chips oder eine andere Art eines nicht flüchtigen Halbleiterspeichers), auf welchen Information permanent gespeichert ist; und (ii) beschreibbare Speichermedien (beispielsweise Disketten in einem Diskettenlaufwerk oder ein Festplattenlaufwerk oder eine andere Art eines Halbleiterspeichers mit wahlfreiem Zugriff), auf welchen änderbare Information gespeichert ist.

[0114] Die Erfindung ist mit Bezug zu speziellen Ausführungsformen beschrieben worden. Der Fachmann erkennt jedoch, dass diverse Modifizierungen und Änderungen daran vorgenommen werden können, ohne von dem breiteren Grundgedanken und dem Schutzbereich der Erfindung abzuweichen, wie sie in den angefügten Patentansprüchen angegeben ist. Die vorhergehende Beschreibung und die Zeichnungen sind daher als anschaulich und nicht als beschränkend zu betrachten.

[0115] Daher ist der Schutzbereich der vorliegenden Erfindung durch die folgenden Patentansprüche festgelegt.

Patentansprüche

1. Ein computerimplementiertes Verfahren zum Füllen mehrerer globaler Puffer, wobei das Verfahren umfasst:
Empfangen von Daten aus einer ersten Verarbeitungseinheit, die eine Anzahl an Einträgen innerhalb eines ersten lokalen Index-Puffers angeben, die von der ersten Verarbeitungseinheit erzeugt sind, wobei die erste Verarbeitungseinheit eine von mehreren Verarbeitungseinheiten ist; und
Senden, an die erste Verarbeitungseinheit, einer ersten Basisadresse für einen globalen Index-Puffer, die eine erste Position in dem globalen Index-Puffer angibt, an der die Einträge aus dem ersten lokalen Index-Puffer gespeichert werden sollten, wobei die erste Basisadresse für den ersten globalen Index-Puffer auf einer Anzahl an Einträgen in dem globalen Index-Puffer beruht, die bereits mindestens einer anderen Verarbeitungseinheit aus den mehreren Verarbeitungseinheiten zugewiesen ist.

2. Ein Grafiksubsystem mit:
einer Datenstromsynchronisier-Einheit, die ausgebildet ist, um:
aus einer ersten Verarbeitungseinheit Daten zu empfangen, die eine Anzahl an Einträgen in einem ersten lokalen Index-Puffer angeben, die von der ersten Verarbeitungseinheit erzeugt sind, wobei die erste Verarbeitungseinheit eine von mehreren Verarbeitungseinheiten ist; und
an die erste Verarbeitungseinheit eine erste Basisadresse für einen globalen Index-Puffer zu senden, die eine erste Position in dem globalen Index-Puffer wiedergibt, an der die Einträge aus dem ersten lokalen Index-Puffer gespeichert werden sollten, wobei die erste Basisadresse für den ersten globalen Index-Puffer auf einer Anzahl von Einträgen in dem globalen Index-Puffer beruht, die bereits mindestens einer anderen Verarbeitungseinheit aus den mehreren Verarbeitungseinheiten zugewiesen ist.

3. Das Grafiksubsystem nach Anspruch 2, wobei die Datenstromsynchronisier-Einheit ferner ausgebildet ist, eine zweite Basisadresse für den globalen Index-Puffer zu erzeugen, indem die erste Basisadresse für den globalen Index-Puffer auf der Grundlage einer Anzahl an Einträgen in dem globalen Index-Puffer aktualisiert wird, die bereits der ersten Verarbeitungseinheit zugewiesen ist.

4. Das Grafiksubsystem nach Anspruch 3, wobei die Datenstromsynchronisier-Einheit ferner ausgebildet ist, um:
aus einer zweiten Datenverarbeitungseinheit Daten zu empfangen, die eine Anzahl an Einträgen in einem zweiten lokalen Index-Puffer angeben, die von der zweiten Verarbeitungseinheit erzeugt sind, wobei die zweite Verarbeitungseinheit eine der mehreren Verarbeitungseinheiten ist;
an die zweite Verarbeitungseinheit die zweite Basisadresse für den globalen Index-Puffer zu senden, die eine zweite Position in dem globalen Index-Puffer wiedergibt, an der die Einträge aus dem zweiten lokalen Index-Puffer gespeichert werden sollten; und
eine dritte Basisadresse für den globalen Index-Puffer zu erzeugen, indem die zweite Basisadresse für den globalen Index-Puffer auf der Grundlage einer Anzahl an Einträgen in dem globalen Index-Puffer, die der zweiten Verarbeitungseinheit zugewiesen ist, aktualisiert wird.

5. Das Grafiksubsystem nach Anspruch 2, wobei die Datenstromsynchronisier-Einheit ferner ausgebildet ist, um:
aus der ersten Verarbeitungseinheit Daten zu empfangen, die eine Anzahl an Einträgen in einem ersten lokalen Vertex-Puffer angeben, die von der ersten Verarbeitungseinheit erzeugt sind; und
an die erste Verarbeitungseinheit eine erste Basisadresse für einen globalen Vertex-Puffer zu senden, die eine erste Position in dem globalen Vertex-Puffer wiedergibt, an der die Einträge aus dem ersten lokalen Vertex-Puffer gespeichert werden sollten, wobei die erste Basisadresse für den globalen Vertex-Puffer auf einer Anzahl an Einträgen in dem globalen Vertex-Puffer beruht, die bereits der mindestens einen anderen Verarbeitungseinheit aus den mehreren Verarbeitungseinheiten zugewiesen ist.

6. Das Grafiksubsystem nach Anspruch 5, wobei die Datenstromsynchronisier-Einheit ferner ausgebildet ist, eine zweite Basisadresse für den globalen Vertex-Puffer zu erzeugen, indem die erste Basisadresse für den globalen Vertex-Puffer auf der Grundlage einer Anzahl an Einträgen in dem globalen Vertex-Puffer aktualisiert wird, die bereits der ersten Verarbeitungseinheit zugewiesen ist.

7. Das Grafiksubsystem nach Anspruch 6, wobei die Datenstromsynchronisier-Einheit ferner ausgebildet ist, um:
aus einer zweiten Verarbeitungseinheit Daten zu empfangen, die eine Anzahl an Einträgen in einem zweiten lokalen Vertex-Puffer angeben, die von der zweiten Verarbeitungseinheit erzeugt sind, wobei die zweite Verarbeitungseinheit eine der mehreren Verarbeitungseinheiten ist;
an die zweite Verarbeitungseinheit die zweite Basisadresse für den globalen Vertex-Puffer zu senden, die eine zweite Position in dem globalen Vertex-Puffer wiedergibt, an der die Einträge aus dem zweiten lokalen Vertex-Puffer gespeichert werden sollten; und
eine dritte Basisadresse für den globalen Vertex-Puffer zu erzeugen, indem die zweite Basisadresse für

den globalen Vertex-Puffer auf der Grundlage einer Anzahl an Einträgen des globalen Vertex-Puffers aktualisiert wird, die der zweiten Verarbeitungseinheit zugewiesen ist.

8. Das Grafiksubsystem nach Anspruch 4, wobei eine Reihenfolge, in der die Datenstromsynchronisier-Einheit Daten verarbeitet, die aus der ersten Verarbeitungseinheit und der zweiten Verarbeitungseinheit empfangen werden, auf der Reihenfolge einer Anwendungsprogrammierschnittstelle (API) beruht.

9. Eine Recheneinrichtung, die zum Füllen mehrerer globaler Puffer ausgebildet ist, mit:
einem Grafikverarbeitungssubsystem, das eine Datenstromsynchronisier-Einheit umfasst, die ausgebildet ist, um:
aus einer ersten Verarbeitungseinheit Daten zu empfangen, die eine Anzahl an Einträgen in einem ersten lokalen Index-Puffer angeben, die von der ersten Verarbeitungseinheit erzeugt sind, wobei die erste Verarbeitungseinheit eine von mehreren Verarbeitungseinheiten ist, und
an die erste Verarbeitungseinheit eine erste Basisadresse für einen globalen Index-Puffer zu senden, die eine erste Position in dem globalen Index-Puffer wiedergibt, an der Einträge aus dem ersten lokalen Index-Puffer gespeichert werden sollten, wobei die erste Basisadresse für den ersten globalen Index-Puffer auf einer Anzahl an Einträgen in dem globalen Index-Puffer beruht, die bereits mindestens einer anderen Verarbeitungseinheit aus den mehreren Verarbeitungseinheiten zugewiesen ist.

10. Die Recheneinrichtung nach Anspruch 9, wobei die Datenstromsynchronisier-Einheit ferner ausgebildet ist, um:
aus der ersten Verarbeitungseinheit Daten zu empfangen, die eine Anzahl an Einträgen in einem ersten lokalen Vertex-Puffer angeben, die von der ersten Verarbeitungseinheit erzeugt sind; und
an die erste Verarbeitungseinheit eine erste Basisadresse für einen globalen Vertex-Puffer zu senden, die eine erste Position in dem globalen Vertex-Puffer wiedergibt, an der die Einträge aus dem ersten lokalen Vertex-Puffer gespeichert werden sollten, wobei die erste Basisadresse für den globalen Vertex-Puffer auf einer Anzahl an Einträgen des globalen Vertex-Puffers beruht, die bereits der mindestens einen anderen Verarbeitungseinheit aus den mehreren Verarbeitungseinheiten zugewiesen ist;
eine zweite Basisadresse für den globalen Vertex-Puffer zu erzeugen, indem die erste Basisadresse für den globalen Vertex-Puffer auf der Grundlage einer Anzahl an Einträgen in dem globalen Vertex-Puffer, die bereits der ersten Verarbeitungseinheit zugewiesen ist, aktualisiert wird.

Es folgen 10 Seiten Zeichnungen

Anhängende Zeichnungen

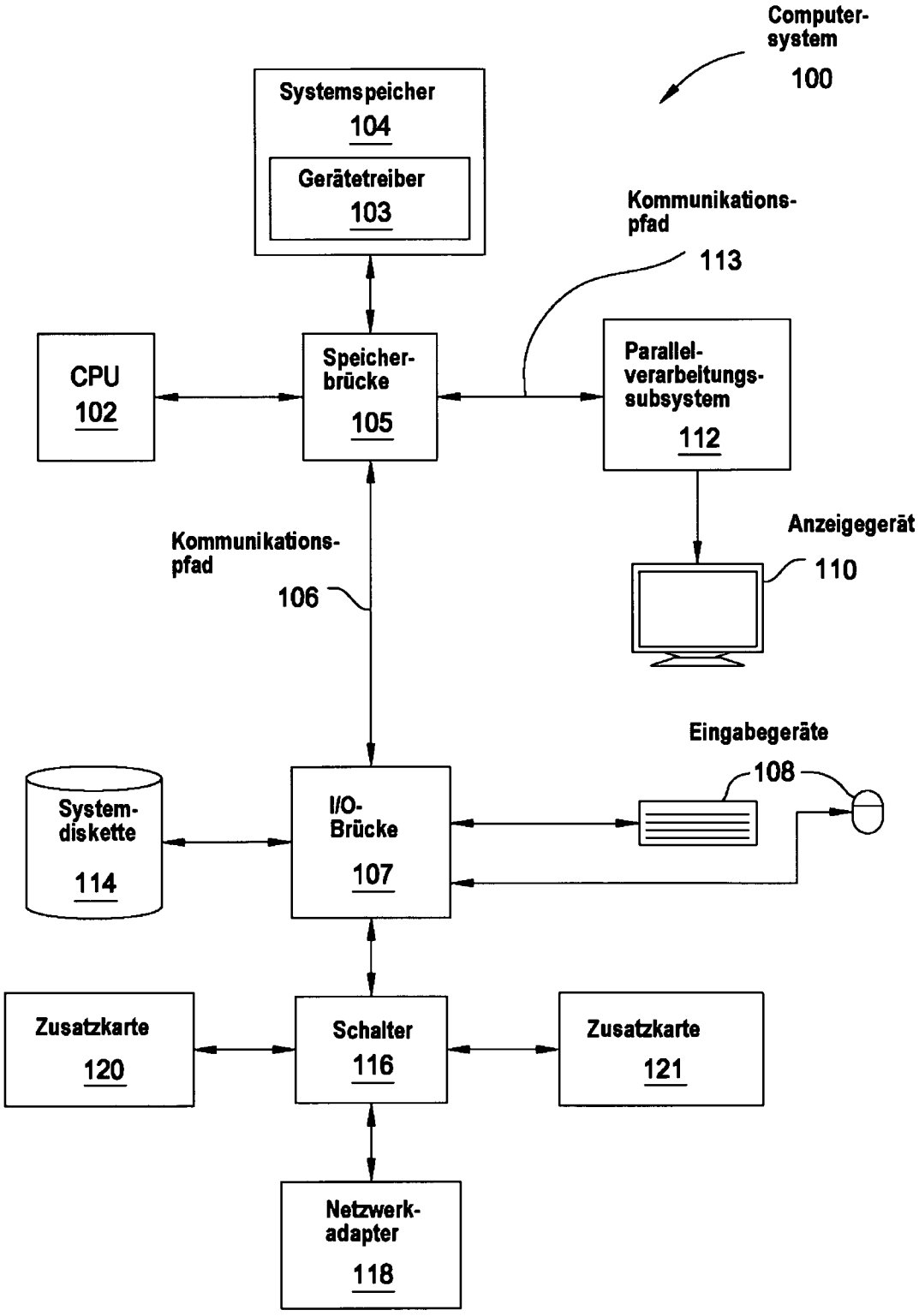


FIG. 1

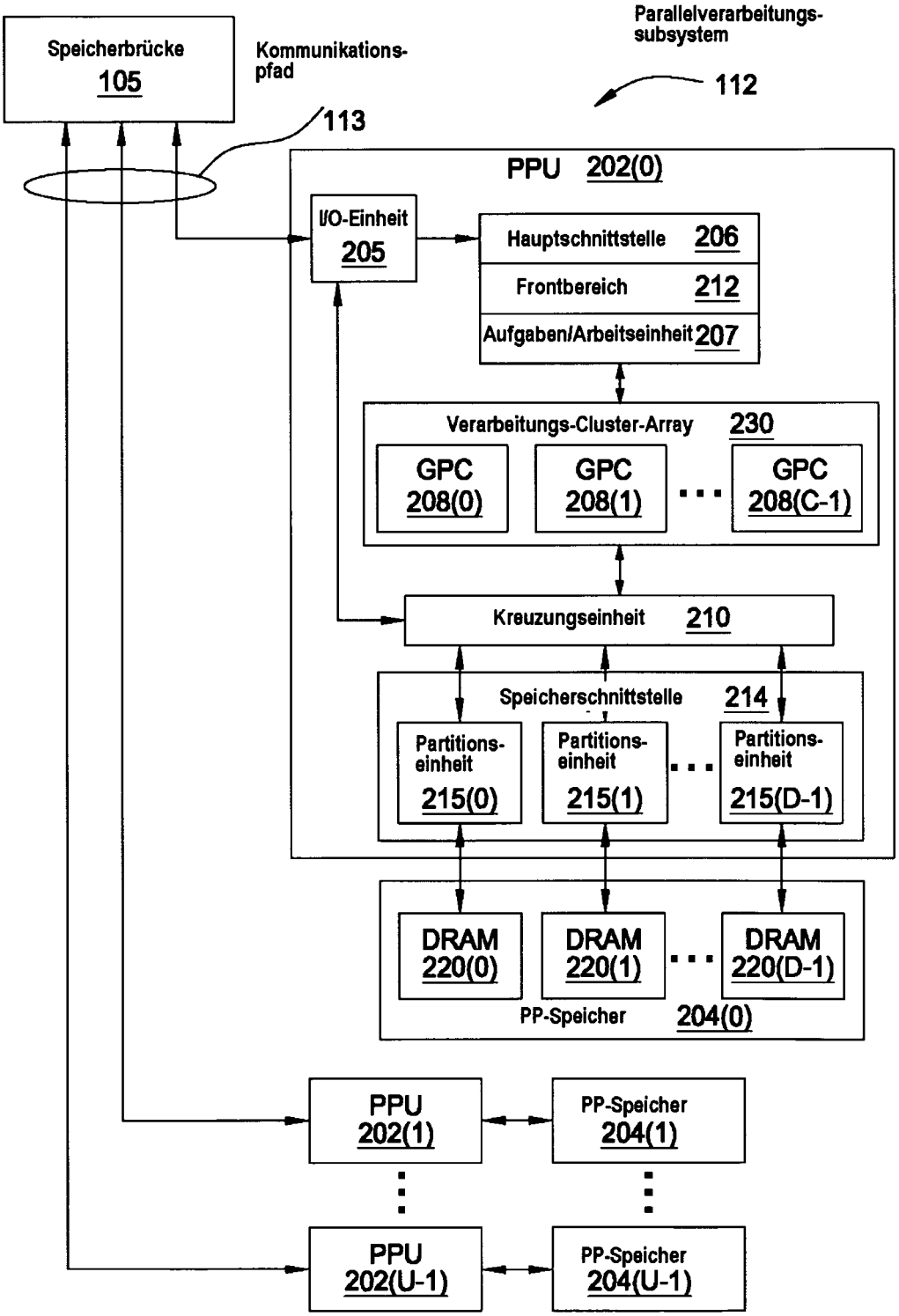

Speicherbrücke
105
Kommunikations-
pfad
113
Parallelverarbeitungs-
subsystem
112
PPU 202(0)
I/O-Einheit
205
Hauptschnittstelle 206
Frontbereich 212
Aufgaben/Arbeitseinheit 207
Verarbeitungs-Cluster-Array 230
GPC
208(0)
GPC
208(1)
GPC
208(C-1)
Kreuzungseinheit 210
Speicherschnittstelle 214
Partitions-
einheit
215(0)
Partitions-
einheit
215(1)
Partitions-
einheit
215(D-1)
DRAM
220(0)
DRAM
220(1)
DRAM
220(D-1)
PP-Speicher 204(0)
PPU
202(1)
PP-Speicher
204(1)
PPU
202(U-1)
PP-Speicher
204(U-1)

FIG. 2

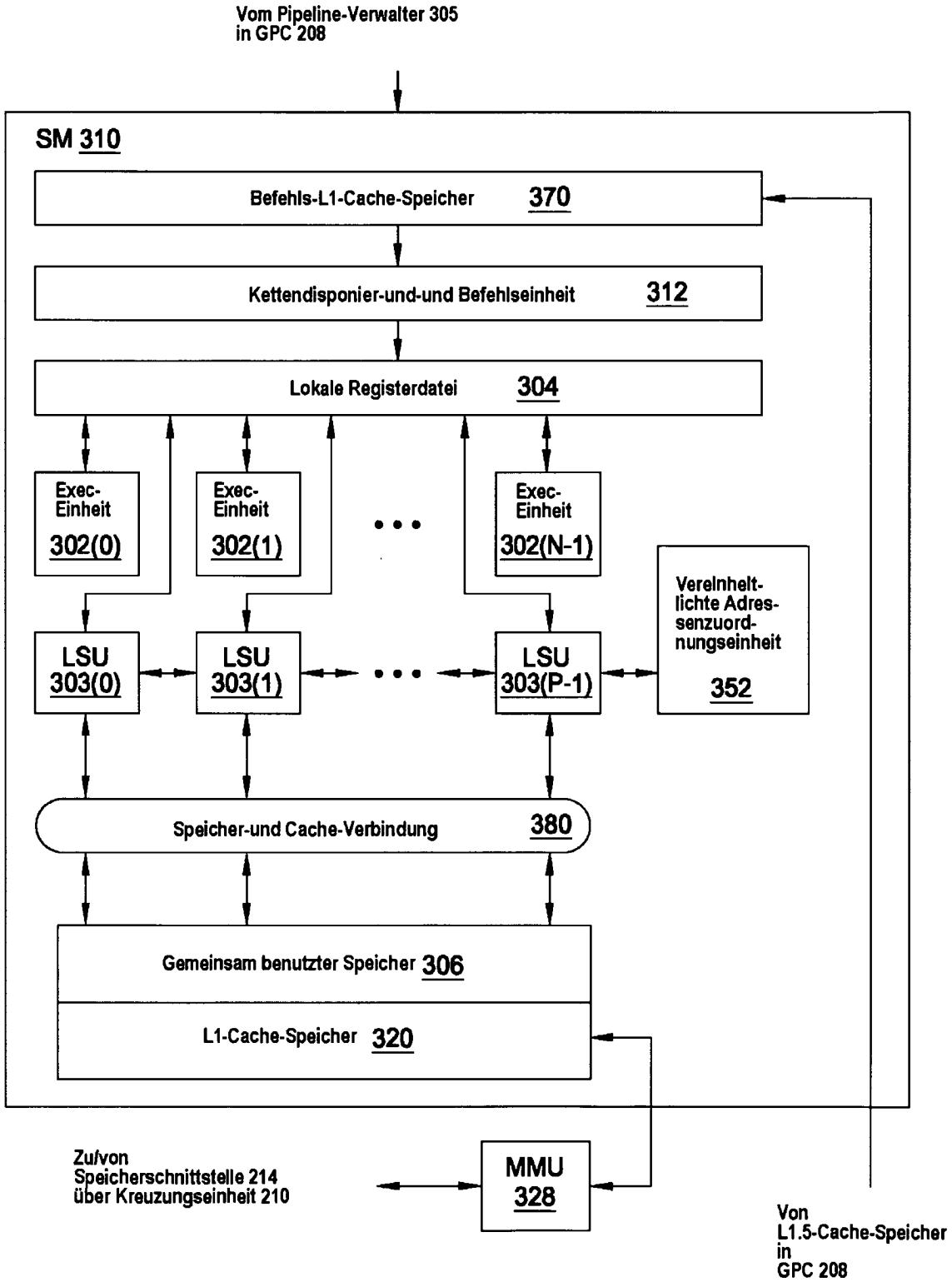

Vom Pipeline-Verwalter 305
in GPC 208
SM 310
Befehls-L1-Cache-Speicher 370
Kettendisponier-und-und Befehlseinheit 312
Lokale Registerdatei 304
Exec-Einheit 302(0)
Exec-Einheit 302(1)
Exec-Einheit 302(N-1)
LSU 303(0)
LSU 303(1)
LSU 303(P-1)
Vereinheitlichte Adressenzuordnungseinheit 352
Speicher-und Cache-Verbindung 380
Gemeinsam benutzter Speicher 306
L1-Cache-Speicher 320
Zu/von Speicherschnittstelle 214 über Kreuzungseinheit 210
MMU 328
Von L1.5-Cache-Speicher in GPC 208

FIG. 3

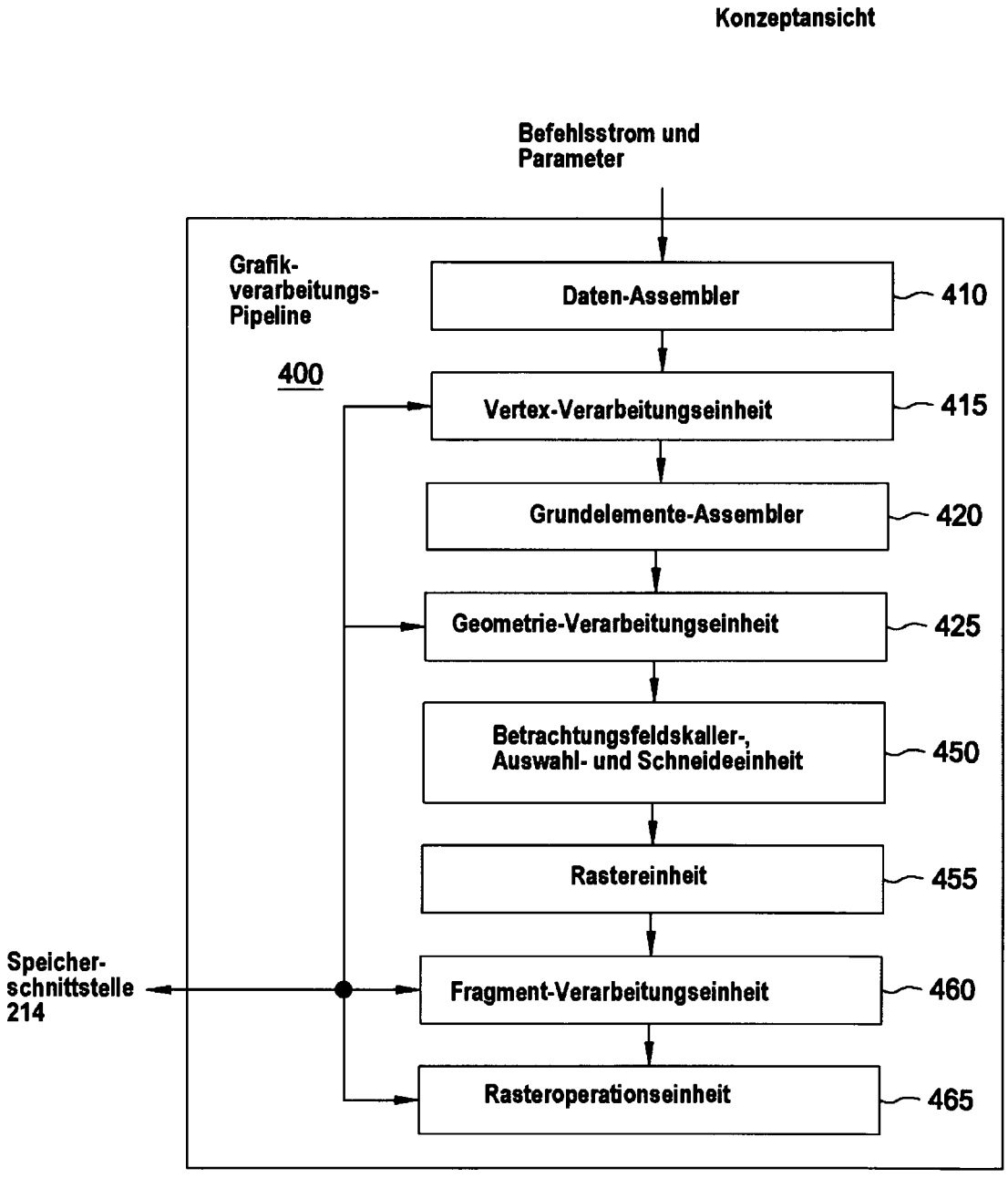

Konzeptansicht
Befehlsstrom und Parameter
Grafik-verarbeitungs-Pipeline
400
Daten-Assembler
410
Vertex-Verarbeitungseinheit
415
Grundelemente-Assembler
420
Geometrie-Verarbeitungseinheit
425
Betrachtungsfeldskaller-, Auswahl- und Schneideeinheit
450
Rastereinheit
455
Speicher-schnittstelle 214
Fragment-Verarbeitungseinheit
460
Rasteroperationseinheit
465

FIG. 4

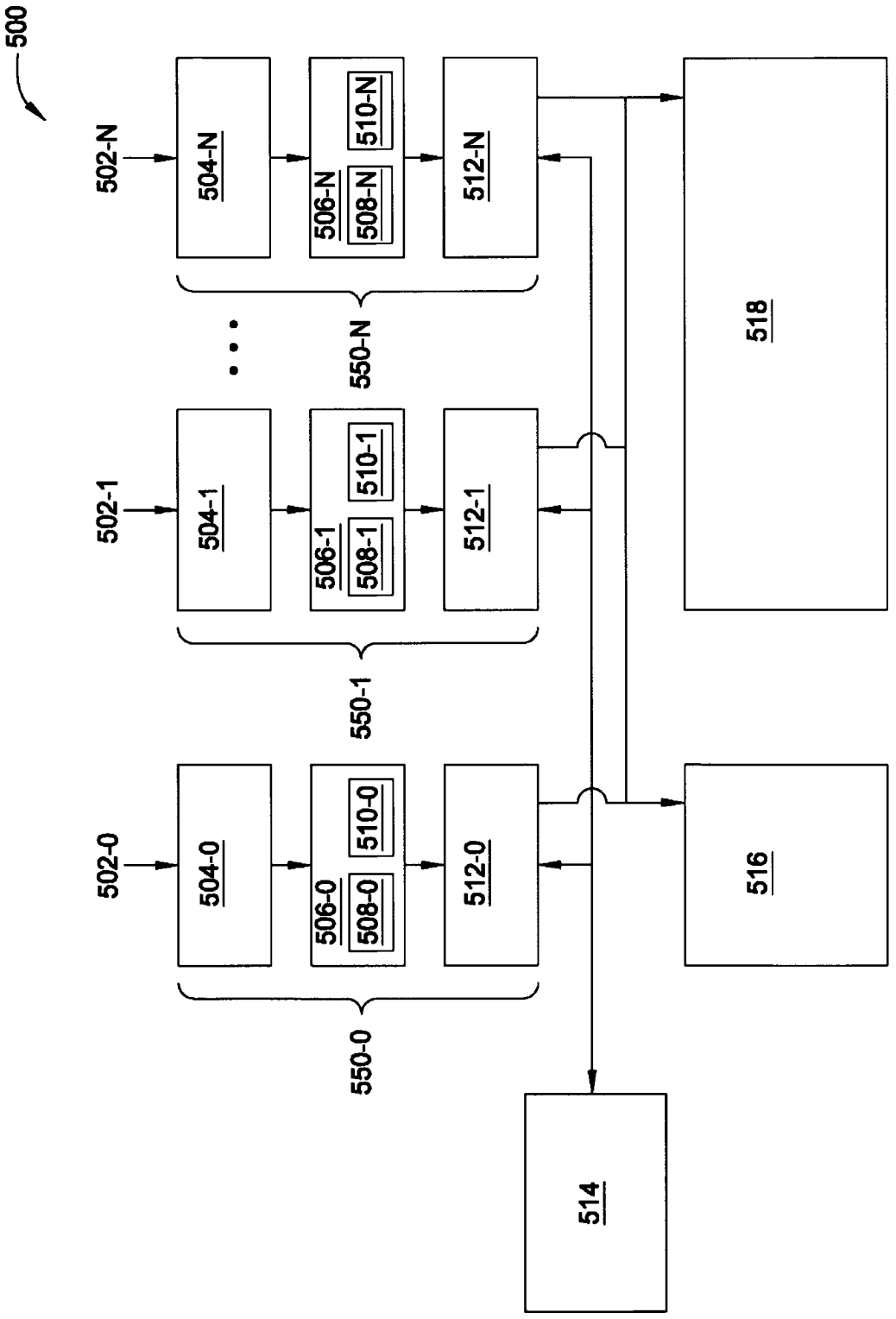

500
502-N
504-N
506-N
508-N
510-N
512-N
550-N
502-1
504-1
506-1
508-1
510-1
512-1
550-1
502-0
504-0
506-0
508-0
510-0
512-0
550-0
514
516
518

FIG. 5

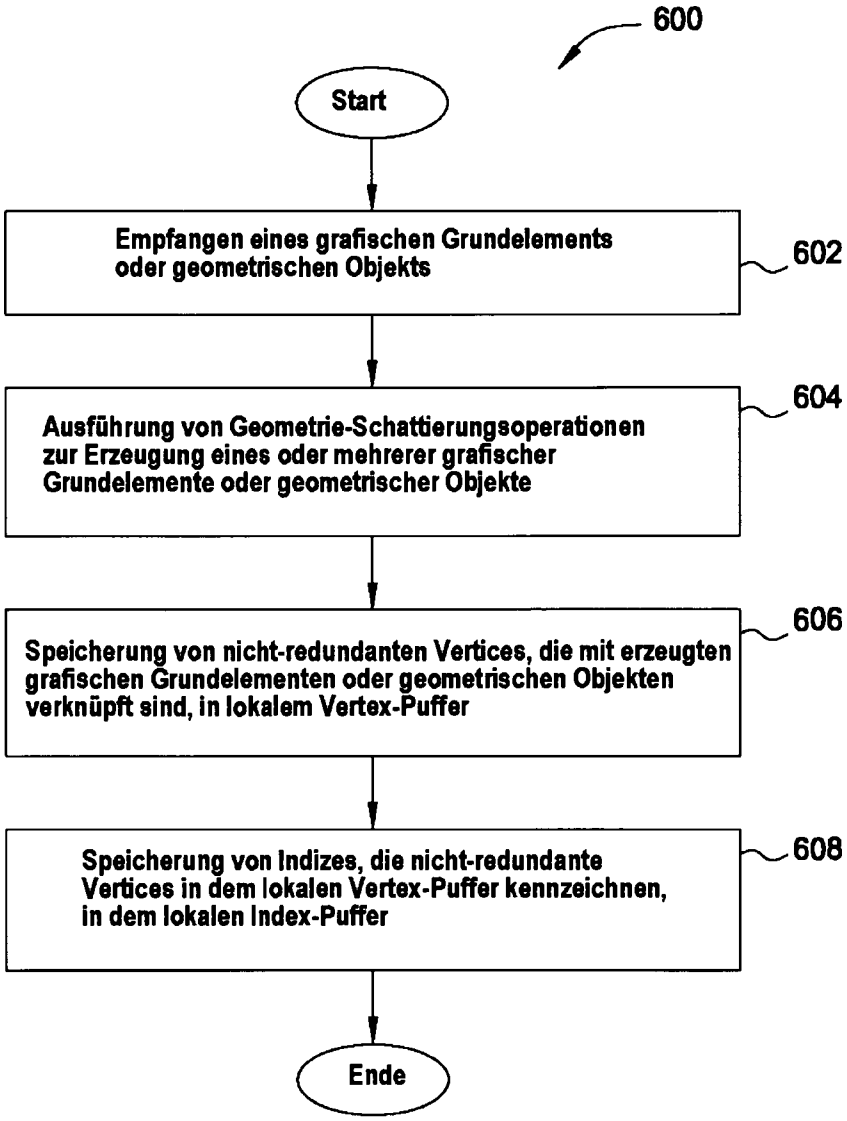

600
Start
Empfangen eines grafischen Grundelements oder geometrischen Objekts
602
Ausführung von Geometrie-Schattierungsoperationen zur Erzeugung eines oder mehrerer grafischer Grundelemente oder geometrischer Objekte
604
Speicherung von nicht-redundanten Vertices, die mit erzeugten grafischen Grundelementen oder geometrischen Objekten verknüpft sind, in lokalem Vertex-Puffer
606
Speicherung von Indizes, die nicht-redundante Vertices in dem lokalen Vertex-Puffer kennzeichnen, in dem lokalen Index-Puffer
608
Ende

FIG. 6

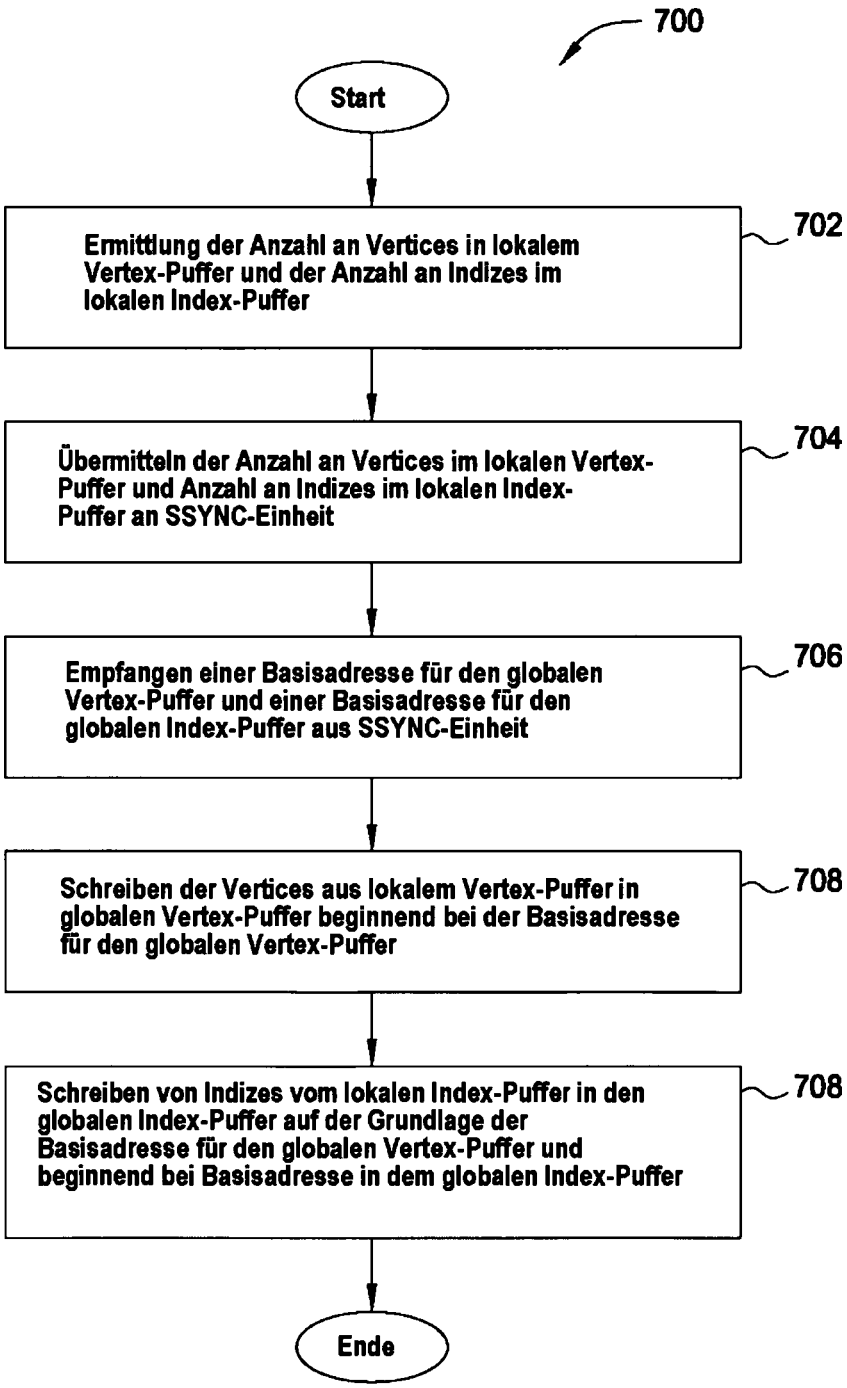

700
Start
Ermittlung der Anzahl an Vertices in lokalem Vertex-Puffer und der Anzahl an Indizes im lokalen Index-Puffer
702
Übermitteln der Anzahl an Vertices im lokalen Vertex-Puffer und Anzahl an Indizes im lokalen Index-Puffer an SSYNC-Einheit
704
Empfangen einer Basisadresse für den globalen Vertex-Puffer und einer Basisadresse für den globalen Index-Puffer aus SSYNC-Einheit
706
Schreiben der Vertices aus lokalem Vertex-Puffer in globalen Vertex-Puffer beginnend bei der Basisadresse für den globalen Vertex-Puffer
708
Schreiben von Indizes vom lokalen Index-Puffer in den globalen Index-Puffer auf der Grundlage der Basisadresse für den globalen Vertex-Puffer und beginnend bei Basisadresse in dem globalen Index-Puffer
708
Ende

FIG. 7

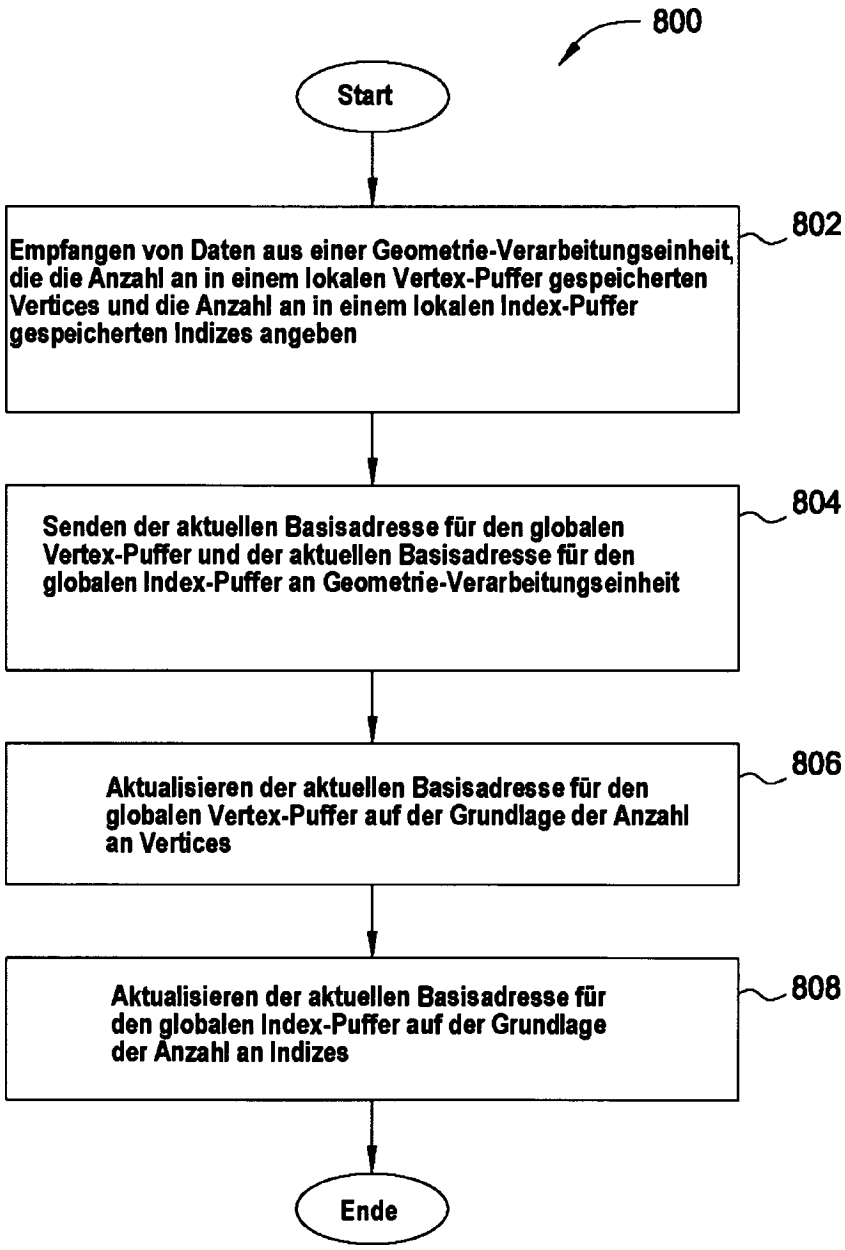

800
Start
Empfangen von Daten aus einer Geometrie-Verarbeitungseinheit, die die Anzahl an in einem lokalen Vertex-Puffer gespeicherten Vertices und die Anzahl an in einem lokalen Index-Puffer gespeicherten Indizes angeben
802
Senden der aktuellen Basisadresse für den globalen Vertex-Puffer und der aktuellen Basisadresse für den globalen Index-Puffer an Geometrie-Verarbeitungseinheit
804
Aktualisieren der aktuellen Basisadresse für den globalen Vertex-Puffer auf der Grundlage der Anzahl an Vertices
806
Aktualisieren der aktuellen Basisadresse für den globalen Index-Puffer auf der Grundlage der Anzahl an Indizes
808
Ende

FIG. 8

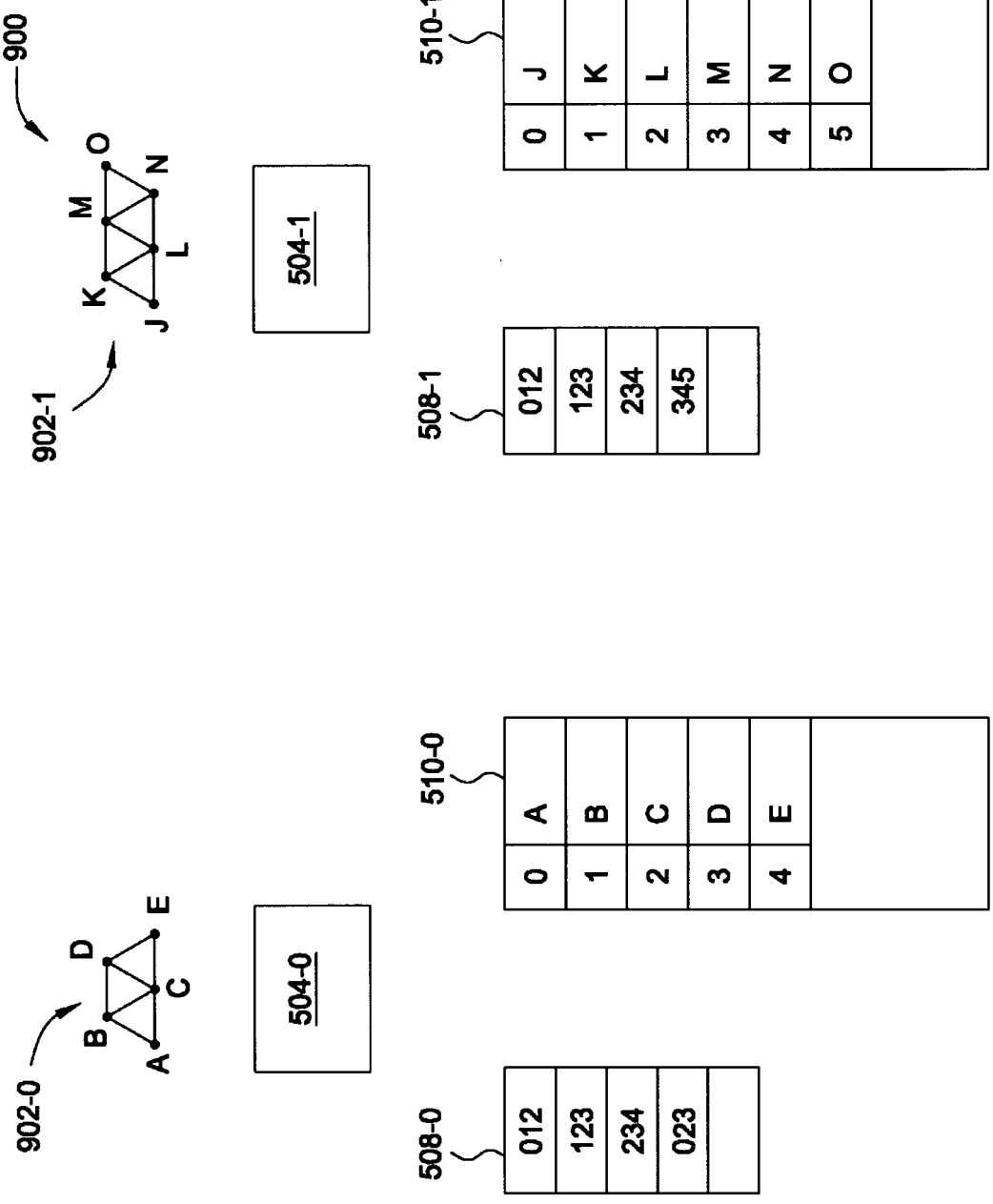

900
902-1
K
M
O
J
L
N
504-1
508-1
012
123
234
345
510-1
0 J
1 K
2 L
3 M
4 N
5 O
902-0
B
D
E
A
C
504-0
508-0
012
123
234
023
510-0
0 A
1 B
2 C
3 D
4 E

FIG. 9

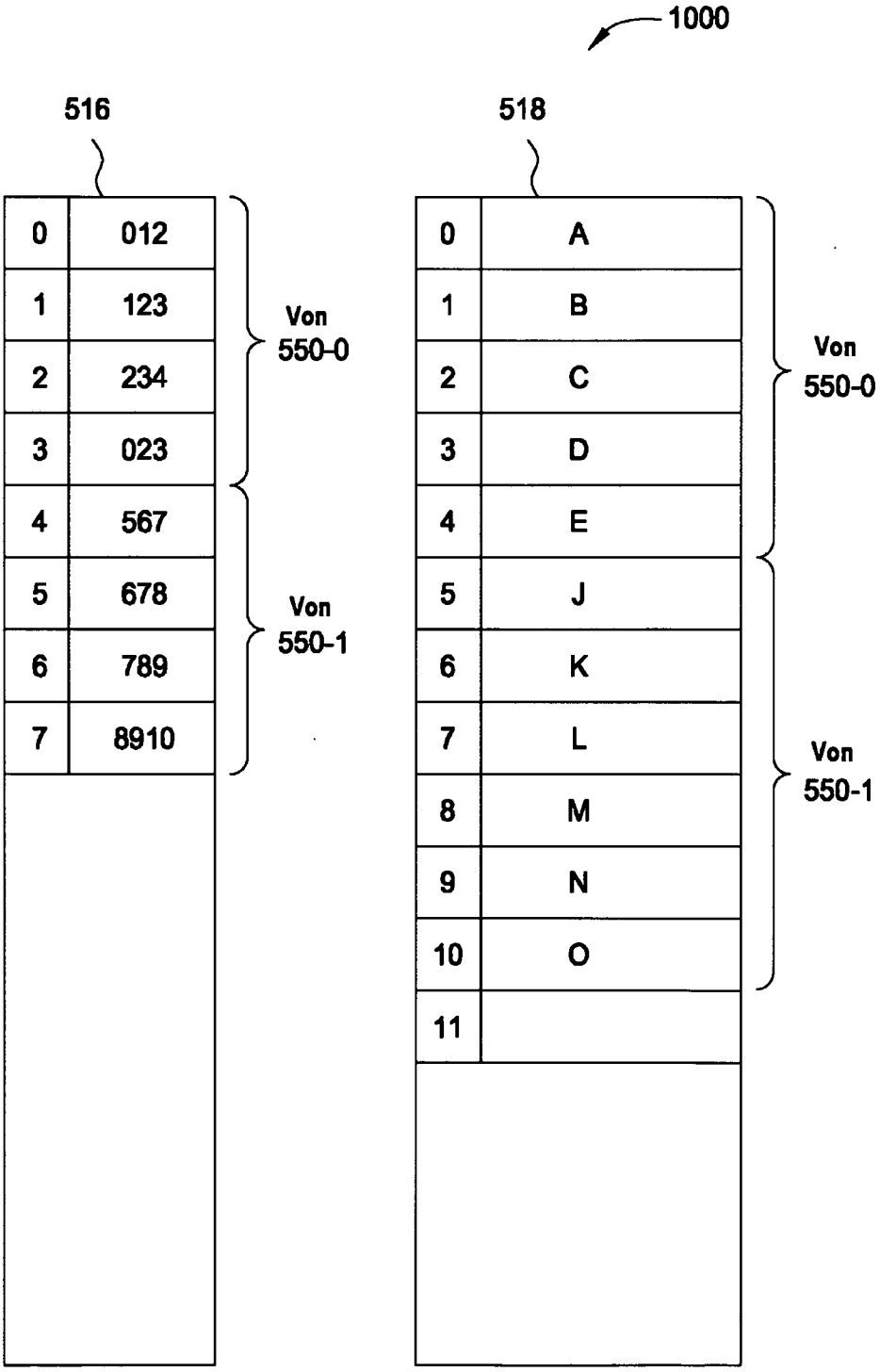

1000
516
518
0
012
1
123
2
234
3
023
4
567
5
678
6
789
7
8910
Von
550-0
Von
550-1
0
A
1
B
2
C
3
D
4
E
5
J
6
K
7
L
8
M
9
N
10
O
11
Von
550-0
Von
550-1

FIG. 10