(19) Bundesrepublik Deutschland
Deutsches Patent- und Markenamt

(10) **DE 198 27 726 B4** 2005.02.24

(12) **Patentschrift**

(21) Aktenzeichen: **198 27 726.1**
(22) Anmeldetag: **22.06.1998**
(43) Offenlegungstag: **07.01.1999**
(45) Veröffentlichungstag der Patenterteilung: **24.02.2005**

(51) Int Cl.[7]: **G06T 15/00**
G06T 5/00

Innerhalb von 3 Monaten nach Veröffentlichung der Erteilung kann Einspruch erhoben werden.

(30) Unionspriorität:
887441 02.07.1997 US

(71) Patentinhaber:
Hewlett-Packard Co. (n.d.Ges.d.Staates Delaware), Palo Alto, Calif., US

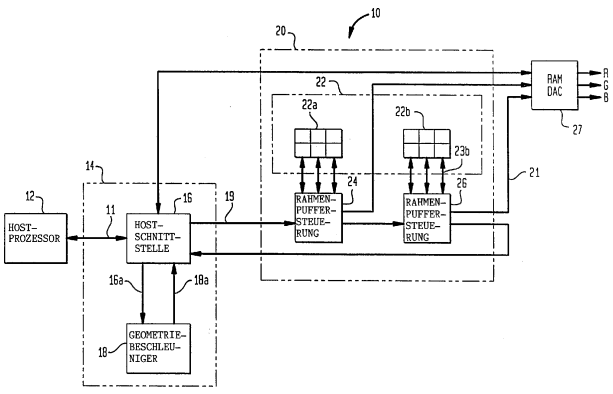
(74) Vertreter:
Schoppe, Zimmermann, Stöckeler & Zinkler, 82049 Pullach

(72) Erfinder:
Barkans, Anthony C., Redmond, Wash., US

(56) Für die Beurteilung der Patentfähigkeit in Betracht gezogene Druckschriften:
US 51 23 085 A
EP 04 30 501 A2
BARKANS, A.C.: Hardware-Assisted Polygon Antialiasing IEEE Computer Graphics and Applications, January 1991, S. 80-88;
FOLEY, J.D.; et al.: Computer Graphics-Principles and Practice, Addison-Wesley, 1996, S. 882-887;

(54) Bezeichnung: **Verfahren und Vorrichtung zum Liefern von Polygonpixelunterabtastinformationen unter Verwendung einer Inkrementaleinrichtung**

(57) Hauptanspruch: Verfahren zum Bestimmen eines Prozentsatzes mindestens eines Pixels (73) auf einem Graphiksystemanzeigebildschirm, das durch ein Primitiv (74) abgedeckt ist, das auf dem Anzeigebildschirm angezeigt werden soll, wobei das Verfahren folgende Schritte aufweist:
(a) Aufteilen des mindestens einen Pixels in eine Mehrzahl von Unterabtastzeilen, die eine Mehrzahl von Unterabtastpunkten für das mindestens eine Pixel definieren, wobei jeder der Mehrzahl von Unterabtastpunkten auf einer entsprechenden der Mehrzahl von Unterabtastzeilen positioniert ist;
(b) Schreiten zwischen der Mehrzahl von Unterabtastzeilen des mindestens einen Pixels (73);
(c) Bestimmen bei jeder der Mehrzahl von Unterabtastzeilen, ob der entsprechende Unterabtastpunkt auf der einen der Mehrzahl von Unterabtastzeilen durch das Primitiv (74) abgedeckt ist; und
(d) Nähern des Prozentsatzes des mindestens einen Pixels, das durch das Primitiv (74) abgedeckt ist, basierend auf der Anzahl der Mehrzahl von Unterabtastpunkten des mindestens einen Pixels (73), die durch das Primitiv (74) abgedeckt wird.



Beschreibung

[0001] Diese Erfindung bezieht sich allgemein auf das Gebiet von Computersystemen und insbesondere auf ein Verfahren und eine Vorrichtung für das Antialiasing (Filterung zur Glättung von Kurven in Rasterbildern) bei Computergraphiksystemen.

[0002] Computergraphiksysteme werden häufig zum Anzeigen von graphischen Darstellungen von zwei- oder dreidimensionalen Objekten auf einem zweidimensionalen Anzeigebildschirm verwendet. Ein herkömmliches Computergraphiksystem umfaßt einen Hostcomputer zum Liefern von Anweisungen und Daten zu einem Graphikgerät, das den Anzeigebildschirm umfaßt. Durch Weiterleiten von Befehlen und Daten zu dem Graphikgerät steuert der Hostcomputer die Anzeige von Bildern auf dem Anzeigebildschirm.

[0003] Bei typischen Computergraphiksystemen wird ein Objekt, das auf einem Anzeigebildschirm dargestellt werden soll, in eine Anzahl von Primitiven zerlegt. Die Primitive sind Grundkomponenten eines Graphikbilds, wie z. B. Punkte, Linien, Vektoren und Polygone (z. B. Dreiecke). Typischerweise arbeitet eine Graphiksoftware, die auf dem Hostcomputer ausgeführt wird, in Verbindung mit dem Graphikgerät, um die Graphikprimitive auf dem Anzeigegerät wiederzugeben (zu zeichnen).

[0004] Um ein zwei- oder dreidimensionales Objekt wiederzugeben, liefert der Hostcomputer eine Sammlung von Primitivdaten, die zusammen das Objekt, das wiedergegeben werden soll, definieren. Beispielsweise können die Primitivdaten, die ein Dreieck definieren, die x-, y-, z-Koordinaten des dreidimensionalen Objektraums und die Rot-, Grün-, Blau- (RGB-)Farbwerte für jeden Scheitelpunkt des Dreiecks umfassen. Die Wiedergabehardware in dem Graphikgerät interpoliert die Primitivdaten, um zu bestimmen, welche Pixel auf dem Anzeigebildschirm verwendet werden, um das Dreieck wiederzugeben, und um die Rot-, Grün- und Blau-Farbwerte für jedes dieser Pixel zu bestimmen.

[0005] Der Prozeß des Umwandelns von Primitivdaten, z. B. einer Linie, einem Punkt oder einem Polygon, zu einem entsprechenden Array von Pixeldaten ist als "Abtastumwandlung" bekannt. Der Abtastumwandlungsprozeß, der bei einem Graphiksystem verwendet wird, wird ziemlich häufig aufgerufen, typischerweise Tausende von Malen für jedes Bild, das erzeugt oder modifiziert wird. Daher ist es wünschenswert, daß der Abtastumwandlungsprozeß so schnell wie möglich ausgeführt wird. Die Ausführgeschwindigkeit erfordert jedoch oftmals einen Kompromiß bezüglich der Bildqualität.

[0006] Einige Abtastumwandlungsverfahren sind schnell. Die Primitive, die wiedergegeben werden, erscheinen jedoch derart, als wenn dieselben gezackte Kanten aufweisen. Das Erscheinungsbild der gezackten Kanten resultiert aus der Auswahl einer nicht optimalen Farbe für die Pixel an den Kanten des Primitivs. Die Primitive, die gezackte Kanten als ein Resultat ihrer Transformation in Pixeldaten aufweisen, werden häufig als "aliased" (= mit auflösungsbedingten Darstellungsfehlern behaftet) bezeichnet. Wenn ein Abtastwandler in einer Betriebsart arbeitet, die nicht versucht, die gezackten Kanten zu korrigieren, wird davon gesprochen, daß derselbe in einer Aliasedbetriebsart arbeitet.

[0007] Beispielsweise stellt **Fig.** 1 ein Primitiv (d. h. ein Dreieck **74**) dar, das durch einen Abtastwandler, der in der Aliasedbetriebsart arbeitet, wiedergegeben wurde. Wenn ein Primitiv in einer Aliasedbetriebsart geliefert wird, wird eine schnelle Bestimmung durchgeführt, ob jedes Pixel ein (d. h. durch das Primitiv abgedeckt) oder aus (nicht durch das Primitiv abgedeckt) ist oder nicht. Die Grundbestimmung wird durch Auswerten des Zustands jedes Pixels bei einem Ursprungspunkt, wie z. B. ein Punkt **1** in **Fig.** 1, bestimmt. Die Adresse oder der Ursprung jedes Pixels in dem x,y-Koordinatenraum ist durch die Koordinaten an den x-y-Schnittpunkten des Gitters in der oberen rechten Ecke des Pixels definiert. Wenn der Ursprungspunkt durch das Primitiv abgedeckt ist, ist das Pixel ein. Wenn nicht, ist das Pixel aus. Da lediglich eine einzige Farbabtastung pro Pixel vorgenommen wird, wird die Wiedergabe des Primitivs in der Aliasedbetriebsart schnell durchgeführt. Wie es jedoch in dem Beispiel von **Fig.** 1 sichtbar ist, ist, obwohl die Aliasedprimitive oftmals schnell wiedergegeben werden können, das resultierende Primitiv oftmals von einer schlechten Qualität.

[0008] Auf den Prozeß des Lieferns eines Primitivs mit einem Erscheinungsbild mit glatten Kanten wird allgemein als "Antialiasing" Bezug genommen. Die Abtastumwandlungsalgorithmen, die in einem Antialiasedbetrieb arbeiten, sind allgemein langsamer, oder dieselben erfordern wesentlich mehr Hardware, da mehr Berechnungen bei jedem Pixel durchgeführt werden, um eine geeignete Farbe auszuwählen. Primitive, die unter Verwendung des Antialiasing wiedergegeben werden, weisen jedoch glattere Kanten als ein Resultat von optimaleren Farbauswahlkriterien auf. Ein Verfahren des Antialiasing verwendet ein Mehrpunktabtasten innerhalb jedes Pixels, um eine optimalere Farbe für Pixel an den Kanten eines Primitivs auszuwählen. Beim Aus-

wählen einer optimaleren Farbe mischt das Mehrpunktabtasten die Farbe des Hintergrunds, einschließlich jeder unterliegenden Geometrie, mit der Farbe des Primitivs, um gezackte Kanten des Primitivs zu glätten. Bei dem Mehrpunktabtasten wird jedes Pixel allgemein bei mehreren Unterpixelpositionen innerhalb jedes Pixels abgetastet, um den Abschnitt des Pixels zu bestimmen, der durch das Primitiv abgedeckt wird. Eine einzige gemischte Farbe für das Pixel wird dann basierend auf der Anzahl der Unterpixel, die durch das Primitiv abgedeckt werden, bestimmt.

[0009] Beispielsweise kann, wenn eine Kante eines roten Primitivs einen schwarzen Hintergrund bei einem gegebenen Pixel abdeckt, derart, daß 50% der Unterpixelabtastpositionen abgedeckt sind, dann die Farbe für das Pixel als eine Mischung von 50% Rot und 50% Schwarz ausgewählt werden. Folglich wird durch Mischen der Farbe der Pixel entlang der Kanten eines Primitivs ein glatter Übergang zwischen dem Primitiv und dem Hintergrund geschaffen.

[0010] Eine Technik, die zum Liefern von Antialiasedbildern unter Verwendung des Mehrpunktabtastens verwendet wird, ist in dem REALITYENGINE-Graphikgerät (REALITYENGINE ist eine eingetragene Marke) vorgesehen, das kommerziell durch die Silicon Graphics Computer Systems, U.S.A., erhältlich ist. Das Silicon-Graphics-Gerät empfängt Ebenengleichungen, die Primitive darstellen, und dasselbe löst nach den einzelnen Parametern jeder Ebenengleichung auf. Die Ebenengleichungsparameter werden verwendet, um auf eine Position in einer Nachschlagtabelle zuzugreifen, die Strukturen von Unterpixeln für jede mögliche Kombination von Parameterwerten speichert. Die Struktur von Unterpixeln in der Nachschlagtabelle wird verwendet, um die Unterpixel zu identifizieren, die durch das Primitiv abgedeckt werden, das durch die Ebenengleichung bei jedem Pixel dargestellt ist.

[0011] Die mathematischen Berechnungen, die verwendet werden, um die Ebenengleichungen zu lösen, sind komplex. Um die Ebenengleichungsparameter mit hoher Leistung zu lösen, sind viele Pixelprozessoren erforderlich. Da die mathematischen Berechnungen und das Nachschlagen mit Hardware durchgeführt werden, ist das Silicon-Graphics-System in der Lage Unterpixeldaten relativ schnell zu liefern. Ein Nachteil des obigen Verfahrens besteht jedoch darin, daß die Hardware, die erforderlich ist, um die mathematischen Funktionen durchzuführen, hinsichtlich der Toranzahl teuer ist, was folglich die Kosten und die Größe der Graphikschaltungsanordnung erhöht.

Stand der Technik

[0012] Eine herkömmliche Antialiasing-Technik ist bei BARKANS, A.C.: Hardware-Assisted Polygon Antialiasing. IEEE Computer Graphics and Applications, January 1991, Seite 80–88, der einen Überabtasten-Lösungsansatz für ein Anti-Aliasing beschreibt. Ein Überabtasten ist eine Technik, bei der das Bild, das wiedergegeben werden soll, in ein größeres Bild skaliert und dann für eine Wiedergabe auf dem Anzeigebildschirm größenmäßig reduziert wird. Während der Bildreduzierung werden Pixelfarbwerte von einem oder mehreren benachbarten Pixeln der übergroßen Bilder interpoliert („gemittelt" bid BARKANS). Bekannte Interpolationstechniken umfassen beispielsweise Interpolationstechniken des nächsten Nachbarn, eine bilineare Interpolation und eine bikubische Interpolation. Im Gegensatz zu dem Abtasten von mehreren Punkten in einem einzelnen Pixel vergrößert der Überabtast-Lösungsansatz für ein Anti-Aliasing ein Bild, um mehr Pixel zu erzeugen, von denen nachfolgend Gruppen kombiniert werden, um einen Mittelwert für jedes Pixel in dem resultierenden Bild zu bestimmen.

[0013] Die EP 0430 501 A2 offenbart ferner einen Überabtast-Lösungsansatz für ein Antialiasing, der in dieser Schrift als „Supersampling" bezeichnet wird. Dieser Überabtast-Lösungsansatz für Anti-Aliasing-Polygonkanten verwendet einen einzelnen Abtastwert pro Pixel und addressiert keine Mehrpunkt-Abtastung von Pixeln zum Durchführen eines Antialiasing.

[0014] Bei FOLEY, J.D.; et al.: Computer Graphics- Principles and Practice. Addison-Wesley, 1996, S. 882–887, ist eine herkömmliche Kenten-Schritttechnik beschrieben, d.h. von einer Abtastlinie zur nächsten und von einem Pixel zum nächsten in jeder Abtastlinie. Ein Abtasten einer Mehrzahl von Punkten entlang einer Mehrzahl von Unterabtastlinien für jedes Pixel ist dieser Schrift nicht zu entnehmen.

[0015] Schließlich ist aus der US 5 123 085 A ein Abtastumwandlungsverfahren bekannt, das auf einem Polygon unter Verwendung einer Einzeldurchlauftechnik durchgeführt wird. Die Pixel, die Kanten und Scheitelpunkte des Polygons umfassen, werden zunächst auf den Scheitelpunkten, die das Polygon definieren, bestimmt. Der Alphakanal umfasst entweder eine Unterpixelmaske, die jedem Pixel zugeordnet ist, die den Betrag der abgedeckten Unterpixelregionen anzeigt, oder einen einzelnen Wert, der den Prozentsatz der Abde-

ckung eines Pixels anzeigt. Ferner wird ein z-Wert, der die Tiefe jedes Pixels anzeigt, beibehalten. Die Pixel zwischen den Kantenpixeln des Polygons werden dann eingeschaltet, wodurch das Polygon gefüllt wird. Die Pixel, die das Polygon aufweist, werden mit den Hintergrundpixeln auf einer Pixel-um-Pixel-Basis zusammengesetzt. Der Tiefenwert jedes Pixels des Polygons (der z-Wert) wird verwendet, um die Zusammensetzungsgleichungen zu bestimmen, die verwendet werden sollen, um jedes Pixel des Polygons mit dem Hintergrund zusammenzusetzen. Die Zusammensetzungsgleichungen aktualisieren die Farbe des Pixels, den z-Pufferwert des Hintergrundpixels und die Unterpixelmaske, um die Hinzufügung von Informationen von der Zusammensetzungsoperation widerzuspiegeln. Durch dieses Verfahren können einem Anti-Aliasing unterworfene Polygone hoher Qualität aufbereitet werden, um die zeitaufwendige Verarbeitung des Sortierens der Polygone in der Tiefenreihenfolge vor dem Zusammensetzen durchzuführen. Diese Schrift offenbart nicht ein Antialiasing eines Grundelements durch das Abtasten von Punkten entlang einer Mehrzahl von Unterabtastlinien für jedes Pixel des Grundelements.

Aufgabenstellung

[0016] Die Aufgabe der vorliegenden Erfindung besteht darin, ein verbessertes Verfahren und eine verbesserte Vorrichtung zum Bestimmen der Unterpixelabdeckung eines Primitivs zur Verwendung bei Antialiasingoperationen zu schaffen.

[0017] Diese Aufgabe wird durch ein Verfahren zum Bestimmen eines Prozentsatzes mindestens eines Pixels auf einem Graphiksystemanzeigebildschirm, das durch ein Primitiv abgedeckt ist, das auf dem Anzeigebildschirm angezeigt werden soll, gemäß Anspruch 1 und durch eine Kantenschrittvorrichtung zum Erzeugen von Koordinatendaten zum Anzeigen von Pixeln auf einem Anzeigegerät, die das Primitiv darstellen, in einem Graphiksystem, das ein Primitiv wiedergeben kann, gemäß Anspruch 6 gelöst.

[0018] Gemäß einem Aspekt der Erfindung ist ein Verfahren zum Nähern eines Prozentsatzes mindestens eines Pixels auf einem Graphiksystemanzeigebildschirm, das durch ein Primitiv abgedeckt ist, das auf dem Anzeigebildschirm angezeigt werden soll, geschaffen. Das Verfahren umfaßt die Schritte des Aufteilens des mindestens einen Anzeigepixels in eine Mehrzahl von Unterabtastzeilen, die eine Mehrzahl von Unterabtastpunkten für das mindestens eine Pixel definieren, wobei jeder der Mehrzahl von Unterabtastpunkten auf einer entsprechenden der Mehrzahl von Unterabtastzeilen positioniert ist. Das Abschreiten wird zwischen der Mehrzahl von Unterabtastzeilen des mindestens einen Pixels durchgeführt. Bei jeder der Mehrzahl von Unterabtastzeilen wird bestimmt, ob der entsprechende Unterabtastpunkt auf der einen der Mehrzahl von Unterabtastzeilen durch das Primitiv abgedeckt wird. Der Prozentsatz des mindestens einen Pixels, der durch das Primitiv abgedeckt ist, wird basierend auf der Anzahl der Mehrzahl von Unterabtastpunkten des mindestens einen Pixels, die durch das Primitiv abgedeckt sind, genähert.

[0019] Gemäß einem weiteren Aspekt der Erfindung ist bei einem Graphiksystem, das in einer Aliasedbetriebsart zum Wiedergeben von Primitiven und in einer Antialiasedbetriebsart zum Wiedergeben von Primitiven mit relativ glatteren Kanten arbeiten kann, eine Kantenschrittvorrichtung zum Erzeugen von Koordinatendaten für Pixel auf einem Anzeigegerät, die ein Primitiv auf dem Anzeigegerät darstellen, geschaffen. Die Kantenschrittvorrichtung empfängt Primitivdaten, die ein erstes und zweites Paar von Kantenkoordinaten und einen Neigungswert für mindestens eine Kante des Primitivs, das auf der Anzeige wiedergegeben werden soll, umfassen. Die Kantenschrittvorrichtung umfaßt einen Addierer zum Inkrementieren mindestens der ersten Endpunktkoordinaten durch einen Inkrementwert, um eine Kanten-x- und eine Kanten-y-Koordinate der mindestens eine Kante des wiedergegebenen Primitivs zu liefern, wobei der Inkrementwert kleiner als eine Größe eines Pixels ist.

[0020] Gemäß einem weiteren Aspekt der Erfindung ist ein Graphikgerät zum Anzeigen eines Bilds auf einem Anzeigegerät, das mit einem Hostprozessor gekoppelt ist, geschaffen, wobei das Anzeigegerät eine Mehrzahl von Abtastzeilen aufweist, und wobei jede der Abtastzeilen eine Mehrzahl von Pixeln aufweist. Das Graphikgerät umfaßt einen Graphikbeschleuniger, der gekoppelt ist, um Daten von dem Hostprozessor zu empfangen, die eine Mehrzahl von Graphikprimitiven darstellen. Der Graphikbeschleuniger erzeugt Kantengleichungen für jedes Primitiv, wobei die Kantengleichungen für mindestens eine Kante des Primitivs Endpunktkoordinatendaten, die Endpunkte der mindestens einen Kante des Primitivs identifizieren, und Neigungsdaten umfassen, die eine Änderungsrate der mindestens einen Kante identifizieren. Zusätzlich umfaßt das Graphikgerät einen Abtastumwandler, der gekoppelt ist, um die Kantengleichungen von dem Graphikbeschleuniger zu empfangen, und um die Ebenengleichungen in Koordinaten von Pixeln des Anzeigegeräts zu übersetzen. Der Abtastumwandler umfaßt eine Kantenschrittvorrichtung, um von einer der Koordinaten eines der Endpunkte der Kante zu einem anderen Endpunkt der Kante zu schreiten, um Pixelkoordinaten für die Kante zu erzeugen, wobei

eine Schrittgröße der Kantenschrittvorrichtung kleiner ist als die Größe eines Anzeigepixels.

Ausführungsbeispiel

[0021] Bevorzugte Ausführungsbeispiele der vorliegenden Erfindung werden nachfolgend unter Bezugnahme auf die beigefügten Zeichnungen näher erläutert. Es zeigen:

[0022] **Fig.** 1 ein Beispiel eines Aliasedwiedergebens eines Primitivs;

[0023] **Fig.** 2 ein Blockdiagramm eines darstellenden Graphiksystems, bei dem die vorliegende Erfindung verwendet werden kann;

[0024] **Fig.** 3 ein Blockdiagramm eines Graphikbeschleunigers zur Verwendung bei dem Graphiksystem von **Fig.** 2;

[0025] **Fig.** 4 ein Beispiel einer Transformierungsmatrix, die zum Skalieren von Primitivdaten bei einem Ausführungsbeispiel der vorliegenden Erfindung verwendet wird;

[0026] **Fig.** 4A ein Beispiel von Kantengleichungen, die durch den Graphikbeschleuniger von **Fig.** 2 erzeugt werden;

[0027] **Fig.** 5 ein Blockdiagramm eines Ausführungsbeispiels einer Rahmenpuffersteuerung eines Graphiksystems, wie dasselbe, das in **Fig.** 2 gezeigt ist, das die vorliegende Erfindung verwendet;

[0028] **Fig.** 6 eine Darstellung eines Primitivs, das auf einem Pixelgitter gezeichnet ist;

[0029] **Fig.** 7 ein Blockdiagramm eines Ausführungsbeispiels einer Kantenschrittvorrichtungseinheit der Rahmenpuffersteuerung von **Fig.** 5 gemäß der vorliegenden Erfindung;

[0030] **Fig.** 8 ein darstellendes Beispiel des Skalierens von Pixeln, das bei einem Ausführungsbeispiel der vorliegenden Erfindung zum Bestimmen von Abtastpunkten während des Mehrpunktabtastens verwendet wird;

[0031] **Fig.** 9 eine darstellende Zuweisung von x- und y-Koordinaten-Adreßbits während des Aliased- und Antialiased-Betriebs gemäß der vorliegenden Erfindung;

[0032] **Fig.** 10 ein Blockdiagramm eines Ausführungsbeispiels einer Sammeleinheit, die zum Sammeln von Mehrpunktabtastungen gemäß der vorliegenden Erfindung verwendet wird;

[0033] **Fig.** 11 eine Darstellung eines Primitivs, das auf einem Pixelgitter zum Erklären des Betriebs der Sammellogik von **Fig.** 10 gezeichnet ist;

[0034] **Fig.** 12 ein Flußdiagramm, das ein Betriebsverfahren einer Datenwegssammeleinheit der Sammellogik von **Fig.** 10 darstellt;

[0035] **Fig.** 13 ein Blockdiagramm eines Ausführungsbeispiels einer Datenwegweiterleitungseinheit der Sammellogik von **Fig.** 10;

[0036] **Fig.** 14 ein Flußdiagramm, das ein Verfahren zum Betreiben der Lesesteuerlogik der Sammeleinheit von **Fig.** 10 darstellt, um die Unterabtastzeilen zu einer Weitenschrittvorrichtung gemäß der vorliegenden Erfindung weiterzuleiten;

[0037] **Fig.** 15 ein Flußdiagramm, das ein Verfahren zum Weiterleiten von Unterabtastzeilen bei der Lesesteuerlogik von **Fig.** 14 darstellt; und

[0038] **Fig.** 16 ein Blockdiagramm eines Ausführungsbeispiels einer Weitenschrittvorrichtungseinheit, die in der Rahmenpuffersteuerung von **Fig.** 5 gemäß der vorliegenden Erfindung verwendet wird.

[0039] Die vorliegende Erfindung ist auf ein verbessertes Verfahren und eine verbesserte Vorrichtung für das Antialiasing von Primitiven in einem Graphikgerät gerichtet, das mit einem Anzeigebildschirm gekoppelt ist, auf

dem ein Bild wiedergegeben wird. Der Anzeigebildschirm umfaßt eine Anzahl von Abtastzeilen, wobei jede Abtastzeile eine Anzahl von Anzeigepixeln umfaßt. Das Graphikgerät umfaßt Software- und Hardware-Werkzeuge zum Zeichnen oder Wiedergeben eines Primitivs auf der Anzeige. Die Graphikwerkzeuge erzeugen eine Startkantenkoordinate und Neigungsinformationen für jedes Primitiv. Bei der Aliasedbetriebsart wird ein Primitiv durch Abschreiten entlang der Anzeigebildschirmabtastzeilen und Identifizieren von Koordinaten von Anzeigepixeln entlang der Kanten des Primitivs wiedergegeben. Um ein Antialiasing eines Primitivs unter Verwendung eines Mehrpunktabtastens durchzuführen, wird bei einem Ausführungsbeispiel der vorliegenden Erfindung ein Bild, das eine Mehrzahl von Abtastzeilen aufweist, in eine Mehrzahl von Unterabtastzeilen unterteilt, derart, daß eine Anzahl von Unterabtastzeilen jedes Pixel kreuzt. Ein Abtastpunkt wird auf jeder Unterabtastzeile plaziert, derart, daß jedes Pixel mehrere Abtastpunkte umfaßt, die verwendet werden können, um zu bestimmen, wie sehr das Anzeigepixel durch das Primitiv abgedeckt ist. Um die Anzahl der Abtastungen zu bestimmen, die durch ein Primitiv abgedeckt sind, wird eine Kantenschrittechnik verwendet, um entlang jeder Kante des Primitivs zu schreiten, was eine Unterabtastzeilenkoordinate für die Kante erzeugt. Sowie jede Kantenunterabtastzeilenkoordinate erzeugt wird, wird dieselbe mit Koordinaten eines Abtastpunkts auf dieser Unterabtastzeile verglichen, um zu bestimmen, ob dieser Abtastpunkt durch das Primitiv abgedeckt ist oder nicht. Eine Maske wird für jedes Pixel erzeugt, wobei jedes Bit der Maske einem der Unterabtastzeilenabtastungen entspricht und anzeigt, ob die entsprechende Unterabtastung durch das Primitiv abgedeckt ist. Als Resultat kann unter Verwendung eines inkrementalen Algorithmusses, bei dem Schrittinkremente und Richtungen auf Parametern basieren, die in der Software vorerzeugt werden, die Position der Pixelkanten zum Auswerten der Pixelabdeckwerte bestimmt werden, ohne daß die komplexe Hardware erforderlich ist, die im Stand der Technik verwendet wird.

[0040] Die Konzepte der vorliegenden Erfindung werden nun unter Bezugnahme auf die angehefteten Zeichnungen beschrieben, in denen sich gleiche Ziffern auf gleiche Elemente in den Zeichnungen beziehen. **Fig.** 2 ist ein Blockdiagramm, das ein exemplarisches Computergraphiksystem **10** zeigt, bei dem das Antialiasingverfahren und die Antialiasingvorrichtung der vorliegenden Erfindung verwendet werden können. Das Computergraphiksystem **10** umfaßt einen Hostprozessor **12**, ein vorgelagertes Untersystem **14** und ein Rahmenpufferuntersystem **20**. Das vorgelagerte Untersystem **14** empfängt Primitivdaten von dem Hostprozessor **12** über den Bus **11**. Wie oben erörtert, wird jedes Primitiv typischerweise durch x-, y-, z-Koordinatendaten, R-, G-, B-Farbdaten und Texturdaten für gewisse Abschnitte des Primitivs spezifiziert, wie z. B. Scheitelpunkte, wenn das Primitiv ein Dreieck ist. Das vorgelagerte Untersystem **14** transformiert die Primitivdaten in Kantengleichungen für jedes Primitiv, die Koordinaten für jeden der Scheitelpunkte des Primitivs und die Neigungen der Kanten des Primitivs identifizieren, und dieselbe überträgt diese Kantengleichungen zu dem Rahmenpufferuntersystem **20**.

[0041] Das Rahmenpufferuntersystem **20** interpoliert die Kantengleichungsdaten, die von dem vorgelagerten Untersystem **14** für jedes Primitiv empfangen werden, um die Pixeladressen in Anzeigebildschirmkoordinaten und Pixelfarbdaten zu berechnen, die verwendet werden, um jedes Primitiv auf dem Anzeigebildschirm darzustellen. Die Pixelfarbdaten umfassen Bits, die Rot-, Grün- und Blau-Farbkomponenten der Pixel identifizieren, und dieselben werden in einem Rahmenpufferspeicher **22** gespeichert. Die Pixelfarbdaten werden hinaus aus dem Rahmenpufferspeicher **22** und durch einen Direktzugriffsspeicher-Digital-zu-Analog-Wandler (RAMDAC; RAMDAC = Random Access Memory Digital to Analog Converter) **27** weitergeleitet, der die binären Farbdaten, die in dem Rahmenpufferspeicher **22** gespeichert sind, in analoge Rot-, Grün- und Blau-Signale zur Anzeige auf einem Anzeigegerät (nicht gezeigt) umwandelt. Es sollte offensichtlich sein, daß die Erfindung nicht auf ein analoges Anzeigegerät begrenzt ist, und daß jeder andere Typ eines Anzeigegeräts (z. B. digital, monochrom oder graustufig) alternativ verwendet werden kann.

[0042] Bei dem darstellenden System von **Fig.** 2 umfaßt das vorgelagerte Untersystem **14** eine Hostschnittstelle **16** und einen dreidimensionalen (3-D-)Geometriebeschleuniger **18**. Die Hostschnittstelle **16** empfängt die x-, y-, z-Scheitelpunktkoordinate und die Farbprimitivdaten über den Bus **11** von dem Hostprozessor **12**. Diese Daten werden dann von der Hostschnittstelle **16** zu dem Geometriebeschleuniger **18** über den Bus **11** geliefert. Der Geometriebeschleuniger **18** führt herkömmliche Geometriebeschleunigerfunktionen, wie z. B. eine dreidimensionale Transformation, die Beleuchtung, das Abschneiden und perspektivische Teiloperationen sowie das Erzeugen von Kantengleichungen im Gleitkommaformat durch. Bei Systemen, bei denen ein Geometriebeschleuniger nicht vorgesehen ist, werden die Kantengleichungen alternativ durch den Hostprozessor geliefert. Bei dem Ausführungsbeispiel von **Fig.** 2 werden die Kantengleichungen durch den Geometriebeschleuniger **18** über den Bus **18a** zu der Hostschnittstelle **16** geliefert, die diese Kantengleichungsdaten auf dem Bus **19** zu dem Rahmenpufferuntersystem **20** weiterleitet.

[0043] Unter Bezugnahme nun auf **Fig.** 3 ist ein Blockdiagramm eines Abschnitts des Geometriebeschleuni-

gers **18** als eine Transformierungseinheit **30**, eine Kantenerzeugungseinheit **34** und eine Formatumwandlungseinheit **36** umfassend gezeigt. Die Transformierungseinheit **30** empfängt Koordinaten- und Farb-Daten für jeden Endpunkt eines Primitivs. Ein Dreiecksprimitiv wird beispielsweise durch drei Scheitelpunkte dargestellt, und dasselbe wird folglich durch drei Scheitelpunktkoordinaten dargestellt. Die Transformierungseinheit **30** umfaßt eine Transformierungsmatrix **32**, die verwendet wird, um geometrisch die Eingabekoordinaten von Primitivscheitelpunkten in Koordinaten zum Darstellen des Primitivs auf einem Anzeigegerät zu transformieren. Kurz bezugnehmend auf **Fig.** 4 ist ein Beispiel einer Transformierungsmatrix **32** als eine Anzahl von Zeilen und Spalten von Matrixwerten a1–a4 ... d1–d4 umfassend gezeigt. Die Matrixwerte werden verwendet, um Koordinaten [x, y, z, w] jedes Pixelwerts in Koordinaten [x', y', z', w'] eines Anzeigegeräts zu transformieren, wobei x, y die Pixelkoordinaten entlang der x- bzw. der y-Achse sind, wobei z die Pixelkoordinate entlang der z-Achse ist und folglich eine visuelle Tiefeninformation über das Pixel liefert, und wobei w die Pixelperspektive, d. h. die Größe des Primitivs relativ zu der Größe der Anzeige ist. Die Matrix ist geräteabhängig, und folglich variieren die Werte in der Matrix basierend auf der Anzahl der Pixel, die auf dem Anzeigegerät angezeigt werden kann. Zusätzlich kann die Transformierungsmatrix **32** verwendet werden, um ein Primitiv durch einen Skalierungswert S durch Multiplizieren der geeigneten Matrixeinträge a1–a4, b1–b4, c1–c4 und d1–d4 mit S unter Verwendung von Standardmatrixmultiplikationstechniken zu skalieren. Diese Fähigkeit wird bei vielen Graphiksystemen vorgesehen, um Bilder zum Vorsehen einer Zoomfähigkeit zu skalieren.

[0044] Sobald die Objektkoordinaten des Primitivs zu Koordinaten für das Anzeigegerät transformiert wurden, werden die transformierten Primitivdaten zu der Kantenerzeugungseinheit **34** (**Fig.** 3) weitergeleitet. Die Kantenerzeugungseinheit **34** wandelt die empfangenen Primitivscheitelpunktdaten in Kantengleichungen um, die das Primitiv darstellen. Bezugnehmend nun kurz auf **Fig.** 4A ist ein Beispiel von einigen Kantengleichungen gezeigt, die für ein Dreieck erzeugt wurden. Die Scheitelpunktkoordinaten, die durch die Transformierungseinheit **30** von der Hostschnittstelle **16** empfangen werden, sind noch nicht skaliert, um die Perspektive des Primitivs zu berücksichtigen, d. h. die Größe des Primitivs relativ zu der Größe der Anzeige. Folglich werden die Gleichungen in einer Gruppe **40** verwendet, um die x-, y- und z-Koordinaten für jeden der Scheitelpunkte v0, v1 und v2 durch einen Perspektivbetrag "wrecip" (der gleich 1/w ist) zu skalieren. Das Skalieren durch Perspektive ermöglicht das Skalieren eines einzelnen Primitivs. Im Gegensatz dazu skaliert die Transformierungsmatrix das gesamte Bild auf der Anzeige. Die Gleichungen in der Gruppe **42** werden dann verwendet, um die Deltas der Kanten zu berechnen. Die Deltas umfassen Längeninformationen, die die Gesamtlänge der Kante identifizieren, und Farbdeltainformationen, die die Änderung der Farbe entlang der Kante identifizieren. Die Gleichungen in einer Gruppe **44** werden verwendet, um die Neigungen der Kanten des Primitivs zu berechnen.

[0045] Während der Erzeugung der Kantengleichungen werden ferner Kantenzuweisungen durchgeführt. Beispielsweise wird, wenn Dreiecke verwendet werden, die Kante des Primitivs mit der längsten y-Achsenweite als Kante **1** ausgewählt, die Kante, die an die Kante **1** bei der niedrigsten y-Koordinate angrenzt, ist Kante **2** und die verbleibende Kante ist Kante **3**. Basierend auf den Kantenzuweisungen wird ein Richtungswert erzeugt, der eine x-Achsenrichtung anzeigt, die von Kante **1** zu Kante **2** fortfährt. Wie hierin im folgenden beschrieben, wird dieser Richtungswert bei einem Ausführungsbeispiel der vorliegenden Erfindung durch das Rahmenpuffersystem **20** zum Bestimmen von Koordinaten von Pixeln, die durch das Primitiv abgedeckt sind, verwendet.

[0046] Die Kantengleichungen und die Kantenzuweisungen werden zu der Formatumwandlungseinheit **36** weitergeleitet, die die Kantengleichungen durch Durchführen einer Gleitkomma-zu-Festkomma-Umwandlung neu formatiert, um spätere Berechnungen unter Verwendung der Gleichungen zu vereinfachen. Die obigen Kantengleichungsdaten werden aus dem Geometriebeschleuniger **18** (**Fig.** 2) heraus über den Bus **18a** zu der Hostschnittstelle **16** weitergeleitet. Die Hostschnittstelle **16** leitet dann die Kantengleichungsdaten zu dem Rahmenpufferuntersystem **20** weiter.

[0047] Bei dem Computergraphiksystem, das in **Fig.** 2 gezeigt ist, umfaßt das Rahmenpufferuntersystem **20** zwei Rahmenpuffersteuerungen **24** und **26**, von denen jede die Anzeige eines Abschnitts (**22a**, **22b**) des Rahmenpufferspeichers **22** steuert. Die Rahmenpuffersteuerung **24** steuert die Inhalte und die Anzeige von Daten in dem Abschnitt **22a** des Rahmenpufferspeichers, und die Rahmenpuffersteuerung **26** steuert die Inhalte und die Anzeige von Daten in dem Abschnitt **22b** des Rahmenpufferspeichers. Obwohl zwei Speichersteuerungen gezeigt sind, umfassen andere Ausführungsbeispiele beispielsweise eine Speichersteuerung, oder es können ferner mehr als zwei Speichersteuerungen verwendet werden. Obwohl jede Rahmenpuffersteuerung **24** und **26** Primitivdaten handhabt, die für unterschiedliche Adressen in dem Rahmenpufferspeicher **22** bestimmt sind, ist die verbleibende Funktionalität der Steuerungen im wesentlichen identisch. Folglich wird nun lediglich eine Rahmenpuffersteuerung detailliert im folgenden beschrieben.

[0048] Bezugnehmend auf **Fig.** 5 ist eine darstellende Anordnung einer Rahmenpuffersteuerung **24** gemäß einem Ausführungsbeispiel der vorliegenden Erfindung als zwei Funktionseinheiten, eine Abtastumwandlungseinheit **50** und eine Speichersteuerung **60** umfassend gezeigt. Die Abtastumwandlungseinheit **50** steuert die Übersetzung der Kantengleichungen in einzelne Pixeldaten. Die Speichersteuerung **60** steuert das Schreiben der Pixeldaten, die von der Abtastumwandlungseinheit **50** empfangen werden, in den Rahmenpufferspeicher **22**. Zusätzlich umfaßt die Speichersteuerung eine Logik **57** zum Durchführen eines Antialiasing unter Verwendung von Daten von der Abtastumwandlungseinheit **50**, die sich auf das Mehrpunktabtasten von Primitiven beziehen. Bei einem Ausführungsbeispiel der Erfindung ist die Speichersteuerung **60**, wie in der U.S.-Patentanmeldung Seriennummer 08/692,350, mit dem Titel "Antialiasing System and Method that Minimize Memory Requirements and Memory Accesses by Storinq a Reduced Set of Subsample Information", eingereicht am 5. August 1996 und hierin durch Bezugnahme aufgenommen, beschrieben, implementiert. Die Speichersteuerung, die in der obigen Patentanmeldung beschrieben ist, beschreibt ein Antialiasingverfahren, das Unterabtastabdeckungsinformationen verwendet, um die Farbe und die wahrgenommenen Tiefeninformationen (d. h. die z-Komponente) für jedes Pixel, das auf dem Anzeigegerät wiedergegeben werden soll, zu bestimmen. Folglich kann die vorliegende Erfindung verwendet werden, um die Unterabtastabdeckungsinformationen zu liefern, die bei dem Antialiasingverfahren verwendet werden, das in der '350er Patentanmeldung beschrieben ist. Das Verfahren und die Vorrichtung zum Erzeugen von Unterabtastinformationen, das hierin beschrieben ist, kann jedoch ferner mit Speichersteuerungen verwendet werden, die auf eine andere Art und Weise implementiert sind, und folglich ist die vorliegende Erfindung nicht auf die Verwendung der '350er Speichersteuerung begrenzt.

[0049] Wie oben erwähnt, leitet die Abtastumwandlungseinheit **50** Pixelkoordinate und Farbdaten zu der Speichersteuerung weiter. Gemäß einem Aspekt der Erfindung leitet die Abtastumwandlungseinheit **50** ferner eine Unterabtastsmaske zu der Speichersteuerung weiter. Jedes der Bits der Maske entspricht einer Unterabtastung eines zugeordneten Pixels. Die Unterabtastsmaske wird während des Antialiasing verwendet, um einen geeigneten Farbwert für das Pixel auszuwählen. Bevor beschrieben wird, wie die Umwandlungseinheit **50** die Unterabtastsmaske bestimmt, wird zuerst eine kurze Beschreibung des Betriebs des Mehrpunktabtastens vorgesehen.

[0050] Unter Bezugnahme nun auf **Fig.** 6 ist ein Abschnitt eines Anzeigebildschirms **70** als ein 6-Zeilen-mal-6-Spalten-Pixelgitter umfassend gezeigt. Die Zeilen des Gitters sind bei Ganzzahlwerten für sowohl x- als auch y-Koordinaten in einem x-, y-, z-Koordinatensystem der Anzeige gezeichnet. Jede der horizontalen Zeilen x stellt eine Abtastzeile für die Anzeige dar. Die Abtastzeile 0 befindet sich bei der Adresse (0, 0), die Abtastzeile 1 befindet sich bei der Adresse (0, 1) usw. Entsprechend einem Ausführungsbeispiel der Erfindung ist eine Anzahl von Unterabtastungen innerhalb jedes Pixels umfaßt, und dieselben sind als eine Anzahl von Punkten, die mit Unter0–Unter3 in Pixel **73** bezeichnet sind, gezeigt. Die Adresse oder der Ursprung jedes Pixels in dem x,y-Koordinatenraum ist durch die Koordinaten an den x,y-Schnittpunkten des Gitters definiert, Jedes Gitter umfaßt die Unterabtastungen innerhalb des Pixels, das sich in einer steigenden x- und steigenden y-Richtung zu dem nächsten Gitterschnittpunkt erstreckt. Folglich umfaßt das Pixel (1, 0) Unterabtastungen in dem Gitterblock **75**.

[0051] Das Primitiv von **Fig.** 6 ist ein Dreieck **74**, das in einer ersten Farbe auf einem Hintergrund einer zweiten Farbe gezeichnet ist. Wie im vorhergehenden unter Bezugnahme auf **Fig.** 1 beschrieben, würde, wenn das Dreieck **74** in einer Aliasedbetriebsart wiedergegeben wird, jedes Pixel, dort wo der Pixelursprungspunkt durch das Primitiv abgedeckt ist, in der ersten Farbe wiedergegeben werden, während Pixel, dort wo der Pixelursprungspunkt nicht durch das Primitiv abgedeckt ist, in der zweiten Farbe wiedergegeben werden. Das resultierende Aliasedprimitiv würde mit gezackten Kanten wiedergegeben, was folglich ein unteroptimales Bild liefert.

[0052] Um die Kanten des Primitivs zu glätten, kann das Primitiv unter Verwendung einer Antialiasedtechnik wiedergegeben werden, die als Mehrpunktabtasten bekannt ist. Bei dem Mehrpunktabtasten wird jedes Pixel bei mehreren Koordinaten innerhalb des Pixels abgetastet, um eine Anzahl von Pixelunterabtastungen zu liefern. Während des Mehrpunktabtastens wird, wenn eine Farbe zum Anzeigen jedes Pixels ausgewählt wird, die Anzahl der Unterabtastungen der Pixel, die durch das Primitiv abgedeckt werden, in Verbindung mit anderen Informationen (wie z. B. Farbdaten und z-Tiefe) verwendet, um die Farbe des Pixels zu bestimmen. Bei dem Beispiel von **Fig.** 6 weist das Pixel **75** an der x,y-Koordinate (1, 0) zwei Unterabtastungen auf, die durch das Primitiv abgedeckt werden. Folglich würde die Farbe des Pixels eine Mischung von 50% der Primitivfarbe und 50% der Hintergrundfarbe sein.

[0053] Das Mehrpunktabtasten ist daher beim Antialiasing nützlich, da dasselbe Informationen liefert, die ver-

wendet werden, um die Farbe an den Kanten eines Primitivs zu modulieren, um dadurch das Erscheinungsbild von gezackten Kanten zu reduzieren. Das Mehrpunktabtasten ist ferner zum Anzeigen von Objekten nützlich, die kleiner sind als die Größe des Pixels. Beispielsweise wird, wenn ein Objekt kleiner als die Größe des Pixels ist, und wenn dasselbe keinen einzigen Abtastpunkt des Pixels abdeckt, dasselbe nicht in der Aliasedbetriebsart angezeigt. Da das Mehrpunktabtasten das Bild bei mehreren Punkten innerhalb eines Anzeigepixels abtastet, das durch das Pixelgitter definiert ist, können Objekte mit Unterpixelgröße abgetastet werden, und die Farbe des Pixels kann auf der Anzeige dargestellt werden. Eine Komponente des Mehrpunktabtastens ist die Bestimmung, wieviele Unterabtastungen bei jedem Pixel durch das Primitiv abgedeckt werden. Die Unterabtastinformationen werden während einer Abtastumwandlung bestimmt, und dieselben werden in der Form einer Maske zu der Speichersteuerung weitergeleitet, die die Pixelfarbe gemäß der Maske moduliert.

[0054] Wie oben erörtert, haben herkömmliche Verfahren zum Bestimmen der Maske komplexe Hardware verwendet, um die Abdeckung des Primitivs über jedes Pixel auszuwerten, um dadurch die Anzahl der Unterabtastungen zu bestimmen, die durch die Maske angezeigt werden. Im Gegensatz dazu werden bei einem Ausführungsbeispiel der vorliegenden Erfindung gut bekannte und existierende inkrementale Abtastumwandlungsverfahren durch vorhergehendes Durchführen von komplexen Kantenparameterberechnungen in der Software und unter Verwendung dieser Kantenparameter mit einer modifizierten herkömmlichen Hardware erweitert, um Maskeninformationen zu liefern. Gemäß einem Ausführungsbeispiel der vorliegenden Erfindung werden Maskeninformationen durch Unterteilen jeder Abtastzeile in eine Mehrzahl von Unterabtastzeilen, wobei die Abtastpunkte auf Unterabtastzeilengrenzen positioniert sind, und dann unter Verwendung von Hardware erzeugt, die bereits in dem System zusammen mit einer Sammelhardware der Erfindung vorgesehen ist, um Informationen über die Abdeckung der Unterabtastungen innerhalb des Pixels zu sammeln.

[0055] Zurück bezugnehmend auf **Fig.** 5 ist der Abtastwandler **50** gemäß einem Ausführungsbeispiel der vorliegenden Erfindung eine Kantenschrittvorrichtung **52**, eine Sammellogik **54** und eine Weitenschrittvorrichtung **56** umfassend gezeigt. Der Betrieb jeder dieser Komponenten wird im folgenden unter Bezugnahme auf den Abschnitt des darstellenden Bilds **70**, das in **Fig.** 6 gezeigt ist, beschrieben. Die Kantenschrittvorrichtungen und die Weitenschrittvorrichtungen sind häufig in Graphikhardware zum Bestimmen von Koordinaten von Pixeln vorgesehen, die durch Primitive abgedeckt sind. Mit lediglich geringen Modifikationen an einer Kantenschrittvorrichtung und einer Weitenschritteinheit können die Kantenschrittvorrichtung **52**, die Sammeleinheit **54** und die Weitenschrittvorrichtungen eines Ausführungsbeispiels der vorliegenden Erfindung ferner Maskeninformationen zur Verwendung durch die Speichersteuerung **60** während der Antialiasingoperationen liefern.

[0056] Die Kantenschrittvorrichtung **52** empfängt die Kantengleichungen, wie dieselben, die in **Fig.** 4A gezeigt sind, von dem Graphikbeschleuniger **18**. Unter Verwendung der Kantengleichungen schreitet die Kantenschrittvorrichtung inkremental entlang den Kanten des Primitivs unter Verwendung eines vorher berechneten Kantenpunkts und von vorher berechneten Kantenneigungsdaten fort, um die Kantenkoordinaten des Primitivs zu bestimmen. Gemäß einem Ausführungsbeispiel der Erfindung kann die Kantenschrittvorrichtung in entweder einer Aliasedbetriebsart oder einer Antialiasedbetriebsart arbeiten. Bei beiden Betriebsarten ist das Betriebsgrundverfahren identisch, lediglich die Eingabeparameter der Kantenschrittvorrichtung unterscheiden sich. Für Zwecke der Erklärung wird zunächst eine Einführung des Kantenschrittprozesses beschrieben, wobei die Kantenschrittvorrichtung in einer herkömmlichen Aliasedbetriebsart arbeitet. Dann werden die Modifikationen an den Eingabeparametern, die verwendet werden, wenn in einer Antialiasedbetriebsart gearbeitet wird, beschrieben.

[0057] Beispielsweise sind die Kanten des Primitivs **74** von **Fig.** 6 als Kante **1**, Kante **2** und Kante **3** gezeigt. Wie im vorhergehenden beschrieben, empfängt die Kantenschrittvorrichtung Kantenstart- und End-Informationen und Neigungs-Informationen für das Primitiv. Die Kantenschrittvorrichtung schreitet entlang jeder Kante, wobei dieselbe Koordinatendaten für jedes der Pixel auf jeder der Kanten sammelt. Wenn dieselbe entlang der Kante **1** schreitet, "schreitet" die Kantenschrittvorrichtung von dem Start der Kante **1** (dort wo die Kante **1** die Abtastzeile mit der niedrigsten y-Koordinate schneidet; d. h. Punkt A in **Fig.** 6) zu der nächsten Pixelkoordinate von Kante **1**, bis dieselbe das Ende der Kante **1**, d. h. den Scheitelpunkt v1, erreicht, wobei dieselbe im wesentlichen eine Linie zwischen den beiden zeichnet und bestimmt, welche Pixel in dem Bild **70** durch die Kante **1** berührt werden. Ein Blockdiagramm eines darstellenden Ausführungsbeispiels einer Kantenschrittvorrichtung zum Implementieren sowohl von einer Aliased- als auch einer Antialiased-Betriebsart der vorliegenden Erfindung ist in **Fig.** 7 gezeigt. Um entlang der Kante **1** zu schreiten, verwendet die Kantenschrittvorrichtungslogik eine Start-y-Komponente und eine Stopp-y-Komponente, die als Teil der Kantengleichungsdaten von dem Graphikbeschleuniger **18** empfangen werden. Die Start-y-Komponente ist die y-Adresse, bei der die Kante **1** zuerst eine Abtatstzeile kreuzt. Die Stopp-y-Komponente ist die y-Komponente des Scheitelpunkts v1. Bei dem Dreiecksprimitivbeispiel von **Fig.** 6 ist der Start-y-Wert (d. h. die y-Komponente des Punkts A in **Fig.** 6) 1,

und der Stopp-y-Wert ist 3,5. Die Start-y-Komponente wird durch den Multiplexer **85** gespeist und in einem Register **80** als Kante1-Y<15:0> gespeichert. Die Kante1-Y-Komponente ist die Y-Komponente, die aus der Kantenschrittvorrichtung **52** zu der Sammellogik **54** weitergeleitet wird. Die Kante1-Y-Komponente wird zu dem Addierer **86** weitergeleitet, bei dem ein Inkrementwert zu der Kante1-Y-Komponente addiert wird, um die nächste Y-Komponente zu liefern. Der Inkrementwert wird zu dem Multiplexer **82** geliefert, der einen einer Vielfalt von Werten basierend auf dem Zustand eines Aliasedbits, das anzeigt, ob die Kantenschrittvorrichtung in einer Aliased- oder Antialiased-Betriebsart arbeitet, und eine Untergitteradresse, die im folgenden detaillierter beschrieben ist, auswählt. Immer wenn das Graphikgerät in der Aliasedbetriebsart arbeitet, wird a+1 zu der Start-y-Komponente addiert, um zu der nächsten Abtastzeile zu inkrementieren, da jede Pixel-y-Komponente auf einer Abtastzeilengrenze ausgerichtet ist. Die verbleibenden möglichen Inkrementwerte, die in den Multiplexer **82** eingegeben werden, werden zum Unterstützen von Operationen in einer Antialiasingbetriebsart geliefert, wie es detaillierter im folgenden beschrieben ist. Folglich umfassen die Modifikationen der Kantenschrittvorrichtungslogik zum Unterstützen der Antialiasedbetriebsart einen Multiplexer **82** und eine Auswahlsteuerung **84** und einen Komparator **93**, die im folgenden erklärt werden.

[0058] Zusätzlich zu der Start-y- und der Stopp-y-Komponente wird eine Start-x-Komponente durch den Multiplexer **95** gespeist und in einem Register **94** gespeichert. Eine Neigung-x-Komponente wird in einem Register **96** gespeichert. Die Start-x- und Neigung-x-Komponente werden als Teil der Kantengleichungsdaten von dem Graphikbeschleuniger **18** empfangen. Die Start-x-Komponente ist die x-Komponente der Kante **1**, dort wo die Kante **1** zuerst eine Abtastzeile kreuzt; d. h. die x-Koordinate, die der Start-y-Koordinate (d. h. die x-Komponente des Punkts A in **Fig.** 6) entspricht. Die Neigung-x ist die Neigung der Kante **1**. Bei dem Beispiel von **Fig.** 6 ist der Start-x-Wert die x-Adresse 2,2 und die Neigung-x ist 1,5.

[0059] Während des ersten Zyklusses des Kantenschrittprozesses in der Aliasedbetriebsart wird die Start-y-Komponente durch den Multiplexer **85** weitergeleitet und in dem Register **80** als ein aktueller y-Wert gespeichert, wenn ein Signal Neues-Primitiv aktiviert ist. Das Signal Neues-Primitiv wird immer dann aktiviert, wenn ein neues Primitiv zu der Kantenschrittvorrichtungseinheit weitergeleitet wird. Die Start-x-Komponente wird durch den Multiplexer **95** weitergeleitet, wenn das Signal Neues-Primitiv aktiviert ist und in dem Register **94** gespeichert wird, was einen aktuellen x-Wert liefert. Der aktuelle x-Wert wird zu einem Addierer **99** weitergeleitet, der entweder a+1 oder a–1 zu dem aktuellen x-Wert aus Gründen addiert, die im folgenden beschrieben werden. Der aktuelle x-Wert wird zu einem Multiplexer **100** weitergeleitet. Der Multiplexer **100** wählt zwischen dem aktuellen x-Wert und dem inkrementierten (oder dekrementierten) x-Wert abhängig von dem Zustand eines Bits Zustand-Kantel (das im folgenden detaillierter beschrieben ist) aus, um einen eingestellten Wert X<31:16> zu liefern. Der eingestellte X-Wert wird zu dem Multiplexer **97** weitergeleitet, bei dem ein verschobener, eingestellter X-Wert oder ein nicht-verschobener, eingestellter X-Wert abhängig von dem Zustand eines Aliasedbits (das den Betrieb in einer Antialiasedbetriebsart anzeigt) Kante1-X<31:16> weitergeleitet. Der Wert Kante1-X<31:16> wird aus der Kantenschrittvorrichtung **52** zu der Sammellogik **54** weitergeleitet. Ähnlich wird der aktuelle Y<31:16> zu dem Multiplexer **83** weitergeleitet, der entweder einen verschobenen aktuellen Y-Wert oder einen nicht-verschobenen aktuellen Y-Wert als einen Wert Kante1-Y<31:16> abhängig von dem Zustand des Aliasedbits auswählt. Folglich ist bei diesem Beispiel das erste Kante-1-X-, Kante1-Y-Komponentenpaar, das aus der Kantenschrittvorrichtung weitergeleitet wird (1, 2).

[0060] Die aktuelle Y-Komponente mit durch +1 inkrementiert, und das Resultat wird zurück durch den Multiplexer **85** gespeist und in dem Register **88** gespeichert. In dem gleichen Zyklus wird die Neigung-x bei dem Multiplizierer **91** mit dem Inkrementwert multipliziert, der von dem Multiplexer **82** (der für die Aliasedbetriebsart +1 ist) geliefert wird, zu der aktuellen X-Komponente bei dem Addierer **98** addiert und als eine Komponente Nächstes-X ausgegeben. Die Komponente Nächstes-X wird durch den Multiplexer **95** weitergeleitet und in das Register **94** geladen. Als Resultat wird ein zweites Koordinatenpaar (2, 3) für die Kante **1** erzeugt. Die aktuelle Y-Komponente wird durch +1 inkrementiert, das aktuelle X wird durch die Neigung inkrementiert und die Resultate werden in die Register **80** bzw. **94** geladen, um Kante1-X-, Kante1-Y-Koordinaten von (3, 5) für den nächsten Zyklus zu liefern. Der Prozeß des Inkrementierens fährt solange fort, bis ein Komparator **92** bestimmt, daß der Nächstes-Y-Wert größer als der Wert ist, der in dem Stopp-y-Register **90** gespeichert ist. Wenn der Nächstes-Y-Wert größer als der Stopp-y-Wert ist, wird das Kantenschreiten des Primitivs abgeschlossen, und das Signal Ende-Primitiv wird aktiviert. Folglich sind die Signale, die aus der Kantenschrittvorrichtung **52** zu der Sammellogik **54** weitergeleitet werden, die Signale Kante1-X, Kante1-Y und Ende-Primitiv.

[0061] Bei einem Ausführungsbeispiel werden die Kante **2** und die Kante **3** als eine Kante (Kante **2**/**3**) behandelt, die durch eine Biegung verbunden sind. Die Kantenschrittlogik für die Kante 2/3 ist ähnlich zu derselben, die für die Kante **1** vorgesehen ist, mit einer Ausnahme. Während des Abschreitens der Kante 2/3 wird das Schreiten zunächst entlang der Kante **2** durchgeführt, bis der Stopp-y-Wert für die Kante **2** angetroffen wird,

und dann fährt das Schreiten entlang der Kante **3** fort. Folglich würde unter Bezugnahme auf das Beispiel in **Fig.** 6 das Kantenschreiten entlang der Kante **2** auftreten, startend von den Kante-2-Start-x- und Start-y-Koordinaten (0,8, 1) (gezeigt durch den Punkt B in **Fig.** 6), und eine Kante-2-Koordinate (0,6, 2) erzeugend. Wenn die y-Koordinate auf die dritte Abtastzeile inkrementiert wird, wird ein Vergleich ähnlich zu dem Vergleich, der durch den Komparator **92** in **Fig.** 7 durchgeführt wird, durchgeführt, um den Nächstes-Y-Wert mit dem Stopp-Y-Wert zu vergleichen, der in einem Register ähnlich zu Register **90** gespeichert ist. Wenn die Kante-2-Y-Koordinate größer als der Wert Stopp-y-Wert der Kante **2** ist, beginnt das Schreiten entlang einer Kante-3-Start-X-Koordinate. Bei dem Beispiel von **Fig.** 6 ist die erste Kante-3-Koordinate, die erzeugt wird, für die Abtastzeile **3**, und dieselbe ist (2,7, 3) (gezeigt durch den Punkt C in **Fig.** 6). Das Schreiten entlang der Kante **3** fährt solange fort, bis das Signal Ende-Primitiv durch die Kante-1-Kantenschrittvorrichtung der Kante **1** aktiviert ist. Folglich wird die Logik in **Fig.** 7 in der Kante-2/3-Schrittvorrichtung durch Hinzufügen eines Multiplexers vor dem Startx-Register **94** (um zwischen einem Kante-2-Start-x und einem Kante-3-Start-x auszuwählen) und Hinzufügen eines Multiplexers vor dem Stopp-Y-Register **90** (um zwischen einem Kante-2-Stopp-Y und einem Kante-1-Stopp-Y auszuwählen, der ähnlich zu dem Kante-3-Stopp-y ist) modifiziert. Zusätzlich wird ein Multiplexer vor dem Register **96** hinzugefügt, um zwischen einer Kante-2-Neigung und einer Kante-3-Neigung auszuwählen, und ein Multiplexer wird vor dem Start-x-Register **95** hinzugefügt, um zwischen einem Kante-2-Start und einem Kante-3-Start auszuwählen. Eine Kante-2/3-Koordinate kann zur gleichen Zeit erzeugt werden, zu der eine Kante-1-Koordinate für jede Abtastzeile erzeugt wird.

[0062] Nach der Erzeugung der Kante-1- und der Kante-2/3-Koordinaten für jede Abtastzeile werden bei der Aliasedbetriebsart die Koordinaten zu der Weitenschrittvorrichtung **56** (**Fig.** 5) weitergeleitet. Die Weitenschrittvorrichtung **56** schreitet entlang der x-Achse von einer Kante-1-Koordinate zu einer Kante-2/3-x-Koordinate auf jeder Abtastzeile fort, die durch das Primitiv abgedeckt ist, um die x-y-Koordinaten für alle Pixel zwischen der Kante **1** und der Kante **2**/**3** zu bestimmen, und folglich für alle Pixel, die durch das Primitiv auf dieser Abtastzeile abgedeckt werden.

[0063] **Fig.** 1 ist ein Beispiel der Wiedergabe von Primitiv **74** in der Aliasedbetriebsart, wobei die Pixelkoordinaten unter Verwendung der Kantenschritt- und der Weitenschritt-Operationen, die im vorhergehenden beschrieben sind, erzeugt wurden. Da lediglich die Pixelkoordinaten während des Weitenschreitens und des Kantenschreitens verglichen und erzeugt werden, sind lediglich diese Pixel "ein", deren vollständige Position durch das Primitiv abgedeckt wird, wobei die verbleibenden Pixel "aus" sind. Zusätzlich gibt es, aufgrund der Aliasedbetriebsart, keine Modulation der Farbe entlang der Kanten des Primitivs, wobei jedes Pixel entweder die Hintergrundfarbe oder die Primitivfarbe aufweist. Folglich ist die Aliasedbetriebsart zum Wiedergeben von Primitiven nützlich, wenn eine hohe Leistung erforderlich ist. Wie in **Fig.** 1 gezeigt, sind jedoch Aliasedkanten häufig gezackt und visuell unbefriedigend.

[0064] Das Antialiasing unter Verwendung der Mehrpunktabtastverfahren erzeugt allgemein Bilder mit höherer Qualität, jedoch mit einer reduzierten Leistung oder höheren Hardwarekosten im Vergleich zu Aliasedbildern, da mehrere Abtastungen jedes Pixels analysiert werden, wenn die Farbe jedes Pixels bestimmt wird. Wie im vorhergehenden beschrieben, werden das Kantenschreiten und das Abtastschreiten in der Aliasedbetriebsart auf einer Abtastzeilenkörnigkeit zum Liefern von Pixelkoordinateninformationen zu der Speichersteuerung durchgeführt. Bei einem Ausführungsbeispiel der Erfindung wird bei einer Antialiasedbetriebsart jedoch jedes Pixel in eine Anzahl von Unterabtastzeilen aufgeteilt, und das Kantenschreiten wird auf einer Unterabtastzeilenkörnigkeit durchgeführt. Unterabtastungen sind auf einer oder mehreren Unterabtastzeilen positioniert, und Informationen darüber, ob jede der Unterabtastungen durch das Primitiv abgedeckt ist, werden während der Antialiasedbetriebsart zu der Speichersteuerung durch die Weitenschrittvorrichtung weitergeleitet. Bei einem Ausführungsbeispiel der Erfindung wird die gleiche Hardware für das Kantenschreiten und das Weitenschreiten in sowohl der Aliased- als auch der Antialiasedbetriebsart vorteilhaft verwendet. Folglich können die Unterabtastungsinformation mit minimalen Änderungen zu der Kantenschrittvorrichtungs- und der Weitenschrittvorrichtungs-Hardware geliefert werden.

[0065] Bei einem Ausführungsbeispiel der Erfindung wird, bevor jede Abtastzeile in eine Mehrzahl von Unterabtastzeilen unterteilt wird, zuerst das Bild durch einen vorbestimmten Betrag skaliert, derart, daß jede der Unterabtastzeilen auf eine Ganzzahladreßgrenze fällt. Auf diese Art und Weise müssen die Weiten- und die Kantenschrittvorrichtungen keine gebrochenen Pixeladressen handhaben, was die Hardware vereinfacht. Obwohl dieses Merkmal vorteilhaft ist, ist die Erfindung jedoch in dieser Hinsicht nicht begrenzt. Folglich muß das Skalieren nicht verwendet werden, und die Unterabtastzeilenkoordinaten können Bruchteilen von Pixelkoordinaten entsprechen. Folglich ist das Skalieren lediglich ein Ausführungsbeispiel zum Teilen einer Abtastzeile in eine Anzahl von Unterabtastzeilen, die den Abtastkoordinaten innerhalb eines Pixels zugeordnet sind.

[0066] Bei dem Skalierungsausführungsbeispiel der Erfindung wird das Skalieren des Bilds in der Transformierungseinheit **30** (**Fig.** 3) des Graphikbeschleunigerchips **18** durch Multiplizieren der Werte in der Transformierungsmatrix **32** (**Fig.** 3) mit einem gewünschten Skalierwert durchgeführt. Es muß keine zusätzliche Funktionalität zu der Transformierungseinheit hinzugefügt werden, um diese Antialiasunterstützung vorzusehen, da dieselbe schon die Skalierfunktionalität zum Liefern von gezoomten Bildern vorsieht. Der Skalierwert wird teilweise durch die Anzahl der Abtastpunkte bestimmt, die für jedes Pixel erwünscht sind. Bei einem Ausführungsbeispiel der Erfindung wird ein Skalierwert von 8 zum Ermöglichen der Plazierung von vier Abtastpunkten verwendet, obwohl andere Skalierwerte und Abtastpunktzahlen alternativ verwendet werden können. Die Auswahl der Anzahl der Skalierpunkte wird durch die Berechnungszeit, die erforderlich ist, um jede der Abtastungen auf den Unterabtastzeilen zu verarbeiten, und durch eine relative Verbesserung des Antialiasedbilds gegenüber einer Aliaseddarstellung des Bilds beeinflußt. Bei einem Ausführungsbeispiel der Erfindung wurde herausgefunden, daß die Verwendung von vier Abtastpunkten pro Pixel eine wesentlich verbesserte Bildqualität gegenüber dem Aliasedbild liefert, während die gute Leistung beibehalten wird.

[0067] Wie im vorhergehenden erörtert, ist bei einem Ausführungsbeispiel der Erfindung der Skalierfaktor (z. B. 8) größer als die Anzahl der Abtastpunkte (z. B. 4). Dies ist dahingehend vorteilhaft, daß dies eine nicht-reguläre Plazierung der Unterabtastungen innerhalb des Pixels erleichtert. Eine nicht-reguläre Plazierung bedeutet, daß die Unterabtastungen jedes Pixels bei Positionen entlang der Unterabtastzeilen positioniert werden können, derart, daß die Unterabtastungen keine reguläre, gitterähnliche Struktur bilden. Das Plazieren der Unterabtastungen bei nicht-regulären Intervallen entlang der Unterabtastzeilen führt zu einer Reduktion der visuellen Bildfehler, die durch die geordneten Strukturen bewirkt werden. Die vorliegende Erfindung ist jedoch nicht, wenn skaliert wird, auf die Verwendung eines Skalierfaktors begrenzt, der größer als die Anzahl der Abtastungen ist. Eine Technik, die zum Bestimmen eines Skalierwerts und der Anzahl der Abtastpunkte zum Erzeugen eines Ausgabebilds mit minimalen visuellen Bildfehlern verwendet werden kann, ist in dem Artikel "Hardware Assisted Polygon Anti-Aliasing", IEEE Computer Graphics & Applications, Januar 1991, von Anthony C. Barkans, der hierin durch Bezugnahme aufgenommen ist, beschrieben.

[0068] Das Skalieren eines Bilds mit 8 liefert eine Anzahl von anderen Vorteilen. Da der Skalierbetrag eine Potenz von 2 ist, wird der Multiplikationsschritt des Skalierens ohne weiteres durch Bitverschieben durchgeführt. Zusätzlich werden, da eine Skaliergröße von 8 ausgewählt wurde, drei Bits zum Bestimmen einer Unterabtastzeilenadresse verwendet. Wie es im folgenden beschrieben werden wird, umfassen die x- und die y-Adressen, die bei einem Ausführungsbeispiel der vorliegenden Erfindung verwendet werden, jeweils einen 16-Bit-Ganzzahlabschnitt, derart, daß die Weiten- und Kanten-Schrittvorrichtungshardware Register umfaßt, die 16-Bit-Ganzzahlpixelkoordinaten unterstützen können. Wenn drei der Ganzzahladreßbits als Unterabtastzeilenadreßbits während des Antialiasing verwendet werden, verbleiben 13 y-Adreßbits zum Adressieren von Pixeln innerhalb eines Bilds. Viele Anwendungen, wie z. B. X-Windows, erfordern einen Bildpixeladreßbereich von 8 K, der 13 Bit erfordert. Die Verwendung eines Skalierfaktors von 8 ermöglicht das Maximieren der Flexibilität von Unterabtastplazierungen, während dennoch Anwendungen, wie z. B. X-Windows, für Antialiasing-Operationen unterstützt werden.

[0069] Obwohl ein Ausführungsbeispiel der vorliegenden Erfindung einen Skalierfaktor von 8 mit vier Unterabtastungen verwendet, ist die vorliegende Erfindung nicht auf diese Werte begrenzt. Es können vielmehr andere Skalierfaktoren und andere Unterabtastzahlen alternativ verwendet werden. Wie im vorhergehenden beschrieben, ist die Auswahl des Skalierwerts abhängig von dem verfügbaren Adreßraum und der Leichtigkeit der Berechnung. Die Auswahl der Unterabtastanzahl wird basierend auf einer gewünschten Bildqualität und einer gewünschten Leistung bestimmt.

[0070] Unter Bezugnahme nun auf **Fig.** 8 ist ein Beispielpixel **97** gezeigt. Vor dem Skalieren werden die Abtastzeilengrenzen des Pixels durch die Abtastzeile 0 und die Abtastzeile 1 definiert. Das Pixel **97** wird durch einen Faktor von 8 skaliert, um skalierte Pixel **99** zu erzeugen. Bei dem skalierten Pixel **99** sind Unterabtastpixel bei der Unterabtastzeile 0, der Unterabtastzeile 3, der Unterabtastzeile 5 und der Unterabtastzeile 7 positioniert. Bei dem Beispiel von **Fig.** 8 sind die Unterabtastungen bei den folgenden Positionen positioniert: Unterabtastung 0 = (0, 0), Unterabtastung 1 = (7, 3), Unterabtastung 2 = (3, 5) und Unterabtastung 3 = (5, 7). Die Unterabtastungen können alternativ bei anderen Punkten entlang der Unterabtastzeilen plaziert werden.

[0071] Obwohl das Mehrpunktabtasten nun hinsichtlich einer Kantenschrittechnik beschrieben wird, kann die obige Skaliertechnik ferner verwendet werden, um die mathematischen Berechnungen bei anderen Verfahren von Antialiasoperationen zu vereinfachen, wobei die mathematischen Berechnungen mindestens auf ein Informationsstück, das sich auf das Bild bezieht, wirken, und wobei die Informationen gebrochene Komponenten umfassen. Beispiele der Informationen können Unterabtastpunkte bei dem Mehrpunktabtasten umfassen, die-

selben können jedoch auch Neigungs- und Kantengleichungs-Parameter sein. Daher kann das Skalieren des Bilds verwendet werden, um die gebrochenen Informationskomponenten zu eliminieren, um dadurch die Notwendigkeit nach komplexen Gleitkommaberechnungen zu eliminieren.

[0072] Das Kantenschreiten wird auf dem skalierten Gitter für Antialiasedbilder auf eine Art und Weise durchgeführt, die ähnlich zu derselben ist, die unter Bezugnahme auf die **Fig.** 7 für Aliasedbilder beschrieben ist. Der einzigste Unterschied liegt darin, wie die Y-Achsenadresse interpretiert wird; d. h. die Y-Achsenadresse ist eine Unterabtastzeilenadresse bei einer Antialiasedbetriebsart, im Gegensatz zu einer Abtastzeilenadresse bei einer Aliasedbetriebsart.

[0073] Unter Bezugnahme nun auf **Fig.** 9 ist ein Ausführungsbeispiel des X- oder Y-Adreßregisters 32 Bit aufweisend gezeigt, die in einem 16.16-Format gespeichert sind. Bei einem 16.16-Format sind die höchstwertigsten 16 Bit Ganzzahlen, und die niedrigstwertigsten 16 Bit stellen einen gebrochenen Abschnitt der Adresse dar. Eine darstellende Zuweisung von X- oder Y-Adreßregisterbits ist, wenn in einer Aliasedbetriebsart gearbeitet wird, als Register **101** gezeigt, wobei der Ganzzahlabschnitt des X- oder Y-Adreßregisters **101a** 16 Bit von X- oder Y-Adreßinformationen aufweist. Eine darstellende Zuweisung der X- oder Y-Adreßregisterbits ist, wenn in einer Antialiasedbetriebsart gearbeitet wird, als Register **103** gezeigt, wobei der Ganzzahlabschnitt der X- oder Y-Adresse eine 13-Bit-X- oder Y-Adresse **103a** und eine 3-Bit-Untergitteradresse **103b** umfaßt, wobei die 3-Bit-Untergitteradresse eine Adresse von einer der 8 Unterabtastzeilen des Pixels codiert. Das Kantenschreiten verwendet bei der Aliasedbetriebsart den Ganzzahlabschnitt der Y-Adreßregisterbits (31:16, in **Fig.** 9 gezeigt) als eine Basis für das inkrementale Schreiten, während das Kantenschreiten bei der Antialiasedbetriebsart die gleichen Y-Bits <31:16> verwendet, wobei jedoch, wie es gezeigt wird, Untergitteradreßbits <18:16> verwendet werden, um einen Teil des Antialiasingprozesses zu steuern.

[0074] Obwohl es acht mögliche Untergitter-Y-Werte gibt, werden bei dem darstellenden Ausführungsbeispiel lediglich vier Unterabtastungen verwendet. Als ein Resultat werden lediglich vier Unterabtastzeilen, auf denen die Unterabtastungen liegen (wie in **Fig.** 8 gezeigt) während des Kantenschrittprozesses analysiert. Daher wird, wenn das Anfangs-Y-Adreß-Untergitterfeld **103b** kein Untergitterwert ist, der einer Unterabtastzeile entspricht, die Start-Y-Adresse inkrementiert, um bei der nächsten Unterabtastzeile innerhalb des Gitters zu starten. Beispielsweise wird, wenn das Y-Untergitteradreßfeld **103** eine 1 ist, der Start-Y-Wert durch +2 bis zu a3 inkrementiert, da dies die nächste Unterabtastzeile ist, die einen Abtastpunkt in dem Gitter aufweist.

[0075] Zurück bezugnehmend auf **Fig.** 7 ist eine Beschreibung darüber angegeben, wie bei einem Ausführungsbeispiel der Erfindung die Kantenschritthardware ferner für das Antialiasedkantenschreiten verwendet wird. Der Start-Y-Wert wird durch den Multiplexer **85** weitergeleitet, in das Register **80** geladen, und als aktuelles Y<31:16> gespeichert. Das aktuelle Y<31:16> wird zu dem Multiplexer **93** weitergeleitet, bei dem entweder ein verschobenes Y oder ein nicht-verschobenes aktuelles Y als der Kante1-Y-Wert abhängig von dem Zustand des Aliasedbits geliefert wird. Wie in **Fig.** 9 beschrieben, werden die Bits <18:16> bei der Antialiasedbetriebsart verwendet, um eine Unterabtastzeilenkörnigkeit für das Schreiten zu liefern. Folglich ist für die Antialiasedbetriebsart der Ganzzahlabschnitt der Y-Komponente aus den Bits <31:16> des aktuellen Y-Werts gebildet, und für die Aliasedbetriebsart ist der Ganzzahlabschnitt der Y-Komponente aus den Bits <28:16> des aktuellen Y-Werts gebildet.

[0076] Der aktuelle Y-Wert wird ferner zu dem Addierer **86** weitergeleitet. Zusätzlich wird ferner ein Inkrement zu dem Addierer **86** weitergeleitet. Der Betrag des Inkrements wird durch die Auswahlsteuerlogik **81** bestimmt, und derselbe basiert auf dem Zustand des Aliasedbits (das, wie oben beschrieben, anzeigt, ob ein Graphikgerät in der Aliasedbetriebsart arbeitet oder nicht) und basiert ferner auf der Anfangsuntergitteradresse. Wie im vorhergehenden beschrieben, addiert, wenn die Untergitteradresse nicht eine Unterabtastzeilenadresse ist, die einen Abtastpunkt aufweist, dann die Anfangsaddieroperation das gewünschte Inkrement, um zu der nächsten Unterabtastzeile zu gehen. Das Inkrement, das zum Erreichen der nächsten Unterabtastzeilenadresse benötigt wird, wird dann basierend auf der aktuellen Unterabtastzeilenadresse und einem Delta-y zu der nächsten Unterabtastzeilenadresse ausgewählt. Beispielsweise ist, wenn die y-Komponente der Unterabtastzeilenadressen (0), (3), (5) und (7) ist, dann das Unterabtastzeilendelta von der Unterabtastzeile 0 zu der Unterabtastzeile 1 +3, von der Unterabtastzeile 1 zu der Unterabtastzeile 2 +2, von der Unterabtastzeile 2 zu der Unterabtastzeile 3 +2, und von Unterabtastzeile 3 zu der Unterabtastzeile 0 des nächsten Pixels +1.

[0077] Unter Verwendung des Beispiels von **Fig.** 8 und schreitend entlang der Kante **1** ist der Start-y-Wert eine 1, der Stopp-y-Wert ist größer als 8,0 (da die Kante sich über das Pixel **99** erstreckt), der Starty-x-Wert ist eine 1 und die Neigung x ist eine 1. Während des ersten Zyklusses wird die Start-y-Komponente durch den Multiplexer **85** weitergeleitet, in den Registern **80** gespeichert und als Kante1-Y zu der Sammellogik **54** aus-

gegeben. Die Start-x-Komponente wird durch den Multiplexer **95** weitergeleitet und in dem Register **94** gespeichert, um einen aktuellen x-Wert zu liefern. Der aktuelle x-Wert wird durch den Multiplexer **100** weitergeleitet und als ein eingestellter X-Wert, der auf eine Art und Weise verschoben wird, die ähnlich zu derselben ist, die im vorhergehenden unter Bezugnahme auf die aktuelle Y-Komponente beschrieben wurde, geliefert, um eine Kante1-X-Komponente zum Weiterleiten zu der Sammellogik **54** zu liefern.

[0078] Während des nächsten Zyklusses des Kantenschrittprozesses wird die aktuelle Y-Komponente von 1 durch +2 inkrementiert, um die Unterabtastzeile **3** zu erhalten. Die x-Koordinate an der Unterabtastzeile **3** wird durch Multiplizieren des Neigungs-x-Werts bei dem Multiplizierer **91** mit dem Inkrementwert von +2 und durch Addieren der Resultate zu dem aktuellen X-Wert bei dem Addierer **98** bestimmt. Die Resultate von dem Addierer **98** werden zurück durch den Multiplexer **95** gespeist, und dieselben werden in einem Register **94** gespeichert, um schließlich Kante1-Y- und Kante1-X-Koordinaten von (3,3) für den nächsten Verarbeitungszyklus von (3,3) zu liefern. Während des nächsten Zyklusses ist der Inkrementbetrag, der benötigt wird, um die nächste Unterabtastzeile (y=5) zu erhalten a+2. Der Prozeß fährt, wie im vorhergehenden beschrieben, fort, was Kantenkoordianten von (5, 5) und (7, 7) in den nächsten zwei Zyklen ergibt.

[0079] Ein Komparator **93** vergleicht die oberen **13** Bits der Nächstes-Y-Komponente von dem Addierer **86** mit den oberen **13** Bits des Kante1-Y-Werts von dem Register **80**. Wie im vorhergehenden erwähnt, stellen die oberen **13** Bits die Abtastzeilenkomponente der Y-Adresse dar. Die zwei sind jedesmal ungleich, wenn eine Abtastzeilengrenze gekreuzt wird. Wenn diese zwei ungleich sind, oder wenn das Signal Ende-Primitiv aktiviert ist, wird ein Signal Neues-Y aktiviert, um den Start des Verarbeitens einer neuen Abtastzeile anzuzeigen. Das Signal Neues-Y wird zu der Sammellogik **54** weitergeleitet.

[0080] Während der Analyse des Pixels **99** werden vier Sätze von Kante1-X-, Kante1-Y-Koordinatenpaaren erzeugt, obwohl lediglich drei der Koordinatenpaare auf die Unterabtastzeilen fallen. Es ist jedoch offensichtlich, daß ein Gesamt von vier Kantenunterabtastzeilenkoordinaten für jedes Pixel für Fälle erzeugt wird, bei denen die Kante sich über alle Unterabtastzeilen des Pixels erstreckt. Sowie jedes Kantenunterabtastzeilenkoordinaten-x-y-Paar erzeugt wird, wird dasselbe zu der Sammellogik **54** (**Fig.** 5) gespeist. Zusätzlich wird zusammen mit jedem Kantenunterabtastzeilenkoordinatenx-y-Paar die y-Untergitteradresse der Unterabtastzeile gesendet. Wie es detaillierter im folgenden beschrieben wird, wird die y-Untergitteradresse verwendet, um zwischen den x-, y-Koordinatenpaaren, die sich auf einer gültigen Unterabtastzeile (d. h. Unterabtastzeilen 0, 3, 5 oder 7) befinden, und denselben, die sich nicht auf einer gültigen Unterabtastzeile befinden (d. h. dieselben, die auf die Unterabtastzeilen 1, 2, 4 oder 6 fallen) befinden, zu unterscheiden.

[0081] Die Kantenschrittvorrichtung **52** umfaßt zusätzlich eine Zustandslogik **84** zum Liefern eines Zustandsbits, das anzeigt, ob eine Unterabtastung auf einer Unterabtastzeile durch eine Kante des Primitivs abgedeckt ist. Das Zustandsbit wird gemäß der x-Koordinate der Kante, sowie dieselbe die Unterabtastzeile kreuzt, der x-Koordinate der Unterabtastung auf der jeweiligen Unterabtastzeile, und des Richtungswerts, der von dem Graphikbeschleuniger **18** empfangen wird, bestimmt.

[0082] Wie im vorhergehenden beschrieben, zeigt der Richtungswert an, in welche Richtung geschritten wird, um von der Kante **1** (die Kante des Primitivs mit der größten Y-Achsenweite) zu der Kante **2** (die Kante, die den obersten Y-Scheitelpunkt mit der Kante **1** teilt) zu gehen. Beispielsweise kann ein "1"-Wert für die Richtung anzeigen, daß die Kante **1** sich links befindet, und daß das Schreiten von links nach rechts durchgeführt wird, und eine "0" kann anzeigen, daß sich die Kante **1** rechts befindet, und daß das Schreiten von rechts nach links durchgeführt wird. Der Richtungswert kann als ein Signal zu der Abtastwandlerlogik **50** auf dem Bus **19** geliefert werden, oder derselbe kann alternativ zu einem Register (nicht gezeigt) der Abtastumwandlungslogik **50** geschrieben werden. Bei einem Ausführungsbeispiel der Erfindung wird das Richtungssignal in das CMD-Feld codiert, das von dem Graphikbeschleuniger **18** zu der Abtastumwandlungslogik **50** weitergeleitet wird.

[0083] Sowie die Kantenschrittvorrichtung entlang der Unterabtastzeilen schreitet, wobei dieselbe x,y-Unterabtastzeilenkoordinatendaten sammelt, wird ein Zustandsbit, das anzeigt, ob die Unterabtastung auf der Unterabtastzeile abgedeckt ist, durch die Zustandslogik **84** (**Fig.** 7) erzeugt. Die Zustandslogik **84** umfaßt einen Komparator **89**, der gekoppelt ist, um die aktuelle x-Koordinate von dem Register **94** zu empfangen. Der Komparator empfängt ferner ein Eingangssignal von einem Multiplexer **87**. Die Dateneingaben in den Multiplexer sind die x-Koordinaten der Unterabtastungen für jede Unterabtastzeile. Wie im vorhergehenden beschrieben, sind die Unterabtastkoordinaten bei dem darstellenden Ausführungsbeispiel (0, 0), (7, 3), (3, 5) und (5, 7). Die Auswahleingänge des Multiplexers **87** werden durch die Untergitteradreßbits <18:16> von **Fig.** 9 gesteuert, die den Unterabtastzeilenadreßbits entsprechen.

[0084] Die Vergleichslogik **89** empfängt ferner das Richtungssignal als ein Eingangssignal. Wenn das Richtungssignal anzeigt, daß die Schrittrichtung von links nach rechts geht, aktiviert die Vergleichslogik die Zustandsleitung für alle Unterabtastzeilen, wobei die x-Koordinate der Kante kleiner ist als die Unterabtast-x-Koordinate (da alle gültigen Unterabtastungen sich rechts der Kante befinden werden). Wenn das Richtungssignal anzeigt, daß die Schrittrichtung von rechts nach links geht, aktiviert die Vergleichslogik die Zustandsleitung für alle Unterabtastzeilen, wobei die x-Koordinate der Kante größer als die Unterabtast-x-Koordinate ist, da alle gültigen Unterabtastungen links der Kante liegen werden. Die Kantenschrittlogik für die Kante 2/3 ist im wesentlichen gleich zu derselben, die in **Fig.** 6 für die Kante **1** gezeigt ist. Ein Unterschied besteht jedoch darin, daß ein invertierter Wert des Richtungssignals in die Zustandslogik **84** eingegeben wird, um sicherzustellen, daß die geeigneten Unterabtastpixel für die Kante 2/3 ausgewählt werden.

[0085] Das Kantel-Zustandssignal wird verwendet, um den Multiplexer **100** zu steuern. Wenn die aktuelle x-Koordinate keine Unterabtastung abdeckt, dann wird das Kantel-Zustandssignal bewirken, daß der Multiplexer **100** einen modifizierten aktuellen x-Wert als die Kante1-X-Komponente auswählt, die zu der Sammellogik weitergeleitet wird. Die modifizierte x-Koordinate ist die nächste folgende Ganzzahl-x-Koordinate in der Richtung des Schreitens relativ zu der aktuellen x-Koordinate. Folglich wird die aktuelle x-Koordinate abhängig von der Richtung des Schreitens entweder mit 1 durch den Addierer **99** inkrementiert oder dekrementiert, wenn die Vergleichoperation, die durch die Zustandslogik **84** durchgeführt wird, anzeigt, daß die aktuelle x-Koordinate keinen Abtastpunkt auf der Unterabtastzeile abdeckt. Das Inkrementieren oder Dekrementieren der x-Koordinate wird vorgesehen, um spezifische Typen von Primitiven identifizieren zu können, die eine Breite aufweisen, die kleiner als die Breite eines Pixels ist. Es genügt zu erwähnen, daß nach der Einstellung der aktuellen x-Koordinaten für sowohl die Kante **1** als auch die Kante **2** es vorkommen kann, daß der Abstand zwischen der Kante **1** und der Kante 2/3 kleiner oder gleich 0 ist. Der Effekt des Aufweisens eines Abstands kleiner oder gleich 0 zwischen den Kanten wird im folgenden hierin beschrieben.

[0086] Folglich werden die Signale Kante1-X, Kante1-Y, Kante23-X, Neues-Y und Ende-Primitiv alle zu der Sammellogik **94** von der Kantenschrittvorrichtung **52** weitergeleitet. Die Kante1-X-, Kante23-X- und Kante1-Y-Koordinaten werden nach jeder Unterabtastzeilenberechnung weitergeleitet. Das Signal Neues-Y wird aktiviert, wenn alle Unterabtastzeilen in einer gegebenen Unterabtastzeile verarbeitet wurden. Das Signal Ende-Primitiv wird aktiviert, wenn das Abschreiten des Primitivs abgeschlossen ist.

[0087] Unter Bezugnahme nun auf **Fig.** 10 ist ein Blockdiagramm eines darstellenden Ausführungsbeispiels der Sammellogik umfassend einer Steuereinheit **104**, eines Datenwegs **106**, eines Y-Koordinatenpuffers **108**, eines Kante1-X-Koordinatenpuffers **110**, eines Längenpuffers **112** und einer Mehrzahl von weiteren Puffern **114** gezeigt, die Farb-, z-Daten und andere Daten speichern, die jeder der Kante entsprechen. Jeder der Puffer **108**, **110**, **112** und **114** umfaßt zwei Sätze von vier Registern, wobei jeder der vier Registersätze eine Sammlung von Unterabtastzeilenkantenkoordinatendaten für das Pixel speichert, das zuletzt durch die Kantenschrittvorrichtung verarbeitet wurde. Daten werden alternativ durch jede Hälfte des Puffers empfangen, derart, daß Daten in eine Hälfte des Puffers eingegeben werden können, während Daten aus der anderen Hälfte des Puffers gelesen werden.

[0088] Die Steuerlogik **104** empfängt die Unterabtastzeilen-Y-Koordinate, das Signal Neues-Y, das Signal Ende-Primitiv und ein CMD-Feld zum Identifizieren einer Funktion, die an den Pixeldaten durchgeführt werden soll. Beispielsweise kann das CMD-Feld decodiert werden, um ein Betriebsartzustandsbit, das den Betrieb in der Antialiasedbetriebsart anzeigt, und einen Richtungswert zu liefern, der die Richtung des Schreitens zeigt. Die Steuereinheit umfaßt zwei Funktionseinheiten, eine Schreibsteuereinheit **104a** und eine Lesesteuereinheit **104b**. Die Schreibsteuerlogik steuert das Schreiben der empfangenen Kante1-Y- und Kante1-X-Koordinatendaten in die Puffer **108** bzw. **110**. Zusätzlich steuert, wie es im folgenden detaillierter beschrieben wird, die Schreibsteuerlogik das Schreiben eines Längenwerts, der den Kante1-Y- und Kante1-X-Werten entspricht, in einen Längenpuffer **112**.

[0089] Die Kante1-X- und Kante23-X-Komponenten werden zu dem Datenweg **106** weitergeleitet. Der Datenweg verarbeitet die Kante1-X- und Kante23-X-Koordinaten, um einen gesammelten Längenwert zu liefern. Die Schreibsteuerlogik steuert das anschließende Schreiben des gesammelten Längenwerts von dem Datenweg **106** zu dem Puffer **112**. Die Schreibsteuerlogik **104a** liefert ferner Signale GÜLTIG<3:0> und ein Signal Neue-Gruppe zu der Schreibsteuerlogik **104b**. Das Signal Neue-Gruppe wird aktiviert, wenn entweder das Signal Neues-Y oder das Signal Ende-Primitiv durch die Kantenschrittvorrichtung **52** aktiviert wird. Jedes Bit des Signals GÜLTIG entspricht einer der vier Unterabtastlinien und zeigt an, ob die Unterabtastzeile durch das Primitiv berührt wurde oder nicht. Bei einem Ausführungsbeispiel der Erfindung werden die Signale GÜLTIG von den y-Untergitter-Adreßbits <18:16> codiert.

[0090] Die Lesesteuerlogik **104b** umfaßt eine Zustandsmaschine, die die Übertragung von Befehls-, Koordinaten-, Längen- und Masken-Daten zu der Weitenschrittvorrichtung (WS) umfaßt. Insbesondere steuert die Steuerlogik die Übertragung von WS-CMD<3:0> (von der Steuereinheit **104**), WS-Y<31:15> (von dem Puffer **108**), WS-E1X<15:0> (von dem Puffer **110**), Signale WS-Länge (von dem Puffer **112**) und Signale Farbe/Z-out von dem Puffer **114** zu der Weitenschrittvorrichtung **56**. Folglich steuert die Schreibsteuerlogik **104a** die Ansammlung von Unterabtastzeileninformationen in den Puffern **108**, **110**, **112** und **114**, während die Lesesteuerlogik **104b** die Übertragung von Daten von Puffern **108**, **110**, **112** und **114** zu der Weitenschrittvorrichtungslogik **58** (**Fig.** 5) steuert. Die Lesesteuerlogik **104b** liefert zusätzlich ein Signal MASKE<3:0>, das zu der Weitenschrittvorrichtung weitergeleitet wird, um die Unterabtastungen zu identifizieren, die durch ein Pixel abgedeckt werden. Ein Signal Mux-Auswahl wird von der Lesesteuereinheit **104b** zu dem Datenweg **106** weitergeleitet, um das Weiterleiten des Signals WS-Länge zu der Weitenschrittvorrichtung zu steuern, wie es im folgenden detaillierter beschrieben ist.

[0091] Der Datenweg **106** umfaßt ähnlich eine Sammeleinheit **128** und eine Weiterleitungseinheit **130**. Die Sammeleinheit **128** empfängt Kante1-X- und Kante23-X-Koordinaten-Daten von der Kantenschrittvorrichtung **52**. Die Kante1-X- und Kante23-X-Koordinaten entsprechen den Koordinaten der Kante **1** und der Kante **2**/**3** für die Y-Unterabtastzeilenkoordinate, die verarbeitet wird. Unter Verwendung dieser Koordinateninformationen kann die Sammeleinheit **128** für jede Unterabtastzeile eine Länge der Unterabtastzeile bestimmen. Die Länge der Unterabtastzeile wird von der Sammeleinheit **128** als Signale Gesammelte-Länge geliefert und in dem Längenpuffer **112** gespeichert.

[0092] Zusätzlich zum Bestimmen der Länge jeder Unterabtastzeile bestimmt die Sammellogik zusätzlich einen Start und ein Ende einer "belegten (= blasted) Region" der Unterabtastzeile, die verarbeitet wird, wobei die belegte Region Pixel innerhalb der Unterabtastzeile identifiziert, bei denen alle Abtastungen durch das Primitiv abgedeckt werden. Unter Bezugnahme nun auf **Fig.** 11 ist ein Primitiv **150** gezeigt. Die Pixel, deren Unterabtastungen alle durch das Primitiv **150** abgedeckt sind, umfassen Pixel **153** und **154**, auf die als "belegte" Regionen des Primitivs Bezug genommen wird.

[0093] Der belegte Startwert entspricht der x,y-Koordinate des Pixels, bei dem die belegte Region beginnt. Der belegte Längenwert entspricht der Anzahl von Pixeln, die entlang der Unterabtastzeile in der belegten Region abgeschritten werden. Während des Auswertens jeder Unterabtastzeile durch die Sammeleinheit wird die x-Koordinate, bei der die Kante des Primitivs die Unterabtastzeile kreuzt, ausgewertet, um zu bestimmen, ob dieselbe die x-Koordinate ist, die am weitesten in der Richtung des Schreitens für den Satz der vier Unterabtastzeilen ist. Unter Verwendung des Primitivs **150** von **Fig.** 11 als ein Beispiel und unter Verwendung der vier Unterabtastzeilen startend von Abtastzeile 2, sind die Kante-1-Pixel-x-Koordinaten, bei denen die Kante die Unterabtastzeilen 0, 1, 2 und 3 schneidet, 5, 4 bzw. 5, wobei die Pixel-X-Koordinate 4 auf der Unterabtastzeile 1 durch die Einstellogik in **Fig.** 7 eingestellt wird.

[0094] Wenn ein belegter Startwert für die Kante **1** erzeugt wird, sowie jede gültige Unterabtastzeilenkoordinate von der Schreibsteuerlogik **104a** empfangen wird, wird derselbe mit einer vorher gespeicherten Unterabtastzeile-x-Koordinate für das Pixel verglichen, um zu bestimmen, ob dieselbe weiter in der Richtung des Schreitens als eine vorhergehende Unterabtastzeilen-x-Koordinate liegt. Folglich ist, wenn der belegte Startwert für das Pixel (5, 2) bestimmt wird, die Pixel-x-Koordinate für die Unterabtastzeile 0 eine 5, die Pixel-x-Koordinate für die Unterabtastzeile 1 ist eine 4,7, die Pixel-x-Koordinate für die Unterabtastzeile 2 ist eine 5,5, und die Pixel-x-Koordinate für die Unterabtastzeile 3 ist eine 5,7. Folglich wird die x-Koordinate 4,7 abgeschnitten und als der belegte Startwert ausgewählt. Es folgt eine ähnliche Operation unter Verwendung der Koordinaten der Kante **2**/**3**, um den belegten Endwert zu bestimmen.

[0095] Es sollte offensichtlich sein, daß das viermalige Weiterleiten der gleichen x,y-Koordinate zu der Speichersteuerung mit unterschiedlichen Maskenbitsätzen die Leistung des Antialiasingsystems beeinflußt. Das Identifizieren jener Pixel, die sich innerhalb der belegten Region befinden, durch Vorsehen einer Maske, deren Bits alle gesetzt sind, reduziert die Anzahl der Übertragungen zwischen der Weitenschrittvorrichtung und der Speichersteuerung, da lediglich eine Speicherübertragung pro Pixel durchgeführt wird.

[0096] Unter Bezugnahme nun auf **Fig.** 12 ist ein Flußdiagramm gezeigt, das den Betrieb der Sammeleinheit **128** zum Liefern des gesammelten Längenwerts, des belegten Startwerts und des belegten Längenwerts darstellt. Der gesammelte Längenwert wird für jede Unterabtastzeile erzeugt. Der belegte Startwert und der belegte Längenwert werden bestimmt, nachdem alle Unterabtastzeilen ausgewertet wurden.

[0097] Beispielsweise wird bei einem Schritt **200** das Richtungssignal ausgewertet. Wenn die Schrittrichtung

von rechts nach links geht, dann wird der Verarbeitungsweg, der mit dem Schritt **201a** beginnt, ausgewählt. Wenn die Schrittrichtung von links nach rechts geht, dann wird der Verarbeitungsweg ausgewählt, der mit einem Schritt **201b** beginnt. Die Verarbeitungswege sind im wesentlichen ähnlich, wobei die Unterschiede in dem Typ der Vergleiche (d. h. größer als im Gegensatz zu kleiner als) liegen, die in jedem Weg durchgeführt werden. Folglich ist lediglich der Verarbeitungsweg, der mit dem Schritt **201a** beginnt, im folgenden beschrieben. Bei einem Schritt **201a** wird der belegte Startwert und der belegte Endwert initialisiert. Abhängig von der Schrittrichtung wird der belegte Startwert entweder auf die Kante1-X-Koordinate oder auf die Kante23-X-Koordinate der Unterabtastzeile eingestellt, während der belegte Endwert als Gegenüber des belegten Starts initialisiert wird. Die belegten Startwerte werden auf diese Art und Weise initialisiert, um folgende Koordinaten zu identifizieren, die gröber (oder kliener) als der initialisierte belegte Startwert (abhängig von der Schrittrichtung) sind, wie es aus der folgenden Beschreibung klarer werden wird.

[0098] Bei einem Schritt **202a** werden die Kante1-X- und Kante23-X-Koordinaten für eine gegebene Unterabtastzeile von der Schreibsteuerlogik **104a** durch die Sammeleinheit **128** empfangen. Bei einem Schritt **205** wird die Kante1-X-Koordinate mit dem belegten Startwert verglichen. Bei dem ersten Durchlauf durch die Schritte von **Fig.** 2 wird der Vergleich mit dem belegten Startwert durchgeführt, während bei den folgenden Durchläufen der Vergleich mit dem belegten Startwert durchgeführt wird, wie er während des Prozesses aktualisiert wird. Wenn die Kante1-X-Koordinate kleiner als der belegte Startwert ist, und die Richtung des Schreitens von rechts nach links geht, dann wird der Kante1-X-Wert als der belegte Startwert bei einem Schritt **209** gespeichert. Bei einem Schritt **203** wird ein Vergleich durchgeführt, um zu bestimmen, ob die Kante23-X-Koordinate größer als der belegte Endwert für eine Rechts-nach-Links-Schrittrichtung ist, und wenn dies der Fall ist, dann wird bei einem Schritt **207** der Kante23-X-Wert als der belegte Endwert gespeichert. Bei einem Schritt **211** wird die Gesamtlänge der Unterabtastzeile durch Subtrahieren der Kante23-X-Koordinate von der Kante1-X-Koordinate bestimmt. Bei einem Schritt **213** wird der Längenwert, der bei dem Schritt **211** berechnet wurde, der gesammelten Länge zugewiesen, die von der Sammeleinheit **128** weitergeleitet wird und in dem Längenpuffer **112** gespeichert wird.

[0099] Der Zustand des Signals Neues-Y, das von der Kantenschrittvorrichtung **52** geliefert wird, wird bei einem Schritt **215** geprüft. Das Signal Neues-Y wird aktiviert, um anzuzeigen, daß die Y-Eingabe-Koordinate sich zu der nächsten vollen Abtastzeile verschoben hat, und daß folglich alle Unterabtastzeilen für eine gegebene Abtastzeile verarbeitet wurden, oder daß das Primitiv nicht weiter verarbeitet wird. Wenn bei dem Schritt **215** das Signal Neues-Y nicht aktiviert ist, wird der Prozeß bei einem Schritt **202a** neu aufgenommen, bei dem neue Signale Kante1-X und Kante23-X von der Kantenschrittvorrichtung **52** empfangen werden. Wenn bei einem Schritt **214** das Signal Neues-Y aktiviert ist, wird ein belegter Längenwert durch Subtrahieren des belegten Starts von dem belegten Ende berechnet, und der belegte Startwert und der belegte Längenwert werden zu der Weiterleitungseinheit **130** des Datenwegs **106** weitergeleitet.

[0100] Sowie eine Gruppe von Unterabtastzeilen gemäß dem Prozeß verarbeitet werden, der in **Fig.** 12 beschrieben ist, wird der gesammelte Längenwert für die Unterabtastzeile in dem Puffer **110** angesammelt. Wenn alle Unterabtastzeilen verarbeitet wurden, d. h. wenn das Signal Neues-Y aktiviert ist, startet die Lesesteuerlogik **104b** den Prozeß des Weiterleitens der Kante1-X-, Kante1-Y-, Längen-, Masken- und CMD-Werte für jede Unterabtastzeile, die in den Puffern gespeichert ist, zu der Weitenschrittvorrichtung **56**. Für jedes Unterabtastzeilen-Kante1-X- und Kante1-Y-Koordinatnepaar jeder Unterabtastzeile, das in den Puffern **108** und **110** gespeichert ist, gibt es potentiell drei Datenübertragungen, die für die Unterabtastzeile auftreten können. Bei allen Datenübertragungen werden die gleichen Kante1-X- und Kante1-Y-Werte weitergeleitet, es können sich jedoch die Masken-, CMD- und Längen-Werte unterscheiden, wie es im folgenden beschrieben ist. Obwohl die gleiche Kante1-X-Koordinate zu der Weitenschrittvorrichtung für jede Übertragung weitergeleitet wird, ist es im folgenden sichtbar, daß diese Kante1-X-Koordinate lediglich zu der Speicherungssteuerung **60** während des ersten Übertragungszyklusses pro Unterabtastzeile übertragen wird.

[0101] Die drei Datenübertragungen werden verwendet, um zwischen den Pixeln auf der Unterabtastzeile vor der belegten Region, den Pixeln innerhalb der belegten Region und den Pixeln über die belegte Region hinaus zu unterscheiden. Die Anzahl der Pixel vor der belegten Region wird als Länge1 bezeichnet, die Anzahl der Pixel innerhalb der belegten Region wird als Länge2, oder belegte Länge, bezeichnet, und die Anzahl der Pixel über die belegte Region hinaus wird als Länge3 bezeichnet. Die Länge1- und Länge3-Werte können sich für jede Unterabtastzeile unterscheiden. Beispielsweise weist unter Bezugnahme wiederum auf **Fig.** 11 die Unterabtastzeile 0 einen Länge1-Wert von 1 auf, da sich ein Pixel an der Position (5, 2) vor der belegten Region befindet. Die Unterabtastzeile 0 weist einen Länge2-Wert von 2 auf, da sich zwei Pixel innerhalb der belegten Region befinden. Der Länge3-Wert für die Unterabtastzeile 0 ist 0, da sich keine Pixel auf der Unterabtastzeile 0 über die belegte Region hinaus befinden. Die Länge1-, Länge2- und Länge3-Werte für die Unterabtastzeile

1 sind 0, 2 bzw. 1. Die Länge1-, Länge2- und Länge3-Werte für die Unterabtastzeile 2 sind 1, 2 bzw. 2 und die Werte für die Unterabtastzeile 3 sind 1, 2 bzw. 3. Die Länge1-, Länge2- und Länge3-Werte werden durch die Weiterleitungseinheit **130** (**Fig.** 10) bestimmt, und dieselben werden verwendet, um den geeigneten Längenwert für jeden der drei Datenübertragungszyklen der Unterabtastzeile auf der Leitung WS-Länge zu der Weitenschrittvorrichtung **56** zu liefern.

[0102] Unter Bezugnahme nun auf **Fig.** 13 ist ein Blockdiagramm der Weiterleitungseinheit **130** als einen ersten Subtrahierer **220** umfassend gezeigt, zum Bestimmen des Länge1-Werts ansprechend auf den WS-E1X-Wert, des belegten Startwerts und des Richtungssignals. Die WS-E1X- und Längen-Werte werden von den Puffern **110** bzw. **112** für jede Unterabtastzeile gesteuert durch das Lesesteuersignal von der Lesesteuerlogik **104b** in der Steuereinheit **104** empfangen. Ein zweiter Subtrahierer **222** berechnet Länge3 durch Subtrahieren der Summe der belegten Länge und des Länge1-Werts von dem Längenwert, der von dem Puffer **112** empfangen wird, der die Gesamtlänge der Unterabtastzeile, wie bei einem Schritt **211**/**212** von **Fig.** 12 berechnet, anzeigt. Die Länge1- und Länge3-Werte werden zu der Testlogik **224** und **226** weitergeleitet, die die Werte testet, um zu bestimmen, ob dieselben kleiner oder gleich Null sind. Wenn dies der Fall ist, werden die Signale Länge_klgl_0 und Länge3_klgl_0 aktiviert und zu der Lesesteuerlogik **104b** weitergeleitet, die dieselben zum Steuern der Übertragung der Pixeldaten zu der Weitenschrittvorrichtung verwendet, wie es unter Bezugnahme auf **Fig.** 15 beschrieben werden wird. Der Längenwert von dem Puffer **112** wird zu der Testlogik **225** gespeist, um ein Signal Länge_klgl_0 zu erzeugen, das zu der Lesesteuerlogik **104b** zum Anzeigen weitergeleitet wird, ob derselbe kleiner oder gleich Null ist. Der Wert der belegten Länge wird ferner zu dem Tester **227** gespeist, um ein Signal Belegt_klgl_0 zu erzeugen, das zu der Lesesteuerlogik **104b** weitergeleitet wird, das ferner zum Steuern der Übertragung von Pixeldaten zu der Weitenschrittvorrichtung verwendet wird, wie es beschrieben werden wird.

[0103] Die Länge1-, Länge3-, Längen- und Belegte-Länge-Werte werden zu einem Multiplexer **228** weitergeleitet. Abhängig davon, welcher Abschnitt der Unterabtastzeile zu der Weitenschrittvorrichtung **56** durch die Lesesteuerung **104b** weitergeleitet wird, wird der geeignete Längenwert als WS-Länge zu der Weitenschrittvorrichtung durch die Steuerlogik **229** ansprechend auf das Signal Mux-Auswah1<2:0> weitergeleitet, das von der Lesesteuerlogik **104b** empfangen wird.

[0104] Unter Bezugnahme nun auf **Fig.** 14 ist ein Flußdiagramm vorgesehen, das den Betrieb einer Zustandsmaschine, die in einer Antialiasedbetriebsart in der Lesesteuerlogik **104b** zum Verarbeiten der Unterabtastzeilen arbeitet, darstellt. Die Zustandsmaschine umfaßt vier Zustände, wobei jeder Zustand dem Verarbeiten einer der Unterabtastzeilen gewidmet ist. Jeder Zustand wird lediglich betreten, wenn sowohl das entsprechende Bit GÜLTIG, das von der Schreibsteuerlogik **104a** empfangen wird, anzeigt, daß die Unterabtastzeile durch das Primitiv an der entsprechenden x,y-Koordinate berührt wurde, als auch, wenn das Signal Länge_klgl_0 für die Unterabtastzeile nicht aktiviert ist; d. h. solange die Gesamtlänge der Unterabtastzeile größer als Null ist. Folglich wird bei einem Schritt **230** GÜLTIG<0> überprüft, um zu bestimmen, ob die Unterabtastzeile 0 gültig ist. Wenn dies der Fall ist, wird bei einem Schritt **232** die Unterabtastzeile 0 verarbeitet, wie es unter Bezugnahme auf **Fig.** 15 beschrieben wird, und ein Signal Maske für die Unterabtastzeile 0 wird als 0001 definiert. Wenn die Unterabtastzeile 0 nicht gültig ist, oder nach dem Verarbeiten der Unterabtastzeile 0, fährt der Prozeß zu dem Schritt **234** fort, wo GÜLTIG<1> und der Längenwert für die Unterabtastzeile 1 überprüft wird, um zu bestimmen, ob die Unterabtastzeile 1 gültig ist. Wenn dies der Fall ist, wird bei einem Schritt **236** die Unterabtastzeile 1, wie in **Fig.** 15 beschrieben, mit einem Wert Maske von 0010 verarbeitet. Der Prozeß fährt durch die Schritte **238**, **240**, **242** und **244** fort, bis jede der Unterabtastzeilen des Pixels ausgewertet wurde, wobei Masken 0100 und 1000 für die Unterabtastzeilen 2 bzw. 3 vorgesehen werden, wenn die Unterabtastzeilen gültig sind.

[0105] Unter Bezugnahme nun auf **Fig.** 15 beginnt ein Flußdiagramm, das das Verarbeiten jeder Unterabtastzeile durch den Prozeß von **Fig.** 14 in der Lesesteuerung **104b** zeigt, wenn das Signal Neue-Gruppe durch die Lesesteuerlogik **104a** aktiviert wird, was folglich anzeigt, daß eine neue Gruppe von Unterabtastzeilen zum Verarbeiten bereit ist. Bei einem Schritt **249** wird der Zustand des Signals Belegt_klgl_0 überprüft, um zu bestimmen, ob es eine belegte Region für den Satz von Unterabtastzeilen gibt. Wenn die belegte Länge kleiner oder gleich Null ist (d. h. wenn es keine belegte Region gibt), dann fährt der Prozeß zu einem Schritt **254** fort, bei dem die Lesesteuerlogik einen WS-E1X-Wert von dem Puffer **110**, einen WS-Y-Wert von dem Puffer **108**, den WS-Länge-Wert von der Weiterleitungseinheit **130** (durch Aktivieren des geeigneten Signals Mux-Auswahl), die Maske, die dem Verarbeitungszustand (**232**, **236**, **240** oder **244**) der Steuerlogik der Lesesteuerung **104b** entspricht, die Farb/z-Daten von dem Puffer **114** und ein Signal WS-CMD zu der Weitenschrittvorrichtung weiterleitet.

[0106] Wenn es jedoch eine belegte Region gibt, fährt der Prozeß bei **250** durch Überprüfen des Signals Länge1_klgl_0 fort, das von dem Datenweg **106** empfangen wird. Wenn das Signal Länge1_klgl_0 zeigt, daß Länge1 gleich Null ist, dann muß kein Verarbeiten für Länge1 durchgeführt werden, und der Prozeß fährt zu einem Schritt **255** fort. Wenn jedoch Länge1 nicht gleich Null ist, dann leitet die Lesesteuerung einen WS-E1X-Wert von dem Puffer **110**, einen WS-Y-Wert von dem Puffer **108**, den WS-Länge-Wert von der Weiterleitungseinheit **130** (durch Aktivieren des geeigneten Signals Mux-Auswahl), die Maske, die dem Verarbeitungszustand (**232**, **236**, **240** oder **244**) der Steuerlogik der Lesesteuerung **104b** entspricht, die Farb/z-Daten von dem Puffer **114** und ein Signal WS-CMD zu der Weitenschrittvorrichtung weiter.

[0107] Bei einem Ausführungsbeispiel weist das Signal WS-CMD die im folgenden in Tabelle I gezeigten Codierungen auf:

TABELLE I

AA_CMD_0	vor der belegten Region
AA_CMD_1	Fortfahren über die belegte Region
AA_CMD_2	Fortfahren vorbei an der belegten Region
AA_CMD_3	Auslassen der belegten Region
AA_CMD_4	Starten durch Ansammeln der belegten Region
AA_CMD_5	Starten durch Auslassen der belegten Region

[0108] Das WS-CMD wird verwendet, um es der Weitenschrittvorrichtung zu ermöglichen, eine geeignete X-Koordinatenadresse auszuwählen, um dieselbe zu der Speichersteuerung **60** weiterzuleiten. Wie es detaillierter im folgenden beschrieben wird, hilft das Signal WS-CMD der Weitenschrittvorrichtung, das nächste Pixel in der Unterabtastzeile zu identifizieren, das verarbeitet werden muß, in dem es dieselbe anweist, vorher angesammelte E1X-, Farbe-, Z- und Textur-Werte und nicht WS-E1X-, Farbe-Z- und Textur-Werte, die durch die Sammellogik geliefert werden, zu verwenden.

[0109] Folglich steuert dann in dem Flußdiagramm von **Fig.** 15 und unter Verwendung des Beispiels von **Fig.** 11 bei der Abtastzeile 2, der Abtastzeile 0, wenn der Länge1-Wert nicht gleich Null ist, bei einem Schritt **252** die Lesesteuerlogik das Weiterleiten von WS-E1X, WS-Y, WS-Länge, Maske und WS-CMD zu der Weitenschrittvorrichtung, wobei WS-CMD = AA_CMD_0 anzeigt, daß die Koordinate, die gesendet wurde, sich vor einer belegten Region befindet, und wobei der WS-Länge-Wert gleich Länge1 ist, und die Maske auf 0001 eingestellt ist, was anzeigt, daß die Unterabtastzeile 0 bei dem Zustand 0001 in dem Prozeß von **Fig.** 14 verarbeitet wird.

[0110] Bei einem Schritt **255** wird bestimmt, ob die Maske 0001 gleicht. Das Vergleichen des Maske-Werts mit 0001 identifiziert, ob die belegte Region schon verarbeitet wurde oder nicht, da die Maske 0000 gleicht, wenn die belegte Region bei der Unterabtastzeile 0 verarbeitet wird. Gemäß einem Ausführungsbeispiel der Erfindung wird die belegte Region lediglich einmal für eine Gruppe von Unterabtastzeilen verarbeitet, um vorteilhaft die Verarbeitungszeit für das Pixel zu reduzieren. Wenn dieselbe schon verarbeitet wurde, fährt der Prozeß zu einem Schritt **258** fort. Wenn die belegte Region noch nicht verarbeitet wurde, werden bei einem Schritt **256** die WS-E1X-, WS-Y- und WS-Längen- (gleich der belegten Länge) Werte zu der Weitenschrittvorrichtung gesendet. Der Maske-Wert, der gesendet wird, ist auf 1111 eingestellt, was folglich anzeigt, daß alle Unterabtastungen des Pixels abgedeckt sind. Der WS-CMD, der gesendet wird, ist AA_CMD_1, was anzeigt, daß die Weitenschrittvorrichtung quer über die belegte Region fortfahren sollte, startend bei der vorher inkrementierten x-Koordinate (d. h. die x-Koordinate nach dem Verarbeiten von Länge1), und durch Fortfahren des Inkrementierens der x-Koordinate von diesem Punkt.

[0111] Der Prozeß fährt weiter zu einem Schritt **258** fort, bei dem das Signal Länge3_klgl_0 von der Weiterleitungseinheit überprüft wird, um zu bestimmen, ob es Pixel in Länge3 der Unterabtastzeile gibt, die auszuwerten sind. Wenn Länge3 nicht Null gleicht, dann werden bei einem Schritt **260** die WS-E1X-, WS-Y- und WS-Länge- (gleich Länge3) Werte zu der Weitenschrittvorrichtung gesendet. Der Maske-Wert, der gesendet wird, wird gemäß dem Verarbeitungszustand des Prozesses von **Fig.** 14 eingestellt, der die Routine von **Fig.** 15 aufruft. Das WS-CMD, das gesendet wird, ist AA_CMD_2, was anzeigt, daß die Weitenschrittvorrichtung das Inkrementieren der x-Koordinate fortfahren sollte, startend bei der x-Koordinate des letzten Pixels in der belegten Region.

[0112] Es gibt weitere Befehle, die zu der Weitenschrittvorrichtung gesendet werden, die zur Klarheit nicht unter Bezugnahme auf das Flußdiagramm von **Fig.** 15 beschrieben wurden. Diese Befehle umfassen AA_CMD_3, um die belegte Region nach dem Starten mit Länge1 auszulassen, den AA_CMD_4, um durch Ansammeln der belegten Region zu starten, und den AA_CMD_5, um durch Auslassen der belegten Region zu starten. Der AA_CMD_3 wird gesendet, wenn Länge1 verarbeitet wurde, die belegte Region schon für eine vorhergehende Abtastzeile verarbeitet wurde und Länge3 nicht gleich Null ist. AA_CMD_3 wird gesendet, wenn Länge1 gleich Null ist, und es aber eine zu verarbeitende belegte Region gibt. Der AA_CMD_5 wird gesendet, wenn Länge1 gleich Null ist, und die belegte Region schon durch eine vorhergehende Unterabtastzeile verarbeitet wurde. Andere Befehle können zusätzlich umfaßt sein, um die Funktionalität der Weitenschrittvorrichtung zu vergrößern.

[0113] Folglich umfaßt jeder Zyklus von Daten, der von der Sammellogik zu der Weitenschrittvorrichtung weitergeleitet wird, einen WS-E1X-, einen WS-Y-, einen WS-Länge-, einen Maske- und einen WS-CMD-Wert. Die Weitenschrittvorrichtung **56** verwendet die obigen Signale, um eine Adresse für jedes Pixel zu bestimmen, das durch das Primitiv abgedeckt wird. Um die Adresse für jedes derartige Pixel zu bestimmen, schreitet die Weitenschrittvorrichtung folgend von einer x-Koordinate zu einer anderen entlang einer Unterabtastzeile auf der Gesamtlänge der Unterabtastzeile. Wenn die Weitenschrittvorrichtung bestimmt, daß ein Pixel abgedeckt ist, wird eine x- und y-Koordinate für dieses Pixel zu der Speichersteuerung **60** weitergeleitet, wobei die Pixel-y-Koordinatenadresse, die weitergeleitet wird, die Abtastzeilenadresse für dieses Pixel ist. Zusätzlich wird eine Vier-Bit-Maske zu der Speichersteuerung **60** weitergeleitet, wobei der Maske-Wert dem Signal Maske entspricht, das von der Sammellogik gesendet wird. Folglich könnte für ein Pixel mit vier Unterabtastzeilen die Weitenschrittvorrichtung **56** potentiell die gleiche x,y-Koordinate viermalig zu der Speichersteuerung weiterleiten, wobei ein anderes Bit in der Maske für jede x,y-Koordinatenübertragung eingestellt ist.

[0114] Wie vorher erwähnt, beeinflußt das viermalige Weiterleiten der gleichen x,y-Koordinaten mit unterschiedlichen eingestellten Maskenbits das Verhalten des Antialiasingsystems. Das Identifizieren jener Pixel, die sich innerhalb der belegten Region befinden, durch Vorsehen einer Maske, deren Bits alle eingestellt sind, reduziert die Anzahl der Übertragungen zwischen der Weitenschrittvorrichtung und der Speichersteuerung, da lediglich eine Speicherübertragung pro Pixel durchgeführt wird.

[0115] Da alle vier Bits der Maske in der belegten Region eingestellt sind, liefert die Identifikation der belegten Region eine Leistungsverbesserung, wenn dieselbe bei dem Ausführungsbeispiel der Erfindung verwendet wird, bei dem dieselbe mit einer Speichersteuerung **60** verwendet wird, die implementiert ist, wie es in der '350er Patentanmeldung, die oben zitiert ist, beschrieben ist. Wie in der Patentanmeldung erörtert, kann die Speichersteuerung **60** Pixel mit entweder lediglich einem eingestellten Bit in der Maske derselben oder Pixel, bei denen alle Bits derselben in der Maske eingestellt sind, verarbeiten. Wenn ein Pixel eine Maske aufweist, bei der zwei oder drei der Maskenbits eingestellt sind, werden die Pixelkoordinaten einmal für jeden Maskenbitsatz, der eingestellt ist, von der Weitenschrittvorrichtungslogik zu der Speicherungssteuerung zwei- oder dreimal weitergeleitet. Wenn jedoch alle vier Bits eingestellt sind, müssen die Pixelkoordinaten lediglich einmal zu der Speichersteuerung weitergeleitet werden. Weitere Details bezüglich der Leistungsvorteile des Identifizierens von belegten Regionen werden im folgenden geliefert.

[0116] Unter Bezugnahme nun auf **Fig.** 16 ist ein Blockdiagramm der Weitenschrittvorrichtungslogik **56** gezeigt. Die Weitenschrittvorrichtung schreitet in einer gegebenen Richtung von der Kante **1** zu der Kante **2** entlang der x-Koordinatenachse, wie es durch das Richtungssignal angezeigt ist. Das Richtungssignal wird verwendet, um einen Multiplexer **262** zu steuern, um entweder etwa einen Wert a+1 oder einen Wert a–1 für das Signal mit dem Ink/Dek Wert zu liefern. Das Signal WS-CMD wird von der Steuereinheit **104** (**Fig.** 10) zu der Zustandsmaschine **263** weitergeleitet, die den WS-CMD in die folgenden Signale decodiert: Laden-Neue-Daten zum Steuern des Ladens der Koordinate WS-Y in das Register **266**, Laden-Neue-Maske zum Steuern des Ladens der Maske in ein Register **264**, ein Signal Ansammeln-Belegt zum Steuern des Inkrementierens der Belegt-Zähllogik **270** und Signale Auswahl<1:0>, die verwendet werden, um die Auswahl einer x-Koordinate

bei einem Multiplexer **265** zu steuern.

[0117] Die Weitenschrittvorrichtung arbeitet allgemein wie folgt. Wenn Signale WS-E1X, WS-Y und WS-CMD bei der Weitenschrittvorrichtung empfangen werden, werden Werte für die Steuerungssignale Laden-Neue-Daten, Laden-Neue-Maske, Ansammeln-Belegt und Auswahl<1:0> durch die Zustandsmaschine **263** erzeugt. Die Signale Auswahl<1:0> wählen eines von vier x-Koordinaten-Eingangssignalen in den Multiplexer **265** aus, und der Multiplexer leitet das ausgewählte Eingangssignal an das Register **262** weiter, das als die Pixel-X-Koordinate zu der Speichersteuerung **60** geliefert werden soll.

[0118] Die vier Eingangssignale in den Multiplexer **265** umfassen die WS-E1X-Koordinate, die von der Sammellogik **54** empfangen wird, eine Nach-Belegt-Start-X-Koordinate, eine Fortfahren-X-Koordinate und eine Länge3-X-Koordinate. Bezugnehmend wiederum auf Tabelle 1 zeigt der Zustand von WS-CMD an, welche x-Koordinate die Weitenschrittvorrichtung als eine Startkoordinate für das Weitenschreiten verwendet. Beispielsweise wird, wenn sich WS-CMD zu AA_CMD_0 decodiert, was anzeigt, daß das Weitenschreiten bei dem Start von Länge1 beginnen sollte, dann die WS-E1X-Koordinate durch den Multiplexer **265** ausgewählt. Wenn sich WS-CMD zu AA_CMD_1 decodiert, was anzeigt, daß das Weitenschreiten das Zählen von dem Ende von Länge1 quer über die belegte Region fortfahren sollte, wird die Fortfahren-X-Koordinate durch den Multiplexer **265** ausgewählt. Die Fortfahren-X-Koordinate wird ferner ausgewählt, wenn sich WS-CMD zu AA_CMD_2 decodiert, was anzeigt, daß das Zählen von dem Ende der belegten Region in Länge3 fortfährt. Wenn sich WS CMD zu AA_CMD_3 decodiert, was anzeigt, daß die Weitenschrittvorrichtung das Schreiten nach der Länge1 starten sollte, jedoch die belegte Region auslassen sollte, wird die Nach-Belegt-X-Koordinate durch den Multiplexer **265** ausgewählt. Wenn sich WS-CMD zu AA_CMD_4 decodiert, was anzeigt, daß die Weitenschrittvorrichtung das Schreiten bei der belegten Region beginnen sollte, da die Länge1 kleiner oder gleich Null ist, wird die WS-E1X-Koordinate ausgewählt. Wenn WS-CMD sich zu AA_CMD_5 decodiert, was anzeigt, daß die Weitenschrittvorrichtung das Schreiten nach der belegten Region beginnen sollte, da die belegte Region vorher verarbeitet wurde, wird die Länge3-X-Koordinate durch den Multiplexer **265** ausgewählt.

[0119] Die Nach-Belegt-Start-X-Koordinate und Länge3-X-Koordinate werden durch die Addierer **267** bzw. **269** geliefert. Das andere Eingangssignal in die Addierer **267** und **269** ist ein Eingangssignal Delta-X. Das Eingangssignal Delta-X wird durch den Belegt-Zähler **270** geliefert, und dasselbe zeigt die Änderung der x-Koordinate quer über die belegte Region an. Der Delta-X-Wert wird durch den Multiplexer **260** initialisiert, der den Ink/Dek-Wert, der von dem Multiplexer **262** empfangen wird, zu dem Register **261** für Initialisierungszwecke weiterleitet. Wenn das Signal Ansammeln-Belegt aktiviert ist (d. h., wenn WS-CMD gleich AA_CMD_1 oder gleich AA_CMD_4 ist), schaltet der Multiplexer **260**, um das Ausgangssignal von dem Addierer **259** zum Weiterleiten zu dem Register **261** auszuwählen. Der Addierer inkrementiert (oder dekrementiert) den Delta-X-Wert für jeden Zyklus, daß eine Pixel-X-Koordinate während des Verarbeitens der belegten Region geliefert wird. Wenn das Signal Ansammeln-Belegt deaktiviert ist, stellt der Wert Delta-X, der in dem Register **261** gespeichert ist, den Änderungsbetrag des Werts der x-Koordinate dar, sowie das Schreiten quer über die belegte Region fortfuhr.

[0120] Die Ansammlungslogik kann ähnlich wie der Belegt-Zähler 270 zum Spurverfolgen und Sichern der totalen Änderung der Farbe und der Textur quer über einer Unterabtastzeile vorgesehen werden. Die Ansammlungslogik würde, eher daß dieselbe einen Wert a +/– 1 zu dem Eingang des Addierers liefert, stattdessen gemäß einem Deltafarb- und Deltatextur-Parameter inkrementieren, der als Teil der Kantengleichung in der Software abgeleitet würde, um eine Gesamtänderung der Farbe oder Textur quer über die Unterabtastzeile zu bestimmen.

[0121] Der Betrieb der Weitenschrittvorrichtung **56** wird nun unter Bezugnahme auf das Weitenschreiten quer über die Zeile 2 des Beispielprimitivs **150** von **Fig.** 11 beschrieben. Während des ersten Weitenschrittzyklusses, wie im vorhergehenden beschrieben, sind die WS-E1X-, WS-Y-Koordinaten (5, 2), der WS-Länge-Wert ist 1, die Maske ist gleich 0001 und der WS-CMD bewirkt, daß die WS-E1X-Koordinate durch den Multiplexer **265** weitergeleitet, in das Register **262** geladen und als Pixel-X-Koordinate ausgegeben wird. Die Pixel-X-Koordinate wird zu den Addierern **267** und **268** weitergeleitet. Da der WS-Länge-Wert eine 1 war, und eine Pixel-X-Koordinate ausgegeben wurde, werden bei dem nächsten Zyklus WS-E1X, WS-Y, Maske, WS-Länge und WS-CMD entsprechend der belegten Region **152** empfangen.

[0122] Unter Verwendung des Beispielprimitivs von **Fig.** 11 gleicht der empfangene WS-CMD zum Verarbeiten der belegten Region der Unterabtastzeile 0 AA_CMD_1. Der Befehl AA_CMD_1 bewirkt, daß die Signale Laden-Neue-Maske, Laden-Neue-Daten und Ansammeln-Belegt aktiviert werden. Wie oben erwähnt, bewirkt der AA_CMD_1-Wert für den WS-CMD, daß die Fortfahren-X-Koordinate für das Schreiten ausgewählt wird.

Die Fortfahren-X-Koordinate wird durch Dekrementieren der Pixel-X-Koordinate (durch Addieren eines negativen Einswerts zu der Pixel-X-Koordinate, da die Schrittrichtung von links nach rechts geht) erhalten, in dem Register **262** gespeichert und als Pixel-X-Koordinate (4, 2) ausgegeben. Die Maske, die in dem Register **264** gespeichert ist, ist gleich 1111, wie es in Schritt **256** von **Fig.** 15 beschrieben ist. Die Fortfahren-X-Koordinate wird weiter für jedes Pixel in der belegten Region ausgewählt. Zum gleichen Zeitpunkt zählt der Delta-X-Wert den Änderungsbetrag der x-Koordinate quer über die belegte Region. Folglich wird in einem nächsten Zyklus eine Pixel-X,Y-Koordinate (3, 2) mit einer Maske 1111 erzeugt. Nachdem das zweite Pixel-X,Y-Koordinatenpaar und die zugeordnete Maske zu der Speichersteuerung weitergeleitet wurden, ist das Abschreiten der belegten Region abgeschlossen, und der gespeicherte Delta-X-Wert ist gleich 2.

[0123] Bei dem Beispiel von **Fig.** 11 fährt, sobald die belegte Region **152** für die Unterabtastzeile 0 verarbeitet wurde, da die Länge3 der Unterabtastzeile 0 gleich Null ist, das Verarbeiten bei der Unterabtastzeile 1 fort. Während des ersten Zyklusses des Verarbeitens der Unterabtastzeile 1 gleicht Länge1 Null, so würde das Verarbeiten mit der belegten Region beginnen. Die belegte Region wurde jedoch schon verarbeitet. Folglich sind die ersten Koordinaten, die weitergeleitet werden WS-E1X, WS-Y, Maske, WS-Länge (von einer Länge3), und WS-CMD = AA_CMD_5, was anzeigt, daß durch Auslassen der belegten Region gestartet werden soll. Die Signale Auswahl<1:0> sind eingestellt, um die Länge3-X-Koordinate von dem Addierer auszuwählen. Folglich sind für die Unterabtastzeile 1 die ersten Pixel-X,Y-Koordinaten, die zu der Speichersteuerung weitergeleitet werden, (2, 2). Die Pixel-Maske, die weitergeleitet wird, wird durch den Verarbeitungszustand des Prozesses von **Fig.** 14 bei Schritt **236** angezeigt und ist folglich 0010 bei diesem Beispiel.

[0124] Folglich wird bei einem Ausführungsbeispiel der Erfindung die Maske entweder lediglich mit einem eingestellten Bit, oder wobei alle Bits eingestellt sind, weitergeleitet. Die Maskenbits werden auf die oben erwähnte Art und Weise weitergeleitet, da das Ausführungsbeispiel der Speichersteuerung, das in der Patentanmeldung beschrieben ist, auf die oben Bezug genommen wurde, lediglich die Kapazität aufweist, um die Pixeldaten entweder als diskrete Unterpixel oder als ein Gesamtpixel zu verarbeiten. Folglich verbessert die Identifikation der belegten Region, während dieselbe die Leistung der Weitenschrittvorrichtungseinheit **58** verbessert, ferner die Gesamtspeicherleistung, dadurch, daß dieselbe es ermöglicht, daß jene Pixelkoordinaten, die als vollständige Pixelaktualisierungen gehandhabt werden können, identifiziert werden können und schnell zu dem Speicher zum Verarbeiten übertragen werden können.

[0125] Es sei jedoch bemerkt, daß die vorliegende Erfindung nicht auf das Übertragen von Unterabtastinformationen als entweder eine vollständige Vier-Bit-Maske oder eine Einzelbitmaske begrenzt ist. Dieser Effekt ist lediglich eine Einschränkung der Speichersteuerung **60**, die bei einem Ausführungsbeispiel der Erfindung verwendet wird. Die Maske für jedes Pixel könnte direkt von der Weitenschrittvorrichtungslogik mit einer einzigen Übertragung gesendet werden, wenn dieselbe mit einer anderen Speichersteuerung verwendet würde, die den Empfang von Maskendaten mit mehr als einem aber weniger als allen Bitsätzen unterstützt.

[0126] Folglich wurde ein Verfahren und eine Vorrichtung beschrieben, die es ermöglichen, daß ein Antialiasing mit lediglich geringen Modifikationen an der existierenden Hardware unterstützt wird. Durch Unterteilen einer Abtastzeile in eine Mehrzahl von Unterabtastzeilen und Durchführen eines Kantenabschreitens entlang der Unterabtastzeilengrenzen können Unterabtastinformationen während eines herkömmlichen Kantenschrittprozesses unter Verwendung einer herkömmlichen Hardware ausfindig gemacht werden.

Patentansprüche

1. Verfahren zum Bestimmen eines Prozentsatzes mindestens eines Pixels (**73**) auf einem Graphiksystemanzeigebildschirm, das durch ein Primitiv (**74**) abgedeckt ist, das auf dem Anzeigebildschirm angezeigt werden soll, wobei das Verfahren folgende Schritte aufweist:
(a) Aufteilen des mindestens einen Pixels in eine Mehrzahl von Unterabtastzeilen, die eine Mehrzahl von Unterabtastpunkten für das mindestens eine Pixel definieren, wobei jeder der Mehrzahl von Unterabtastpunkten auf einer entsprechenden der Mehrzahl von Unterabtastzeilen positioniert ist;
(b) Schreiten zwischen der Mehrzahl von Unterabtastzeilen des mindestens einen Pixels (**73**);
(c) Bestimmen bei jeder der Mehrzahl von Unterabtastzeilen, ob der entsprechende Unterabtastpunkt auf der einen der Mehrzahl von Unterabtastzeilen durch das Primitiv (**74**) abgedeckt ist; und
(d) Nähern des Prozentsatzes des mindestens einen Pixels, das durch das Primitiv (**74**) abgedeckt ist, basierend auf der Anzahl der Mehrzahl von Unterabtastpunkten des mindestens einen Pixels (**73**), die durch das Primitiv (**74**) abgedeckt wird.

2. Verfahren gemäß Anspruch 1, bei dem das Primitiv (**74**) eine Mehrzahl von Kanten aufweist, und bei

dem der Schritt (c) folgende Schritte aufweist:
Bestimmen für jede der Mehrzahl von Kanten des Primitivs (**74**) von x- und y-Koordinaten jedes Schnittpunktes, bei dem eine der Mehrzahl von Kanten des Primitivs die eine der Mehrzahl von Unterabtastzeilen kreuzt; und
Bestimmen ansprechend auf die x- und y-Koordinaten jedes Schnittpunktes und die x- und y-Koordinaten des Unterabtastpunkts, der auf der einen der Mehrzahl von Unterabtastzeilen positioniert ist, ob der Unterabtastpunkt, der auf einer der Mehrzahl von Unterabtastzeilen positioniert ist, durch das Primitiv (**74**) abgedeckt ist.

3. Verfahren gemäß Anspruch 2, bei dem der Schritt des Bestimmens der x- und y-Koordinaten jedes Schnittpunktes ferner folgende Schritte aufweist:
Bestimmen mit Software von Startkoordinaten und Neigungen für jede Kante des Primitivs;
Schreiten entlang jeder der Mehrzahl von Kanten beginnend bei den Startkoordinaten für die eine der Mehrzahl von Kanten; und
Bestimmen der x-Koordinate für jede der Kanten bei jeder y-Koordinate, die einer der Mehrzahl von Unterabtastzeilen entspricht, ansprechend auf eine Neigung der Kante.

4. Verfahren gemäß einem beliebigen der Ansprüche 1–3, das ferner folgende Schritte aufweist:
Identifizieren, für die Mehrzahl der Unterabtastzeilen, von Pixeln (**153**, **154**) in dem Primitiv (**150**), bei denen alle jeweiligen Unterabtastungen jeder der Unterabtastzeilen durch das Primitiv (**150**) abgedeckt sind; und
Übertragen der Pixel, bei denen alle der Mehrzahl von Unterabtastpunkten durch das Primitiv abgedeckt sind, zu einer Speichersteuerung (**60**) mit einer einzigen Übertragungsoperation.

5. Verfahren gemäß einem beliebigen der Ansprüche 1–4, bei dem die Koordinaten der Unterabtastungen auf der Unterabtastzeile derart ausgewählt werden, daß die Unterabtastungen eine nicht-regelmäßige Struktur in dem Pixel bilden.

6. Kantenschrittvorrichtung (**52**) bei einem Graphiksystem (**10**), das ein Primitiv (**74**) wiedergeben kann, zum Erzeugen von Koordinatendaten zum Anzeigen von Pixeln auf einem Anzeigegerät, die das Primitiv (**74**) darstellen, wobei die Kantenschrittvorrichtung (**52**) Kantengleichungsdaten (**40**, **42**, **44**) empfängt, die ein erstes und ein zweites Paar von Endpunktkoordinaten (V_0, V_1) und einen Neigungswert (**44**) für mindestens eine Kante des Primitivs, das wiedergegeben werden soll, aufweisen, wobei die Kantenschrittvorrichtung (**52**) folgendes Merkmal aufweist:
ein Paar von Addierern (**86**, **98**), einer für X-Koordinaten- und einer für Y-Koordinaten-Operationen, zum Inkrementieren einer der ersten Endpunktkoordinaten durch Inkrementwerte, um Kanten-x- bzw. Kanten-y-Koordinaten der mindestens einen Kante des Primitivs, das wiedergegeben wird, zu liefern, wobei der Inkrementwert kleiner als eine Größe eines Anzeigepixels bei einer Betriebsart ist.

7. Kantenschrittvorrichtung (**52**) gemäß Anspruch 6, bei der die Inkrementwerte aus einer ersten Gruppe von Inkrementwerten ausgewählt (**81**) werden, wenn die Kantenschrittvorrichtung in einer Aliasedbetriebsart arbeitet, und dieselben aus einer zweiten Gruppe von Inkrementwerten ausgewählt werden, wenn die Kantenschrittvorrichtung in einer Antialiasedbetriebsart arbeitet.

8. Kantenschrittvorrichtung (**52**) gemäß Anspruch 7, bei der jedes Pixel (**99**) des Primitivs in eine Mehrzahl von Unterabtastzeilen aufgeteilt wird, wenn die Kantenschrittvorrichtung (**52**) in der Antialiasedbetriebsart arbeitet, und bei der die zweite Gruppe von Inkrementwerten Inkremente zum Schreiten zwischen der Mehrzahl von Unterabtastzeilen aufweist.

9. Kantenschrittvorrichtung gemäß einem beliebigen der Ansprüche 6–8, die ferner eine Einrichtung zum Bestimmen eines Prozentsatzes (**126**) jedes Pixels aufweist, das durch das Primitiv abgedeckt ist.

10. Kantenschrittvorrichtung gemäß Anspruch 9, bei der die Einrichtung zum Bestimmen eines Prozentsatzes jedes Pixel, das durch ein Primitiv abgedeckt ist, ferner folgende Merkmale aufweist:
eine Einrichtung (**21**) zum Bestimmen für jede Kante einer x-Richtung, die eine Richtung hin zu einer gegenüberliegenden Kante des Primitivs anzeigt; und
einen Komparator (**89**), der gekoppelt ist, um die Kanten-x- und Kanten-y-Koordinaten für jede Unterabtastzeile zu empfangen, und um die Kanten-x- und Kanten-y-Koordinaten mit Koordinaten einer Unterabtastung zu vergleichen, die auf der Unterabtastzeile positioniert ist, um ansprechend auf die x-Richtung zu bestimmen, ob die Unterabtastung durch das Primitiv abgedeckt ist.

Es folgen 16 Blatt Zeichnungen

Anhängende Zeichnungen

FIG. 1

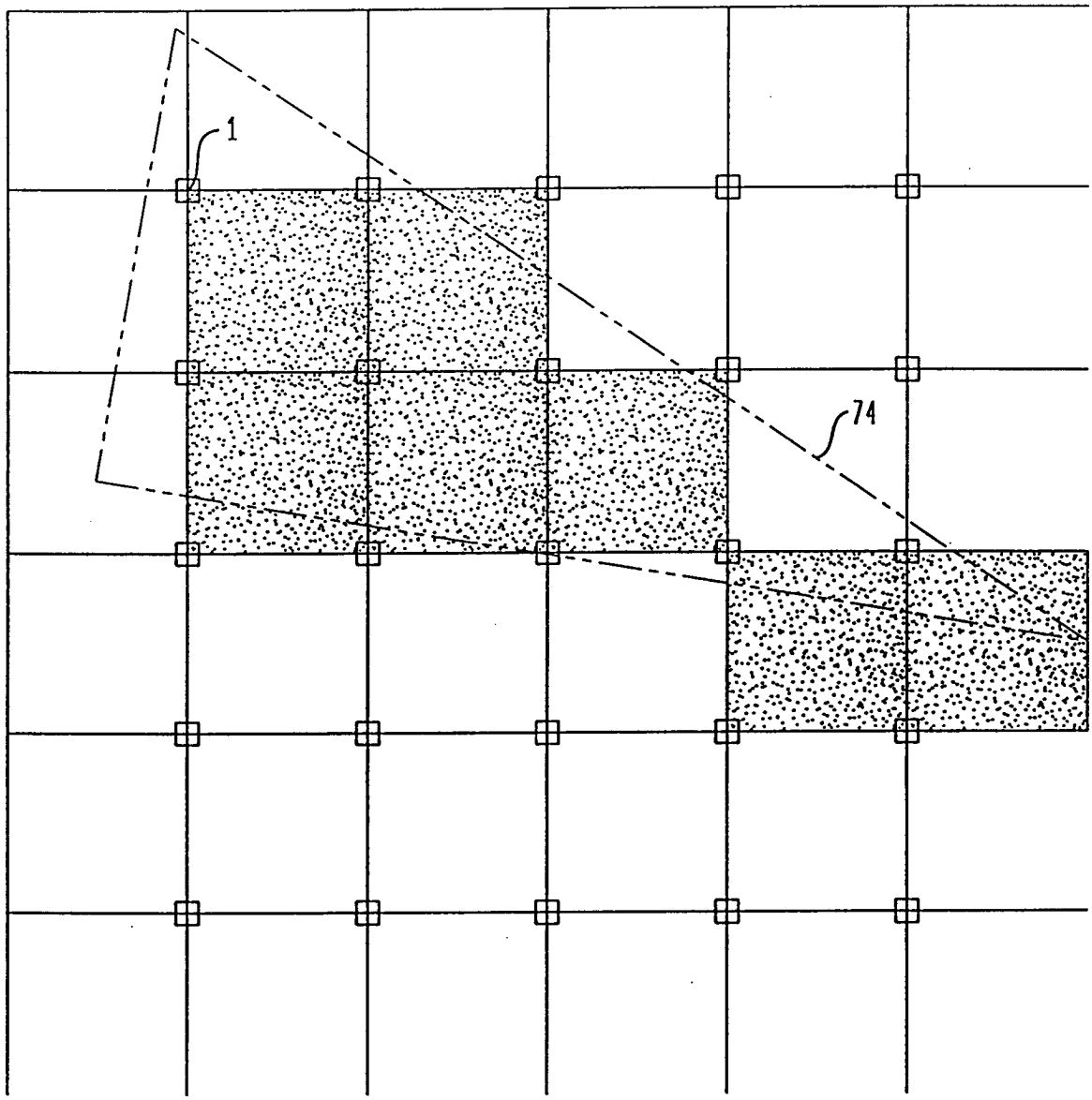


FIG. 2

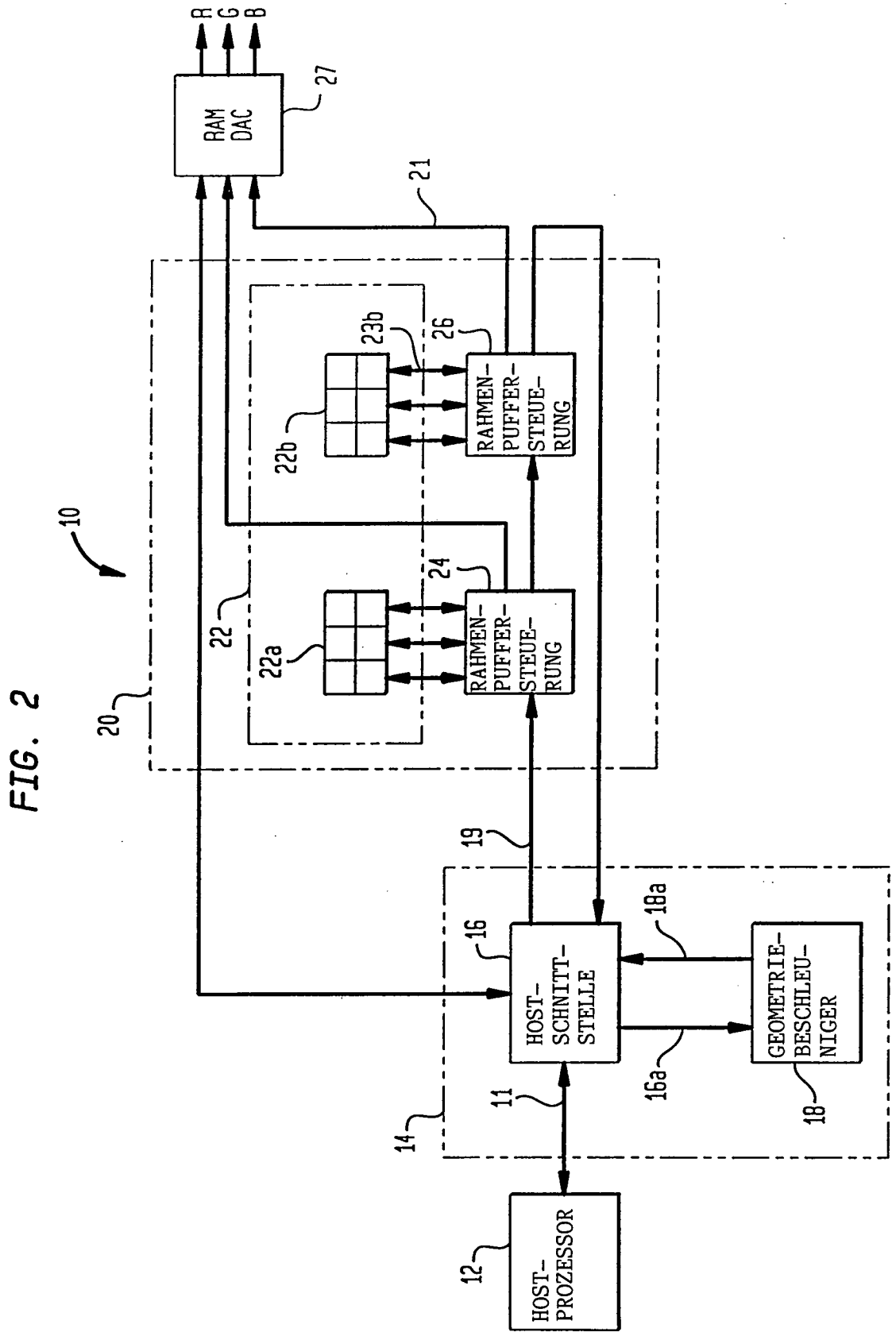

R
G
B
RAM DAC
27
21
10
20
22
22a
22b
23b
24
26
RAHMEN-PUFFER-STEUE-RUNG
RAHMEN-PUFFER-STEUE-RUNG
19
16
18a
HOST-SCHNITT-STELLE
GEOMETRIE-BESCHLEU-NIGER
16a
18
11
14
12
HOST-PROZESSOR

FIG. 3

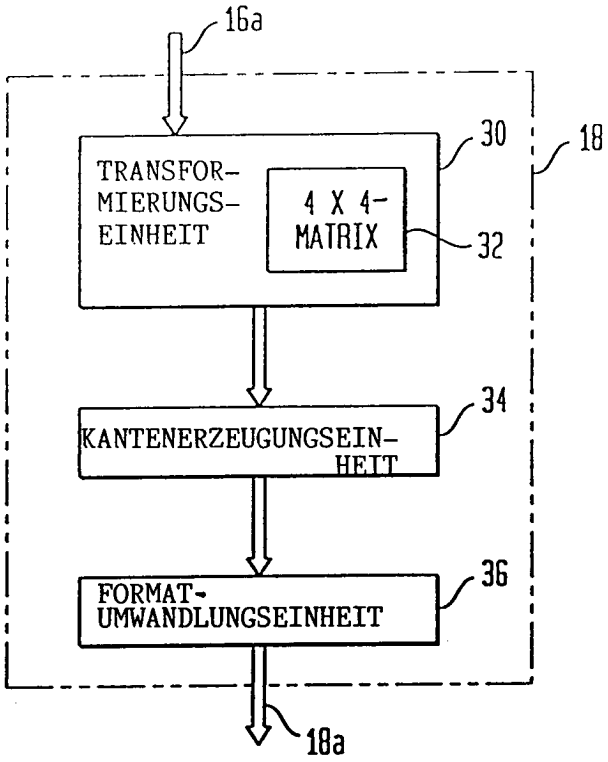


FIG. 4

$$[x'y'z'w'] = [x\ y\ z\ w] \times \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{bmatrix}$$

HOMOGENE GERÄTEKOORDINATEN | OBJEKTKOORDINATEN | TRANSFORMIERUNGSMATRIX

FIG. 4A

```
   BERECHNEN DER PERSPEKTIVEGETEILTEN KOORDINATEN
   (NUR WENN DIE PERSPEKTIVE EINGESTELLT IST):

40 { v0.x = v0.x • v0.wrecip   v0.y = v0.y • v0.wrecip   v0.z = v0.z • v0.wrecip
     v1.x = v1.x • v1.wrecip   v1.y = v1.y • v1.wrecip   v1.z = v1.z • v1.wrecip
     v2.x = v2.x • v2.wrecip   v2.y = v2.y • v2.wrecip   v2.z = v2.z • v2.wrecip

   BERECHNEN ALLER KANTENDELTAS:

     mnx2 = v0.x - v2.x/*KANTE1dx*/  nx1 = v1.x - v0.x/*KANTE2dx*/
     dxe3 = v2.x - v1.x/*KANTE3dx*/
     mny2 = v0.y - v2.y/*KANTE1dy*/  ny1 = v1:y - v0.y/*KANTE2dy*/
     dye3 = v2.y - v1.y/*KANTE3dy*/
42 { mnz2 = v0.z - v2.z/*KANTE1dz*/  nz1 = v1.z - v0.z/*KANTE2dz*/
     mnrot2 = v0.rot-v2.rot/*KANTE1drot*/ nrot1 = v1.rot-v0.rot/*KANTE2drot*/
     mngrün2 = v0.grün-v2.grün/*KANTE1dgrün*/
       ngrün1 = v1.grün-v0.grün/*KANTE2dgrün*/
     mnblau2 = v0.blau-v2.blau(KANTE1dblau)
       nblau1 =,v1.blau-v0.blau(KANTE2dblau)

     BERECHNEN ALLER KANTENNEIGUNGEN:
44 {   KANTE1_NEIGUNG = mnx2 + mny2
       KANTE2_NEIGUNG = nx1 + ny1
       KANTE3_NEIGUNG = dxe3 + dye3
```

FIG. 5

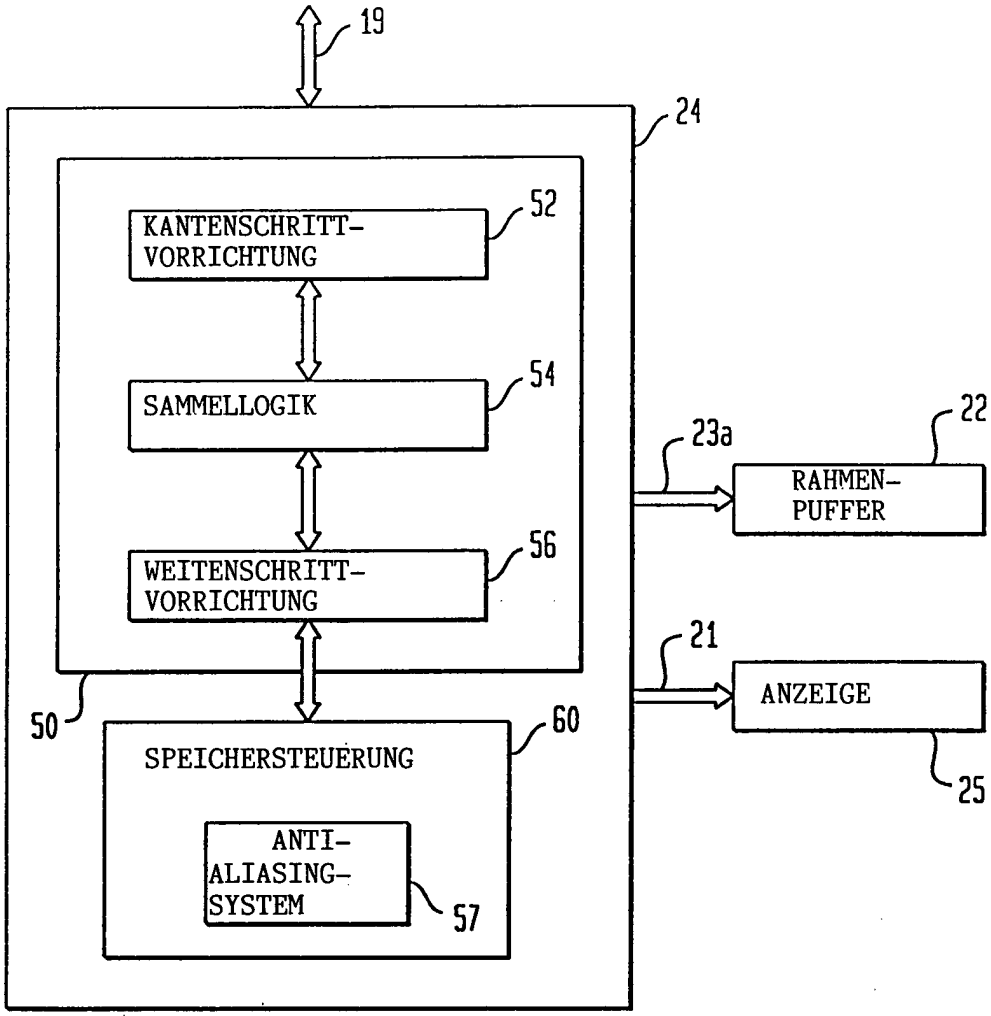

19
24
52
KANTENSCHRITT-
VORRICHTUNG
54
SAMMELLOGIK
56
WEITENSCHRITT-
VORRICHTUNG
50
60
SPEICHERSTEUERUNG
ANTI-
ALIASING-
SYSTEM
57
23a
22
RAHMEN-
PUFFER
21
ANZEIGE
25

FIG. 6

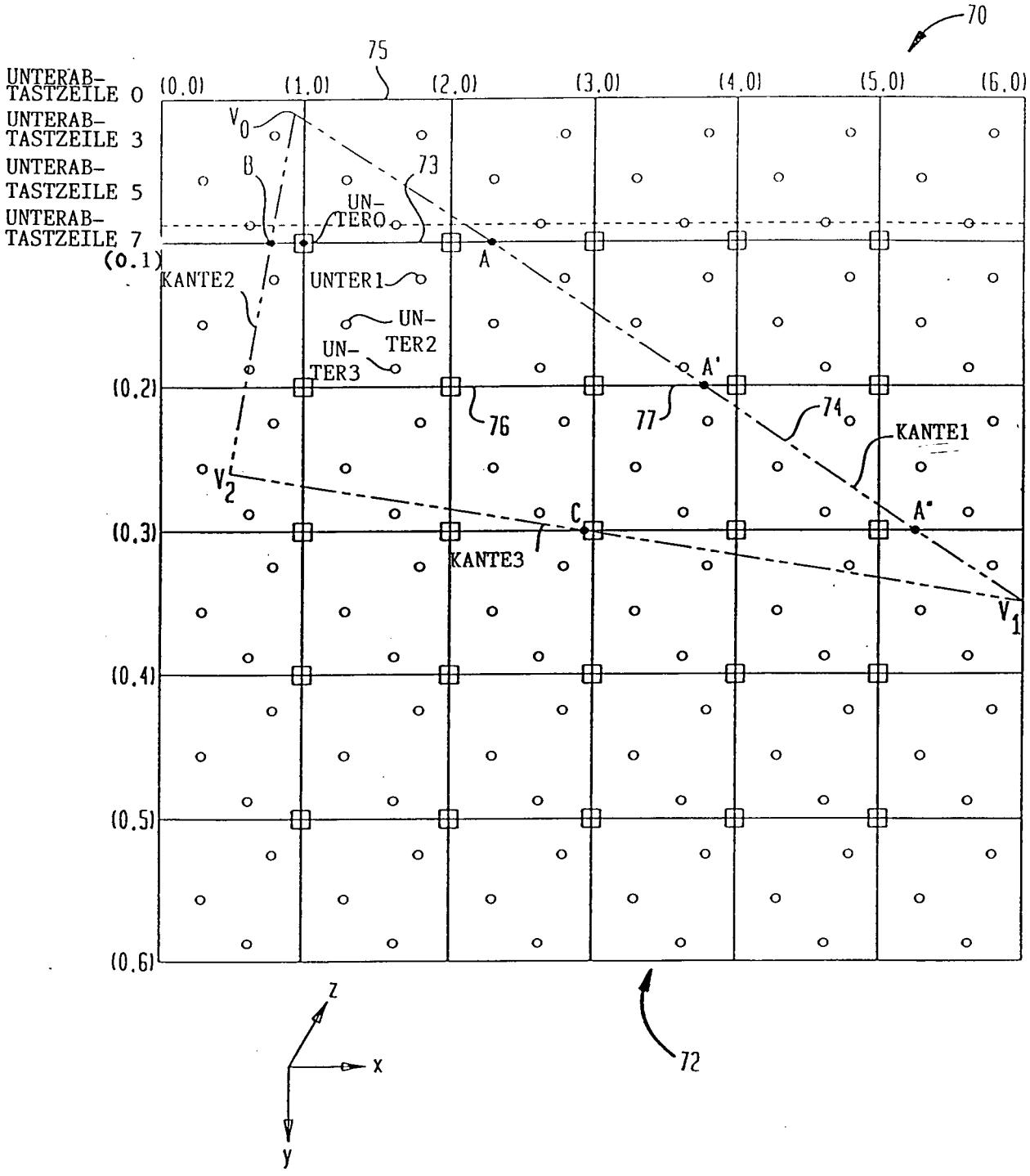

70
75
UNTERAB-
TASTZEILE 0
UNTERAB-
TASTZEILE 3
UNTERAB-
TASTZEILE 5
UNTERAB-
TASTZEILE 7
(0.0)
(1.0)
(2.0)
(3.0)
(4.0)
(5.0)
(6.0)
(0.1)
(0.2)
(0.3)
(0.4)
(0.5)
(0.6)
V0
B
73
UN-
TERO
A
KANTE2
UNTER1
UN-
TER2
UN-
TER3
A'
76
77
74
KANTE1
V2
C
A"
KANTE3
V1
z
x
y
72

FIG. 7

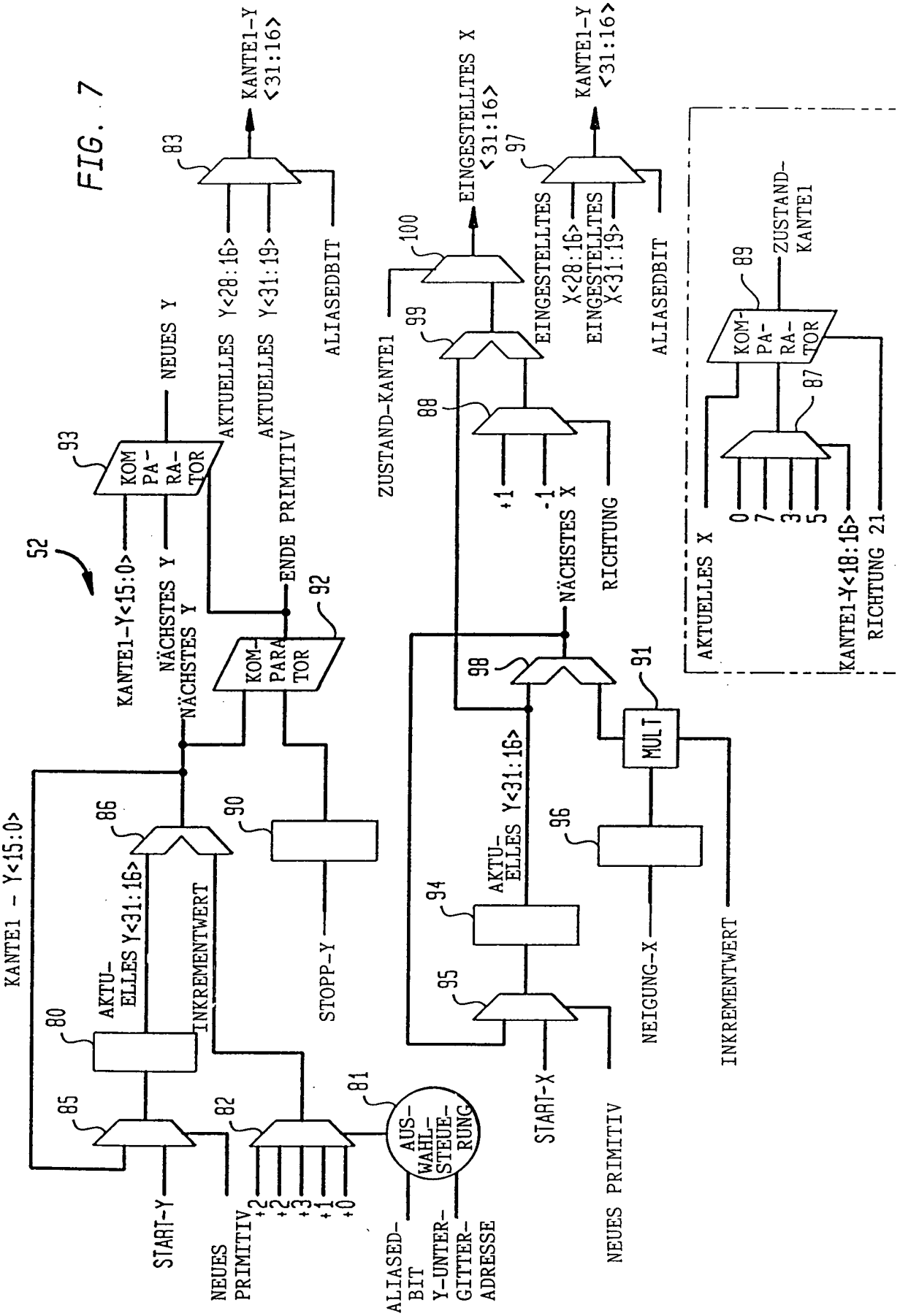

52
KANTE1-Y<15:0>
NÄCHSTES Y
NÄCHSTES Y
ENDE PRIMTIV
AKTUELLES Y<28:16>
AKTUELLES Y<31:19>
ALIASEDBIT
KANTE1-Y <31:16>
NEUES Y
KOM PA- RA- TOR
KOM- PARA TOR
83
93
92
KANTE1 - Y<15:0>
AKTU- ELLES Y<31:16>
INKREMENTWERT
STOPP-Y
86
90
80
85
82
81
START-Y
NEUES PRIMITIV
+2
+2
+3
+1
+0
ALIASED- BIT
Y-UNTER- GITTER- ADRESSE
AUS- WAHL- STEUE- RUNG
ZUSTAND-KANTE1
100
99
88
+1
-1
NÄCHSTES X
RICHTUNG
EINGESTELLTES X <31:16>
EINGESTELLTES X<28:16>
EINGESTELLTES X<31:19>
ALIASEDBIT
KANTE1-Y <31:16>
97
98
91
MULT
AKTU- ELLES Y<31:16>
94
95
96
START-X
NEIGUNG-X
NEUES PRIMITIV
INKREMENTWERT
AKTUELLES X
0
7
3
5
KANTE1-Y<18:16>
RICHTUNG 21
KOM- PA- RA- TOR
89
87
ZUSTAND- KANTE1

FIG. 8

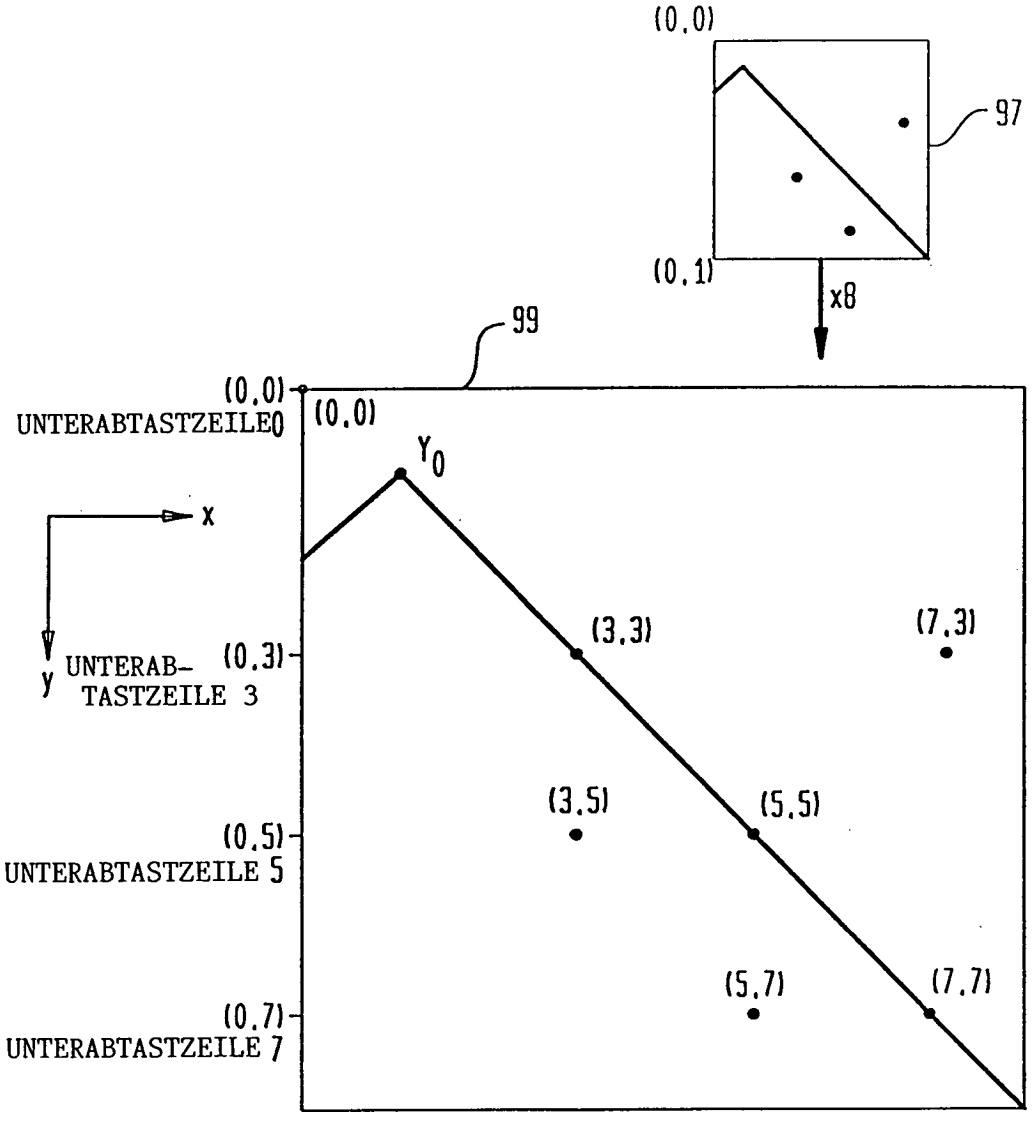
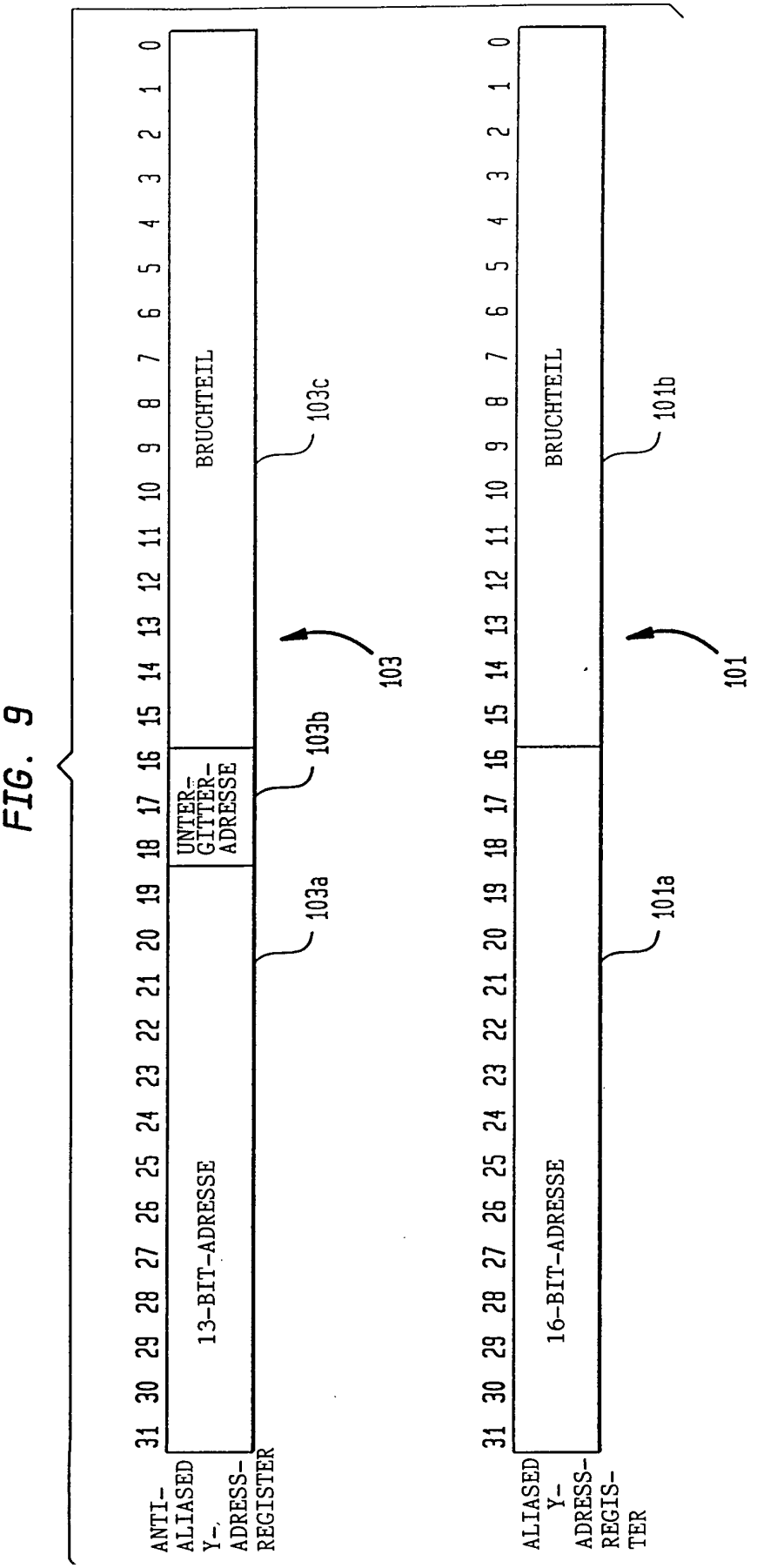

(0,0)
97
(0,1)
x8
99
(0,0)
UNTERABTASTZEILE 0
(0,0)
Y_0
x
y
UNTERAB-
TASTZEILE 3
(0,3)
(3,3)
(7,3)
(3,5)
(5,5)
(0,5)
UNTERABTASTZEILE 5
(5,7)
(7,7)
(0,7)
UNTERABTASTZEILE 7

FIG. 9


ANTI-ALIASED Y-ADRESS-REGISTER
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
13-BIT-ADRESSE
UNTER-GITTER-ADRESSE
BRUCHTEIL
103a
103b
103c
103
ALIASED Y-ADRESS-REGISTER
16-BIT-ADRESSE
BRUCHTEIL
101a
101b
101

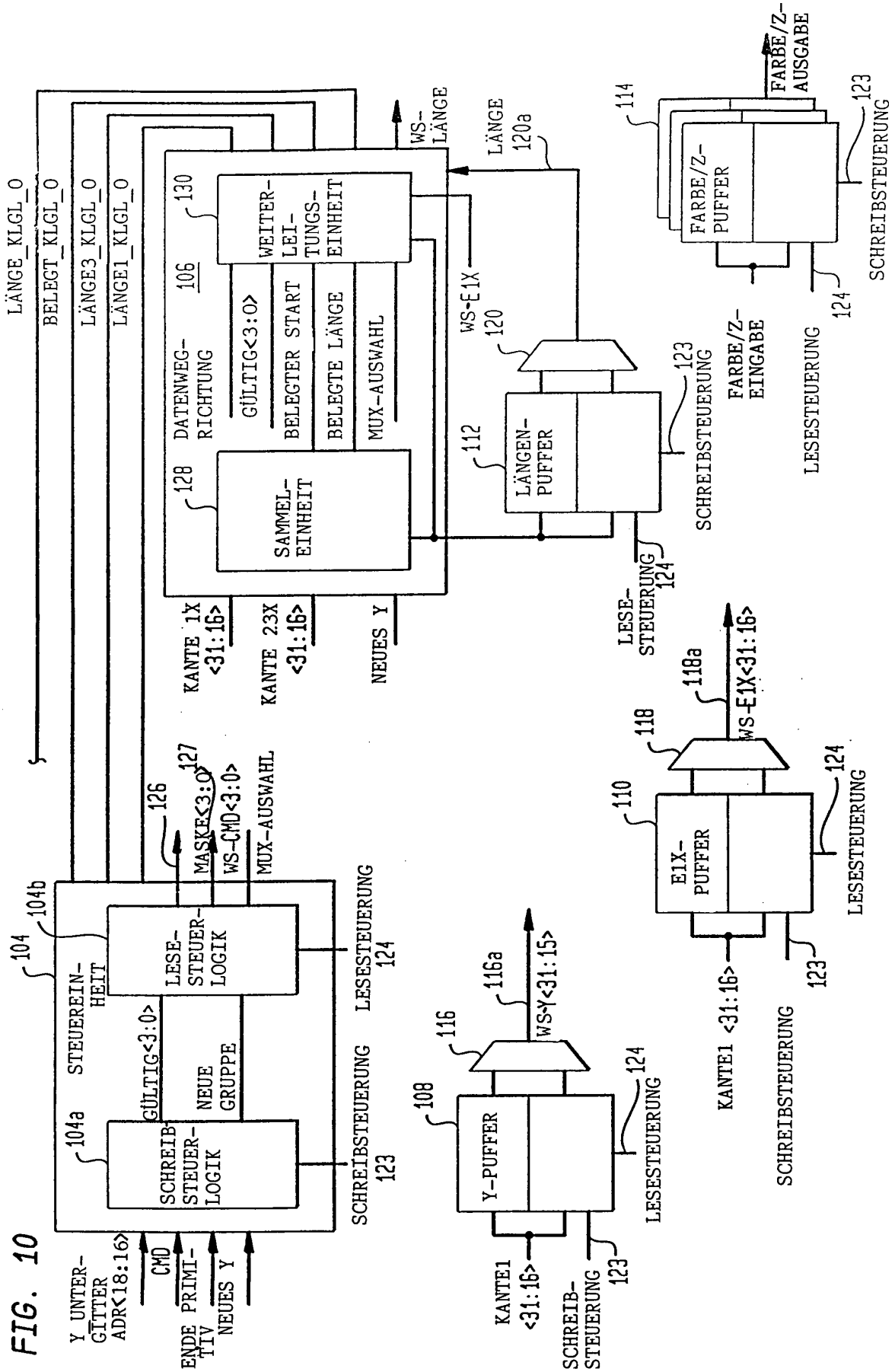

LÄNGE_KLGL_0
BELEGT_KLGL_0
LÄNGE3_KLGL_0
LÄNGE1_KLGL_0
WS-LÄNGE
LÄNGE
120a
WS-E1X
120
112
LÄNGEN-PUFFER
123
SCHREIBSTEUERUNG
FARBE/Z-EINGABE
LESESTEUERUNG
124
114
FARBE/Z-PUFFER
FARBE/Z-AUSGABE
SCHREIBSTEUERUNG
123
130
WEITER-LEI-TUNGS-EINHEIT
106
DATENWEG-RICHTUNG
GÜLTIG<3:0>
BELEGTER START
BELEGTE LÄNGE
MUX-AUSWAHL
128
SAMMEL-EINHEIT
KANTE 1X <31:16>
KANTE 2X <31:16>
NEUES Y
LESE-STEUERUNG
124
118a
WS-E1X<31:16>
118
110
E1X-PUFFER
124
LESESTEUERUNG
KANTE1 <31:16>
123
SCHREIBSTEUERUNG
116a
WS-Y<31:15>
116
108
Y-PUFFER
124
LESESTEUERUNG
KANTE1 <31:16>
SCHREIB-STEUERUNG
123
104
104b
STEUEREIN-HEIT
LESE-STEUER-LOGIK
126
127
MASKE<3:0>
WS-CMD<3:0>
MUX-AUSWAHL
LESESTEUERUNG
124
GÜLTIG<3:0>
NEUE GRUPPE
104a
SCHREIB-STEUER-LOGIK
SCHREIBSTEUERUNG
123
Y UNTER-GITTER ADR<18:16>
CMD
PRIMI-TIV
ENDE
NEUES Y

FIG. 10

FIG. 11

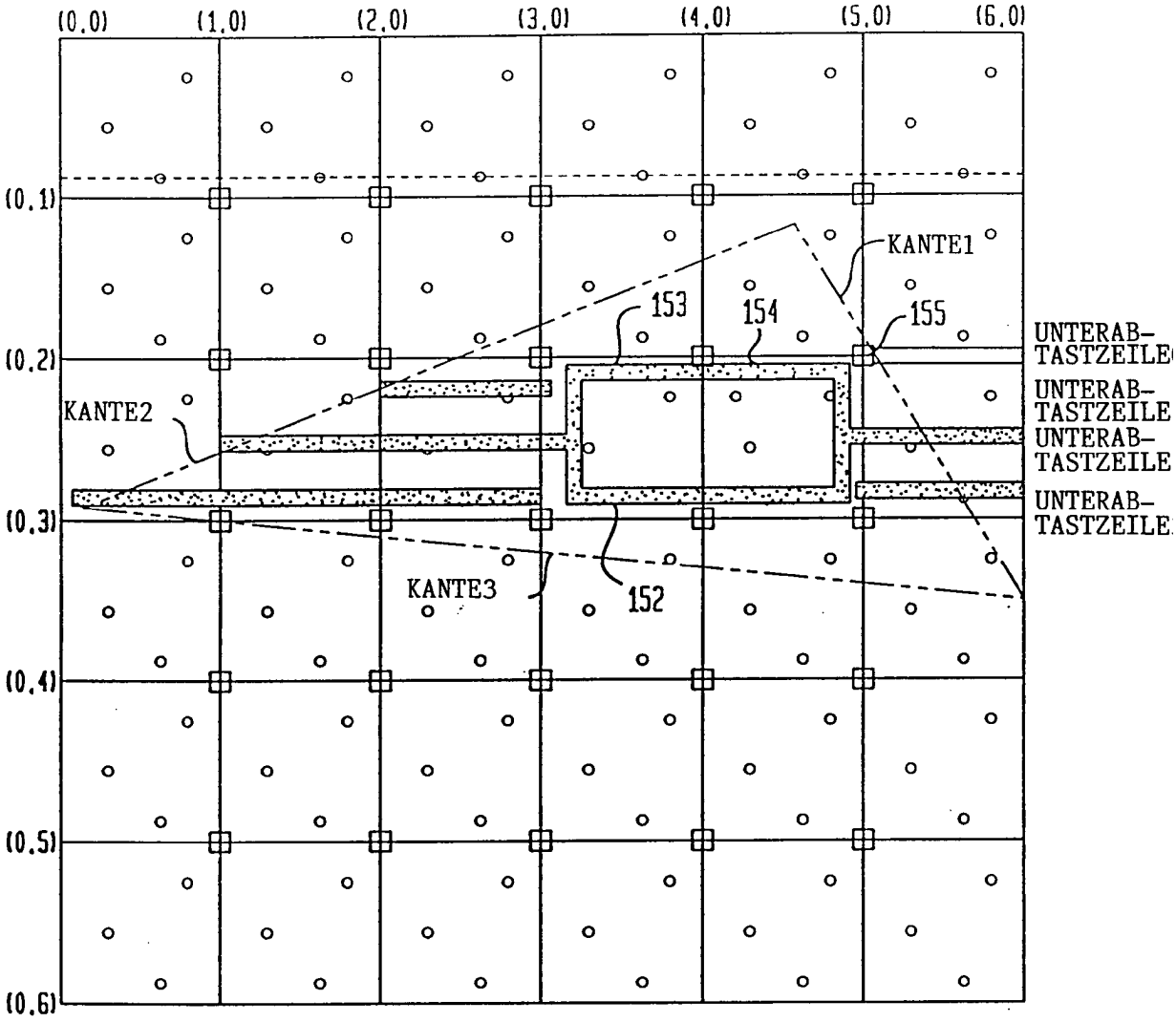

(0,0)
(1,0)
(2,0)
(3,0)
(4,0)
(5,0)
(6,0)
(0,1)
(0,2)
(0,3)
(0,4)
(0,5)
(0,6)
KANTE1
KANTE2
KANTE3
153
154
155
152
UNTERAB-
TASTZEILE
UNTERAB-
TASTZEILE
UNTERAB-
TASTZEILE
UNTERAB-
TASTZEILE

FIG. 12

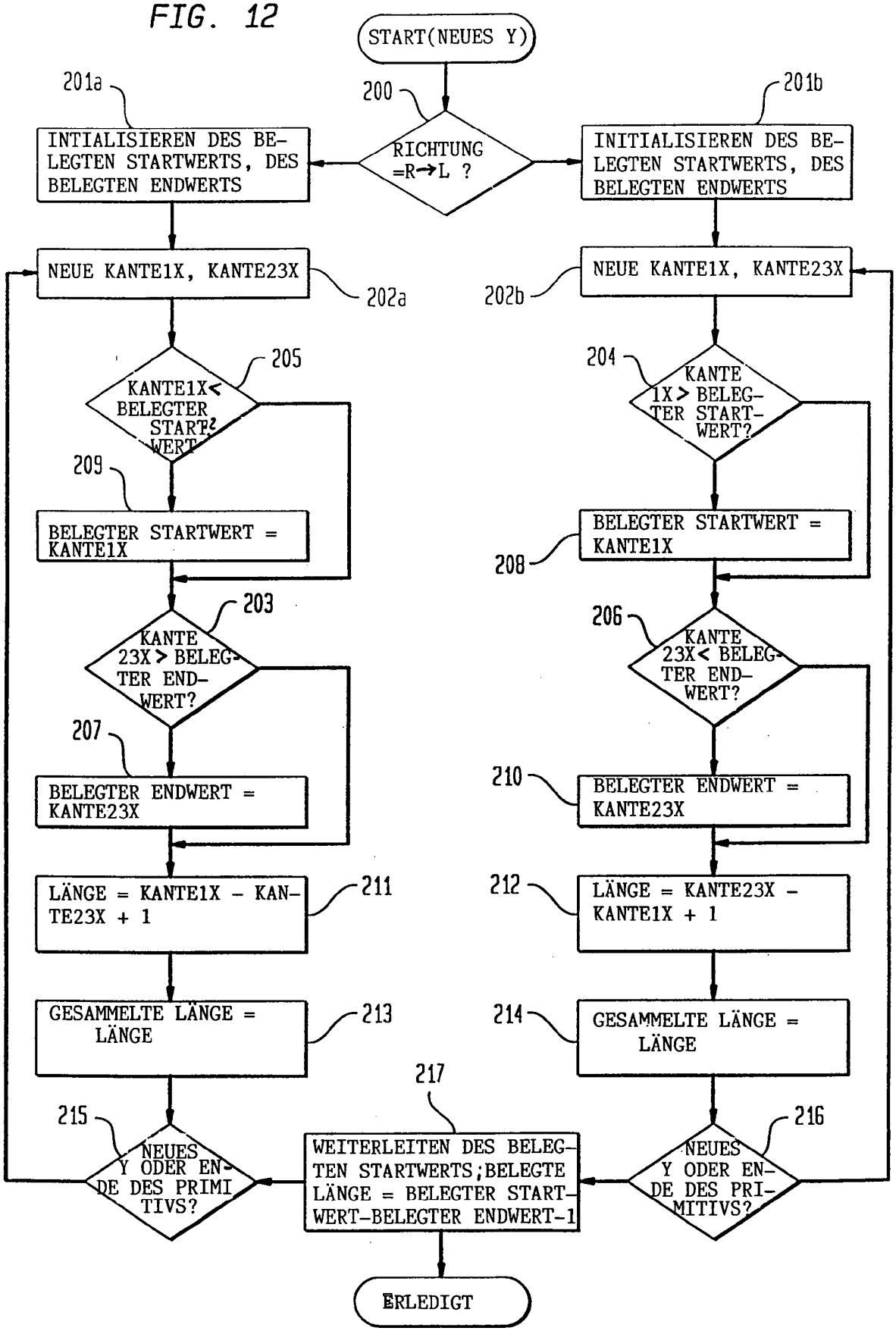

START(NEUES Y)
200
RICHTUNG =R→L ?
201a
INTIALISIEREN DES BELEGTEN STARTWERTS, DES BELEGTEN ENDWERTS
201b
INITIALISIEREN DES BELEGTEN STARTWERTS, DES BELEGTEN ENDWERTS
NEUE KANTE1X, KANTE23X
202a
NEUE KANTE1X, KANTE23X
202b
205
KANTE1X< BELEGTER START-WERT?
204
KANTE 1X > BELEGTER START-WERT?
209
BELEGTER STARTWERT = KANTE1X
208
BELEGTER STARTWERT = KANTE1X
203
KANTE 23X > BELEGTER END-WERT?
206
KANTE 23X < BELEGTER END-WERT?
207
BELEGTER ENDWERT = KANTE23X
210
BELEGTER ENDWERT = KANTE23X
LÄNGE = KANTE1X - KANTE23X + 1
211
LÄNGE = KANTE23X - KANTE1X + 1
212
GESAMMELTE LÄNGE = LÄNGE
213
GESAMMELTE LÄNGE = LÄNGE
214
217
215
NEUES Y ODER ENDE DES PRIMITIVS?
WEITERLEITEN DES BELEGTEN STARTWERTS;BELEGTE LÄNGE = BELEGTER START-WERT-BELEGTER ENDWERT-1
216
NEUES Y ODER ENDE DES PRIMITIVS?
ERLEDIGT

FIG. 13

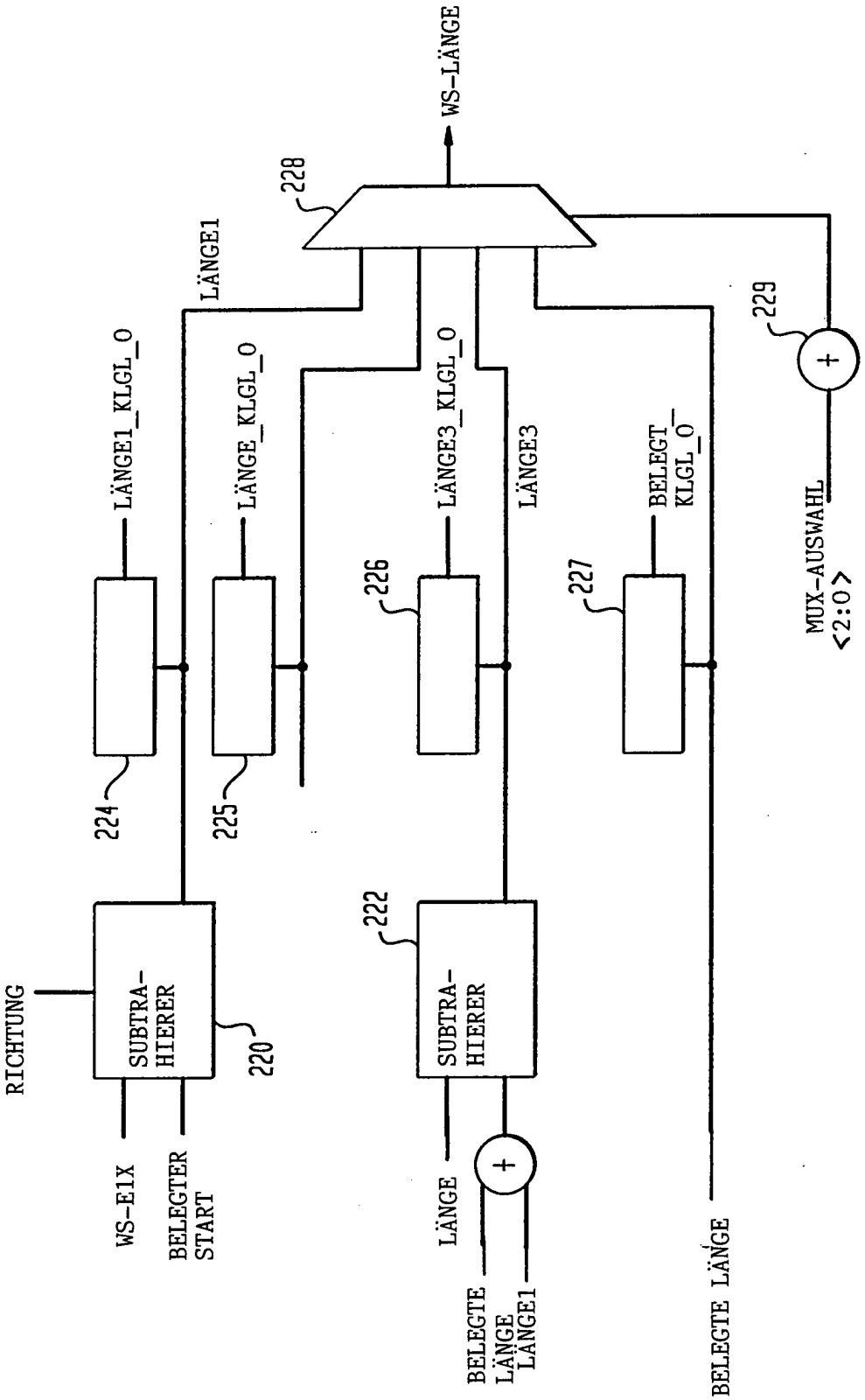

WS-LÄNGE
228
LÄNGE1
LÄNGE1_KLGL_0
LÄNGE_KLGL_0
LÄNGE3_KLGL_0
LÄNGE3
BELEGT_KLGL_0
229
MUX-AUSWAHL <2:0>
224
225
226
227
RICHTUNG
SUBTRA-HIERER
220
222
SUBTRA-HIERER
WS-E1X
BELEGTER START
LÄNGE
BELEGTE LÄNGE LÄNGE1
BELEGTE LÄNGE

FIG. 14

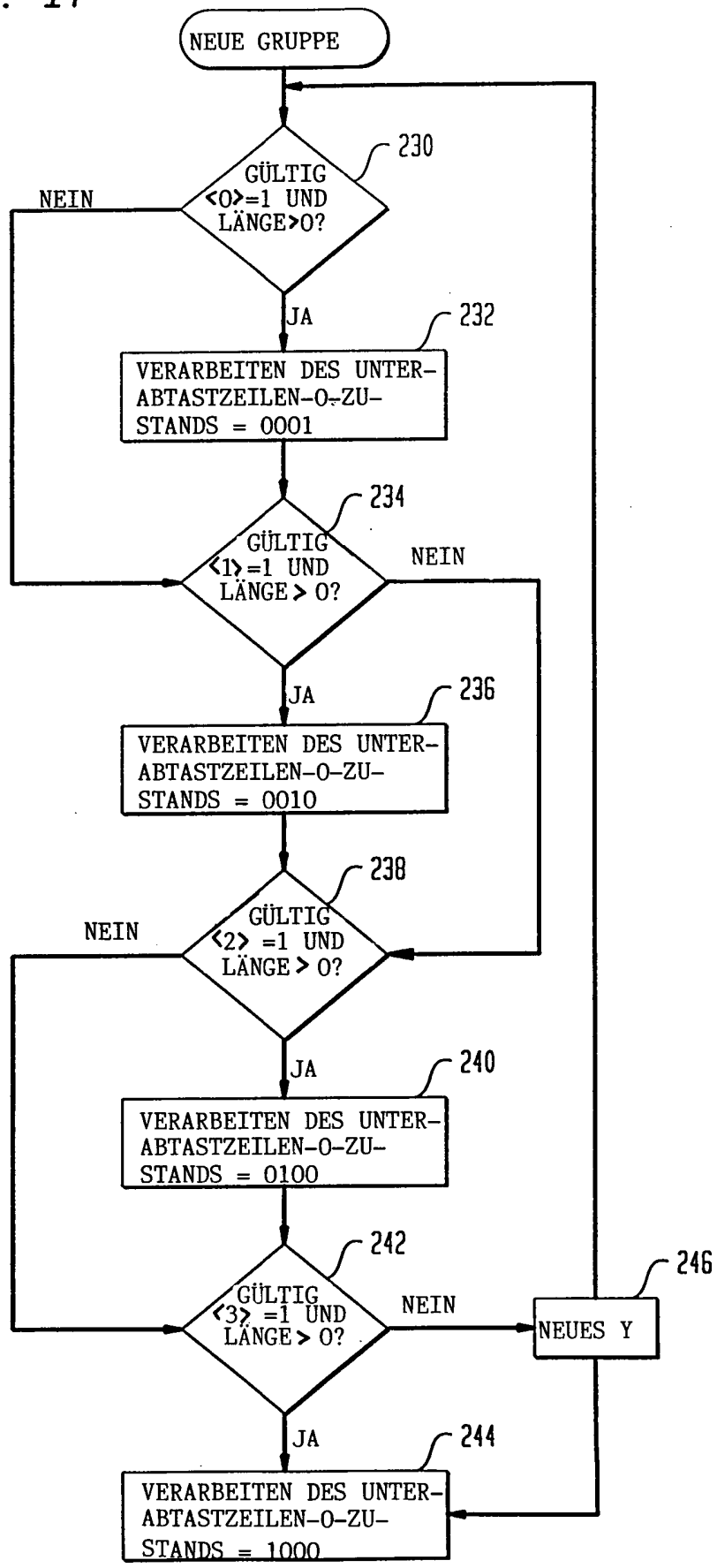
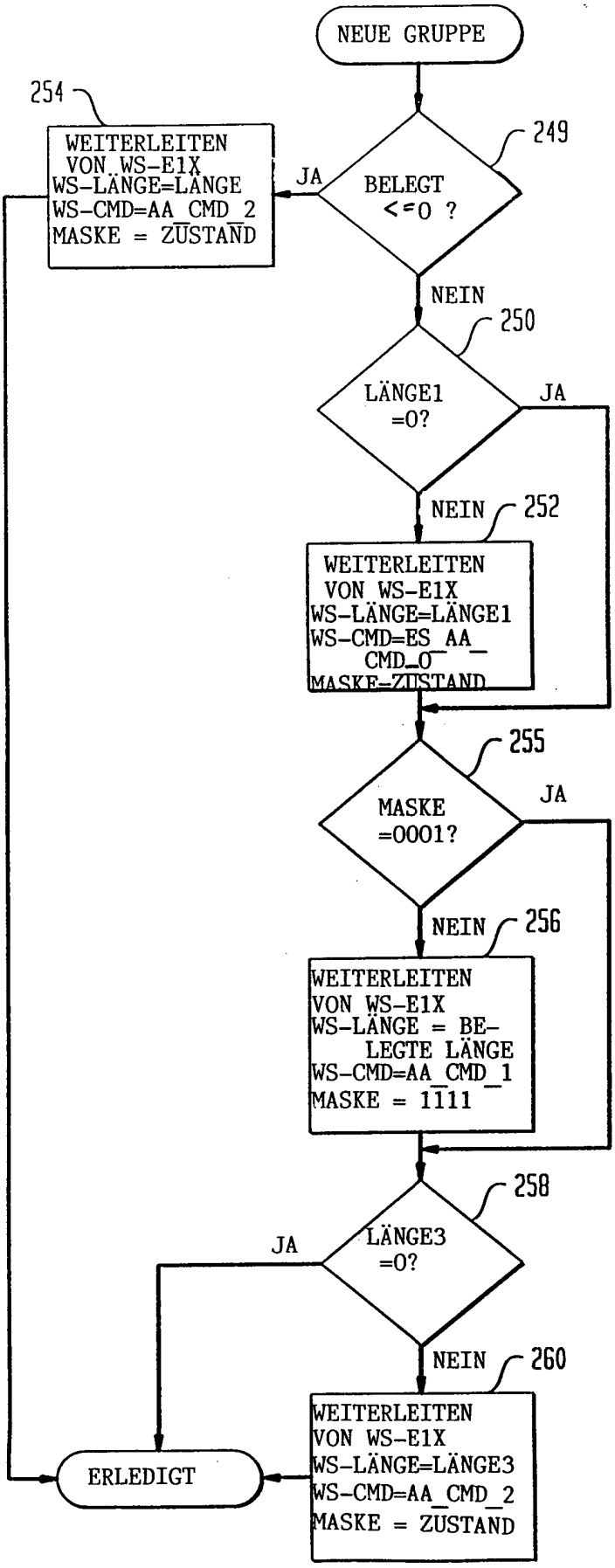

NEUE GRUPPE
230
GÜLTIG <0>=1 UND LÄNGE>0?
NEIN
JA
232
VERARBEITEN DES UNTER-ABTASTZEILEN-0-ZU-STANDS = 0001
234
GÜLTIG <1>=1 UND LÄNGE > 0?
NEIN
JA
236
VERARBEITEN DES UNTER-ABTASTZEILEN-0-ZU-STANDS = 0010
238
GÜLTIG <2> =1 UND LÄNGE > 0?
NEIN
JA
240
VERARBEITEN DES UNTER-ABTASTZEILEN-0-ZU-STANDS = 0100
242
GÜLTIG <3> =1 UND LÄNGE > 0?
NEIN
246
NEUES Y
JA
244
VERARBEITEN DES UNTER-ABTASTZEILEN-0-ZU-STANDS = 1000

FIG. 15


NEUE GRUPPE
249
BELEGT <=0 ?
JA
254
WEITERLEITEN VON WS-E1X
WS-LÄNGE=LÄNGE
WS-CMD=AA_CMD_2
MASKE = ZUSTAND
NEIN
250
LÄNGE1 =0?
JA
NEIN
252
WEITERLEITEN VON WS-E1X
WS-LÄNGE=LÄNGE1
WS-CMD=ES_AA_CMD_0
MASKE=ZUSTAND
255
MASKE =0001?
JA
NEIN
256
WEITERLEITEN VON WS-E1X
WS-LÄNGE = BELEGTE LÄNGE
WS-CMD=AA_CMD_1
MASKE = 1111
258
LÄNGE3 =0?
JA
NEIN
260
WEITERLEITEN VON WS-E1X
WS-LÄNGE=LÄNGE3
WS-CMD=AA_CMD_2
MASKE = ZUSTAND
ERLEDIGT

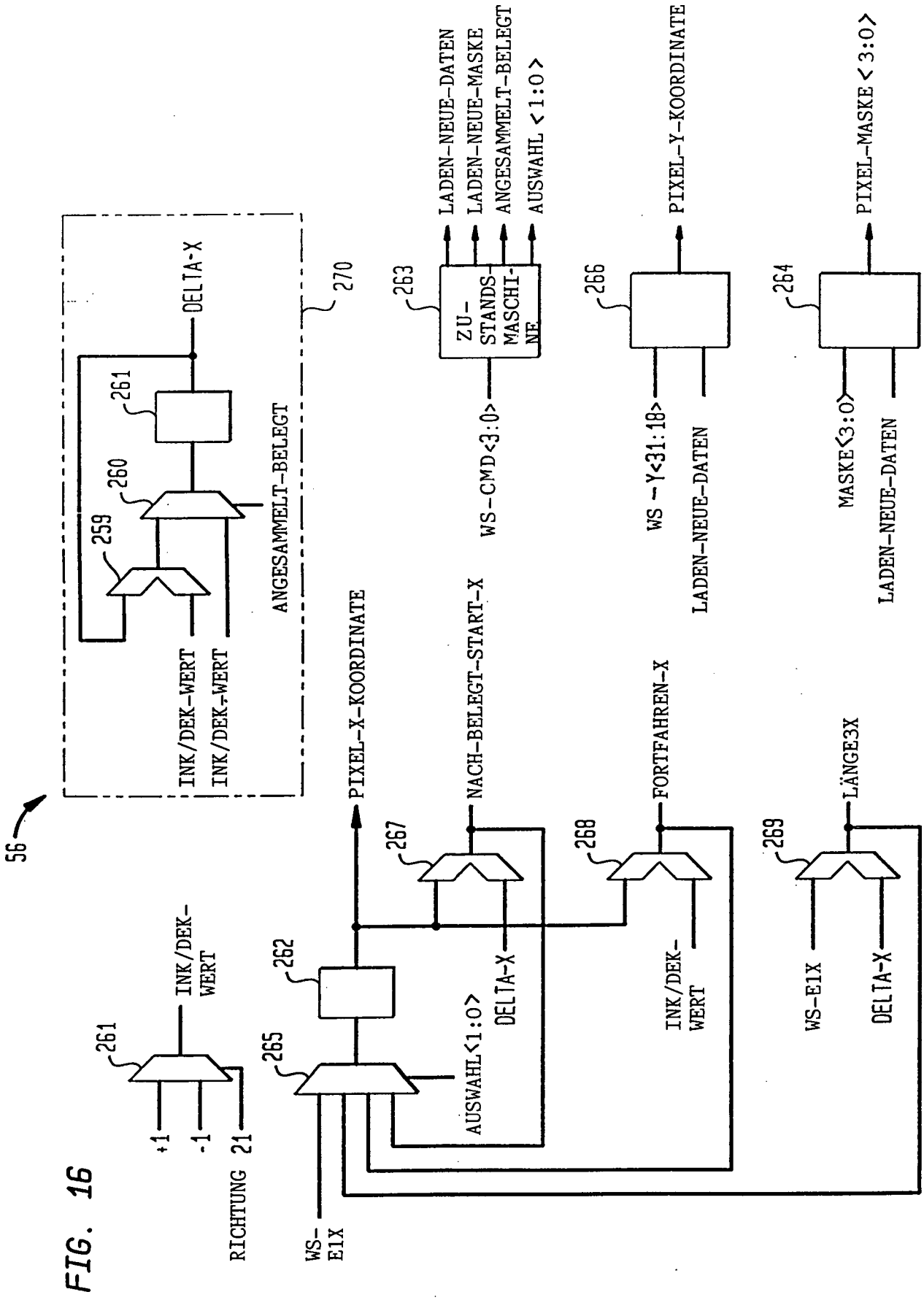

56
270
DELTA-X
261
260
259
ANGESAMMELT-BELEGT
INK/DEK-WERT
INK/DEK-WERT
263
ZU-STANDS-MASCHI-NE
LADEN-NEUE-DATEN
LADEN-NEUE-MASKE
ANGESAMMELT-BELEGT
AUSWAHL <1:0>
WS-CMD<3:0>
266
PIXEL-Y-KOORDINATE
WS -Y<31:18>
LADEN-NEUE-DATEN
264
PIXEL-MASKE <3:0>
MASKE<3:0>
LADEN-NEUE-DATEN
PIXEL-X-KOORDINATE
NACH-BELEGT-START-X
FORTFAHREN-X
LÄNGE3X
267
268
269
262
265
261
INK/DEK-WERT
+1
-1
RICHTUNG
21
AUSWAHL<1:0>
DELTA-X
INK/DEK-WERT
WS-E1X
DELTA-X
WS-E1X

FIG. 16