(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 696 34 073 T2** 2006.03.02

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 0 803 837 B1**
(21) Deutsches Aktenzeichen: **696 34 073.9**
(96) Europäisches Aktenzeichen: **96 116 705.3**
(96) Europäischer Anmeldetag: **17.10.1996**
(97) Erstveröffentlichung durch das EPA: **29.10.1997**
(97) Veröffentlichungstag
der Patenterteilung beim EPA: **22.12.2004**
(47) Veröffentlichungstag im Patentblatt: **02.03.2006**

(51) Int Cl.[8]: ***G06K 15/02*** (2006.01)

(30) Unionspriorität:
637060 **24.04.1996** **US**

(73) Patentinhaber:
Hewlett-Packard Development Co., L.P., Houston, Tex., US

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

(84) Benannte Vertragsstaaten:
DE, FR, GB

(72) Erfinder:
Campbell, Russell, Boise, Idaho 83706, US

(54) Bezeichnung: **Verfahren zum Drucken mit Überlagerungsmustern**

Anmerkung: Innerhalb von neun Monaten nach der Bekanntmachung des Hinweises auf die Erteilung des europäischen Patents kann jedermann beim Europäischen Patentamt gegen das erteilte europäische Patent Einspruch einlegen. Der Einspruch ist schriftlich einzureichen und zu begründen. Er gilt erst als eingelegt, wenn die Einspruchsgebühr entrichtet worden ist (Art. 99 (1) Europäisches Patentübereinkommen).

Die Übersetzung ist gemäß Artikel II § 3 Abs. 1 IntPatÜG 1991 vom Patentinhaber eingereicht worden. Sie wurde vom Deutschen Patent- und Markenamt inhaltlich nicht geprüft.

Beschreibung

GEBIET DER ERFINDUNG

[0001] Diese Erfindung bezieht sich auf digitale Drucker und insbesondere auf ein Verfahren zum Betreiben eines digitalen Druckers, um ein gedrucktes Bild auf einem Druckmedium zu erzeugen.

HINTERGRUND DER ERFINDUNG

[0002] Digitale Drucker sind in der Lage, Vollfarbbilder mit hoher Qualität und Präzision zu erzeugen. Derartige Drucker werden durch ein Druckertreiberprogramm gesteuert, das eine Schnittstelle zwischen einem auf einem Hostprozessor laufenden Anwendungsprogramm und dem Drucker liefert.

[0003] Manche durch Drucker erzeugte Bilder umfassen Muster. Ein Muster kann man sich als einen Hintergrund oder eine Ausfüllung vorstellen. Ein Muster wird üblicherweise auf ein sogenanntes Objekt oder eine sogenannte Quelle angewendet, das bzw. die als Teil eines gedruckten Bildes auf einem Druckmedium erscheint.

[0004] Bei einem Drucker werden Muster bzw. Strukturen durch eine Speicherbitabbildung definiert, die Daten speichert, die den „Ein"- oder „Aus"-Zustand einer Mehrzahl von Pixeln, die ein Muster bilden, definieren. In der Vergangenheit musste eine Speicherbitabbildung, die ein Muster definierte, über eine Breite, die der Breite einer zu druckenden Seite zugeordnet war, vorab repliziert werden. Der Grund dafür liegt darin, dass die Position einer zu musternden Quelle oder eines zu musternden Objekts nicht bekannt war. Damit also ein Muster bzw. eine Struktur an der richtigen Stelle innerhalb eines Objekts erschien, musste die gesamte Speicherbitabbildung, die diesem Muster bzw. dieser Struktur entsprach, die gesamte physische Seite der Breite nach überspannen und vertikal ein Minimum der Höhe eines Feldes bzw. einer Fliese der Struktur überspannen. Wenn ein Muster beispielsweise einen halben Zoll hoch war, mussten für einen 600dpi-Drucker (dpi = dot per inch = Punkte pro Zoll) etwa 300 Zeilen über die Seite hinweg repliziert werden, wobei jede Zeile eine Breite von etwa 6.000 Pixeln aufwies. Für einen 1.200dpi-Drucker waren die Speicheranforderungen zum Vorabreplizieren einer Bitabbildung sogar noch größer. Diese vorab replizierte Bitabbildung wird auch als vorab gefliestes Muster bezeichnet. Es muss wohl nicht extra erwähnt werden, dass das Vorabreplizieren einer Speicherbitabbildung, die einem Muster entspricht, viel Speicher erfordert. Als weiteres Beispiel musste eine Speicherbitabbildung für ein 48 Pixel breites und 48 Pixel hohes Muster von 288 Bytes auf etwa das Dreizehnfache bzw. 4K erweitert werden. Desgleichen musste ein Muster, das einen Zoll im Quadrat aufwies, von 45K auf etwa 370K erweitert werden.

[0005] Diese Erfindung entstand aus einem Erfordernis, die Art und Weise, auf die digitale Drucker gedruckte gemusterte bzw. strukturierte Bilder erzeugen, zu verbessern und den mit der Herstellung derartiger gemusterter Bilder verbundenen Speicherverbrauch zu verringern.

[0006] Die JP-A-8072317 bezieht sich auf ein Verfahren zum Steuern eines digitalen Druckers zum Erzeugen eines gedruckten Bildes, bei dem eine Seite in eine Mehrzahl feststehender Regionen in Form von parallelen Bändern unterteilt wird. Die Bänder werden so ausgewählt und geordnet, dass sie der Erzeugung von gerasterten Daten zur Ausgabe an eine Druckvorrichtung entsprechen.

[0007] Die vorliegende Erfindung beruht auf der Aufgabe, ein Verfahren zum Betreiben eines digitalen Druckers bereitzustellen, bei dem der Abschnitt des Musters, der tatsächlich benötigt wird, bereitgestellt wird und bei dem das Erfordernis, das gesamte Muster über die Breite der Seite vorab zu replizieren, beseitigt wird.

[0008] Diese Aufgabe wird durch das Verfahren gemäß Anspruch 1 gelöst.

BESCHREIBUNG DER ZEICHNUNGEN

[0009] Fig. 1 ist ein grobes Blockdiagramm eines Druckers, der angepasst ist, um die Erfindung auszuführen.

[0010] Fig. 2 ist ein Blockdiagramm, das eine Art und Weise, das erfundene Verfahren auszuführen, veranschaulicht.

[0011] Fig. 3 ist ein grobes Flussdiagramm, das ein bevorzugtes Verfahren zum Implementieren der Erfindung veranschaulicht.

[0012] Fig. 4 ist ein Diagramm, das ein Pixel-um-Pixel-Kopiermodell veranschaulicht, das verwendet wird, um das Konzept zu veranschaulichen, anhand dessen ein bevorzugtes 32Bit-Wort-Kopiermodell arbeitet.

[0013] Fig. 5 veranschaulicht diverse Definitionselemente, die beim Implementieren des 32Bit-Wort-Kopiermodells verwendet werden.

[0014] Fig. 6 ist ein grobes Flussdiagramm, das das 32Bit-Wort-Kopiermodell veranschaulicht.

[0015] Fig. 7 ist ein Flussdiagramm, das eine variable Einrichtungsprozedur darlegt.

[0016] Fig. 8 ist ein Flussdiagramm, das die soge-

nannten s-d-Fälle darlegt.

[0017] Fig. 9 ist ein Flussdiagramm, das einen sogenannten Gleichmäßige-Breite-Übertrag-Fall darlegt.

[0018] Fig. 10 ist ein Flussdiagramm, das einen sogenannten Nicht-Übertragsverarbeitungsfall darlegt.

[0019] Fig. 11 ist ein Flussdiagramm, das einen sogenannten Übertragsverarbeitungsfall darlegt.

AUSFÜHRLICHE BESCHREIBUNG DER ERFINDUNG

[0020] Diese Offenbarung der Erfindung wird zur Unterstützung der verfassungsmäßigen Zwecke der U.S.-Patentgesetze „zur Förderung des Fortschritts der Wissenschaft und nützlicher Fachgebiete" vorgelegt. U.S.-Verfassung, Artikel 1, Absatz 8.

DRUCKERSYSTEM

[0021] In der Fig. 1 umfasst ein digitaler Drucker **10** einen Prozessor **12**, eine Druckmaschine **14** und ein Eingangs-/Ausgangstor (I-/O-Tor) **16**, die alle durch einen Bus **18** verbunden sind. Die Druckmaschine **14** umfasst einen Laserdrucker, der im Betrieb bei einer konstanten Geschwindigkeit läuft und bei einer Geschwindigkeit, die mit seinem Betrieb Schritt hält, mit Videorasterdruckdaten versorgt werden muss.

[0022] Ein Direktzugriffsspeicher (RAM) **20** und ein Nur-Lese-Speicher (ROM) **22** sind ebenfalls mit dem Bus **18** verbunden und enthalten Prozeduren und andere notwendige Daten zum Betreiben des Druckers und insbesondere der Druckmaschine **14**. Der Betrieb des Druckers **10** wird in meiner U.S.-Patentschrift Nr. 5,479,587, deren Offenbarung durch Bezugnahme ausdrücklich in das vorliegende Dokument aufgenommen ist, ausführlicher dargelegt.

[0023] Die vorliegende Erfindung verringert den Verbrauch an Druckerspeicher, der mit dem Drucken eines gemusterten Bildes verbunden ist, indem sie bestimmt, wo eine bestimmte Struktur bzw. ein bestimmtes Muster erscheinen muss, den richtigen Umfang der benötigten Struktur erzeugt und anschließend diese Struktur an einer gewünschten Stelle auf ein Druckmedium wie z.B. Papier druckt, die gemäß einer Festlegung seitens des Benutzers an der Seite verankert ist. Der soeben verwendete Begriff „Anker" bedeutet einen benutzerdefinierten Startpunkt. Das heißt, dass diese Erfindung insofern ein zusätzliches Maß an Flexibilität liefert, als (ein) bestimmtes) Muster sozusagen irgendwo auf einer Seite, je nach einer Festlegung seitens des Benutzers, verankert oder angebracht werden könnte. Folglich werden lediglich benötigte Abschnitte einer Struktur erzeugt und auf dem Druckmedium wiedergegeben. Dadurch wird das Erfordernis, ein Muster über die gesamte Seite vorab zu replizieren, wie dies in der Vergangenheit geschah, voll-ständig eliminiert, es sei denn, die elementare Musterfeldbreite deckt die gesamte Seite ab.

[0024] Fig. 2 zeigt ein einfaches Blockdiagramm, das eine Art und Weise, auf die der oben genannte Vorteil erzielt wird, veranschaulicht. Im Einzelnen empfängt der Bildprozessor **12** drei Eingaben, eine Quelle **24**, ein Muster **26** und ein altes Ziel **28**. Die Quelle **24** ist eine Abstraktion, die Grenzpunkte für ein Objekt liefert, das als gedrucktes Bild auf einem Druckmedium erscheinen soll. Die Quelle definiert einen sogenannten Ausfüllbereich, der mit einem Abschnitt eines Überlagerungsmusters ausgefüllt werden soll. Das Muster **26** stellt zumindest einen Abschnitt einer Überlagerungsmuster-Bitabbildung dar, die ein Überlagerungsmuster definiert. Die Überlagerungsmuster-Bitabbildung kann in dem Speicher des Druckers vorliegen und ist in der Lage, zum Erzeugen eines gedruckten gemusterten Bildes gelesen zu werden, was nachfolgend ausführlicher erörtert wird. Das alte Ziel **28** stellt einen sogenannten Streifen dar, der durch den Drucker erzeugt wird und dem Drucker beim Drucken eines gedruckten Bildes behilflich ist. Ein Ziel oder Streifen definiert einen Puffer zum Halten einer Mehrzahl von Bits, die verwendet werden, um einen Laser zu modulieren, um dunkle oder helle Bereiche auf einer gedruckten Seite zu erzeugen, wie Fachleuten einleuchten wird. Der Prozessor **12** bearbeitet die Quelle **24**, das Muster **26** und das alte Ziel **28**, um ein neues Ziel **30** als Ausgabe zu definieren. Das neue Ziel **30** ist ein Ausgabepuffer, der Daten speichert, die durch den Drucker zum Erzeugen eines Abschnitts eines gedruckten Bildes auf dem Druckmedium verwendet werden. Man wird verstehen, dass die oben erwähnten Eingaben lediglich dem Zweck der Veranschaulichung der Erfindung dienen und die Erfindung in keinster Weise einschränken sollen. Wie Fachleute verstehen werden, veranschaulicht das Obige z.B. eine Rasteroperation, die aufgrund der Verwendung der drei Eingaben als ROP3 bekannt ist. Es gibt jedoch andere Rasteroperationen, die andere Anzahlen von Eingaben verwenden. Die Verwendung solcher anderer Rasteroperationen in Verbindung mit der vorliegenden Erfindung ist in der Wesensart und dem Schutzumfang der Erfindung enthalten. Durch Erweiterung dieser monochromen Verfahren kann ein Farbdrucken bewerkstelligt werden.

[0025] Fig. 3 zeigt ein grobes Blockdiagramm, das ein bevorzugtes Verfahren zum Implementieren der vorliegenden Erfindung veranschaulicht. Bei Schritt **32** umfasst das Verfahren ein Definieren zumindest eines Ausfüllbereichs, der mit zumindest einem Abschnitt eines Überlagerungsmusters gedruckt werden soll. Der Ausfüllbereich entspricht der Quelle **24** der Fig. 2, wie sie oben erörtert wurde. Ein Ausfüllbe-

reich kann jegliche Gestalt aufweisen, wenn er als Teil eines gedruckten Bildes auf einem Druckmedium erscheint. Derartige Gestalten können Ovale, Ellipsen, Dreiecke, Quadrate, Rechtecke oder jegliche andere geeignete Gestalt umfassen, die durch einen Satz von Vektoren beschrieben wird, der den Ausfüllbereich zu Zwecken des Druckerbetriebs angemessen definieren kann. Der vorliegende typische Ausfüllbereich wird durch einen Satz eines oder mehrerer Vektoren, die einen eingeschlossenen Bereich bilden, definiert. Komplexe Gestalten mit Innenlöchern, z.B. Donut-Gestalten, werden zuerst zu elementaren geometrischen Gestalten ohne Innenlöcher zerlegt. Man wird jedoch verstehen, dass derartige geschlossene Gestalten die Erfindung in keinster Weise einschränken sollen und dass andere, nicht-geschlossene Gestalten, die andere Quellen definieren, in der Wesensart und dem Schutzumfang der Erfindung enthalten sind. Das Überlagerungsmuster wird durch eine Überlagerungsmuster-Bitabbildung definiert, die in einer Speicherstelle, z.B. dem RAM **20** oder ROM **22**, in dem Drucker **10** gespeichert oder festgehalten ist.

[0026] Als nächstes wird bei Schritt **34** ein Abschnitt der Überlagerungsmuster-Bitabbildung aus dem Speicher gelesen. Vorzugsweise ist der Abschnitt der Überlagerungsmuster-Bitabbildung, der gelesen wird, eine einzelne Zeile aus der Bitabbildung oder zumindest ein Abschnitt einer einzelnen Zeile. Wie nachstehend ausführlicher beschrieben wird, umfasst der Abschnitt einer Zeile, der aus der Überlagerungsmuster-Bitabbildung gelesen wird, zu Effizienzzwecken ein Mehrbitwort, das vorzugsweise ein 32Bit-Wort ist. Bei Schritt **36** sind zumindest ein Grenzpunkt und vorzugsweise zwei Grenzpunkte vorgesehen, die einen Ausfüllbereich-Teilsatz definieren. Ein Ausfüllbereich-Teilsatz dient dazu, einen Abschnitt eines Ausfüllbereichs (der eine horizontale Zeile sein könnte und vorzugsweise ist) auszufüllen, und definiert einen Abschnitt eines gedruckten Bildes, das darin erscheint. Als nächstes wird der Abschnitt der Überlagerungsmuster-Bitabbildung, der aus dem Speicher gelesen wurde, auf den Ausfüllbereich-Teilsatz aufgebracht. Bei Schritt **40** wird der Ausfüllbereich-Teilsatz, der den aufgebrachten Abschnitt der Überlagerungsmuster-Bitabbildung enthält, an einen Ausgabepuffer geschrieben, der oben auch als neuer Zielort **30** bezeichnet wird, der Daten speichert, die zumindest einen Abschnitt des gedruckten Bildes definieren. Schließlich wird bei Schritt **42** der Inhalt des durch Schritt **40** definierten Ausgabepuffers transferiert, um ein Drucken eines gedruckten Bildes zu bewirken. Man wird verstehen, dass der Schritt des Transferierens des Inhalts des Ausgabepuffers umfasst, dass der Ausgabepuffer als Mustereingabe in den Bildprozessor oder ROP, um ein Drucken zu bewirken, verwendet wird. Das Verfahren der Schritte **32** bis **42** wird für andere nachfolgende Zeilen in der Quelle wiederholt, bis der gesamte Quellenbereich gemalt oder mit einem Muster gefüllt ist.

PIXEL-UM-PIXEL-KOPIERMODELL

[0027] Fig. 4 veranschaulicht eine Implementierung des in Fig. 3 gezeigten Verfahrens auf einer Pixel-um-Pixel-Basis. Man wird verstehen, dass Fig. 4 und die folgende Erläuterung des Pixel-um-Pixel-Kopiermodells dargelegt sind, um das Verständnis des nachfolgend erläuterten 32Bit-Wort-Kopiermodells zu fördern.

[0028] Fig. 4 zeigt eine Überlagerungsmuster-Bitabbildung **44**. Die Bitabbildung **44** weist eine Höhe (height) h und eine Breite (width) w auf. Die Bitabbildung **44** definiert ein Überlagerungsmuster, das beim Drucken eines gedruckten Bildes auf ein Druckmedium verwendet wird. Eine Quelle oder ein Objekt **46** ist gezeigt und definiert einen Ausfüllbereich **48**, der auf eine durch die Druckersteuerungssprache angeleitete Weise mit einem Abschnitt des Überlagerungsmusters gedruckt werden soll. Wie oben erörtert wurde, ist die Quelle **46** eine Abstraktion, die Grenzpunkte liefert, die die Anfangs- und Endpunkte zum Erstellen eines Musters definieren. Eine Ausgangszeile **50**, die eine volle Seitenbreite aufweist, wird bereitgestellt und schneidet die Quelle **46** an zwei Punkten **52**, **54**. Die Quelle **46** und die Ausgangszeile **50** definieren ein Paar von Grenzpunkten **52**, **54**, die die Grenzpunkte, zwischen denen ein Abschnitt eines Musters auf einer gedruckten Seite erscheint, angeben oder bereitstellen. Das Segment, in diesem Fall ein Liniensegment, zwischen den Grenzpunkten **52**, **54** definiert einen Ausfüllbereich-Teilsatz **58**, der zumindest einen Abschnitt des Ausfüllbereichs **48** ausfüllt. Mit Informationen der Grenzpunkte **52**, **54** kann eine Musterkomponente erstellt werden, die verwendet wird, um den Teil der Zeile bzw. Linie des replizierten Musters zu erzeugen, den der Bildprozessor benötigt, um die bevorstehende Zeile der Quelle oder des Ausfüllbereichs abzubilden.

[0029] Das Konzept des Anzeigens oder Druckens des selben Musterfeldes über den Bereich einer Quelle beinhaltet das Konzept eines „Ankers". Der Ankerpunkt wird manchmal als Musterursprung bezeichnet, da man ihn sich als einen einzelnen Punkt vorstellen kann, von dem aus die Felder von einem zentralen Punkt aus in allen Richtungen erzeugt werden können. An diesem Punkt erfolgt keine Erläuterung bezüglich der Einzelheiten eines vertikalen Verankerns, da das Problem auf eine einzige horizontale Linie reduziert wurde, die aus dem Musterfeld oder der Überlagerungs-Bitabbildung zu der einzigen Ausgangszeile **50** repliziert werden muss. Das Verfahren des vertikalen Verankerns wurde verwendet, um diese bestimmten Linien auszuwählen.

[0030] Wenn der Prozess des horizontalen Veran-

kerns ausführlicher erörtert wird, wird man erkennen, dass das erfundene Verfahren darauf ausgerichtet ist, den Arbeitsumfang, den der Drucker auf einer bestimmten Zeile bewerkstelligen muss, zu minimieren. Zu Erläuterungszwecken sei angenommen, dass der Feldeinteilungsprozess einer ist, bei dem die Felder direkt vertikal aufeinander gestapelt werden. Es sei weiter angenommen, dass der Ankerpunkt auf einem Musterfeld dessen obere linke Ecke ist. Man kann sich ein Verfahren vorstellen, das den ersten AnkerX-Punkt rechts von dem Quellenschnittpunkt bei **52** fand. Bei Fig. 4 ist dieser Punkt durch den Punkt **53** dargestellt. Dies ist der Punkt, zu dem das Pixel aus der Überlagerungsmuster-Bitabbildung bei Punkt **49** kopiert wird. Pixel, die sich rechts von dem Punkt **49** der Überlagerungsmuster-Bitabbildung werden rechts von **53** auf der Linie **50** kopiert. Die Pixel links von **53** werden von den äußersten rechten Pixeln in der Nähe von **47** kopiert.

[0031] Der soeben beschriebene Prozess zeigt, wie der horizontale AnkerX verwendet werden kann, er muss jedoch zum Zweck einer schnelleren Links-Nach-Rechts-Operation, die bei Punkt **52** auf der Ausgangszeile beginnt und sich so schnell wie möglich nach rechts vorarbeitet, bis das Pixel bei **54** kopiert wird, optimiert werden. Somit besteht die bevorzugte Implementierung darin, unter Verwendung der geeigneten algebraischen Abbildung herauszufinden, welches Pixel **45** zu **52** kopiert werden soll, und ein nach rechts gerichtetes Lesen von Pixeln aus der Überlagerungsmuster-Bitabbildung fortzusetzen und sie zu der Ausgangszeile **50** zu kopieren, bis das Ende an dem Punkt **47** erreicht ist, wo das Lesen eine Schleife zurück zu dem Beginn der Lesezeile bei **49** macht.

[0032] Vorzugsweise wird die Quelle **46** in eine Mehrzahl von Ausfüllbereich-Teilsätzen, z.B. den einen bei **58** gezeigten, der als horizontaler Streifen gezeigt ist, zerlegt. Derartige Teilsätze können so definiert sein, dass sie in der Quelle **46** von oben nach unten verlaufen. Jeder Ausfüllbereich-Teilsatz wird durch einen Satz von Grenzpunkten, Punkten und Längen oder durch andere Ausfüllbereich-Teilsatz-Definitionen definiert, die verwendet werden, um zu bestimmen, wo ein bestimmter Abschnitt eines Musters auf einem Druckmedium erscheinen wird. Jeder Satz von Grenzpunkten, oder eine andere Definition für einen Ausfüllbereich-Teilsatz, wird bestimmt, und anschließend wird ein entsprechender Abschnitt der Überlagerungsmuster-Bitabbildung wie zuvor beschrieben aufgebracht. Dabei werden Bilddaten, die einen Abschnitt des gedruckten Bildes definieren, erzeugt und in einen Ausgabebandpuffer geschrieben, der danach durch den Drucker zum Drucken eines gemusterten Bildes auf ein Druckmedium verwendet wird.

[0033] Das oben beschriebene Verfahren ist vorteilhaft, da es lediglich denjenigen Abschnitt eines Musters liefert, der tatsächlich benötigt wird, und das Erfordernis, die gesamte Musterhöhe über die Breite der Seite vorab zu replizieren, beseitigt. Der Abschnitt des benötigten Musters entspricht dem Ausfüllbereich **48**, da dies der Bereich ist, der einen Abschnitt des Überlagerungsmusters halten soll, wenn ein Bild gedruckt wird. Ferner erfordert dieses Verfahren einen Druckerspeicher für lediglich eine Ausgangszeile **50**, wohingegen bisher ein Speicherpuffer benötigt wurde, der h mal größer ist (wobei h die Höhe der Überlagerungsmuster-Bitabbildung ist), da jede Zeile in der Überlagerungsmuster-Bitabbildung vorab repliziert werden musste, bevor das Papier unter den Laserdruckprozess bewegt werden konnte. Dies stellt eine beträchtliche Ersparnis dar. Für ein Musterfeld, das einen halben Zoll hoch ist, liegen bei 1.200 dpi Speicherersparnisse von 600 zu 1 vor, da 600 Zeilen in der Höhe h vorliegen würden, um ein Muster, das einen halben Zoll hoch ist, auszufüllen.

WORT-KOPIERMODELL

[0034] Das oben erörterte Pixel-Um-Pixel-Kopiermodell veranschaulicht ein Konzept, anhand dessen ein Muster auf ein Druckmedium gedruckt wird. Die folgende Erörterung beschreibt ein bevorzugtes 32Bit-Wort-Kopiermodell, das in mancherlei Hinsicht dem Pixel-Um-Pixel-Kopiermodell ähnelt. Das Verfahren unterscheidet sich in mancherlei Hinsicht, da das Niveau der Programmierungskomplexität aufgrund von Nuancen und speziellen Anforderungen, die damit verbunden sind, 32 Bits gleichzeitig zu kopieren und zu schreiben, steigt, wie Fachleuten einleuchten wird.

[0035] Um Ausfüllbereich-Teilsätze auf effiziente Weise zu verarbeiten und ein gedrucktes, gemustertes Bild auf einem Druckmedium wiederzugeben, muss zu einem bestimmten Zeitpunkt mehr als ein Pixel gelesen werden. Fig. 5 bis Fig. 11 veranschaulichen ein System, bei dem 32Bit-Wörter aus der Überlagerungsmuster-Bitabbildung **44** gelesen werden. Man wird jedoch erkennen, dass auch Wörter gelesen werden können, die andere Längen als 32 Bits aufweisen. In Verbindung mit einem Lesen von Wörtern, die größere Längen aufweisen, z.B. 32Bit-Wörtern, ergeben sich Probleme aufgrund der Tatsache, dass die Breite der Überlagerungsmuster-Bitabbildung **44** eventuell nicht gleichmäßig durch **32** teilbar ist. Ferner können andere Komplikationen vorliegen, wenn die Koordinate in dem ersten Wort (als s-Koordinate bekannt) aus dem ersten Wort, das aus der Überlagerungsmuster-Bitabbildung gelesen werden soll, nicht gleich der Koordinate in dem ersten Wort (als d-Koordinate bekannt) ist, das in den Zielort geschrieben werden soll. Diese Probleme werden in den Fig. 5–Fig. 11 behandelt und werden nachfolgend erläutert.

[0036] Bevor diese Probleme und eine bevorzugte Lösung derartiger Probleme erläutert werden, werden einige grundlegende Definitionen dargelegt. Fig. 6–Fig. 11 zeigen grundlegende Flussdiagramme, die die oben erwähnten Probleme behandeln. Die Flussdiagramme sind in der C-Programmierungssprache beschrieben, welche eine vielfach verwendete und akzeptierte Programmiersprache ist, die Fachleuten bekannt ist.

GRUNDLEGENDE DEFINITIONEN

[0037] Fig. 5 zeigt einige grundlegende Definitionselemente in Form verschiedener Register und Zeiger, die verwendet werden, um einen bevorzugten Aspekt dieser Erfindung zu implementieren. In diesem Abschnitt werden Anführungszeichen verwendet, um die verschiedenen Begriffe, die identifiziert werden, zu benennen. In späteren Abschnitten werden die Anführungszeichen weggelassen. Bei **44** ist eine Überlagerungsmuster-Bitabbildung gezeigt. Die Bitabbildung **44** kann man auch als Eingangsfeld-Bitabbildung oder als Musterspeicherblock bezeichnen. Wie oben beschrieben wurde, definiert die Überlagerungsmuster-Bitabbildung ein Überlagerungsmuster, das beim Drucken eines gedruckten Bildes auf ein Druckmedium verwendet wird. „W_Ursprung" („W_origin") ist ein Zeiger, der auf den Beginn der Ausgangszeile **50** im Anschluss an zwei Wörter eines sogenannten Anlauf-Notizblockspeichers zeigt. W Ursprung und andere Zeiger in der Fig. 5 sind Wortmuster, die auf Byte-Adressen zeigen, die gleichmäßig durch vier teilbar sind. „RStart" zeigt auf den Beginn einer Zeile von der Bitabbildung **44**. Das „R-Zeiger-Register" (oder „R-Zgr-Register", wie es in Fig. 5 erscheint) zeigt auf das nächste Lesewort in der Eingangszeile. „*R" stellt Daten in dem Speicher an der Zeigeradresse dar, die aus der Eingangszeile gelesen werden. Das „W-Zeiger-Register" (oder „W-Zgr-Register", wie es in Fig. 5 erscheint) zeigt auf das nächste Schreibwort in der musterreplizierten Zeile. "*W" stellt Daten in dem Speicher an der Zeigeradresse dar, die in die Ausgangszeile geschrieben werden. Das „REnde-Zeiger-Register" (oder „REnde-Zgr-Register", wie es in Fig. 5 erscheint) zeigt auf das letzte Wort, das aus der Eingangszeile gelesen werden soll. Das „WEnde-Zeiger-Register" (oder „WEnde-Zgr-Register", wie es in Fig. 5 erscheint) zeigt auf das letzte Schreibwort in der musterreplizierten Zeile für einen bestimmten Durchlauf.

[0038] Das „I-Register" ist ein vorübergehendes Register, das die 32Bit-Leseeingabe aus dem Speicher bei „*R" hält. Das „C-Register" ist ein vorübergehendes Register, das einen Übertrag-Leseeingang für einen nachfolgend erörterten Übertragsverarbeitungsfall hält. Das „N-Register" hält Anz_Übertrag_Bits, was die Anzahl von Bits zur Linken des „C-Registers", die übertragen werden, ist.

[0039] Manche zusätzlichen Konventionen für die C-Programmierungssprache, bei denen Variablen zu den nicht mit Zeichen versehenen 32Bit-Ganzzahlen getippt werden, lauten wie folgt. Ein „>>"-Zeichen bedeutet: Verschiebe nach rechts und ziehe Nullen von links. Ein „<<"-Zeichen bedeutet Verschiebe nach links und ziehe Nullen von rechts. „I>>N" bedeutet: Verschiebe das „I-Register" um N Bits nach rechts, wobei N der Inhalt des „N-Registers" ist. Alle Variablen werden als ohne Zeichen „getippt" angesehen, so dass Verschiebungsoperationen immer null Bits von derjenigen Seite, die der Verschiebungsrichtung entgegengesetzt ist, ziehen. In diesem Fall von „I>>N" werden null Bits von links hereingezogen. Desgleichen bedeutet „I<<N": Verschiebe das „I-Register" N Bits nach links, wobei null Bits von rechts hereingezogen werden. Das „&" bedeutet die logische UND-Verknüpfung. Ein „%" ist die Modulo-Operation, die den einfachen Rest nach einer normalen ganzzahligen Teilung ergibt.

[0040] Die Verwendung von Klammern bedeutet, dass die darin enthaltenen Operationen vor jeglicher außerhalb liegenden Operation durchzuführen sind. Wie bei C weisen Multiplikation und Division eine höhere Priorität auf als Addition und Subtraktion. Eine Verschiebung weist eine niedrigere Priorität auf als Addition oder Subtraktion, so dass in den meisten Fällen ein Klammernsatz einen Zählwert, der in einer Verschiebungsberechnung abgelegt ist, umgibt. Beispielsweise bedeutet „I<<(32-N)": Zuerst den Wert in dem N-Register von 32 subtrahieren, dann I um diesen Wert nach links verschieben.

[0041] Wie oben erwähnt wurde, ist „R" der Lese-Zeiger, der auf einen Speicher an einer 32Bit-Speicherstelle zeigt. „R" kann als Speicheradresse betrachtet werden. Der tatsächliche binäre Wert, der an dem Speicher gespeichert ist, auf den durch „R" gezeigt wird, wird durch „*R" dargestellt. Somit ist „R" der Zeiger, und „*R" stellt den Speicherinhalt dar. Desgleichen ist „W" der Schreib-Zeiger, der in dem Speicher auf ein Wort in dem Speicher zeigt, dessen Inhalt „*W" ist, nachdem ein Schreibvorgang abgeschlossen ist. Somit ist „W" der Zeiger, und „*W" stellt den Speicherinhalt dar, der geschrieben wird. Wenn „++" nach dem „*R" oder „*W" erscheint, bedeutet dies: Inkrementiere den jeweiligen Zeiger nach dem Lese- oder der Schreibvorgang um ein Wort. Das Inkrement bezüglich dieser Zeiger bewerkstelligt, dass die Zeiger schnell durch einen Lese- und Schreibraum bewegt werden. Üblicherweise erscheint „*W" oder „*W++" auf der linken Seite der Gleichung und symbolisiert das, was in den Speicher geschrieben wird, und auf der rechten Seite einer Gleichung erscheint „*R" oder „*R++", was angibt, welcher Wert aus dem Speicher gelesen wird. Wenn der logische Operator „ODER" verwendet wird, bedeutet dies eine logische ODER-Bit-Um-Bit-Operation an zwei 32Bit-Registern, von denen eines oder

mehrere vorübergehende Register in der CPU sein kann bzw. können. Berechnungen auf der rechten Seite des Gleichheitszeichens werden zuerst durchgeführt, und dann werden die Ergebnisse in das Register oder die Speicherstellen abgelegt, die auf der linken Seite der Gleichung symbolisiert sind. Somit symbolisiert das Gleichheitszeichen in einer Gleichung den Ablegevorgang. Beispielsweise symbolisiert X = Y + 1 den Vorgang des Hinzuaddierens von Eins zu dem Y-Register (oder Y-Speicherwert) und ein anschließendes Ablegen des Ergebnisses in dem X-Register (oder der Speicherstelle). Wenn kein Zielort benötigt wird, fehlen er und das Gleichheitszeichen. Oft bestimmt der Compiler, ob ein Register oder eine Speicherstelle verwendet wird, um den Wert eines Registers zu speichern. Bei C ist es möglich, anzufordern, dass der Compiler Register für einige spezifizierte Variablen verwendet. Die kluge Auswahl dessen, welche Variablen (oder wann diese Variablen) als Register implementiert werden sollten, kann manchmal durch Optimierung von Compilern und manchmal durch den Programmierer gesteuert werden. Nachdem die Definitionselemente dargelegt wurden, folgt nun eine ausführlichere Erläuterung des 32Bit-Wort-Kopiermodells unter Bezugnahme auf Fig. 6 – Fig. 11.

[0042] Fig. 6 ist ein Flussdiagramm, das auf globale Weise das 32Bit-Wort-Kopiermodell darlegt, das eine Zeile eines replizierten Musters erstellt. Verschiedene der Kästchen in Fig. 6 zweigen zum Zweck einer ausführlicheren Erläuterung der durch dieselben durchgeführten Operation zu anderen Figuren (Fig. 7–Fig. 11) ab. Beispielsweise ist Kästchen **100** ausführlicher in Fig. 7 erklärt, die eine Variabler-Satz-Prozedur darlegt, die eine Anzahl von nützlichen Variablen für ein Computerprogramm definiert, das die Erfindung implementiert; Kästchen **114** ist ausführlicher in Fig. 8 erläutert, die die Situation angeht, bei der das erste Bit in dem ersten Lesewort bzw. den ersten Leseworten zu dem ersten Schreibbit (d) in dem Zielort passt oder auch nicht passt; Kästchen **118** wird ausführlicher in Fig. 9 erläutert, die den Fall darlegt, bei dem eine Feldbreite ein Vielfaches von **32** ist, was bedeutet, dass sich weder das Ausmaß des Übertrags noch das Ausmaß der Verschiebung am Ende jeglichen Lesedurchlaufs über das Eingangsfeld verändern kann; das Kästchen **128** wird ausführlicher in Fig. 10 erklärt, die den Fall darlegt, bei dem kein Erfordernis besteht, etwaige Bits zwischen Wortlesungen der Feldbreite zu übertragen; und Kästchen **126** ist ausführlicher in Fig. 11 erklärt, die den umgekehrten Fall darlegt, bei dem Bits auf jeden Wortlesevorgang hin übertragen werden. In dem Übertrag-Fall kann sich die Anzahl von Übertragsbits akkumulieren, um die Anzahl, die in dem 32Bit-Übertragsregister gehalten werden kann, zu überschreiten oder gleich derselben zu sein, wie in Fig. 11 gezeigt ist.

[0043] In der nachfolgenden Erläuterung wird der Leser auf Fig. 6 verwiesen, und während Fig. 6 durchlaufen wird, wird jedes der oben erwähnten Kästchen, die Gegenstand einer ausführlicheren Erläuterung sind, erkundet, während man beim Durchlaufen von Fig. 6 auf dieselben trifft.

EINRICHTUNGSVARIABLEN

[0044] Der erste Schritt in Fig. 6 ist bei Kästchen **100** gezeigt und umfasst einen Variableneinrichtungsschritt, bei dem ausgewählte Variablen definiert werden. Diese Variablen werden ausführlicher in Fig. 7 dargelegt, die ein Flussdiagramm ist, das eine Variableneinrichtungsprozedur darlegt. Im Einzelnen definiert Kästchen **200** unter Bezugnahme auf Fig. 7 eine Anzahl von Variablen. Wenn die Routineeingaben, abschneiden_x1 und rechts_x2 die inklusive Vorrichtungsraumpixel-Koordinaten sind, bei denen die Quelle die benötigte bevorstehende Ziellinie schneidet, und wenn auch Rstart, der Startwortzeiger, gegeben ist, berechnet Kästchen **200** manche nützlichen Wort-orientierten Anfangs/Endlese- und -schreibkoordinaten und -zeiger. Genauer gesagt werden abschneiden_x1_Wort, rechts_x2_Wort und die Zeiger REndeMinus, W und WEnde initialisiert. Bei Kästchen **210** ist AnkerX die Koordinate in einem Zielkoordinatenraum, wo eine spezifizierte Position in dem Musterfeld festgelegt ist. Diese Position ist oft die obere linke Ecke des Feldes, sie kann jedoch ein beliebiger Punkt auf dem Feld sein. Hier wird die obere linke Ecke als Punkt auf dem Feld verwendet, der verankert ist. Unter Bezugnahme auf Kästchen **210**–**218** wird ein Beispiel dargelegt, das eine Pixelkoordinate von Null für den äußersten linken druckbaren Punkt auf der Seite verwendet. Der spezifizierte AnkerX für das Muster und abschneiden_x1 weisen verschiedene Rückspringfälle auf, die dadurch vereinfacht werden, dass man die relativen Koordinaten jedes Falles in dem Kästchen **210** betrachtet. In beiden Fällen berechnen die Kästchen **212** und **214** die relative Anfangsposition für ein Eingangsfeldzeilenlesen, mod_x. Anschließend berechnet Kästchen **216** die relativen Bitkoordinaten in dem ersten 32Bit-Wort zum Lesen und Schreiben, die s bzw. d sind. Ein s-Wert von null bedeutet, dass das höchstwertige linksgerichtete Bit in dem Lesewort verwendet werden wird. Ein Bit_Delta wird bei Kästchen **218** berechnet, das die relative Verschiebung in Bits zwischen dem Lese- und dem Schreibwort ist. Der Lesezeiger R und VerbleibendeBits werden ebenfalls berechnet, die der anfängliche Lesezeiger und die Anzahl von herabhängenden Bits am Ende der Lesefeldzeile sind. Wenn die Feldbreite z.B. 33 ist, so sind VerbleibendeBits 1, da 1 Bit von dem Ende des ersten Wortes herabhängt.

[0045] Unter erneuter Bezugnahme auf Fig. 6 wird bei Abschluss des Variableneinrichtungsschritts bei Kästchen **100** bei Kästchen **110** ein Test durchge-

führt, um zu bestimmen, ob sowohl Bit_Delta als auch VerbleibendeBits null sind. Wenn beide null sind, kann bei Kästchen **112** eine einfache Kopie auf beliebig viele Arten und Weisen erstellt werden, einschließlich einer Verwendung von standardmäßigen Kopiebibliotheken. Dies ist der einfachste Fall, der eine Schleife eines Lesens und Schreibens von 32Bit-Wörtern, bis das äußerste rechte Wort des Feldes kopiert ist, und eines Wiederholens des Vorganges, bis eine ausreichende Ausgabe erzeugt wird, ist. Eine Implementierung des Kästchens **112** könnte das in Fig. 10 gezeigte sein, indem das Kästchen **520** durch eine Schleife zurück zu dem obigen Kästchen **500** ersetzt wird. Eine ausreichende Ausgabe wird durch einen Test erzeugt, der durchgeführt wird, um zu bestimmen, ob das W über den WEnde-Zeiger hinaus fortgeschritten ist. Wenn die Antwort an dem Kästchen **110** dagegen „nein" lautet, schreitet das Verfahren zu Kästchen **114** fort, was als die s-d-Fälle bezeichnet wird.

S-D-FÄLLE

[0046] Die s-d-Fälle des Kästchens **114** werden in Fig. 8 ausführlicher dargelegt. Die in den Kästchen **300** bis **320** dargelegten Berechnungen vereinfachen die Ausgangsbedingungen zum Verarbeiten auf einer Zeile, so dass die Anfangsbedingungen leichter zu verstehen sind. Falls der relative Beginn, das Bit s in dem ersten Lesewort zu lesen, zu der ersten d-Schreibbitzahl bei Kästchen **310** passt, wird das Problem bei Kästchen **312** vereinfacht, indem s und d auf null gesetzt werden und keine weitere Verarbeitung notwendig ist als die, die angegebenen Variablen ordnungsgemäß mit Initialen zu versehen. Wenn bei Kästchen **310** jedoch bestimmt wird, dass s größer als d ist, so wird bei Kästchen **314** s durch den Wert von d angepasst, d wird angepasst, und abschneiden_x1 wird durch abschneiden_x1 = (abschneiden_x1_Wort<<5) so angepasst, dass es einheitlich ist. Das Übertragsregister wird auf C = (*R++)<<s initialisiert. Dies bedeutet, dass das Speicherwort, auf das durch R gezeigt wird (z.B. *R) gelesen wird und dann um den s-Wert nach links verschoben wird und dass das Ergebnis in das C-Register platziert wird. Das ++ gibt an, dass der R-Zeiger nachinkrementiert wird. Ferner wird die Anzahl von Übertragsbits auf 32-s eingestellt, da dies die Anzahl von Bits in dem linken Teil des C-Registers ist, die für diesen Fall übertragen werden. Die ungenutzten rechten Bits in dem C-Register werden in einem Nullzustand gehalten.

[0047] Wenn s bei Kästchen **310** geringer ist als d, passt Kästchen **316d** um den Wert s an, setzt s gleich null und berechnet den Wert abschneiden_x1 neu, wie gezeigt ist. Anschließend werden die folgenden Schritte durchgeführt: Lade I von dem R-Zeiger, inkrementiere R mit I=*R++; als Nächstes schreib mit *W++ = (I>>d), was in die Adresse schreibt, auf die durch W gezeigt wird, mit dem Wert des I-Registers, nachdem eine Verschiebung um den Wert d nach rechts erfolgte. Außerdem initialisiere C mit C = I <<(32–d), was I ist, das um den Wert (32-d) nach links verschoben ist. Schließlich stelle N auf den Wert von d ein. Bei Kästchen **318** wird eine Prüfung dahingehend durchgeführt, ob der R-Zeiger über den REndeMinus-Wert hinausgeht. Falls dies der Fall ist, wird eine spezielle Anpassung eines ähnlich gelagerten Falls, bei Kästchen **320** gezeigt, durchgeführt. Falls dies nicht der Fall ist, sind die s-d-Fälle abgeschlossen.

[0048] Unter erneuter Bezugnahme auf Fig. 6 wird nach Kästchen **114** bei Kästchen **116** eine Prüfung dahingehend durchgeführt, ob VerbleibendeBits gleich null ist. Falls die Antwort „ja" ist, liegt bei Kästchen **118** ein Gleichmäßige-Feldbreite-Übertrag-Fall vor. Der vollständige Gleichmäßige-Breite-Übertrag-Fall des Kästchens **118** ist in Fig. 9 ausführlicher dargelegt.

GLEICHMÄSSIGE-FELDBREITE-ÜBERTRAG-FALL

[0049] Ein Gleichmäßige-Breite-Übertrag-Fall bedeutet, dass die Feldbreite in Pixel ein einfaches ganzzahliges Vielfaches von 32 ist. Beispiele möglicher Feldbreiten sind Breiten von 32, 64, 96 ... Pixeln. Eine derartige Feldbreite zu haben bedeutet, dass sich am Ende jeglichen Lesedurchlaufs über das Eingangsfeld weder der Umfang des Übertrags noch das Ausmaß der Verschiebung verändern kann. Diese können sich nicht verändern, da keine herabhängenden Bits vorliegen, um sie zu ändern. Somit bleibt der Inhalt des Übertragsregisters C während der Schleifenbildung invariant, ebenso wie die Anzahl von Übertragsbits N. Im Vergleich zu dem nachfolgend erörterten Übertragsfall machen ein konstantes C und N diesen Fall relativ einfach.

[0050] Bei Kästchen **400** wird REndeMinus auf einen Punkt inkrementiert, der gerade über REnde, die Lesung des letzten Wortes, hinausgeht. Dies ist eine Vereinfachung, die die Sequenz des gleichmäßigen Lesestopps nutzt. Da an dem Ende des Worts keine Bits herabhängen, ist keine Vorwarnung bezüglich der Lesefeldschleife nötig. Nachdem REndeMinus angepasst wurde, sind alle Lesevorgänge für die Bedingung, bei der R geringer ist als REndeMinus oder (R<REndeMinus). Bei Kästchen **410** wird eine Prüfung dahingehend durchgeführt, ob W geringer als oder gleich WEnde ist und ob der Lesezeiger R weit genug links ist, so dass vier Wörter gelesen werden können, ohne über das Ende des Feldes hinaus zu lesen. Wenn zumindest vier Lesevorgänge vorliegen, die für diesen Durchlauf über die Feldbreite hinaus verbleiben, werden bei Kästchen **412** vier sehr schnelle Lese/Schreibkombinationen durchgeführt. Das Kästchen **412** ist als Viererschritt-Schnellüber-

tragskopie bekannt. Man sollte verstehen, dass an dem Prozessor und Compiler, der verwendet wird, eine empirische oder theoretische Analyse durchgeführt werden kann, um zu bestimmen, ob eine Durchführung von Viererschritten, z.B. bei Kästchen **412**, die Verarbeitung beschleunigt oder verlangsamt. Wenn der Viererschritt-Prozess das CPU-System verlangsamt, wird der Viererschrittprozess nicht verwendet, und es erfolgt ein Vorgehen in einzelnen Schritten, wie bei Kästchen **416**. Dieselbe Analyse gilt auch für andere, spätere Fälle, bei denen in den Flussdiagrammen Viererschritte gezeigt sind. Bei manchen Prozessoren mit einem Anweisungs-Cache-Speicher ist es aus der Theorie möglich, auszuwählen, welche Viererschritte die Verarbeitung beschleunigen. Wenn Viererschritte schneller sind, liegt es üblicherweise zum Teil daran, dass die Kosten des Schleifendurchlauf-Zusatzaufwands für die Schleifenbeendigung verringert werden, indem eine Verteilung über mehrere Operationen hinweg erfolgt. Wenn eine Viererschrittverarbeitung durchgeführt wird, wie bei Kästchen **412**, kann die Ausgabe um bis zu drei ganze Wörter über WEnde hinausrutschen bzw. -gleiten. Wenn also das Viererschrittverfahren oder eine beliebige Erweiterung des Viererschrittverfahrens verwendet wird, muss darauf geachtet werden, einen zusätzlichen „Rutsch"-Schreibpufferraum auf dem rechten Teil des Puffers bereitzustellen, so dass das Rutschen niemals auf Puffer treten kann, die zu anderen logischen Funktionen gehören.

[0051] Unter genauerer Bezugnahme auf das Kästchen **412** ist jeder der Schritte genau derselbe wie ein einzelner Schritt bei Kästchen **416**, wenn wir uns innerhalb von vier Wörtern der rechten Seite des Feldes befinden. Jeder Schritt lautet wie folgt:

- I = *R++; lade I-Register von dem Speicher, auf dem durch das R-Zeiger-Register gezeigt wird, anschließend inkrementiere den R-Zeiger.
- *W++ = C ODER (I>>N); schreibe dorthin, wo das Schreibregister W in dem Zielortmusterraum die logische ODER-Kombination des C-Registers und die Rechtsverschiebung des I-Registers um N Bits zeigt, wobei N die Anzahl von Bits ist, die in dem C- bzw. Übertragsregister verwendet werden. Die ODER-Funktion ist hier eine Möglichkeit des Kombinierens der oberen N Bits von C mit (32-N) Bits des I-Registers. Es gibt auch andere Möglichkeiten, dies zu bewerkstelligen, z.B. als einfache, nicht mit Vorzeichen versehene 32Bit-Ganzzahl ADD von C und (I>>N). Eine ADD würde funktionieren, da die rechts gerichteten (32-N) Bits des C-Registers null sind und die links gerichteten N Bits von (I>>N) null sind, wenn I ein nicht mit Vorzeichen versehenes Register ist. Somit ist die ODER-Funktion lediglich ein beispielhaftes Verfahren zum Bewerkstelligen der Kombination von C und (I>>N).
- C = I<<(32-N); der neue Wert des C-Registers ist das um den Wert (32-N) verschobene I-Register. An diesem Punkt kann N niemals größer als 31 sein.

[0052] Wenn die Antwort bei Kästchen **410** „nein" lautet, so werden an den Kästchen **414** und **416** jegliche verbleibenden Wörter der Leseoperation aus diesem Durchlauf auf dem Feld fertiggestellt. Bei Kästchen **418** wird das R-Zeiger-Register neu geladen, um auf die Anfangsposition auf der linken Seite der Eingangsfeldzeile bei RStart zu zeigen. Bei Kästchen **420** wird eine Prüfung dahingehend durchgeführt, ob ein ausreichendes Schreiben bewerkstelligt wurde. Falls dies nicht der Fall ist, wird an den Kästchen **410** und **412** eine Schleife zurück zu dem nächsten Satz von Viererschritten gemacht, und/oder es werden einzelne Schritte an den Kästchen **414** und **416** durchgeführt. Wenn bei Kästchen **420** W>WEnde, wird bei Kästchen **422** ein Wert zurückgegeben, der die Wortzeigeradresse ist, die durch den Bildprozessor verwendet werden wird, um ein Lesen des Musters an der Schnittpunktkoordinate mit der Quelle zu beginnen. Diese Adresse lautet (W_origin + abschneiden_x1_Wort).

[0053] Unter erneuter Bezugnahme auf Fig. 6 muss bei Kästchen **116**, wenn VerbleibendeBits nicht null ist, die Verarbeitung der Schleife **120** bis **130** erfolgen, was oft die Übertragsverarbeitung des Kästchens **126** beinhaltet. Für manche Durchläufe können die Bedingungen einen in Kästchen **128** gezeigten Kein-Übertrag-Fall ergeben. Eine Prüfung, ob eine Übertragsverarbeitung nötig ist, erfolgt bei Kästchen **124** einfach dadurch, dass der Wert in dem N-Register geprüft wird. Falls N null ist, gibt es keine Übertrag-genutzten Bits in dem C-Register, und es kann der Fall der schnelleren Kein-Übertrag-Verarbeitung verwendet werden. Der Fall der Kein-Übertrag-Verarbeitung des Kästchens **128** ist in Fig. 10 ausführlicher dargelegt.

KEIN-ÜBERTRAG-VERARBEITUNG

[0054] Eine Kein-Übertrag-Verarbeitung ist einfacher als der Übertragsfall, da niemals das Erfordernis vorliegt, etwaige Bits zwischen Lesevorgängen der Feldbreite zu übertragen, während der R-Zeiger auf volle 32Bit-Wörter einer gültigen Feldeingabe zeigt. Die Organisation von Viererschritten und einzelnen Schritten (Kästchen **500** bis **514**) ist dieselbe wie für den Fall eines vollständigen Gleichmäßige-Breite-Übertrags, und hier gelten die selben Vorsichtsmaßnahmen bezüglich eines Bestimmens der relativen Vorteile des Integrierens der Viererschritt-Optimierungen, die zuvor erörtert wurden. Man beachte, dass bei Kästchen **514** die Eingabe aus dem R-Zeiger gelesen wird (wobei der Wert *R ist) und die Ausgangsspeicherstelle, auf die durch W gezeigt wird, geschrieben wird (durch *W symbolisiert). Anschließend werden sowohl der W- als auch der R-Zeiger inkrementiert, wie durch das ++Postfix angegeben ist.

Bei Kästchen **516** wird eine Prüfung dahingehend durchgeführt, ob dieses jeweilige Band über die Feldbreite mit Ausnahme des Teils der herabhängenden VerbleibendenBits abgeschlossen wurde. Durch Vergleichen des W-Zeigers mit dem WEnde-Zeiger bei Kästchen **516** wird eine Prüfung dahingehend durchgeführt, ob die Replikation abgeschlossen ist. Falls sie abgeschlossen ist, wird die ganze Teilroutine verlassen, wie zuvor, die eine unmittelbare Rückkehr oder Verzweigung zum Kästchen **122** der Fig. 6 verwenden würde. Falls die Routine noch nicht abgeschlossen ist, wird bei Kästchen **520** die Übertragsverarbeitung wieder aufgenommen, indem das C-Register mit den VerbleibendenBits, die von dem letzten Wort des Feldes herabhängen, geladen wird. Ferner wird N gleich dem VerbleibendeBits-Wert gesetzt.

[0055] Unter erneuter Bezugnahme auf Fig. 6 muss, falls die Antwort bei Kästchen **124** „ja" lautet, bei Kästchen **126** eine Übertragsverarbeitung durchgeführt werden. Der Durchlauf der Übertragsverarbeitung wird in Fig. 11 ausführlicher dargelegt.

ÜBERTRAGSVERARBEITUNG

[0056] Die Übertragsverarbeitung ist der komplizierteste Fall, da sich die Anzahl von Übertragsbits ansammeln kann, um die Anzahl, die in dem 32Bit-Übertragsregister gehalten werden kann, zu überschreiten oder gleich derselben zu sein.

[0057] Kästchen **600** bis **614** sind dieselben, wie sie oben für den Fall des vollständigen Gleichmäßige-Breite-Übertrags beschrieben wurden (Kästchen **410** bis **416**). Der Leser sei auf die obige Erörterung in Bezug auf eine derartige Operation verwiesen. Bei Kästchen **616** wird eine Prüfung dahingehend durchgeführt, ob das Schreiben des Ausgangs abgeschlossen ist. Falls die Antwort bei Kästchen **616** "ja" lautet, kehre unmittelbar zum Kästchen **122** in Fig. 6 zurück. Falls nicht, werden bei Kästchen **618** neue herabhängende Bits zu dem Übertragsregister hinzugefügt. Bei Kästchen **618** wird das nächste C-Register aus den alten C-Registerbits (von denen lediglich N der äußersten Linken bedeutsam sind), die ODER-verknüpft sind mit dem Wert (*R>>N) im Speicher an der Position, auf die durch den R-Zeiger gezeigt wird, errechnet. Dann wird ein Wert Überschiebung (OverShift) zur späteren Verwendung dem Wert 32-N zugewiesen. Das OverShift ist die Anzahl ungenutzter Bits, die das Verarbeiten bei Kästchen **618** beginnen. Das neue N wird gleich dem alten Wert + VerbleibendeBits gesetzt, da VerbleibendeBits die zusätzliche Anzahl von Bits ist, die an dem nächsten Durchlauf mitgenommen entlang werden müssen, wobei sie die herabhängende Menge sind, die von dem Ende des Feldes gelesen wird, um den aktuellen Durchlauf abzuschließen. Bei Kästchen **620** gibt es zwei mögliche Fälle. Entweder ist N über 31 Bits hinaus übergelaufen oder nicht. Wenn das neue N 32 Bits beträgt, ist das Übertragsregister voll, und es kann in einen Speicher geschrieben und unmittelbar gelöscht werden. Wenn N 32 überschreitet, muss das Übertragsregister ebenfalls in den Speicher geschrieben werden, und bei Kästchen **628** wird ein neues Übertragsregister erneut berechnet. Zusammenfassend gesagt wird bei Kästchen **620** eine Prüfung dahingehend durchgeführt, ob C bei dem Kästchen **622** geschrieben wird. Falls C bei dem Kästchen **622** geschrieben wird, wird N als Teil der selben Operation des Kästchens **622** mit N = N – 32 entsprechend angepasst. Wenn N bei Kästchen **620** geringer ist als 32, dann fährt die Verarbeitung fort, ohne das Übertragsregister zu schreiben, da keine Überlaufbedingung vorliegt, und das Übertragsregister muss bei dem nächsten Bandlesevorgang über das Eingangsfeld in den nächsten Übertragsverarbeitungsfall übertragen werden. Bei Kästchen **624** wird eine Prüfung von N durchgeführt. Falls N null ist, wird C bei Kästchen **626** gleich null gesetzt, da bei dem nächsten Durchlauf kein Übertrag vorliegt. Andernfalls werden die Überlaufbits bei Kästchen **628** durch C = *R<<OverShift gelesen, was die Überlaufbits in den linken Teil des C-Registers platziert und ein Austreten zu dem Kästchen **130** in Fig. 6 ermöglicht. Bei Kästchen **130** wird in dem linken Teil des Lesefeldes bei R = RStart für den nächsten Durchlauf, der das Feld bei Kästchen **120** zurückliest, ein Start durchgeführt.

[0058] Schließlich gibt es eine Erweiterung des gezeigten Verfahrens, die die Bildverarbeitungsarbeit für manche Fälle verringert, in denen die Breite der Quelle ausreichend groß ist (z.B. über eine klug gewählte Schwelle) und der verwendete ROP den Effekt aufweist, dass die Mustereingabe einfach direkt in das Ausgabevideoband kopiert wird. In diesem Fall ist es möglich, die zuvor erläuterte Operation zu verändern, um direkt in den Zielvideopuffer zu schreiben, wodurch der Bildverarbeitungsschritt eliminiert wird. Der Algorithmus müsste angepasst werden, so dass das Rutschen während des Anhaltens in der richtigen Koordinate eingeschränkt wäre. Ferner müssten sowohl die Anfangs- als auch die Endverarbeitung geändert werden, um bei den richtigen Pixelkoordinaten zu beginnen und zu enden.

KOSTENMODELL

[0059] Die oben beschriebene Erfindung ist aufgrund der durch dieselben bewirkten inhärenten Druckerspeichereinsparungen nützlich. Dies ist darauf zurückzuführen, dass im Gegensatz zu bekannten Systemen ein Muster nicht über eine gesamte Seitenbreite vorab repliziert werden muss. Deshalb wird wertvoller Speicher eingespart. Man wird verstehen, dass bestimmte digitale Drucker, z.B. Laserdrucker, ein Druckmedium wie z.B. Papier bei einer konstanten Geschwindigkeit durch den Druckvorgang bewegen. Diese Geschwindigkeit wird von Anfang bis Ende beibehalten. Somit wurde es in der Vergangen-

heit zweckmäßig, ein Muster über eine gesamte Seitenbreite vorab zu replizieren, um zu gewährleisten, dass der Drucker, und insbesondere der Laser, genügend Arbeit hatte. Dadurch war man sicher, dass dem Drucker nicht die Arbeit ausgehen würde und er dadurch nicht einen sogenannten Punt erzeugen würde, den man sich als Arbeitsausfall vorstellen kann. Folglich wurde es zur üblichen Praxis, die Druckerarbeit sich ansammeln zu lassen, bevor man das Papier auf seine Druckreise schickte. Nachdem sich genügend Arbeit angesammelt hatte, wurde das Papier auf seine Druckreise geschickt.

[0060] Bei der vorliegenden Erfindung erfolgt eine Musterverarbeitung in Echtzeit, während der Laser einer Wettlaufsituation unterzogen („Racing") wird. Ein Racing des Lasers erfordert, eine Bestimmung darüber durchzuführen, wie der beste Kompromiss zwischen Druckerspeicher- und Echtzeitverarbeitungsanforderungen zu erzielen ist.

[0061] Ein Nachteil des Durchführens einer Musterreplikation, während man den Laser einer Wettlaufsituation unterzieht, besteht darin, dass dieser Vorgang Bildprozessorzeit kostet. Wenn die inkrementalen Kosten zu hoch sind, könnten sie bewirken, dass der Bildprozessor den Wettlauf mit der Ausgabevideoaufgabe verliert, was zu einem sogenannten Drucküberlauf oder Punt führt. Derartige Fehldrucke können vermieden werden, indem die Geschwindigkeit der Musterverarbeitung optimiert wird und indem jegliche Videobandpuffer, die einen Drucküberlauf verursachen würden, vorrasterisiert werden. Es folgt nun eine Erläuterung dessen, wie auf präzise Weise bestimmt werden kann, welche Videostreifen vorrasterisiert werden müssen, wodurch die Möglichkeit von Punts eliminiert wird, ohne Speicher zu verschwenden, indem zu viele Videobänder vorrasterisiert werden.

[0062] Üblicherweise stellt man sich den Echtzeit-Druckvorgang bei einem Laserdrucker als Wettlauf zwischen zwei Aufgaben vor. Bei einem ordnungsgemäß funktionierenden Drucker liegt niemals ein Drucküberlauf vor, weil die Bildprozessoraufgabe es gerade schafft, jeden Wettlauf mit der DMA-Ausgangsaufgabe zu gewinnen. Es ist nicht wünschenswert, Drucküberläufe zu vermeiden, indem man jedes Videoband unilateral vorrasterisiert, da dies (auch bei Komprimierung) zu viel wertvollen Druckerspeicher für Video-DMA-Puffer in Anspruch nimmt. Es wurde ein Verfahren zum Ermöglichen einer Minimierung der Anzahl von vorrasterisierten Videopuffern entwickelt, das in der U.S.-Patentschrift Nr. 5,129,049 an Cuzzo u. a., deren Offenbarung durch Bezugnahme ausdrücklich in das vorliegende Dokument aufgenommen ist, offenbart wurde. Dies wurde in der U.S.-Patentschrift Nr. 5,479,587 an Campbell u. a. in Bezug auf Komprimierungs- und empirische Bildprozessorkostenmessungen erweitert.

[0063] Das nachfolgend erläuterte bevorzugte Modell zielt darauf ab, ein Mittel bereitzustellen, um zu gewährleisten, dass die minimale Anzahl von Video-DMA-Bandpuffern vorrasterisiert wird, so dass die Bildprozessoraufgabe gerade jeden Wettlauf mit dem Video-DMA gewinnen kann, ohne übermäßig viel Druckerspeicher zu verwenden.

[0064] Als nächstes folgt das bevorzugte Kostenmodell für eine Musterverarbeitung gemäß der vorliegenden Erfindung. Es liefert ein präzises und unkompliziertes Modell, das entworfen ist, um die beste Balance zwischen Speichernutzung und der Vermeidung von Punts zu erzielen:

Kosten = Zusatzkosten + (ObjektBreite· BreiteKosten) + (Replikationen * ReplikationsKosten)

[0065] Das obige Modell betrachtet drei Parameter: Zusatzkosten, Breitekosten und Replikationskosten. Das Modell bewertet auf einer Zeile-Um-Zeile-Basis (bzw. Ausfüllbereich-Teilsatz-Um-Ausfüllbereich-Teilsatz-Basis) die Echtzeitverarbeitungsanforderungen zum Zweck des Identifizierens, welche Zeilen, Streifen oder Ausfüllbereich-Teilsätze eine Vorrasterisierung benötigen, um einen Drucküberlauf oder einen sogenannten Punt zu vermeiden. Die Zusatzkosten sind ein Parameter, der die mit dem Vorgang verbundenen Zusatzzeitanforderungen berücksichtigt. Der Parameter der Breite-Kosten (das Produkt der ObjektBreite und der BreiteKosten) berücksichtigt, wie viel Zeit damit verbunden ist, ein Objekt, das eine bestimmte Breite aufweist, zu verarbeiten. Der Parameter der ReplikationsKosten (das Produkt aus Replikationen und ReplikationsKosten) berücksichtigt die Zeit, die für Schleifenbildungsentscheidungen aufgewendet wird. Die drei Parameter werden durch empirische Messungen bei Wort-Fall-Szenarios bestimmt, wobei die zuvor beschriebenen schnelleren Pfade vermieden werden. Es könnten Erweiterungen erfolgen, um die Verbesserungen bezüglich der schnellen Pfade zu berücksichtigen, dies ist jedoch allgemein unnötig, es sei denn als Mittel zum Vermeiden einer Speicher-Aus-Bedingung in dem Drucker. Gemäß dem obigen Modell können die entstandenen Kosten verwendet werden, um zu bestimmen, welche Streifen vorrasterisiert werden müssen, um einen Drucküberlauf zu vermeiden. Dies kann zweckmäßigerweise durch einen Vergleich der entstandenen Kosten mit einer bekannten Schwelle bestimmt werden.

[0066] Zusätzlich zu einer Bestimmung, welche Streifen eine Vorabrasterisierung benötigen, kann das Modell verwendet werden, um einen Kompromiss zwischen einer Vorrasterisierung und einer Vorab-Feldeinteilung bereitzustellen. Beispielsweise kann der maximale Quellenbereich oder Auffüllbereich, auf den ein Muster aufgebracht werden soll, aufgezeichnet werden, während eine bestimmte Seite aufgebaut wird. Die maximale Breite ist die Breite,

die zur Vorab-Feldeinteilung verwendet werden würde. In vielen Fällen wäre diese Breite wesentlich geringer als die physikalische Seitenbreite. Somit wären die Speicheranforderungen sogar in dem Fall, in dem eine Vorab-Feldeinteilung verwendet würde, bei äußerst komplexen Seiten manchmal beträchtlich geringer.

[0067] Somit unterstützt das obige Kostenmodell die vorliegende Erfindung darin, den Laser während der Vorbereitung eines Musters auf ein Drucken auf einem Druckmedium einer Wettlaufsituation zu unterziehen oder vor der Laseroperation zu bleiben.

VEKTORMUSTER

[0068] Die vorliegende Erläuterung erforderte einen Bitabbildungsmuster-Speicherblock **44**. In vielen Fällen kann die rechteckige Bitabbildung durch einen Satz von Vektorausfüllbefehlen beschrieben werden. Wenn der Satz von Vektorausfüllbefehlen einfach genug ist, ist es möglich, die Vorbereitung des Musterspeicherblocks zu überspringen und die schreibmusterreplizierte Zeile direkt zu bilden, indem dieselben Typen von Beginnen-Überspringen-Wiederholen-Schritten, die zuvor gelehrt wurden, erweitert werden. Ein derartiges Vektorausfüllmuster ist in den verschiedenen Mitteln zum Anwenden dieser Erfindung enthalten.

[0069] Vorschriftsgemäß wurde die Erfindung in einer Sprache beschrieben, die mehr oder weniger spezifisch für strukturelle und methodische Merkmale ist. Man sollte jedoch verstehen, dass die Erfindung nicht auf die gezeigten und beschriebenen spezifischen Merkmale beschränkt ist, da die hierin offenbarten Mittel bevorzugte Formen zum Umsetzen der Erfindung umfassen. Somit wird die Erfindung in jeglicher ihrer Formen oder Modifizierungen in dem ordnungsgemäßen Schutzumfang der beigefügten Patentansprüche beansprucht, die gemäß der Äquivalenten-Doktrin entsprechend interpretiert werden.

Patentansprüche

1. Ein Verfahren zum Betreiben eines digitalen Druckers (**10**) zum Erzeugen eines gedruckten Bildes auf einem Druckmedium; wobei der Drucker zumindest einen Speicher (**20**) zum Halten einer Überlagerungsmuster-Bitabbildung (**44**), die ein Überlagerungsmuster definiert, von dem zumindest Abschnitte beim Drucken des gedruckten Bildes auf das Druckmedium verwendet werden, und einen Bildprozessor (**12**) zum Verarbeiten von Bilddaten, die zum Erzeugen des gedruckten Bildes verwendet werden, umfasst, wobei das Verfahren folgende Schritte umfasst:
a. Definieren (**32**) zumindest eines Ausfüllbereichs (**48**), der mit zumindest einem Abschnitt des Überlagerungsmusters bedruckt werden soll;
b. Lesen (**34**) zumindest eines Abschnitts der Überlagerungsmuster-Bitabbildung (**44**) aus dem Speicher, wobei der Abschnitt eine Mehrzahl von Bits umfasst;
c. Bereitstellen (**36**) zumindest eines Grenzpunktes (**52**, **54**), der einen Ausfüllbereich-Teilsatz (**58**) teilweise definiert; wobei der Ausfüllbereich-Teilsatz (**58**) dazu dient, zumindest einen Abschnitt des Ausfüllbereichs (**48**) zu füllen und ein in demselben erscheinendes gedrucktes Bild zu definieren, wobei das Lesen der Überlagerungsmuster-Bitabbildung (**44**) ein Verwenden des zumindest einen den Ausfüllbereich-Teilsatz definierenden Grenzpunktes (**52**, **54**), um ein Startbit zu bestimmen, das aus der Überlagerungsmuster-Bitabbildung (**44**) gelesen werden soll und das zu einem ersten Schreibbit in dem Ausfüllbereich-Teilsatz (**58**) gehört, umfasst;
d. Beginnen mit dem Startbit und dem ersten Schreibbit, Kopieren des zumindest einen Abschnitts der Überlagerungsmuster-Bitabbildung von dem Speicher zu dem Ausfüllbereich-Teilsatz (**58**); und
e. Schreiben (**40**) des Ausfüllbereich-Teilsatzes (**58**) in einen Ausgabepuffer, der Daten, die zumindest einen Abschnitt des gedruckten Bildes definieren, speichert.

2. Das Verfahren gemäß Anspruch 1, das ferner ein Bestimmen, ob die zum Verarbeiten eines Ausfüllbereich-Teilsatzes (**58**) benötigte Zeit eine vordefinierte Schwelle überschreitet, und, falls dies der Fall ist, ein Vorrasterisieren des Ausfüllbereich-Teilsatzes (**58**) umfasst.

3. Das Verfahren gemäß Anspruch 1, das ferner ein Bestimmen, ob die zum Verarbeiten einer Mehrzahl von Ausfüllbereich-Teilsätzen (**58**), die einen Ausfüllbereich (**58**) umfassen, benötigte Zeit eine vordefinierte Schwelle überschreitet, und falls dies der Fall ist, Vorrasterisieren von Ausfüllbereich-Teilsätzen (**58**), die die vordefinierte Schwelle überschreiten.

4. Das Verfahren gemäß Anspruch 1, bei dem der Ausfüllbereich-Teilsatz (**58**) durch zwei Grenzpunkte (**52**, **54**) definiert ist und der zumindest eine Abschnitt der aus dem Speicher gelesenen Überlagerungsmuster-Bitabbildung so bemessen ist, dass er bei einer Anwendung auf den Ausfüllbereich-Teilsatz in die Grenzpunkte passt.

5. Das Verfahren gemäß Anspruch 1, bei dem der Ausfüllbereich (**48**) eine Mehrzahl von Ausfüllbereich-Teilsätzen (**58**) umfasst; und bei dem Schritte b–e auf zumindest den meisten der Ausfüllbereich-Teilsätze durchgeführt werden, um einen Ausfüllbereich zu definieren, der zumindest einen Abschnitt des Überlagerungsmusters enthält.

6. Das Verfahren gemäß Anspruch 1, bei dem der Ausfüllbereich (**48**) eine Mehrzahl von Ausfüllbereich-Teilsätzen (**58**) umfasst und bei dem zumindest die meisten der Ausfüllbereich-Teilsätze durch zwei

Grenzpunkte (**52**, **54**) definiert sind; und bei dem die Schritte b–e auf zumindest den meisten der Ausfüllbereich-Teilsätze durchgeführt werden, um einen Ausfüllbereich zu definieren, der zumindest einen Abschnitt des Überlagerungsmusters enthält.

7. Das Verfahren gemäß Anspruch 1, das ferner ein Transferieren (**42**) des Inhalts des Ausgabepuffers, um ein Drucken zumindest eines Abschnitts des gedruckten Bildes zu bewirken, umfasst.

8. Das Verfahren gemäß Anspruch 1, bei dem der Ausfüllbereich (**48**) eine Mehrzahl von Ausfüllbereich-Teilsätzen (**58**) umfasst; und Schritte b–e auf zumindest den meisten der Ausfüllbereich-Teilsätze durchgeführt werden, um einen Ausfüllbereich zu definieren, der zumindest einen Abschnitt des Überlagerungsmusters enthält; und das ferner ein Transferieren (**42**) des Inhalts des Ausgabepuffers, um ein Drucken zumindest eines Abschnitts des gedruckten Bildes zu bewirken, umfasst.

9. Das Verfahren gemäß Anspruch 1, bei dem der Ausfüllbereich (**48**) eine Mehrzahl von Ausfüllbereich-Teilsätzen (**58**) umfasst und bei dem zumindest die meisten der Ausfüllbereich-Teilsätze durch zwei Grenzpunkte (**52**, **54**) definiert sind; und bei dem die Schritte b–e auf zumindest den meisten der Ausfüllbereich-Teilsätze durchgeführt werden, um einen Ausfüllbereich zu definieren, der zumindest einen Abschnitt des Überlagerungsmusters enthält; und das ferner ein Transferieren (**42**) des Inhalts des Ausgabepuffers, um ein Drucken zumindest eines Abschnitts des gedruckten Bildes zu bewirken, umfasst.

10. Das Verfahren gemäß Anspruch 1, bei dem der Ausfüllbereich-Teilsatz eine Linie (**50**) aufweist, die sich durch den zumindest einen Ausfüllbereich erstreckt.

11. Das Verfahren gemäß Anspruch 1, bei dem der Leseschritt ein Lesen zumindest eines Mehrbitwortes aus der Überlagerungsmuster-Bitabbildung umfasst.

12. Das Verfahren gemäß Anspruch 1, bei dem der Leseschritt ein Lesen zumindest eines Abschnitts eines 32-Bit-Worts umfasst.

13. Das Verfahren gemäß Anspruch 1, das ferner ein Verwenden des Ausgabepuffers als Eingabemusterleitung in den ROP-Bildprozessorvorgang, um eine Zielbitabbildung zu erzeugen, die gedruckt oder angezeigt wird, umfasst.

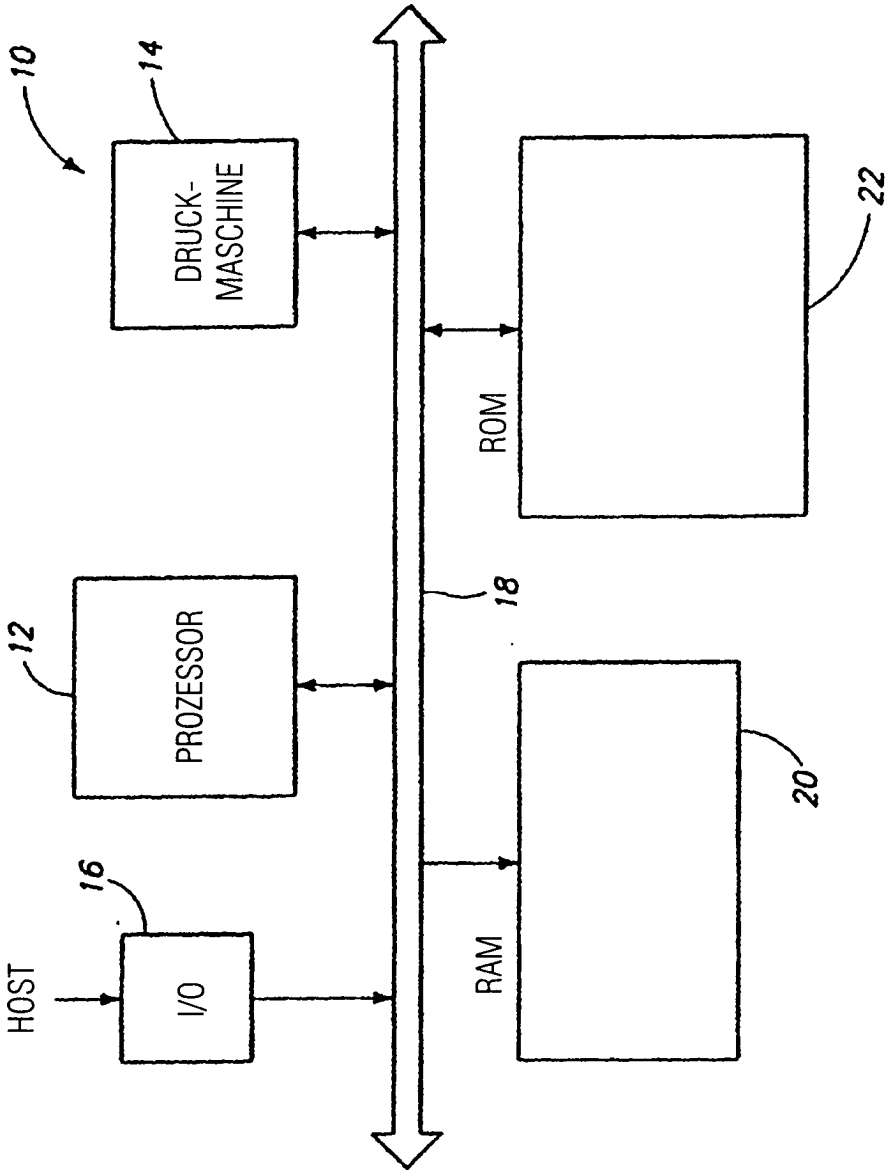
14. Das Verfahren gemäß Anspruch 1, das ferner ein Schätzen von Bildprozessorkosten, die mit einer Erstellung eines Satzes von Ausgabemusterleitungen auf einem Videoband verbunden sind, umfasst.

15. Das Verfahren gemäß Anspruch 1, das ferner eine Verwendung eines Schätzmodells als Teil eines Modells für Bildverarbeitungskosten umfasst, um zu bestimmen, ob bestimmte Videobänder eine Vorrasterisierung erfordern, um einen Zustand einer Druckerüberlastung zu vermeiden.

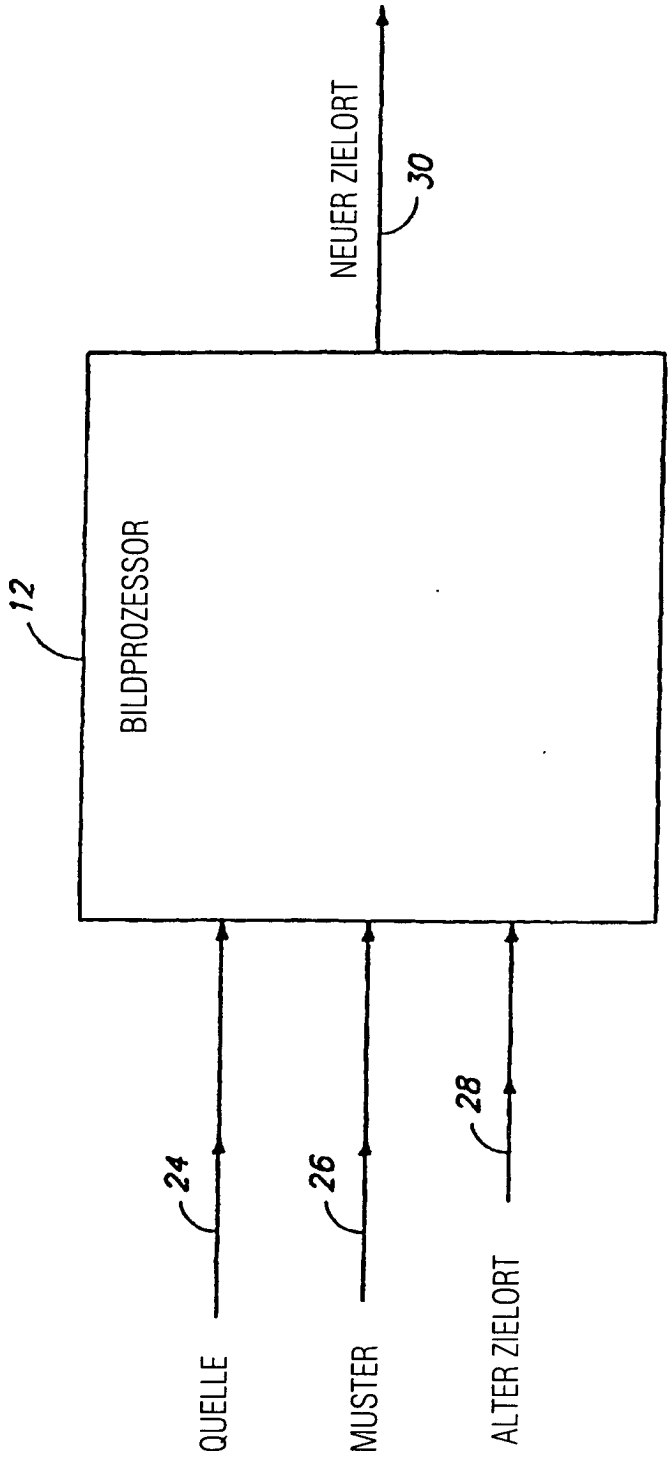
16. Das Verfahren gemäß Anspruch 1, bei dem der Ausgabepuffer einen Zielvideopuffer umfasst.

Es folgen 10 Blatt Zeichnungen

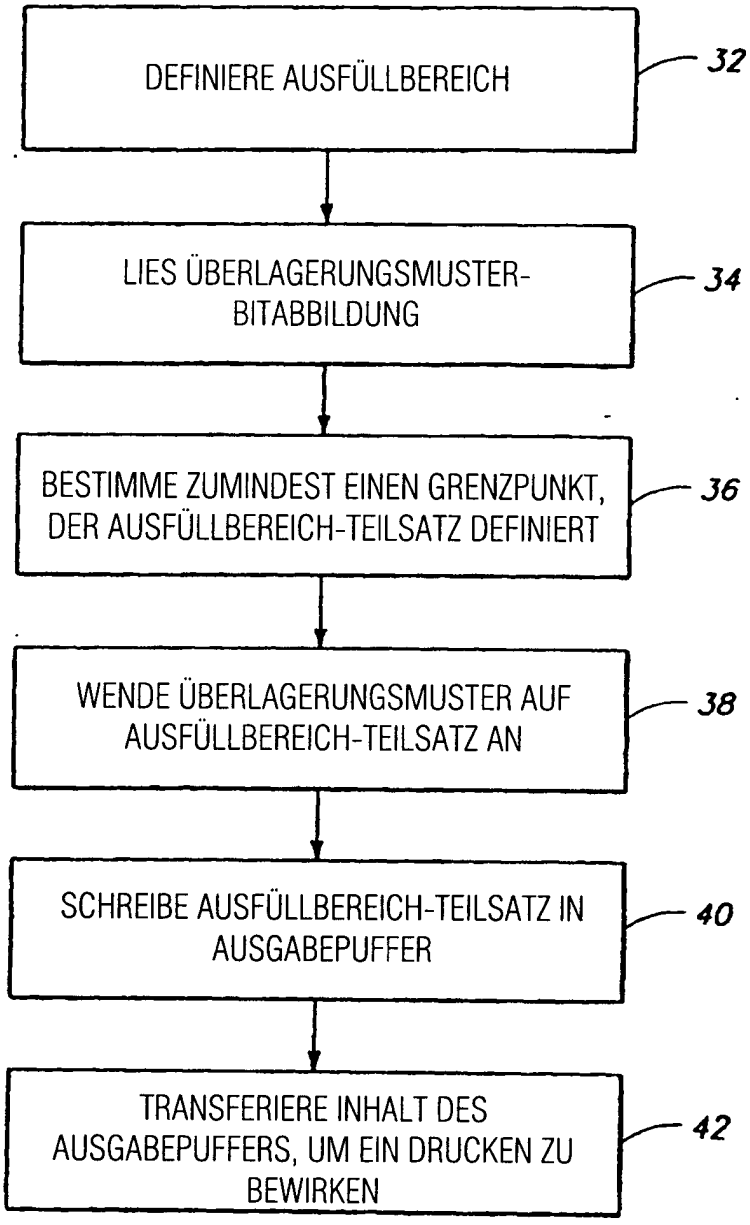
Anhängende Zeichnungen



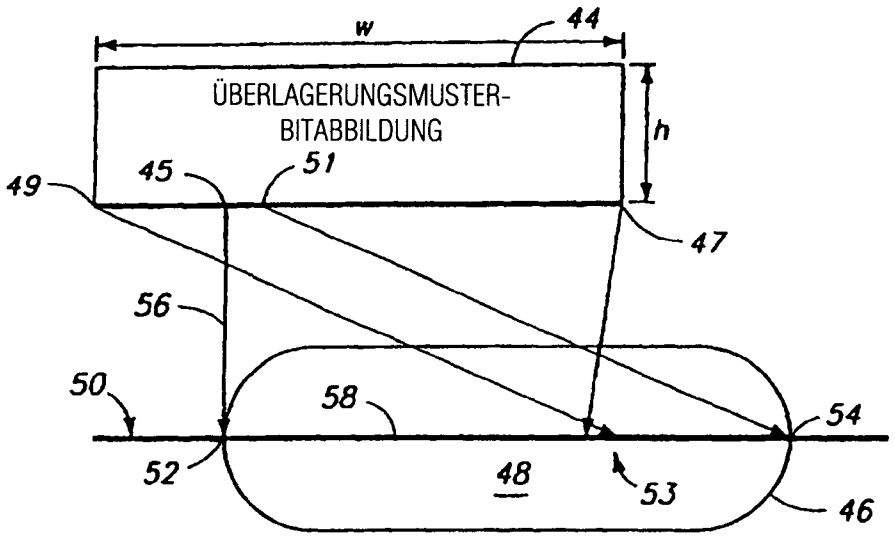
FIGUR 1


NEUER ZIELORT
30
BILDPROZESSOR
12
QUELLE
24
MUSTER
26
ALTER ZIELORT
28

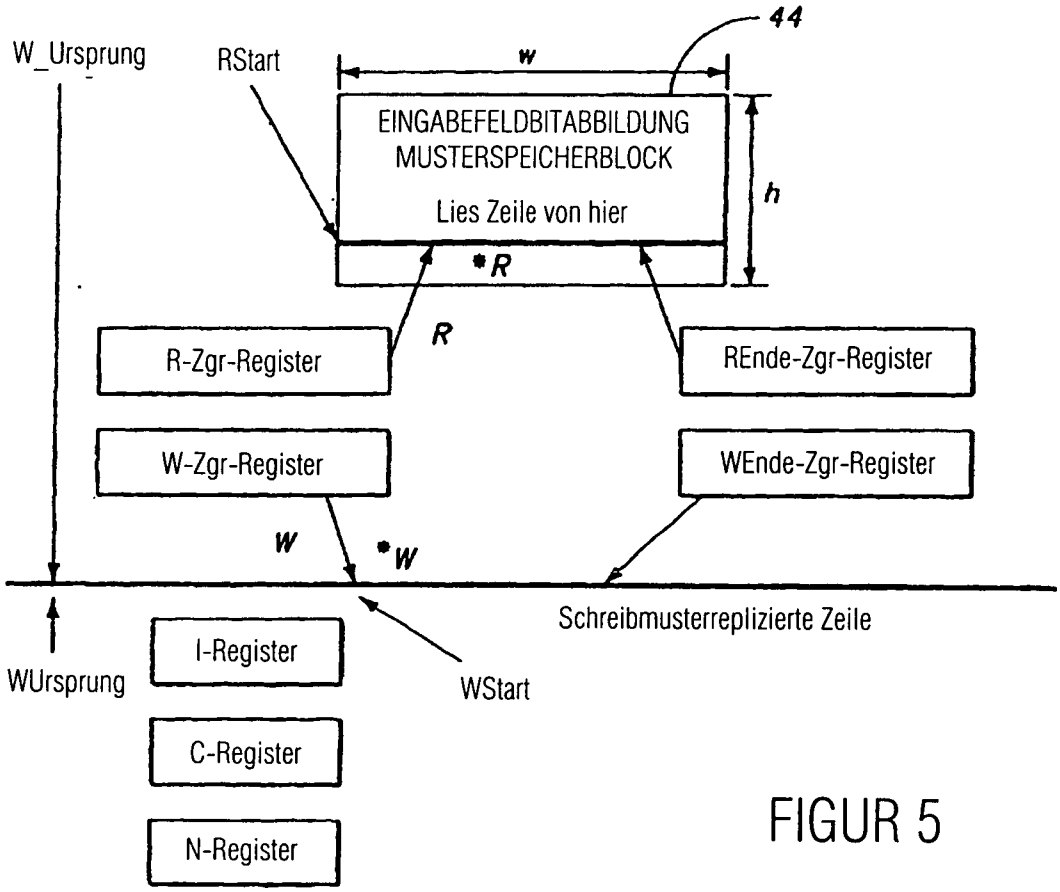
FIGUR 2


DEFINIERE AUSFÜLLBEREICH
32
LIES ÜBERLAGERUNGSMUSTER-BITABBILDUNG
34
BESTIMME ZUMINDEST EINEN GRENZPUNKT, DER AUSFÜLLBEREICH-TEILSATZ DEFINIERT
36
WENDE ÜBERLAGERUNGSMUSTER AUF AUSFÜLLBEREICH-TEILSATZ AN
38
SCHREIBE AUSFÜLLBEREICH-TEILSATZ IN AUSGABEPUFFER
40
TRANSFERIERE INHALT DES AUSGABEPUFFERS, UM EIN DRUCKEN ZU BEWIRKEN
42

FIGUR 3

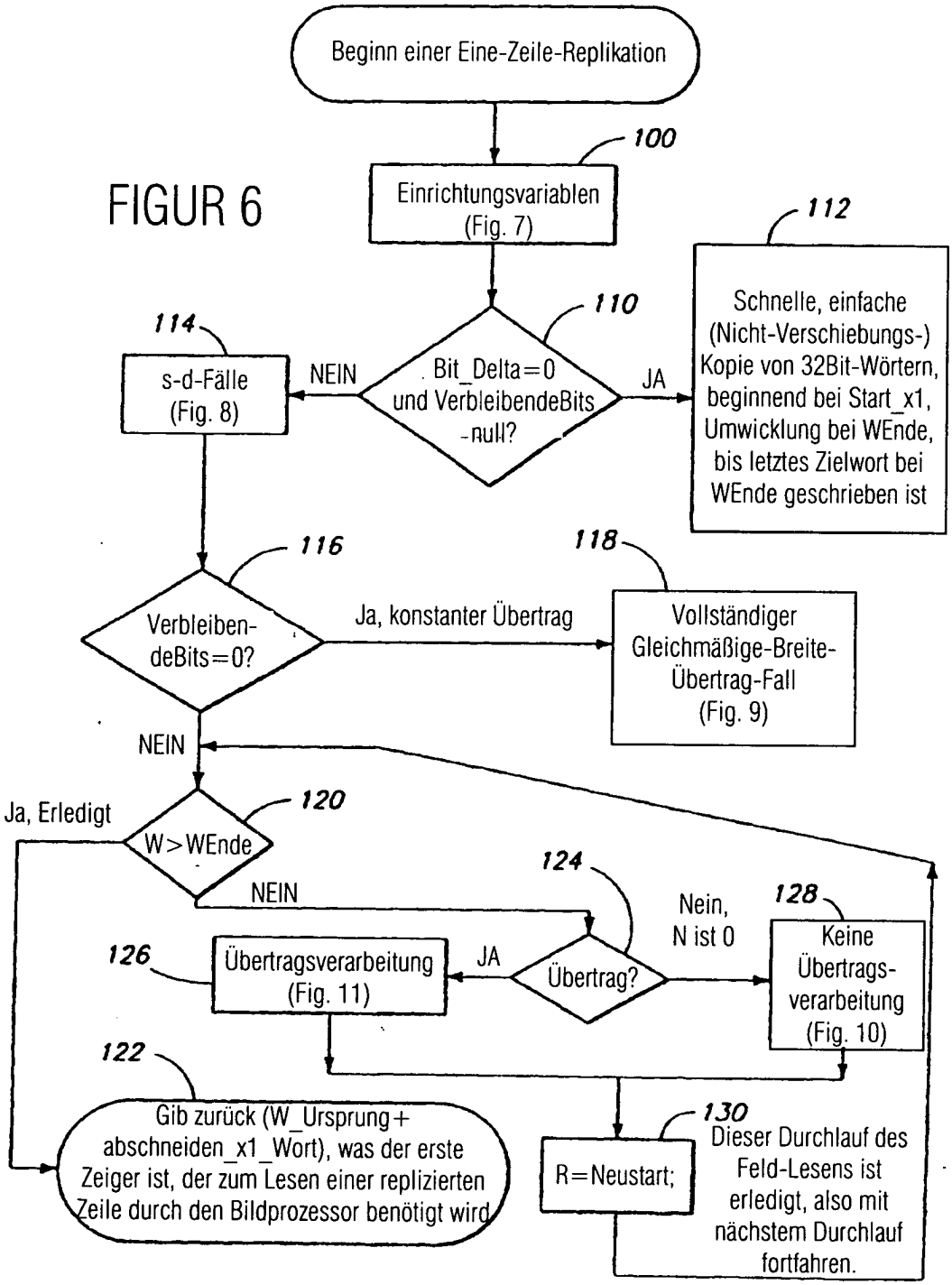


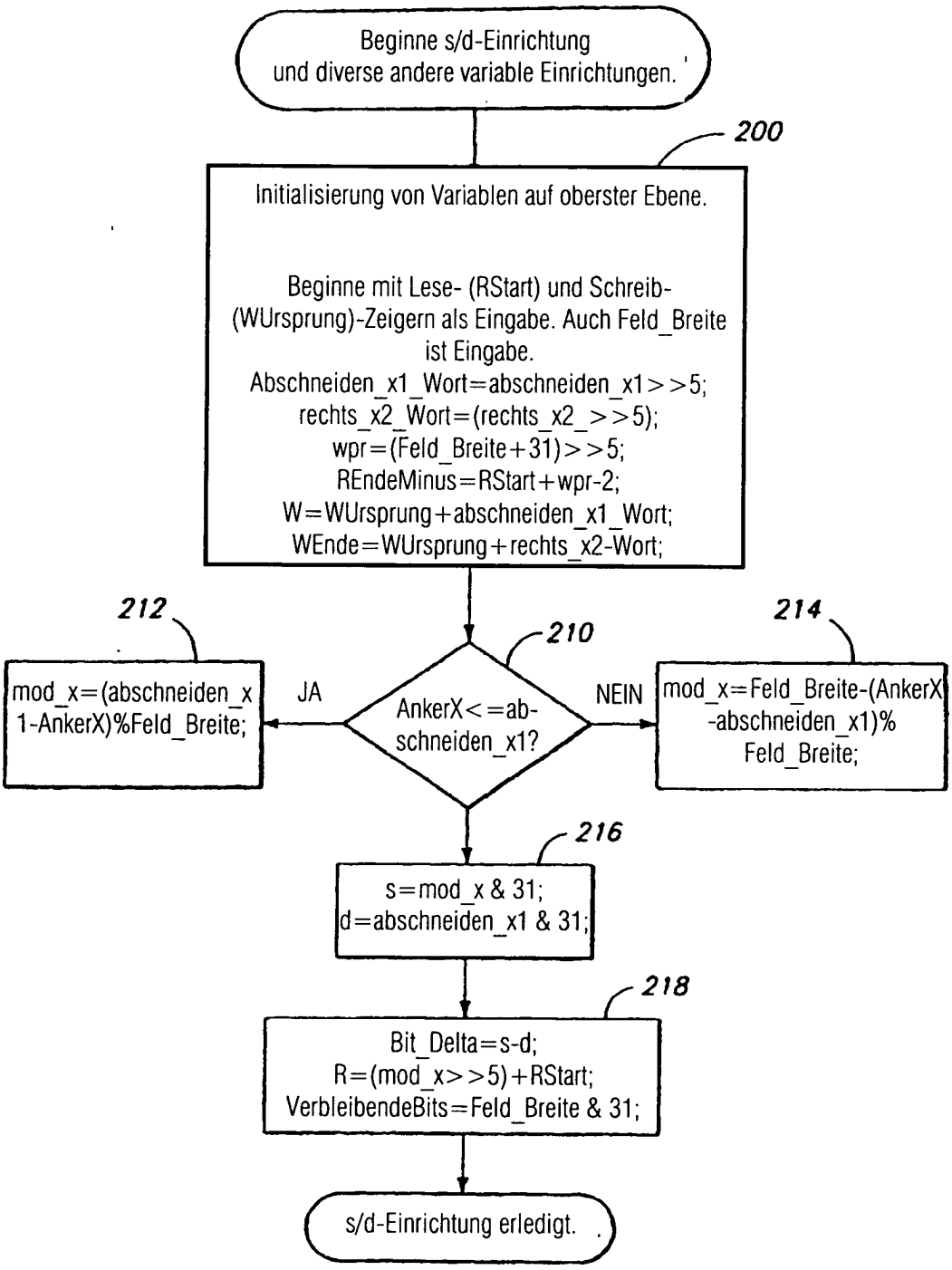
FIGUR 4



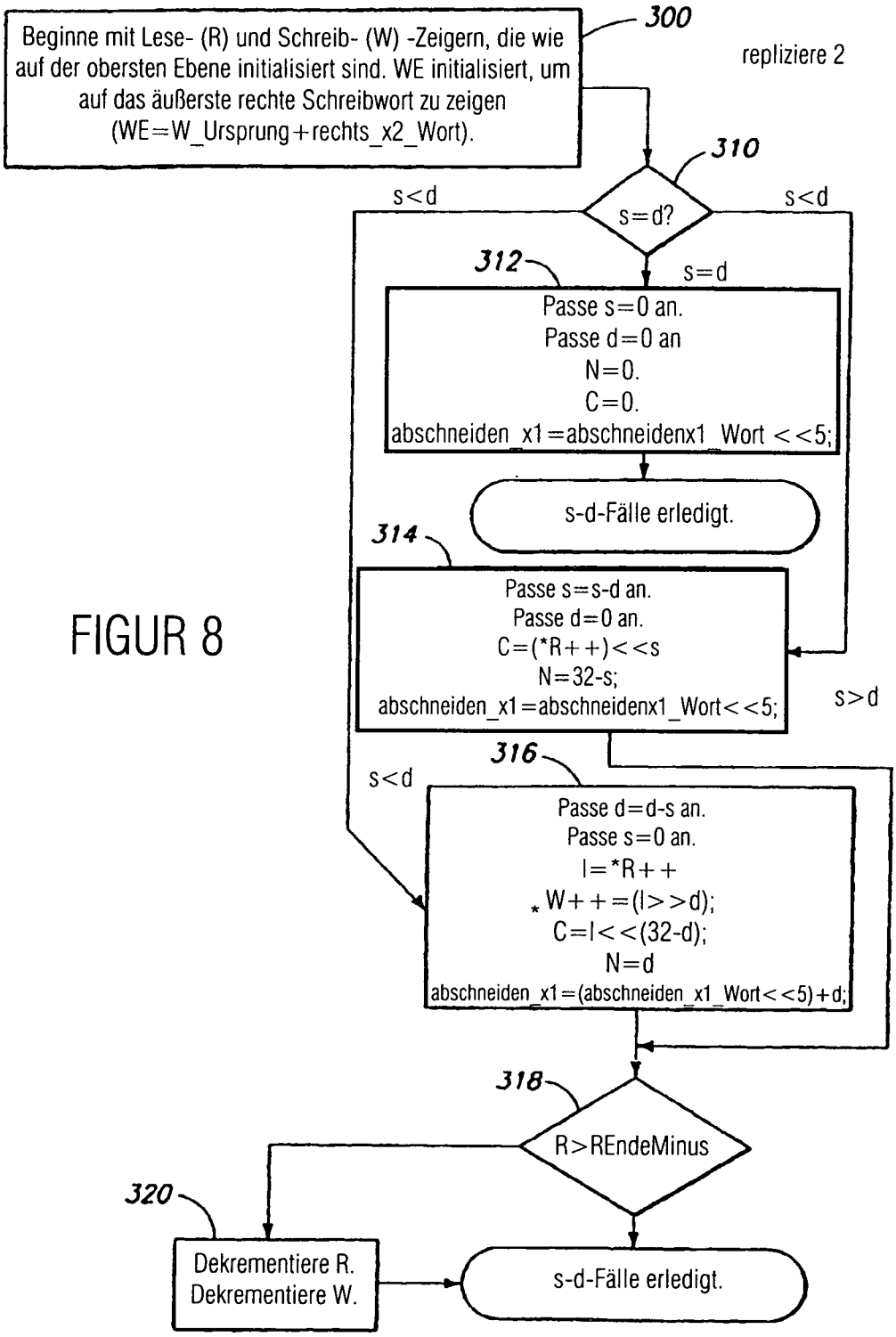
FIGUR 5

FIGUR 6

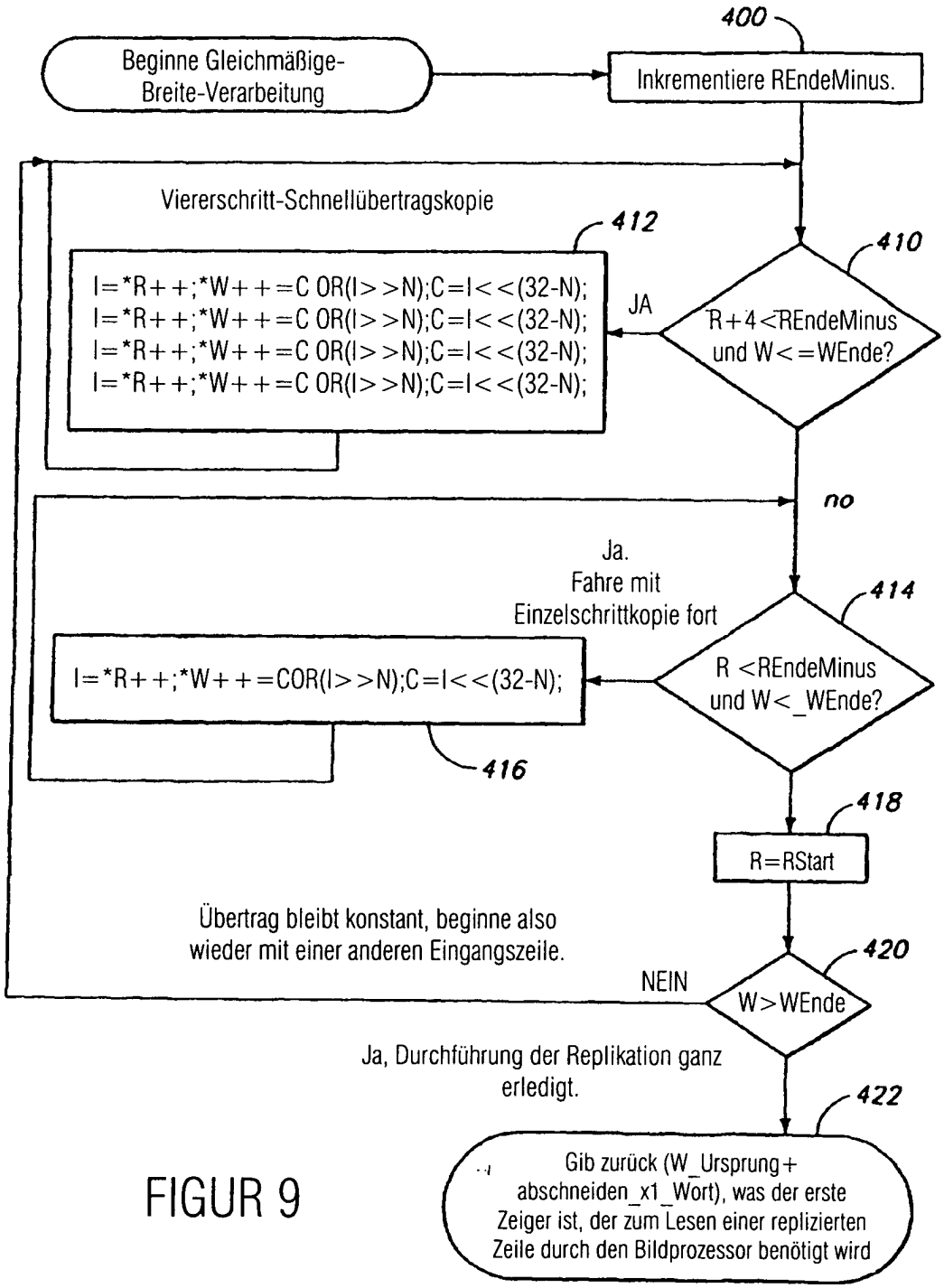

Beginn einer Eine-Zeile-Replikation
100
Einrichtungsvariablen (Fig. 7)
112
Schnelle, einfache (Nicht-Verschiebungs-) Kopie von 32Bit-Wörtern, beginnend bei Start_x1, Umwicklung bei WEnde, bis letztes Zielwort bei WEnde geschrieben ist
110
Bit_Delta=0 und VerbleibendeBits -null?
JA
NEIN
114
s-d-Fälle (Fig. 8)
116
Verbleiben-deBits=0?
Ja, konstanter Übertrag
118
Vollständiger Gleichmäßige-Breite-Übertrag-Fall (Fig. 9)
NEIN
Ja, Erledigt
120
W>WEnde
NEIN
124
Übertrag?
Nein, N ist 0
128
Keine Übertrags-verarbeitung (Fig. 10)
126
Übertragsverarbeitung (Fig. 11)
JA
122
Gib zurück (W_Ursprung+ abschneiden_x1_Wort), was der erste Zeiger ist, der zum Lesen einer replizierten Zeile durch den Bildprozessor benötigt wird
130
R=Neustart;
Dieser Durchlauf des Feld-Lesens ist erledigt, also mit nächstem Durchlauf fortfahren.


Beginne s/d-Einrichtung
und diverse andere variable Einrichtungen.
200
Initialisierung von Variablen auf oberster Ebene.
Beginne mit Lese- (RStart) und Schreib-(WUrsprung)-Zeigern als Eingabe. Auch Feld_Breite ist Eingabe.
Abschneiden_x1_Wort=abschneiden_x1>>5;
rechts_x2_Wort=(rechts_x2_>>5);
wpr=(Feld_Breite+31)>>5;
REndeMinus=RStart+wpr-2;
W=WUrsprung+abschneiden_x1_Wort;
WEnde=WUrsprung+rechts_x2-Wort;
212
mod_x=(abschneiden_x1-AnkerX)%Feld_Breite;
JA
210
AnkerX<=abschneiden_x1?
NEIN
214
mod_x=Feld_Breite-(AnkerX-abschneiden_x1)%Feld_Breite;
216
s=mod_x & 31;
d=abschneiden_x1 & 31;
218
Bit_Delta=s-d;
R=(mod_x>>5)+RStart;
VerbleibendeBits=Feld_Breite & 31;
s/d-Einrichtung erledigt.

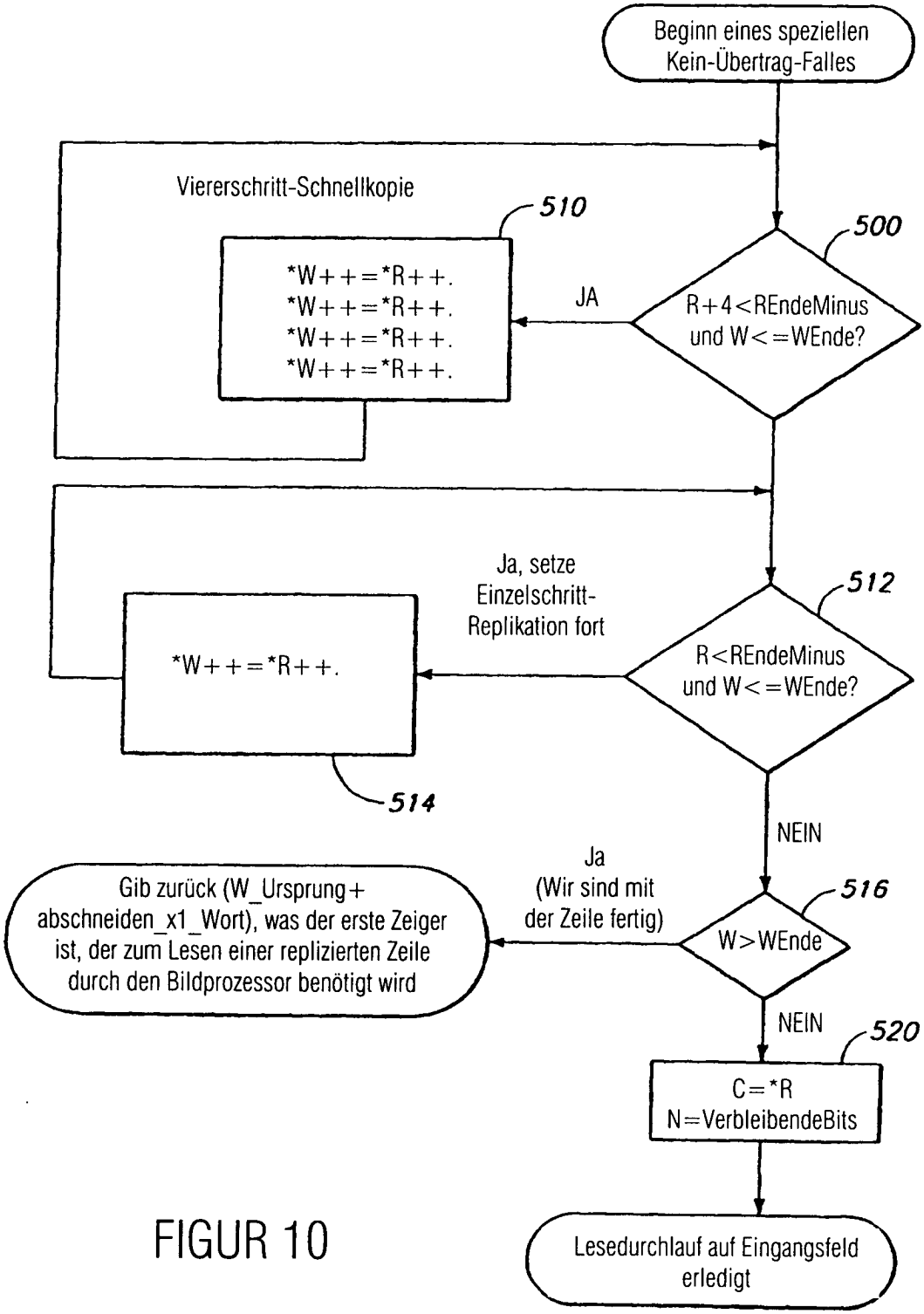
FIGUR 7


Beginne mit Lese- (R) und Schreib- (W) -Zeigern, die wie auf der obersten Ebene initialisiert sind. WE initialisiert, um auf das äußerste rechte Schreibwort zu zeigen (WE=W_Ursprung+rechts_x2_Wort).
300
repliziere 2
310
s=d?
s<d
s<d
s=d
312
Passe s=0 an.
Passe d=0 an
N=0.
C=0.
abschneiden_x1=abschneidenx1_Wort <<5;
s-d-Fälle erledigt.
314
Passe s=s-d an.
Passe d=0 an.
C=(*R++)<<s
N=32-s;
abschneiden_x1=abschneidenx1_Wort<<5;
s>d
316
s<d
Passe d=d-s an.
Passe s=0 an.
l=*R++
*W++=(l>>d);
C=l<<(32-d);
N=d
abschneiden_x1=(abschneiden_x1_Wort<<5)+d;
318
R>REndeMinus
320
Dekrementiere R.
Dekrementiere W.
s-d-Fälle erledigt.

FIGUR 8

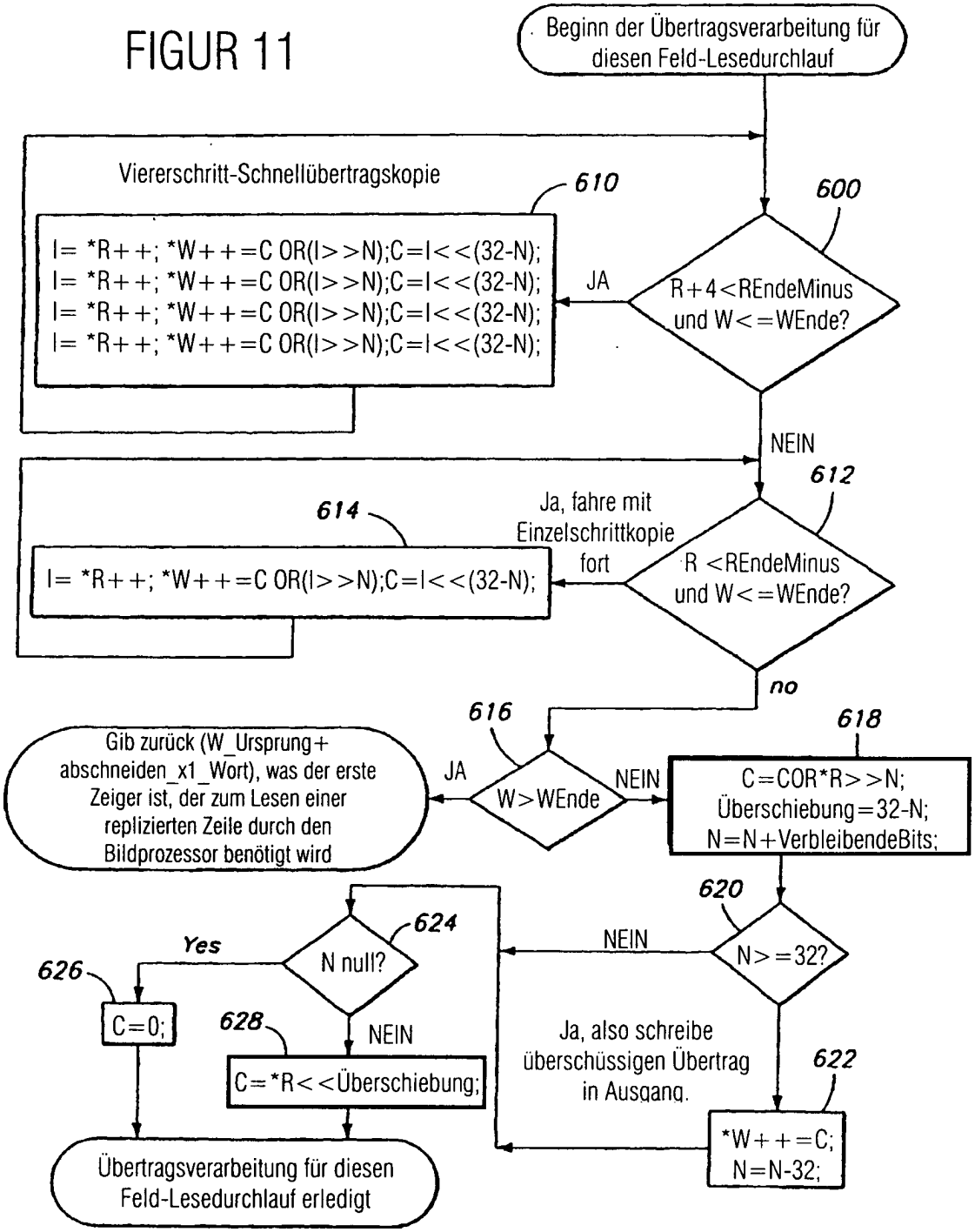

400
Beginne Gleichmäßige-Breite-Verarbeitung
Inkrementiere REndeMinus.
Viererschritt-Schnellübertragskopie
412
I=*R++;*W++=C OR(I>>N);C=I<<(32-N);
I=*R++;*W++=C OR(I>>N);C=I<<(32-N);
I=*R++;*W++=C OR(I>>N);C=I<<(32-N);
I=*R++;*W++=C OR(I>>N);C=I<<(32-N);
JA
410
R+4<REndeMinus und W<=WEnde?
no
Ja. Fahre mit Einzelschrittkopie fort
414
I=*R++;*W++=COR(I>>N);C=I<<(32-N);
R <REndeMinus und W<_WEnde?
416
418
R=RStart
Übertrag bleibt konstant, beginne also wieder mit einer anderen Eingangszeile.
420
NEIN
W>WEnde
Ja, Durchführung der Replikation ganz erledigt.
422
Gib zurück (W_Ursprung+ abschneiden_x1_Wort), was der erste Zeiger ist, der zum Lesen einer replizierten Zeile durch den Bildprozessor benötigt wird

FIGUR 9


Beginn eines speziellen Kein-Übertrag-Falles
Viererschritt-Schnellkopie
510
*W++=*R++.
*W++=*R++.
*W++=*R++.
*W++=*R++.
JA
500
R+4<REndeMinus und W<=WEnde?
Ja, setze Einzelschritt-Replikation fort
512
*W++=*R++.
R<REndeMinus und W<=WEnde?
514
NEIN
Ja (Wir sind mit der Zeile fertig)
516
Gib zurück (W_Ursprung+ abschneiden_x1_Wort), was der erste Zeiger ist, der zum Lesen einer replizierten Zeile durch den Bildprozessor benötigt wird
W>WEnde
NEIN
520
C=*R
N=VerbleibendeBits
Lesedurchlauf auf Eingangsfeld erledigt

FIGUR 10

FIGUR 11


Beginn der Übertragsverarbeitung für diesen Feld-Lesedurchlauf
600
R+4<REndeMinus und W<=WEnde?
JA
610
Viererschritt-Schnellübertragskopie
I= *R++; *W++=C OR(I>>N);C=I<<(32-N);
I= *R++; *W++=C OR(I>>N);C=I<<(32-N);
I= *R++; *W++=C OR(I>>N);C=I<<(32-N);
I= *R++; *W++=C OR(I>>N);C=I<<(32-N);
NEIN
612
R <REndeMinus und W<=WEnde?
Ja, fahre mit Einzelschrittkopie fort
614
I= *R++; *W++=C OR(I>>N);C=I<<(32-N);
no
616
W>WEnde
JA
Gib zurück (W_Ursprung+ abschneiden_x1_Wort), was der erste Zeiger ist, der zum Lesen einer replizierten Zeile durch den Bildprozessor benötigt wird
NEIN
618
C=COR*R>>N;
Überschiebung=32-N;
N=N+VerbleibendeBits;
620
N>=32?
NEIN
Ja, also schreibe überschüssigen Übertrag in Ausgang.
622
*W++=C;
N=N-32;
624
N null?
Yes
626
C=0;
NEIN
628
C=*R<<Überschiebung;
Übertragsverarbeitung für diesen Feld-Lesedurchlauf erledigt