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

(10) **DE 600 33 677 T2** 2007.11.22

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 221 139 B1**
(21) Deutsches Aktenzeichen: **600 33 677.8**
(86) PCT-Aktenzeichen: **PCT/AU00/01075**
(96) Europäisches Aktenzeichen: **00 962 075.8**
(87) PCT-Veröffentlichungs-Nr.: **WO 2001/020549**
(86) PCT-Anmeldetag: **08.09.2000**
(87) Veröffentlichungstag der PCT-Anmeldung: **22.03.2001**
(97) Erstveröffentlichung durch das EPA: **10.07.2002**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **28.02.2007**
(47) Veröffentlichungstag im Patentblatt: **22.11.2007**

(51) Int Cl.[8]: ***G06T 5/00*** (2006.01)

(30) Unionspriorität:
PQ289099 16.09.1999 AU

(84) Benannte Vertragsstaaten:
AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LI, LU, MC, NL, PT, SE

(73) Patentinhaber:
Silverbrook Research Pty. Ltd., Balmain, New South Wales, AU

(74) Vertreter:
Kroher, Strobel Rechts- und Patentanwälte, 80336 München

(72) Erfinder:
LAPSTUN, Paul, Rodd Point, NSW 2046, AU;
WALMSLEY, Simon Robert, Balmain, NSW 2041, AU

(54) Bezeichnung: **GERÄT UND VERFAHREN ZUR HERSTELLUNG EINES PRINTS AUS EINEM "BAYERBILD"**

Beschreibung

Gebiet der Erfindung

[0001] Die vorliegende Erfindung betrifft ein Verfahren und eine Vorrichtung zum Erzeugen eines Ausdrucks von einem Bayer-Bild.

[0002] Die Erfindung wurde in erster Linie für eine Digitalkamera mit einem integrierten Drucker zum Erzeugen eines Papierausdrucks eines Bildes, das von der Kamera erfasst wurde, entwickelt und wird nachfolgend unter Bezugnahme auf diese Anwendung beschrieben. Es versteht sich jedoch, dass die Erfindung nicht auf dieses spezielle Anwendungsgebiet beschränkt ist.

Zusammenfassung der Erfindung

[0003] Gemäß eines ersten Aspekts der Erfindung wird ein Verfahren zum Bereitstellen eines Bildes zum Drucken bei einer vorbestimmten zweiwertigen Rasterpunktauflösung, die einer vorbestimmten Auflösung mit gleichmäßigem Tonverlauf entspricht, bereitgestellt, wobei das Verfahren die folgenden Schritte aufweist:
Empfangen eines ersten Datensatzes, der das Bild angibt, wobei der erste Datensatz aus einem planarisierten Bayer-Format besteht, das sich aus unterschiedlichen Farbebenen (**45**, **46**, **47**) zusammensetzt, wobei mindestens eine der Farbebenen (**46**) eine Auflösung aufweist, die sich von der der anderen Farbebene oder -ebenen (**45**, **47**) unterscheidet;
Durchführen einer Bildrekonstruktion und erneuten Abtastung (**64**) jeder Farbebene des ersten Datensatzes, um einen zweiten Datensatz der vorbestimmten Auflösung mit gleichmäßigem Tonverlauf zu erzeugen, die größer ist als die Auflösung jeder der Farbebenen für den ersten Datensatz;
Umwandeln (**67**) des zweiten Datensatzes in einen dritten Datensatz der vorbestimmten zweiwertigen Rasterpunktauflösung, die größer ist als die vorbestimmte Auflösung mit gleichmäßigem Tonverlauf des zweiten Datensatzes; und
zur Verfügung stellen des dritten Datensatzes für einen Drucker (**69**) in der vorbestimmten zweiwertigen Rasterpunktauflösung.

[0004] Vorzugsweise stimmt die erste Auflösung mit der vorbestimmten zweiwertigen Rasterpunktauflösung überein. In anderen Ausführungsformen ist jedoch die erste Auflösung größer als die vorbestimmte zweiwertige Rasterpunktauflösung. In noch weiteren Ausführungsformen ist die erste Auflösung geringer als die vorbestimmte zweiwertige Rasterpunktauflösung.

[0005] Ebenfalls mit Vorteil ist der erste Datensatz in einem RGB-Format (rot, grün und blau) vorhanden und der Drucker reagiert auf CMY-Format (cyan, magenta und gelb) und das Verfahren umfasst den zusätzlichen Schritt des Umwandelns des dritten Datensatzes aus einem RGB-Format in ein CMY-Format.

[0006] In einer vorteilhaften Form umfasst das Verfahren den Schritt des Schärfens des zweiten Datensatzes. Alternativ umfasst das Verfahren den bevorzugten Schritt des Schärfens des ersten Datensatzes.

[0007] Mit Vorteil wird der erste Datensatz von einer Sensoreinrichtung ermittelt und das Verfahren umfasst den Schritt des Abgleichens des ersten Datensatzes aufgrund von Nicht-Linearitäten in der Sensoreinrichtung. Noch vorteilhafter umfasst der Schritt des Abgleichens das Umwandeln des ersten Datensatzes aus einer Vielzahl von x Bitproben in eine Vielzahl von y Bitproben, wobei $x > y$ gilt. Mit noch weiterem Vorteil ist $x = 10$ und $y = 8$.

[0008] Ebenfalls vorteilhafter Weise umfasst das Verfahren den Schritt des Planarisierens des ersten Datensatzes in eine rote Ebene, eine grüne Ebene und eine blaue Ebene.

[0009] In einer bevorzugten Form umfasst das Verfahren die folgenden Schritte:
Bestimmen der m % dunkelsten Pixel und der n % hellsten Pixel für den ersten Datensatz;
Anpassen des ersten Datensatzes, um die m % dunkelsten Pixel auszugleichen; und
Anpassen des ersten Datensatzes, um die n % hellsten Pixel auszugleichen.

[0010] Mit Vorteil umfasst das Verfahren den zusätzlichen Schritt des Anpassens des ersten Datensatzes, um einen vorbestimmten Weißabgleich bereitzustellen. Mit weiterem Vorteil umfasst das Verfahren den zusätzlichen Schritt des Anpassens des ersten Datensatzes, um eine vorbestimmte Bereichserweiterung bereitzustellen. Sogar noch vorteilhafter wird die Farbauflösung des ersten Datensatzes erhöht, wobei dieselbe räumliche

Auflösung beibehalten wird.

[0011] In einer bevorzugten Form wird der erste Datensatz wahlweise angepasst, um das Bild in einer bestimmten Rotationsorientierung bereitzustellen.

[0012] Gemäß eines zweiten Aspektes der Erfindung wird eine Vorrichtung zum Bereitstellen eines Bildes zum Drucken bei einer vorbestimmten zweiwertigen Rasterpunktauflösung, die einer vorbestimmten Auflösung mit gleichmäßigem Tonverlauf entspricht, bereitgestellt, wobei die Vorrichtung umfasst:
Eingabemittel zum Empfangen eines ersten Datensatzes, der das Bild angibt, wobei der erste Datensatz in einem planarisierten Bayer-Format vorliegt, das unterschiedliche Farbebenen (**45**, **46**, **47**) aufweist, wobei mindestens eine der Farbebenen (**46**) eine Auflösung aufweist, die unterschiedlich zu der der anderen Farbebene oder -ebenen (**45**, **47**) ist,
Tastmittel zur Bildrekonstruktion und zum erneuten Abtasten bzw. Resampling (**64**) des ersten Datensatzes, um einen zweiten Datensatz der vorbestimmten Auflösung mit gleichmäßigem Tonverlauf zu erzeugen, die größer ist als die Auflösung jeder der Farbebenen des ersten Datensatzes;
Verarbeitungsmittel zum Umwandeln (**67**) des zweiten Datensatzes in einen dritten Datensatz der vorbestimmten zweiwertigen Rasterpunktauflösung, die größer ist als die vorbestimmte Auflösung mit gleichmäßigem Tonverlauf des zweiten Datensatzes; und
Ausgabemittel zum Bereitstellen des dritten Datensatzes (**69**) für einen Drucker zum Drucken in der vorbestimmten zweiwertigen Rasterpunktauflösung.

[0013] Vorzugsweise stimmt die erste Auflösung mit der vorbestimmten zweiwertigen Rasterpunktauflösung überein. Alternativ ist die erste Auflösung größer als die vorbestimmte zweiwertige Rasterpunktauflösung. In anderen Ausführungsformen ist jedoch die erste Auflösung geringer als die vorbestimmte zweiwertige Rasterpunktauflösung.

[0014] Ebenfalls vorzugsweise ist der erste Datensatz in einem RGB-Format (rot, grün und blau) vorhanden und der Drucker reagiert auf ein CMY-Format (cyan, magenta und gelb), wobei die Verarbeitungsmittel den zweiten Datensatz von einem RGB-Format in ein CMY-Format umwandeln.

[0015] Ebenfalls mit Vorteil schärft die Vorrichtung den zweiten Datensatz. in einer anderen Ausführungsform schärft jedoch die Vorrichtung den ersten Datensatz.

[0016] In einer bevorzugten Form wird der erste Datensatz aus einer Sensorvorrichtung erhalten und die Eingabemittel gleichen den ersten Datensatz bezüglich Nicht-Linearitäten in der Sensorvorrichtung ab. In vorteilhafterer Weise umfasst der Abgleich bezüglich Nicht-Linearitäten das Umwandeln des ersten Datensatzes aus einer Vielzahl von x Bitproben in eine Vielzahl von y Bitproben, wobei $x > y$ gilt. Sogar noch vorteilhafter gilt $x = 10$ und $y = 8$.

[0017] Vorteilhafterweise planarisieren die Eingabemittel den ersten Datensatz in eine rote Ebene, eine grüne Ebene und eine blaue Ebene.

[0018] Mit besonderem Vorteil:
bestimmen die Eingabemittel für den ersten Datensatz die m % dunkelsten Pixel und die n % hellsten Pixel;
passen die Eingabemittel den ersten Datensatz an, um die m % dunkelsten Pixel abzugleichen; und
passen die Eingabemittel den ersten Datensatz an, um die n % hellsten Pixel abzugleichen.

[0019] In einer bevorzugten Form passen die Eingabemittel den ersten Datensatz an, um einen vorbestimmten Weißabgleich bereitzustellen. Noch vorteilhafter passen die Eingabemittel den ersten Datensatz an, um eine vorbestimmte Bereichserweiterung bereitzustellen. Sogar mit noch größerem Vorteil erhöhen die Eingabemittel die Farbauflösung des ersten Datensatzes, während dieselbe räumliche Auflösung beibehalten wird.

[0020] Mit Vorteil passen die Eingabemittel wahlweise den ersten Datensatz an, um das Bild in einer vorbestimmten Rotationsorientierung bereitzustellen.

[0021] Gemäß eines dritten Aspekts der Erfindung wird eine Kamera bereitgestellt, umfassend:
ein CCD-Array zum Bereitstellen eines Bayer-Bildes;
einen Drucker zum wahlweisen Bereitstellen eines gedruckten Bildes; und
eine wie oben beschriebene Vorrichtung zum Empfangen des Bayer-Bildes und Beliefern des Druckers mit dem dritten Datensatz, so dass das gedruckte Bild erzeugt wird.

Kurze Beschreibung der Figuren

[0022] Bevorzugte Ausführungsformen der Erfindung werden nun lediglich beispielhaft unter Bezugnahme auf die nachfolgende Beschreibung und die Figuren beschrieben.

[0023] Fig. 1 zeigt einen abstrahierten Bildfluss des PCP.

[0024] Fig. 2 zeigt ein Blockschaltbild des PCP für sich genommen.

[0025] Fig. 3 zeigt ein Blockschaltbild des PCP, der mit der Printcam-Hardware verbunden ist.

[0026] Fig. 4 zeigt einen 4-Zoll-Memjet-Druckkopf.

[0027] Fig. 5 zeigt die Anordnung von Segmenten in einem 4-Zoll-Druckkopf.

[0028] Fig. 6 zeigt die Anordnung von Düsen in einem Sockel, nummeriert nach Auslass-Reihenfolge.

[0029] Fig. 7 zeigt die Anordnung von Düsen in einem Sockel, nummeriert nach Lade-Reihenfolge.

[0030] Fig. 8 zeigt einen Chromapod bzw. Farbensockel.

[0031] Fig. 9 zeigt eine Sockelgruppe.

[0032] Fig. 10 zeigt eine Phasengruppe.

[0033] Fig. 11 zeigt die Beziehung zwischen Segmenten, Auslassgruppen, Phasengruppen, Sockelgruppen und Chromapods.

[0034] Fig. 12 zeigt AEnable und BEnable-Impulsprofile während des Druckens eines ungeraden und eines geraden Punktes.

[0035] Fig. 13 zeigt die Orientierung von Druckformaten basierend auf dem CFA-Bild.

[0036] Fig. 14 zeigt ein Blockschaltbild der Bilderfassungskette.

[0037] Fig. 15 zeigt die Anordnung von Pixeln in einem Bayer-CFA-2G-Mosaik.

[0038] Fig. 16 zeigt den Prozess des Linearisierens von RGB.

[0039] Fig. 17 zeigt den Prozess des Planarisierens von RGB.

[0040] Fig. 18 zeigt ein Blockschaltbild der Bilddruckkette.

[0041] Fig. 19 zeigt einen beispielhaften Farbbereich für eine einfache Farbebene.

[0042] Fig. 20 zeigt die Schritte, die zum Weißabgleich und zur Bereichserweiterung gehören.

[0043] Fig. 21 zeigt ein Blockschaltbild der Vorrichtung, die geeignet ist, einen Weißabgleich und eine Bereichserweiterung durchzuführen.

[0044] Fig. 22 zeigt die unterschiedlichen Farbebenenpixel in Beziehung zur CFA-Auflösung.

[0045] Fig. 23 zeigt die Wirkung der Rotation der grünen Ebene um 45 Grad.

[0046] Fig. 24 zeigt den Abstand zwischen rotierten Pixeln für die grüne Ebene.

[0047] Fig. 25 zeigt den Prozess des Abbildens der Bewegung von nicht rotiertem CFA-Raum zu rotiertem CFA-Raum.

[0048] Fig. 26 zeigt ein Blockschaltbild des Schärfungsprozesses.

[0049] Fig. 27 zeigt den Prozess, der zur Hochpassfilterung eines einzelnen Helligkeitspixels mit einem 3 × 3-Kern gehört.

[0050] Fig. 28 zeigt die Transformation bei der Umwandlung von RGB zu CMY.

[0051] Fig. 29 zeigt die Umwandlung von RGB zu CMY durch trilineare Interpolation.

[0052] Fig. 30 zeigt die Pixelreplikation eines einzelnen Pixels auf einen 5 × 5-Block.

[0053] Fig. 31 zeigt ein Blockschaltbild des Halbton-Prozesses (half-toning).

[0054] Fig. 32 zeigt den Prozess des Reformatierens von Punkten für den Drucker.

[0055] Fig. 33 zeigt ein Blockschaltbild der Bilderfassungseinheit.

[0056] Fig. 35 zeigt ein Blockschaltbild der Bildzugangseinheit.

[0057] Fig. 36 zeigt ein Blockschaltbild der Bildhistogrammeinheit.

[0058] Fig. 37 zeigt ein Blockschaltbild der Druckerschnittstelle.

[0059] Fig. 38 zeigt das Blockschaltbild der Memjet-Schnittstelle.

[0060] Fig. 39 zeigt die Generierung von AEnable und BEnable-Impulsbreiten.

[0061] Fig. 40 zeigt ein Blockschaltbild der Punktezähllogik.

[0062] Fig. 41 zeigt die Schnittstelle der Druckerzeugungseinheit.

[0063] Fig. 42 zeigt ein Blockschaltbild der Druckerzeugungseinheit.

[0064] Fig. 43 zeigt ein Blockschaltbild der Testmusterzugangseinheit.

[0065] Fig. 44 zeigt ein Blockschaltbild des Zwischenspeichers 5.

[0066] Fig. 45 zeigt ein Blockschaltbild des Zwischenspeichers 4.

[0067] Fig. 46 zeigt ein Blockschaltbild des Hochinterpolier-, Halbton- und Reformatierungsprozesses.

[0068] Fig. 47 zeigt, wie man von einer Standard-Dither-Zelle zu einer versetzten Dither-Zelle abbildet.

[0069] Fig. 48 zeigt ein Blockschaltbild des Prozesses Umwandeln-von-RGB-zu-CMY.

[0070] Fig. 49 zeigt ein Blockschaltbild des Zwischenspeichers 2.

[0071] Fig. 50 zeigt einen Basis-Hochpass-Raumfilter unter Verwendung eines 3 × 3-Kerns.

[0072] Fig. 51 zeigt ein Blockschaltbild der Schärfungseinheit.

[0073] Fig. 52 zeigt die Struktur des Zwischenspeichers 1.

[0074] Fig. 53 zeigt ein Blockschaltbild des Prozesses "Erneut Abtasten und Helligkeitskanal Erzeugen" (Resample and Create Luminance Channel).

[0075] Fig. 54 zeigt ein Blockschaltbild der Faltungseinheit.

[0076] Fig. 55 zeigt die Reihenfolge von Pixeln, die von dem Rezeptor erzeugt wurden.

[0077] Fig. 56 zeigt die Bewegung in x- oder y-Richtung im gedrehten und ungedrehten Raum.

[0078] Fig. 57 zeigt die Adresse von Einträgen in dem grünen Unter-Zwischenspeicher des Zwischenspeichers 1.

[0079] Fig. 58 zeigt die Beziehung zwischen grünen Einträgen, die von Rotation abhängig sind.

[0080] Fig. 59 zeigt eine 4 × 4-Abtastung des grünen Kanals.

[0081] Fig. 60 zeigt ein 4 × 4-grün-Abtasten vom Typ 1.

[0082] Fig. 61 zeigt ein 4 × 4-grün-Abtasten vom Typ 2.

[0083] Fig. 62 zeigt die zwei Arten von Zeilenadressierung für Grün.

[0084] Fig. 63 zeigt die Adressierung von Einträgen in den roten und blauen Unter-Zwischenspeichern des Zwischenspeichers 1.

[0085] Fig. 64 zeigt die ersten 16 Abtastproben, die für die Berechnung des ersten Pixels gelesen wurden.

[0086] Fig. 65 zeigt die überlappende Worst-Case-4 × 4-Ablesung aus den blauen und roten Zwischenspeichern.

[0087] Fig. 66 zeigt ein Blockschaltbild der Drehungs-, Weißabgleichs- und Bereichserweiterungseinheiten.

[0088] Fig. 67 zeigt den aktiven Bildbereich innerhalb des erzeugten Koordinatenraumes.

1. ÜBERBLICK DES PCP

1.1 ABSTRAKTER FUNKTIONALER ÜBERBLICK

[0089] Der zentrale Printcam-Prozessor (Printcam Central Processor; PCP) weist die gesamte Verarbeitungsleistung für eine Printcam auf und ist insbesondere dafür ausgebildet, in dem Printcam-Digitalkamerasystem verwendet zu werden. Der PCP **3** ist mit einem Bildsensor **1** (zur Bilderfassung) und einem Memjet-Drucker **2** zum Bildausdruck verbunden. In Bezug auf die Bildverarbeitung kann man sich den PCP vorstellen als Übersetzer von Bildern von der Erfassung zum Drucken, wie in Fig. 1 dargestellt:

- Der Bildsensor **1** ist ein CMOS-Bildsensor, der ein 1500 × 1000 RGB-Bild aufnimmt. Der Bildsensor ist die Bildeingabevorrichtung.
- Der Druckkopf **2** ist ein 4 Zoll langer 1600 dpi-Memjet-Drucker, der geeignet ist, in drei Farben zu drucken: cyan, magenta und gelb. Der Druckkopf ist die Bildausgabevorrichtung.
- Der PCP **3** nimmt ein Bild von dem Bildsensor **1**, verarbeitet es und sendet die finale Form des Bildes zu dem Druckkopf **2** zum Drucken. Da der Bildsensor **1** das Bild in RGB erfasst und der Druckkopf **2** in CMY ausdruckt, muss der PCP **3** von dem RGB-Farbraum in den CMY-Farbaum übersetzen. Der PCP **3** enthält alle Anforderungen für die Zwischenbildverarbeitung einschließlich Weißabgleich, Farbkorrektur und Farbraumanpassung (gamut mapping), Blldschärfung und Rasterung (half toning). Zusätzlich steuert der PCP **3** die Benutzerschnittstelle und den gesamten Druckprozess, indem er Unterstützung für eine Vielzahl von Bildformaten bereitstellt. Der PCP **3** enthält ebenfalls Schnittstellen, die den Export und den Import von Fotos ermöglichen, indem er dem DPOF (Digital Print Order Format) Standard entspricht.

1.2 ABSTRAKTE INTERNE ÜBERSICHT

[0090] Der PCP **3** ist konzipiert, unter Verwendung eines 0,25 µm-CMOS-Prozesses hergestellt zu werden, mit etwa 10 Millionen Transistoren, von denen fast die Hälfte Flash-Speicher oder statischer RAM-Speicher ist. Dies führt zu einer geschätzten Fläche von 16 mm^2. Die geschätzten Herstellungskosten betragen 4 US$ im Jahr 2001. Der PCP **3** weist ein relativ überschaubares Design auf, und der Designaufwand kann durch die Verwendung von Datenpfadkompilierungstechniken, Makrozellen und IP-Kernen reduziert werden. Der PCP **3** umfasst:

- eine langsame CPU/Mikrocontroller-Kern **10**
- 1,5 MByte mehrstufigen Flash-Speicher (2-Bits pro Zelle) **11**
- eine CMOS-Bildsensorschnittstelle **98** innerhalb einer Bilderfassungseinheit **12**
- 16 KByte Flash-Speicher für Programmspeicher **13**
- 4 KByte RAM-Speicher für Programmvariablenspeicher **14**

[0091] Der PCP **3** wurde geplant, um bei einer Taktfrequenz von etwa 100 MHz bei 3V extern und 1,5V intern zu laufen, um den Stromverbrauch zu minimieren. Die tatsächliche Arbeitsfrequenz ist ein ganzzahliges Vielfaches der Arbeitsfrequenz des Druckkopfes. Die CPU **10** soll eine einfache CPU im Stile eines Mikrocontrollers sein, die etwa bei 1 MHz läuft. Sowohl die CPU **10** als auch die CMOS-Sensorschnittstelle **12** können Kerne sein, die von Lieferanten geliefert werden.

[0092] Fig. 2 zeigt ein Blockschaltbild des PCP **3** alleine.

[0093] Der PCP **3** ist für die Verwendung in Printcam-Systemen konzipiert. Fig. 3 zeigt ein Blockdiagramm des PCP **3**, der mit dem Rest der Printcam-Hardware verbunden ist.

2. DRUCKKOPF-HINTERGRUND

[0094] Der PCP **3** ist insbesondere konzipiert, um mit einem 4 Zoll (10 cm) – Memjet-Druckkopf **2** verbunden zu werden. Der Druckkopf **2** wird als ein Querformat-Drucker verwendet, indem er ein 4 Zoll breites gedrucktes Bild erzeugt, ohne bewegt werden zu müssen. Stattdessen wird auf das Papier **20** gedruckt, wenn es sich am Druckkopf **2** vorbei bewegt, wie in Fig. 4 gezeigt.

2.1 ZUSAMMENSETZUNG DES 4-ZOLL-DRUCKKOPFES

[0095] Jeder 4-Zoll-Druckkopf **2** besteht aus 8 Segmenten, wobei jedes Segment 0,5 Zoll lang ist. Jedes der Segmente **21** druckt zweiwertig cyan, magenta und gelbe Punkte über einen unterschiedlichen Anteil der Seite, um das letztendliche Bild zu erzeugen. Die Positionen der Segmente sind Fig. 5 dargestellt.

[0096] Da der Druckkopf **2** Punkte mit 1600 dpi druckt, ist jeder Punkt 22,5 µm im Durchmesser und 15,875 µm voneinander beabstandet. Damit druckt jedes 0,5 Zoll große Segment 800 Punkte, wobei die 8 Segmente folgenden Positionen entsprechen:

Tabelle 1. Finale Bildpunkte, die von jedem Segment adressiert werden

Segment	Erster Punkt	Letzter Punkt
0	0	799
1	800	1.599
2	1.600	2.399
3	2.400	3.199
4	3.200	3.999
5	4.000	4.799
6	4.800	5.599
7	5.600	6.399

[0097] Obwohl jedes Segment **21** 800 Punkte des endgültigen Bildes erzeugt, wird jeder Punkt durch eine Kombination von zweiwertiger cyan, magenta und gelber Tinte dargestellt. Da der Druck zweiwertig erfolgt, sollte das Eingabebild gedithert oder Fehler-diffundiert (errordiffused) werden, um beste Ergebnisse zu erzielen.

[0098] Jedes Segment **21** enthält dann 2400 Düsen: jeweils 800 für cyan, magenta und gelb. Ein 4-Zoll-Druckkopf **2** umfasst 8 solcher Segmente **21** und damit eine Gesamtzahl von 19.200 Düsen.

2.1.1 Gruppieren der Düsen innerhalb eines Segments

[0099] Die Düsen **22** innerhalb eines einzelnen Segments **21** werden zum Zwecke der physikalischen Stabilität sowie der Minimierung des Stromverbrauchs während des Druckens gruppiert. Bezüglich der physikalischen Stabilität teilen sich eine Gesamtzahl von 10 Düsen das gleiche Tintenreservoir. Hinsichtlich des Strom-

verbrauchs werden die Gruppierungen ausgeführt, um einen Printmodus mit niedriger Geschwindigkeit und einen Printmodus mit hoher Geschwindigkeit zu ermöglichen.

[0100] Der Druckkopf **2** unterstützt zwei Druckgeschwindigkeiten, um verschiedene Geschwindigkeit-/Stromverbrauch-Kompromisse zu ermöglichen, die in unterschiedlichen Produktkonfigurationen ausgeführt werden.

[0101] Beim Druckmodus mit niedriger Geschwindigkeit werden 96 Düsen **22** gleichzeitig aus jedem 4-Zoll-Druckkopf **2** gezündet. Die gezündeten Düsen sollten maximal voneinander beabstandet sein, deshalb werden 12 Düsen **22** aus jedem Segment gezündet. Um sämtliche 19.200 Düsen zu öffnen, müssen 200 unterschiedliche Sätze von 96 Düsen geöffnet werden.

[0102] In dem Druckmodus mit hoher Geschwindigkeit werden 192 Düsen **22** gleichzeitig aus jedem 4-Zoll-Druckkopf **2** gezündet. Die gezündeten Düsen **22** sollten maximal voneinander entfernt sein, deshalb werden 24 Düsen aus jedem Segment gezündet. Um sämtliche 19.200 Düsen zu zünden, müssen 100 unterschiedliche Sätze von 192 Düsen gezündet werden.

[0103] Der Stromverbrauch in dem Modus mit niedriger Geschwindigkeit ist halb so groß wie in dem Modus mit hoher Geschwindigkeit. Es ist jedoch anzumerken, dass die verbrauchte Energie, um eine Zeile zu drucken, und damit eine Seite, in beiden Fällen dieselbe ist.

[0104] In einem Szenario wie z. B. einer Batterie getriebenen Printcam geben die Stromverbrauchsanforderungen die Verwendung des Druckens mit niedriger Geschwindigkeit vor.

2.1.1.1 10 Düsen bilden einen Sockel

[0105] Ein einzelner Sockel **23** besteht aus 10 Düsen **22**, die sich ein gemeinsames Tintenreservoir teilen. 5 Düsen **22** sind in einer Zeile angeordnet und 5 in einer weiteren. Jede Düse **22** erzeugt Punkte von etwa 22,5 µm im Durchmesser, die auf einem 15,875 µm-Raster beabstandet sind. Fig. 6 zeigt die Anordnung eines einzelnen Sockels, wobei die Düsen **22** gemäß der Reihenfolge nummeriert sind, in der sie gezündet werden müssen.

[0106] Obwohl die Düsen **22** in dieser Reihenfolge gezündet werden, ist die Beziehung der Düsen **22** und die physikalische Anordnung von Punkten auf der gedruckten Seite unterschiedlich. Die Düsen **22** aus einer Zeile stellen die geradzahligen Punkte einer Zeile auf der Seite dar, und die Düsen auf der anderen Zeile stellen die ungeradzahligen Punkte von der daran angrenzenden Zeile auf der Seite dar. Fig. 7 zeigt denselben Sockel **23**, wobei die Düsen **22** entsprechend der Reihenfolge, in der sie geladen werden müssen, nummeriert sind.

[0107] Die Düsen **22** innerhalb eines Sockels **23** sind deshalb durch die Breite eines Punktes logisch getrennt. Der exakte Abstand zwischen den Düsen **22** hängt von den Eigenschaften des Memjet-Zündmechanismus ab. Der Druckkopf **2** ist mit versetzten Düsen konzipiert, die gestaltet sind, um dem Fluss von Papier **20** zu entsprechen.

2.1.1.2 3 Sockel bilden ein Chromapod bzw. Farbensockel

[0108] Ein Sockel **23** aus jeder Farbe (cyan, magenta und gelb) wird in ein Chromapod **24** gruppiert. Ein Chromapod **24** stellt unterschiedliche Farbkomponenten desselben horizontalen Satzes von 10 Punkten auf unterschiedlichen Zeilen dar. Der exakte Abstand zwischen unterschiedlichen Farbsockeln **23** hängt von den Memjet-Betriebsparametern ab, und kann je nach Memjet-Design variieren. Der Abstand wird als eine konstante Anzahl von Punktbreiten betrachtet und muss deshalb beim Drucken berücksichtigt werden: die Punkte, die von den cyan-farbigen Düsen gedruckt werden, sind für andere Zeilen als die, die von den magenta-farbigen oder gelben Düsen gedruckt werden. Der Druckalgorithmus muss einen variablen Abstand bis zu etwa 8 Punktbreiten zwischen den Farben ermöglichen (siehe Tabelle 3 zu den Einzelheiten). Fig. 8 veranschaulicht ein einzelnes Chromapod **24**.

2.1.1.3 5 Chromapods bilden eine Sockelgruppe

[0109] 5 Chromapods **24** sind in einer einzelnen Sockelgruppe bzw. podgroup **25** organisiert. Da jedes Chromapod **30** Düsen **22** umfasst, enthält jede Sockelgruppe 150 Düsen **22**: 50 cyan-farbige, 50 magenta-farbige und 50 gelbe Düsen. Die Anordnung ist in Fig. 9 dargestellt, wobei die Chromapods mit 0-4 nummeriert sind. Es ist anzumerken, dass der Abstand zwischen aneinander grenzenden Chromapods zum Zwecke der

Klarheit übertrieben dargestellt ist.

2.1.1.4 2 Sockelgruppen bilden eine Phasengruppe

[0110] 2 Sockelgruppen **25** sind in einer einzelnen Phasengruppe **26** organisiert. Die Phasengruppe **26** wird so genannt, da Gruppen von Düsen **23** innerhalb einer Phasengruppe gleichzeitig während einer gegebenen Zündphase gezündet werden (dies wird detaillierter nachfolgend erläutert). Die Bildung einer Phasengruppe aus 2 Sockelgruppen **25** dient vollständig dem Zweck des Druckens mit niedriger und hoher Geschwindigkeit über 2 PodgroupEnable-Zeilen.

[0111] Während des Druckens mit niedriger Geschwindigkeit wird nur eine der beiden PodgroupEnable-Leitungen in einem gegebenen Zündimpuls gesetzt, so dass nur eine Sockelgruppe der beiden Düsen zündet. Während des Druckens mit hoher Geschwindigkeit sind beide PodgroupEnable-Leitungen gesetzt, so dass beide Sockelgruppen Düsen zünden. Folglich benötigt ein Druck mit niedriger Geschwindigkeit doppelt so lange wie ein Druck mit hoher Geschwindigkeit, da der Druck mit hoher Geschwindigkeit doppelt so viele Düsen zur selben Zeit zündet.

[0112] Fig. 10 veranschaulicht die Zusammensetzung einer Phasengruppe. Der Abstand zwischen aneinander grenzenden Sockelgruppen ist zum Zwecke der Klarheit übertrieben dargestellt.

2.1.1.5 2 Phasengruppen bilden eine Zündgruppe

[0113] Zwei Phasengruppen (PhasegroupA und PhasegroupB) sind in einer einzelnen Zündgruppe **27** organisiert, mit 4 Zündgruppen in jedem Segment. Die Zündgruppen **27** werden so genannt, da sie alle dieselben Düsen **27** gleichzeitig zünden. Zwei Einschaltleitungen AEnable und BEnable ermöglichen das Zünden von PhasegroupA-Düsen und PhasegroupB-Düsen unabhängig voneinander als verschiedene Zündphasen. Die Anordnung ist in Fig. 11 dargestellt. Der Abstand zwischen aneinander grenzenden Gruppierungen ist zum Zwecke der Klarheit übertrieben dargestellt.

2.1.1.6 Zusammenfassung der Düsengruppierung

[0114] Tabelle 2 ist eine Zusammenfassung der Düsengruppierungen in einem Druckkopf.

Tabelle 2. Düsengruppierungen für einen einzelnen 4-Zoll-Druckkopf

Name der Gruppierung	**Zusammensetzung**	**Nachbildungs-verhältnis**	**Düsenzahl**
Düse 22	Basiseinheit	1:1	1
Sockel 23	Düsen pro Sockel	10:1	10
Chromapod 24	Sockel pro CMY-Chromapod	3:1	30
Sockelgruppe 25	Chromapods pro Sockelgruppe	5:1	150
Phasengruppe 26	Sockelgruppen pro Phasengruppe	2:1	300
Zündgruppe 27	Phasengruppen pro Zündgruppe	2:1	600
Segment 21	Zündgruppen pro Segment	4:1	2.400
4-Zoll-Druckkopf 2	Segmente pro 4-Zoll-Druckkopf	8:1	19.200

2.2 LADE- UND DRUCKZYKLEN

[0115] Ein einzelner 4-Zoll-Druckkopf **2** enthält eine Gesamtzahl von 19.200 Düsen **22**. Zu einem Druckzyklus gehört das Zünden von bis zu allen dieser Düsen, in Abhängigkeit der zu druckenden Informationen. Zu einem Ladezyklus gehört das Beladen des Druckkopfes mit den während des nachfolgenden Druckzyklus zu druckenden Informationen.

[0116] Jede Düse **22** weist ein zugehöriges NozzleEnable-Bit auf, das bestimmt, ob die Düse während des Druckzyklus zünden wird oder nicht. Die NozzleEnable-Bits (eins pro Düse) werden über einen Satz von Schieberegistern geladen.

[0117] Logisch gibt es drei Schieberegister pro Segment (eins pro Farbe), jeweils 800 lang. Da die Bits in das Schieberegister für eine gegebene Farbe geschoben werden, werden sie zu den unteren und oberen Düsen bei wechselnden Impulsen geleitet. Intern besteht jedes 800-tiefe Schieberegister aus zwei 400-tiefen Schieberegistern: einem für die oberen Düsen und einem für die unteren Düsen. Abwechselnde Bits werden in die abwechselnden internen Register geschoben. Sofern jedoch die externe Schnittstelle betroffen ist, existiert ein einzelnes 800-tiefes Schieberegister.

[0118] Sobald alle Schieberegister vollständig geladen sind (800 Ladeimpulse) werden sämtliche Bits parallel zu den entsprechenden NozzleEnable-Bits übertragen. Dies gleicht einem einzelnen Paralleltransfer von 19.200 Bits. Sobald die Übertragung stattgefunden hat, kann der Druckzyklus beginnen. Der Druckzyklus und der Ladezyklus können gleichzeitig auftreten, sofern das parallele Laden aller NozzleEnable-Bits am Ende des Druckzyklus passiert.

2.2.1 Ladezyklus

[0119] Der Ladezyklus betrifft das Laden der Schieberegister des Druckkopfes mit den NozzleEnable-Bits des nächsten Druckzyklus.

[0120] Jedes Segment **21** weist 3 Eingänge auf, die direkt mit den cyan, magenta und gelben Schieberegistern in Beziehung stehen. Diese Eingänge werden CDataln, MDataln und YDataln genannt. Da 8 Segmente vorhanden sind, gibt es eine Gesamtzahl von 24 Farbeingangsleitungen pro 4-Zoll-Druckkopf. Ein einzelner Impuls auf der SRClock-Leitung (die sich alle 8 Segmente teilen) überträgt die 24 Bits in die entsprechenden Schieberegister. Abwechselnde Impulse übertragen Bits zu den unteren bzw. oberen Düsen. Da es 19.200 Düsen gibt, ist eine Gesamtzahl von 800 Impulsen für den Transfer erforderlich. Sobald alle 19.200 Bits übertragen wurden, sorgt ein einzelner Impuls auf der gemeinsamen PTransfer-Leitung für den parallelen Transfer von Daten aus den Schieberegistern zu den entsprechenden NozzleEnable-Bits.

[0121] Der parallele Transfer über einen Impuls auf PTransfer muss stattfinden, nachdem der Druckzyklus abgeschlossen ist. Andernfalls werden die NozzleEnable-Bits für die zu druckende Zeile falsch.

[0122] Da alle 8 Segmente **21** mit einem einzelnen SRClock-Impuls geladen werden, muss jeder Druckvorgang die Daten in der korrekten Reihenfolge für den Druckkopf erzeugen. Als ein Beispiel überträgt der erste SRClock-Impuls die CMY-Bits für den Punkt 0, 800, 1600, 2400, 3200, 4000, 4800 und 5600 des nächsten Druckzyklus. Der zweite SRClock-Impuls überträgt die CMY-Bits für den Punkt 1, 801, 1601, 2401, 3201, 4001, 4801 und 5601 des nächsten Druckzyklus. Nach 800 SRClock-Impulsen kann der PTransfer-Impuls gegeben werden.

[0123] Es ist wichtig anzumerken, dass die ungeraden und geraden CMY-Ausgaben, obwohl sie während desselben Druckzyklus gedruckt werden, nicht auf derselben physikalischen Ausgabezeile erscheinen. Die physikalische Trennung von ungeraden und geraden Düsen innerhalb des Druckkopfes sowie die Trennung zwischen den Düsen von unterschiedlichen Farben gewährleistet, dass sie Punkte auf unterschiedlichen Zeilen der Seite erzeugen. Dieser relative Unterschied muss berücksichtigt werden, wenn man die Daten in den Druckkopf lädt. Der tatsächliche Unterschied bei den Zeilen hängt von den Eigenschaften des Inkjet-Mechanismus ab, der in dem Druckkopf verwendet wird. Die Unterschiede können durch die Variablen D_1 und D_2 definiert werden, wobei D_1 der Abstand zwischen Düsen verschiedener Farben ist und D_2 der Abstand zwischen Düsen derselben Farbe ist. Tabelle 3 zeigt die auf das Segment n eines Druckkopfs bei den ersten 4 Impulsen übertragenen Punkte.

Tabelle 3. Reihenfolge von auf einen 4-Zoll-Druckkopf übertragenen Punkten

Impuls	Punkt	Gelbe Zeile	Magenta Zeile	Cyan Zeile
1	800S[a]	N	N+D_1[b]	N+2D_1
2	800S+1	N+D_2[c]	N+D_1+D_2	N+2D_1+D_2
3	800S+2	N	N+D_1	N+2D_1
4	800S+3	N+D_2	N+D_1+D_2	N+2D_1+2D_2

a. S = Segmentnummer (0-7)
b. D_1 = Anzahl von Zeilen zwischen den Düsen einer Farbe und der nächsten (wahrscheinlich = 4-8)
c. D_2 = Anzahl von Zeilen zwischen zwei Reihen von Düsen derselben Farbe (wahrscheinlich = 1)

[0124] Und so weiter für alle 800 Impulse.

[0125] Daten können in den Druckkopf mit einer maximalen Taktrate von 200 MHz getaktet werden, wobei die gesamten Daten für die nächste Zeile in 40 µs geladen werden.

2.2.2 Druckzyklus

[0126] Ein 4-Zoll-Druckkopf **2** enthält 19.200 Düsen **22**. Sie alle gleichzeitig zu zünden würde zuviel Strom verbrauchen und wäre problematisch hinsichtlich Tintenwiederauffüllung und Düsenbeeinflussung. Folglich werden zwei Zündmodi definiert: ein Druckmodus mit geringer Geschwindigkeit und ein Druckmodus mit hoher Geschwindigkeit:

- In dem Druckmodus mit geringer Geschwindigkeit gibt es 200 Phasen, wobei jede Phase 96 Düsen zündet. Dies entspricht 12 Düsen pro Segment oder 3 pro Zündgruppe.
- Im Druckmodus mit hoher Geschwindigkeit gibt es 100 Phasen, wobei jede Phase 192 Düsen zündet. Dieses entspricht 24 Düsen pro Segment oder 6 pro Zündgruppe.

[0127] Die zu zündenden Düsen in einem vorgegebenen Zündimpuls werden bestimmt durch

- 3 Bits ChromapodSelect (Auswählen von 1 aus 5 Chromapods **24** aus einer Zündgruppe **27**)
- 4 Bits NozzleSelect (Auswählen von 1 von 10 Düsen **22** aus einem Sockel **23**)
- 2 Bits von PodgroupEnable-Leitungen (Auswählen von 0, 1 oder 2 Sockelgruppen **25**, um zu zünden).

[0128] Wenn eine der PodgroupEnable-Leitungen gesetzt ist, werden nur die 4 Düsen der bestimmten Sockelgruppe zünden, die durch ChromapodSelect und NozzleSelect bestimmt ist. Wenn beide PodgroupEnable-Leitungen gesetzt sind, werden beide Sockelgruppen ihre Düsen zünden. Für den Modus mit niedriger Geschwindigkeit sind zwei Zündimpulse erforderlich, wobei PodgroupEnable = 10 bzw. 01. Für den Modus mit hoher Geschwindigkeit ist lediglich ein Zündimpuls erforderlich, wobei PodgroupEnable = 11 gilt.

[0129] Die Dauer der Zündimpulse ist durch die AEnable- und BEnable-Leitungen gegeben, die die PhasegroupA- und PhasegroupB-Düsen von allen Zündgruppen entsprechend zünden. Die typische Dauer eines Zündimpulses ist 1,3–1,8 µs. Die Dauer eines Impulses hängt von der Viskosität der Tinte (abhängig von der Temperatur und den Tinteneigenschaften) und der Leistungsmenge ab, die dem Druckkopf zur Verfügung steht. Siehe auch Abschnitt 2.3 zu Einzelheiten über die Rückmeldung von dem Druckkopf, um eine Temperaturveränderung auszugleichen.

[0130] AEnable und BEnable sind getrennte Leitungen, derart, dass die Zündimpulse überlappen können. Deshalb bestehen die 200 Phasen eines Druckzyklus mit niedriger Geschwindigkeit aus 100 A-Phasen und 100 B-Phasen, woraus sich effektiv 100 Sätze von Phase A und Phase B ergeben. In ähnlicher Weise bestehen die 100 Phasen eines Druckzyklus mit hoher Geschwindigkeit aus 50 A-Phasen und 50 B-Phasen, was effektiv 50 Phasen aus Phase A und Phase B ergibt.

[0131] Fig. 12 zeigt die AEnable- und BEnable-Leitungen während eines typischen Druckzyklus. In einem Druck mit hoher Geschwindigkeit gibt es 50 2-µs-Zyklen, während in einem Druck mit niedriger Geschwindigkeit 100 2-µs-Zyklen vorhanden sind.

[0132] Für den Druckmodus mit hoher Geschwindigkeit ist die Zündreihenfolge:

- ChromapodSelect 0, NozzleSelect 0, PodgroupEnable 11 (Phasen A und B)
- ChromapodSelect 1, NozzleSelect 0, PodgroupEnable 11 (Phasen A und B)
- ChromapodSelect 2, NozzleSelect 0, PodgroupEnable 11 (Phasen A und B)
- ChromapodSelect 3, NozzleSelect 0, PodgroupEnable 11 (Phasen A und B)
- ChromapodSelect 4, NozzleSelect 0, PodgroupEnable 11 (Phasen A und B)
- ChromapodSelect 0, NozzleSelect 1, PodgroupEnable 11 (Phasen A und B)
- ...
- ChromapodSelect 3, NozzleSelect 9, PodgroupEnable 11 (Phasen A und B)
- ChromapodSelect 4, NozzleSelect 9, PodgroupEnable 11 (Phasen A und B)

[0133] Für den Druckmodus mit niedriger Geschwindigkeit ist die Zündreihenfolge ähnlich. Für jede Phase des Modus mit hoher Geschwindigkeit, wo PodgroupEnable 11 war, werden zwei Phasen PodgroupEnable = 01 und 10 wie folgt ersetzt:

- ChromapodSelect 0, NozzleSelect 0, PodgroupEnable 01 (Phasen A und B)
- ChromapodSelect 0, NozzleSelect 0, PodgroupEnable 10 (Phasen A und B)
- ChromapodSelect 1, NozzleSelect 0, PodgroupEnable 01 (Phasen A und B)
- ChromapodSelect 1, NozzleSelect 0, PodgroupEnable 10 (Phasen A und B)
- ...
- ChromapodSelect 3, NozzleSelect 9, PodgroupEnable 01 (Phasen A und B)
- ChromapodSelect 3, NozzleSelect 9, PodgroupEnable 10 (Phasen A und B)
- ChromapodSelect 4, NozzleSelect 9, PodgroupEnable 01 (Phasen A und B)
- ChromapodSelect 4, NozzleSelect 9, PodgroupEnable 10 (Phasen A und B)

[0134] Wenn eine Düse **22** zündet, braucht es in etwa 100 µs, damit sie wieder befüllt wird. Die Düse **22** kann nicht gezündet werden, bevor diese Wiederbefüllungszeit verstrichen ist. Dies beschränkt die schnellste Druckgeschwindigkeit auf 100 µs pro Zeile. Im Druckmodus mit hoher Geschwindigkeit beträgt die Zeit, um eine Zeile zu drucken, 100 µs, so dass die Zeit zwischen dem Zünden einer Düse von einer Zeile zur nächsten mit der Wiederbefüllungszeit übereinstimmt, wodurch der Druckmodus mit hoher Geschwindigkeit akzeptabel wird. Der Druckmodus mit niedriger Geschwindigkeit ist langsamer als dieser und ist deshalb ebenfalls akzeptabel.

[0135] Das Zünden einer Düse **22** verursacht ebenfalls akustische Störungen für einen begrenzten Zeitraum innerhalb des gemeinsamen Tintenreservoirs dieses Sockels **23** der Düse. Die Störungen können das Zünden einer anderen Düse innerhalb des gleichen Sockels **23** beeinflussen. Folglich sollte das Zünden von Düsen innerhalb eines Sockels jeweils voneinander soweit wie möglich versetzt sein. Wir zünden deshalb drei Düsen von einem Chromapod **24** (eine Düse **22** pro Farbe) und bewegen uns anschließend weiter zum nächsten Chromapod **24** innerhalb der Sockelgruppe **25**.

- Bei dem Druckmodus mit niedriger Geschwindigkeit werden die Sockelgruppen **25** getrennt voneinander gezündet. Deshalb müssen die 5 Chromapods **24** innerhalb von beiden Sockelgruppen alle gezündet werden, bevor der erste Chromapod erneut zündet, was insgesamt 10 × 2-µs-Zyklen ergibt. Folglich wird jeder Sockel **23** einmal pro 20 µs gezündet.
- Bei dem Druckmodus mit hoher Geschwindigkeit werden die Sockelgruppen **25** zusammen gezündet. Deshalb müssen die 5 Chromapods **24** innerhalb einer einzelnen Sockelgruppe alle gezündet werden, bevor der erste Chromapod erneut zündet, woraus sich insgesamt 5 × 2-µs-Zyklen ergeben. Folglich wird jeder Sockel **23** einmal pro 10 µs gezündet.

[0136] Da der Tintenkanal 300 µm lang ist und die Schallgeschwindigkeit in der Tinte etwa 1500 m/s beträgt, liegt die Resonanzfrequenz des Tintenkanals bei 2,6 MHz, damit ermöglicht der Modus mit niedriger Geschwindigkeit, dass der akustische Impuls über 50 Resonanzzyklen hinweg abklingt, und der Modus mit hoher Geschwindigkeit ermöglicht 25 Resonanzzyklen. Damit ist jede akustische Interferenz in beiden Fällen minimal.

2.2.3 Abtastzeitpunkte

[0137] Als ein Beispiel betrachte man den Zeitverlauf des Druckens eines 4" × 6" Fotos in 2 Sekunden, wie es von einer Printcam erforderlich ist. Um ein Foto in 2 Sekunden zu drucken, muss der 4-Zoll-Druckkopf 9600 Zeilen (6 × 1600) drucken. Rundet man dies auf 10.000 Zeilen in 2 Sekunden auf, ergibt sich eine Zeilenzeit von 200 µs. Ein einzelner Druckzyklus und ein einzelner Ladezyklus müssen beide innerhalb dieser Zeit abgeschlossen werden. Zusätzlich muss ein physikalischer Prozess, der außerhalb des Druckkopfs verläuft, das

Papier um einen entsprechenden Betrag bewegen.

[0138] Aus Drucksicht erlaubt der Druckmodus mit niedriger Geschwindigkeit, dass ein 4-Zoll-Druckkopf eine vollständige Zeile in 200 µs druckt. In dem Druckmodus mit niedriger Geschwindigkeit zünden 96 Düsen **22** pro Zündimpuls, wodurch das Drucken einer gesamten Zeile innerhalb der bestimmten Zeit ermöglicht wird.

[0139] Die 800 SRClock-Impulse an den Druckkopf **2** (jeder Taktimpuls überträgt 24 Bits) müssen ebenfalls innerhalb der 200-µs-Zeilenzeit stattfinden. Die Länge eines SRClock-Impulses kann 200 µs/800 = 250 ns nicht übersteigen, womit angegeben wird, dass der Druckkopf mit 4 MHz getaktet werden muss. Darüber hinaus darf die Durchschnittszeit, um jeden Bitwert (für jeden der 19.200 Düsen) zu berechnen, 200 µs/19.200 = 10 ns nicht überschreiten. Dies erfordert einen Punktgenerator, der bei einer der folgenden Geschwindigkeiten läuft:

- 100 MHz, die 1 Bit (Punkt) pro Zyklus erzeugen
- 50 MHz, die 2 Bits (Punkte) pro Zyklus erzeugen
- 25 MHz, die 4 Bits (Punkte) pro Zyklus erzeugen.

2.3 RÜCKMELDUNG VON DEM DRUCKKOPF

[0140] Der Druckkopf **2** erzeugt mehrere Rückmeldeleitungen (aufgesammelt von den 8 Segmenten). Die Rückmeldeleitungen werden verwendet, um das Timing der Zündimpulse anzupassen. Obwohl jedes Segment **21** dieselbe Rückmeldung erzeugt, teilen sich die Rückmeldung von allen Segmenten dieselben drei Zustands-Busleitungen. Folglich kann lediglich ein Segment **21** zu einem Zeitpunkt eine Rückmeldung geben.

[0141] Ein Impuls auf der SenseSegSelect-Leitung mit logisch UND verknüpft mit Daten auf Cyan schaltet die Abfrageleitungen für dieses Segment ein. Die Rückmeldungs-Abfrageleitungen kommen von dem ausgewählten Segment bis zum nächsten SenseSegSelect-Impuls. Die Rückmelde-Abfrageleitungen sind wie folgt:

- Tsense informiert die Steuereinheit, wie heiß der Druckkopf ist. Dies ermöglicht der Steuereinheit, das Timing der Zündimpulse anzupassen, da die Temperatur die Viskosität der Tinte beeinflusst.
- Vsense informiert die Steuereinheit, wie hoch die verfügbare Spannung auf das Stellglied ist. Dies ermöglicht der Steuereinheit, eine leere Batterie oder eine Hochspannungsquelle durch Anpassen der Impulsbreite auszugleichen.
- Rsense informiert die Steuereinheit über den spezifischen Widerstand (Ohm pro Fläche) der Stellgliedheizeinrichtung. Dies ermöglicht, dass die Steuereinheit die Impulsbreiten anpasst, um einen konstanten Energiewert aufrecht zu erhalten, unabhängig vom spezifischen Widerstand der Heizeinrichtung.
- Wsense informiert die Steuereinheit über die Breite des kritischen Bestandteils der Heizeinrichtung, die bis zu ± 5 % aufgrund von Lithographie- und Ätzabweichungen variieren kann. Dadurch wird ermöglicht, dass die Steuereinheit die Impulsbreite in geeigneter Weise anpassen kann.

2.4 SPEZIALZYKLEN

2.4.1 Vorheizzyklus

[0142] Der Druckprozess hat eine starke Tendenz dazu, bei der Gleichgewichtstemperatur zu verbleiben. Um sicherzustellen, dass der erste Abschnitt des gedruckten Fotos eine beständige Punktgröße aufweist, muss die Gleichgewichtstemperatur erreicht sein, bevor irgendwelche Punkte gedruckt werden. Dies wird durch einen Vorheizzyklus erreicht.

[0143] Der Vorheizzyklus umfasst einen einzelnen Ladezyklus für alle Düsen mit einer Sekunde (d. h. Einstellen aller Düsen, um zu Zünden), und eine Anzahl von kurzen Zündimpulsen an jede Düse. Die Dauer des Impulses darf nicht ausreichend sein, um die Tropfen herauszulassen, aber genug, um die Tinte zu erhitzen. Insgesamt sind etwa 200 Impulse für jede Düse erforderlich, die zyklisch in der gleichen Reihenfolge wie ein standardmäßiger Druckzyklus durchgeführt werden.

[0144] Rückmeldung während des Vorheizmodus wird durch Tsense bereitgestellt und läuft weiter, bis die Gleichgewichtstemperatur erreicht ist (etwa 30° C oberhalb der Umgebungstemperatur). Die Dauer des Vorheizmodus beträgt etwa 50 ms und hängt von der Tintenzusammensetzung ab.

[0145] Das Vorheizen wird vor jedem Druckjob durchgeführt. Dies beeinflusst die Druckleistung nicht, da es durchgeführt wird, während die Seitendaten zu dem Drucker übertragen werden.

2.4.2 Reinigungszyklus

[0146] Um die Wahrscheinlichkeit zu reduzieren, dass die Düsen verstopft werden, kann ein Reinigungszyklus vor jedem Druckzyklus durchgeführt werden. Jede Düse wird eine Anzahl von Malen in einen absorbierenden Schwamm gezündet.

[0147] Der Reinigungszyklus umfasst einen einzelnen Ladezyklus für alle Düsen mit 1 s (d. h. das Einstellen aller Düsen, um zu zünden) und eine Anzahl von Zündimpulsen an jede Düse. Die Düsen werden über dieselbe Düsenzündsequenz gereinigt wie ein standardmäßiger Druckzyklus. Die Anzahl von Malen, die jede Düse **22** gezündet wird, hängt von der Tintenzusammensetzung und der Zeit ab, die der Drucker nicht in Benutzung war; so wie das Vorheizen hat der Reinigungszyklus keine Auswirkung auf die Druckerleistung.

2.5 ZUSAMMENFASSUNG DER DRUCKKOPF-SCHNITTSTELLE

[0148] Ein einzelner 4-Zoll-Druckkopf **2** hat die folgenden Verbindungen:

Tabelle 4. Vier-Zoll-Druckkopfverbindungen

Name	**#Pins**	**Beschreibung**
ChromapodSelect	3	Auswahl, welcher Chromapod zünden soll (0-4)
NozzleSelect	4	Auswahl, welche Düse von dem Sockel zünden wird (0-9)
PodgroupEnable	2	Einschalten der Sockelgruppen, um zu zünden (Auswahl aus: 01, 10, 11)
AEnable	1	Zündimpuls für Phasengruppe A
BEnable	1	Zündimpuls für Phasengruppe B
CDataIn[0-7]	8	Cyan-Eingabe für Cyan-Schieberegister der Segmente 0-7
MDataIn[0-7]	8	Magenta-Eingabe an Magenta-Schieberegister der Segmente 0-7
YDataIn[0-7]	8	Gelb-Eingabe an gelbes Schieberegister der Segmente 0-7
SRClock	1	Ein Impuls auf SRClock (ShiftRegisterClock) lädt die aktuellen Werte aus CDataIn [0-7], MDataIn [0-7] und YDataIn [0-7] in die 24 Schieberegister.
PTransfer	1	Paralleler Transfer der Daten aus den Schieberegistern in die internen NozzleEnable-Bits (eines pro Düse).
SenseSegSelect	1	Ein Impuls auf SenseSegSelect UND-verknüpft mit Daten auf CDataIn[n] wählt die Abfrageleitungen für das Segment n aus.
Tsense	1	Temperaturabfrage

Vsense	1	Spannungsabfrage
Rsense	1	Abfrage des spezifischen Widerstands
Wsense	1	Breitenabfrage
Logic GND	1	Logisch Erde
Logic PWR	1	Logisch Strom
V -	Bus-Schiene	Stellglied Erde
V +		Stellglied Strom
GESAMT	44	

[0149] Intern im 4-Zoll-Druckkopf weist jedes Segment die folgenden Verbindungen zu den Bond-Verbindungspunkten auf:

Tabelle 5. Vier-Zoll-Druckkopf interne Segmentverbindungen

Name	**#Pins**	**Beschreibung**
ChromapodSelect	3	Auswahl, welcher Chromapod zünden soll (0-4)
NozzleSelect	4	Auswahl, welche Düse von dem Sockel zünden wird (0-9)
PodgroupEnalble	2	Einschalten der Sockelgruppen, um zu zünden (Auswahl aus: 01, 10, 11)
AEnable	1	Zündimpuls für Phasengruppe A
BEnable	1	Zündimpuls für Phasengruppe B
CDataIn	1	Cyan-Eingabe an Cyan-Schieberegister
MDataIn	1	Magenta-Eingabe an Magenta-Schieberegister
YDataIn	1	Gelb-Eingabe an gelbes Schieberegister
SRClock	1	Ein Impuls auf SRClock (ShiftRegisterClock) lädt die aktuellen Werte aus CDataIn, MDataIn und YDataIn in die 3 Schieberegister.
PTransfer	1	Paralleler Transfer der Daten aus den Schieberegistern in die internen NozzleEnable-Bits (eines pro Düse).
SenseSegSelect	1	Ein Impuls auf SenseSegSelect UND verknüpft mit Daten auf CDataIn wählt die Abfrageleitungen für dieses Segment aus.
Tsense	1	Temperaturabfrage
Vsense	1	Spannungsabfrage
Rsense	1	Abfrage des spezifischen Widerstands
Wsense	1	Breitenabfrage

Logic GND	1	Logisch Erde
Logic PWR	1	Logisch Strom
V -	21	Stellglied Erde
V +	21	Stellglied Strom
GESAMT	65	(65 x 8 Segmente = 520 für alle Segmente)

3. BILDVERARBEITUNGSKETTEN

[0150] Die vorstehenden Abschnitte haben sich lediglich mit einem abstrakten Überblick der PCP-Funktionalität beschäftigt – der des Zuordnens von CFA-Bildern zu einer Vielzahl von Ausgabedruckformaten. In der Tat gehören eine Vielzahl von Schritten zum Aufnehmen eines Bildes von dem Bildsensor und zum Herstellen eines qualitativ hochwertigen Ausgabedrucks. Man kann den abstrakten Prozess in zwei Bildverarbeitungsketten herunter brechen, jede mit einer Anzahl von Schritten:

- Bilderfassungskette
- Druckkette

[0151] Die Bilderfassungskette betrifft das Erfassen des Bildes von dem Bildsensor und dessen lokaler Speicherung innerhalb der Printcam. Die Druckkette betrifft das Herausnehmen des gespeicherten Bildes und dessen Druck. Diese beiden Ketten lassen sich der grundlegenden Printcam-Funktionalität wie folgt zuordnen:

- Aufnehmen & Drucken = Bilderfassungskette gefolgt von der Druckkette
- Neuer Ausdruck/Reprint = Druckkette

[0152] Zum Beispiel kann ein Benutzer ein Miniaturbild (Aufnehmen & Drucken) ausdrucken und wenn er mit den Ergebnissen zufrieden ist, mehrere Standardkopien ausdrucken (neu Ausdrucken/Reprint).

[0153] Dieses Kapitel beschreibt eine implementierungsunabhängige Bildverarbeitungskette, die den Qualitätsanforderungen der Printcam genügt. Bei diesem Schritt berücksichtigen wir nicht genau, wie die Verarbeitung hinsichtlich der Hardware durchgeführt wird, sondern eher, was durchgeführt werden muss. Diese Funktionen müssen den unterschiedlichen Einheiten innerhalb des PCP zugeordnet werden.

[0154] Unabhängig von der PCP-Implementierung gibt es eine Reihe von Randbedingungen:

- Das Eingabebild ist ein CFA-basiertes RGB-Bild mit gleichmäßigem Tonverlauf.
- Das Ausgabebild für einen Memjet-Druckkopf (zweiwertige Punkte bei 1600 dpi) ist im CMY-Farbraum und weist immer dieselbe Ausgabebreite auf (4 Zoll breit).

3.0.1 Unterstützte Druckformate

[0155] Der PCP **3** unterstützt eine Vielzahl von Ausgabedruckformaten, wie in Fig. 6 dargestellt. In sämtlichen Fällen beträgt die Breite des Bildes 4 Zoll (stimmt mit der Druckkopf-Breite überein). Lediglich die Länge des Ausdrucks verändert sich.

Tabelle 6. Unterstützte Bildformate

Formatname	**Bildseiten-verhältnis**	**Ausgabegröße (Zoll)**	**Ausgabeauflösung (bei 1600 dpi)**	**Drehung**
Standardformat 30	2 : 3	4" x 6"	6400 x 9600	90
Hochformat 31	2 : 3	4" x 6"	6400 x 9600	90
Querformat 33	4 : 6	4" x 12"	6400 x 19200	90
Miniaturformat 32	2 : 3	4" x 2,67"	6400 x 4267	0

[0156] Der Bildsensor stellt keine Ausrichtungs- bzw. Formatinformationen bereit. Alle Eingabebilder werden mit der gleichen Auflösung (1500 × 1000) erfasst und sind möglicherweise vor dem Ausdruck um 90 Grad zu rotieren. Fig. 13 veranschaulicht die Zuordnung zwischen dem erfassten CFA-Bild und den verschiedenen unterstützten Druckformaten. Es ist anzumerken, dass, obwohl das Bild um 90 Grad gegen den Uhrzeigersinn dargestellt ist, das Bild sowohl im Uhrzeigersinn als auch gegen den Uhrzeigersinn gedreht werden kann.

3.1 BILDERFASSUNGSKETTE

[0157] Die Bilderfassungskette ist dafür verantwortlich, ein Bild aus dem Bildsensor herauszunehmen und es lokal innerhalb der Printcam zu speichern. Die Bilderfassungskette bezieht eine Anzahl von Prozessen mit ein, die nur während der Bilderfassung durchgeführt werden müssen. Die Bilderfassungskette ist in Fig. 14 dargestellt, wobei die nachfolgenden Abschnitte die Teilkomponenten im Detail beschreiben.

3.1.1 Bildsensor **1**

[0158] Das Eingabebild kommt von einem Bildsensor **1**. Obwohl eine Vielzahl von Bildsensoren verfügbar ist, betrachten wir lediglich das Bayer CFA (color filter array; Farbfiltermatrix)-Bild. Das Bayer CFA-Bild hat eine Anzahl von Attributen, die hier definiert werden.

[0159] Es wird angenommen, dass das durch den CMOS-Sensor **1** (über eine Aufnahmelinse) erfasste Bild ausreichend gefiltert wurde, um jegliche Aliasing-Artefakte zu entfernen. Der Sensor selbst weist ein Bildseitenverhältnis von 3:2 mit einer Auflösung von 1500 × 1000 Rasterpunkten auf. Die wahrscheinlichste Pixelanordnung ist das Bayer CFA (color filter array)-Bild wobei jeder 2 × 2 Pixel-Block in einem 2G-Mosaik, wie in Fig. 15 dargestellt, angeordnet ist:
Jede Rasterung aus R, G oder B (entsprechend rot, grün bzw. blau) mit gleichmäßigem Tonverlauf besteht aus 10 Bits. Es ist anzumerken, dass jedes Pixel des Mosaiks nur Informationen über eine der Farben R, G oder B enthält. Schätzungen der fehlenden Farbinformationen müssen gemacht werden, bevor das Bild ausgedruckt werden kann.

[0160] Es gilt, dass das CFA-Bild eine geeignete Unterdrückung der feststehenden Störstruktur (fixed pattern noise; FPN) durchführt.

3.1.2 RGB-Linearisierer **40**

- Es ist unwahrscheinlich, dass der Bildsensor **40** eine vollständig lineare Antwort aufweist. Deshalb müssen die 10-Bit RGB-Rasterpunkte aus der Farbfiltermatrix als nicht linear betrachtet werden. Diese nicht linearen Abtastpunkte werden in 8-Bit-lineare Abtastpunkte mittels Nachschlagetabellen (eine Tabelle pro Farbe) übersetzt.

[0161] Pixel aus den Farbfiltermatrix-Zeilen 0, 2, 4 usw. indizieren in die R- und G-Tabellen, während Pixel aus den Farbfiltermatrix-Zeilen 1, 3, 5 usw. in die G- und B-Tabellen indizieren. Dies ist völlig unabhängig von der Orientierung der Kamera. Der Prozess ist in Fig. 16 dargestellt. Die gesamte erforderliche Speichermenge für jede Nachschlagetabelle beträgt 2^{10} × 8 -Bits. Die 3 Nachschlagetabellen **45** erfordern daher eine Gesamtgröße von 3 KBytes (3 × 2^{10} Bytes).

3.1.3 RGB-Planarisieren **41**

[0162] Die aus der Farbfiltermatrix (CFA) erhaltenen Pixel weisen Farbebenen auf, die aufgrund des Wesens des Bayer-Mosaiks aus Pixeln verschachtelt sind. Damit meinen wir, dass auf geradzahligen horizontalen Zeilen auf ein rotes Pixel ein grünes Pixel und anschließend noch ein rotes Pixel folgt – die unterschiedlichen Farbebenen sind miteinander verschachtelt. In einigen Bildverarbeitungssystemen ist ein verschachteltes Format in hohem Maße nützlich. In dem Printcam-Verarbeitungssystem sind die Algorithmen jedoch effizienter, wenn man auf planarem RGB arbeitet.

[0163] Ein planarisiertes Bild ist eines, das in seine Bestandteilfarben aufgetrennt wurde. In dem Fall des Farbbildmatrix-RGB-Bildes gibt es 3 unterschiedliche Bilder: ein Bild, das nur die roten Pixel enthält, ein Bild, das nur die blauen Pixel enthält, und ein Bild, das nur die grünen Pixel enthält. Man bemerke, dass jede Ebene nur die Pixel der Farbe darstellt, die tatsächlich abgetastet wurden. Kein erneutes Abtasten bzw. Resampling wird während des Planarisierungsprozesses durchgeführt. Als Ergebnis sind die R-, G- und B-Ebenen nicht miteinander registriert, und die G-Ebene ist doppelt so groß wie die R- bzw. B-Ebenen. Der Prozess ist in

Fig. 17 abgebildet.

[0164] Der tatsächliche Prozess ist relativ einfach – je nach Farbe der eingelesenen Pixel werden die Ausgabepixel zu der nächsten Position in dem Bild der entsprechenden Farbebene gesendet (deshalb in der gleichen Orientierung wie die Farbfiltermatrix).

[0165] Die roten **45** und blauen **47** planaren Bilder betragen exakt ein Viertel der Größe des ursprünglichen Farbfiltermatrix-Bildes. Sie weisen exakt die Hälfte der Auflösung in jeder Dimension aus. Die roten und blauen Bilder sind deshalb jeweils 750 × 500 Pixel groß, wobei das rote Bild implizit von dem blauen Bild um ein Pixel im Farbfiltermatrix-Raum (1500 × 1000) sowohl in x- als auch in y-Richtung versetzt ist.

[0166] Obwohl das grüne planare Bild **46** die Hälfte der Größe des ursprünglichen Farbfiltermatrix-Bildes beträgt, ist es nicht so geradlinig dargestellt wie die roten oder blauen Ebenen. Grund dafür ist das Schachbrettmuster von grün. Auf einer Zeile ist jedes ungerade Pixel grün und auf der nächsten Zeile ist jedes gerade Pixel grün. Deshalb stellen abwechselnde Zeilen der grünen Ebene ungerade und gerade Pixel innerhalb des Farbfiltermatrix-Bildes dar. Dadurch beträgt das grüne planare Bild 750 × 1000 Pixel. Dies hat Auswirkungen auf den Resampling-Prozess (siehe erneutes Abtasten/Resample **64** weiter unten).

3.1.4 Gespeichertes Bild **42**

[0167] Jede Farbebene des linearisierten RGB-Bildes wird zur Zwischenspeicherung in den Speicher geschrieben. Der Speicher sollte Flash-Speicher **11** sein, so dass das Bild beibehalten wird, nachdem der Strom abgeschaltet worden ist.

[0168] Die gesamte Menge an erforderlichem Speicher für das planarisierte lineare RGB-Bild beträgt 1.500.000 Bytes (etwa 1,5 MB), die wie folgt verteilt sind:

- R: 750 × 500 = 375.000 Bytes
- B: 750 × 500 = 375.000 Bytes
- G: 750 × 1000 = 750.000 Bytes

3.2 DRUCKKETTE

[0169] Die Druckkette betrifft das Herausnehmen eines existierenden Bildes aus dem Speicher **42** und das Ausdrucken des Bildes zu einem Memjet-Drucker **2**. Typischerweise wird ein Bild gedruckt, sobald es erfasst worden ist, obwohl es auch erneut ausgedruckt werden kann (d. h. ohne es erneut zu erfassen).

[0170] Es existiert eine Anzahl von Schritten, die in der Bildverarbeitungskette erforderlich sind, um qualitativ hochwertige Ausdrucke aus mit Farbfiltermatrix erfassten Bildern zu erzeugen. Fig. 18 veranschaulicht die Druckkette. Die Kette ist in 3 Arbeitsauflösungen aufgeteilt. Die erste ist der ursprüngliche Bilderfassungsraum **50** (derselbe Raum wie die Farbfiltermatrix), die zweite ist eine Zwischenauflösung **51** (Zeilen von 1280 Pixeln mit gleichmäßigem Tonverlauf) und die finale Auflösung ist die Druckerauflösung **52** mit Zeilen aus 6400 zweiwertigen Punkten.

3.2.1 Eingabebild

[0171] Das Eingabebild ist ein linearisiertes RGB-Bild **42**, das in planarer Form gespeichert ist, wie es durch die Bilderfassungskette gespeichert wurde, die in Abschnitt 3.1.4 beschrieben ist.

3.2.2 Erfassen von Kenngrößen **60**

[0172] Einen Anzahl von Kenngrößen im Hinblick auf das gesamte Bild müssen erfasst werden, bevor Prozesse wie Weißabgleich und Bereichserweiterung durchgeführt werden können. Diese Kenngrößen brauchen nur einmal für alle Ausdrucke eines bestimmten erfassten Bildes **42** erfasst zu werden und können getrennt voneinander aus den roten, grünen und blauen planaren Bildern gesammelt werden.

3.2.2.1 Aufbau des Histogramms

[0173] Der erste Schritt ist, ein Histogramm für jeden 8-Bit-Wert der Farbebene aufzubauen. Jedes 1500 × 1000 Farbfiltermatrix-Bild enthält eine Gesamtzahl von:

- 375.000 roten Pixel (mindestens ein 19-Bit-Zähler erforderlich)

- 375.000 blauen Pixel (mindestens ein 19-Bit-Zähler erforderlich)
- 750.000 grünen Pixel (mindestens ein 20-Bit-Zähler erforderlich)

[0174] Deshalb ist eine einzelne 256 × 20 Bit-Tabelle erforderlich, um das Histogramm aufzunehmen.

[0175] Der Aufbauprozess des Histogramms ist einfach, wie durch den nachfolgenden Pseudocode veranschaulicht:

```
For I = 0 to 255
     Entry [I] = 0
EndFor
For Pixel = ImageStart to ImageEnd
     p = Image [Pixel]
     Entry[p] = Entry[p]+1
EndFor
```

3.2.2.2 Bestimmen der oberen und unteren Grenzwerte

[0176] Sobald das Histogramm für die Farbebene aufgebaut worden ist, kann es verwendet werden, um einen oberen und unteren Grenzwert zu bestimmen. Diese Grenzwerte können verwendet werden, um später den Weißabgleich und die Bereichserweiterung während des Druckprozesses zu automatisieren.

[0177] Indem die Grenzwerte auf der Anzahl von Pixeln aus dem Histogramm basiert werden, berücksichtigen wir, dass die n % dunkelsten Pixel entbehrlich und deshalb gleich sind. In der gleichen Weise erwarten wir, dass die n % hellsten Pixel entbehrlich und somit gleich sind. Es wird erwartet, dass der genaue Wert für n etwa bei 5% liegt, das hängt jedoch von den Farbfiltermatrix-Antworteigenschaften ab.

[0178] Der Prozess des Bestimmens der n % dunkelsten Werte ist unkompliziert. Er umfasst das schrittweise Durchgehen durch das Histogramm der Farbebene von dem Zähler von 0 an aufwärts (d. h. 0, 1, 2, 3 usw.), bis der Gesamtwert n % erreicht ist oder man sich weiter als ein eingestellter Betrag von 0 bewegt hat. Der höchste dieser Werte wird als unterer Grenzwert für die Farbebene betrachtet. Obwohl es einen Unterschied zwischen diesen dunkelsten Werten gibt, kann der Unterschied als vernachlässigbar für die Zwecke der Bereichserweiterung und des Farbabgleichs betrachtet werden.

[0179] Der Prozess des Bestimmens der n % hellsten Werte ist ähnlich. Er umfasst das schrittweise Durchlaufen durch das Histogramm der Farbebene von dem Zähler 255 an abwärts (d. h. 255, 254, 253 usw.), bis der Gesamtwert n % erreicht ist, oder man sich weiter als ein fester Betrag von 255 bewegt hat. Der niedrigste dieser Werte wird als oberer Grenzwert der Farbebene betrachtet. Obwohl es einen Unterschied zwischen diesen hellsten Werten gibt, kann der Unterschied für die Zwecke der Bereichserweiterung und des Farbabgleichs als vernachlässigbar betrachtet werden.

[0180] Der Grund für das Anhalten nach einem eingestellten Abstand von 0 oder 255 ist, zwei Arten von Bildern abzugleichen:

- wo der ursprüngliche dynamische Bereich gering ist oder
- wo kein weiß oder schwarz in einem Bild vorhanden ist.

[0181] In diesen beiden Fällen will man nicht betrachten, dass die gesamten n % an oberen und unteren Werten vernachlässigbar bzw. entbehrlich sind, da man mit einem niedrigen Bereich am Anfang beginnt. Wir können mit Sicherheit den hohen Grenzwert **73** und den niedrigen Grenzwert **72** einstellen, so dass sie außerhalb des Bereichs von Pixelwerten liegen, die tatsächlich gerastert wurden. Der genaue Abstand hängt von der Farbfiltermatrix ab, er wird jedoch aus zwei Konstanten bestehen.

[0182] Ein beispielhafter Farbbereich für eine Farbebene ist in Fig. 19 dargestellt. Es ist anzumerken, dass, obwohl der gesamte 0-250-Bereich für die Pixel einer Bildfarbebene möglich ist, dieses besondere Bild einen

kleineren Bereich aufweist. Es ist ebenfalls anzumerken, dass derselbe n%-Histogrammbereich **70**, **71** durch einen größeren Bereich am unteren Ende **70** als am oberen Ende **71** dargestellt ist. Dies liegt daran, da dass Histogramm mehr Pixel mit höheren Werten enger zusammen verglichen mit dem unteren Ende enthalten muss.

[0183] Die oberen und unteren Grenzwerte **73** bzw. **72** müssen für jede Farbebene individuell bestimmt werden. Diese Information wird verwendet, um Bereichsskalen- und Offsetfaktoren zu berechnen, die beim späteren Weißabgleich- und Bereichserweiterungsprozess verwendet werden.

[0184] Der nachfolgende Pseudocode veranschaulicht den Prozess des Bestimmens jedes der beiden Grenzwerte (um den unteren Grenzwert zu finden, ist StartPosition = 255 und Delta = 1. Um den hohen Grenzwert zu finden, ist Startposition = 0 und Delta = –1). Bei dem Pseudocode gilt, dass Threshold ein 8-Bit-Wert ist, der sich während der Addition durchschiebt.

```
Threshold = StartPosition
Total = 0
TotalDelta = 0
While (TotalDelta < MaxDelta) AND (Total < MaxPixels))
     Threshold = Threshold + Delta
     Total = Total + Entry[Threshold]
     TotalDelta = TotalDelta + 1
EndWhile
Return Threshold
```

3.2.3 Drehen eines Bildes **61**

[0185] Die Drehung des Bildes **61** ist ein optionaler Schritt sowohl beim Erfassungs- als auch beim Druck- und Reprint-Prozess.

[0186] Unterschiedliche Druckformate machen es erforderlich, dass das Bild entweder um 0 oder 90 Grad relativ zur Farbfiltermatrix-Orientierung gedreht wird, wie in Fig. 13 dargestellt. Der Drehbetrag bzw. -winkel hängt von dem aktuell ausgewählten Druckformat ab. Obwohl die Drehrichtung nicht wichtig ist (sie kann im Uhrzeigersinn oder gegen den Uhrzeigersinn sein, da die neue Ausrichtung lediglich die Druckkopf-Breite unterstützt), beeinflusst die Drehrichtung die relative Registrierung der drei Farbebenen. Tabelle 7 fasst die Drehung zusammen, die für jedes Druckformat aus der ursprünglichen Farbfiltermatrix-Orientierung erforderlich ist.

Tabelle 7. Drehungen aus der Farbfiltermatrix-Orientierung für Druckformate

Druckformat	Drehung
Standardformat 30	90
Hochformat 31	90
Querformat 33	90
Miniaturbildformat 32	0

[0187] Da man nur um 0 oder 90 Grad dreht, geht während des Drehprozesses keine Information verloren. Für eine Drehung von 0 Grad kann das Bild Zeile für Zeile gelesen werden und für eine Drehung um 90 Grad kann das Bild Spalte für Spalte gelesen werden. Die Registrierung der 3 Farbebenen muss die Drehrichtung

berücksichtigen.

3.2.4 Weißabgleich **62** und Bereichserweiterung **63**

[0188] Ein Foto wird selten unter idealen Lichtverhältnissen aufgenommen. Sogar die extreme Auffassung von "perfekten Lichtverhältnissen" ist voller Subjektivität, sowohl hinsichtlich des Fotografen als auch hinsichtlich des Gegenstandes. Jedoch ist in allen Fällen der Gegenstand eines Fotos entweder von Licht einer Lichtquelle (wie z. B. der Sonne oder Innenbeleuchtung) oder seinem eigenen Licht (wie z. B. ein Neon-Reklameschild) beleuchtet.

[0189] Bei den meisten Lichtverhältnissen ist das, was dem Fotografen als "weißes" Licht erscheinen mag, üblicherweise weit weg von weiß. Innenbeleuchtung beispielsweise hat typischerweise einen Gelbstich und dieser Gelbstich wird auf einem nicht korrigierten Foto sichtbar. Für die meisten Menschen ist der Gelbstich auf einem finalen, unkorrigierten Foto falsch. Obwohl er mit den Sichtbedingungen zum Zeitpunkt der Aufnahme des Fotos übereinstimmt, stimmt er nicht mit der empfundenen Farbe des Objekts überein. Es ist deshalb entscheidend, einen Weißabgleich des Fotos durchzuführen, bevor man es ausdruckt.

[0190] Auf dieselbe Art und Weise kann ein Bild als von einer höheren Qualität empfunden werden, wenn der dynamische Bereich der Farben ausgedehnt wird, um sich dem vollen Bereich in jeder Farbebene anzupassen. Dies ist insbesondere nützlich durchzuführen, bevor ein Bild auf eine höhere Auflösung erneut abgetastet bzw. resampled wird. Wenn der dynamische Bereich höher ist, können Zwischenwerte an interpolierten Pixelpositionen verwendet werden, wodurch ein abgestuftes oder blockartiges Bild vermieden wird. Die Bereichserweiterung ist konzipiert, um den tatsächlich abgetasteten Werten den vollen 256 Werte umfassenden Bereich zu geben. In dem besten Falle wird dem niedrigsten Wert die 0 zugeordnet, und dem höchsten Wert die 255. Alle Zwischenwerte werden proportional Zwischenwerten zwischen 0 und 255 zugeordnet.

[0191] Mathematisch ist die durchgeführte Operation eine Translation von LowThreshold 72 auf 0 gefolgt von einem Maßstab. Die Formel ist nachfolgend dargestellt:

Pixel' = (Pixel – LowThresho ld) × RangeScaleFactor

wobei

$$RangeScaleFactor = \frac{256}{(HighThreshold - LowThreshold)}$$

[0192] RangeScaleFactor sollte auf einen Maximalwert begrenzt sein, um das Risiko zu reduzieren, den Bereich zu weit zu erweitern. Zu den Einzelheiten der Berechnung von LowThreshold 72 siehe Abschnitt 3.2.2 "Kenngrößen erfassen". Diese Werte (LowThreshold und RangeScaleFactor) sind unterschiedlich für jede Farbebene und brauchen nur einmal pro Bild berechnet zu werden.

[0193] Beide Aufgaben können gleichzeitig durchgeführt werden, wie in **Fig. 20** gezeigt:
Da dieser Schritt einen Skalierungsprozess umfasst, bleibt einem ein gewisser Bruchbestandteil in dem zugeordneten Wert übrig, z. B. kann der Wert 12 auf 5,25 abgebildet werden. Um den Bruchbestandteil eher nicht zu verwerfen, geben wir ein 10-Bit-Ergebnis (8 Bits der ganzen Zahl, 2 des Bruchteils) an die nächste Stufe der Bildverarbeitungskette weiter. Wir können uns den Speicher nicht leisten, das gesamte Bild mit mehr als 8 Bits zu speichern, aber wir können eine die höhere Auflösung bei der Resampling-Stufe gut verwenden. Demgemäß hat das Eingabebild 8 Bits, und das Ausgabebild weist 10 Bits pro Farbbestandteil auf. Der logische Prozess ist in **Fig. 21** dargestellt.

[0194] Es ist wichtig, einen Grundwert von 0 während der Subtraktion zu haben, so dass alle Werte unterhalb von LowThreshold 42 der 0 zugeordnet werden. In gleicher Weise muss die Multiplikation eine obere Schwelle von 255 für den ganzzahligen Bestandteil des Ergebnisses aufweisen, so dass Eingabewerte höher als HighThreshold 73 der 255 zugeordnet werden.

3.2.5 Neu Abtasten/Resample **64**

[0195] Die Farbfiltermatrix stellt lediglich einen einzelnen Farbbestandteil pro Pixel- (x, y) Koordinate bereit. Um das finale gedruckte Bild zu erzeugen, müssen wir die anderen Farbbestandteilwerte bei jedem Pixel erhalten. Schließlich brauchen wir die cyan, magenta und gelben Farbbestandteile bei jedem Pixel, aber um zu cyan, magenta und gelb zu gelangen, brauchen wir rot, grün und blau. Mit unserem Eine-Farbe-pro-Pixel könn-

ten wir den roten Bestandteil für eine bestimmte Position haben, wir müssen jedoch blau und grün schätzen. Oder wir haben grün und müssen rot und blau schätzen.

[0196] Auch wenn wir die vollen roten, grünen und blauen Farbbestandteile für jedes Farbfiltermatrix-Auflösungspixel hätten, ist das Farbfiltermatrix-Auflösungsbild nicht die letztliche Ausgabeauflösung. Darüber hinaus ist, obwohl sich das Ausgabeformat verändert, die physikalische Breite des gedruckten Bildes konstant (4 Zoll bei 1600 dpi). Die konstante Breite des Druckkopfes beträgt deshalb 6400 Punkte.

[0197] Es gibt zwei extreme Fälle zu berücksichtigen:

- Interpolieren auf Farbfiltermatrix-Auflösung (minimale Interpolation), und anschließend Durchführen von Schärfung, Farbumwandlung. Schließlich Hinaufskalieren auf die Druckauflösung. Dies hat den Vorteil eines konstanten Schärfungskerns und einer konstanten Farbumwandlung bei der geringen Auflösung. Es hat jedoch den Nachteil, dass mehr als 8 Bit pro Farbbestandteil erforderlich sind, der für das interpolierte Bild zu speichern ist, oder Zwischenwerte falsch während des letzten Aufskalierens auf die Druckauflösung interpoliert werden. Es hat ebenfalls den Nachteil, dass eine Aufskalier-Einheit erforderlich ist, die geeignet ist, einen Druckauflösungs-interpolierten Wert pro Zyklus zu erzeugen.
- Interpolieren auf Druckauflösung, anschließend Durchführung der Schärfung und der Farbumwandlung. Dies hat den Vorteil lediglich eines Resampling-Prozesses, wodurch maximale Exaktheit bereitgestellt wird. Es hat jedoch den Nachteil, dass eine Aufskalier-Einheit erforderlich ist, die geeignet ist, einen bi-kubischen interpolierten Wert pro Zyklus zu erzeugen, sowie die Durchführung der Schärfung und Farbumwandlung, alles auf einem Durchschnitt eines einzigen Zyklus. Der Schärfungskern muss groß genug sein, um den Farbfiltermatrix-Auflösungskern auf das Bild mit hoher Auflösung anzuwenden. Noch schlimmer, es müssen für die Schärfung mindestens 3 Fenster auf dem Ausgabebild behalten werden (jedes enthält eine Anzahl von 6400 Zeileneinträgen), da bei einem einzigen Druckzyklus die cyan-farbigen, magenta-farbigen und gelben Punkte Punkte aus 6 verschiedenen Zeilen darstellen.

[0198] Keiner dieser Fälle berücksichtigt die Tatsache, dass es sich bei der finalen Druckausgabe um einen zweiwertigen und nicht einen mit gleichmäßigem Tonverlauf handelt. Folglich kann hier ein Kompromiss bezüglich des Resamplings erzielt werden, und wir können das Beste aus beiden Verfahren erreichen.

[0199] Die Lösung ist, auf eine Zwischenauflösung zu interpolieren. Schärfung und Farbumwandlung passieren bei der Zwischenauflösung, gefolgt von einem Aufskalieren auf die Druckauflösung. Die Zwischenauflösung muss gering genug sein, um die Vorteile der kleinen Schärfungskerngröße und des Farbumwandlungszeitverlaufs zu ermöglichen. Die Zwischenauflösung muss jedoch hoch genug sein, so dass kein Qualitätsverlust beim Aufskalieren auf das zweiwertige Bild in Druckauflösung entsteht. Die Wirkung muss dieselbe sein, als wenn es eine einzige Interpolation auf die Druckauflösung gäbe (und nicht zwei).

[0200] Da das Druckbild als 1600 dpi geditherte zweiwertige Punkte gedruckt wird, kann es sicher als ein 320 dpi-Bild mit gleichmäßigem Tonverlauf (contone) dargestellt werden. Folglich bietet eine Zwischenauflösung von 1280 Pixel mit gleichmäßigem Tonverlauf keinen empfundenen Qualitätsverlust gegenüber 6400 zweiwertigen Punkten. Die spätere Skalierung von 1280 auf 6400 ist deshalb ein exaktes Skalierungsverhältnis von 1:5.

[0201] Um zu entscheiden, wie man am besten neu abtastet bzw. resamplet, ist es am besten, jede Farbebene in Relation zur Farbfiltermatrix-Auflösung zu betrachten. Dies ist in **Fig. 22** für eine Drehung von 0 Grad dargestellt.

3.2.5.1 Rot **45** und blau **47**

[0202] Betrachtet man die roten und blauen Ebenen **45** bzw. **47**, kann die volle Farbfiltermatrix-Auflösungsversion der Farbebene durch Aufskalieren der Anzahl der abgetasteten Pixel in jeder Dimension durch den Faktor 2 erzeugt werden. Die Zwischenpixel können mittels eines Rekonstruktionsfilters (wie z. B. ein Lanczos- oder exponentieller Filter) erzeugt werden. Es ist lediglich eine Dimension in dem Kern erforderlich, da der Kern symmetrisch ist. Da rot und blau unterschiedliche Versätze bzw. Offsets hinsichtlich ihrer anfänglichen Darstellung innerhalb des Farbfiltermatrix-Abtastraumes aufweisen, sind ihre Anfangspositionen in dem Kern unterschiedlich.

[0203] Die Zuordnung der Ausgabekoordinaten (im 1280er Raum) zu Eingabekoordinaten hängt von der aktuellen Drehung des Bildes ab, da die Registrierung der Pixel sich mit der Drehung ändert (entweder 0 Grad oder 90 Grad je nach Druckformat). Für rot und blau gilt dann die folgende Beziehung:

$$\left.\begin{aligned} x' &= \left(\frac{x}{mps}\right) + k_1 \\ y' &= \left(\frac{y}{mps}\right) + k_2 \end{aligned}\right\}$$

wobei
x, y = Koordinate im Raum mit mittlerer Auflösung
x', y' = Koordinate im Eingaberaum
mps = Pixel in mittlerer Auflösung pro Eingaberaumabtastung
$k_{1,2}$ = {0, –0,5} je nach Drehung.

[0204] Dies bedeutet, dass man bei einer gegebenen Startposition im Eingaberaum eine neue Zeile aus Pixeln mit mittlerer Auflösung durch Hinzufügen eines Δx und Δy von 1/mps bzw. 0 1279 Mal erzeugen kann. Die Bruchteile von x und y im Eingaberaum können direkt zum Nachschlagen der Kernkoeffizienten zur Bildrekonstruktion und zum Resampling verwendet werden.

[0205] Es ist anzumerken, dass k_1 und k_2 0 und –0,5 sind, je nachdem, ob das Bild um 0 oder 90 Grad gedreht wurde. Tabelle 8 zeigt die Werte für k_1 und k_2 in den roten und blauen Ebenen unter der Annahme, dass die Drehung 90 Grad gegen den Uhrzeigersinn erfolgt.

Tabelle 8. Wirkung der Drehung auf k1 und k2 (Drehung ist gegen den Uhrzeigersinn)

Format	Drehung der ursprünglichen Farbfiltermatrix	Rot		Blau	
		k_1	k_2	k_1	k_2
Standardformat 30	90	0	-0,5	-0,5	0
Hochformat 31	90	0	-0,5	-0,5	0
Querformat 33	90	0	-0,5	-0,5	0
Miniaturformat 32	0	0	0	-0,5	-0,5

[0206] Die Anzahl der Pixel in mittlerer Auflösung pro Abtastung, mps, hängt vom Druckformat ab. Gibt man vor, dass das planarisierte RGB-Bild die folgenden roten und blauen planaren Auflösungen aufweist, wenn es nicht gedreht ist: R: 750 × 500, B: 750 × 500; die Skalierungsfaktoren für die unterschiedlichen Ausgabeformate (siehe **Fig. 13**) sind in Tabelle 9 dargestellt. Es ist anzumerken, dass im Hochformat das gesamte Bild auf 1/4 des Ausgaberaumes resamplet wird.

Tabelle 9. Rote und blaue Skalierungsfaktoren für Bildformate

Format	**Zuordnung**	**mps**	**1/mps**
Standardformat 30	500 ⇒ 1280	2,56	0,390625
Hochformat 31	500 ⇒ 640	1,28	0,78125
Querformat 33	250 ⇒ 1280	5,12	0,1953125
Miniaturformat 32	750 ⇒ 1280	1,71	0,5848

[0207] Wie aus Tabelle 9 ersichtlich werden die roten und blauen Bilder für alle Bildformate hochskaliert. Folglich werden keine Aliasing-Artefakte durch den Resampling-Prozess eingeführt.

3.2.5.2 Grün **46**

[0208] Die grüne Ebene **46** kann nicht einfach auf dieselbe Art und Weise wie rot oder blau hochskaliert werden, da jede Zeile der grünen Ebene unterschiedliche Pixel darstellt – entweder die geraden oder ungeraden Pixel auf abwechselnden Zeilen. Obwohl es hinsichtlich der Anzahl von Pixeln repräsentativ ist, zu sagen, dass das grüne Bild 750 × 1000 beträgt, könnte das Bild in gleicher Weise als 1500 × 500 bezeichnet werden. Diese Verwirrung tritt aufgrund des schachbrettartigen Wesens der grünen Pixel auf, wobei der Abstand zwischen Pixeln in x- und y-Richtung nicht gleich ist, und sich nicht gut zur Bildrekonstruktion oder zum Resampling zuordnen lässt. Die Anzahl der Interpolationsverfahren, die durch andere Systeme für die Rekonstruktion der grünen Ebene verwendet werden, ist ein Beleg dafür – von der Nachbildung des nächsten Nachbarn (nearest neighbor replication) über lineare Interpolation zur bi-linearen Interpolation und heuristischen Rekonstruktion.

[0209] Das Zuordnen der Ausgabekoordinaten (im 1280er Raum) zu Eingabekoordinaten ist konzeptionell dasselbe für grün wie für rot und blau. Die Zuordnung hängt von der aktuellen Drehung des Bildes ab, da die Registrierung der Pixel sich mit der Drehung ändert (entweder 0 oder 90 Grad je nach Druckformat). Für die grüne Ebene gilt die folgende Beziehung:

$$\left.\begin{aligned} x' &= \left(\frac{x}{mps}\right) + k_1 \\ y' &= \left(\frac{y}{mps}\right) + k_2 \end{aligned}\right\}$$

wobei
x, y = Koordinate im Raum mit mittlerer Auflösung
x', y' = Koordinate im Eingaberaum
mps = Pixel in mittlerer Auflösung pro Eingaberaumabtastung
$k_{1,2}$ = {0, –0,5} je nach Drehung.

[0210] Wie bei den roten und blauen Ebenen **45** bzw. **47** hängt die Anzahl der Pixel in mittlerer Auflösung pro Abtastung, mps, von dem Druckformat ab. Gibt man vor, dass das planarisierte RGB-Bild die folgenden planaren Auflösungen aufweist, wenn es ungedreht ist: R: 750 × 500, B: 750 × 500, G: 750 × 1000, die Skalierungsfaktoren für die unterschiedlichen Ausgabeformate (siehe Fig. 13) sind in Tabelle 10 dargestellt. Es ist anzumerken, dass bei dem Hochformat das gesamte Bild auf 1/4 des Ausgaberaumes resamplet wird.

Tabelle 10. Skalierungsfaktoren der grünen Ebene für Bildformate

Format	Zuordnung	mps	1/mps
Standardformat 30	1000 ⇒ 1280	1,28	0,78125
Hochformat 31	1000 ⇒ 640	0,64	1,5625
Querformat 33	500 ⇒ 1280	2,56	0,390625
Miniaturformat 32	1500 ⇒ 1280	0,85	1,17648

[0211] Diese Skalierungsfaktoren erlauben das Zuordnen von Koordinaten zwischen dem Farbfiltermatrix-Auflösungseingaberaum und dem Raum mit mittlerer. Auflösung. Sobald man jedoch eine Koordinate im Farbfiltermatrix-Auflösungseingaberaum hat, kann man keine Bildrekonstruktion und kein Resampling für die Abtastpunkte auf die gleiche Weise wie bei rot oder blau aufgrund der Schachbrett-Eigenschaft der grünen Ebene **46** durchführen.

[0212] Stattdessen kann man für die Zwecke der qualitativ hochwertigen Bildrekonstruktion und Resampling den grünen Kanal als ein Bild betrachten, das um 45 Grad gedreht ist. Wenn wir die Pixel in diesem Licht betrachten, wie in Fig. 23 dargestellt, wird eine qualitativ hochwertige Bildkonstruktion und ein Resampling-Verfahren deutlich.

[0213] Betrachtet man Fig. 23, dann ist der Abstand zwischen den gerasterten Pixeln in der X- und Y-Rich-

tung nun gleich. Der tatsächliche Abstand zwischen gerasteten Pixeln ist $\sqrt{2}$, wie in Fig. 24 dargestellt.

[0214] Die Lösung für den grünen Kanal ist damit, eine Bildrekonstruktion und ein Resampling im gedrehten Raum durchzuführen. Obwohl derselbe Rekonstruktionsfilter verwendet wird wie für das Resampling von rot und blau, sollte der Kern unterschiedlich sein. Dies deshalb, weil die Beziehung zwischen der Abtastrate für grün und der höchsten Frequenz in dem Signal anders als das Verhältnis für die roten und blauen Ebenen ist. Darüber hinaus sollte der Kern normalisiert sein, so dass der $\sqrt{2}$-Abstand zwischen den Abtastpunkten 1 wird, soweit die Kern-Koordinaten gehen (die nicht normalisierten Abstände zwischen Resampling-Koordinaten müssen jedoch immer noch verwendet werden, um zu bestimmen, ob ein Aliasing auftritt). Deshalb sind zwei Transformationen erforderlich:

- Die erste ist, den nicht gedrehten Farbfiltermatrixraum im gedrehten Farbfiltermatrixraum abzubilden. Dies kann durch Multiplizieren jeder Koordinate mit $1/\sqrt{2}$ erreicht werden, da wir um 45 Grad drehen (cos 45 = sin 45 = $1/\sqrt{2}$).
- Die zweite ist, die Koordinaten zu skalieren, um den normalisierten Kern anzupassen, was durch Multiplizieren jeder Ordinate mit $1/\sqrt{2}$ erreicht werden kann.

[0215] Diese beiden Transformationen kombinieren sich, um einen Multiplikationsfaktor von 1/2 zu erzeugen. Folglich erhöhen wir, indem wir im nicht gedrehten Farbfiltermatrixraum x um k voranschreiten, um k/2 im Kern x und vermindern um k/2 im Kern y. In ähnlicher Weise erhöhen wir, sobald wir in y um k voranschreiten, um k/2 in Kern x und erhöhen um k/2 im Kern y.

[0216] Die Beziehungen zwischen diesen unterschiedlichen Koordinatensystemen können durch Berücksichtigung dessen veranschaulicht werden, was passiert, wenn wir eine Zeile von Pixeln mit mittlerer Auflösung aus einem Farbfiltermatrixraum-Eingabebild erzeugen. Gibt man eine Start-y-Ordinate im Farbfiltermatrix-Eingaberaum vor, beginnen wir bei x = 0, und schreiten 1280 Mal mit 1/mps voran, wobei wir ein neues Pixel an jedem neuen Ort erzeugen. Die Bewegung im ungedrehten Farbfiltermatrixraum um 1/mps kann in eine Bewegung in x-Richtung und eine Bewegung in y-Richtung im gedrehten Farbfiltermatrixraum zerlegt werden. Der Prozess ist in Fig. 25 dargestellt.

[0217] Da cos 45 = sin 45 = $1/\sqrt{2}$ gilt, ist die Bewegung im ungedrehten Farbfiltermatrixraum um 1/mps gleich der gleichen Bewegung in x-Richtung und y-Richtung um $1/(\text{mps}\sqrt{2})$. Dieser Betrag muss nun skaliert werden, um den normalisierten Kern abzugleichen. Die Skalierung gleicht einer weiteren Multiplikation um $1/\sqrt{2}$. Folglich gleicht eine Bewegung von 1/mps im ungedrehten Farbfiltermatrixraum einer Bewegung um 1/2mps im Kern x und im Kern y. Die Tabelle 11 listet die Beziehung zwischen den 3 Koordinatensystemen für die unterschiedlichen Formate auf:

Tabelle 11. Δ-Werte des Kerns der grünen Ebene für Bildformate

Format	**Skalierungs-faktor (mps)**	**Ungedrehter Farbfiltermatrix-raum Δ** $\frac{1}{mps}$	**Gedrehter Farbfiltermatrix raum Δ** $\frac{1}{mps\sqrt{2}}$	**Kern Δ** $\frac{1}{2mps}$
Standardformat	1,28	0,78125	0,552	0,391
Hochformat	0,64	1,5625	1,105	0,781
Querformat	2,56	0,391	0,276	0,195
Miniaturformat	0,85	1,17648	0,832	0,601

[0218] Tabelle 11 zeigt, dass die Bewegung im Kernraum immer um eine Zahl kleiner als 1 ist, im gedrehten Farbfiltermatrixraum weist jedoch nur das Hochformatbild einen Δ-Wert von größer als 1 auf. Als Ergebnis tritt Aliasing für das Hochformat-Druckformat auf, jedoch nicht für eines der anderen Formate. Wenn man jedoch bedenkt, dass das Δ fast 1 ist und dass jedes der 4 Bilder lediglich 1/4 der Größe beträgt, wird sich das Aliasing nicht bemerkbar machen, insbesondere weil wir eine ideale Tiefpassfilterung bei grün während der Bilderfassung annehmen.

3.2.5.3 Rekonstruktionsfilter für rot, blau und grün

[0219] Der zu verwendende exakte Rekonstruktionsfilter hängt von einer Anzahl von Punkten ab. Es gibt immer eine Austauschbeziehung zwischen der Anzahl von Abtastpunkten, die bei der Konstruktion des ursprünglichen Signals verwendet werden, der für die Signalrekonstruktion verwendeten Zeit und der Qualität des erneut abgetasteten Bildes. Ein zufrieden stellender Ausgleich in diesem Fall ist 5 Pixel-Abtastpunkte aus der zu rekonstruierenden Dimension, mittig angeordnet um die geschätzte Position X, d. h. X-2, X-1, X, X+1, X+2. Aufgrund der Art der Rekonstruktion mit 5 Abtastpunkten brauchen wir lediglich 4 Koeffizienten für den Eintrag in dem Faltungskern.

[0220] Wir erzeugen eine Kernkoeffizienten-Nachschlagetabelle mit n Einträgen für jeden Farbbestandteil. Jeder Eintrag weist 4 Koeffizienten auf. Wenn wir im Ausgaberaum voranschreiten, ordnen wir die Änderungen im Ausgaberaum den Änderungen im Eingaberaum und im Kernraum zu. Die signifikantesten Bits der Bruchbestandteile in dem aktuellen Kernraum werden verwendet, um in die Kernkoeffiziententabelle zu indizieren. Wenn 64 Einträge in der Kerntabelle vorhanden sind, werden die ersten 6 Bestandteil-Bits verwendet, um die Koeffizienten nachzuschlagen. 64 Einträge ist allemal ausreichend für das Resampling in der Printcam.

3.2.6 Schärfen **65**

[0221] Das durch die Farbfiltermatrix erfasste Bild muss vor dem Ausdruck geschärft werden. Idealerweise sollte der Schärfungsfilter in dem Farbfiltermatrix-Auflösungsbereich angewendet werden. Bei der Bilderfassungsauflösung haben wir jedoch nicht die volle Farbinformationen bei jedem Pixel zur Verfügung. Stattdessen haben wir nur rot, blau oder grün an einer gegebenen Pixelposition. Schärfung jeder Farbebene unabhängig voneinander gibt Anlass zu Farbverschiebungen. Die Schärfung sollte stattdessen auf den Helligkeitskanal eines Bildes angewendet werden, so dass der Farbton und die Sättigung eines gegebenen Pixels unverändert bleibt.

[0222] Das Schärfen umfasst dann die Übersetzung eines RGB-Bildes in einen Farbraum, wo die Helligkeit vom Rest der Farbinformation (wie z. B. HLS oder Lab) **80** getrennt ist. Der Helligkeitskanal **81** kann anschließend geschärft werden (durch Hinzufügen in einem Verhältnis der Hochpass-gefilterten Version der Helligkeit). Schließlich sollte das gesamte Bild in RGB werden zurück konvertiert **83** (oder zu CMY, da wir in CMY ausdrucken werden). Der Prozess ist in **Fig. 26** dargestellt.

[0223] Wir können jedoch viele dieser Farbumwandlungsschritte vermeiden, wenn wir die Wirkung des Hinzufügens eines Hochpass-gefilterten L zurück in das Bild berücksichtigen – die Wirkung ist eine Veränderung bei der Helligkeit des Bildes. Eine Veränderung in der Helligkeit eines gegebenen Pixels kann gut durch eine gleichwertige Änderung im linearen R, G und B angenähert werden. Deshalb erzeugen wir einfach L, Hochpass-gefiltertes L und wenden ein Verhältnis des Ergebnisses in gleicher Weise auf R, G und B an.

3.2.6.1 Umwandeln von RGB in L **80**

[0224] Wir betrachten den CIE 1976 L*a*b* – Farbraum, wobei L von der Wahrnehmung her gleichmäßig ist. Um von RGB auf L (den Helligkeitskanal, engl. luminance channel) umzuwandeln, mitteln wir das Minimum und das Maximum von R, G und B wie folgt:

$$L = \frac{MIN(R,G,B) + MAX(R,G,B)}{2}$$

3.2.6.2 Hochpassfilter L **84**

[0225] Ein Hochpassfilter **84** kann anschließend auf die Helligkeitsinformationen angewendet werden. Da wir eher im Raum der mittleren Auflösung als im Farbfiltermatrix-Auflösungsraum filtern, kann die Größe des Schärfungskerns hoch skaliert werden, oder das Hochpassergebnis kann in entsprechender Weise skaliert werden. Die exakte Schärfungsmenge hängt von der Farbfiltermatrix ab, aber ein 3 × 3-Faltungskern **85** ist ausreichend, um gute Ergebnisse zu erzielen.

[0226] Wenn wir die Größe des Kern erhöhen sollten, zeigt Tabelle 12 die effektive Skalierung **86**, die für eine 3 × 3-Faltung im Farbfiltermatrix-Raum erforderlich ist, wie sie auf einen 1280er Auflösungsraum angewendet wird, wobei der grüne Kanal als Basis für die Skalierung des Kerns verwendet wird. Aus dieser Tabelle wird klar, dass ein 7 × 7 großer Kern, angewendet auf den Raum mit mittlerer Auflösung, für die gesamte Schärfung hinreichend ist.

Tabelle 12: Skalierungsfaktoren für Faltungsfilter

Format	Maßstab	3 x 3	Kern im Raum mit mittlerer Auflösung (1280)
Standardformat 30	1,28	3,84	3 x 3 oder 5 x 5
Hochformat 31	0,64	1,92	kein, oder 3 x 3
Querformat 33	2,56	7,68	7 x 7
Miniaturformat 32	0,85	2,55	kein, oder 3 x 3

[0227] Wenn ein 3 × 3-Filter **85** auf ein Bild mit mittlerer Auflösung angewendet würde, wird das Ergebnis entsprechend dem Skalierungsfaktor skaliert **86**, der bei dem allgemeinen Bildskalierungsvorgang verwendet wird. Mit den in Tabelle 12 angegebenen Beträgen (insbesondere dem Standard-Druckformat), können wir einen 3 × 3-Filter **85** verwenden und anschließend die Ergebnisse skalieren. Der Prozess des Erzeugens eines einzelnen gefilterten L-Pixels ist in Fig. 27 dargestellt.

[0228] Der tatsächlich verwendetet Kern kann ein beliebiger aus einem Satz von Standard-Hochpassfilter-Kernen sein. Ein einfacher, aber ausreichender Hochpassfilter ist in dieser Implementierung des PCP in Fig. 50 dargestellt.

3.2.6.3 Hinzufügen von gefiltertem L zu RGB

[0229] Der nächste Schritt ist, einen gewissen Anteil der resultierenden Hochpass-gefilterten Helligkeitswerte wieder dem Helligkeitskanal hinzuzufügen. Das Bild kann dann zurück zu RGB (oder stattdessen zu CMY) umgewandelt werden. Eine Helligkeitsänderung kann jedoch auch in vernünftiger Weise durch eine gleichmäßige Änderung bei R, G und B angenähert werden (solange der Farbraum linear ist). Folglich kann man die Farbumwandlungen komplett durch Hinzufügen von gleichen Anteilen des Hochpass-gefilterten Helligkeitswerts zu R, G und B vermeiden. Der exakte Anteil des Hochpass-gefilterten Bildes kann mittels eines Skalierungsfaktors definiert werden.

[0230] Wenn L das Hochpass-gefilterte Helligkeitspixel ist und k der konstante Skalierungsfaktor, können wir die Transformation der Schärfung von R, G und B wie folgt definieren:

$$\left.\begin{aligned} R' &= R + kL \\ G' &= G + kL \\ B' &= B + kL \end{aligned}\right\} \text{ (begrenzt auf jeweils 255)}$$

[0231] Natürlich kann der auf L angewendete Skalierungsfaktor mit dem Skalierungsfaktor in dem Hochpassfilterprozess (siehe Abschnitt 3.2.6.2) zu einem einzigen Skalierungsfaktor kombiniert werden.

[0232] Sobald die Schärfung auf das RGB-Pixel angewendet wurde, kann das Bild in CMY **83** umgewandelt werden, um ausgedruckt zu werden.

3.2.7 Umwandeln in CMY **83**

[0233] In theoretischer Darstellung ist die Umwandlung von RGB zu CMY einfach:

C = 1 – R

M = 1 – G

Y = 1 – B

[0234] Bei dieser Umwandlung gilt jedoch, dass der CMY-Raum eine lineare Antwort aufweist, was sicher nicht für Pigmenttinten und nur teilweise für Farbstofftinten gilt. Das individuelle Farbprofil einer bestimmten Vorrichtung (Eingabe und Ausgabe) kann erheblich variieren. Folglich ist, um eine genaue Umwandlung sowie für zukünftige Sensoren, Tinten und Drucker zu ermöglichen, ein exakteres Modell für eine Printcam erforder-

lich.

[0235] Die erforderlichen Transformationen sind in Fig. 28 dargestellt. Lab wird ausgewählt, da es von der Wahrnehmung her gleichmäßig ist (anders als XYZ). Hinsichtlich der Zuordnung von der Bildsensorskala zur Druckerskala ist typischerweise die Druckerskala vollständig innerhalb der Sensorskala enthalten.

[0236] Anstatt diese Transformationen erschöpfend durchzuführen, können ausgezeichnete Ergebnisse über eine tri-lineare Umwandlung basierend auf 3 Sätzen von 3D-Nachschlagetabellen erhalten werden. Die Nachschlagetabellen enthalten die resultierenden Transformationen für den speziellen Eintrag, wie er durch RGB indiziert ist. Drei Tabellen sind erforderlich: eine Tabelle **90**, die RGB zu C zuordnet, eine Tabelle **91**, die RGB zu M zuordnet und eine Tabelle **92**, die RGB zu Y zuordnet. Die tri-lineare Interpolation kann verwendet werden, um das letztliche Ergebnis für die Einträge zu ergeben, die nicht in den Tabellen enthalten sind. Der Prozess ist in Fig. 29 dargestellt.

[0237] Die tri-lineare Interpolation erfordert das Lesen von 8 Werten aus der Nachschlagetabelle und die Durchführung von 7 linearen Interpolationen (4 in der ersten Dimension, 2 in der zweiten und 1 in der dritten). Eine hohe Genauigkeit kann für die Zwischenwerte verwendet werden, obwohl der Ausgabewert lediglich 8 Bit aufweist.

[0238] Die Größe der erforderlichen Nachschlagetabelle hängt von der Linearität der Transformation ab. Die empfohlene Größe für jede Tabelle in dieser Anwendung ist 17 × 17 × 17(Obwohl eine 17 × 17 × 17-Tabelle exzellente Ergebnisse liefert, kann es möglich sein, mit nur einer 9 × 9 × 9 Umwandlungstabelle auszukommen (729 Bytes). Die exakte Größe kann durch Simulation bestimmt werden. Der 5KB-Ansatz "konservativ-aber-exakte-Ergebnisse" wurde für die Zwecke dieses Dokuments ausgewählt.), wobei jeder Eintrag 8 Bits aufweist. Eine 17 × 17 × 17 Tabelle weist 4913 Bytes auf (weniger als 5 KB).

[0239] Um in die 17-pro-Dimension-Tabellen zu indizieren, werden die 8-Bit-Eingabefarbbestandteile als Festkommazahlen (4:4) behandelt. Die 4 Bits des Ganzzahlwertes ergeben den Index, und die 4 Bits des Hochteils werden für die Interpolation verwendet.

3.2.8 Hochinterpolieren **67**

[0240] Das CMY-Bild in mittlerer Auflösung (1280 breit) muss nun auf die schließliche Druckauflösung (6400 breit) hochinterpoliert werden. Das Verhältnis ist genau 1:5 in beiden Dimensionen.

[0241] Obwohl es sicher möglich ist, die 25 Werte bi-linear zu interpolieren (1:5 sowohl in X- als auch in Y-Richtung), werden die resultierenden Werte nicht mit gleichmäßigem Farbverlauf gedruckt. Die Ergebnisse werden gedithert und zweiwertig gedruckt. Wenn man bedenkt, dass die 1600 dpi-Ergebnisse mit gleichmäßigem Farbverlauf in geditherte zweiwertige Punkte verwandelt werden, wird die Genauigkeit der bi-linearen Interpolation von 320 dpi auf 1600 dpi nicht sichtbar (genau aus diesem Grunde wurde die mittlere Auflösung gewählt). Die Pixelwiederholung bzw. -replikation liefert deshalb gute Ergebnisse.

[0242] Die Pixelreplikation umfasst einfach das Nehmen eines einzelnen Pixels und seine Verwendung als den Wert für einen größeren Bereich. In diesem Fall replizieren wir ein einzelnes Pixel auf 25 Pixel (einen 5 × 5-Block). Wenn jedes Pixel einen gleichmäßigen Farbverlauf aufweisen würde, kann das Ergebnis blockartig erscheinen, da jedoch die Pixel gedithert werden müssen, ist die Wirkung, dass die 25 resultierenden zweiwertigen Punkte den gleichmäßigen Farbwert annehmen. Der Prozess ist in Fig. 30 dargestellt.

3.2.9 Halbtönen **68**

[0243] Der Druckkopf **2** ist lediglich geeignet, Punkte in einer zweiwertigen Art und Weise zu drucken. Man muss deshalb von dem gleichmäßigen CMY- zu einem geditherten CMY-Bild umwandeln. Insbesondere erzeugen wir ein verteiltes punktgeordnetes Dithering unter Verwendung einer stochastischen Dither-Zelle, wobei wir ein CMY-Bild mit gleichmäßigem Farbverlauf in ein gedithertes zweiwertiges CMY-Bild konvertieren.

[0244] Der 8 Bit 1600 dpi Wert mit gleichmäßigem Tonverlauf wird mit der aktuellen Position in der Dither-Zelle **93** verglichen. Wenn der 8-Bit-Wert mit gleichmäßigem Tonverlauf größer ist als der Dither-Zellen-Wert, wird ein Ausgabebit von 1 erzeugt. Anderenfalls wird ein Ausgabebit von 0 erzeugt. Dieses Ausgabebit wird letztendlich an den Druckkopf gesendet und steuert eine einzelne Düse, um einen einzelnen C-, M- oder Y-Punkt hervorzubringen. Das Bit steht dafür, ob eine bestimmte Düse für eine vorgegebene Farbe und Position zündet

oder nicht.

[0245] Dieselbe Position in der Dither-Zelle **93** kann für C, M und Y verwendet werden. Dies geschieht deshalb, weil der tatsächliche Druckkopf **2** die C-, M- und Y-Punkte für unterschiedliche Zeilen in demselben Druckzyklus erzeugt. Das Versetzen der unterschiedlich gefärbten Punkte führt wirksam zu einem Versetzen in der Dither-Zelle.

[0246] Der Halbtonungsprozess kann in Fig. 31 angesehen werden.

[0247] Die Größe der Dither-Zelle **93** hängt von der Auflösung der Ausgabepunkte ab. Da wir Punkte mit 1600 dpi erzeugen, sollte die Zellengröße größer als 32 × 32 sein. Darüber hinaus sollte, um zu ermöglichen, dass die Punktverarbeitungsreihenfolge zu den Druckkopf-Segmenten passt, die Größe der Dither-Zelle idealerweise gerade durch 800 teilbar sein (da 800 Punkte in jedem Segment des Druckkopfs vorhanden sind).

[0248] Eine Dither-Zellengröße von 50 × 50 ist groß genug, um qualitativ hochwertige Ergebnisse zu erzeugen und lässt sich gerade durch 800 teilen (16 Mal). Jeder Eintrag der Dither-Zelle hat 8 Bits und ergibt eine Gesamtzahl von 2500 Bytes (ungefähr 1,5 KB).

3.2.10 Umformatieren für Drucker **69**

[0249] Der letzte Prozess für die Punkte, bevor sie zu dem Drucker gesendet werden, ist, in der richtigen Reihenfolge für das Senden an den Druckkopf formatiert zu werden. Die Punkte müssen in der richtigen Reihenfolge an den Druckkopf gesendet werden – jeweils 24 Punkte wie in Abschnitt 2.2.1 definiert.

[0250] Wenn die Punkte in der richtigen Reihenfolge für das Drucken erzeugt werden können (d. h. die Hochinterpolier- und Ditherfunktionen erzeugen ihre Daten in der richtigen Reihenfolge), dann können diese Punktwerte (jeder Wert ist 1 Bit) einfach gesammelt werden und in Gruppen von 24 abgesendet werden. Der Prozess ist in Fig. 32 dargestellt.

[0251] Die 24 Bit-Gruppen können dann an den Druckkopf **2** über die Memjet-Schnittstelle **15** gesendet werden.

4. CPU-KERN UND SPEICHER

4.1 CPU-KERN **10**

[0252] Der PCP **3** umfasst einen einfachen Mikocontroller-CPU-Kern **10**, um die Bilderfassungs- und Bilddruckverarbeitungsketten zu synchronisieren und die Aufgaben des allgemeinen Betriebssystems der Printcam einschließlich der Benutzerschnittstelle durchzuführen. Eine große Vielzahl von CPU-Kernen ist dazu geeignet: Es kann jeder Prozessorkern mit ausreichend Verarbeitungsleistung sein, um die erforderlichen Berechnungen und Steuerfunktionen schnell genug durchzuführen, um die Kundenerwartungen zu erfüllen.

[0253] Da die gesamte Bildverarbeitung von spezieller Hardware durchgeführt wird, muss die CPU keine Pixel verarbeiten. Daraus ergibt sich, dass die CPU sehr einfach sein kann. Sie muss jedoch schnell genug sein, um den Schrittmotor während eines Druckes anzusteuern (der Schrittmotor erfordert einen 5 KHz-Prozess). Ein Beispiel eines geeigneten Kerns ist ein Philips 8051 Mikrocontroller, der bei etwa 1 MHz läuft.

[0254] Es besteht kein Bedarf, eine Befehlssatz-Kontinuität zwischen unterschiedlichen Printcam-Modellen aufrecht zu erhalten. Unterschiedliche PCP-Chip-Designs können von unterschiedlichen Herstellern hergestellt werden, ohne dass es erforderlich ist, den CPU-Kern zu lizenzieren oder zu portieren. Diese Geräteunabhängigkeit verhindert die Marktsperre (lock-in) für den Chip-Hersteller, wie es auf dem PC-Markt mit Intel aufgetreten ist.

[0255] Verbunden mit dem CPU-Kern ist ein Programm-ROM-Speicher **13** und ein kleiner Programm-Scratch-RAM-Speicher **14**.

[0256] Die CPU **10** kommuniziert mit den anderen Einheiten innerhalb des PCP **3** über Speicher-zugeordneten E/A. Bestimmte Adressbereiche lassen sich auf bestimmte Einheiten zuordnen, und innerhalb jedes Bereichs zu bestimmten Registern innerhalb dieser bestimmten Einheit. Dies umfasst die seriellen und parallelen Schnittstellen.

4.2 PROGRAMM-ROM-SPEICHER **13**

[0257] Ein kleiner Programm-Flash-ROM-Speicher **13** ist in den PCP **3** eingebaut. Die ROM-Speichergröße hängt von der ausgewählten CPU ab, sollte jedoch nicht mehr als 16-32 KB betragen.

4.3 PROGRAMM-RAM-SPEICHER **14**

[0258] In ähnlicher Weise ist ein kleiner Scratch-RAM-Speicherbereich **14** in den PCP **3** eingebaut. Da der Programm-Code keine Bilder manipulieren muss, besteht keine Notwendigkeit für einen großen Scratch-Bereich. Die RAM-Speichergröße hängt von der ausgewählten CPU ab (z. B. Stapelmechanismus, Aufrufkonventionen von Subroutinen, Registergrößen usw.), sollte jedoch nicht mehr als 4 KB betragen.

4.4 CPU-SPEICHER-DECODER **16**

[0259] Der CPU-Speicher-Decoder **16** ist ein einfacher Decoder, um den CPU-Datenzugriffen zu genügen. Der Decoder übersetzt die Datenadressen in interne PCP-Registerzugriffe über den internen Niedriggeschwindigkeitsbus und ermöglicht damit Speicher-zugeordneten E/A des PCP-Registers.

5. KOMMUNIKATIONSSCHNITTSTELLEN

5.1 SERIELLE USB-SCHNITTSTELLE **17**

[0260] Dies ist ein standardmäßiger serieller USB Anschluss, verbunden mit dem internen Chip-Niedriggeschwindigkeitsbus **18**. Der serielle USB-Anschluss wird durch die CPU **10** gesteuert. Der serielle Anschluss ermöglicht die Übertragung von Bildern zu und von der Printcam und erlaubt, dass DPOF-(Digital Print Order Format)Drucken von übertragenen Fotos unter externer Steuerung.

5.2 SERIELLE QA-CHIP-SCHNITTSTELLE **19**

[0261] Dies sind zwei standardmäßige serielle Niedriggeschwindigkeitsanschlüsse, verbunden mit dem internen Chip-Niedriggeschwindigkeitsbus **18**. Das CPU-vermittelte Protokoll zwischen den beiden wird verwendet, um die Druckrolle [1,2] zu authentifizieren, und für die folgenden Funktionen:

- Erfassen der Tinteneigenschaften
- Erfassen des empfohlenen Tropfenvolumens
- Verfolgen der gedruckten Papiermenge und Anfordern einer neuen Druckrolle, wenn nicht mehr genügend Papier vorhanden ist, um das angeforderte Druckformat auszudrucken.

[0262] Der Grund für das Vorhandensein von zwei Anschlüssen ist, sowohl mit dem auf der Kamera vorhandenen QA-Chip **4** als auch mit dem QA-Chip **5** der Druckrolle unter Verwendung von getrennten Leitungen zu verbinden. Die beiden QA-Chips werden als Authentifizierungs-Chips [2] implementiert. Wenn lediglich eine einzelne Leitung verwendet wird, könnte sich ein Hersteller einer nachgemachten Druckrolle den Authentifizierungsmechanismus widerrechtlich aneignen [1].

5.2.1 QA Chip **5** der Druckrolle

[0263] Jede verbrauchbare Druckrolle enthält ihren eigenen QA-Chip **5**. Der QA-Chip enthält Informationen, die für die Aufrechterhaltung der bestmöglichen Druckqualität erforderlich sind, und wird unter Verwendung eines Authentifizierungs-Chips [2] implementiert. Die 256 Bits an Daten werden wie folgt allokiert:

Tabelle 13. 256 Bits (16) der Druckrolle

M[n]	Zugriff	Beschreibung
0	RO[a]	Grund-Kopfsatz, Merker usw. (16 Bits)
1	RO	Seriennummer (16 Bits)
2	RO	Batch-Nummer (16 Bits)
3	DO[b]	verbleibendes Papier in mm (16 Bits)
4	RO	Eigenschaften der cyan-Tinte (32 Bits)
5	RO	
6	RO	Eigenschaften der magenta-Tinte (32 Bits)
7	RO	
8	RO	Eigenschaft der gelben Tinte (32 Bits)
9	RO	
10-12	RO	für zukünftige Erweiterung = 0 (48 Bits)
13-15	RO	Zufalls-Bits, unterschiedlich in jedem Bit (48 Bits)

a. nur Lesen
b. nur abnehmend

[0264] Vor jedem Druck wird die Menge an verbleibendem Papier von der CPU überprüft, um sicherzustellen, dass für das aktuell bestimmte Druckformat genug Papier vorhanden ist. Nachdem jeder Druck begonnen hat, muss die Menge an verbleibendem Papier in dem QA-Chip der Druckrolle durch die CPU verringert werden.

5.3 PARALLELE SCHNITTSTELLE **6**

[0265] Die parallele Schnittstelle **6** verbindet den PCP **3** mit individuellen statischen elektrischen Signalen. Die CPU ist geeignet, jede dieser Verbindungen als Speicher-zugeordneten E/A über den Niedriggeschwindigkeitsbus zu steuern (siehe Abschnitt 4.4 zu mehr Einzelheiten des Speicher-zugeordneten E/A).

[0266] Tabelle 14 zeigt die Verbindungen zu der parallelen Schnittstelle

Tabelle 14. Verbindungen zur parallelen Schnittstelle

Verbindung	Richtung	Anschlüsse
Papiertransport-Schrittmotor	Aus	4
Guillotine-Motor	Aus	1
Fokus-Motor	Aus	1
Kappungsmagnet	Aus	1
Flash-Trigger	Aus	1
Status-LCD Segmenttreiber	Aus	7
Status-LDC allgemeine Treiber	Aus	4
Papierzugsensor	An	1
Knöpfe	An	4
	GESAMTANZAHL	24

5.4 JTAG-SCHNITTSTELLE **7**

[0267] Eine Standard JTAG(Joint Test Action Group)-Schnittstelle **7** ist in dem PCP **3** zu Testzwecken enthalten. Aufgrund der Komplexität des Chips ist eine Vielzahl von Testtechniken einschließlich BIST (Built In Self Test) und funktionaler Blockisolation erforderlich. Ein Overhead von 10 % beim Chipbereich wird für die gesamte Chiptestschaltung angenommen.

6. BILD-RAM-SPEICHER **11**

[0268] Der Bild-RAM-Speicher **11** wird verwendet, um das erfasste Bild **42** zu speichern. Der Bild-RAM-Speicher ist mehrstufiger Flash-Speicher (2 Bits pro Zelle), so dass das Bild behalten wird, nachdem die Stromversorgung abgeschaltet worden ist.

[0269] Die erforderliche Gesamtspeichermenge für das planarisierte lineare RGB-Bild beträgt 1.500.000 Bytes (ungefähr 1,5 MB), die wie folgt angeordnet sind:

- R: 750 × 500 = 375.000 Bytes
- B: 750 × 500 = 375.000 Bytes
- G: 750 × 1000 = 750.000 Bytes

[0270] Das Bild wird durch die Bilderfassungseinheit geschrieben und sowohl von der Bild-Histogrammeinheit **8** und der Druckerzeugungseinheit **99** gelesen. Die CPU **10** hat keinen Direktzugriff auf diesen Bildspeicher. Sie muss auf die Bildpixel über die Bildzugriffseinheit zugreifen.

7. BILDERFASSUNGSEINHEIT **12**

[0271] Die Bilderfassungseinheit enthält sämtliche von der Bilderfassungskette erforderliche Funktionalität, wie in Abschnitt 3.1 beschrieben. Die Bilderfassungseinheit akzeptiert Pixeldaten über die Bildsensorschnittstelle **98**, linearisiert die RGB-Daten über eine Nachschlagetabelle **96** und schreibt schließlich das linearisierte RGB-Bild in planarem Format hinaus auf den RAM-Speicher. Der Prozess ist in Fig. 33 dargestellt.

7.1 BILDSENSORSCHNITTSTELLE **98**

[0272] Die Bildsensorschnittstelle (Image Sensor Interface; ISI) **98** ist ein Zustandsautomat, der Steuerinformationen an den CMOS-Bildsensor sendet, einschließlich Vollbild-Synchronisierungsimpulse und Pixeltaktimpulse, um das Bild zu lesen. Mit einiger Wahrscheinlichkeit ist das meiste der ISI eine ausgelagerte Zelle von dem Bildsensorhersteller. Die ISI wird selbst durch den Bilderfassungseinheit-Zustandsautomaten **97** gesteuert.

7.1.1 Bildsensorformat

[0273] Obwohl eine Vielzahl von Bildsensoren verfügbar ist, berücksichtigen wir nur die Bayer-Farbfiltermatrix (CFA). Die Bayer-CFA weist eine Anzahl von Attributen auf, die hierin definiert sind.

[0274] Das durch den CMOS-Sensor (über eine Aufnahmelinse) erfasste Bild wird als ausreichend gefiltert angenommen, um irgendwelche Aliasing-Artefakte zu entfernen. Der Sensor selbst weist ein Bild-Seitenverhältnis von 3:2 mit einer Auflösung von 1500 × 1000 Tastpunkten auf. Die wahrscheinlichste Pixelanordnung ist die Bayer-Farbfiltermatrix (CFA), wobei jeder 2 × 2 Pixelblock in einem 2G-Mosaik angeordnet ist, wie in Fig. 15 dargestellt:
Jeder Abtastpunkt mit gleichmäßigem Tonverlauf aus R, G oder B (entsprechend rot, grün bzw. blau) ist 10 Bits groß. Es ist anzumerken, dass jedes Pixel des Mosaiks Informationen über nur eine der Farben R, G oder B enthält. Schätzungen der fehlenden Farbinformationen müssen durchgeführt werden, bevor das Bild ausgedruckt werden kann.

[0275] Es wird berücksichtigt, dass die Farbfiltermatrix einen gewissen Betrag der Unterdrückung der feststehenden Störstruktur (Fixed Pattern Noise; FPN) durchführt. Eine zusätzliche FPN-Unterdrückung kann erforderlich sein.

7.2 NACHSCHLAGETABELLE **96**

[0276] Die Nachschlagetabelle **96** ist ein ROM-Speicher, der das RGB des Sensors auf ein lineares RGB abbildet. Das Abbilden stimmt mit dem Prozess des Linearisierens des RGB **40** überein, der in Abschnitt 3.1.2 beschrieben wurde. Als solches ist der ROM-Speicher 3 KBytes groß (3 × 1024 × 8 Bits). 10 Bits der Adresse kommen von der ISI, während die 2 Bits des TableSelect durch den Zustandsautomaten **97** der Bilderfassungseinheit erzeugt werden.

7.3 ZUSTANDSAUTOMAT **97**

[0277] Der Zustandsautomat **97** der Bilderfassungseinheit erzeugt Steuersignale für die Bildsensorschnittstelle **1** und erzeugt Adressen zum Linearisieren des RGB **40** und zum Planarisieren der Bilddaten **41**.

[0278] Die Steuersignale, die an die ISI **98** gesendet werden, informieren die ISI, mit der Erfassung von Pixeln zu beginnen, mit der Erfassung von Pixeln zu stoppen usw.

[0279] Die an die Nachschlagetabelle **96** gesendete 2-Bit-Adresse stimmt mit der aktuellen Zeile überein, die aus der ISI gelesen wird. Für gerade Zeilen (0, 2, 4 usw.) ist die 2-Bit-Adresse rot, grün, rot, grün usw. Für ungerade Zahlen (1, 3, 5 usw.) ist die 2-Bit-Adresse grün, blau, grün, blau. Dies ist unabhängig von der Orientierung der Kamera richtig.

[0280] Die an den Bild-RAM-Speicher **11** gesendete 21-Bit-Adresse ist die Schreibadresse für das Bild. Drei Register halten die momentane Adresse für jede der roten, grünen und blauen Ebenen. Die Adressen erhöhen sich mit dem Schreiben der Pixel auf jede Ebene.

7.3.9 Register

[0281] Die Bilderfassungseinheit enthält eine Anzahl von Registern:

Tabelle 15. Register in der Bilderfassungseinheit

Name	Bits	Beschreibung
MaxPixels	12	Anzahl der Pixel in jeder Zeile
MaxRows	12	Anzahl der Zeilen von Pixel im Bild
CurrentPixel	12	Pixel, das gerade geholt wird
CurrentRow	12	Zeile, die gerade verarbeitet wird
NextR	21	Die Adresse im Bild-RAM-Speicher, um das nächste rote Pixel zu speichern. Wird auf die Startadresse der roten Ebene vor der Bilderfassung gesetzt. Nach der Bilderfassung zeigt dieses Register auch das Byte nach der roten Ebene.
NextG	21	Die Adresse im Bild-RAM-Speicher, um das nächste grüne Pixel zu speichern. Wird auf die Startadresse der grünen Ebene vor der Bilderfassung gesetzt. Nach der Bilderfassung zeigt dieses Register auch das Byte nach der grünen Ebene.
NextB	21	Die Adresse im Bild-RAM-Speicher, um das nächste blaue Pixel zu speichern. Wird auf die Startadresse der blauen Ebene vor der Bilderfassung gesetzt. Nach der Bilderfassung zeigt dieses Register auch das Byte nach der blauen Ebene.
EvenEven	2	Für gerade Zeilen/gerade Pixel zu verwendende Adresse
EvenOdd	2	Für gerade Zeilen/ungerade Pixel zu verwendende Adresse
OddEven	2	Für ungerade Zeilen/gerade Pixel zu verwendende Adresse
OddOdd	2	Für ungerade Zeilen/ungerade Pixel zu verwendende Adresse
Go	1	Schreibt man eine 1 hier, beginnt die Erfassung. Schreibt man eine 0 hier, hält die Bilderfassung an. Eine 0 wird hier automatisch durch den Zustandsautomaten geschrieben, nachdem MaxRows von MaxPixels erfasst wurden.

[0282] Darüber hinaus enthält die Bildsensorschnittstelle **98** eine Reihe von Registern. Die exakten Register hängen von dem ausgewählten Bildsensor **1** ab.

8. BILDZUGRIFFSEINHEIT **9**

[0283] Die Bildzugriffseinheit (Image Access Unit; IAU) **9** erzeugt die Mittel für die CPU **10**, um auf das Bild in dem Bild-RAM-Speicher **11** zuzugreifen. Die CPU **10** kann Pixel aus dem Bild in dem Bild-RAM-Speicher **11** lesen und Pixel zurück schreiben.

[0284] Pixel können zum Zwecke der Bildspeicherung (z. B. über die USB-Schnittstelle) **17** oder zur einfachen Bildverarbeitung gelesen werden. Pixel können in den Bild-RAM-Speicher **11** nach der Bildverarbeitung geschrieben werden, wie ein zuvor abgespeichertes Bild (über USB geladen) oder Bilder für Testmusterzwecke. Testmuster können künstliche Bilder sein, besondere Testbilder (über die USB-Schnittstelle geladen) oder könnten 24-Bit-Düsenzündwerte sein, die direkt in den Druckkopf über den Testmodus der Druckerzeugungseinheit **99** geladen werden.

[0285] Die Bildzugriffseinheit **9** stellt einen geradlinigen Zugriffsmechanismus auf den Bild-RAM-Speicher **11** dar und funktioniert recht einfach hinsichtlich der 3 Register, wie in Tabelle 16 dargestellt.

Tabelle 16. IAU-Register

Name	Bits	Beschreibung
ImageAddress	21	Adresse zum Lesen oder Schreiben im Bild-RAM-Speicher
Modus	3	0 = Lesen aus ImageAddress in Wert 1 = Schreibe Wert nach ImageAddress.
Wert	6	Bei ImageAddress gespeicherter Wert (falls Modus=lesen) ImageAddress zu speichernder Wert (falls Modus=schreiben)

[0286] Die Struktur der Bildzugriffseinheit ist sehr einfach, wie in Fig. 35 dargestellt.

[0287] Der Zustandsautomat **101** führt einfach das Lesen/Schreiben vom/in den Bild-RAM-Speicher **11** durch, immer dann, wenn die CPU **10** in Modus-Register schreibt.

9. BILDHISTOGRAMMEINHEIT **8**

[0288] Die Bildhistogrammeinheit (Image Histogram Unit; IHU) **8** ist dazu bestimmt, Histogramme der Bilder zu erzeugen, wie sie von der in Abschnitt 3.2.2 beschriebenen Druckbildverarbeitungskette erforderlich sind. Die IHU erzeugt nur Histogramme für Bilder im planaren Format mit Abtastpunkten von jeweils 8 Bit.

[0289] Die Bildhistogrammeinheit **8** wird typischerweise drei Mal pro Ausdruck verwendet. Drei verschiedene Histogramme werden erfasst, eines pro Farbebene. Jedes Mal, wenn ein Histogramm erfasst wird, werden die Ergebnisse analysiert, um die unteren und oberen Grenzwerte, Skalierungsfaktoren usw. zur Verwendung bei dem Rest des Druckprozesses zu bestimmen. Für mehr Informationen, wie das Histogramm verwendet werden soll, sei auf Abschnitt 3.2.2.2 und Abschnitt 3.2.4 verwiesen.

9.1 HISTOGRAMM-RAM-SPEICHER **102**

[0290] Das Histogramm selbst wird in einem RAM-Speicher **102** mit 256 Einträgen gespeichert, wobei jeder Eintrag **20** Bits umfasst. Auf den Histogramm-RAM-Speicher wird lediglich von innerhalb der IHU zugegriffen. Einzelne Einträge werden gelesen oder geschrieben als 20-Bit-Mengen.

9.2 ZUSTANDSAUTOMAT UND REGISTER **103**

[0291] Der Zustandsautomat **103** folgt dem in Abschnitt 3.2.2.1 beschriebenen Pseudocode. Er wird durch die in Tabelle 17 dargestellten Register gesteuert.

Tabelle 17. Register in Bildhistogrammeinheit

Name	Bits	Beschreibung
TotalPixels	20	Anzahl der zu zählenden Pixel (nimmt bis 0 ab)
StartAddress	21	Wo mit dem Zählen zu beginnen ist
PixelsRemaining	20	Wie viele zu zählende Pixel verbleiben
PixelValue	8	Ein Schreiben in dieses Register lädt PixelCount mit dem PixelValue-Eintrag aus dem Histogramm.
PixelCount	20	Die Anzahl der PixelValue-Pixels die in dem momentanen

		Histogramm gezählt wurden. Es ist nach einem Schreiben in PixelValue gültig.
ClearCount	1	Bestimmt, ob der Histogramm-Zähler an dem Beginn des Histogramm-Prozesses geleert wird. Eine 1 führt dazu, dass die Zähler geleert werden, und eine 0 führt dazu, dass die Zähler unverändert bleiben (d. h. das nächste Histogramm fügt zu den existierenden Zählern hinzu).
Go	1	Das Hineinschreiben einer 1 beginnt hier den Histogramm-Prozess. Das Hineinschreiben einer 0 stoppt hier den Histogramm-Prozess. Eine 0 wird hier automatisch in den Zustandsautomaten hineingeschrieben, nachdem TotalPixels bis 0 herunter gezählt ist.

[0292] Die typische Verwendung der Register ist, TotalPixels mit der Gesamtanzahl an Pixeln zu belegen, die in dem Zähler umfasst sind (z. B. 375.000 für rot), StartAddress mit der Adresse der roten Ebene, ClearCount mit 1, und eine 1 in das Go-Register zu schreiben. Sobald die Zählung beendet ist, können die individuellen Werte in dem Histogramm durch Schreiben von 0-255 in PixelValue und Lesen des entsprechenden PixelCount bestimmt werden.

10. DRUCKKOPF-SCHNITTSTELLE **105**

[0293] Die Druckkopf-Schnittstelle (Printhead Interface; PHI) **105** ist die Vorrichtung, mit der der PCP **3** den Memjet-Druckkopf **2** mit den zu druckenden Punkten lädt und den aktuellen Punktdruckprozess steuert. Die PHI ist eine logische Klammer für eine Anzahl von Einheiten, nämlich:

- Eine Memjet-Schnittstelle (Memjet Interface; MJI) **15**, die Daten zu dem Memjet-Druckkopf überträgt und die Düsenzündabfolge während eines Drucks steuert.
- Eine Druckerzeugungseinheit (Print Generator Unit; PGU) **99** ist eine Implementierung der meisten Bestandteile der in Abschnitt 3.2 beschriebenen Druckkette, und stellt eine Vorrichtung zum Erzeugen von Testmustern zur Verfügung. Die PGU nimmt ein aus einem im CFA-Format erfassten Bild erhaltenes planarisiertes lineares RGB aus dem Bild-RAM-Speicher **11** und erzeugt ein 1600 dpi gedithertes CMY-Bild in Echtzeit, wie es von der Memjet-Schnittstelle **15** erfordert wird. Darüber hinaus weist die PGU einen Testmuster-Modus auf, der der CPU **10** ermöglicht, genau zu spezifizieren, welche Düsen während eines Drucks zu zünden sind.

[0294] Die Einheiten innerhalb der PHI werden durch eine Anzahl von Registern gesteuert, die durch die CPU programmiert werden.

[0295] Die interne Struktur der Druckkopf-Schnittstelle ist in Fig. 37 dargestellt.

10.1 MEMJET-SCHNITTSTELLE **15**

[0296] Die Memjet-Schnittstelle (Memjet Interface; MJI) **15** verbindet den PCP mit dem externen Memjet-Druckkopf, indem sie sowohl Daten als auch geeignete Signale bereitstellt, um die Lade- und Zündabfolgen der Düse während eines Ausdrucks zu steuern.

[0297] Die Memjet-Schnittstelle **15** ist einfach ein Zustandsautomat **106** (siehe Fig. 38), der der Druckkopflade- und Zündreihenfolge folgt, die in Abschnitt 2.2 beschrieben wurde, und die Funktionalität des Vorheizzyklus und des Reinigungszyklus umfasst, wie sie in Abschnitt 2.4.1 und Abschnitt 2.4.2 beschrieben wurden.

[0298] Die MJI **15** lädt Daten in den Druckkopf aus einer Auswahl aus 2 Datenquellen:

- Lauter 1 (Einsen). Dies bedeutet, dass alle Düsen während eines nachfolgenden Druckzyklus zünden, und dies ist der Standardmechanismus für das Laden des Druckkopfes für einen Vorheiz- oder Reinigungszyklus.
- Aus dem 24-Bit-Eingang, der in dem Transfer-Register der PGU **99** gehalten wird. Dieses ist das Standardmittel beim Drucken eines Bildes, ob es nun ein erfasstes Foto oder ein Testmuster ist. Der 24-Bit-Wert aus der PGU wird direkt zu dem Druckkopf gesendet und ein 1-Bit-"Advance"-Steuerimpuls wird zu der PGU gesendet. Am Ende jeder Zeile wird ein 1-Bit-"AdvanceLine"-Impuls ebenfalls an die PGU gesendet.

[0299] Die MJI **15** muss gestartet werden, nachdem die PGU **99** den ersten 24-Bit-Transferwert vorbereitet hat. Dies ist deshalb, damit der 24-Bit-Dateneingang für den ersten Transfer an den Druckkopf gültig ist.

[0300] Die MJI ist deshalb direkt mit der Druckerzeugungseinheit **99** und dem externen Druckkopf **2** verbunden. Die Basisstruktur ist in Fig. 38 dargestellt.

10.1.1 Verbindungen zum Druckkopf

[0301] Die MJI **15** weist die folgenden Verbindungen zu dem Druckkopf **2** auf, wobei die Richtung von Eingang und Ausgang bezüglich der MJI **15** sind. Die Namen stimmen mit den Anschlussverbindungen auf dem Druckkopf überein (siehe Abschnitt 2).

Tabelle 18. Druckkopf-Verbindungen

Name	#Anschlüsse	E/A	Beschreibung
Chromapod-Select	4	A	Auswählen, welches Chromapod zündet (0-9)
NozzleSelect	4	A	Auswählen, welche Düse aus dem Sockel feuert (0-9)
AEnable	1	A	Zündimpuls für Phasengruppe A
BEnable	1	A	Zündimpuls für Phasengruppe B
CDataIn[0-7]	8	A	Cyan Ausgabe an cyan Schieberegister der Segmente 0-7
MDataIn[0-7]	8	A	Magenta Eingang für magenta Schieberegister der Segmente 0-7
YDataIn[0-7]	8	A	Gelber Eingang für gelbes Schieberegister der Segmente 0-7
SRClock	1	A	Ein Impuls auf SRClock (ShiftRegisterClock) lädt die aktuellen Werte aus CDataIn [0-7], MDataIn [0-7] und YDataIn [0-7] in die 24 Schieberegister des Druckkopfs.
PTransfer	1	A	Paralleler Transfer von Daten aus den Schieberegistern in die internen NozzleEnable-Bits des Druckkopfs (eins pro Düse)
SenseSeg-Enable	1	A	Ein Impuls auf SenseSegEnable UND-verknüpft mit Daten auf CDataIn[n] wählt die Abfrageleitungen für Segment n aus.
Tsense	1	E	Temperaturabfrage
Vsense	1	E	Spannungsabfrage
Rsense	1	E	Abfrage des spezifischen Widerstands
Wsense	1	E	Abfrage der Breite
GESAMTANZAHL	41		

10.1.2 Zündimpulsdauer

[0302] Die Dauer der Zündimpulse auf den AEnable und BEnable-Leitungen hängt von der Viskosität der Tinte (die abhängig ist von der Temperatur und den Tinteneigenschaften) und der für den Druckkopf verfügbaren Leistungsmenge ab. Der typische Pulsdauerbereich beträgt 1,3 bis 1,8 µs. Die MJI enthält deshalb eine pro-

grammierbare Pulsdauertabelle, die durch Rückmeldung von dem Druckkopf indiziert ist. Die Tabelle der Impulsdauern ermöglicht die Verwendung einer preiswerteren Stromversorgung und hilft dabei, eine genauere Tropfenabgabe aufrecht zu erhalten.

[0303] Die Pulsdauertabelle weist 256 Einträge auf und ist durch die aktuellen Vsense- und Tsense-Einstellungen indiziert. Die oberen 4 Bits der Adresse kommen von Vsense und die unteren 4 Bits der Adresse kommen von Tsense. Jeder Eintrag ist 8 Bit lang und stellt einen Festkommawert in dem Bereich von 0-4 µs dar. Der Prozess des Erzeugens der AEnable- und BEnable-Leitungen ist in Fig. 39 dargestellt.

[0304] Die 256-Byte Tabelle wird vor dem Drucken des Fotos von der CPU **10** geschrieben. Jeder 8 Bit-Impulsdauereintrag in der Tabelle kombiniert:

- Helligkeitseinstellungen
- Viskositätskurve der Tinte (aus dem QA-Chip) **5**
- Rsense
- Wsense
- Tsense
- Vsense

10.1.3 Punktzähler

[0305] Die MJI **15** pflegt einen Zähler der Anzahl von Punkten jeder von dem Druckkopf **2** gezündeten Farbe. Der Punktzähler für jede Farbe ist ein 32-Bit-Wert, der individuell unter der Prozessorsteuerung gelöscht wird. Jeder Punktzähler kann einen maximalen Abdeckungspunktzähler von 69 6-Zoll-Ausdrucken halten, obwohl bei einem typischen Gebrauch der Punktzähler nach jedem Druck gelesen und gelöscht wird.

[0306] Während bei dem ursprünglichen Printcam-Produkt das Verbrauchsmaterial sowohl Papier als auch Tinte enthält, ist es vorstellbar, dass ein anderes Printcam-Modell ein austauschbares Verbrauchsmaterial nur für Tinte aufweist. Das ursprüngliche Printcam-Produkt kann den Millimeterbetrag, der an Papier verbleibt (in dem QA-Chip **5** gespeichert – siehe Abschnitt 5.2) herunter zählen, um zu wissen, ob noch genug Papier verfügbar ist, um das gewünschte Format zu drucken. Es ist ausreichend Tinte für eine volle Deckung des gesamten angelieferten Papiers vorhanden. Bei dem alternativen Printcam-Produkt können die Punktzähler von der CPU **10** verwendet werden, um den QA-Chip **5** zu aktualisieren, um vorherzusagen, wann der Tintenkartusche die Tinte ausgeht. Der Prozessor kennt das Tintenvolumen in der Kartusche jeweils für C, M und Y aus dem QA-Chip **5**. Das Zählen der Anzahl von Tropfen macht den Bedarf nach Tintensensoren überflüssig und verhindert, dass die Tintenkanäle trocken laufen. Ein aktualisierter Tropfenzähler wird nach jedem Druck in den QA-Chip **5** geschrieben. Es wird kein neues Foto gedruckt, sofern nicht genügend Tinte vorhanden ist, und dies ermöglicht es dem Benutzer, die Tinte zu wechseln, ohne ein fehlerhaftes Foto zu erhalten, das erneut gedruckt werden muss.

[0307] Das Layout des Punktzählers für cyan ist in Fig. 40 dargestellt. Die verbleibenden beiden Punktzähler (MDotCount und YDotCount, für magenta bzw. gelb) weisen eine identische Struktur auf.

10.1.4 Register

[0308] Die CPU **10** kommuniziert mit der MJI **15** über einen Registersatz. Die Register erlauben es der CPU, einen Druck zu parametrisieren sowie eine Rückmeldung über den Druckfortschritt zu erhalten.

[0309] Die folgenden Register sind in der MJI enthalten:

Tabelle 19. Memjet-Schnittstellenregister

Registername	**Beschreibung**
Druckparameter	
NumTransfers	Die Anzahl der erforderlichen Transfers, um den Druckkopf zu laden (üblicherweise 800). Dies ist die Anzahl von Impulsen auf SRClock

	und die Anzahl von 24-Bit-Datenwerten, die für eine gegebene Zeile zu transferieren sind.
PulseDuration	Festkommazahl, um die Dauer eines einzelnen Impulses auf den ColorEnable-Leitungen zu bestimmen. Bereich der Dauer = 0 - 6µs
NumLines	Die Anzahl der durchzuführenden Lade-/Druckzyklen.
Überwachen des Drucks	
Status	Das Statusregister der Memjet-Schnittstelle
LinesRemaining	Die Anzahl der verbleibenden zu druckenden Zeilen. Nur gültig, sofern Go = 1. Anfangswert ist NumLines.
TransfersRemaining	Die Anzahl der verbleibenden Transfers, bevor der Druckkopf als für die aktuelle Zeile geladen betrachtet wird. Nur gültig, sofern Go = 1.
SenseSegment	Der 8-Bit-Wert, der auf den cyan-Datenleitungen anzuordnen ist, während eines nachfolgenden Rückmelde-SenseSegSelect-Impulses. Nur 1 der 8 Bits sollte gesetzt sein, entsprechend dem einen der 8 Segmente.
SetAllNozzles	Falls nicht 0 ist der auf den Druckkopf während des LoadDots-Prozesses geschriebene 24-Bit-Wert lauter Einsen (1), so dass alle Düsen während des nachfolgenden PrintDots-Prozesses gezündet werden. Dieser wird während der Vorheiz- und Reinigungszyklen verwendet. Falls 0, kommt der in den Druckkopf geschriebene 24-Bit-Wert aus der Druckerzeugungseinheit. Dies ist der Fall während des tatsächlichen Druckens des Fotos und beliebiger Testbilder.
Aktionen	
Reset	Ein Schreiben auf dieses Register setzt die MJI zurück, stoppt irgendwelche Lade- oder Druckprozesse und lädt alle Register mit 0.
SenseSegSelect	Ein Schreiben auf dieses Register mit einem beliebigen Wert löscht das Rückmelde-Bit des Statusregisters und sendet einen Impuls auf der SenseSegSelect-Leitung, sofern die LoadingDots und PrintingDots-Statusbits alle 0 sind. Wenn eines der Statusbits gesetzt ist, wird das Rückmelde-Bit gelöscht und nichts weiter wird durchgeführt. Sobald die verschiedenen Abfrageleitungen getestet wurden, werden die Werte in die Tsense-, Vsense-, Rsense- und Wsense-Register platziert und anschließend wird das Rückmelde-Bit des Statusregisters gesetzt. Die Rückmeldung geht während beliebiger nachfolgender Druckvorgänge weiter.
Go	Ein Schreiben von 1 in dieses Bit beginnt die LoadDots-/PrintDots-Zyklen. Eine Gesamtzahl von NumLines Zeilen wird gedruckt, die jeweils NumTransfers 24-Bit-Transfers enthalten. Sobald jede Zeile gedruckt wird, wird LinesRemaining um 1 herunter gezählt, TransfersRemaining wird erneut mit NumTransfers geladen. Das Statusregister enthält Druckstatusinformationen. Bei Fertigstellung von NumLines hält der Lade-/Druckprozess an und das Go-Bit wird

	gelöscht. Während des finalen Druckzyklus wird nichts in den Druckkopf hinein geladen. Ein Schreiben von 0 auf dieses Bit hält den Druckprozess an, löscht jedoch nicht die anderen Register.
ClearCounts	Ein Schreiben auf dieses Register löscht die CDotCount, MDotCount und YDotCount Register, sofern die Bits 0, 1 bzw. 2 gesetzt sind. Folglich hat ein Schreiben von 0 keine Wirkung.
Rückmeldung	
Tsense	Nur-Lesen-Rückmeldung von Tsense aus dem letzten SenseSegSelect-Impuls, der zu dem Segment SenseSegment gesendet wurde. Ist nur gültig, wenn das FeedbackValid-Bit des Statusregisters gesetzt ist.
Vsense	Nur-Lesen-Rückmeldung von Vsense aus dem letzten SenseSegSelect-Impuls, der zu dem Segment SenseSegment gesendet wurde. Ist nur gültig, wenn das FeedbackValid-Bit des Statusregisters gesetzt ist.
Rsense	Nur-Lesen-Rückmeldung von Rsense aus dem letzten SenseSegSelect-Impuls, der zu dem Segment SenseSegment gesendet wurde. Ist nur gültig, wenn das FeedbackValid-Bit des Statusregisters gesetzt ist.
Wsense	Nur-Lesen-Rückmeldung von Wsense aus dem letzten SenseSegSelect-Impuls, der zu dem Segment SenseSegment gesendet wurde. Ist nur gültig, wenn das FeedbackValid-Bit des Statusregisters gesetzt ist.
CDotCount	Nur Lesen 32-Bit-Zähler, der an den Druckkopf gesendeten cyan Punkte.
MDotCount	Nur Lesen 32-Bit-Zähler, der an den Druckkopf gesendeten magenta Punkte.
YDotCount	Nur Lesen 32-Bit-Zähler, der an den Druckkopf gesendeten gelb Punkte.

[0310] Das Statusregister der MJI ist ein 16-Bit-Register mit den nachfolgenden Bit-Bedeutungen:

Tabelle 20. MJI-Statusregister

Name	**Bits**	**Beschreibung**
LoadingDots	1	Wenn gesetzt, lädt die MJI momentan Punkte, wobei die Anzahl der Punkte, die noch transferiert werden müssen, in TransfersRemaining steht. Wenn nicht gesetzt, dann lädt die MJI aktuell keine Punkte.
PrintingDots	1	Wenn gesetzt, druckt die MJI momentan Punkte.

		Wenn nicht gesetzt, druckt die MJI aktuelle keine Punkte.
PrintingA	1	Dieses Bit ist gesetzt, während ein Impuls auf der AEnable-Leitung vorhanden ist.
PrintingB	1	Dieses Bit ist gesetzt, während ein Impuls auf der BEnable-Leitung vorhanden ist.
FeedbackValid	1	Dieses Bit ist gesetzt, während die Rückmeldewerte Tsense, Vsense, Rsense und Wsense gültig sind.
Reserved	3	–
PrintingChromapod	4	Dies beinhaltet das aktuelle Chromapod, das gezündet wird, während das PrintingDots-Statusbit gesetzt ist.
PrintingNozzles	4	Dies beinhaltet die momentane Düse, die gezündet wird, während das PrintingDots-Statusbit gesetzt ist.

10.1.5 Vorheiz- und Reinigungszyklen

[0311] Die Reinigungs- und Vorheizzyklen werden einfach durch Setzen der entsprechenden Register erreicht:

- SetAllNozzles = 1
- Setzen des PulseDuration-Registers auf entweder eine niedrige Dauer (in dem Fall des Vorheizmodus) oder auf eine geeignete Tropfenauswurfdauer für den Reinigungsmodus.
- Setzen von NumLines auf die Anzahl von Malen, die die Düsen gezündet werden sollen.
- Setzen des Go-Bits und anschließendes Warten, dass das Go-Bit gelöscht wird, wenn die Print-Zyklen geschlossen sind.

10.2 DRUCKERZEUGUNGSEINHEIT **99**

[0312] Die Druckerzeugungseinheit (Print Generator Unit; PGU) **99** ist eine Implementierung des Großteils der Druckkette, die in Abschnitt 3.2 beschrieben wurde, und stellt auch eine Einrichtung zum Erzeugen von Testmustern zur Verfügung.

[0313] Aus einfachster Sicht stellt die PGU die Schnittstelle zwischen dem Bild-RAM-Speicher **11** und der Memjet-Schnittstelle **15** dar, wie in Fig. 41 dargestellt. Die PGU nimmt ein planarisiertes lineares RGB, das aus einem im CFA-Format erfassten Bild aus dem Bild-RAM-Speicher erhalten wurde, und erzeugt ein 1600 dpi gedithertes CMY-Bild in Echtzeit, wie es von der Memjet-Schnittstelle erforderlich ist. Darüber hinaus weist die PGU **99** einen Testmuster-Modus auf, der der CPU **10** ermöglicht, genau zu bestimmen, welche Düsen während eines Drucks gezündet werden. Die MJI **15** versorgt die PGU **99** mit einem Advance-Impuls, sobald die 24 Bits verwendet wurden, und mit einem AdvanceLine-Impuls am Ende der Zeile.

[0314] Die PGU **99** weist 2 Bildverarbeitungsketten auf. Die erste, der Testmuster-Modus, liest einfach Daten direkt aus dem Bild-RAM-Speicher **11** und formatiert sie in einem Zwischenspeicher fertig zur Ausgabe an die MJI. Die zweite enthält den Großteil der Druckketten-Funktionen (siehe Abschnitt 3.2). Die in Fig. 18 dargestellte Druckkette enthält die Funktionen:

- Erfassen von Kenngrößen **60**
- Drehen des Bildes **61**
- Weißabgleich **62**
- Bereichserweiterung **63**
- Neuabtasten/Resample **64**
- Schärfen **65**
- Umwandeln zu CMY **66**
- Hochinterpolieren **67**
- Halbtönen **68**
- Reformatieren für Drucker **69**

[0315] Die PGU **99** enthält alle diese Funktionen mit der Ausnahme des Erfassens von Kenngrößen **60**. Um

den Schritt Erfassen von Kenngrößen durchzuführen, ruft die CPU **10** die Bild-Histogrammeinheit **8** drei Mal auf (einmal pro Farbkanal) und wendet einige einfache Algorithmen an. Der Rest der Funktionen liegt in dem Bereich der PGU **99** aus Gründen der Exaktheit und der Geschwindigkeit: Exaktheit, da es zu viel Speicher erfordern würde, das gesamte Bild mit einer hohen Genauigkeit zu beinhalten, und Geschwindigkeit, weil eine einfache CPU **10** nicht mit den Hochgeschwindigkeits-Echtzeit-Anforderungen des Memjet-Druckkopfes **2** mithalten kann.

[0316] Die PGU **99** nimmt eine Vielzahl von Parametern als Input einschließlich der RGB-zu-CMY-Umwandlungstabellen, Konstanten zur Durchführung des Weißabgleichs und der Bereichserweiterung, Skalierungsfaktoren für das Resampling und Bildzugriffsparameter, die eine Drehung ermöglichen.

[0317] Die beiden Prozessketten sind in Fig. 42 ersichtlich. Die direkteste Kette verläuft von dem Bild-RAM-Speicher **11** in den Zwischenspeicher 5 über den Testmuster-Zugriffsprozess **110**. Die andere Kette besteht aus 5 Prozessen, die alle parallel laufen. Der erste Prozess **111** führt die Bilddrehung, den Weißabgleich und die Bereichserweiterung durch. Der zweite Prozess **112** führt das Resampling durch. Der dritte Prozess **65** führt die Schärfung, der vierte Prozess **66** die Farbumwandlung durch. Der letzte Prozess **113** führt die Hochskalierung, das Halbtönen und das Reformatieren für den Drucker durch. Die Prozesse sind über Zwischenspeicher miteinander verbunden, nur wenige Bytes zwischen einigen Prozessen und wenige KBytes für andere.

[0318] Wir wenden uns diesen Prozessen und Zwischenspeichern in einer hauptsächlich umgekehrten Reihenfolge zu, da der Zeitablauf für den Druckkopf den gesamten Prozess antreibt. Die Zeitabläufe für bestimmte Prozesse und Zwischenspeicher der Größenanforderungen werden dann klarer. In der Zusammenfassung sind die Zwischenspeichergrößen jedoch in Tabelle 21 dargestellt.

Tabelle 21. Zwischenspeichergrößen für Druckerzeugungseinheit

Zwischenspeicher	**Größe (Bytes)**	**Zusammensetzung des Zwischenspeichers**
Zwischenspeicher 1	188	Roter Zwischenspeicher = 6 Zeilen aus 6 Einträgen @ jeweils 10 Bits = 45 Bytes Blauer Zwischenspeicher = 6 Zeilen aus 6 Einträgen @ jeweils 10 Bits = 45 Bytes Grüner Zwischenspeicher = 13 Zeilen aus 6 Einträgen @ jeweils 10 Bits = 97,5 Bytes
Zwischenspeicher 2	24	6 x 4 RAM-Speicher 3 Zeilen aus 4 Einträgen von L @ jeweils 8 Bits = 12 Bytes 3 Farben x 4 Einträge @ jeweils 8 Bits = 12 Bytes.
Zwischenspeicher 3	3	3 Farben (RGB) @ jeweils 8 Bits
Zwischenspeicher 4	23.040	3 Farben (CMY) x 6 Zeilen x 1280 Pixel mit gleichmäßigem Farbverlauf @ jeweils 8 Bits
Zwischenspeicher 5	9	3 x 24 Bits
GESAMT	23.264	

[0319] Abgesehen von einer Anzahl von Registern weisen einige Prozesse erhebliche Nachschlagetabellen oder Speicherbestandteile auf. Diese sind in Tabelle 22 zusammen gefasst.

Tabelle 22. Speicheranforderungen innerhalb der PGU-Prozesse

Einheit	Größe (Bytes)	Zusammensetzung der Anforderungen
Drehen/Weißabgleich/Bereichs-erweiterung	0	
Resample / in L umwandeln	1.152	3 Kerne, jeweils 64 x 4 x 12 Bits
Schärfen	0	
Umwandeln in CMY	14.739	3 Umwandlungstabellen, jeweils 17 x 17 x 17 x 8-Bits.
Hochinterpolieren /Halbtönen / Reformatieren	2.500	Dither-Zelle 50 x 50 x 8 Bits
Testmusterzugriff	0	
GESAMT	18.391	

10.2.1 Testmusterzugriff

[0320] Der Testmusterzugriffsprozess **110** ist die Einrichtung, durch die Testmuster erzeugt werden. Unter normalen Benutzerumständen wird dieser Prozess nicht benutzt werden. Er dient hauptsächlich zu diagnostischen Zwecken.

[0321] Der Testmusterzugriff **110** liest den Bild-RAM-Speicher **11** und übergibt die 8 Bit-Werte direkt an den Zwischenspeicher 5 **118** zur Ausgabe an die Memjet-Schnittstelle. Er modifiziert die 8 Bit-Werte nicht auf irgendeine Weise. Die Daten in dem Bild-RAM-Speicher **11** würden durch die CPU **10** unter Verwendung der Bildzugriffseinheit **9** erzeugt.

[0322] Die von dem Bild-RAM-Speicher **11** gelesenen Daten werden in einer sehr einfachen Bildumlauf-Weise eingelesen. Zwei Register werden verwendet, um die Testdaten zu beschreiben: die Startadresse des ersten Bytes und die Anzahl von Bytes. Wenn das Ende der Daten erreicht ist, dann werden die Daten erneut vom Anfang an eingelesen.

[0323] Der Aufbau der Testmusterzugriffseinheit **110** ist in Fig. 43 dargestellt.

[0324] Wie aus Fig. 43 ersichtlich ist, ist die Testmusterzugriffseinheit **110** ein wenig mehr als ein Adressengenerator **119**. Sobald gestartet, und mit jedem AdvanceLine-Signal liest der Generator 3 Bytes, erzeugt einen TransferWriteEnable-Impuls, liest die nächsten 3 Bytes ein und wartet anschließend auf einen Advance-Impuls. Bei dem Advance-Impuls wird der TransferWriteEnable-Impuls gegeben, die nächsten 3 Bytes werden gelesen und das Warten findet erneut statt. Dies geht bis zum AdvanceLine-Impuls so weiter, bei welchem der Prozess erneut von der aktuellen Adresse beginnt.

[0325] Hinsichtlich des Lesens von 3 Bytes liest der Adressengenerator **119** einfach drei 8-Bit-Werte aus dem Bild-RAM-Speicher **11** und schreibt sie in den Zwischenspeicher 5 **118**. Der erste 8-Bit-Wert wird auf die 8-Bit-Adresse 0 des Zwischenspeichers 5 geschrieben, der nächste wird auf die 8-Bit-Adresse 1 des Zwischenspeichers 5 geschrieben und der dritte wird auf die 8-Bit-Adresse 2 des Zwischenspeichers 5 geschrieben. Der Adressengenerator **119** wartet anschließend auf einen Advance-Impuls, bevor er damit beginnt, dasselbe erneut durchzuführen.

[0326] Die für den Bild-RAM-Speicher **11** erzeugten Adressen basieren auf einer Start-Adresse und einem Byte-Zähler wie in Tabelle 23 dargestellt.

Tabelle 23. Testmusterzugriffsregister

Registername	Beschreibung
TestModeEnabled	Wenn 1, ist TestMode eingeschaltet. Wenn 0, ist TestMode nicht eingeschaltet.
DataStart	Startadresse der Testdaten im Bild-RAM-Speicher
DataLength	Anzahl von 3 Bytes in Testdaten

[0327] Der nachfolgende Pseudocode veranschaulicht die Adressengenerierung. Die Impulse AdvanceLine und Advance sind nicht dargestellt.

```
Do Forever
   Adr = DataStart
   Remaining = DataLength
   Read Adr into Buffer 5 (0), Adr=Adr+1
   Read Adr into Buffer 5 (1), Adr=Adr+1
   Read Adr into Buffer 5 (2), Adr=Adr+1
   Remaining = Remaining - 1
   if  (Remaining = 0)
      Remaining = DataLength
EndDo
```

[0328] Es liegt in der Verantwortung der CPU **10**, sicherzustellen, dass die Daten für den Druckkopf **2** aussagekräftig sind. Byte 0 sind die Düsenzünd-Daten für die 8 Segmente cyan (Bit 0 = Segment 0 usw.). Byte 1 ist dasselbe für magenta und Byte 2 für gelb. Abwechselnde Sätze von 24 Bits sind für die ungeraden/geraden Pixel, die durch eine horizontale Punktlinie getrennt sind.

10.2.2 Zwischenspeicher 5 **118**

[0329] Der Zwischenspeicher 5 **118** beinhaltet die erzeugten Punkte aus dem gesamten Druckerzeugungsprozess. Der Zwischenspeicher 5 besteht aus einem 24 Bit-Schieberegister, um Punkte zu beinhalten, die einer nach dem anderen von der UHRU **113** (Hochinterpolations-Halbton- und Reformatier-Einheit) erzeugt wurden, 3 8-Bit-Register um die Daten zu beinhalten, die von der TPAU (Testmusterzugriffseinheit) erzeugt wurden, und ein 24-Bit-Register, das als Zwischenspeicher für den Datentransfer zu der MJI (Memjet-Schnittstelle) verwendet wird. Der Advance-Impuls von der MJI lädt die 24-Bit-Transfer-Register mit sämtlichen 24 Bits, entweder aus den 3 8-Bit-Registern oder einem einzelnen 24-Bit-Schieberegister.

[0330] Der Zwischenspeicher 5 agiert deshalb als ein doppelter Zwischenspeichermechanismus für die erzeugten Punkte und weist eine wie in Fig. 44 dargestellte Struktur auf.

10.2.3 Zwischenspeicher 4 **117**

[0331] Der Zwischenspeicher 4 **117** beinhaltet das Bild in gleichmäßigem Tonverlauf mit berechneter CMY-Zwischenauflösung (1280er Auflösung). Der Zwischenspeicher 4 wird durch den Farbumwandlungsprozess **66** erzeugt und es wird auf ihn durch den Hochinterpolier-, Halbton- und Reformatierprozess **113** zugegriffen, um die Ausgabepunkte für den Drucker zu erzeugen.

[0332] Die Größe des Zwischenspeichers mit gleichmäßigem Tonverlauf (Contone Buffer) hängt von dem physikalischen Abstand zwischen den Düsen auf dem Druckkopf ab. Da die Punkte für eine Farbe für eine phy-

sikalische Zeile erzeugt werden, werden Punkte für eine andere Farbe auf einer anderen Zeile erzeugt. Der Netto-Effekt ist, dass 6 unterschiedliche physikalische Zeilen zur gleichen Zeit von dem Drucker gedruckt werden – ungerade und gerade Punkte von unterschiedlichen Ausgabezeilen und unterschiedliche Zeilen pro Farbe. Dieses Konzept wird in Abschnitt 2.1.1 erläutert, und die Abstände sind dort definiert.

[0333] Das praktische Ergebnis ist, dass es einen vorgegebenen Abstand bei den Punkten mit hoher Auflösung von den geraden cyan Punkten über die magenta Punkte zu den ungeraden gelben Punkten gibt. Um die Generierung von RGB zu minimieren und damit von CMY, werden die Pixel mit gleichmäßigem Farbverlauf in mittlerer Auflösung, die diese Punkte in hoher Auflösung erzeugen, im Zwischenspeicher 4 zwischengespeichert.

[0334] Da das Verhältnis von Zeilen in mittlerer Auflösung zu Zeilen in hoher Auflösung 1:5 ist, wird jede Zeile in mittlerer Auflösung 5 Mal abgetastet in jeder (Richtungs-) Ausdehnung. Für die Zwecke von Zwischenspeicherzeilen berücksichtigen wir nur 1 Ausdehnung, deshalb betrachten wir nur 5 Punkt-Zeilen, die von einer einzelnen Pixelzeile kommen. Der Abstand zwischen Düsen unterschiedlicher Farbe beträgt 4-8 Punkte (je nach Memjet-Parameter). Wir nehmen deshalb 8 an, woraus sich ein Trennabstand von 16 Punkten oder 17 Punkten einschließlich Abstand ergibt. Das Worst-Case-Szenario ist, dass die 17-Punkt-Zeilen die letzte Punktzeile einer gegebenen Pixelzeile umfassen. Dies impliziert 5 Pixelzeilen, wobei Punktzeilen bei 1, 5, 5, 5, 1, erzeugt werden, und ermöglicht eine Erhöhung der Düsentrennung auf 10.

[0335] Um sicherzustellen, dass das Schreiben des Generierungsprozesses mit gleichmäßigem Tonverlauf in den Zwischenspeicher nicht das Lesen des PunktGenerierungsprozesses aus dem Zwischenspeicher stört, fügen wir eine zusätzliche Zeile in mittlerer Auflösung pro Farbe hinzu, zu einer Gesamtanzahl von 6 Zeilen pro Farbe.

[0336] Der Zwischenspeicher mit gleichmäßigem Farbverlauf beträgt deshalb 3 Farben zu 6 Zeilen, wobei jede Zeile 1280 8-Bit-Werte mit gleichmäßigem Farbverlauf aufweist. Der gesamte erforderliche Speicher ist 3 × 6 × 1280 = 23.040 Bytes (22,5 KB). Der Speicher erfordert lediglich ein einziges 8-Bit-Lesen pro Zyklus und ein einzelnes 8-Bit-Schreiben alle 25 Zyklen (jedes Pixel mit gleichmäßigem Farbverlauf wird 25 Mal gelesen). Die Struktur des Zwischenspeichers 4 ist in **Fig. 45** dargestellt.

[0337] Der Zwischenspeicher 4 kann als Einzelzyklus-Doppelzugriffs-(Lesen und Schreiben)-RAM-Speicher implementiert werden, der bei der nominalen Geschwindigkeit des Druckkopf-PunktGenerierungsprozesses läuft, oder kann als RAM-Speicher implementiert werden, der 4 % schneller läuft mit lediglich einem einzigen Lese- oder Schreibzugriff pro Zyklus.

[0338] Zwischenspeicher 4 wird auf weiß (lauter Nullen) gesetzt, bevor der Druckprozess beginnt.

10.2.4 Hochinterpolieren, Halbtönen und Reformatieren für Drucker

[0339] Obwohl die Aufgaben des Hochinterpolierens, Halbtönens und Reformatierens für den Drucker **113** als getrennte Aufgaben in Abschnitt 3.2.8, Abschnitt 3.2.9 bzw. Abschnitt 3.2.10 definiert sind, werden sie als ein einziger Prozess in der Hardware-Implementierung des PCP **3** implementiert.

[0340] Der Eingang zu der Hochinterpolier-, Halbtön- und Reformatier-Einheit (Up-Interpolate, Halftone, and Reformate Unit; UHRU) **113** ist der Zwischenspeicher mit gleichmäßigem Farbverlauf (Zwischenspeicher 4) **117**, der das vorberechnete CMY-Bild in 1280er Auflösung (Zwischenauflösung) enthält. Die Ausgabe ist ein Satz von 24-Bit-Werten in der richtigen Reihenfolge, um an die Memjet-Schnittstelle **15** zur nachfolgenden Ausgabe an den Druckkopf über den Zwischenspeicher 5 **118** gesendet zu werden. Die 24 Ausgabebits werden 1-Bit-weise erzeugt und an das 24-Bit-Schieberegister in dem Zwischenspeicher 5 **118** gesendet.

[0341] Die Steuerung dieses Prozesses geschieht von den Advance- und AdvanceLine-Signalen von der MJI **15**. Wenn die UHRU **113** hochfährt und nach jedem AdvanceLine-Impuls werden 24 Bits erzeugt und werden in das 24-Bit-Schieberegister des Zwischenspeichers 5 durch ein ShiftWriteEnable-Signal getaktet. Nachdem das 24. Bit eingetaktet wurde, wird ein TransferWriteEnable-Impuls gegeben und die nächsten 24 Bits werden erzeugt. Danach wartet die UHRU **113** auf den Advance-Impuls von der MJI. Wenn der Advance-Impuls ankommt, wird der TransferWriteEnable-Impuls an den Zwischenspeicher 5 **118** gegeben und die nächsten 24 Bits werden berechnet, bevor wieder gewartet wird. In der Praxis tritt, sobald der erste Advance-Impuls gegeben ist, Synchronisation auf und zukünftige Advance-Impulse werden danach alle 24 Zyklen auftreten.

[0342] Der Hochinterpolier-, Halbtön- und Reformatier-Prozess kann in Fig. 46 betrachtet werden.

[0343] Die Halbtön-Aufgabe wird durch den einfachen 8-Bit vorzeichenlosen Komparator **120** durchgeführt. Die beiden Eingänge in den Komparator kommen von der versetzten Dither-Zelle **121** und dem Zwischenspeicher 4 **117**. Die Reihenfolge, in der diese Werte dem vorzeichenlosen Komparator **120** dargeboten werden, wird durch den Adressengenerator-Zustandsautomaten **122** bestimmt, der sicherstellt, dass die Adressen in das Bild mit 1280er Auflösung mit der Segment-orientierten Reihenfolge übereinstimmen, die für den Druckkopf erforderlich ist. Der Adressengenerator-Zustandsautomat **122** führt deshalb die Hochinterpolier- und Reformatier-für-Drucker-Aufgaben durch. Anders als einfach auf eine gesamte Zeile zu einem Zeitpunkt bei hoher Auflösung zuzugreifen und anschließend die Zeile entsprechend der Druckernachschlageanforderungen (wie in Abschnitt 3.2.10 beschrieben) zu reformatieren, wird das Reformatieren durch das geeignete Adressieren des Zwischenspeichers mit gleichmäßigem Tonverlauf (Zwischenspeicher 4) **117** erzielt und stellt sicher, dass der Komparator **120** den korrekten Nachschlagewert aus der Dither-Zelle **121** verwendet, um mit den versetzten Adressen übereinzustimmen.

[0344] Die Halbtön-Aufgabe ist identisch mit der in Abschnitt 3.2.9 beschriebenen Aufgabe. Da jedoch die Punktausgaben in der richtigen Reihenfolge für den Druckkopf erzeugt werden, wird die Größe der Dither-Zelle **121** so gewählt, dass sie sich exakt durch 800 teilen lässt. Folglich wird eine vorgegebene Position in der Dither-Zelle für ein Segment dieselbe sein wie für die verbleibenden 7 Segmente. Eine 50 × 50-Dither-Zelle bietet ein zufriedenstellendes Ergebnis. Wie in Abschnitt 3.2.9 beschrieben, kann dieselbe Position in der Dither-Zelle für unterschiedliche Farben verwendet werden aufgrund der Tatsache, dass unterschiedliche Zeilen gleichzeitig für jede der Farben erzeugt werden. Die Adressierung für die Dither-Zelle ist deshalb relativ einfach. Man beginnt bei einer bestimmten Zeile in der versetzten Dither-Zelle (z. B. Zeile 0). Der erste Dither-Zelleneintrag, der verwendet wird, ist Eintrag 0. Wir verwenden diesen Eintrag 24 Mal (24 Zyklen), um die 3 Farben für alle 8 Segmente zu erzeugen, und gehen anschließend weiter zu Eintrag 1 der Zeile 0. Nach Eintrag 49 kehren wir zurück zu Eintrag 0. Dies geht für alle 19.200 Zyklen weiter, um alle 19.200 Punkte zu erzeugen. Die Halbtön-Einheit hält dann an und wartet auf den AdvanceLine-Impuls, der dazu führt, dass der Adressengenerator zur nächsten Zelle in der Dither-Zelle weiter geht.

[0345] Die versetzte Dither-Zelle **121** wird so genannt, da sie sich von einer regulären Dither-Zelle dadurch unterscheidet, dass sie die ungeraden und geraden Zeilen versetzt aufweist. Dies tritt deshalb auf, da wir ungerade und gerade Pixel (beginnend vom Pixel 0) auf unterschiedlichen Zeilen erzeugen, und erspart dem Adressengenerator **122**, dass er zur nächsten Zeile und wieder zurück bei abwechselnden Sätzen von 24 Pixel voranschreiten muss. Fig. 25 zeigt eine einfache Dither-Zelle **93**, und wie man sie auf eine versetzte Dither-Zelle **121** derselben Größe abbildet. Man bemerke, dass zur Bestimmung der "Ungeradheit" einer gegebenen Position wir die Pixel in einer vorgegebenen Zeile 0, 1, 2 usw. nummerieren.

[0346] Der 8-Bit-Wert aus dem Zwischenspeicher 4 **117** wird (vorzeichenlos) verglichen mit dem 8-Bit-Wert aus der versetzten Dither-Zelle **121**. Wenn der Pixel-Wert im Zwischenspeicher 4 größer oder gleich dem Dither-Zellen-Wert ist, wird ein "1"-Bit an das Schieberegister des Zwischenspeichers 5 **118** ausgegeben. Andernfalls wird ein "0"-Bit an das Schieberegister des Zwischenspeichers 5 ausgegeben.

[0347] Um die 19.200 Pixel mit gleichmäßigem Farbverlauf zu halbtönen, müssen die 19.200 Pixel mit gleichmäßigem Farbverlauf eingelesen werden. Die Adressengeneratoreinheit **122** führt diese Aufgabe durch, indem sie die Adressen in den Zwischenspeicher 4 **117** erzeugt, wodurch sie wirksam die Hochinterpolier-Aufgabe implementiert. Die Adressengenerierung zum Lesen des Zwischenspeichers 4 ist etwas komplizierter als die Adressengenerierung für die Dither-Zelle, jedoch nicht allzu sehr.

[0348] Der Adressengenerator zum Lesen des Zwischenspeichers 4 beginnt nur, sobald die erste Zeile des Zwischenspeichers 4 geschrieben worden ist. Die verbleibenden Zeilen des Zwischenspeichers 4 sind 0, deshalb werden sie effektiv weiß sein (keine gedruckten Punkte).

[0349] Jede der 6 wirksamen Ausgabezeilen weist ein Register mit einem ganzzahligen und einem Bruchbestandteil auf. Der ganzzahlige Anteil des Registers wird verwendet, um auszuwählen, welche Zwischenspeicher-Zeile gelesen wird, um die Farbe für die ungeraden und geraden Pixel dieser bestimmten Farbe wirksam hochzuinterpolieren. 3 Pixelzähler werden verwendet, um die aktuelle Position innerhalb des Segments 0 aufrecht zu erhalten, und ein einzelner temporärer Zähler P_ADR (Pixel Adresse) wird verwendet, um in die restlichen 7 Segmente zu versetzen.

[0350] Zusammenfassend erfordert die Adressengenerierung zum Lesen des Zwischenspeichers 4 also die

nachfolgenden Register, wie in Tabelle 24 dargestellt.

Tabelle 24. Erforderliche Register zum Lesen des Zwischenspeichers 4

Registername	Größe
CyanEven	6 Bits (3:3)
CyanOdd	6 Bits (3:3)
MagentaEven	6 Bits (3:3)
MagentaOdd	6 Bits (3:3)
YellowEven	6 Bits (3:3)
YellowOdd	6 Bits (3:3)
Cyan_P_ADR	14 Bits (11:3)
Mangenta_P_ADR	14 Bits (11:3)
Yellow_P_ADR	14 Bits (11:3)
P_ADR	11 Bits (beinhaltet nur den ganzzahligen Anteil von X_P_ADR)

[0351] Die Anfangswerte für die 6 Zwischenspeicherzeilen-Register ist der physikalische Punktabstand zwischen den Düsen (man erinnere sich, dass der Bruchbestandteil effektiv ein Teiler durch 5 ist). Zum Beispiel wären, wenn die ungeraden und geraden Ausgabepunkte einer Farbe durch einen Abstand von 1 Punkt voneinander getrennt sind, und wenn die Düsen einer Farbe von den Düsen der nächsten Farbe um 8 Punkte getrennt sind, die Anfangswerte wie in Spalte "erste Zeile" in Tabelle 25. Sobald jeder Satz von 19.200 Punkten erzeugt worden ist, muss jeder dieser Zähler um einen Bruchbestandteil erhöht werden, wodurch der Tatsache Rechnung getragen wird, dass wir jedes Pixel 5 Mal in vertikaler Ausdehnung abtasten. Die sich ergebenden Werte sind dann so, wie in der Spalte "zweite Zeile" in Tabelle 25 dargestellt. Es ist anzumerken, dass 5:4 + 1 = 0:0, da lediglich 6 Zwischenspeicher-Zeilen vorhanden sind.

Tabelle 25. Beispielhaftes Anfangssetup und Werte für "zweite Zeile" für die 6 Zwischenspeicher-Zeilenregister

Name	Berechnung	Erste Zeile		Zweite Zeile	
		Wert	Zwischen-speicher	Wert	Zwischen-speicher
CyanEven	Anfangsposition	0:0	0	0:1	0
CyanOdd	CyanEven+0:1	0:1	0	0:2	0
MagentaEven	CyanOdd+1:3(8)	1:4	1	2:0	2
MagentaOdd	MagentaEven+0:1	2:0	2	2:1	2
YellowEven	MagentaOdd+1:3(8)	3:3	3	3:4	3
YellowOdd	YellowEven+0:1	3:4	3	4:0	4

[0352] Die 6 Zwischenspeicher-Zeilenregister bestimmen anschließend, welche der Zwischenspeicherzeile für die ungeraden oder geraden Pixel einer gegebenen Farbe gelesen werden müssen. Um zu bestimmen, welche der 1280 Pixel in mittlerer Auflösung aus der bestimmten Zeile des Zwischenspeichers 4 gelesen werden, verwenden wir 3 Pixeladressenzähler, einen für jede Farbe, und einen einzelnen temporären Zähler (P_ADR), der verwendet wird, in jedes Segment zu indizieren. Jedes Segment ist von dem nächsten durch 800 Punkte getrennt. Bei Pixeln mittlerer Auflösung ist dieser Abstand 160. Da 800 ohne Rest durch 5 teilbar ist, muss man nur den ganzzahligen Anteil der 3 Pixeladresszähler verwenden. Wir erzeugen die 8 Adressen für

gerade cyan Pixel, anschließend die 8 Adressen für die geraden in magenta und schließlich die 8 Adressen für die geraden gelben. Anschließend tun wir dasselbe für die ungeraden cyan, magenta und gelben Pixel. Dieser Prozess aus 2 Sätzen von 24 Bits – 24 gerade, dann 24 ungerade, wird 400 Mal durchgeführt. Dann können wir den Pixeladresszähler (X_P_ADR) auf 0 zurücksetzen und die 6 Zwischenspeicherzeilenregister fortschreiten. Alle 5 Zeilenfortschritte ist die nächste Zwischenspeicherzeile nun frei und bereit für die Aktualisierung (durch den Umwandeln-zu-CMY-Prozess). Tabelle 26 listet die Schritte in einer einfachen Form auf.

Tabelle 26. Adressengenerierung für das Lesen des Zwischenspeichers 4

#	Adresse	Berechnung	Kommentar
		P_ADR= Cyan_P_ADR Cyan_P_ADR+=1 (mod5)	Adresse generieren für gerades Pixel im cyan Segment 0 und weitergehen zum nächsten Pixel für cyan
1	CyanEven:P_ADR	P_ADR+=160	Weiter zu Segment 1 (cyan)
2	CyanEven:P_ADR	P_ADR+=160	Weiter zu Segment 2 (cyan)
3	CyanEven:P_ADR	P_ADR+=160	Weiter zu Segment 3 (cyan)
4	CyanEven:P_ADR	P_ADR+=160	Weiter zu Segment 4 (cyan)
5	CyanEven:P_ADR	P_ADR+=160	Weiter zu Segment 5 (cyan)
6	CyanEven:P_ADR	P_ADR+=160	Weiter zu Segment 6 (cyan)
7	CyanEven:P_ADR	P_ADR+=160	Weiter zu Segment 7 (cyan)
8	CyanEven:P_ADR	P_ADR= Magenta_P_ADR Magenta_P_ADR+=1 (mod5)	Adresse generieren für gerades Pixel im magenta Segment 0 und weitergehen zum nächsten Pixel für magenta
9	MagentaEven:P_ADR	P_ADR+=160	Weiter zu Segment 1 (magenta)
10	MagentaEven:P_ADR	P_ADR+=160	Weiter zu Segment 2 (magenta)
11	MagentaEven:P_ADR	P_ADR+=160	Weiter zu Segment 3 (magenta)
12	MagentaEven:P_ADR	P_ADR+=160	Weiter zu Segment 4 (magenta)
13	MagentaEven:P_ADR	P_ADR+=160	Weiter zu Segment 5 (magenta)
14	MagentaEven:P_ADR	P_ADR+=160	Weiter zu Segment 6 (magenta)
15	MagentaEven:P_ADR	P_ADR+=160	Weiter zu Segment 7 (magenta)
16	MagentaEven:P_ADR	P_ADR= Yellow_P_ADR Yellow_P_ADR+=1 (mod5)	Adresse generieren für gerades Pixel im gelben Segment 0 und weitergehen zum nächsten Pixel für gelb
17	YellowEven:P_ADR	P_ADR+=160	Weiter zu Segment 1 (gelb)
18	YellowEven:P_ADR	P_ADR+=160	Weiter zu Segment 2 (gelb)
19	YellowEven:P_ADR	P_ADR+=160	Weiter zu Segment 3 (gelb)
20	YellowEven:P_ADR	P_ADR+=160	Weiter zu Segment 4 (gelb)
21	YellowEven:P_ADR	P_ADR+=160	Weiter zu Segment 5 (gelb)

22	YellowEven:P_ADR	P_ADR+=160	Weiter zu Segment 6 (gelb)
23	YellowEven:P_ADR	P_ADR+=160	Weiter zu Segment 7 (gelb)
24	YellowEven:P_ADR	P_ADR= Cyan_P_ADR Cyan_P_ADR+=1 (mod5)	Adresse generieren für gerades Pixel im cyan Segment 0 und weitergehen zum nächsten Pixel für cyan
25	CyanOdd:P_ADR	P_ADR+=160	Weiter zu Segment 1 (cyan)
etc.			

[0353] Der Pseudocode zum Erzeugen der Zwischenspeicher 4 117-Adressen ist hier dargestellt. Es ist anzumerken, dass er als sequentieller Satz von Schritten aufgelistet ist. Tabelle 26 zeigt eine bessere Ansicht der parallelen Eigenschaft der Vorgänge während der Adressengenerierung.

```
% Calculate start positions
CyanEven = 0:0
CyanOdd = CyanEven + 0:1
MagentaEven = CyanOdd + 1:3
MagentaOdd = MagentaEven + 0:1
YellowEven = MagentaOdd + 1:3
YellowOdd = Yellow Even + 0:1

Do N times (depends on print size)
    Cyan_P_ADR = 0
    Magenta_P_ADR = 0
    Yellow_P_ADR = 0
    Do 400 times
       % generate the even pixels for the first set of 24 bits
       P_ADR = Integer portion of Cyan_P_ADR
       Cyan_P_ADR += 0:1
       Do 8 times
          ReadBeffer4(line=CyanEven, pixel=P_ADR)
          P_ADR += 160
       EndDo
```

```
P_ADR = Integer portion of Magenta_P_ADR
Magenta_P_Adr += 0:1
Do 8 times
   ReadBuffer4(line=MagentaEven, pixel=P_ADR
   P_ADR += 160
EndDo
P_ADR = Integer portion of Yelow_P_ADR
Yellow_P_Adr += 0:1
Do 8 times
   ReadBuffer4(line=YellowEven, pixel=P_ADR)
   P_ADR += 160
EndDo

% generate the odd pixels for the first set of 24 bits
P_ADR = Integer portion of Cyan_P_ADR
Cyan_P_ADR += 0:1
Do 8 times
   ReadBuffer4(line=CyanOdd, pixel=P_ADR
   P_ADR += 160
EndDo
P_ADR = Integer portion of Magenta_P_ADR
Magenta_P_ADR += 0:1
Do 8 times
   ReadBuffer4(line=MagentaOdd, pixel=P_ADR)
   P_ADR += 160
EndDo
P_ADR = Integer portion of Yellow_P_ADR
Yellow_P_ADR += 0:1
Do 8 times
   ReadBuffer4(line=YellowOdd, pixel=P_ADR)
   P_ADR += 160
EndDo

% Now can advance to next "line"
CyanEven += 0:1
```

```
        CyanOdd += 0:1
        MagentaEven += 0:1
        MagentaOdd += 0:1
        YellowEven += 0:1
        YellowOdd += 0:1
      EndDo
EndDo
```

10.2.5 Zwischenspeicher 3 **116**

[0354] Der Zwischenspeicher 3 ist ein einfacher Satz von 8-Bit R-, G-, B-Werten. Diese RGB-Werte sind die geschärften Pixel in mittlerer Auflösung (1280er Auflösung), die durch den Schärfungsprozess **65** erzeugt wurden, und von dem Umwandeln-in-CMY-Prozess **66** gelesen werden.

[0355] Es ist nicht notwendig, den Zwischenspeicher 3 **116** doppelt zwischenzuspeichern. Dies deshalb, weil der Lese-(Umwandeln-zu-CMY) Prozess **66** nur die RGB-Werte für die ersten 39 Zyklen erfordert, wohingegen der Schreib-(Schärfungs-)Prozess **65** 49 Zyklen benötigt, bevor er bereit ist, die RGB-Werte tatsächlich zu aktualisieren.

10.2.6 Umwandeln-zu-CMY **66**

[0356] Die Umwandlung von RGB zu CMY wird im mittleren Auflösungsraum (1280er Auflösung) wie in Abschnitt 3.2.7 beschrieben durchgeführt.

[0357] Der Umwandlungsprozess **66** muss die Zwischenspeicherpixel mit gleichmäßigem Farbverlauf (Zwischenspeicher 4) **117** mit einer Geschwindigkeit erzeugen, die schnell genug ist, um mit dem Hochinterpolier-Halbtön-Reformatier-Prozess **113** mitzuhalten. Da jeder Wert mit gleichmäßigem Farbverlauf für 25 Zyklen verwendet wird (5 Mal in sowohl x- als auch y-Richtung), kann der Umwandlungsprozess bis zu 25 Zyklen benötigen. Dies führt zu einer Gesamtanzahl von 75 Zyklen für alle 3 Farbbestandteile.

[0358] Der wie hierin beschriebene Prozess benötigt lediglich 14 Zyklen pro Farbbestandteil, wobei die Eingabe-RGB-Werte tatsächlich nach 39 Zyklen freigegeben werden. Wenn der Prozess mit einer Logik implementiert wird, die den Zugriff auf die Eingabe-RGB-Werte für länger als 49 Zyklen erfordert, dann erfordert der Zwischenspeicher 3 **116** eine Doppel-Zwischenspeicherung, da die Werte durch den Schärfungsprozess **65** nach diesem Mal aktualisiert werden.

[0359] Die Umwandlung wird als tri-lineare Interpolation durchgeführt. Drei 17 × 17 × 17 Nachschlagetabellen werden für den Umwandlungsprozess verwendet: RGB zu cyan **90**, RGB zu magenta **91** und RGB zu gelb **92**. Da man jedoch 25 Zyklen hat, um jede tri-lineare Interpolation durchzuführen, besteht kein Bedarf nach einer schnellen tri-linearen Interpolationseinheit. Stattdessen sind 8 Aufrufe an einen linearen Interpolationsprozess **130** mehr als ausreichend.

[0360] Die Adressengenerierung zum Indizieren in die Nachschlagetabellen ist einfach. Wir verwenden die 4 signifikantesten Bits jedes 8-Bit Farbbestandteils zur Adressengenerierung, und die 4 am wenigsten signifikanten Bits jedes 8-Bit-Farbbestandteils zur Interpolation zwischen den aus den Umwandlungstabellen ermittelten Werten. Die Adressierung in die Nachschlagetabelle erfordert einen Addierer aufgrund der Tatsache, dass die Nachschlagetabelle 17 Größen und nicht 16 aufweist. Glücklicherweise ergibt das Multiplizieren einer 4-Bit Zahl X mit 17 eine 8-Bit Zahl XX, und erfordert deshalb keinen Addierer oder Multiplizierer, und das Multiplizieren einer 4-Bit Zahl mit 17^2 (289) ist nur unwesentlich komplizierter, indem es eine einzige Addition erfordert.

[0361] Obwohl die Interpolation schneller durchgeführt werden könnte, wird ein einzelner Addierer verwendet, um Adressen zu erzeugen und um eine einzige Zyklusinterpolationseinheit zu haben. Folglich sind wir in der Lage, die Interpolation zum Erzeugen eines einzelnen Farbbestandteils aus RGB in 14 Zyklen zu berechnen, wie in Tabelle 27 dargestellt. Der Prozess muss 3 Mal wiederholt werden, um cyan, magenta und gelb zu erzeugen. Schnellere Verfahren sind möglich, jedoch nicht notwendig.

Tabelle 27. Tri-lineare Interpolation zur Farbumwandlung

Zyklus	Laden	Effektiver Abruf	ADR-Register anpassen	Interpolieren
1			ADR = 289R	
2			ADR = ADR +17G	
3			ADR = ADR + B	
4	P1	RGB	ADR = ADR + 1	
5	P2	RGB +1	ADR = ADR + 16	
6	P1	RG+1B	ADR = ADR + 1	P3 = P1 zu P2 mit B
7	P2	RG+1B+1	ADR = ADR + 271	
8	P1	R+1GB	ADR = ADR + 1	P4 = P1 zu P2 mit B
9	P2	R+1GB+1	ADR = ADR + 16	P5 = P3 zu P4 mit G
10	P1	R+1G+1B	ADR = ADR + 1	P3 = P1 zu P2 mit B
11	P2	R+1G+1B+1		
12				P4 = P1 zu P2 mit B
13				P6 = P3 zu P4 mit G
14				V = P5 zu P6 mit R

[0362] Wie in Tabelle 27 dargestellt, können ein einziges ADR-Register und ein Addierer zur Adressengenerierung in die Nachschlagetabellen verwendet werden. 6 Sätze von 8-Bit Registern können verwendet werden, um die Zwischenergebnisse zu halten – 2 Register beinhalten die von den Nachschlagetabellen geladenen Werte, und 4 Register werden für die Ausgabe aus der Interpolationseinheit verwendet. Es ist anzumerken, dass der Eingang in die lineare Interpolationseinheit immer ein Paar von 8-Bit Registern P1/P2, P3/P4 und P5/P6 ist. Dies wird mit Absicht durchgeführt, um die Registerauswahllogik zu vermindern. In Zyklus 14 hält das "V"-Register **131** den schließlich berechneten 8-Bit Wert. Das 8-Bit Ergebnis kann an den entsprechenden Ort im Zwischenspeicher 4 **117** während des nächsten Zyklus geschrieben werden.

[0363] Ein Blockschaltbild des Umwandeln-zu-CMY-Prozesses **66** kann in Fig. 48 angesehen werden.

[0364] Unter der Annahme, dass der Prozess zuerst abläuft, um cyan zu erzeugen, wird das sich ergebende cyan Pixel mit gleichmäßigem Farbverlauf in dem cyan Zwischenspeicher mit gleichmäßigem Farbverlauf in 1280er Auflösung gespeichert. Der Prozess läuft anschließend erneut mit demselben RGB-Eingang, um das magenta Pixel zu erzeugen. Dieses magenta Pixel mit gleichmäßigem Farbverlauf wird in den Zwischenspeicher mit gleichmäßigem Farbverlauf in 1280er Auflösung gespeichert. Schließlich wird das gelbe Pixel mit gleichmäßigem Farbverlauf aus demselben RGB-Eingang erzeugt, und das resultierende gelbe Pixel wird in den gelben Zwischenspeicher mit gleichmäßigem Farbtonverlauf in 1280er Auflösung gespeichert).

[0365] Die Adressengenerierung zum Schreiben in den Zwischenspeicher mit gleichmäßigem Farbtonverlauf (Zwischenspeicher 4) **117** ist geradlinig. Eine einzelne Adresse (und die begleitenden ColorSelect-Bits) wird verwendet, um in jeden der 3 Farbzwischenspeicher zu schreiben. In den cyan Zwischenspeicher wird bei Zyklus 15 geschrieben, in den magenta bei Zyklus 30 und in den gelben bei Zyklus 40. Die Pixeladresse wird um 1 alle 75 Zyklen erhöht (nachdem alle 3 Farben geschrieben wurden). Die Zeile, in die geschrieben wird, erhöht sich mit Überlauf jedes Mal bei jedem 5. AdvanceLine-Impuls. Die Reihenfolge der geschrieben werdenden Zeilen ist einfach 0-1-2-3-4-5-0-1-2-3 usw. ... Damit gleichen die Schreibvorgänge (25 × 1280 × 3) sich mit den Lesevorgängen (19200 × 5) aus.

10.2.7 Zwischenspeicher 2 **115**

[0366] Der Zwischenspeicher 2 nimmt den Ausgang des Resample-Helligkeit-erzeugen-Prozesses **112** an, wobei ein volständiges RGB- und L-Pixel für eine gegebene Pixelkoordinate erzeugt wird. Die Ausgabe des Zwischenspeichers 2 **115** geht zu dem Schärtungsprozess **65**, der einen 3 × 3-Satz von Helligkeitswerten **135** erfordert, die um das zu schärfende Pixel herum zentriert sind.

[0367] Folglich besteht während des Schärfungsprozesses **65** ein Zugriffsbedarf auf das 3 × 3-Array von Helligkeitswerten sowie den entsprechenden RGB-Wert **136** für das zentrale Helligkeitspixel. Gleichzeitig müssen die nächsten 3 Helligkeitswerte und der entsprechende RGB-Zentrumswert durch den Resample-Helligkeit-erzeugen-Prozess **112** berechnet werden. Die logische Ansicht von Zugriffen auf den Zwischenspeicher 2 **115** ist in Fig. 49 dargestellt.

[0368] Die tatsächliche Implementierung des Zwischenspeichers 2 **115** ist einfach als ein 4 × 6 (24 Einträge) 8-Bit-RAM-Speicher, wobei das Adressieren beim Lesen und Schreiben das wirksame Verschieben der Werte bildet. Ein 2-Bit Spaltenzähler kann mit Überlauf erhöht werden, um einen zyklischen Zwischenspeicher bereitzustellen, wodurch effektiv das Äquivalent des Schiebens der gesamten Daten des Zwischenspeichers um 1 Spaltenposition implementiert wird. Die Tatsache, dass die 4. Spalte der RGB-Daten nicht erforderlich ist, ist nicht relevant, und verwendet lediglich 3 Bytes bei gleichzeitiger Ersparnis, eine komplizierte Schiebe- und Lese-/Schreibe-Logik nicht implementieren zu müssen. In einem gegebenen Zyklus kann der RAM-Speicher entweder beschrieben werden oder daraus gelesen werden. Die Lese-und-Schreib-Prozesse weisen 75 Zyklen auf, in denen abzuschließen ist, um mit dem Druckkopf mitzuhalten.

10.2.8 Schärfen

[0369] Die Schärfungseinheit **65** führt die in Abschnitt 3.2.6 beschriebene Schärfungsaufgabe durch. Da die geschärften RGB-Pixel in den Zwischenspeicher 3 **116** gespeichert werden, muss die Schärfungseinheit **65** mit dem Umwandeln-zu-CMY-Prozess **66** mithalten, wodurch impliziert wird, dass ein komplettes RGB-Pixel innerhalb 75 Zyklen geschärft werden muss.

[0370] Der Schärfungsprozess umfasst einen Hochpass-Filter von L (ein erzeugter Kanal aus den RGB-Daten und in dem Zwischenspeicher 2 gespeichert) und das Addieren des gefilterten L zurück in die RGB-Bestandteile, wie in Tabelle 12 innerhalb des Abschnitts 3.2.6.2 beschrieben. Der verwendete Hochpass-Filter ist ein einfacher Hochpass-Filter, der einen 3 × 3-Faltungskern verwendet, wie in Fig. 50 dargestellt.

[0371] Der Hochpass-Filter wird über 10 Zyklen berechnet. Der erste Zyklus lädt die temporären Register **140** mit 8 Mal dem zentralen Pixelwert (das zentrale Pixel um 3 Bits nach links verschoben). Die nächsten 8 Zyklen subtrahieren die verbleibenden 8 Pixelwerte mit einer Untergrenze von 0. Damit kann der gesamte Ablauf durch einen Addierer erreicht werden. Zyklus 10 umfasst die Multiplikation des Ergebnisses mit einer Konstanten **141**. Diese Konstante ist die Darstellung von 1/9, ist jedoch ein Register, um zu ermöglichen, dass der Betrag mittels Software durch einen bestimmten Skalierungsfaktor geändert wird.

[0372] Der Gesamtbetrag wird anschließend zu den R-, G- und B-Werten addiert (mit einem oberen Grenzwert von 255) und in den Zwischenspeicher 3 während der Zyklen 72, 73 und 74 geschrieben. Das Berechnen/Schreiben der geschärften RGB-Werte während der letzten 3 Zyklen des 75-Zyklen-Satzes beseitigt den Bedarf nach einer doppelten Zwischenspeicherung im Zwischenspeicher 3.

[0373] Die Struktur der Schärfungseinheit kann aus Fig. 51 ersehen werden.

[0374] Die mit dem Zwischenspeicher 2 **115** verbundene Addierer-Einheit **142** ist ein Subtrahierer mit einem unteren Grenzwert von 0. TMP **140** wird mit 8 × dem ersten L-Wert während des Zyklus 0 (von 75) geladen, und anschließend werden die nächsten 8 L-Werte davon subtrahiert. Das Ergebnis wird nicht mit einem Vorzeichen versehen, da die Subtraktion einen unteren Grenzwert von 0 aufweist.

[0375] Während des 10. Zyklus (Zyklus 9) wird der 11-Bit Gesamtwert in TMP **140** mit einem Skalierungsfaktor multipliziert (typischerweise 1/9, aber unter Softwaresteuerung, so dass der Faktor angepasst werden kann) und zurück in den TMP 140 geschrieben. Nur die 8 ganzzahligen Bits des Ergebnisses werden in den TMP geschrieben (der Bruchteil wird abgeschnitten), deshalb beträgt die Grenze der Multiplizier-Einheit 255. Wenn ein Skalierungsfaktor von 1/9 verwendet wird, ist der maximale Wert, der geschrieben wird, 226 (225 × 8/9). Der Skalierungsfaktor ist 8 Bits des Bruchteils, wobei das höchste Bit 1/8 darstellt. Der variable Skalie-

rungsfaktor kann die Tatsache berücksichtigen, dass unterschiedliche Druckformate das Ergebnis des Skalierens des Farbfiltermatrix-Bildes mit einem anderen Betrag sind (und damit die 3 × 3-Faltung entsprechend skalierte Ergebnisse erzeugt).

[0376] Die geschärften Werte für rot, grün und blau werden während Zyklus 72, Zyklus 73 und Zyklus 74 berechnet, und in die R-, G- und B-Register des Zwischenspeichers 3 **116** jeweils mit einem Schreiben pro Zyklus geschrieben. Die in diesen 3 Zyklen durchgeführte Berechnung ist einfach die Addition von TMP zu den R-, G- und B-Werten des Zwischenspeichers 2, die dem zentralen Pixel entsprechen.

[0377] Die Adressengenerierung ist unkompliziert. Das Schreiben in den Zwischenspeicher 3 **116** ist einfach R, G und B in den Zyklen 72, 73 bzw. 74. Das Lesen aus dem Zwischenspeicher 2 **115** macht Gebrauch von der zyklischen Eigenschaft des Zwischenspeichers 2. Die Adresse besteht aus einem 2-Bit-Spaltenanteil (der darstellt, welche der 4 Spalten gelesen werden soll), und einem 3-Bit-Wert, der L1, L2, L3, R, G oder B darstellt. Die Spaltenzahl beginnt bei 1 in jeder Zeile und erhöht sich (mit Überlauf) alle 75 Zyklen. Die Reihenfolge des Lesens des Zwischenspeichers 2 ist in Tabelle 28 dargestellt. Das C-Register ist der 2-Bit Spaltenbestandteil der Adresse. Alle Additionen zu C sind modulo 4 (Überlauf innerhalb 2 Bits).

Tabelle 28. Lesezugriff auf Zwischenspeicher 2 während des 75-Zyklen-Satzes

Zyklus	Adresse	Aktualisieren von C
0	C, L2	C=C-1
1	C, L1	
2	C, L2	
3	C, L3	C=C+1
4	C, L1	
5	C, L3	C=C+1
6	C, L1	
7	C, L2	
8	D, L3	C=C-1
9-71	kein Zugriff	
72	C, R	
73	C, G	
74	C, B	C=C-1

[0378] Nach Zyklus 74 beinhaltet das C-Register die Spaltenzahl für den nächsten Berechnungssatz, wodurch der Abruf während des nächsten Zyklus 0 gültig gemacht wird. Die Schärfung kann nur beginnen, wenn ausreichend L- und RGB-Pixel in den Zwischenspeicher 2 geschrieben worden sind (so dass der Hochpass-Filter gültig ist). Der Schärfungsprozess muss deshalb anhalten, bis der Zwischenspeicher 2-Schreibprozess um 3 Spalten fortgeschritten ist.

10.2.9 Zwischenspeicher 1 **114**

[0379] Der Zwischenspeicher 1 beinhaltet die weiß-abgeglichenen und bereichserweiterten Pixel in der räumlichen Auflösung der ursprünglichen Erfassung. Jedes Pixel wird mit 10 Bits an Farbauflösung gespeichert, im Vergleich zu der Bildspeicherfarbauflösung des Bild-RAM-Speichers von 8 Bits pro Pixel.

[0380] Der Zwischenspeicher 1 ist als 3 getrennt adressierbare Zwischenspeicher angeordnet – einer für jede Farbebene aus rot **145**, grün **146** und blau **147**. Ein einfacher Überblick der Zwischenspeicher ist in Fig. 52 dargestellt.

[0381] Während des Ablaufs von 75 Zyklen werden 16 Einträge von jedem der 3 Zwischenspeicher 3 Mal durch den Resampling-Prozess **112** gelesen, und bis zu 29 neue Werte werden in die 3 Zwischenspeicher geschrieben (die exakte Anzahl hängt vom Skalierungsfaktor und der aktuellen Sub-Pixel-Position während des Resamplings ab).

[0382] Die Zwischenspeicher müssen breit genug sein, so dass das Lesen und Schreiben vonstatten gehen kann, ohne einander zu stören. Während des Leseprozesses werden 4 Pixel von jeder der 6 Zeilen gelesen. Wenn der Skalierungsfaktor sehr groß ist (z. B. skalieren wir auf Panoramagröße), kann das selbe Eingabepixel mehrfach gelesen werden (unter Verwendung einer unterschiedlichen Kernposition für das Resampling). Letztendlich jedoch sind die nächsten Pixel erforderlich. Wenn wir nicht so weit hochskalieren, können die neuen Pixel vor dem nächsten PixelGenerierungszyklus (d. h. innerhalb 75 Taktzyklen) verlangt sein.

[0383] Betrachtet man die Skalierungsfaktoren in Tabelle 9 und Tabelle 11 ist der Worst Case für die Skalierung das Hochformat **31**:

- Die grüne Ebene weist einen Δ-Wert für Hochformat von 1,5625 auf, wodurch angegeben ist, dass 4 Orte innerhalb von 6 Farbfiltermatrix-Pixelpositionen enthalten sein können. Jede Zeile von grünen Abtastpunkten jedoch beinhaltet lediglich jedes 2. Pixel. Dies bedeutet, dass nur 4 Abtastpunkte pro Zeile erforderlich sind (der Worst Case ist 4, nicht 3 aufgrund einer Worst Case-Anfangsposition). Die Bewegung in Y-Richtung gibt das Erfordernis einer zusätzlichen Abtastspalte an, woraus sich 5 ergibt. Schließlich ist eine zusätzliche Abtastspalte für das Schreiben erforderlich. Dies ergibt eine Gesamtzahl von 6 Abtastpunkten pro Zeile. 7 Zeilen sind für einen einzelnen Abtastpunkt erforderlich. Um die 3 Sätze von RGB-Pixel für jede x-Position zu erzeugen, ist die maximale Bewegung in y-Richtung 4 Zeilen (3,125 = 2 × 1,5625). Die Bewegung in X-Richtung fügt eine Abtastzeile darüber und darunter hinzu. Folglich ist eine Gesamtzahl von 13 Zeilen erforderlich. Zu weiteren Einzelheiten siehe Abschnitt 10.2.10.
- Die roten und blauen Ebenen weisen einen Δ-Wert für das Hochformat von 0,78125 auf, wodurch angegeben ist, dass 4 Orte innerhalb von 4 Abtastpunkten enthalten sein können. Ein zusätzlicher Abtastpunkt ist für das Schreiben erforderlich, während die restlichen 4 gelesen werden. Dies ergibt eine Gesamtzahl von 5 Abtastpunkten pro Zeile, was des Weiteren auf 6 Abtastpunkte erhöht wird, um auf die grüne Ebene anzupassen (für Hochstartzwecke). 6 Zeilen sind erforderlich, um für die Bewegung in y-Richtung zu sorgen. Zu weiteren Einzelheiten siehe Abschnitt 10.2.10.

[0384] Jeder Unter-Zwischenspeicher wird als ein RAM-Speicher implementiert mit Decodierfunktion, um einen einzelnen 10-Bit-Abtastpunkt pro Zyklus zu lesen oder zu schreiben. Die Unter-Zwischenspeicher sind in Tabelle 29 zusammen gefasst und verbrauchen weniger als 200 Bytes.

Tabelle 29. Zusammenfassung der Unter-Zwischenspeicher

Zwischenspeicher	Zusammensetzung	Bits
Roter Zwischenspeicher	6 Zeilen x 6 Abtastpunkte x 10 Bits	360
Blauer Zwischenspeicher	6 Zeilen x 6 Abtastpunkte x 10 Bits	360
Grüner Zwischenspeicher	6 Zeilen x 6 Abtastpunkte x 10 Bits	780
	GESAMTZAHL	1500

10.2.10 Resampeln und Erzeugen des Helligkeitskanals

[0385] Der Resample- und Erzeuge-Helligkeitskanal-Prozess **112** ist verantwortlich für das Erzeugen des RGB-Pixelwerts im mittleren Auflösungsraum durch geeignetes Resampeln der weiß-abgeglichenen und bereichserweiterten R, G und B planaren Bilder, wie in Abschnitt 3.2.5 beschrieben. Darüber hinaus müssen die Helligkeitswerte für die gegebenen RGB-Pixel sowie die Helligkeitswerte für die Pixel oberhalb und unterhalb des RGB-Pixels für die Verwendung in dem späteren Schärfungsprozess erzeugt werden.

[0386] Die erlaubte Zeit zum Erzeugen des RGB-Werts und der 3 L-Werte beträgt 75 Zyklen. Wenn man bedenkt, dass L einfach der Durchschnittswert des Minimums und des Maximums von R, G und B für einen gegebenen Pixelort ist (siehe Abschnitt 3.2.6.1), müssen wir wirksam RGB-Werte für 3 Pixelkoordinaten erzeugen – das fragliche Pixel und das Pixel darüber und darunter. Damit haben wir 75 Zyklen, in denen die 3 RGB-Abtastwerte mittlerer Auflösung und ihre entsprechenden L-Werte berechnet werden müssen.

[0387] Das Zwischenspeichern der L-Werte (und damit der RGB-Werte), um eine Neuberechnung zu sparen, erfordert zu viel Speicher, und in jedem Fall steht uns nicht ausreichend Zeit zur Verfügung, um die RGB-Werte zu erzeugen. Der Zwischenspeicher 4 **117** enthält Pixel mittlerer Auflösung, kann jedoch nicht verwendet werden, da er die geschärften CMY-Pixel beinhaltet (anstatt der ungeschärften RGB-Pixel).

10.2.10.1 Resampling

[0388] Der Resampling-Prozess kann als 3 Sätze von RGB-Generierung gesehen werden, wobei jeder innerhalb von 25 Zyklen vollendet sein muss (damit man auf eine gesamte maximal verstrichene Zeit von 75 Zyklen kommt). Der Prozess des Generierens eines einzigen RGB-Werts kann wiederum als 3 Prozesse angesehen werden, die parallel durchgeführt werden: die Berechnung von R, die Berechnung von G und die Berechnung von B, alle für eine vorgegebene Pixelkoordinate mittlerer Auflösung. Die Theorie des Erzeugens jedes dieser Werte kann in Abschnitt 3.2.5 gefunden werden, aber das Fazit ist der effektive Ablauf von 3 Bildrekonstruktionsfiltern, einen auf jedem Kanal des Bildes. In dem Falle des PCP führen wir die Bildrekonstruktion mit 5 Abtastpunkten durch, was 4 Koeffizienten in dem Faltungskern erforderlich macht (da ein Koeffizient immer 0 ist und deshalb der Abtastpunkt nicht erforderlich ist).

[0389] Folglich wird die Berechnung des R-Pixels mittlerer Auflösung durch Ablauf eines Bildrekonstruktionsfilters auf den R-Daten erzielt. Die Berechnung des G-Pixels mittlerer Auflösung wird durch Ablauf eines Bildrekonstruktionsfilters auf den G-Daten erzielt, und die Berechnung des B-Pixels mittlerer Auflösung wird durch Ablauf eines Bildrekonstruktionsfilters auf den B-Daten erreicht. Obwohl die Kerne in x- und y-Richtung symmetrisch sind, sind sie nicht dieselben für jede Farbebene. R und B sind wahrscheinlich dieselben Kerne aufgrund ihrer ähnlichen Bildeigenschaften, aber die G-Ebene muss aufgrund der für die Bildrekonstruktion erforderlichen Drehung einen anderen Kern aufweisen. Die abstrakte Übersicht des Prozesses kann in Fig. 53 betrachtet werden. Die Adressengenerierung ist nicht dargestellt.

[0390] Der Resampling-Prozess kann nur beginnen, wenn ausreichend Pixel im Zwischenspeicher 1 für die aktuelle Pixel-Zeile, die erzeugt wird, vorhanden sind. Dies wird dann der Fall sein, sobald 4 Spalten von Daten auf jede der Farbebenen im Zwischenspeicher 1 **114** geschrieben worden sind. Der Resampling-Prozess **112** muss bis zu diesem Zeitpunkt anhalten.

[0391] Um den Pixelwert mittlerer Auflösung einer gegebenen Farbebene zu berechnen, stehen 25 Zyklen zur Verfügung. Um den Kern auf den 4 × 4-Abtastbereich anzuwenden, wendet man den 1D-Kern (durch x indiziert) auf jede der 4 Zeilen der 4 Eingabeabtastwerte an. Anschließend wenden wir den 1D-Kern (indiziert mit y) auf den resultierenden 4 Pixelwerten an. Das Endergebnis ist das ausgegebene resampelte Pixel. Das Anwenden eines einzelnen Koeffizienten in jedem Zyklus ergibt eine Gesamtzahl von 16 Zyklen, um die 4 Zwischenwerte zu erzeugen, und 4 Zyklen, um den finalen Pixelwert zu erzeugen, woraus sich eine Gesamtzahl von 20 Zyklen ergibt.

[0392] Bezüglich der Genauigkeit sind die Eingabepixel jeweils 10 Bits (8:2) und die Kern-Koeffizienten 12 Bits lang. Wir behalten 14 Bits Genauigkeit während der 4 Schritte jeder Anwendung des Kerns bei (8:6), aber heben nur 10 Bits für das Ergebnis auf (8:2). Dadurch kann dasselbe Faltungsmodul verwendet werden, wenn man in x- und y-Richtung faltet. Die finale Ausgabe oder R, G oder B beträgt 8 Bit.

[0393] Das Herz des Resampling-Prozesses ist die Faltungseinheit **150**, wie in Fig. 54 dargestellt.

[0394] Der Prozess des Resampelns umfasst dann 20 Zyklen, wie in Tabelle 30 dargestellt. Es ist anzumerken, dass Zeile 1, Pixel 1 usw. sich auf den Eingang aus dem Zwischenspeicher 1 **114** bezieht und durch den Adressierungsmechanismus erledigt wird (siehe unten).

Tabelle 30. Das 20 Zyklen Resampeln

Zyklus	Kern	Anwenden des Kerns auf:	Speichern des Ergebnisses in
1	X[1]	Zeile 1, Pixel 1	TMP
2	X[2]	Zeile 1, Pixel 2	TMP

3	X[3]	Zeile 1, Pixel 3	TMP
4	X[4]	Zeile 1, Pixel 4	TMP, V1
5	X[1]	Zeile 2, Pixel 1	TMP
6	X[2]	Zeile 2, Pixel 2	TMP
7	X[3]	Zeile 2, Pixel 3	TMP
8	X[4]	Zeile 2, Pixel 4	TMP, V2
9	X[1]	Zeile 3, Pixel 1	TMP
10	X[2]	Zeile 3, Pixel 2	TMP
11	X[3]	Zeile 3, Pixel 3	TMP
12	X[4]	Zeile 3, Pixel 4	TMP, V3
13	X[1]	Zeile 4, Pixel 1	TMP
14	X[2]	Zeile 4, Pixel 2	TMP
15	X[3]	Zeile 4, Pixel 3	TMP
16	X[4]	Zeile 4, Pixel 4	TMP, V4
17	Y[1]	V1	TMP
18	Y[2]	V2	TMP
19	Y[3]	V3	TMP
20	Y[4]	V4	TMP (für Ausgabe)

10.2.10.2 Generierung von L 8-

[0395] Wie in Abschnitt 3.2.6.1 beschrieben, muss man 80 von RGB auf L für den nachfolgenden Schärfungsprozess umwandeln. Wir betrachten den CIE 1976 L*a*b*-Farbraum, wobei L von der Wahrnehmung her gleichmäßig ist. Um von RGB auf L (den Helligkeitskanal) umzuwandeln, mitteln wir das Minimum und das Maximum von R, G und B wie folgt:

$$L = \frac{MIN(R,G,B) + MAX(R,G,B)}{2}$$

[0396] Die Generierung der R-, G- und B-Werte eines gegebenen Pixels wird parallel durchgeführt, was 20 Zyklen benötigt. Die Gesamtzeit für die Generierung von L wie hierin beschrieben beträgt 4 Zyklen. Dies ergibt eine Gesamtzeit des Erzeugens eines RGBL-Pixelsatzes von 24 Zyklen, wobei ein Zyklus ausgespart wird (da der Prozess innerhalb von 25 Zyklen vollendet sein muss).

[0397] Der Wert für L kann deshalb in dem 25. Zyklus sicher in den Zwischenspeicher 2 **115** hinausgeschrieben werden. Die Adressengenerierung ist unten beschrieben.

[0398] Ein einzelner 8-Bit-Komparator kann 3 Bits in 3 Zyklen erzeugen, was nachfolgend für das Auswählen der 2 Eingänge in den Addierer verwendet werden kann, wie in Tabelle 31 dargestellt. Die Teilung durch 2 kann einfach in den Addierer mit eingebaut werden.

Tabelle 31. Auswahl von Min und Max basierend auf 3 Vergleichen

MIN	MAX	R>G	G>B	R>B
R	B	1	1	x[a]
R	G	1	0	1
G	R	0	1	0
G	B	0	1	1
B	R	0	0	x
B	G	1	0	0

a. gleichgültiger Zustand

[0399] Da der Addierer lediglich das Minimum auf den Maximalwert addiert, ist die Reihenfolge unwichtig. Folglich kann von den 2 Eingängen in den Addierer Input1 eine Auswahl zwischen R und G sein, wohingegen Input2 eine Auswahl aus G und B ist. Die Logik ist eine Minimierung der geeigneten Bitmuster aus Tabelle 31.

10.2.10.3 Adressengenerierung für Zwischenspeicher 2

[0400] Die Ausgabe aus dem Resampler ist ein einzelnes RGB-Pixel und 3 Helligkeits-(L-)Pixel, die vertikal um das RGB-Pixel zentriert sind. Die 3 L-Werte können in den Zwischenspeicher 2 geschrieben werden, einer alle 25 Zyklen. Die R-, G- und B-Werte müssen nach Zyklus 45 und vor Zyklus 50 geschrieben werden, da das zweite erzeugte Pixel das zentrale Pixel ist, dessen RGB-Werte behalten werden müssen. Die Zwischenspeicher 2-Adresse besteht aus einem 2-Bit Spaltenbestandteil (der darstellt, in welche der 4 Spalten geschrieben werden muss), und einem 3-Bit-Wert, der L1, L2, L3, R, G oder B darstellt. Die Spaltenzahl beginnt bei 0 in jeder Zeile und erhöht sich (mit Überlauf) alle 75 Zyklen (d. h. nach dem Herausschreiben von L3).

10.2.10.4 Adressengenerierung für Kern-Nachschlagen

[0401] Das Berechnungsverfahren der Kernadresse ist dasselbe wie am Ende des Abschnitts 3.2.5 beschrieben. Jeder Kern ist 1-dimensional mit 64 Einträgen in der Tabelle. Die 6 signifikantesten Bits (abgeschnitten) des Bruchbestandteils in dem aktuellen Kern-Raum werden verwendet, um in die Kern-Koeffiziententabelle zu indizieren. Für die ersten 16 Zyklen wird die X-Ordinate verwendet, um den Kern zu indizieren, während in den nächsten 4 Zyklen die Y-Ordinate verwendet wird. Da der Kern symmetrisch ist, kann derselbe Kern für sowohl X als auch Y verwendet werden.

[0402] Für jeden der 1280 resampelten Werte muss man 3 Pixel erzeugen – das fragliche Pixel **161** und die Pixel oberhalb **160** und unterhalb **162** dieses Pixels. Um nicht ein zentrales Pixel zu erzeugen und anschließend sich nach oben und nach unten von diesem zentralen Pixel zu bewegen, erzeugen wir ein Pixel **160** und erzeugen die beiden Pixel **161**, **162** darunter. Das zweite erzeugte Pixel **161** wird als das zentrale Pixel genommen. Anschließend kehren wir zu der ursprünglichen Zeile zurück und erzeugen die nächsten 3 Pixel in der nächsten Ausgabeposition. Auf diese Weise erzeugen wir, wie in Fig. 55 dargestellt, 3 Pixel für jede der 1280 Positionen.

[0403] Damit haben wir eine aktuelle Position im Kernraum. Wenn man sich zu dem nächsten Pixel in X- oder Y-Richtung im originalen Eingaberaum weiter bewegt, fügt man geeignete Δ-Werte zu diesen Kern-Koordinaten hinzu. Betrachtet man Fig. 56, sieht man die beiden Fälle für den gedrehten und nicht gedrehten Eingaberaum.

[0404] Wir betrachten die Bewegung in X- und Y-Richtung als ΔX und ΔY, wobei deren Werte von dem Druckformat abhängen, und damit die Werte von mps (siehe Abschnitt 3.2.5). Für den grünen Kanal ist ΔX = ΔY = 1/2mps. Für die roten und blauen Kanäle ist ΔX = 1/mps und ΔY = 0. Siehe Tabelle 9 und Tabelle 11 für die geeigneten Werte von ΔX und ΔY.

[0405] Wir können nun die ΔX- und ΔY-Werte auf die Bewegung innerhalb des Kerns anwenden. Folglich addieren wir, wenn wir in X-Richtung voranschreiten, ΔX auf X und subtrahieren ΔY von Y. In dem nicht gedrehten

Fall subtrahiert dies lediglich 0 von Y. In ähnlicher Weise addieren wir, wenn wir in Y-Richtung voranschreiten, ΔY zu X und ΔX zu Y. Wir können dies so durchführen, da die Bewegung in X- und Y-Richtung sich um 90 Grad unterscheiden.

[0406] Die Adressengenerierung für das Kern-Nachschlagen nimmt eine Startposition an, die durch Software gesetzt wird, und 2 Δ-Werte ΔX und ΔY bezüglich der Bewegung in Y-Richtung im Kernraum. Die Adressengenerierungslogik ist in dem nachfolgenden Pseudocode dargestellt:

```
ColumnKernelY = StartKernelY
ColumnKernelX = StartKernelX
Do NLines times (however many output lines there are to
process)
   KernelX = ColumnKernelX
   KernelY = ColumnKernelY
   Do 1280 times
       Generate Pixel
       KernelX = KernelX + DeltaY (movement in Y)
       KernelY = KernelY + DeltaX (movement in Y)
       Generate Pixel
       KernelX = KernelX + DeltaY (movement in Y)
       KernelY = KernelY + DeltaX (movement in Y)
       Generate Pixel
       KernelX = ColumnKernelX + DeltaX (movement in X)
       KernelY = ColumnKernelY - DeltaY (movement in X)
   EndDo
   ColumnKernelX = ColumnKernelY + DeltaX (movement in Y)
   ColumnKernelX = ColumnKernelX + DeltaY (movement in Y)
EndDo
```

[0407] Wie in dem Pseudocode dargestellt, erfolgt die Generierung von 3 Pixeln 1280 Mal. Verbunden mit der Generierung jedes Pixels sind 2 Additionen, die im Laufe der GeneratePixel 25 Zyklen-Aufgabe durchgeführt werden. Jede GeneratePixel-Aufgabe ist 25 Zyklen lang, besteht aus 4 Sätzen zu 4 Zyklen, die den Kern über KernelX (Koeffizient 0, 1, 2, 3) indizieren, gefolgt von 4 Zyklen, die den Kern über KernelY (Koeffizienten 0, 1, 2, 3) indizieren, gefolgt von 9 Wartezyklen.

[0408] Es ist anzumerken, dass alle Werte positiv und lediglich Bruchteile sind. Die beiden Überträge von dem Aktualisieren der X- und Y-Kernwerte werden zur Adressengenerierung des Zwischenspeichers 1 ausgegeben (siehe Abschnitt 10.2.10.5). Diese Übertrag-Merker geben einfach an, ob die bestimmten Ordinaten für den Kern während der mathematischen Operation übergelaufen sind oder nicht. Der Überlauf kann entweder über 1 oder unter 0 liegen, aber das Ergebnis ist immer positiv.

[0409] Die beiden Übertrag-Bits werden auch an die Dreh-/Weißabgleich-/Bereichserweitungs-Einheit gesendet zur Verwendung bei der Bestimmung der relativen Eingabezeilen von dem Bild.

10.2.10.5 Adressengenerierung für Zwischenspeicher 1

[0410] Der Resampler **112** liest aus dem Zwischenspeicher 1 **114**, der aus drei individuell adressierbaren Zwischenspeichern **145**, **146** und **147** besteht – einer für jede Farbebene. Während jedes Zyklus kann entweder aus jedem Zwischenspeicher gelesen oder darauf geschrieben werden.

[0411] Der Leseprozess von 75 Zyklen wird in 3 Sätze von 25 Zyklen herunter gebrochen, einen Satz von 25 Zyklen für die Generierung jedes Pixels. Jeder Satz von 25 Zyklen umfasst 16 Lesevorgänge aus dem Zwischenspeicher 1 gefolgt von 9 Zyklen mit keinem Zugriff. Während dieser 9 Zyklen wird auf den Zwischenspeicher 1 geschrieben. Die 16 Lesevorgänge aus dem Zwischenspeicher 1 **114** sind tatsächlich 4 Sätze von 4 Lesevorgängen und fallen mit 4 Gruppen mit 4 Lesevorgängen des Kerns für jede Farbebene zusammen.

[0412] Die Adressengenerierung umfasst dann das Erzeugen von 16 Adressen für die Berechnung des ersten Pixels (gefolgt von 9 Wartezyklen), das Erzeugen von 16 Adressen zur Berechnung des zweiten Pixels (gefolgt von 9 Wartezyklen) und schließlich das Erzeugen von 16 Adressen für das dritte Pixel (gefolgt von 9 Wartezyklen).

[0413] Jede Farbebene weist ihre eigenen Start-Adressparameter des Zwischenspeichers 1 auf. Da die 3 Sätze von 16 Adressen für jede der 1280 Positionen entlang der Zeile erzeugt werden, und da der Abtaster/Sampler von einer Zeile von 1280 Abtastpunkten zur nächsten fortschreitet, werden die beiden Übertrag-Bits aus der Kern-Adressengenerierungseinheit verwendet, um diese Adressparameter des Zwischenspeichers 1 zu aktualisieren.

10.2.10.6 Grüner Zwischenspeicher **146**

[0414] Die Adressengenerierung für den grünen Unter-Zwischenspeicher **146** innerhalb des Zwischenspeichers 1 **114** ist aus den beiden folgenden Hauptgründen komplizierter als die der roten Unter-Zwischenspeicher **145** und blauen Unter-Zwischenspeicher **147**:

- Der grüne Kanal stellt ein schachbrettartiges Muster in der Farbfiltermatrix dar. Abwechselnde Zeilen bestehen lediglich aus ungeraden oder geraden Pixeln. Um den grünen Kanal zu resamplen, muss man den Kanal wirksam um 45 Grad drehen.
- Es gibt doppelt so viele grüne Pixel wie rote oder blaue Pixel. Ein Resampling bedeutet das Lesen von mehr Abtastpunkten in demselben Zeitraum – es sind immer noch 16 Abtastpunkte zu lesen, um jedes Pixel in dem mittleren Auflösungsraum zu erzeugen, aber es existiert eine höhere Wahrscheinlichkeit, den Zwischenspeicher jedes Mal fortzuschreiben. Die exakte Wahrscheinlichkeit hängt von dem verwendeten Skalierungsfaktor ab.

[0415] Es wird jedoch dasselbe Konzept der Verwendung eines RAM-Speichers als ein zyklischer Zwischenspeicher für den grünen Kanal verwendet. Der grüne Unter-Zwischenspeicher ist ein RAM-Speicher mit 78 Einträgen mit einer logischen Anordnung von 13 Zeilen, die jeweils 6 Einträge aufweisen. Die Beziehung zwischen RAM-Adresse und logischer Position ist in **Fig. 57** dargestellt.

[0416] Die Abtastpunkte im Zwischenspeicher 1 **146** stellen ein Schachbrettmuster innerhalb der Farbfiltermatrix dar. Folglich können Abtastpunkte in einer Zeile (z. B. die Adresse 0, 13, 26, 39, 52, 65) ungerade oder gerade Pixel je nach aktueller Zeile innerhalb des gesamten Bildes darstellen, und je nachdem, ob das Bild um 90 Grad gedreht wurde oder nicht. Dies ist in **Fig. 58** veranschaulicht.

[0417] Folglich gibt es, wenn man einen 4 × 4-Abtastbereich auf den Zwischenspeicher abbildet, zwei Möglichkeiten für die Interpretation der Abtastpunkte. Als ein Ergebnis gibt es zwei Arten der Adressierung, je nachdem ob die aktuelle Zeile durch ungerade oder gerade Pixel dargestellt wird. Dies bedeutet, dass gerade Zeilen mit Bilddrehung 0 dieselbe Adressierung aufweisen wie ungerade Zeilen mit Bilddrehung 90 Grad, da sie beide ungerade Pixel enthalten. In gleicher Weise haben die ungeraden Zeilen mit Bilddrehung 0 dieselbe Adressierung wie gerade Zeilen mit Bilddrehung 90 Grad, da sie beide gerade Pixel enthalten. Die Entscheidung ist in Tabelle 32 zusammengefasst.

Tabelle 32. Bestimmung des Abtast-Typs

Drehung	aktuelle Zeile		Pixel	Typ
0	gerade Zeile	8	ungerade	Typ 2
0	ungerade Zeile	8	gerade	Typ 1
90	gerade Zeile	8	gerade	Typ 1
90	ungerade Zeile	8	ungerade	Typ 2

[0418] Das tatsächliche 4 × 4-Abtastfenster ist die Art und Weise, wie wir wirksam den Zwischenspeicher um 45 Grad drehen. Die 45-Grad-Drehung ist für ein effektives Resampling notwendig, wie in Abschnitt 3.2.5 beschrieben.

[0419] Nimmt man in diesem Moment an, dass man nur ein einziges Resample erzeugen muss, betrachtet man die Zwischenspeicher-Adressierung durch Untersuchen der beiden Typen von 4 × 4-Abtastfenstern, wie in Fig. 59 dargestellt.

[0420] Obwohl die beiden 4 × 4-Abtast-Typen ähnlich aussehen, kommt der Unterschied von der Art und Weise, in der die 4 × 4-Abbildung in dem planaren Bild dargestellt ist. Fig. 60 veranschaulicht die Abbildung des 4 × 4-Abtastens vom Typ 1 in dem grünen Unter-Zwischenspeicher. Nur die oberen 7 Zeilen und die ganz rechts befindlichen Spalten sind dargestellt, da der 4 × 4-Abtastbereich vollständig innerhalb dieses Bereichs enthalten ist.

[0421] Die Zuordnung der Zwischenspeicherpixel zu Abtastzeilen für den Typ 2-Abtastprozess ist sehr ähnlich und kann aus Fig. 61 ersehen werden.

[0422] Sowohl bei der Typ-1 und Typ-2-Adressierung der 16 Abtastpunkte gibt es zwei Möglichkeiten, eine Zeile zu verarbeiten. Die Verarbeitung der Zeilen 1 und 3 der Typ-1-Adressierung ist dieselbe (relativ gesprochen) wie die Verarbeitung der Zeilen 2 und 3 des Typs 2. In ähnlicher Weise ist die Verarbeitung der Zeilen 2 und 4 des Typs 1 dieselbe (relativ gesprochen) wie die Verarbeitung der Zeilen 1 und 3 des Typs 2. Wir nennen dieses Zeilen-Adressierungsverfahren Typ A170 und Typ B171, wie in Fig. 62 dargestellt.

[0423] Gibt man eine Startposition für das 4 × 4-Fenster (WindowStartAdr) und einen Starttyp (WindowStartType) vor, können wir die Adressen für die 16 Abtastpunkte mittels einer Tabelle mit 8 Einträgen (zum Durchqueren der beiden Sätze aus 4 Abtastpunkten) erzeugen. Wenn wir den ersten Abtastwert lesen, fügen wir einen Versatz bzw. Offset von der Tabelle hinzu, um an der nächsten Abtastpunktposition anzukommen. Der Offset hängt von dem Typ ab (A, B = 0, 1). Der Offset von dem vierten Abtastpunkt ist der Betrag, der benötigt wird, um an dem ersten Abtastpunkt für die nächste Zeile anzukommen (und muss die Anzahl der Abtastspalten berücksichtigen). Nach dem Erzeugen jeder Zeile von 4 Abtastpunkten wechseln wir zwischen Typ A und Typ B. Die Logik zum Erzeugen der Adressen für einen einzelnen Satz von 16 Abtastpunkten ist in dem nachfolgenden Pseudocode dargestellt. Die Addition modulo 78 sorgt für den zyklischen Zwischenspeicher.

```
Adr = WindowStartAdr
TypeAB = WindowStartType
Do 4 times
   For N = 0 to 4
       Fetch Adr
       Adr = (Adr + Table[TypeAB,N]) mod 78
   EndFor
   TypeAB = NOT TypeAB
EndDo
```

[0424] Die Nachschlagetabelle besteht aus 8 Einträgen – 4 für Typ A 170 und 4 für Typ B 171 Adressversatz-Generierung. Die Versätze bzw. Offsets beziehen sich alle auf die aktuelle Abtastposition (Adr).

Tabelle 33. Offset-Werte für 16-PunktAdressengenerierung

TypeAB	N	Offset
0	0	14
0	1	1
0	2	14
0	3	37
1	0	1
1	1	14
1	2	1
1	3	37

[0425] Am Ende der 16 Lesevorgänge ist das TypeAa-Bit dasselbe wie der ursprüngliche Wert (aus WindowStartType geladen).

[0426] Das Lesen eines einzelnen Satzes von 16 Abtastpunkten ist nicht ausreichend. 3 Sätze von 16 Abtastpunkten müssen gelesen werden (die 3 unterschiedliche Positionen in Y-Richtung in dem ungedrehten Eingaberaum darstellen). An dem Ende des ersten und des zweiten Satzes von 16 Abtastpunkten werden die Kernpositionen durch den Kern-Adressengenerator aktualisiert. Die Übertrag-Bits von dieser Aktualisierung werden verwendet, um das Fenster für den nächsten Satz von 16 Abtastpunkten einzustellen. Die beiden Übertrag-Bits indizieren in eine Tabelle, die einen Offset und einen 1-Bit-Merker enthält. Der Offset wird zu WindowStartAdr hinzu addiert, und der Merker wird verwendet, um zu bestimmen, ob WindowStartType zu invertieren ist oder nicht. Die Werte für die Tabelle sind in Tabelle 34 dargestellt.

Tabelle 34. Aktualisieren von WindowStartAdr und WindowStartType

KernX Übertrag	KernY Übertrag	Offset	Type
0	0	0	keine Veränderung
0	1	1	Invertieren
1	0	14	Invertieren
1	1	2	Keine Veränderung

[0427] Am Ende des dritten Satzes von 16 Abtastpunkten werden die Kernpositionen aktualisiert, um das Fortschreiten in X-Richtung im ungedrehten Eingaberaum auszugleichen. Dieses Mal wird eine andere Bewegungsrichtung erzeugt, so dass eine andere Offset/TypeAB-Veränderungstabelle verwendet wird. Wir können diese Offsets nicht auf den aktuellen WindowStartAdr-Wert addieren, da dieser eine Position mit zwei Bewegungen in Y-Richtung weg von dem Punkt darstellt, wo wir die Bewegung beginnen wollen. Folglich laden wir WindowStartAdr und WindowStartType aus einem anderen Satz von Variablen:
TopStartAdr und TopStartAdr, die den ersten Eintrag in der aktuellen Zeile aus 1280 darstellen. Die beiden Übertrag-Merker aus dem Kern-Adressengenerator werden verwendet, um die Tabelle 35 nachzuschlagen, um den Offset zu bestimmen, der auf TopStartAdr zu addieren ist, und ob TopStartType zu invertieren ist oder nicht. Wie zuvor ist die Addition modulo 78 (die Größe des grünen RAM-Speichers). Die Ergebnisse werden zu WindowStartAdr und WindowStartType zur Verwendung bei der Generierung der nächsten 3 Sätze von 16 Abtast-

punkten kopiert.

Tabelle 35. Aktualisieren von TopStartAdr und TopStartType

KernX Übertrag	KernY Übertrag	Offset	Type
0	0	0	keine Veränderung
0	1	12	Invertieren
1	0	14	Invertieren
1	1	13	Keine Veränderung

[0428] Nach der Verarbeitung von 1280 Sätzen von 3 Sätzen aus 16 Abtastpunkten beginnt die nächste Zeile aus 1280 Punkten. Es muss jedoch die Adresse des ersten Abtastpunktes für Position 0 innerhalb der nächsten Zeile bestimmt werden. Da die Abtastpunkte immer in die korrekten Stellen im Zwischenspeicher 1 geladen werden, können wir immer von exakt derselben Position im Zwischenspeicher 1 starten (d. h. TopStartAdr kann aus einer konstanten Position0Adr geladen werden). Es muss jedoch berücksichtigt werden, um welchen Typ es sich handelt, da der Typ davon abhängt, wie viel man voran schreitet. Folglich gibt es einen Anfangswert für Position0Type, der in Abhängigkeit der Übertrag-Merker aus dem Kernadressengenerator aktualisiert werden muss. Da wir uns in dem ungedrehten Y-Eingaberaum bewegen, ist die verwendete Logik dieselbe wie für das Aktualisieren von WindowStartType mit der Ausnahme, dass sie stattdessen auf Position0Type durchgeführt wird. Der neue Wert für Position0Type wird in TopStartType kopiert und WindowStartAdr, um das Abtasten der ersten Position der neuen Zeile zu beginnen.

[0429] Der Abtastprozess für eine gegebene 1280er Positionszeile kann nicht beginnen, sofern nicht ausreichend Einträge im Zwischenspeicher 1 vorhanden sind, die dort durch die Dreh, Weißabgleichs-, Bereichserweiterungs-Einheit platziert wurden. Dies passiert 128 Zyklen nach dem Start jeder neuen Zeile (siehe Abschnitt 10.2.11).

10.2.10.7 Rote und blaue Zwischenspeicher

[0430] Der rote Unter-Zwischenspeicher **145** und blaue Unter-Zwischenspeicher **147** des Zwischenspeichers 1 sind einfach zwei RAM-Speicher, auf die als zyklische Zwischenspeicher zugegriffen wird. Jeder Zwischenspeicher ist 30 Bytes groß, hat jedoch eine logische Anordnung von 6 Zeilen, die jeweils 6 Einträge aufweisen. Das Verhältnis zwischen RAM-Adressen und logischer Position ist in Fig. 63 dargestellt.

[0431] Für rot und blau sind die ersten zu lesenden 16 Abtastpunkte immer die oberen 4 × 4 Einträge. Auf die restlichen 2 Spalten von Abtastpunkten wird durch den Lesealgorithmus zu diesem Zeitpunkt nicht zugegriffen.

[0432] Die Adressengenerierung für diese ersten 16 Abtastpunkte ist einfach eine Startposition (in diesem Falle 0) gefolgt von 16 Schritten der Addition modulo 36, wie in dem folgenden Pseudocode dargestellt:

```
Adr = StartAdr
Do 4 times
   Do 4 times
       ADR = ADR + 6 MOD 36
   End Do
   ADR = ADR + 13 MOD 36
EndDo
```

[0433] Dieser Adressengenerierungsmechanismus ist jedoch anders als der für den grünen Kanal. Um nicht

zwei Adressierungsmechanismen zu konzipieren, ist es möglich, das grüne Adressierungsschema auf die roten und blauen Kanäle anzuwenden, und einfach unterschiedliche Werte in den Tabellen zu verwenden. Dies vermindert die Entwicklungskomplexität. Der einzige Unterschied ist dann die Addition modulo 36 anstatt der Addition modulo 78. Dies kann durch einen einfachen Multiplexer berücksichtigt werden.

[0434] Betrachtet man die unterschiedlichen Adressengenerierungstabellen für grün und berücksichtigt man diese als auf rot und blau angewendet, so wird deutlich, dass kein Erfordernis für einen Typ vorhanden ist, da sowohl der rote als auch der blaue Kanal nicht um 45 Grad gedreht werden muss. Damit kann man sicher den Typ-Wert ignorieren, das rote/blaue Äquivalent zu Tabelle 33, in Tabelle 36 dargestellt, weist 2 Sätze von identischen 4 Einträgen auf.

Tabelle 34. Offset-Werte für 16-PunktAdressengenerierung (rot/blau)

TypeAB	N	Offset
0	0	6
0	1	6
0	2	6
0	3	13
1	0	6
1	1	6
1	2	6
1	3	13

[0435] Wie bei der grünen Adressengenerierung bewegen wir uns 2 Mal in Y-Richtung, bevor wir zu dem nächsten Eintrag von 1280 voranschreiten. Für rot und blau existiert keine Skalierung zwischen der Bewegung im Kernraum und der Bewegung im Eingaberaum. Es ist auch keine Drehung vorhanden. Sowie wir uns in Y-Richtung bewegen, wird das ΔY von 0 zu KernelX (siehe Kern-Adressengenerierung in Abschnitt 10.2.10.4) addiert. Als Ergebnis wird der Übertrag aus KernelX niemals gesetzt werden. Betrachtet man Tabelle 34 sind die einzigen möglichen Vorkommnisse KernelX/KernelY-Werte von 00 oder 01. In dem Fall von 00 ist die grüne Lösung keine Änderung für sowohl WindowStartAdr noch WindowStartType, deshalb ist dies auch für rot und blau korrekt. In dem Fall von 01 wollen wir 1 zu WindowStartAdr hinzu addieren und kümmern uns nicht um WindowStartType. Die grünen Werte können deshalb sicher für rot und blau verwendet werden. Der Worst Case ist das Voranschreiten um 1 in der Adresse bei beiden Malen, woraus ein überlappender Worst Case wie in Fig. 65 dargestellt resultiert.

[0436] Am Ende des dritten Satzes von 16 Abtastpunkten müssen TopStartAdr und TopStartType aktualisiert werden. Da wir uns in X-Richtung bewegen (und ΔY = 0 zu KernelY addieren), wird der Übertrag aus KernelY immer bei 0 sein. Das rote/blaue Äquivalent zu Tabelle 35 ist hier in Tabelle 37 dargestellt. Es ist anzumerken, dass es keine Type-Spalte gibt, da Type für rot oder blau nicht wichtig ist.

Tabelle 37. Aktualisieren von TopStartAdr. und TopStartType (rot/blau)

KernelX Übertrag	KernelY Übertrag	Offset
0	0	0
0	1	-
1	0	6
1	1	-

[0437] Der Prozess des Voranschreitens von einer Zeile von 1280 Sätzen von 3 Pixeln zu der nächsten ist derselbe wie für grün. Die Position0Adr ist dieselbe wie für den ersten Satz von 16 Abtastpunkten für eine gegeben Zeile (Position0Adr = 0 für rot und blau), und Type ist nicht relevant. Die Generierung der nächsten Zeile kann nicht beginnen, bis ausreichend Abtastpunkte in dem Zwischenspeicher 1 vorhanden sind. Die rote und blaue Generierung muss zum selben Zeitpunkt wie die grüne Generierung beginnen, sie kann folglich nicht beginnen, bis 128 Zyklen nach dem Beginn einer neuen Zeile verstrichen sind (siehe Abschnitt 10.2.11).

10.2.11 Drehung, Weißabgleich und Bereichserweiterung **111**

[0438] Die eigentliche Aufgabe des Laden des Zwischenspeichers 1 **114** aus dem Bild-RAM-Speicher **11** umfasst die Schritte des Drehens, des Weißabgleichs und der Bereichserweiterung **111**, wie in Abschnitt 3.2.3 und Abschnitt 3.2.4 beschrieben. Die Pixel müssen für den Zwischenspeicher 1 schnell genug für deren Verwendung durch den Resampling Prozess **112** erzeugt werden. Dies bedeutet, dass während einer einzelnen Gruppe von 75 Zyklen diese Einheit in der Lage sein muss, 6 rote Pixel, 6 blaue Pixel und 13 grüne Pixel zu lesen, zu verarbeiten und zu speichern.

[0439] Der optionale Drehschritt wird durch Einlesen der Pixel in der geeigneten Reihenfolge durchgeführt. Sobald ein gegebenes Pixel aus der entsprechenden Ebene in dem Bildspeicher gelesen wurde, muss es weiß-abgeglichen werden und sein Wert muss entsprechend der Bereichserweitungsberechnung angepasst werden, die in Abschnitt 3.2.4 definiert ist. Der Prozess umfasst einfach eine einzelne Subtraktion (unterer Bodenwert 0) und eine Multiplikation (oberer Wert 255), beides gegen farbspezifische Konstanten. Die Struktur dieser Einheit ist in **Fig. 66** dargestellt.

[0440] Die roten, grünen und blauen unteren Grenzwerte **72** werden zusammen mit den roten, grünen und blauen Skalierungsfaktoren **173** durch die CPU **10** nach dem Erzeugen der Histogramme für jede Farbebene durch die Bildhistogrammeinheit 8 (siehe Abschnitt 9) bestimmt.

[0441] Je nachdem, ob das aktuell verarbeitete Pixel in der Reihe rot, grün oder blau ist, werden der entsprechende untere Grenzwert und Skalierungsfaktor gleichzeitig in die Subtrahiereinheit und in die Multipliziereinheit gesendet, wobei die Ausgabe in die entsprechende Farbebene in den Zwischenspeicher 1 geschrieben wird.

[0442] Die Subtraktionseinheit **172** subtrahiert den 8-Bit unteren Grenzwert von dem 8-Bit Bild-RAM-Speicherpixelwert und weist einen unteren Bodenwert von 0 auf. Das 8-Bit Ergebnis wird an die spezialisierte 8 × 8-Multiplikationseinheit weiter gegeben, die den 8-Bit Wert mit dem 8-Bit Skalierungsfaktor multipliziert (8 Bit Bruchteil, Ganzzahlwert = 1). Nur die oberen 10 Bits des Ergebnisses werden behalten und stellen 8 Bits an Ganzzahlwert und 2 Bits Bruchteils dar. Der Multiplizierer **174** hat einen Ergebnisoberwert von 255, so dass, wenn ein beliebiges Bit höher als Bit 7 als ein Ergebnis der Multiplikation gesetzt worden wäre, das gesamte 8-Bit ganzzahlige Ergebnis auf 1 (Einsen) gesetzt wird und der Bruchteil wird auf 0 gesetzt.

[0443] Außer der Subtraktionseinheit **172** und der Multiplikationseinheit **174** wird der Hauptteil der Arbeit in dieser Einheit durch den Adressengenerator **175** durchgeführt, der tatsächlich der Zustandsautomat für die Einheit ist. Die Adressengenerierung wird durch zwei Faktoren beherrscht: bei einem gegebene Zyklus kann lediglich ein Zugriff auf den Bild-RAM-Speicher **11** durchgeführt werden, und bei einem gegebenen Zyklus kann lediglich ein Zugriff auf den Zwischenspeicher 1 **114** durchgeführt werden. Von den 75 verfügbaren Zyklen werden 3 Sätze von 16 Zyklen zum Lesen des Zwischenspeichers 1 verwendet. Die tatsächliche Verwendung ist 3 Sätze von 25 Zyklen, wobei 16 Lesevorgänge von 9 Wartezyklen gefolgt werden. Das ergibt eine Gesamtzahl von 27 verfügbaren Zyklen für 25 Schreibvorgänge (6 rot, 6 blau, 6 grün). Die bedeutet, dass den Randbedingungen genüge getan wird, wenn die Zeitpunkte der Schreibvorgänge in den Zwischenspeicher 1 mit den Wartezyklen des Resamplers **112** zusammen fallen.

10.2.11.1 Adressengenerierung für Zwischenspeicher 1

[0444] Sobald der Resampling-Prozess läuft, kümmert man sich nur noch um das Schreiben in den Zwischenspeicher 1 während des Zeitraums, in dem der Resampler **112** nicht aus ihm (dem Zwischenspeicher 1) liest. Da der Resampler 3 Sätze von 16 Lesevorgängen in jedem 75-Zyklen-Zeitraum aufweist, sind 27 Zyklen zum Schreiben verfügbar. Wenn der Resampler nicht läuft, will man den Zwischenspeicher 1 so schnell wie möglich hoch laden, was einen Schreibvorgang in den Zwischenspeicher 1 **114** in jedem Zyklus bedeutet. Die Adressengenerierung für den Zwischenspeicher läuft folglich auf einem Zustandsautomaten ab, der diese beiden Fälle berücksichtigt. Immer dann, wenn ein Wert aus dem Bild-RAM-Speicher **11** geladen wird, wird der ange-

passte Wert auf die entsprechende Farbe in dem Zwischenspeicher 1 einen Zyklus später geschrieben.

[0445] Die Adressengenerierung für den Zwischenspeicher 1 umfasst deshalb einen einzelnen Adresszähler für jeden der roten, blauen und grünen Unter-Zwischenspeicher. Die Anfangsadresse für RedAdr, BlueAdr, GreenAdr ist 0 am Start jeder Zeile in jedem Fall, und nach jedem Schreibvorgang in den Zwischenspeicher 1 erhöht sich die Adresse um 1, mit einem Überlauf bei 36 oder 78 je nachdem, ob der Zwischenspeicher, der beschrieben wird, rot, grün oder blau ist. Es wird nicht in alle Farben während jeder 75-Zyklen-Periode geschrieben. Eine Spalte grün erfordert in typischer Weise ein Auffüllen mit der doppelten Geschwindigkeit von z. B. rot oder blau.

[0446] Die Logik ist in dem nachfolgenden Pseudocode dargestellt:

```
If the color to write is Red
   Write to Red Buffer1 at RedAdr
   RedAdr = RedAdr + 1 mod 36
Else
If the color to write is Blue
   Write to Blue Buffer1 at BlueAdr
   BlueAdr = BlueAdr + 1 mod 36
Else
If the color to write is Green
   Write to Green Buffer1 at GreenAdr
   GreenAdr = GreenAdr + 1 mod 78
EndIf
```

10.2.11.2 Adressengenerierung für Bild-RAM-Speicher

[0447] Jede Ebene kann in einer von zwei Orientierungen gelesen werden – gedreht um 0 Grad oder um 90 Grad (gegen den Uhrzeigersinn). Dies überträgt sich wirksam als zeilenweiser oder spaltenweiser Lesezugriff auf das planare Bild. Darüber hinaus ermöglichen wir eine Kantenpixelreplizierung oder konstante Farbe für Lesevorgänge außerhalb der Bildgrenzen sowie eine Bildumhüllung (Image Wrapping) für Druckformate wie Hochformat **31**.

[0448] An den Beginn jeder Druckzeile müssen wir den Bild-RAM-Speicher **11** lesen, um den Zwischenspeicher 1 **114** so schnell wie möglich aufzuladen. Dies entspricht einem einzelnen Zugriff auf einen Abtastpunkt in jedem Zyklus. Ein Resampling kann nur auftreten, sobald 5 Spalten geladen worden sind, was 5 Spalten aus 6, 6 und 13 Abtastpunkten für eine Gesamtzahl von 125 Zyklen bedeutet. Zusätzlich ein extra Zyklus für den letzten Wert, der in den Zwischenspeicher 1 **114** geschrieben werden muss, nachdem er aus dem Bild-RAM-Speicher **11** geladen wurde. Um das Zählen zu vereinfachen, runden wir auf 128 Zyklen auf.

[0449] Nach den ersten 128 Zyklen tritt das Überprüfen der Anforderung, die nächste Spalte der Abtastpunkte für jede der 3 Farben zu laden, alle 75 Zyklen auf, wobei die entsprechenden Abtastpunkte während der nachfolgenden 75 Zyklen geladen werden. Die Anfangseinstellung jedoch, ob während des ersten Satzes von 75 Zyklen zu laden ist, ist für jede Farbe immer 1. Dies ermöglicht es, dass die letzte 6. Spalte jeder Farbe innerhalb des Zwischenspeichers 1 gefüllt wird.

[0450] An dem Ende jedes 75-Zyklen-Zeitraums wird der KernelXCarryOut-Merker aus jeder Farbebene des Kern-Adressengenerators in dem Resampler **112** überprüft, um zu bestimmen, ob die nächste Spalte an Abtastpunkten gelesen werden soll. In ähnlicher Weise startet der AdvanceLine-Impuls den Prozess auf der folgenden Zeile neu, wenn der KernelYCarryout-Merker gesetzt ist.

[0451] Da aus jedem "Lesen" tatsächlich 6 oder 13 Lesevorgänge werden, um eine Spalte im Zwischenspeicher 1 zu füllen, behält man eine Anfangsposition bei, um zu dem nächsten "Lesen" fortzuschreiten. Wir bewahren auch einen Koordinatenwert auf, um die Generierung von Koordinaten außerhalb der Grenzen zu ermöglichen, um Kantenpixelreplizierung, konstante Farbe und Bildumhüllung zu ermöglichen.

[0452] Wir berücksichtigen, dass das aktive Bild **180** innerhalb einer bestimmten Grenze liegt, wobei gewisse Tätigkeiten ausgeführt werden müssen, wenn Koordinaten außerhalb des aktiven Bereichs liegen. Die Koordinaten können entweder vor dem Bild, innerhalb des Bildes oder nach dem Bild liegen, sowohl hinsichtlich der Zeilen als auch der Pixel. Dies ist in Fig. 67 gezeigt, obwohl der Raum außerhalb des aktiven Bereichs zum Zwecke der Klarheit übertrieben wurde:
Es ist anzumerken, dass, da wir (0, 0) als den Start der Koordinatengenerierung verwenden, MaxPixel und MaxLine ebenfalls Pixel- und Zeilenzähler sind. Da die Adressengenerierung jedoch von den Kern-Stellenüberträgen und den AdvanceLine-Impulsen von der MJI **15** ausgeführt wird, sind diese äußeren Grenzen nicht erforderlich. Die Adressengenerierung fährt für eine Zeile einfach fort, bis der AdvanceLine-Impuls empfangen wird, und kann Kantenreplizierung, konstante Farben für außerhalb der Grenzen oder Bildpixelumhüllung umfassen.

[0453] Wenn wir eine Adresse, Adr, der aktuellen Abtastung haben, und wir wollen uns zu der nächsten Abtastung weiter bewegen, entweder auf der nächsten Zeile oder auf derselben Zeile, ändert sich die Koordinate der Abtastung wie erwartet, die Art und Weise, in der wir die Adresse verändern, hängt jedoch davon ab, ob wir um das aktive Bild herum hüllen, und falls notwendig, Kantenpixelreplizierung herstellen müssen.

[0454] Wenn keine Umhüllung des Bildes vorhanden ist, (d. h. alle Druckformate außer Hochformat **31**), führen wir die Aktionen in Tabelle 38 durch, sobald wir in der Zeile oder Pixel voranschreiten. Um ein Bild um 90 Grad zu drehen, vertauscht die CPU **10** einfach die ΔLine- und ΔPixel-Werte.

[0455] Betrachtet man Tabelle 38, so ist das einzige Mal, dass ADR sich ändert, um Δ Pixel, wenn PixelSense 0 ist, und um ΔLine, wenn LineSense 0 ist. Durch Befolgen dieser einfachen Regeln ist Adr für die Kantenpixelreplizierung gültig. Natürlich kann, wenn eine konstante Farbe für die Koordinaten außerhalb der Grenzen gewünscht ist, dieser Wert anstatt des in den entsprechenden Adressen gespeicherten Wertes ausgewählt werden.

[0456] Um Umhüllung zu ermöglichen, vergleichen wir einfach die vorherige Abfrage (-, 0, +) für Zeile und Pixel mit der neuen Abfrage.

Tabelle 38. Durchzuführende Aktionen beim Voranschreiten in Pixel oder Zeilen

Zeile[a]	Pixel[b]	Pixelveränderung	Zeilenveränderung
-	-		
-	0	Adr = Adr + Δ Pixel	
-	+		
0	-		Adr = Adr + Δ Line
0	0	Adr = Adr + Δ Pixel	Adr = Adr + Δ Line
0	+		Adr = Adr + Δ Line
+	-		
+	0	Adr = Adr + Δ Pixel	
+	+		

a. Wir vergleichen die aktuelle Zeilenordinate mit ActiveStartLine und AcitveEndLine.
Wenn Zeile < ActiveStartLine, rufen wir den Wert "-" auf.
Wenn ActiveStartLine ≤ ActiveEndLine, rufen wir den Wert "0" auf.

Wenn ActiveEndLine ≤ Zeile, rufen wir den Wert "+" auf.
b. Wir vergleichen die aktuelle Pixelordinate mit ActiveStartPixel und AcitveEndPixel.
Wenn Pixel < ActiveStartPixel, rufen wir den Wert "-" auf.
Wenn ActiveStartPixel ≤ Line < ActiveEndPixel, rufen wir den Wert "0" auf.
Wenn ActiveEndPixel ≤ Pixel, rufen wir den Wert "+" auf.

[0457] Wenn die Abfrage "-" ist, verwenden wir das Fortschreiten, wie in Tabelle 38 beschrieben, wenn die Ordinate jedoch außerhalb der Grenzen liegt (d. h. Bewegen von 0 nach +), aktualisieren wir Adr mit einem neuen Wert, der nicht auf einem Δ (delta) basiert. Unter der Annahme, dass wir die Startadresse für die aktuelle Zeile behalten, so dass wir auf den Beginn der nächsten Zeile weitergehen können, sobald die aktuelle Zeile erzeugt wurde, können wir das Nachfolgende ausführen:

- Wenn eine Veränderung in Pixel auftritt, und die Pixelabfrage ändert sich von 0 auf + (wodurch angegeben wird, dass wir über den Rand des Bildes hinausgegangen sind), ersetzen wir Adr mit LineStartAdr und ersetzen Pixel mit ActiveStartPixel. Die Zeile bleibt dieselbe.
- Wenn es eine Veränderung in der Zeile gibt, und die Zeilenabfrage ändert sich von 0 auf + (wodurch angegeben wird, dass wir über den Rand des Bildes hinausgegangen sind), subtrahieren wir DeltaColumn von Adr und ersetzen Zeile mit ActiveStartLine. Pixel bleibt dasselbe. DeltaColumn ist der Adressenversatz für das Erzeugen der Adresse von (Pixel, ActiveStartLine) aus (Pixel, ActiveEndLine-1).

[0458] Die Logik für das Laden der eingestellten Anzahl von Abtastungen (entweder 6 oder 13 je nach Farbe ist in dem nachfolgenden Pseudocode dargestellt.

```
line = FirstSampleLine
pixel = FirstSamplePixel
adr = FirstSampleAdr
Do N times (6 or 13)
   oldPixelSense = PixelSense (pixel)
   oldLineSense = LineSense (gLine)
   inActive = ((oldLineSense == InActive) AND (oldPixelSense
   ==InActive))
   If ((NOT inactive) AND UseConstant)
      Sample = ConstantColor
   Else
      Sample = Fetch(adr)
   EndIf
   line = line + 1
   If ((LineSense(line) == "+") AND wrapImage)
      adr = adr - DeltaColumn
      line = ActiveStartLine
   ElseIf ((LineSense(line) == "0") AND ((oldLineSense ==
   "0"))
      adr = adr + DeltaLine
   EndIf
EndDo
```

[0459] Die Einstellung für solche Variabeln wie FirstSampleLine, FirstSamplePixel und FirstSampleAdr ist in dem Adressengeneratorabschnitt, der auf die Stellenübertrag-Merker aus dem Kernadressengenerator reagiert, sowie auf die AdvanceLine-Impulse aus der MJI. Die Logik für diesen Teil der Adressengenerierung ist in dem nachfolgenden Pseudocode dargestellt:

```
FirstSamplePixel = 0
FirstSampleLine = 0

FirstSampleAdr = FirstLineSampleAdr = ActiveStartAddress
```

```
count = 0
Do Forever
   If ((KernelXCarryOut) OR (AdvanceLine AND KernelYCarryOut)
   OR (count <5))
      Do N Samples for this color plane (see pseudocode above)
   EndIf
   oldPixelSense = PixelSense (FirstSamplePixel)
   oldLineSense = LineSense (FirstSampleLine)
   If (AdvanceLine AND KernelYCarryOut)
      count = 0
      FirstSampleLine = FirstSampleLine + 1
      FirstSamplePixel = 0
      If ((LineSense (FirstSampleLine) == "+") AND warpImage)
          FirstLineSampleAdr = StartAddress
          FirstSampleLine = ActiveStartLine
      ElseIf   ((LineSense   (FirstSampleLine)   ==   "0")   AND
      (oldLineSense == "0"))
          FirstLineSampleAdr = FirstLineSampleAdr + DeltaLine
      EndIf
      FirstSampleAdr = FirstLineSampleAdr
   ElseIf (KernelXCarryOut OR (count < 5))
      FirstSamplePixel = FirstSamplePixel + 1
      count = count + 1
      If ((PixelSense(FirstSamplePixel) == "+" AND wrapImage)
          FirstSampleAdr = FirstLineSampleAdr
          FirstSamplePixel = ActiveStartPixel
      ElseIf   ((PixelSense(FirstSamplePixel)   ==   "0")   AND
      (oldPixelSense == "0"))
          FirstSampleAdr = FirstLineSampleAdr + DeltaPixel
      EndIf
   EndIf
EndDo
```

10.2.11.3 Zusammenfassung der Register

[0460] Es gibt eine Anzahl von Registern, die eingestellt werden müssen, bevor ein Bild gedruckt wird. Diese sind hier in Tabelle 39 zusammen gefasst. Um ein Bild um 90 Grad zu drehen, ändert man einfach die DeltaLine und DeltaPixel-Werte und stellt einen neuen DeltaColumn-Wert bereit.

Tabelle 39. Register, die von dem Aufrufer vor dem Drucken eingestellt werden müssen.

Registername	**Beschreibung**
Bildzugriffsparameter	
WrapImage	Lesevorgänge der Bildteile, um das Bild zu replizieren, wenn außerhalb der Bildgrenzen
UseConstant	Falls 0, Bildrandreplizierung oder Umhüllung tritt bei Lesevorgängen außerhalb der Bildgrenzen auf Wenn 1, wird eine konstante Farbe zurückgegeben.
Rot	
ActiveStartAddressR	Die Adresse des roten Abtastpunkts (ActiveStartPixel, ActiveStartLine) in dem Bild-RAM-Speicher
ActveStartLineR	Die erste gültige Zeile für das Bild im roten Raum (bzgl. der Zeile 0)
ActiveEndeLineR	Die erste Zeile außerhalb der Grenzen für das Bild im roten Raum
ActiveStartPixelR	Das erste gültige Pixel für das Bild im roten Raum (bzgl. des Pixels 0)
ActiveEndPixelR	Das erste Pixel außerhalb der Grenzen für das Bild im roten Raum
DeltaLineR	Der Betrag, der zur aktuellen Adresse hinzuzufügen ist, um sich von einer Zeile in die nächste Zeile im roten Raum fortzubewegen
DeltaPixelR	Der Betrag, der zur aktuellen Adresse hinzuzufügen ist, um sich von einem Pixel zu dem nächsten auf derselben Zeile im roten Raum fortzubewegen
DeltaColumnR	Der Betrag, der zur aktuellen Adresse hinzuzufügen ist, um sich von einem Pixel in der letzten Zeile des aktiven Bildbereichs zu demselben Pixel auf der ersten Zeile des aktiven Bildbereichs im roten Raum fortzubewegen.
ConstantColorR	Zu verwendender roter Farbwert, wenn die Adresse außerhalb der Grenzen und UseConstant=1
Grün	
ActiveStartAddressG	Die Adresse des grünen Abtastpunkts (ActiveStartPixel, ActiveStartLine) in dem Bild-RAM-Speicher

ActveStartLineG	Die erste gültige Zeile für das Bild im grünen Raum (bzgl. der Zeile 0)
ActiveEndeLineG	Die erste Zeile außerhalb der Grenzen für das Bild im grünen Raum
ActiveStartPixelG	Das erste gültige Pixel für das Bild im grünen Raum (bzgl. des Pixels 0)
ActiveEndPixelG	Das erste Pixel außerhalb der Grenzen für das Bild im grünen Raum
DeltaLineG	Der Betrag, der zur aktuellen Adresse hinzuzufügen ist, um sich von einer Zeile in die nächste Zeile im grünen Raum fortzubewegen
DeltaPixelG	Der Betrag, der zur aktuellen Adresse hinzuzufügen ist, um sich von einem Pixel zu dem nächsten auf derselben Zeile im grünen Raum fortzubewegen
DeltaColumnG	Der Betrag, der zur aktuellen Adresse hinzuzufügen ist, um sich von einem Pixel in der letzten Zeile des aktiven Bildbereichs zu demselben Pixel auf der ersten Zeile des aktiven Bildbereichs im grünen Raum fortzubewegen.
ConstantColorG	Zu verwendender grüner Farbwert, wenn die Adresse außerhalb der Grenzen und UseConstant=1
Blau	
ActiveStartAddressB	Die Adresse des blauen Abtastpunkts (ActiveStartPixel, ActiveStartLine) in dem Bild-RAM-Speicher
ActveStartLineB	Die erste gültige Zeile für das Bild im blauen Raum (bzgl. der Zeile 0)
ActiveEndeLineB	Die erste Zeile außerhalb der Grenzen für das Bild im blauen Raum
ActiveStartPixelB	Das erste gültige Pixel für das Bild im blauen Raum (bzgl. des Pixels 0)
ActiveEndPixelB	Das erste Pixel außerhalb der Grenzen für das Bild im blauen Raum
DeltaLineB	Der Betrag, der zur aktuellen Adresse hinzuzufügen ist, um sich von einer Zeile in die nächste Zeile im blauen Raum fortzubewegen
DeltaPixelB	Der Betrag, der zur aktuellen Adresse hinzuzufügen ist, um sich von einem Pixel zu dem nächsten auf derselben Zeile im blauen Raum fortzubewegen
DeltaColumnB	Der Betrag, der zur aktuellen Adresse hinzuzufügen ist, um sich von einem Pixel in der letzten Zeile des aktiven Bildbereichs zu demselben Pixel auf der ersten Zeile des aktiven Bildbereichs im blauen Raum fortzubewegen.
ConstantColorB	Zu verwendender blauer Farbwert, wenn die Adresse außerhalb der Grenzen und UseConstant=1
Weißabgleich und Bereichserweiterungsparameter	
RedLowThreshold	8-Bit Wert subtrahiert von den roten Eingabewerten
GreenLowThreshold	8-Bit Wert subtrahiert von den grünen Eingabewerten

BlueLowThreshold	8-Bit Wert subtrahiert von den blauen Eingabewerten
RedScaleFactor	8-Bit Skalierungsfaktor, der für Bereichserweiterung der roten Pixel verwendet wird
GreenScaleFactor	8-Bit Skalierungsfaktor, der für Bereichserweiterung der grünen Pixel verwendet wird
BlueScaleFactor	8-Bit Skalierungsfaktor, der für Bereichserweiterung der blauen Pixel verwendet wird

11. REFERENZEN

[1] Silverbrook Research, 1998, Aufhentication of Consumables
[2] Silverbrook Research, 1998, Aufhentication Chip.

[0461] Obwohl die Erfindung unter Bezugnahme auf bestimmte Beispiele beschrieben wurde, wird es von dem einschlägigen Fachmann gewürdigt werden, dass sie in vielen anderen Formen verkörpert werden kann. Die nachfolgenden durchgezählten Abschnitte stellen dem Adressaten eine weitere Angabe des Umfangs der Erfindung bereit, obwohl andere neue und erfinderische Merkmale und Kombinationen von Merkmalen ebenfalls aus der hierin beschriebenen Offenbarung deutlich werden.

Patentansprüche

1. Verfahren zum Bereitstellen eines Bildes zum Drucken bei einer vorbestimmten zweiwertigen Rasterpunktauflösung, die einer vorbestimmten Auflösung mit gleichmäßigem Tonverlauf (continuous tone) entspricht, wobei das Verfahren die folgenden Schritte aufweist:
Empfangen eines ersten Datensatzes, der das Bild angibt, wobei der erste Datensatz aus einem planarisierten Bayer-Format besteht, das sich aus unterschiedlichen Farbebenen (**45**, **46**, **47**) zusammensetzt, wobei mindestens eine der Farbebenen (**46**) eine Auflösung aufweist, die sich von der der anderen Farbebene oder -ebenen (**45**, **47**) unterscheidet;
Durchführen einer Bildrekonstruktion und erneuten Abtastung (**64**) jeder Farbebene des ersten Datensatzes, um eine zweiten Datensatz der vorbestimmten Auflösung mit gleichmäßigem Tonverlauf zu erzeugen, die größer ist als die Auflösung jeder der Farbebenen für den ersten Datensatz;
Umwandeln (**67**) des zweiten Datensatzes in einen dritten Datensatz der vorbestimmten zweiwertigen Rasterpunktauflösung, die größer ist als die vorbestimmte Auflösung mit gleichmäßigem Tonverlauf des zweiten Datensatzes; und
zur Verfügung Stellen des dritten Datensatzes für einen Drucker (**69**) in der vorbestimmten zweiwertigen Rasterpunktauflösung.

2. Verfahren nach Anspruch 1, wobei der erste Datensatz in einem RGB-Format (Rot, Grün und Blau) vorhanden ist und der Drucker auf CMY-Format (Cyan, Magenta und Gelb) reagiert, wobei das Verfahren den Schritt des Umwandelns (**66**) des zweiten Datensatzes von einem RGB-Format in ein CMY-Format umfasst.

3. Verfahren nach Anspruch 1, das den Schritt des Schärfens (**65**) des zweiten Datensatzes umfasst.

4. Verfahren nach Anspruch 1, wobei der erste Datensatz von einer Sensorvorrichtung ermittelt wird und das Verfahren den Schritt des Abgleichens des ersten Datensatzes aufgrund der Nichtlinearitäten in der Sensorvorrichtung umfasst.

5. Verfahren nach Anspruch 4, wobei der Schritt des Abgleichens das Umwandeln des ersten Datensatzes aus einer Vielzahl von x Bitproben in eine Vielzahl von y Bitproben umfasst, wobei $x > y$ gilt.

6. Verfahren nach Anspruch 5, wobei $x = 10$ und $y = 8$ ist.

7. Verfahren nach Anspruch 1, das die folgenden weiteren Schritte aufweist:
Bestimmen der m % dunkelsten Pixel und der n % hellsten Pixel für den ersten Datensatz;
Anpassen des ersten Datensatzes, um die m % dunkelsten Pixel auszugleichen; und
Anpassen des ersten Datensatzes, um die n % hellsten Pixel auszugleichen.

8. Verfahren nach Anspruch 1, das den zusätzlichen Schritt des Anpassens des ersten Datensatzes umfasst, um einen vorbestimmten Weißabgleich (**62**) bereitzustellen.

9. Verfahren nach Anspruch 1, das den zusätzlichen Schritt des Anpassens des ersten Datensatzes umfasst, um eine vorbestimmte Bereichserweiterung (**63**) bereitzustellen.

10. Verfahren nach Anspruch 8 oder 9, wobei die Auflösung jeder Farbebene für den ersten Datensatz erhöht wird, während dieselbe räumliche Auflösung beibehalten wird.

11. Verfahren nach Anspruch 1, wobei der erste Datensatz wahlweise angepasst wird, um das Bild in einer vorbestimmten Rotationsorientierung bereitzustellen.

12. Vorrichtung zum Bereitstellen eines Bildes zum Drucken bei einer vorbestimmten zweiwertigen Rasterpunktauflösung, die einer vorbestimmten Auflösung mit gleichmäßigem Tonverlauf entspricht, wobei die Vorrichtung umfasst:
Eingabemittel zum Empfangen eines ersten Datensatzes, der das Bild angibt, wobei der erste Datensatz in einem planarisierten Bayer-Format vorliegt, das unterschiedliche Farbebenen (**45**, **46**, **47**) aufweist, wobei mindestens eine der Farbebenen (**46**) eine Auflösung aufweist, die unterschiedlich zu der der anderen Farbebene oder -ebenen (**45**, **47**) ist,
Abtastmittel zur Bildrekonstruktion und zum erneuten Abtasten (**64**) des ersten Datensatzes, um einen zweiten Datensatz der vorbestimmten Auflösung mit gleichmäßigem Tonverlauf zu erzeugen, die größer ist als die Auflösung jeder der Farbebenen des ersten Datensatzes;
Verarbeitungsmittel zum Umwandeln (**67**) des zweiten Datensatzes in einen dritten Datensatz der vorbestimmten zweiwertigen Rasterpunktauflösung, die größer ist als die vorbestimmte Auflösung mit gleichmäßigem Tonverlauf des zweiten Datensatzes; und
Ausgabemittel zum Bereitstellen des dritten Datensatzes (**69**) für einen Drucker zum Drucken in der vorbestimmten zweiwertigen Rasterpunktauflösung.

13. Vorrichtung nach Anspruch 12, wobei der erste Datensatz in einem RGB-Format (Rot, Grün und Blau) vorliegt und der Drucker auf ein CMY-Format (Cyan, Magenta und Gelb) reagiert, wobei die Verarbeitungsmittel den zweiten Datensatz von einem RGB-Format in ein CMY-Format umwandeln.

14. Vorrichtung nach Anspruch 12, die des Weiteren einen Schärfungsfilter zum Schärfen (**65**) des zweiten Datensatzes umfasst.

15. Vorrichtung nach Anspruch 12, wobei der erste Datensatz aus einer Sensorvorrichtung erhalten wird und die Eingabemittel den ersten Datensatz bezüglich Nichtlinearitäten in der Sensorvorrichtung abgleichen.

16. Vorrichtung nach Anspruch 15, wobei der Abgleich bezüglich Nichtlinearitäten das Umwandeln des ersten Datensatzes von einer Vielzahl von x Bitproben in eine Vielzahl von y Bitproben umfasst, wobei $x > y$ gilt.

17. Vorrichtung nach Anspruch 16, wobei $x = 10$ und $y = 8$ ist.

18. Verfahren nach Anspruch 12, wobei die Eingabemittel:
für den ersten Datensatz die m % dunkelsten Pixel und die n % hellsten Pixel bestimmen;
den ersten Datensatz anpassen, um die m % dunkelsten Pixel auszugleichen; und
den ersten Datensatz anpassen, um die n % hellsten Pixel auszugleichen.

19. Vorrichtung nach Anspruch 12, wobei die Eingabemittel den ersten Datensatz anpassen, um einen vorbestimmten Weißabgleich (**62**) bereitzustellen.

20. Vorrichtung nach Anspruch 12, wobei die Eingabemittel den ersten Datensatz anpassen, um eine vorbestimmte Bereichserweiterung (**63**) bereitzustellen.

21. Vorrichtung nach Anspruch 19 oder 20, wobei die Eingabemittel die Auflösung jeder Farbebene des ersten Datensatzes erhöhen, während dieselbe räumliche Auflösung beibehalten wird.

22. Vorrichtung nach Anspruch 12, wobei die Eingabemittel wahlweise den ersten Datensatz anpassen, um das Bild in einer vorbestimmten Rotationsorientierung bereitzustellen.

23. Kamera umfassend:
ein CCD-Array zum Bereitstellen eines Bayer-Bildes;
einen Drucker zum wahlweisen Bereitstellen eines gedruckten Bildes; und
eine Vorrichtung nach Anspruch 12 zum Empfangen des Bayer-Bildes und Beliefern des Druckers mit dem dritten Datensatz, so dass das gedruckte Bild erzeugt wird.

Es folgen 43 Blatt Zeichnungen

Anhängende Zeichnungen

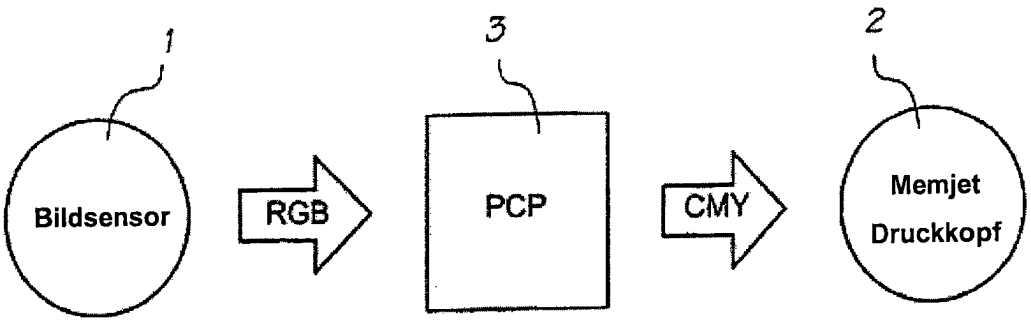


FIG. 1

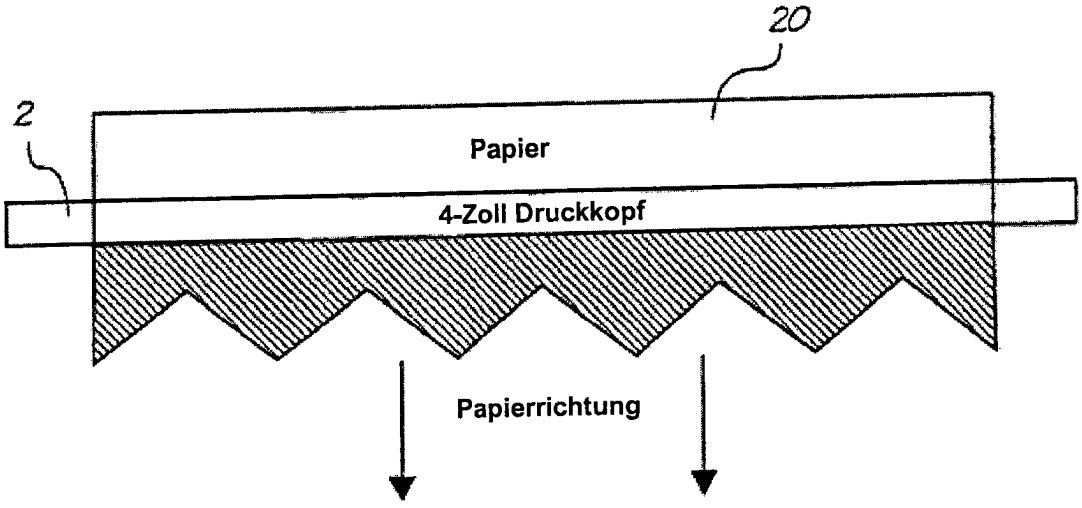


FIG. 4

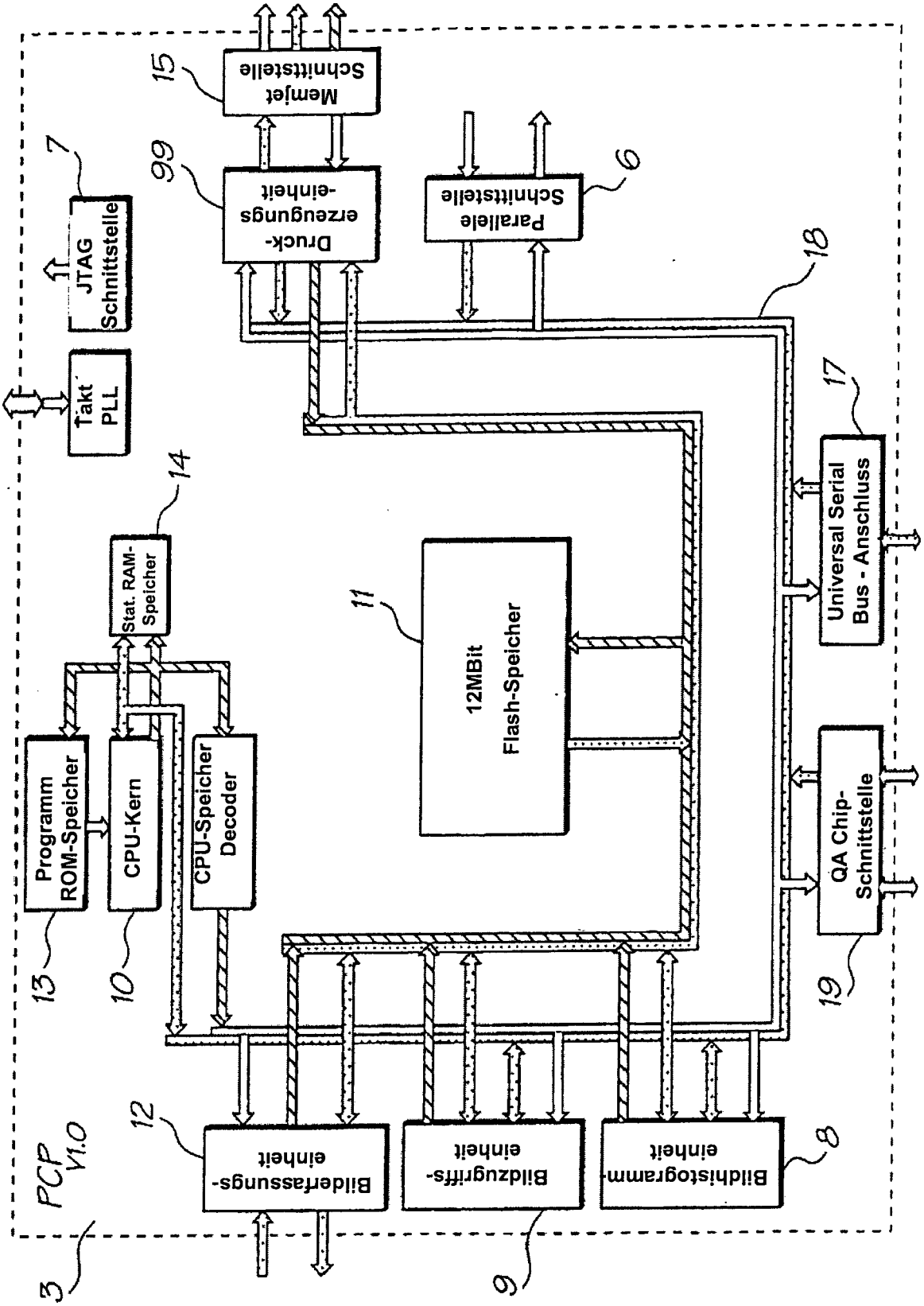

MemJet Schnittstelle
15
Druck-erzeugungs-einheit
99
Parallele Schnittstelle
6
7
JTAG Schnittstelle
Takt PLL
18
17
Universal Serial Bus - Anschluss
14
Stat. RAM-Speicher
11
12MBit Flash-Speicher
Programm ROM-Speicher
CPU-Kern
CPU-Speicher Decoder
13
10
QA Chip-Schnittstelle
19
PCP V1.0
12
Bilderfassungs-einheit
Bildzugriffs-einheit
Bildhistogramm-einheit
8
3
9

FIG. 2

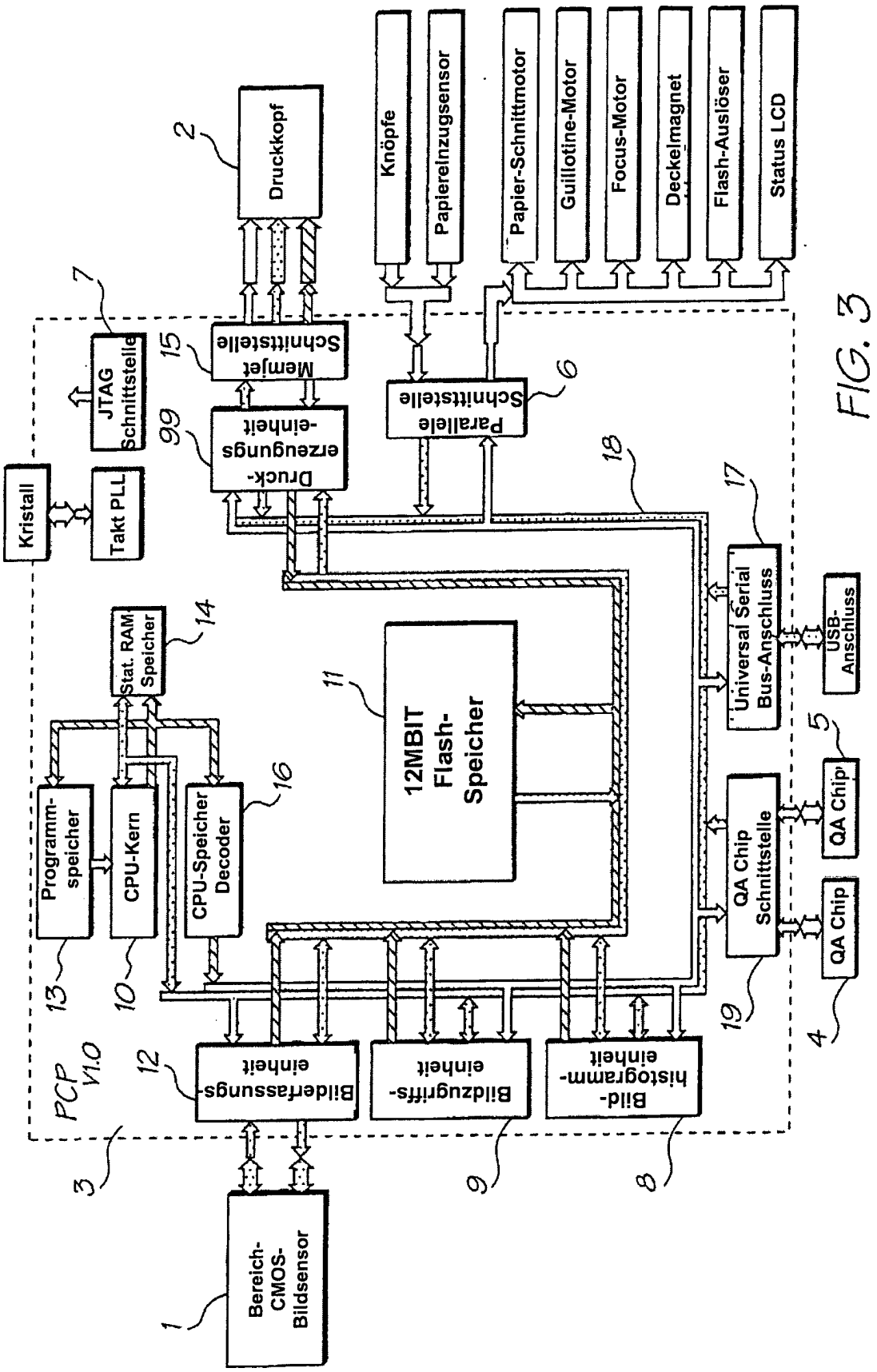

Bereich-CMOS-Bildsensor
1
PCP V1.0
3
Bilderfassungs-einheit
12
Bildzugriffs-einheit
9
Bild-histogramm-einheit
8
Programm-speicher
13
CPU-Kern
10
CPU-Speicher Decoder
16
Stat. RAM Speicher
14
12MBIT Flash-Speicher
11
Druck-erzeugungs-einheit
99
Memjet Schnittstelle
15
Parallele Schnittstelle
6
JTAG Schnittstelle
7
Kristall
Takt PLL
Druckkopf
2
Knöpfe
Papiereinzugsensor
Papier-Schnittmotor
Guillotine-Motor
Focus-Motor
Deckelmagnet
Flash-Auslöser
Status LCD
18
17
Universal Serial Bus-Anschluss
USB-Anschluss
QA Chip Schnittstelle
19
QA Chip
5
QA Chip
4

FIG. 3

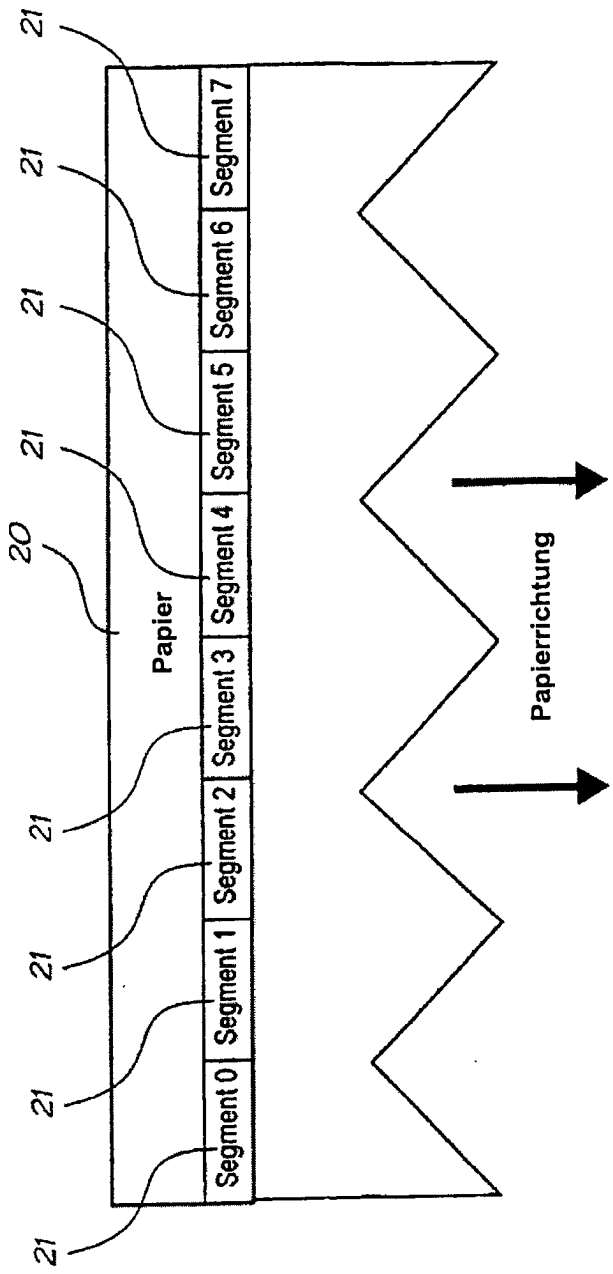

20
21
Papier
Segment 0
Segment 1
Segment 2
Segment 3
Segment 4
Segment 5
Segment 6
Segment 7
Papierrichtung
1 Segment
= Zoll
= 12700 µm

FIG. 5

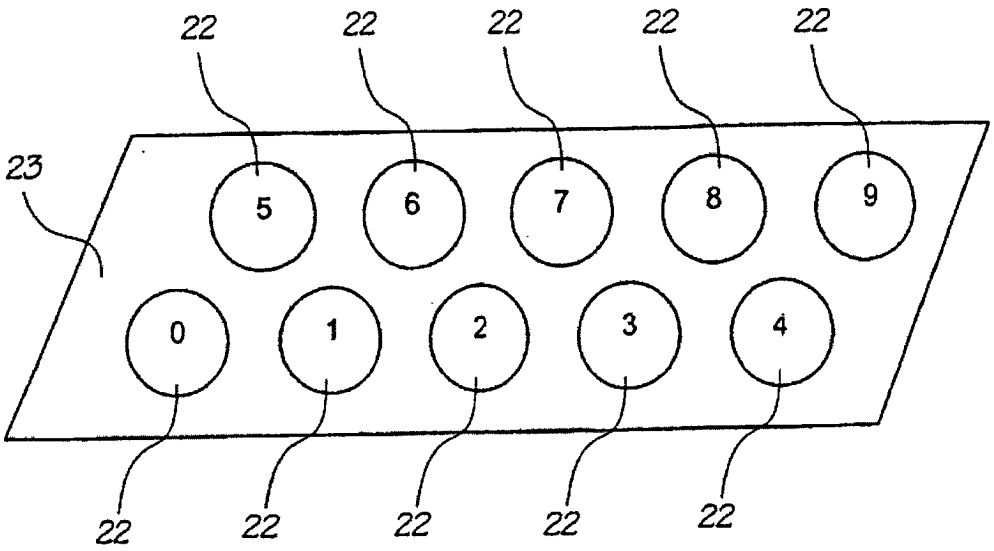

22
22
22
22
22
23
5
6
7
8
9
0
1
2
3
4
22
22
22
22
22

FIG. 6

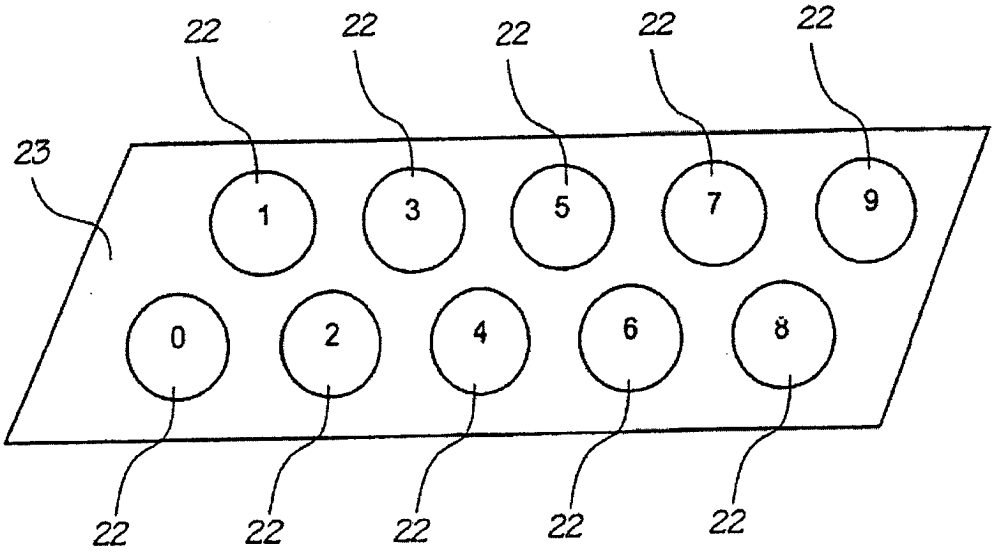

22
22
22
22
22
23
1
3
5
7
9
0
2
4
6
8
22
22
22
22
22

FIG. 7

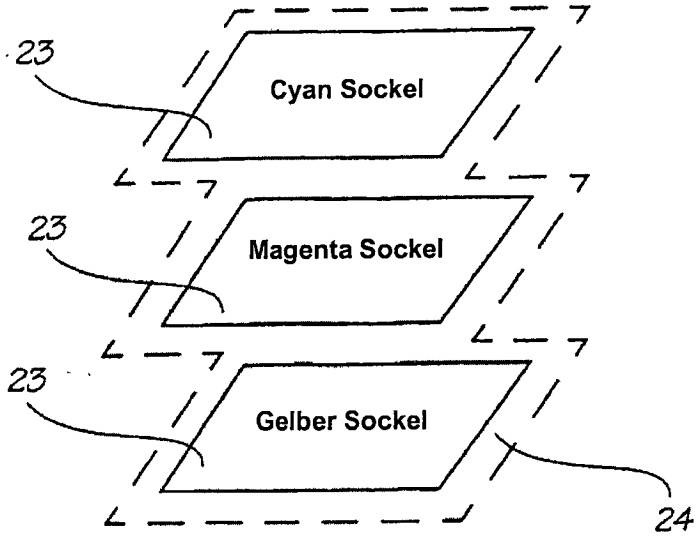

23
Cyan Sockel
23
Magenta Sockel
23
Gelber Sockel
24

FIG. 8

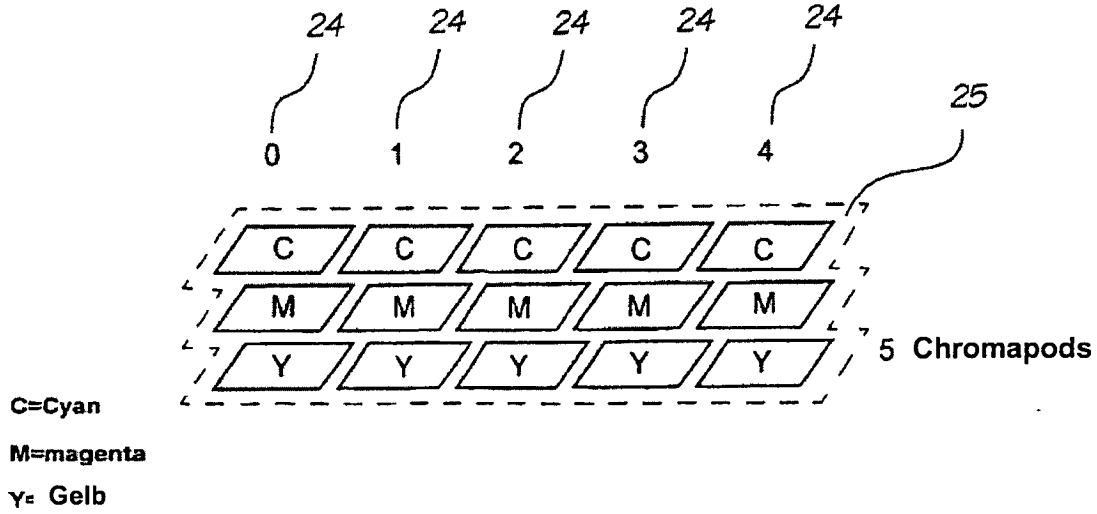

24
24
24
24
24
25
0
1
2
3
4
C
C
C
C
C
M
M
M
M
M
Y
Y
Y
Y
Y
5 Chromapods
C=Cyan
M=magenta
Y= Gelb

FIG. 9

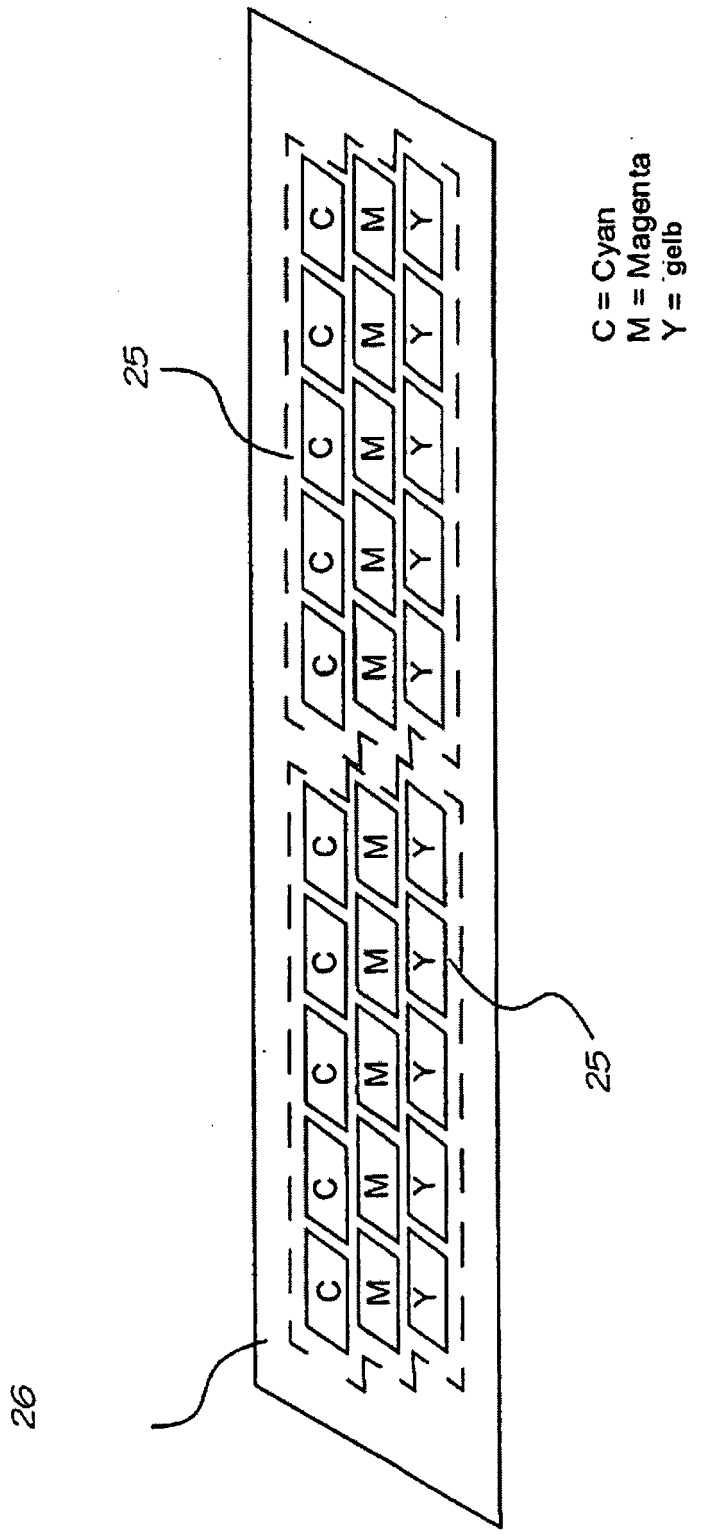


FIG. 10

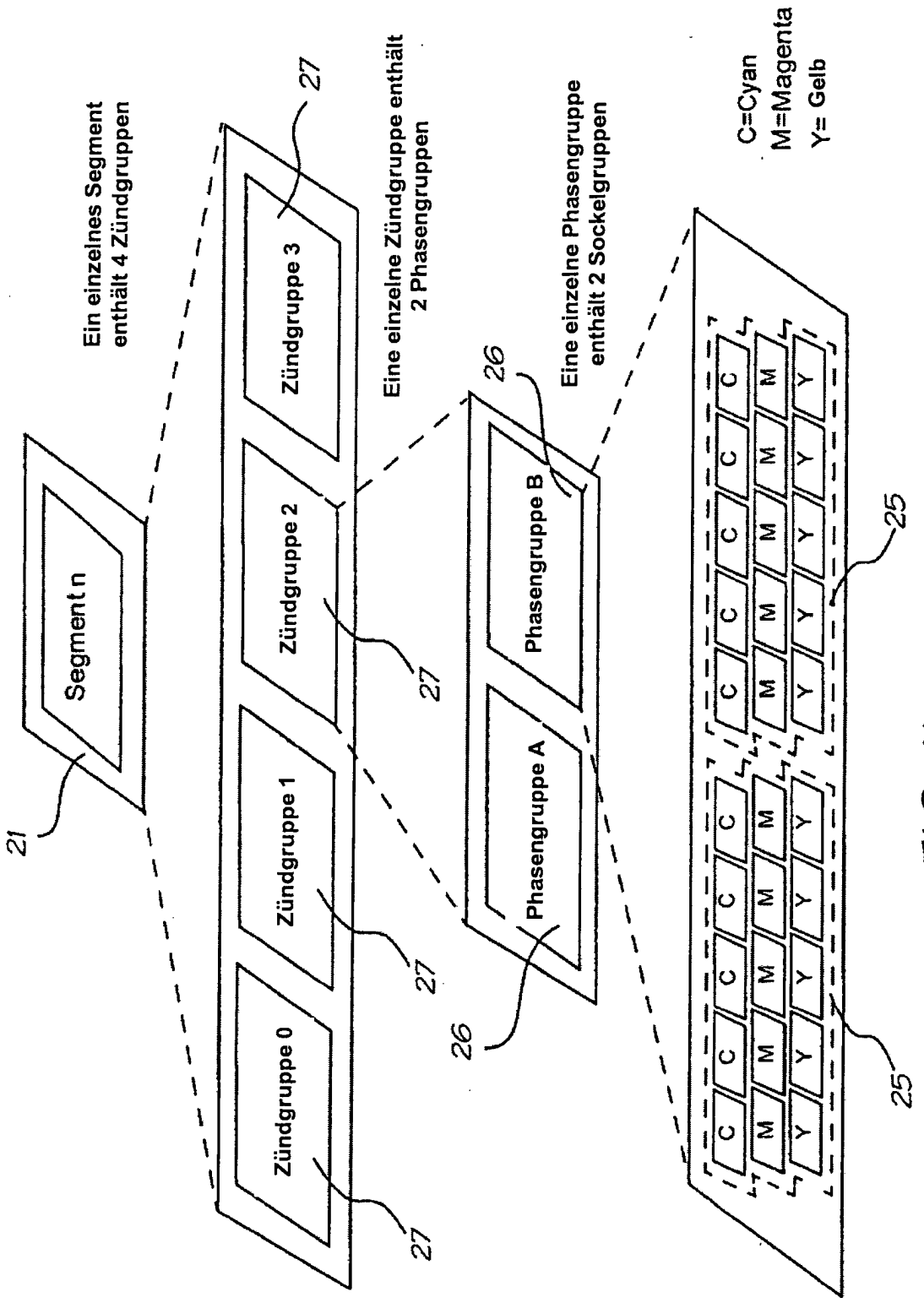

Ein einzelnes Segment enthält 4 Zündgruppen
Eine einzelne Zündgruppe enthält 2 Phasengruppen
Eine einzelne Phasengruppe enthält 2 Sockelgruppen
C=Cyan
M=Magenta
Y= Gelb
Segment n
Zündgruppe 0
Zündgruppe 1
Zündgruppe 2
Zündgruppe 3
Phasengruppe A
Phasengruppe B
21
25
26
27

FIG. 11

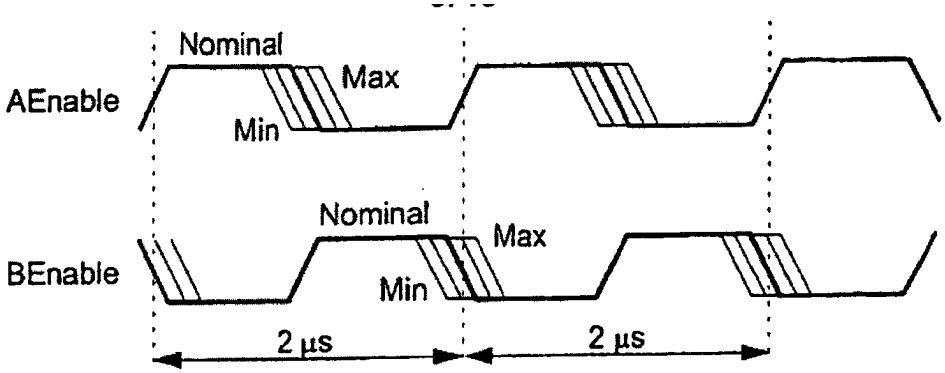

AEnable
Nominal
Max
Min
BEnable
Nominal
Max
Min
2 µs
2 µs

FIG. 12

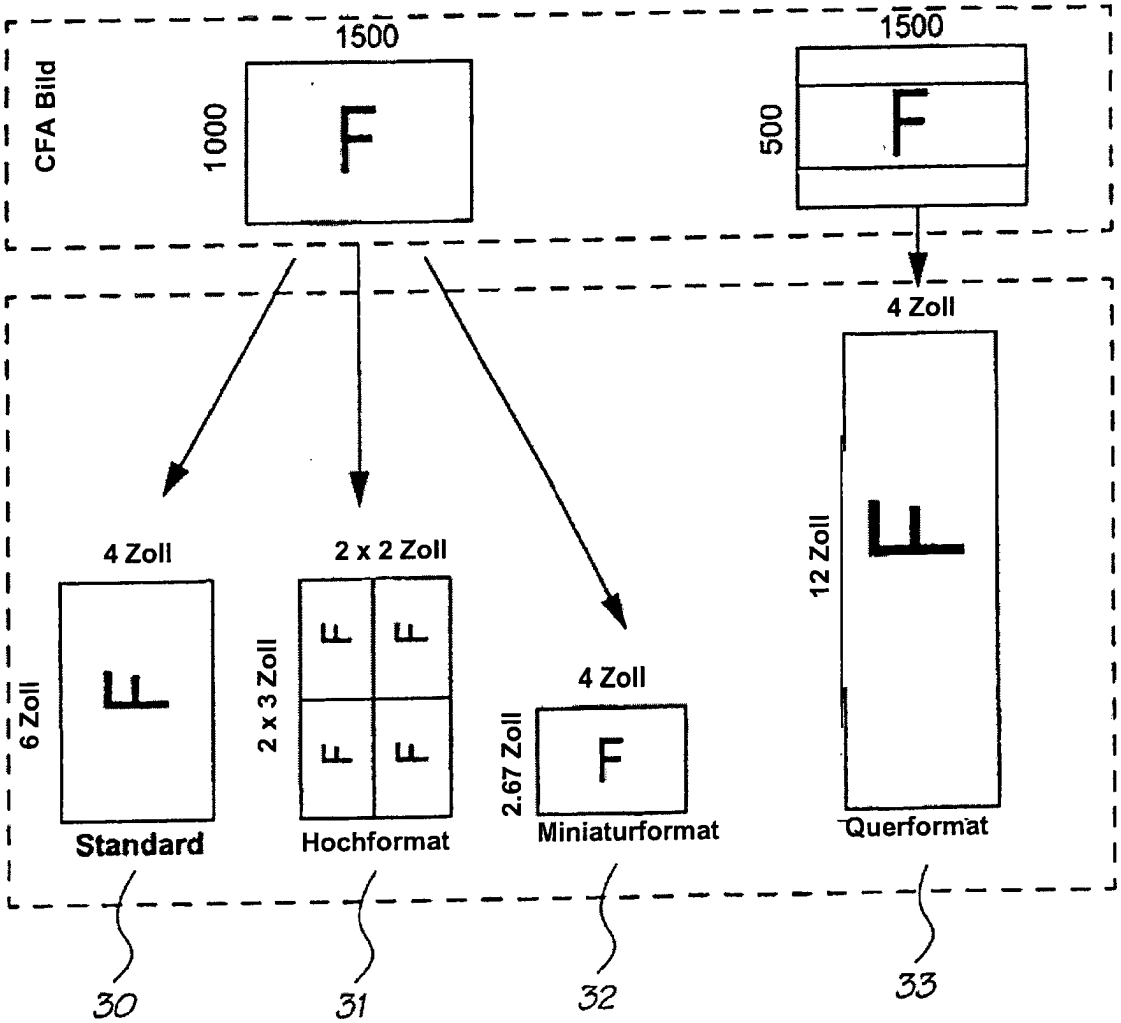

CFA Bild
1500
1000
F
1500
500
F
4 Zoll
12 Zoll
4 Zoll
6 Zoll
Standard
2 x 2 Zoll
2 x 3 Zoll
Hochformat
4 Zoll
2.67 Zoll
Miniaturformat
Querformat
30
31
32
33

FIG. 13

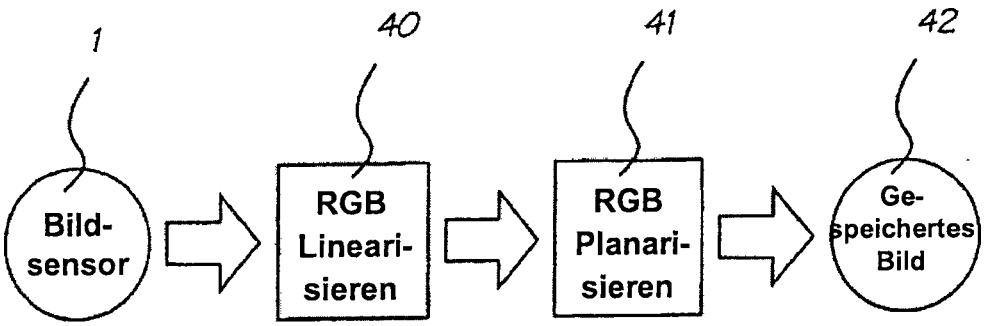

1
40
41
42
Bild-sensor
RGB Linearisieren
RGB Planarisieren
Gespeichertes Bild

FIG. 14

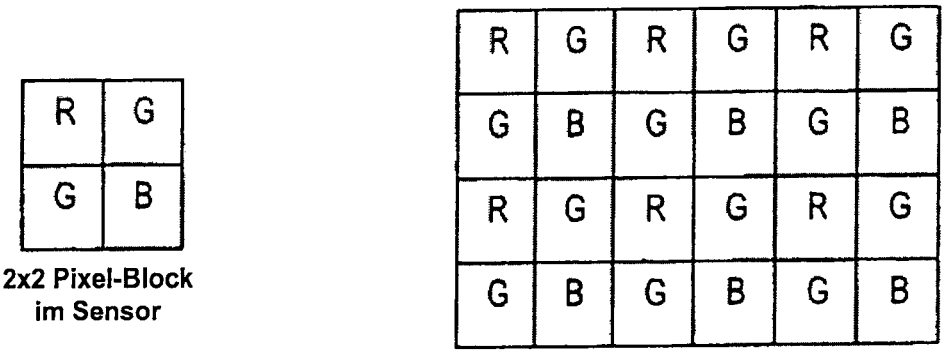

R G
G B
2x2 Pixel-Block im Sensor
R G R G R G
G B G B G B
R G R G R G
G B G B G B
Bayer-Mosaik einiger 2x2 Pixelblöcke

FIG. 15

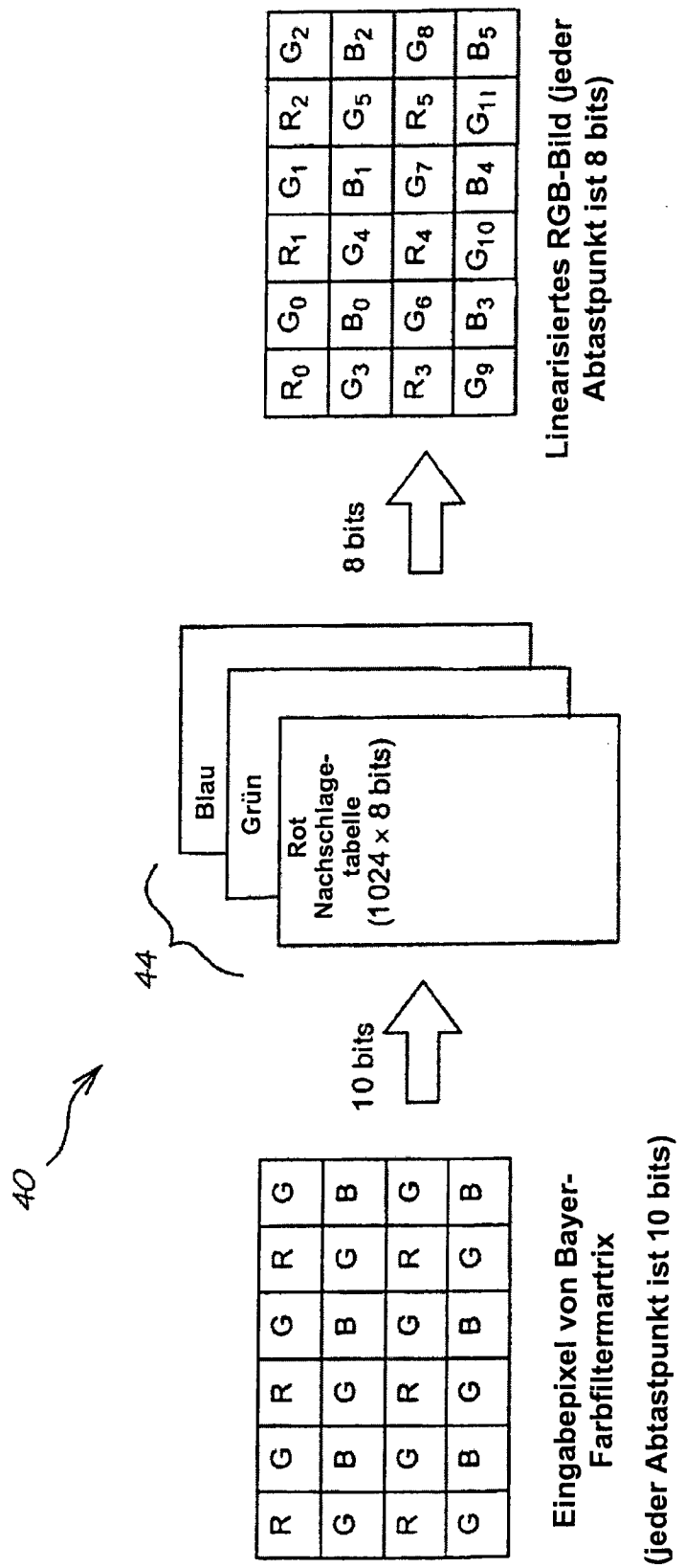

40
44
Blau
Grün
Rot
Nachschlage-
tabelle
(1024 x 8 bits)
10 bits
8 bits
Eingabepixel von Bayer-
Farbfiltermartrix
(jeder Abtastpunkt ist 10 bits)
Linearisiertes RGB-Bild (jeder
Abtastpunkt ist 8 bits)

FIG. 16

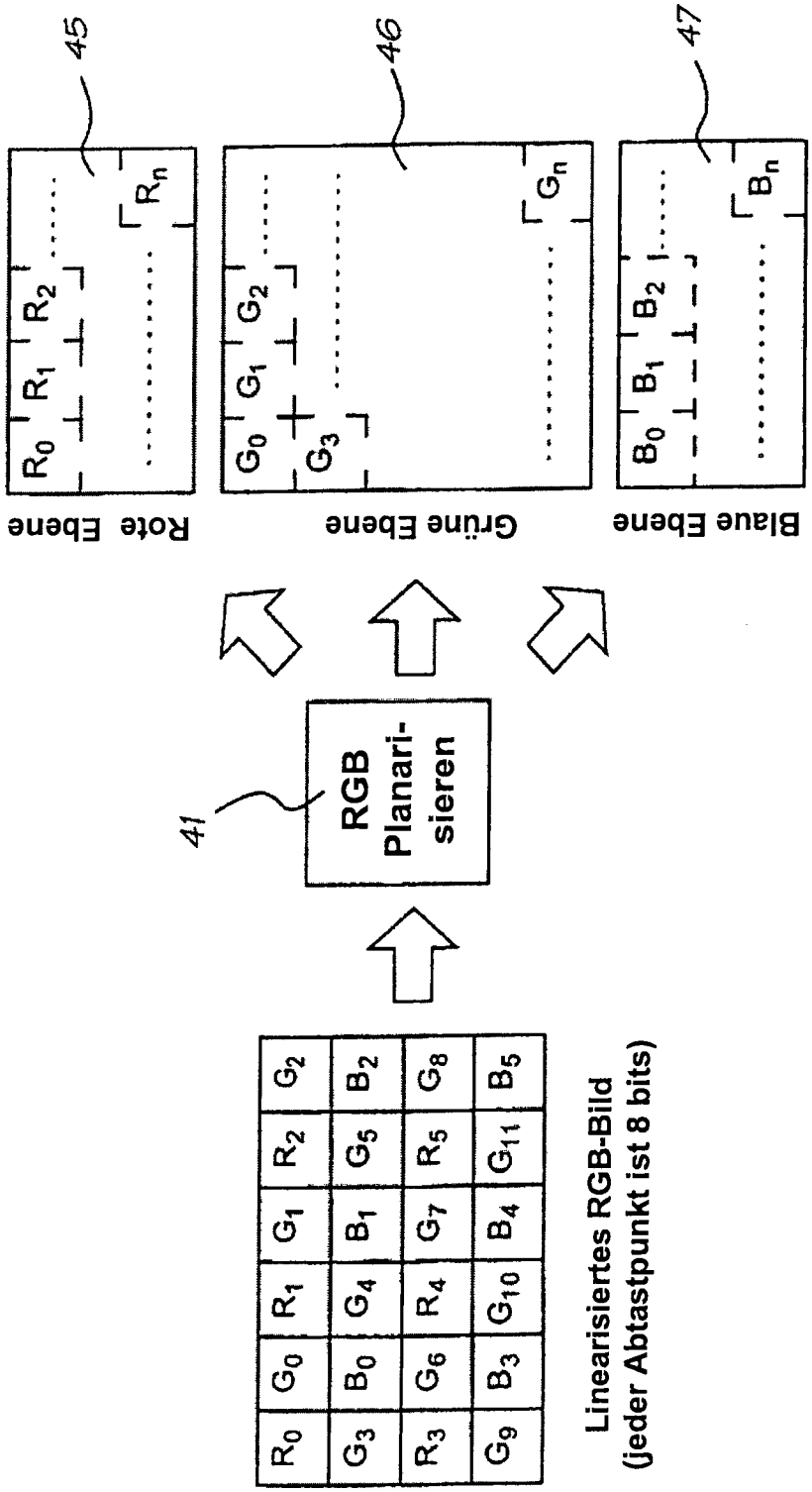

Linearisiertes RGB-Bild
(jeder Abtastpunkt ist 8 bits)
41
RGB Planari-sieren
45
46
47
Rote Ebene
Grüne Ebene
Blaue Ebene

FIG. 17

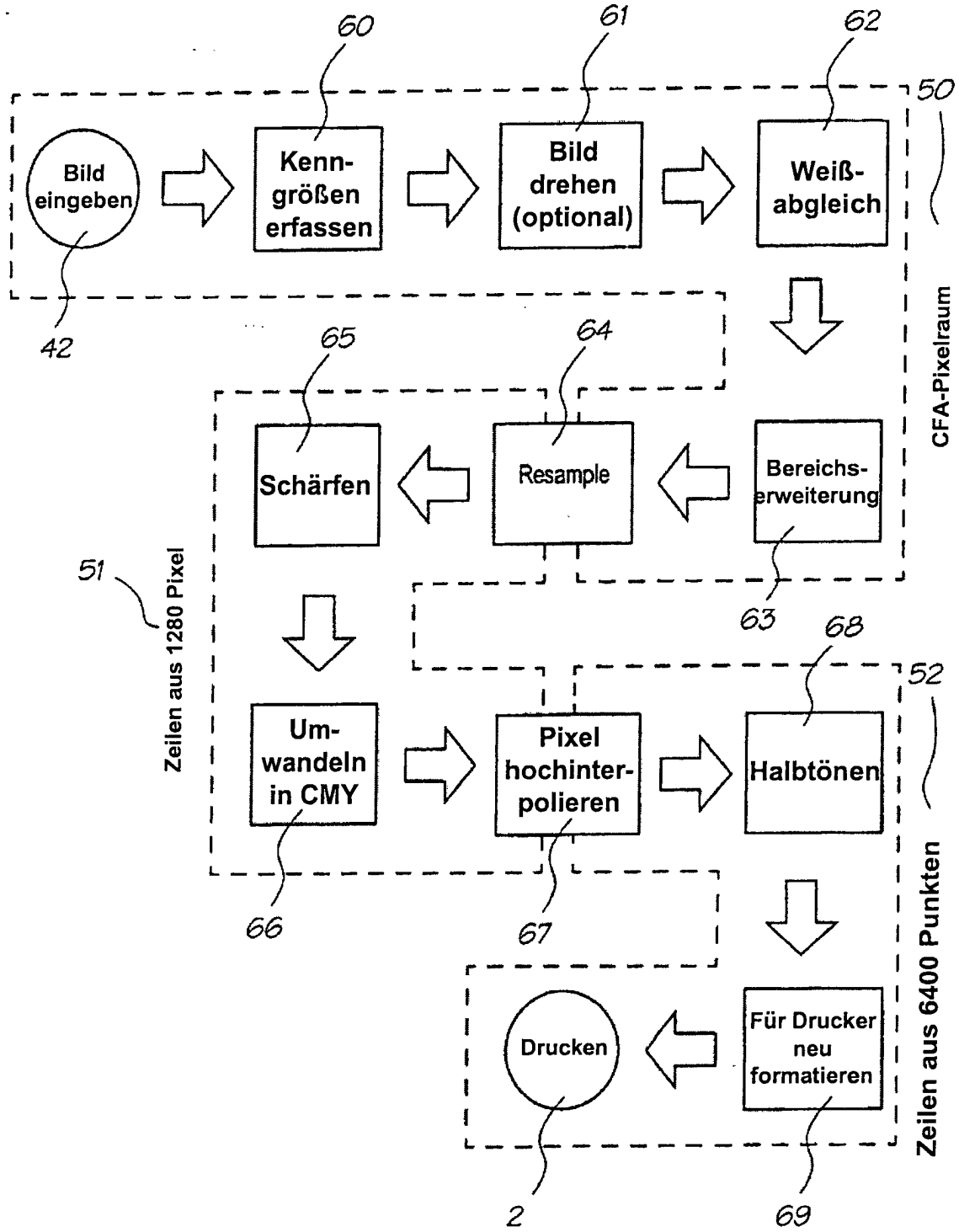

60
61
62
50
Bild eingeben
Kenn-größen erfassen
Bild drehen (optional)
Weiß-abgleich
42
CFA-Pixelraum
65
64
Schärfen
Resample
Bereichs-erweiterung
51
Zeilen aus 1280 Pixel
63
68
52
Um-wandeln in CMY
Pixel hochinter-polieren
Halbtönen
66
67
Zeilen aus 6400 Punkten
Drucken
Für Drucker neu formatieren
2
69

FIG. 18

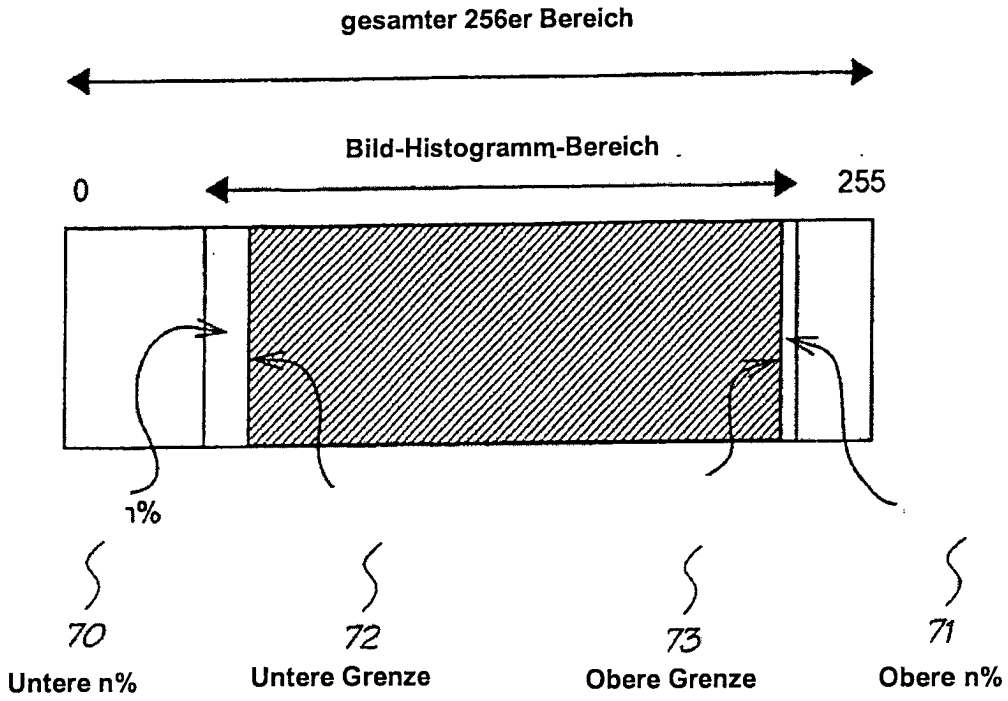

gesamter 256er Bereich
Bild-Histogramm-Bereich
0
255
n%
70
72
73
71
Untere n%
Untere Grenze
Obere Grenze
Obere n%

FIG. 19

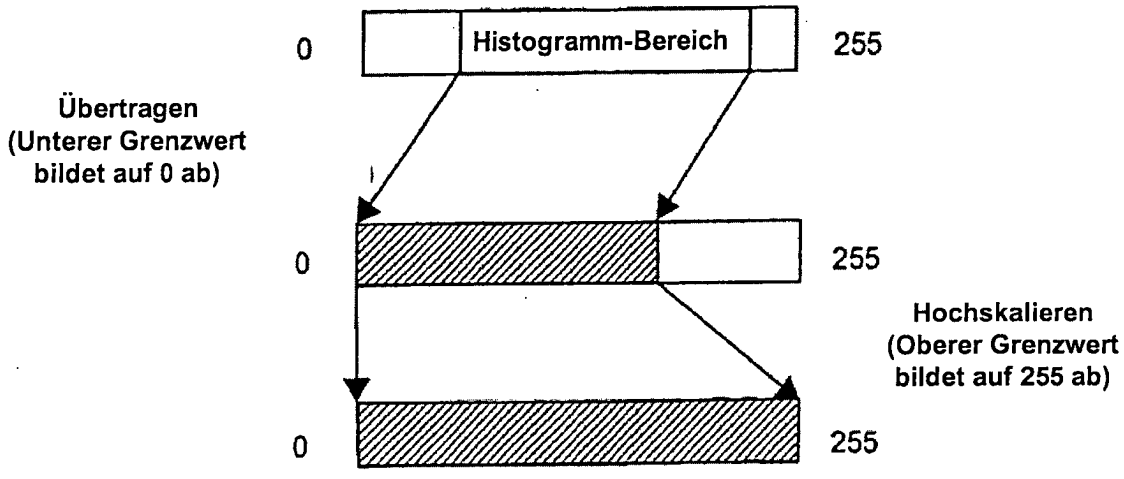

Übertragen
(Unterer Grenzwert
bildet auf 0 ab)
0
Histogramm-Bereich
255
0
255
Hochskalieren
(Oberer Grenzwert
bildet auf 255 ab)
0
255

FIG. 20

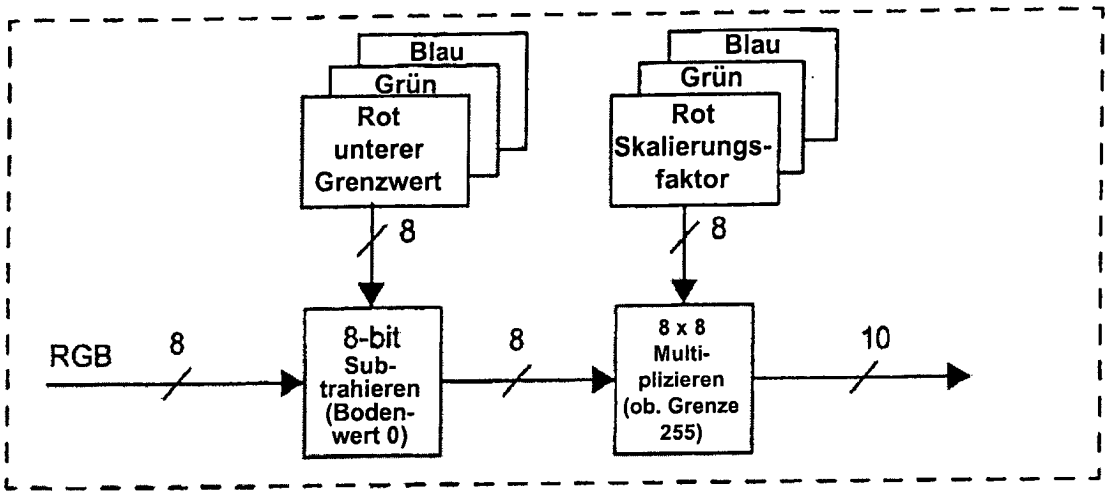
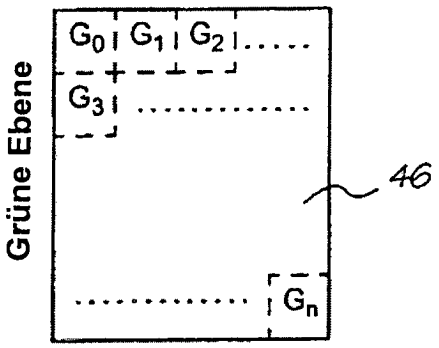
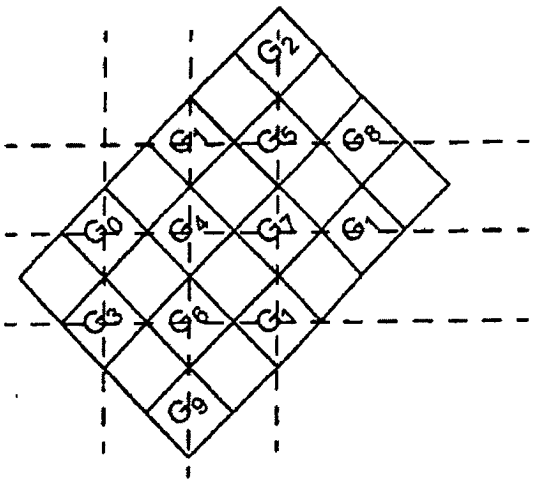


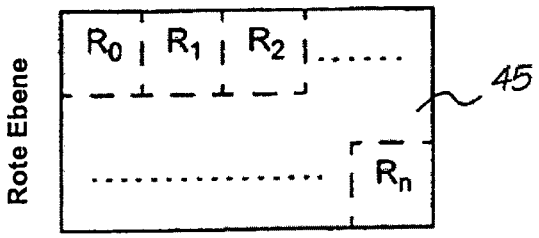
FIG. 21





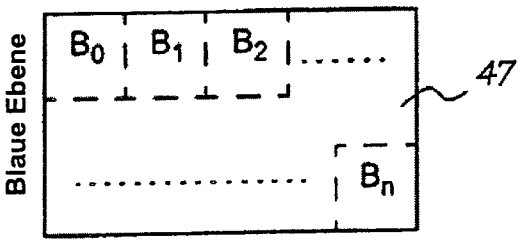
Grüne Ebene in voller CFA-Auflösung (erfasste Pixel sind markiert)

FIG. 23



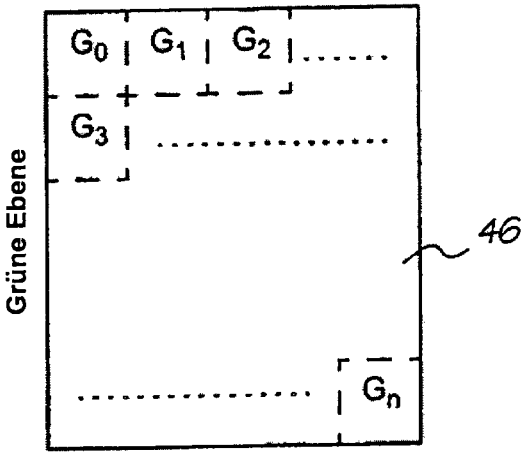
R_0		R_1		R_2	
R_3		R_4		R_5	

Rote Ebene in voller CFA-Auflösung
(= erfasste Pixel sind markiert)



	B_0		B_1		B_2
	B_3		B_4		B_5

Blaue Ebene in voller CFA-Auflösung
(= erfasste Pixel sind markiert)



	G_0		G_1		G_2
G_3		G_4		G_5	
	G_6		G_7		G_8
G_9		G_{10}		G_{11}	

Grüne Ebene in voller CFA-Auflösung
(= erfasste Pixel sind markiert)

FIG. 22

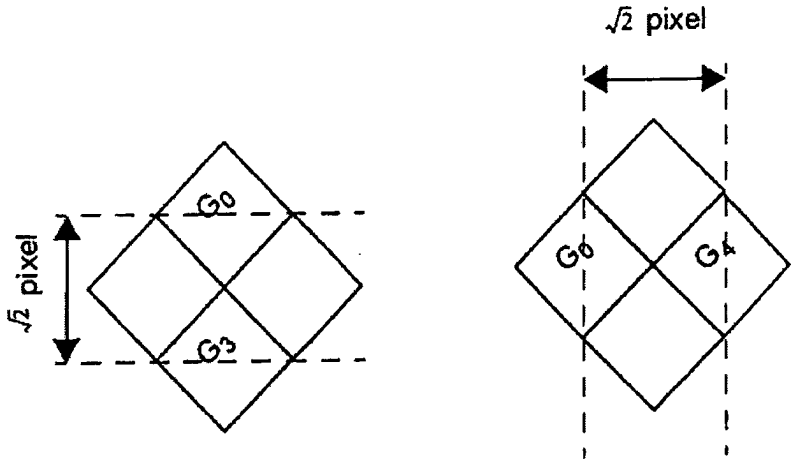

√2 pixel
√2 pixel
G0
G3
G0
G4

FIG. 24

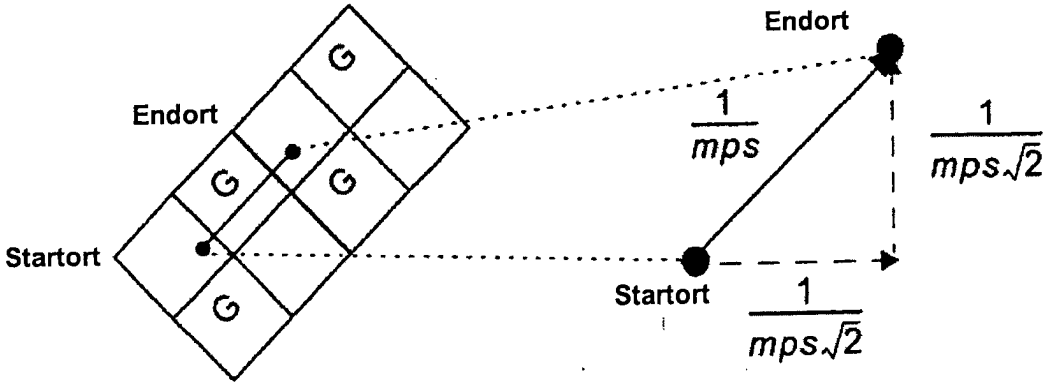

Endort
G
G
G
G
Startort
Endort
1/mps
1/mps√2
Startort
1/mps√2

FIG. 25

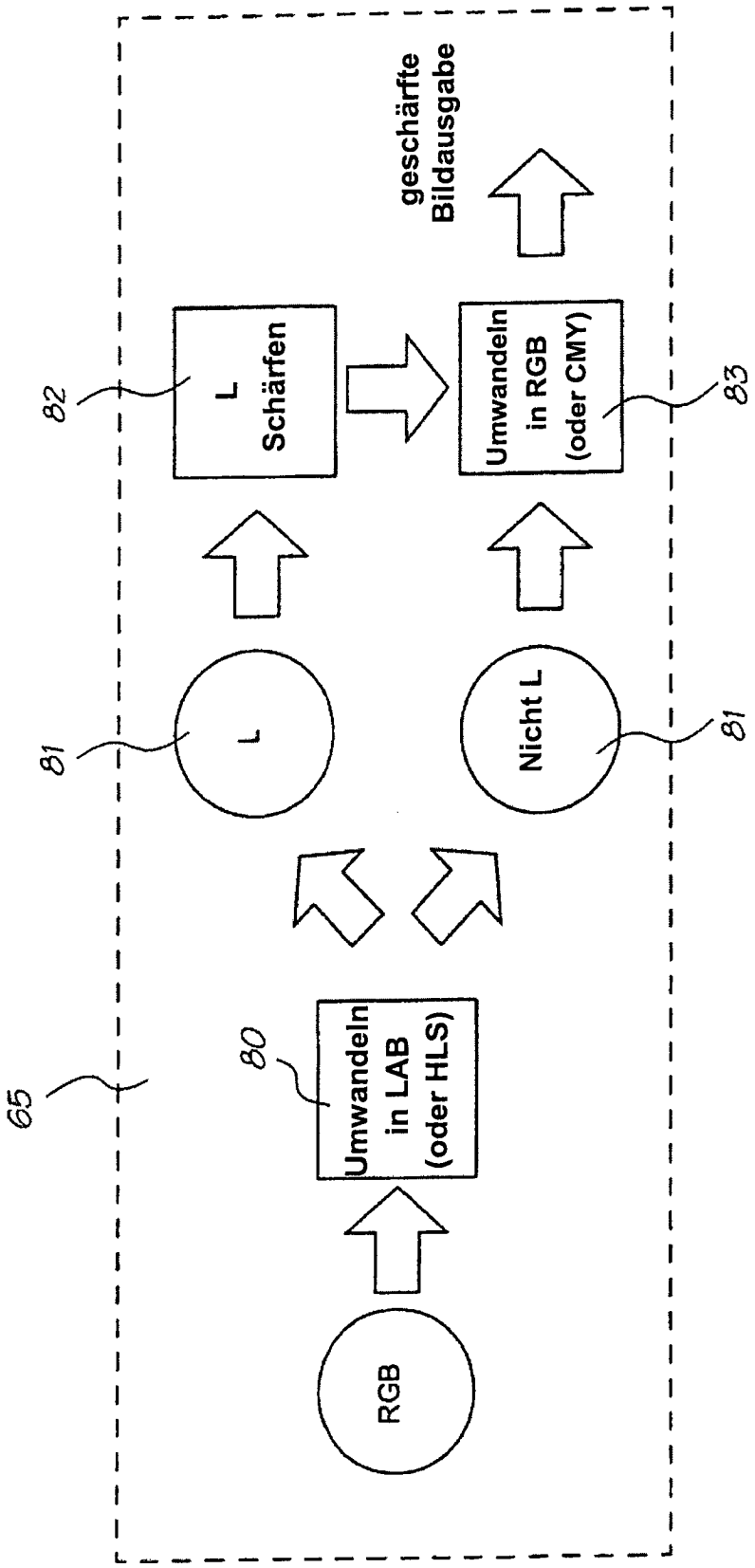

65
80
Umwandeln in LAB (oder HLS)
RGB
81
L
Nicht L
81
82
L Schärfen
Umwandeln in RGB (oder CMY)
83
geschärfte Bildausgabe

FIG. 26

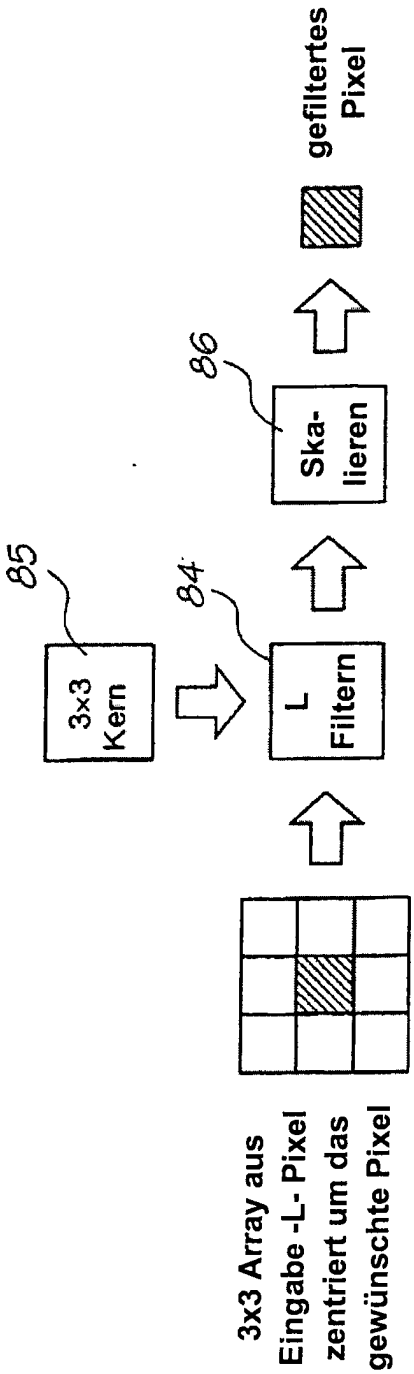

3x3 Array aus Eingabe -L- Pixel zentriert um das gewünschte Pixel
3x3 Kern
L Filtern
Ska-lieren
gefiltertes Pixel
85
84
86

FIG. 27

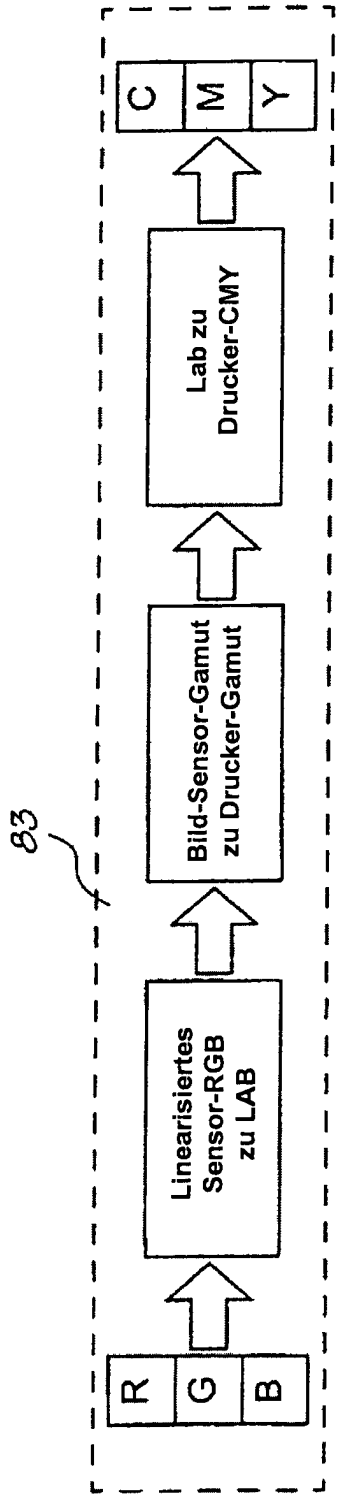

R
G
B
Linearisiertes Sensor-RGB zu LAB
Bild-Sensor-Gamut zu Drucker-Gamut
Lab zu Drucker-CMY
C
M
Y
83

FIG. 28

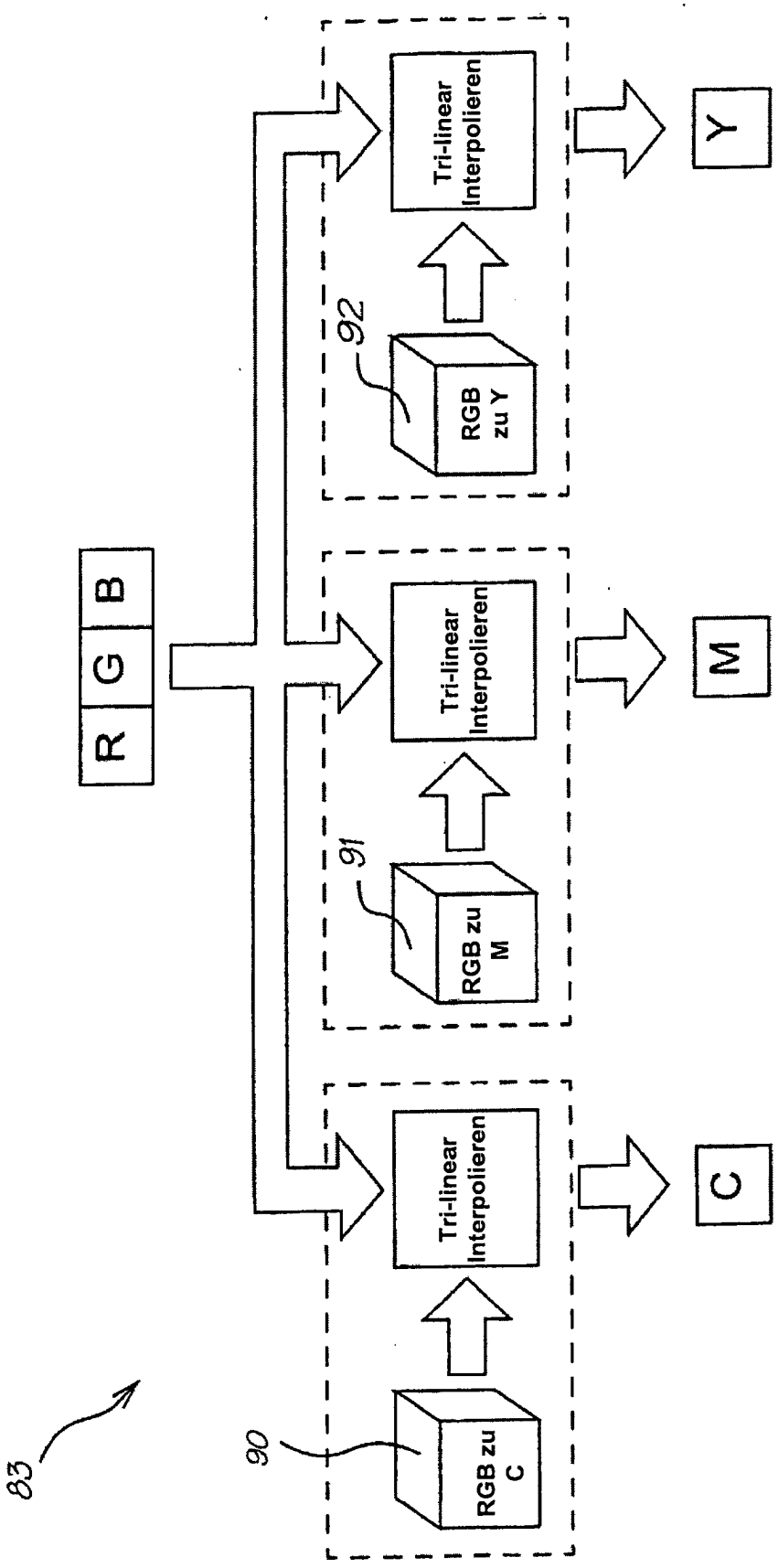

R
G
B
Tri-linear Interpolieren
RGB zu C
RGB zu M
RGB zu Y
C
M
Y
83
90
91
92

FIG. 29

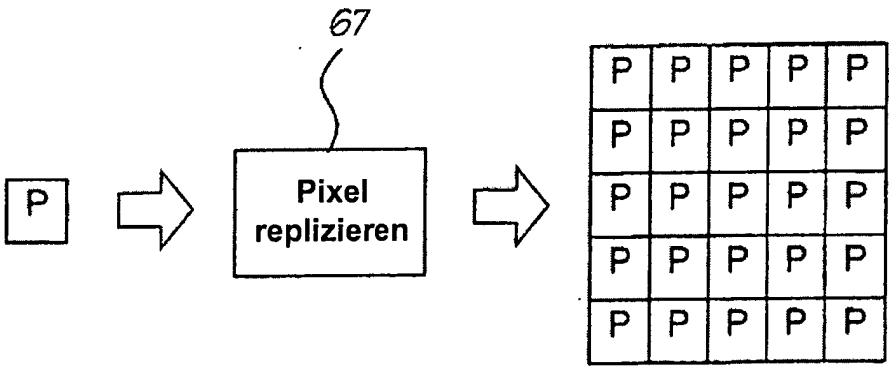

67
P
Pixel replizieren
P P P P P
P P P P P
P P P P P
P P P P P
P P P P P

FIG. 30

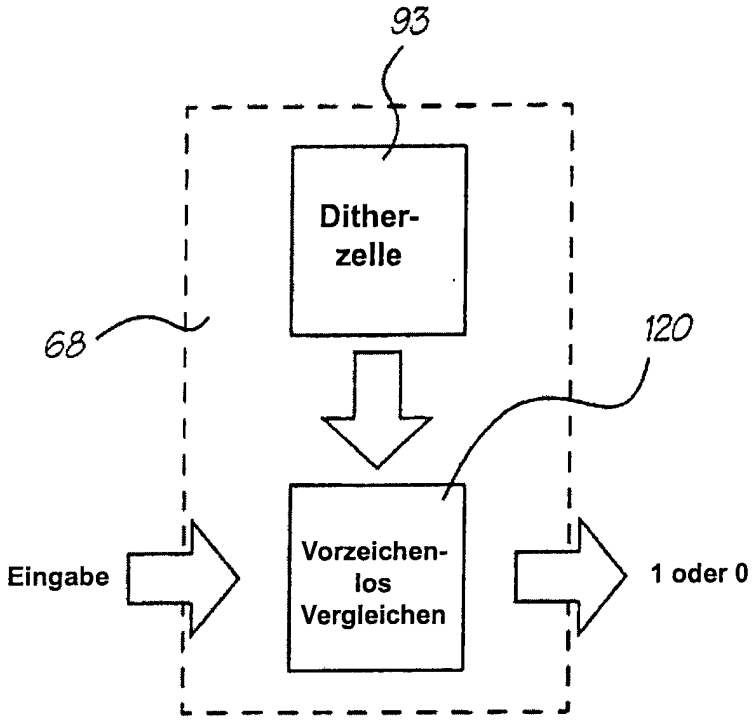

93
Dither-zelle
68
120
Eingabe
Vorzeichen-los Vergleichen
1 oder 0

FIG. 31

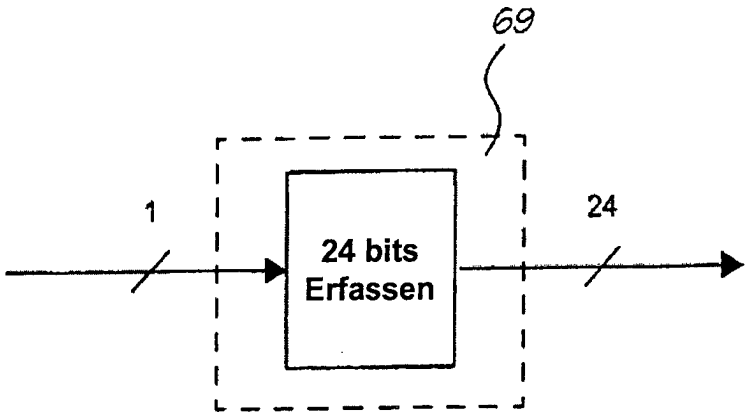

69
1
24 bits
Erfassen
24

FIG. 32

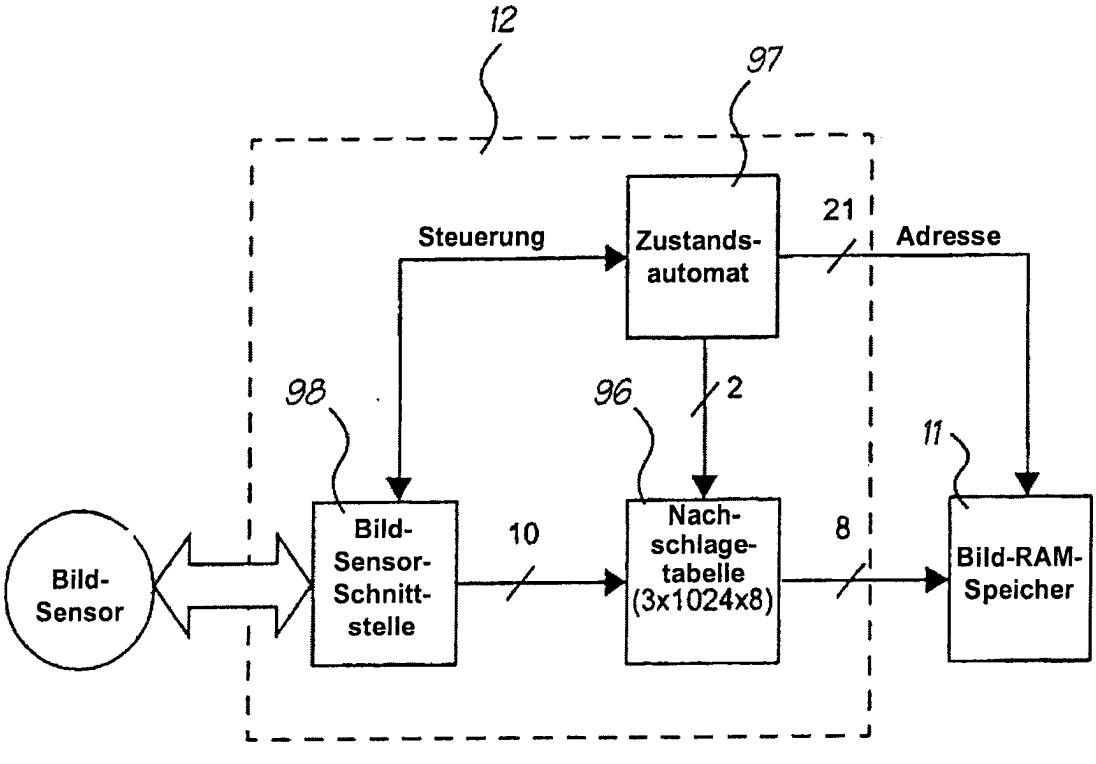

12
97
Steuerung
Zustands-
automat
21
Adresse
98
96
2
11
Bild-
Sensor
Bild-
Sensor-
Schnitt-
stelle
10
Nach-
schlage-
tabelle
(3x1024x8)
8
Bild-RAM-
Speicher

FIG. 33

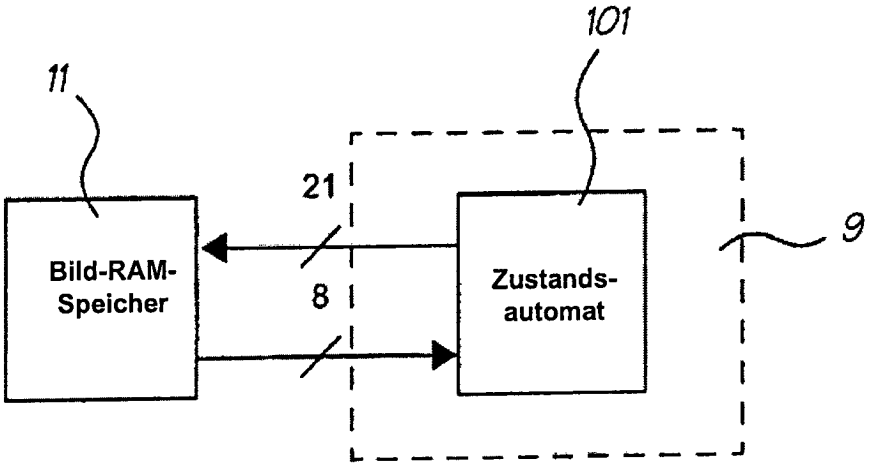

11
101
21
9
Bild-RAM-
Speicher
8
Zustands-
automat

FIG. 35

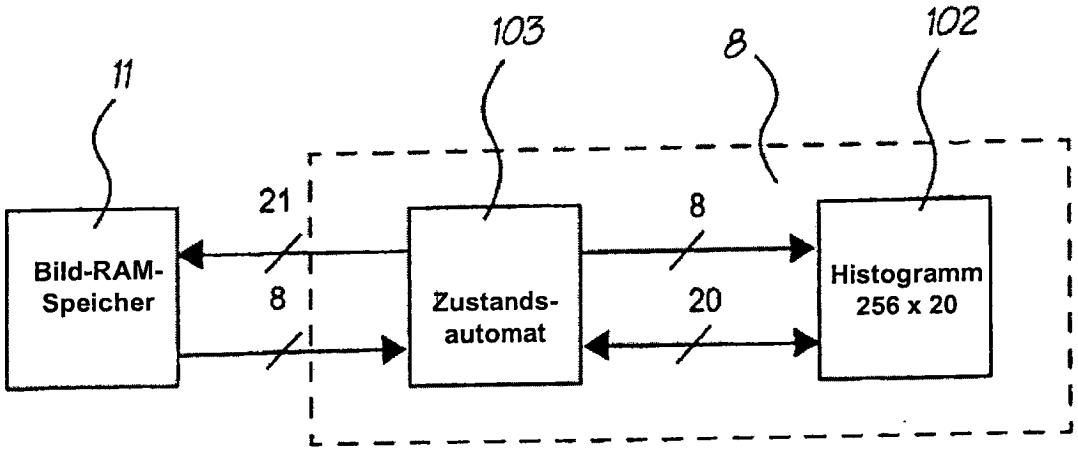

11
103
8
102
21
8
Bild-RAM-
Speicher
8
Zustands-
automat
20
Histogramm
256 x 20

FIG. 36

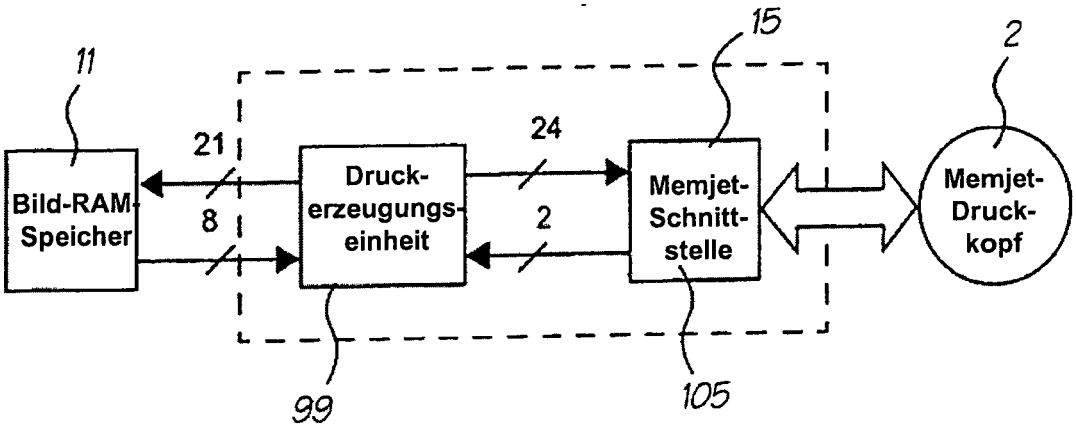


FIG. 37

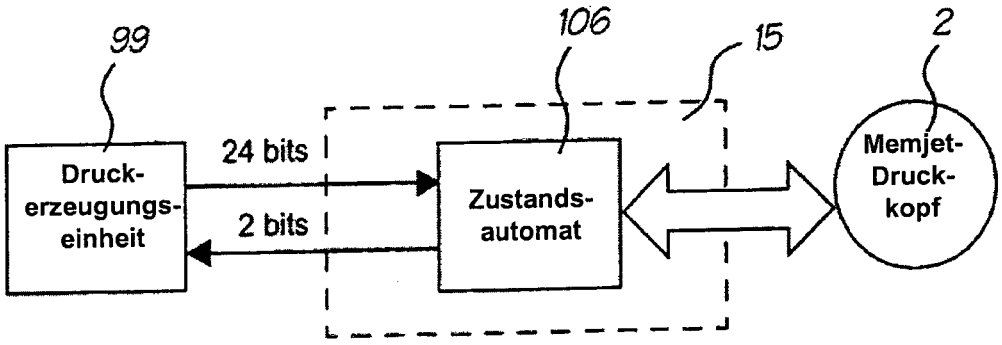


FIG. 38

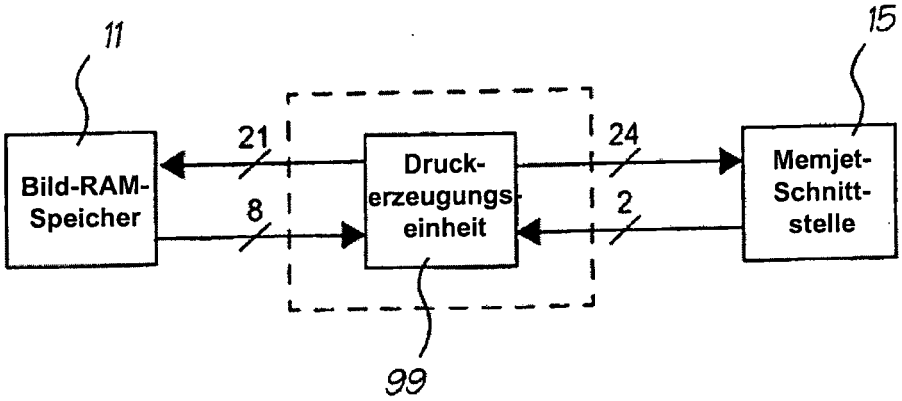


FIG. 41

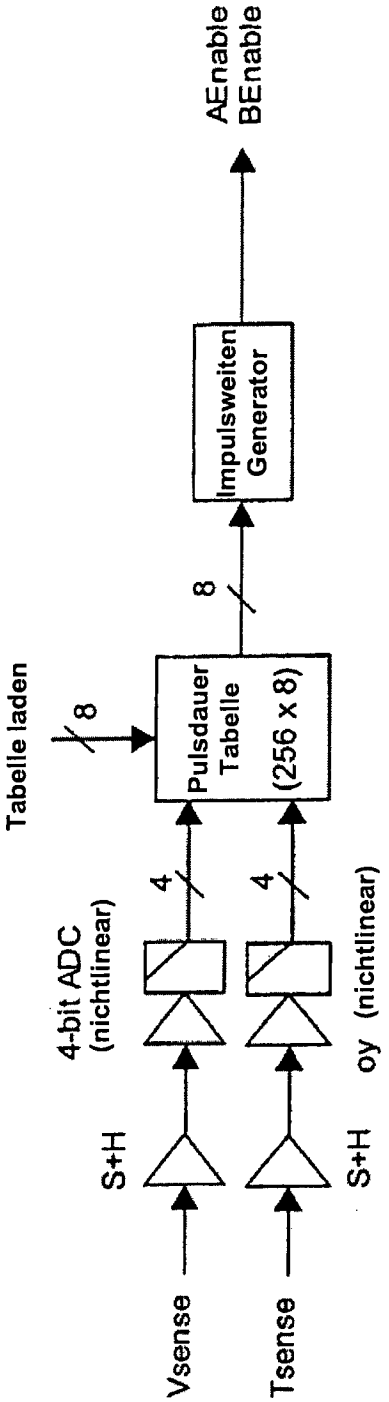

Vsense
Tsense
S+H
S+H
4-bit ADC
(nichtlinear)
oy (nichtlinear)
4
4
Tabelle laden
8
Pulsdauer
Tabelle
(256 x 8)
8
Impulsweiten
Generator
AEnable
BEnable

FIG. 39

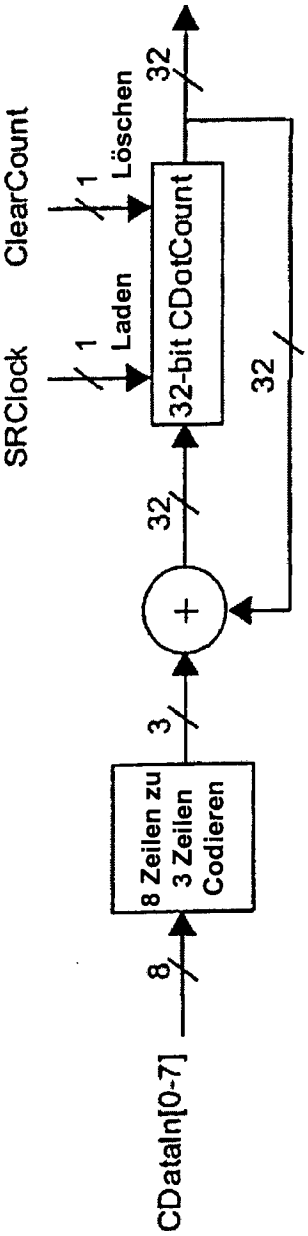

CDataIn[0-7]
8
8 Zeilen zu
3 Zeilen
Codieren
3
+
32
32-bit CDotCount
SRClock
1
Laden
ClearCount
1
Löschen
32
32

FIG. 40

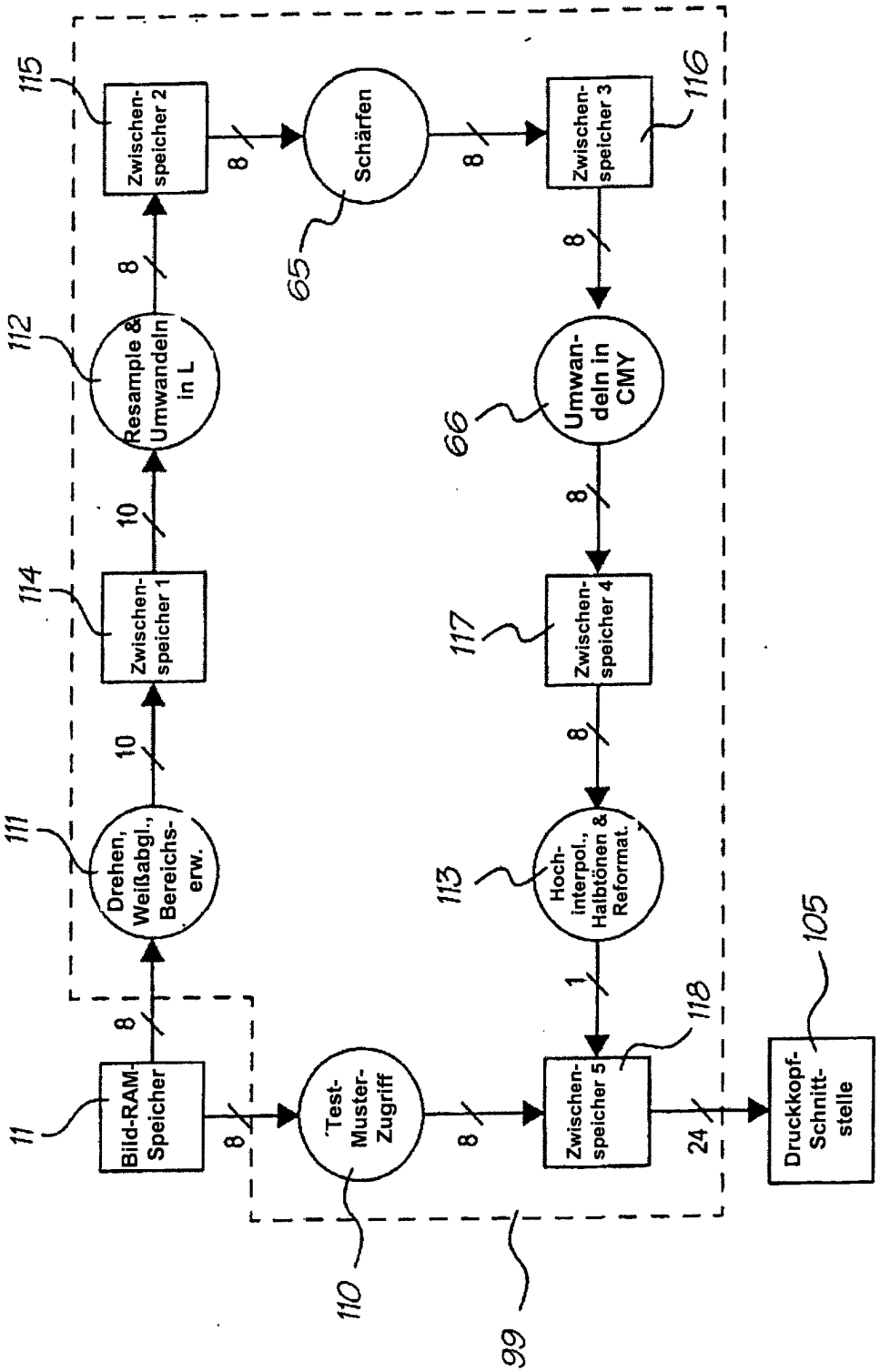


Bild-RAM-Speicher
Drehen, Weißabgl., Bereichserw.
Zwischenspeicher 1
Resample & Umwandeln in L
Zwischenspeicher 2
Test-Muster-Zugriff
Schärfen
Umwandeln in CMY
Zwischenspeicher 3
Zwischenspeicher 4
Hochinterpol., Halbtönen & Reformat.
Zwischenspeicher 5
Druckkopf-Schnittstelle
11
111
114
112
115
110
65
66
116
117
113
118
105
99
8
10
1
24

FIG. 42

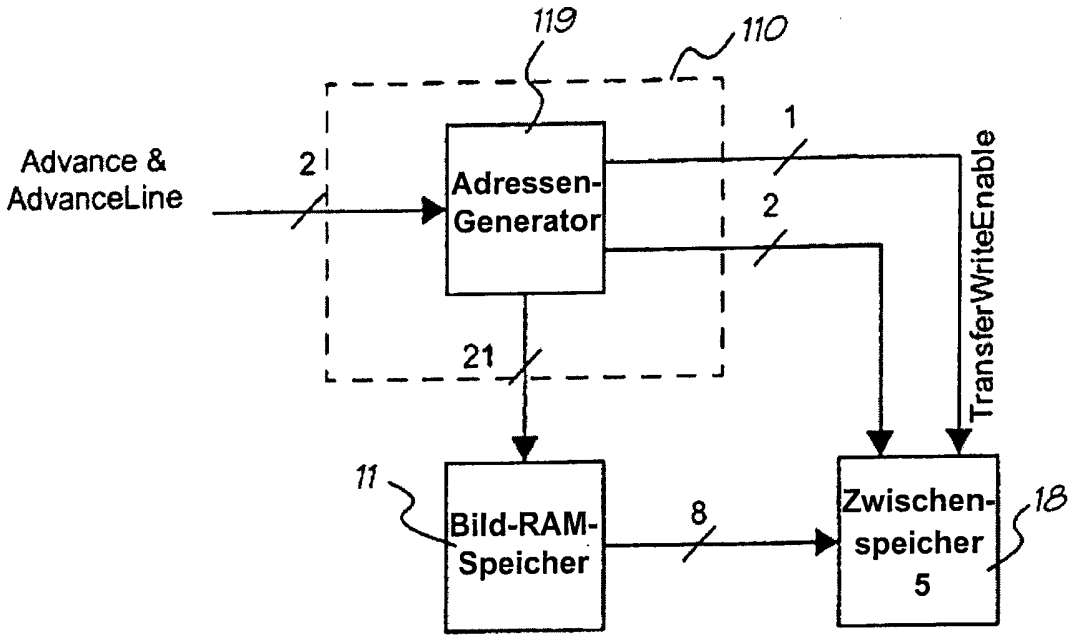

119
110
Advance &
AdvanceLine
2
Adressen-
Generator
1
2
TransferWriteEnable
21
11
Bild-RAM-
Speicher
8
Zwischen-
speicher
5
18

FIG. 43

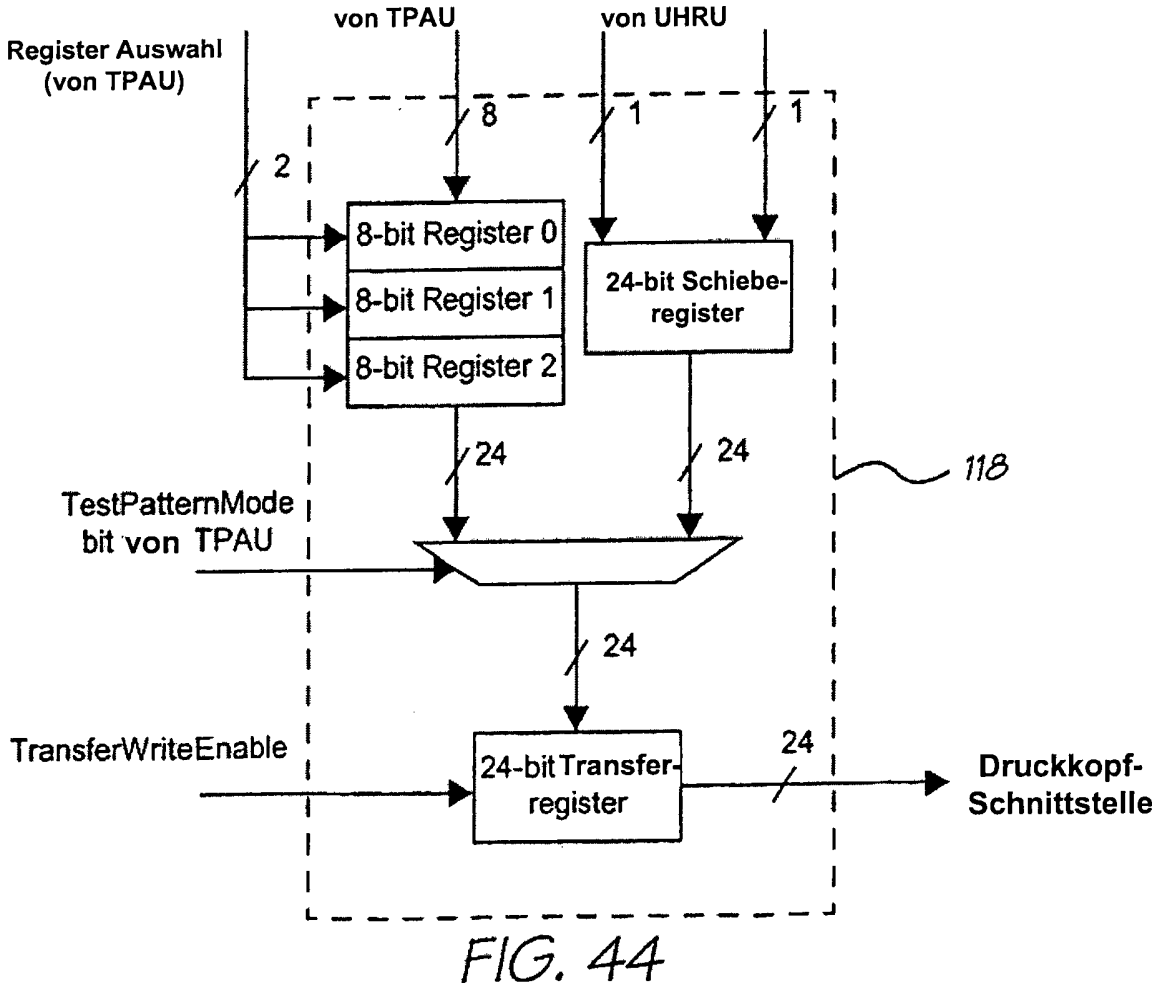

Register Auswahl
(von TPAU)
von TPAU
von UHRU
8
1
1
2
8-bit Register 0
8-bit Register 1
8-bit Register 2
24-bit Schiebe-
register
24
24
118
TestPatternMode
bit von TPAU
24
TransferWriteEnable
24-bit Transfer-
register
24
Druckkopf-
Schnittstelle

FIG. 44

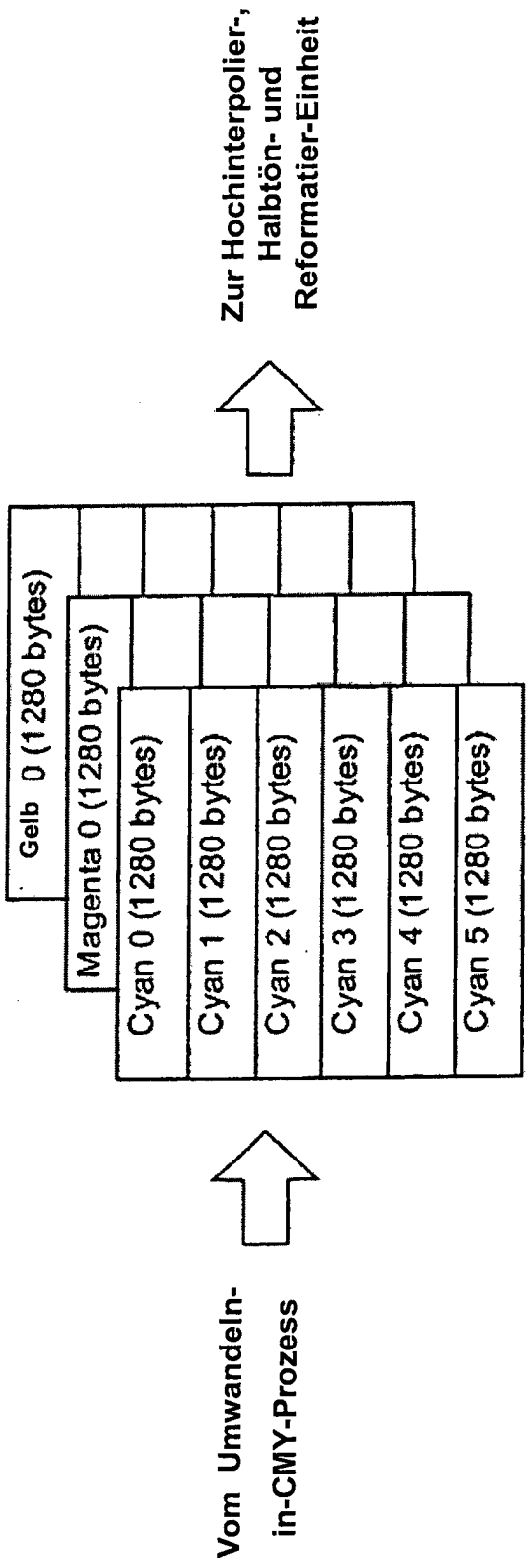

Vom Umwandeln-in-CMY-Prozess
Gelb 0 (1280 bytes)
Magenta 0 (1280 bytes)
Cyan 0 (1280 bytes)
Cyan 1 (1280 bytes)
Cyan 2 (1280 bytes)
Cyan 3 (1280 bytes)
Cyan 4 (1280 bytes)
Cyan 5 (1280 bytes)
Zur Hochinterpolier-, Halbtön- und Reformatier-Einheit

FIG. 45

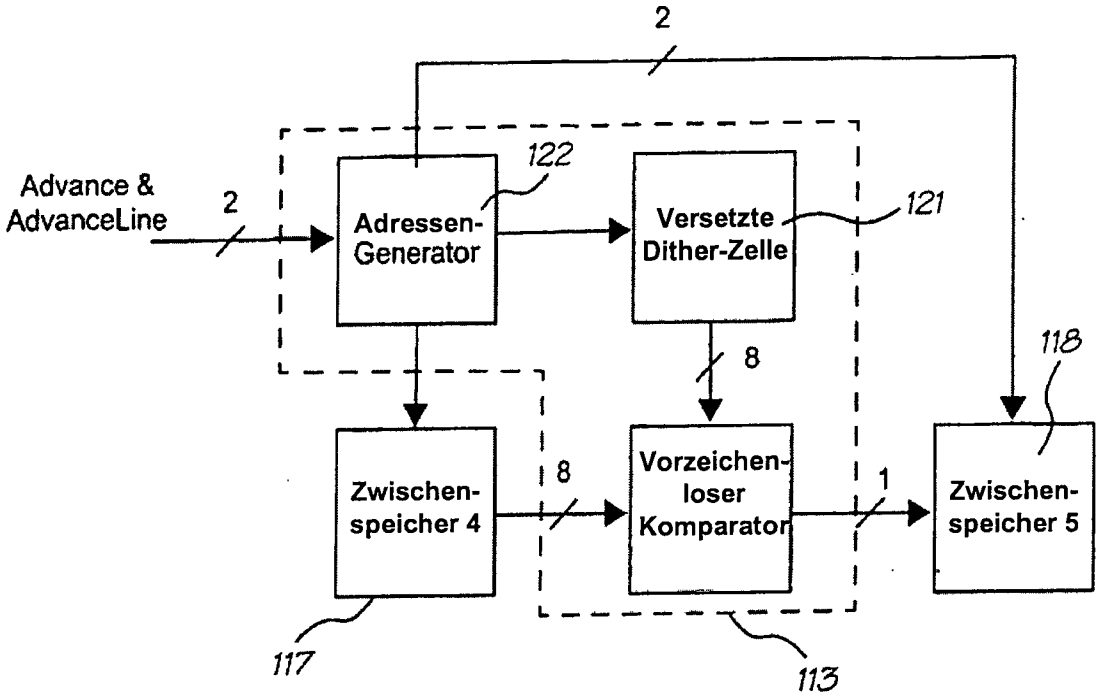

2
Advance &
AdvanceLine
2
Adressen-
Generator
122
Versetzte
Dither-Zelle
121
8
118
Zwischen-
speicher 4
8
Vorzeichen-
loser
Komparator
1
Zwischen-
speicher 5
117
113

FIG. 46

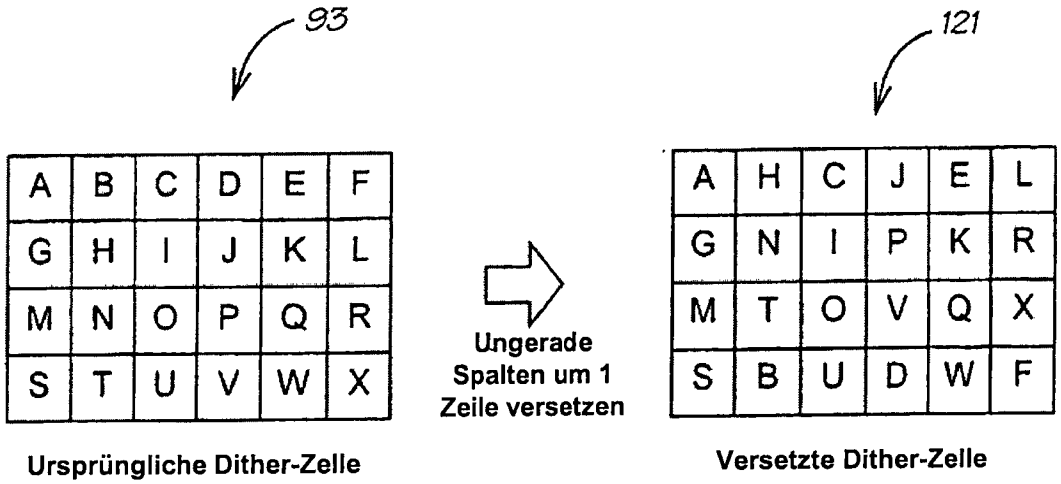

93
121
A B C D E F
G H I J K L
M N O P Q R
S T U V W X
Ursprüngliche Dither-Zelle
Ungerade
Spalten um 1
Zeile versetzen
A H C J E L
G N I P K R
M T O V Q X
S B U D W F
Versetzte Dither-Zelle

FIG. 47

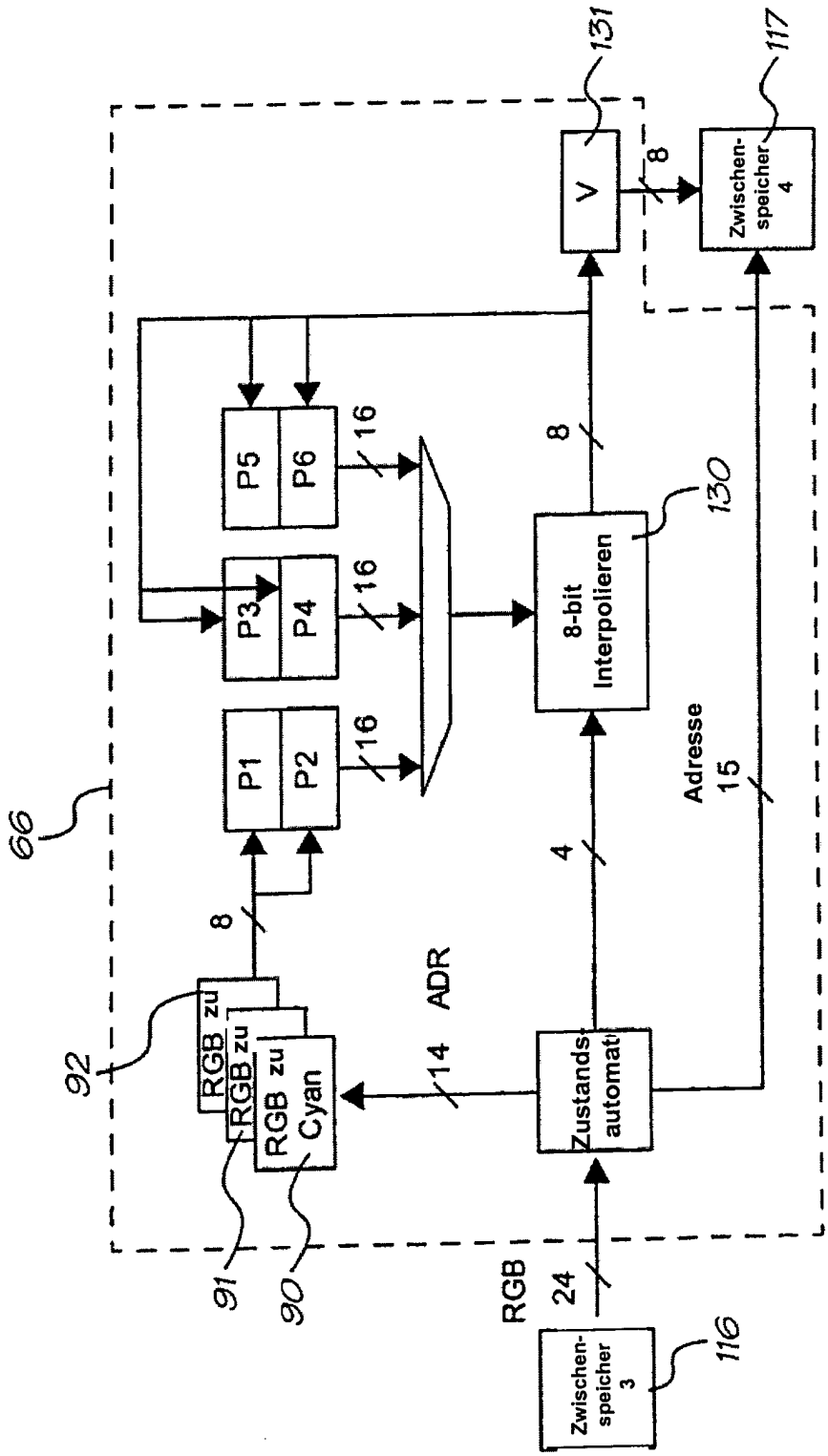

66
92
91
90
RGB zu
RGB zu
RGB zu
Cyan
8
P1
P2
P3
P4
P5
P6
16
16
16
ADR
14
4
RGB
24
Zustands-
automat
8-bit
Interpolieren
130
8
V
131
8
Zwischen-
speicher
4
117
Adresse
15
Zwischen-
speicher
3
116

FIG. 48

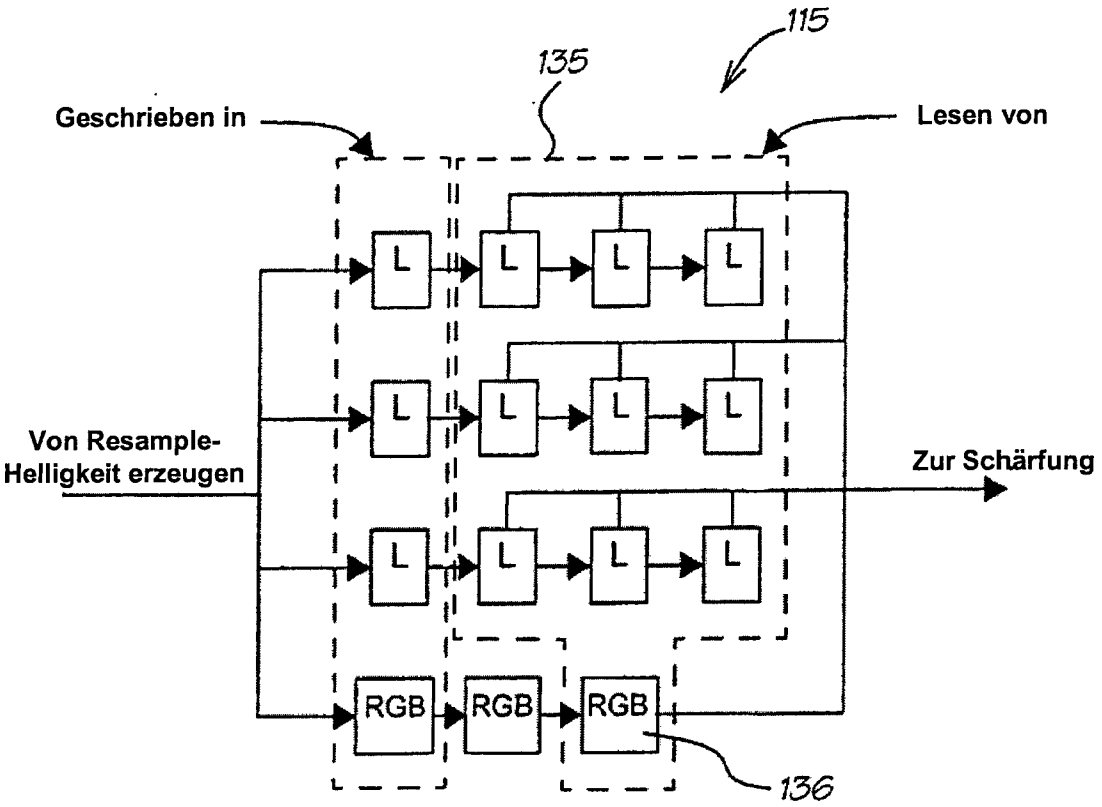


FIG. 49

$\frac{1}{9} \times$

-1	-1	-1
-1	8	-1
-1	-1	-1

FIG. 50

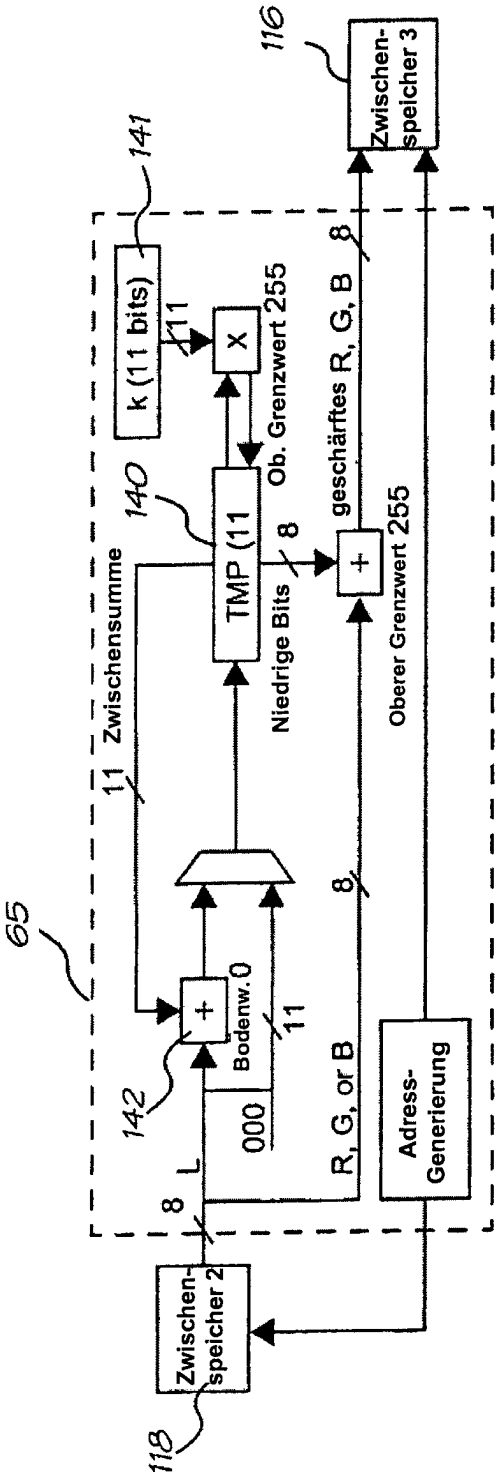

65
11 Zwischensumme
141
k (11 bits)
11
X
Ob. Grenzwert 255
140
TMP (11
8
Niedrige Bits
geschärftes R, G, B
8
116
Zwischen-speicher 3
Oberer Grenzwert 255
8
R, G, or B
Adress-Generierung
11
Bodenw.0
000
142
L
8
Zwischen-speicher 2
118

FIG. 51

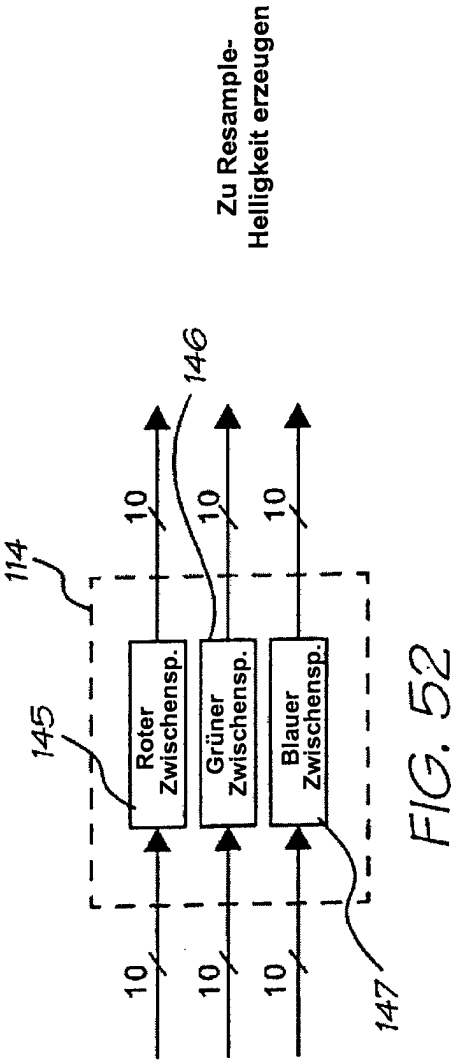

Zu Resample-Helligkeit erzeugen
114
146
10
10
10
145
Roter Zwischensp.
Grüner Zwischensp.
Blauer Zwischensp.
10
10
10
147
Von Drehen-Weißabgleich-Bereichserweiterung

FIG. 52

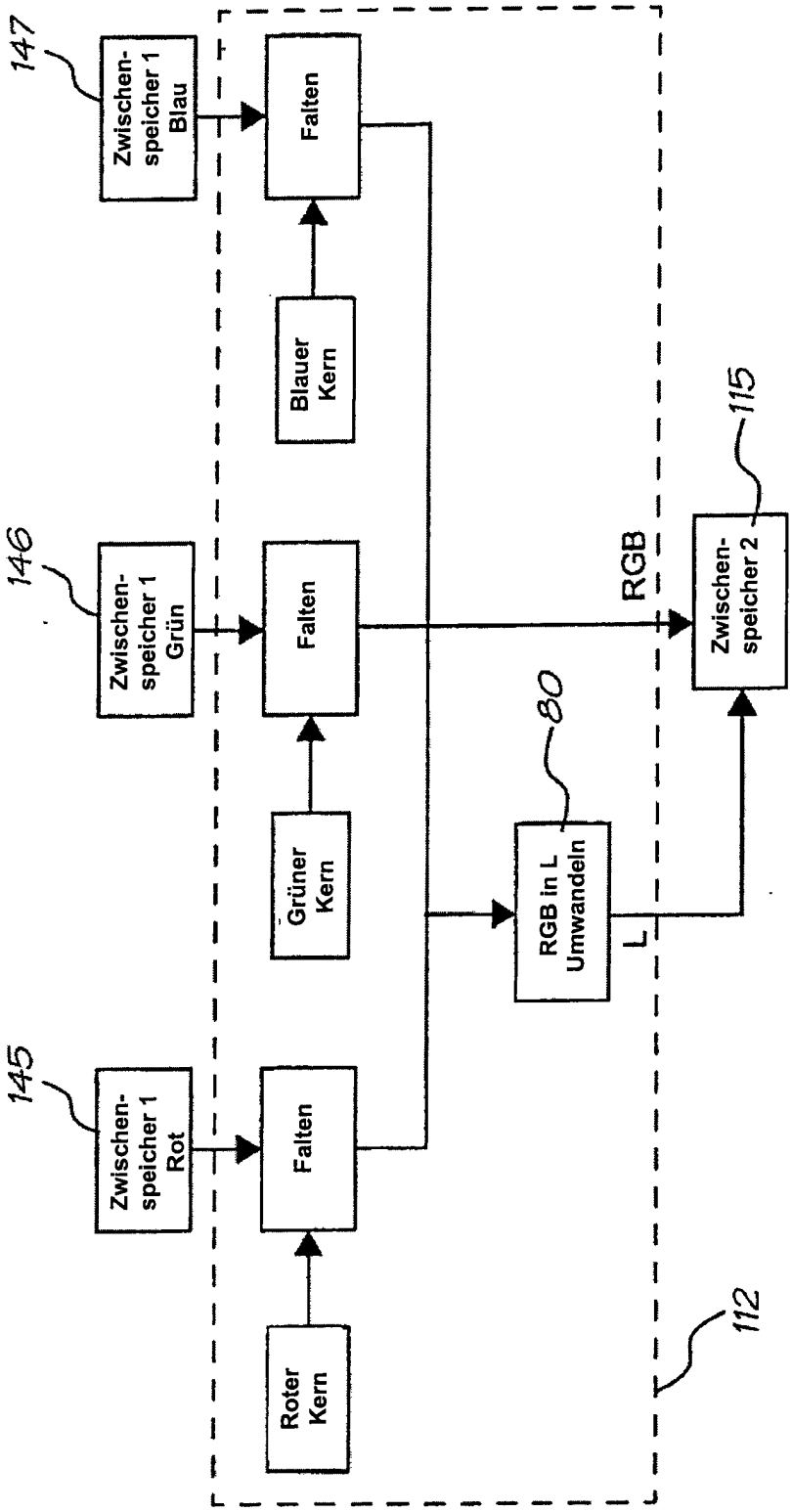

147
Zwischen-speicher 1 Blau
Falten
Blauer Kern
146
Zwischen-speicher 1 Grün
Falten
Grüner Kern
145
Zwischen-speicher 1 Rot
Falten
Roter Kern
80
RGB in L Umwandeln
L
RGB
115
Zwischen-speicher 2
112

FIG. 53

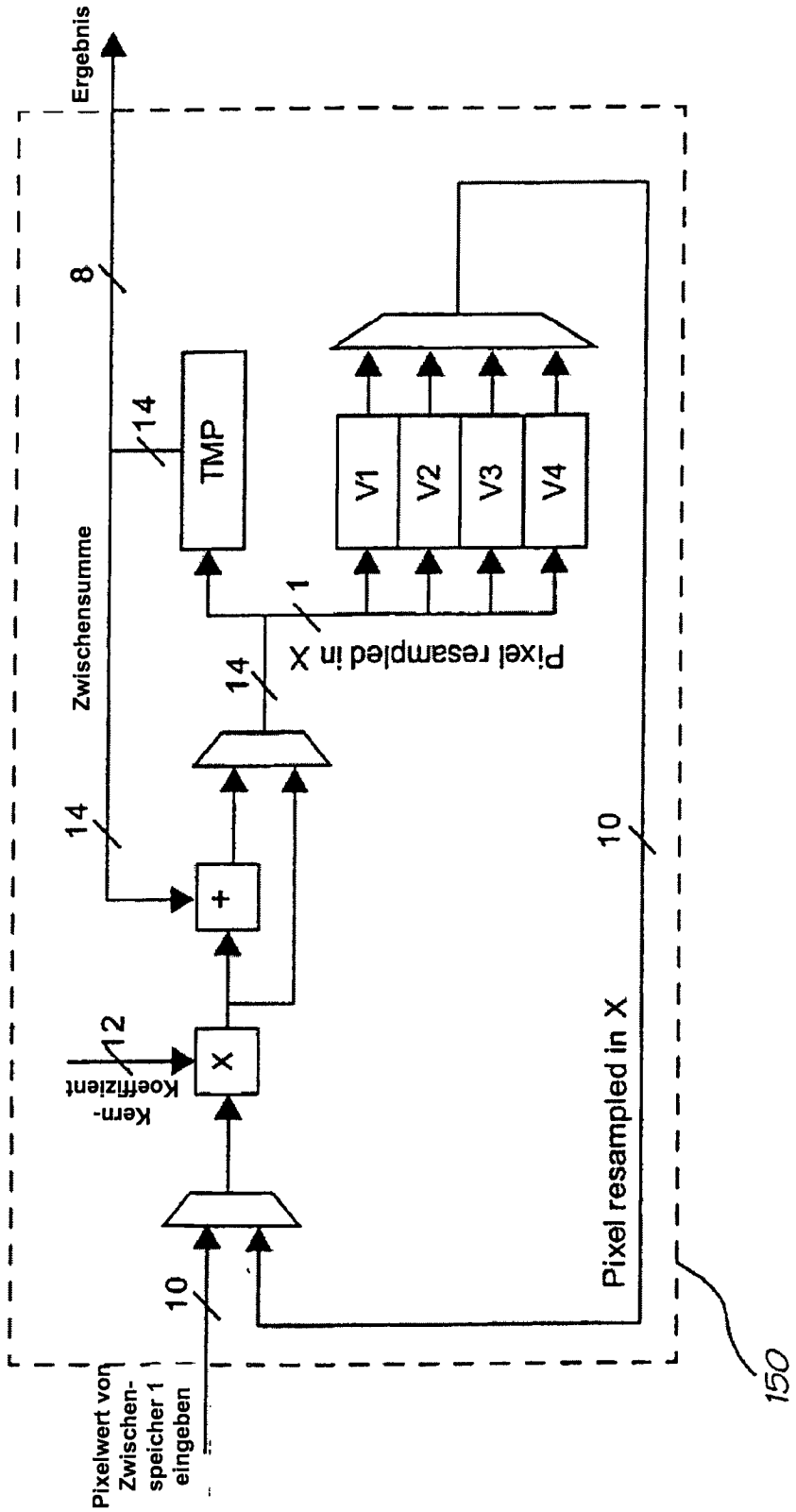

Ergebnis
8
14
TMP
Zwischensumme
V1
V2
V3
V4
1
Pixel resampled in X
14
14
+
12
Kern-Koeffizient
X
10
Pixel resampled in X
10
Pixelwert von Zwischen-speicher 1 eingeben
150

FIG. 54

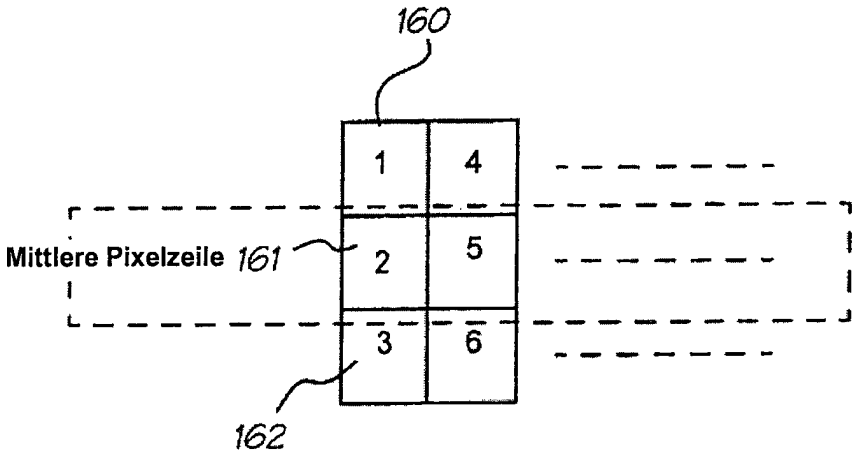


FIG. 55

0	13	26	39	52	65
1	14	27	40	53	66
10	23	36	49	62	75
11	24	37	50	63	76
12	25	38	51	64	77

FIG. 57

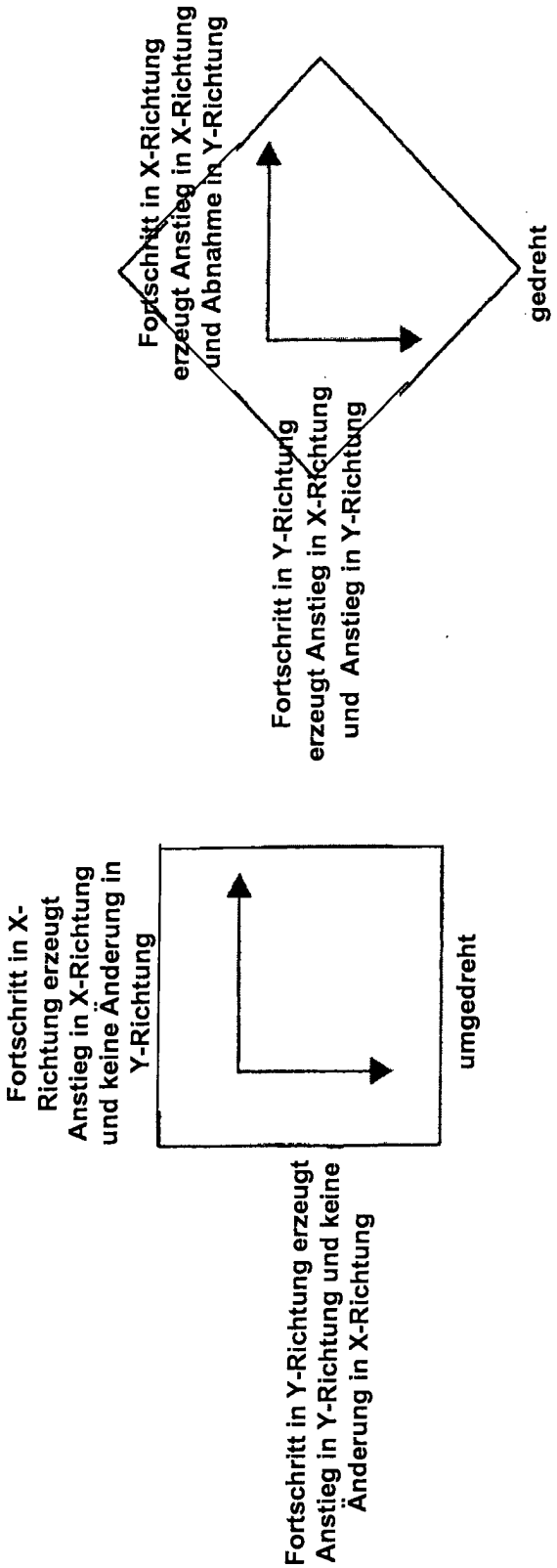
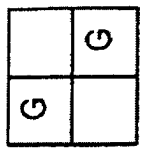


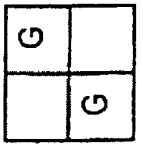
FIG. 56



Gerade Zeilen haben Abtastpunkte bei geraden Pixel

Ungerade Zeilen haben Abtastpunkte bei ungeraden Pixel

Basis 2x2 CFA-Zelle (Drehung = 90)



Gerade Zeilen haben Abtastpunkte bei ungeraden Pixel

Ungerade Zeilen haben Abtastpunkte bei geraden Pixel

Basis 2x2 CFA-Zelle (Drehung = 0)

FIG. 58

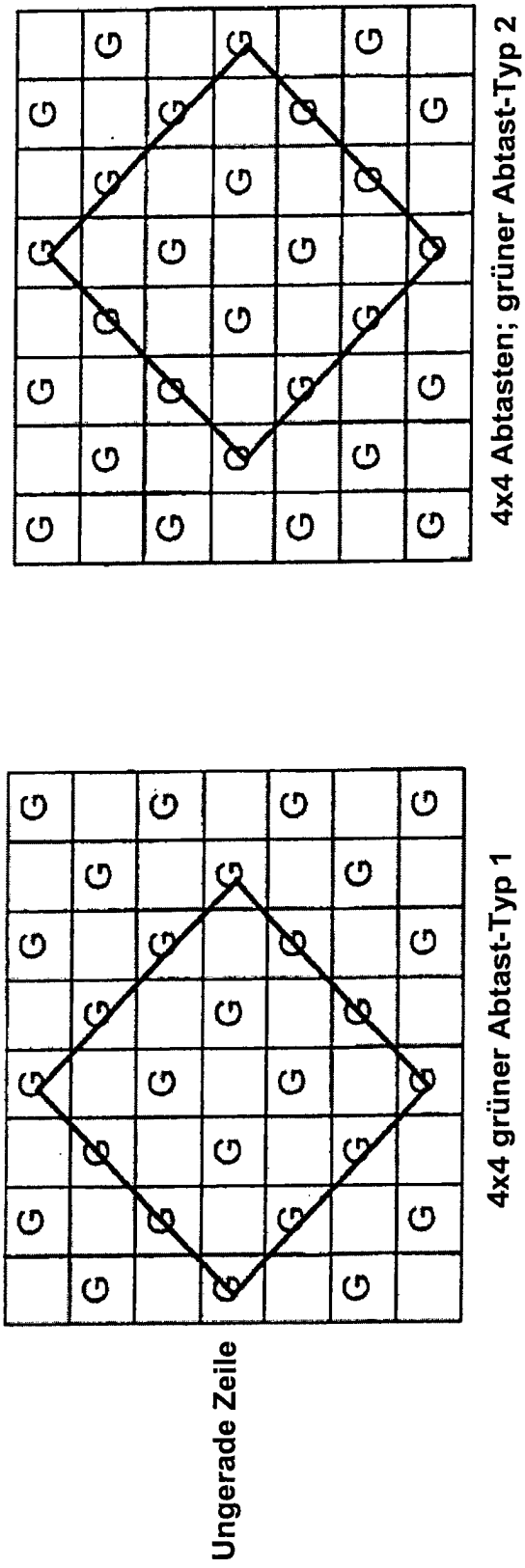
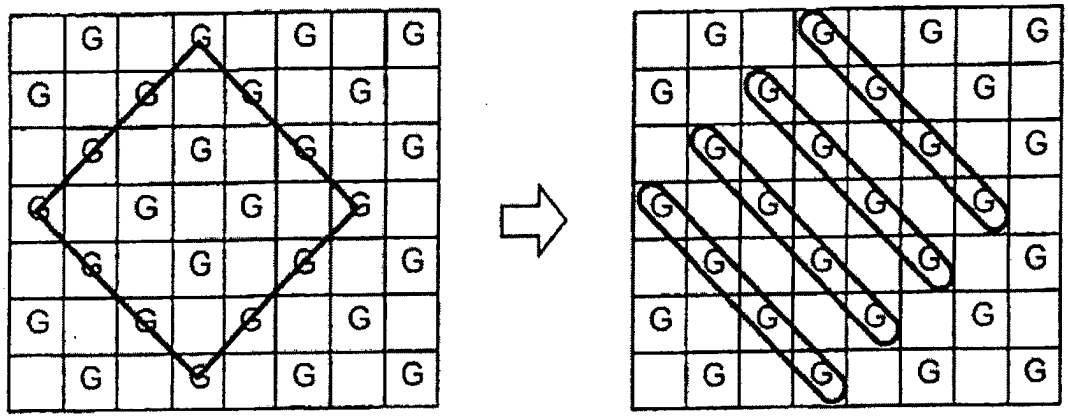

4x4 Abtasten; grüner Abtast-Typ 2
4x4 grüner Abtast-Typ 1
Ungerade Zeile

FIG. 59


4x4 grüner Abtast - Typ 1

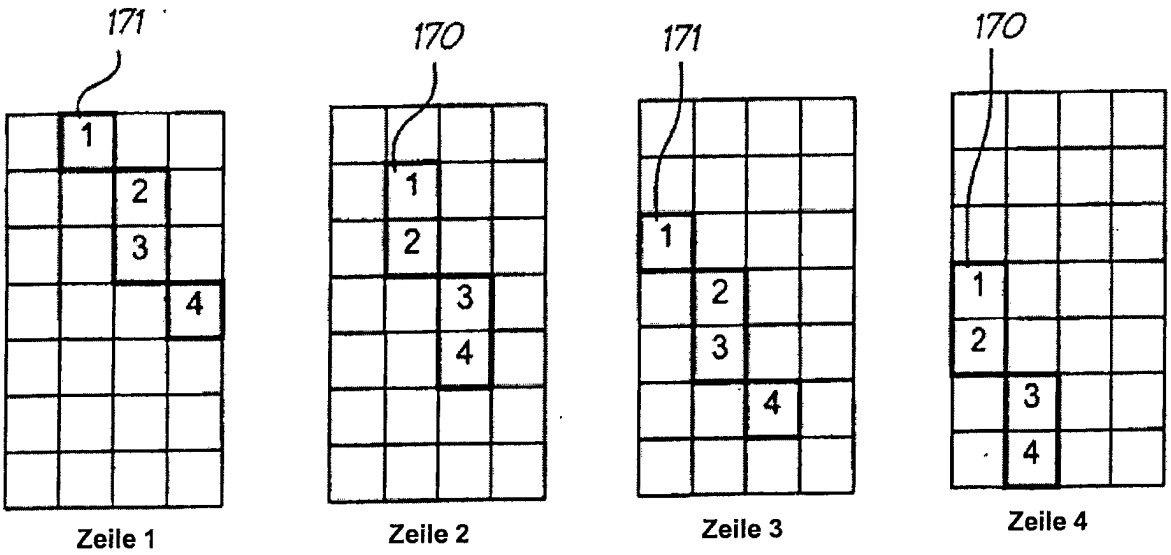
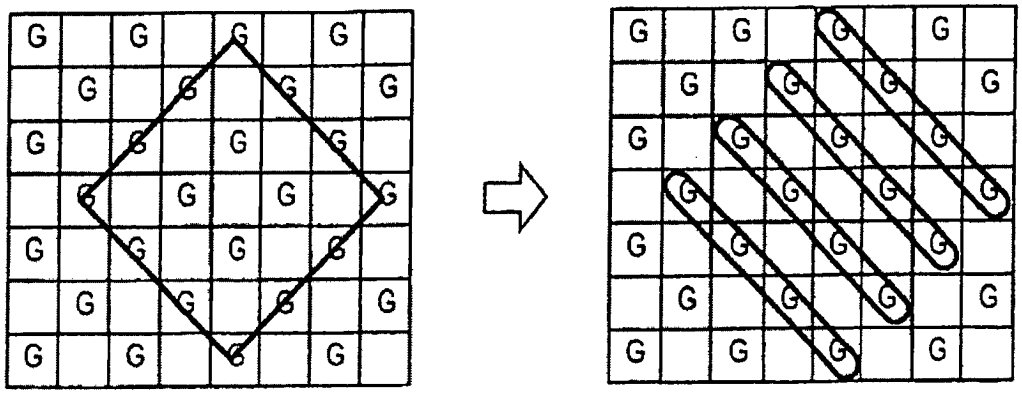

171
170
171
170
Zeile 1
Zeile 2
Zeile 3
Zeile 4

FIG. 60


4x4 grüner Abtast - Typ 2

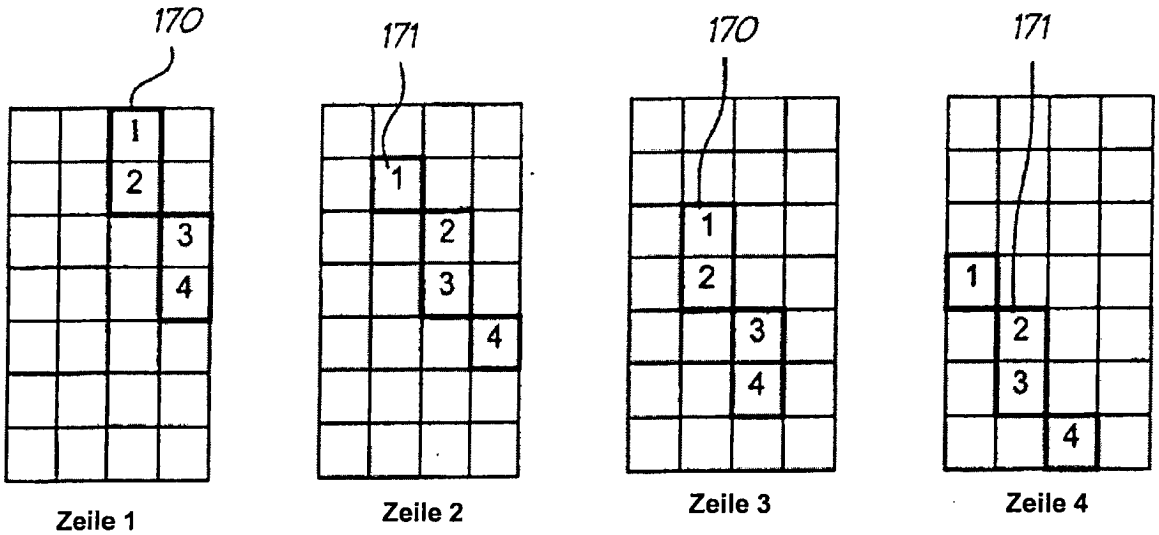

170
171
170
171
Zeile 1
Zeile 2
Zeile 3
Zeile 4

FIG. 61

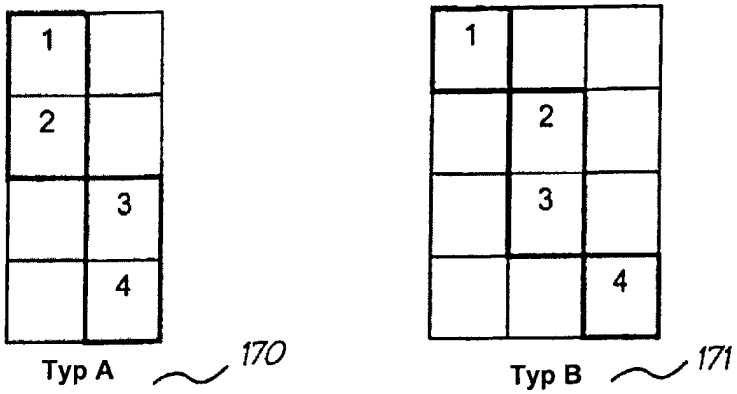


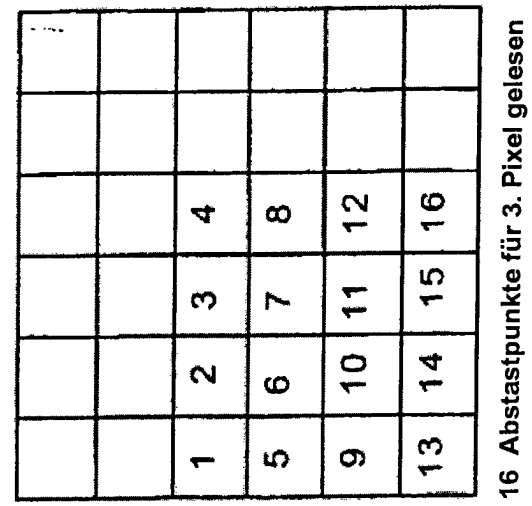
FIG. 62

0	6	12	18	24	30
1	7	13	19	25	31
2	8	14	20	26	32
3	9	15	21	27	33
4	10	16	22	28	34
5	11	17	23	29	35

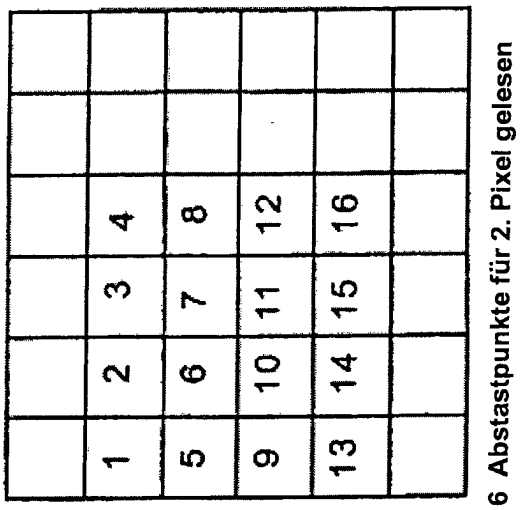
FIG. 63

1	2	3	4		
5	6	7	8		
9	10	11	12		
13	14	15	16		

FIG. 64



16 Abstastpunkte für 1. Pixel gelesen



16 Abstastpunkte für 2. Pixel gelesen

16 Abstastpunkte für 3. Pixel gelesen

FIG. 65

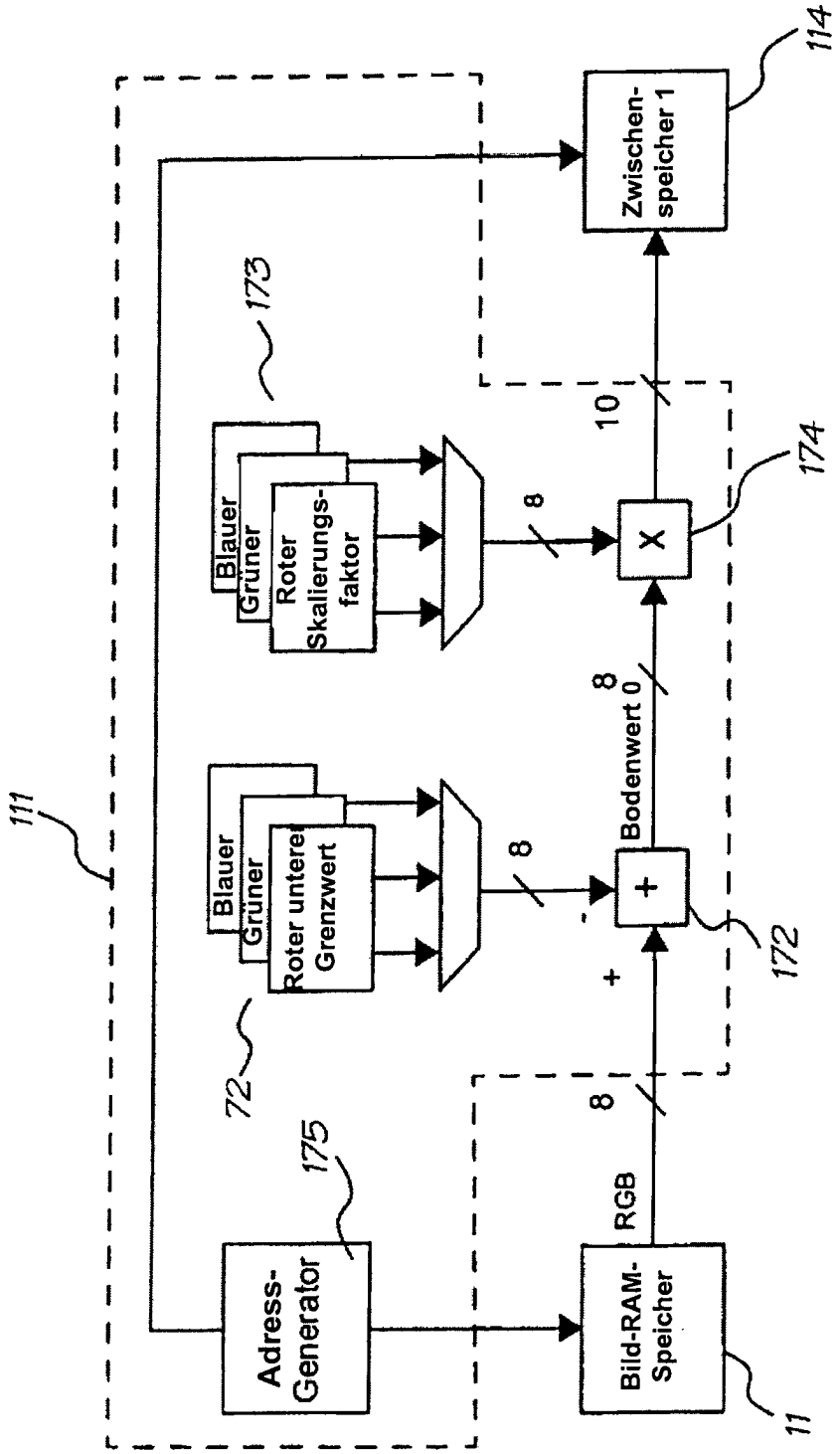

111
Zwischen-speicher 1
114
173
Blauer
Grüner
Roter Skalierungs-faktor
8
10
174
X
Blauer
Grüner
Roter unterer Grenzwert
8
Bodenwert 0
8
+
-
172
72
8
175
RGB
Adress-Generator
Bild-RAM-Speicher
11

FIG. 66

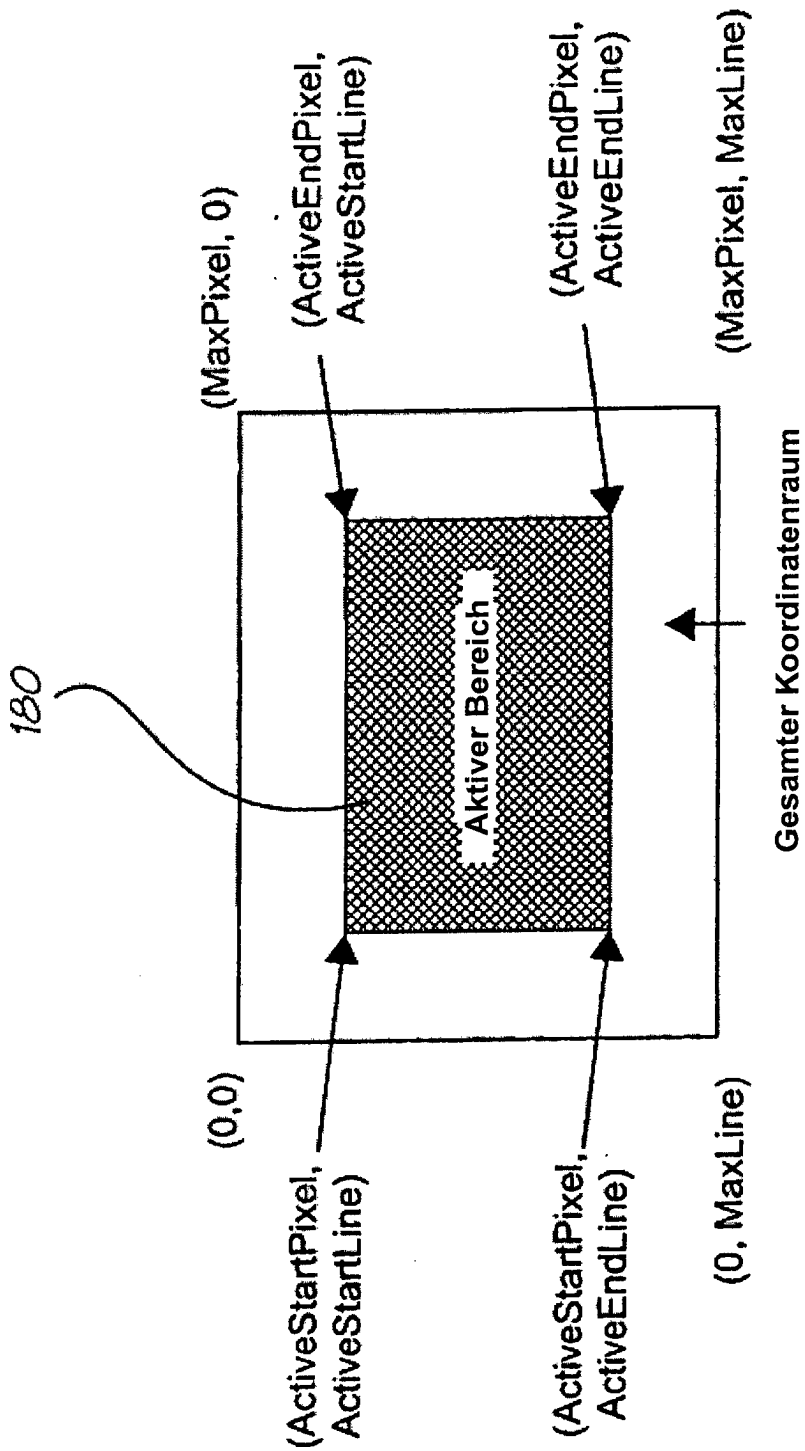

180
(MaxPixel, 0)
(ActiveEndPixel, ActiveStartLine)
(ActiveEndPixel, ActiveEndLine)
(MaxPixel, MaxLine)
Aktiver Bereich
Gesamter Koordinatenraum
(0,0)
(ActiveStartPixel, ActiveStartLine)
(ActiveStartPixel, ActiveEndLine)
(0, MaxLine)

FIG. 67