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

(10) **DE 697 32 271 T2** 2006.01.05

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 0 814 614 B1**
(21) Deutsches Aktenzeichen: **697 32 271.8**
(96) Europäisches Aktenzeichen: **97 304 069.4**
(96) Europäischer Anmeldetag: **11.06.1997**
(97) Erstveröffentlichung durch das EPA: **29.12.1997**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **19.01.2005**
(47) Veröffentlichungstag im Patentblatt: **05.01.2006**

(51) Int Cl.[8]: ***H04N 7/30*** (2006.01)

(30) Unionspriorität:
666964 **19.06.1996** **US**

(84) Benannte Vertragsstaaten:
DE, ES, FR, GB

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

(72) Erfinder:
Hintzman, Jeffrey A., Corvallis, US; Jung, Brian R., Poway, US

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

(54) Bezeichnung: **Hochbitrate Huffman Dekodierung**

Beschreibung

Hintergrund der Erfindung

1. Gebiet der Erfindung

[0001] Die vorliegende Erfindung bezieht sich im Allgemeinen auf eine Datenkomprimierung und -Dekomprimierung unter Verwendung einer Huffman-Codierung, insbesondere auf eine Huffman-Codierung bei Standbild-(JPEG) und Videobild-(MPEG)Anwendungen und speziell auf eine Huffman-Decodierung mit hoher Bitrate.

2. Beschreibung der verwandten Technik

[0002] Das Verdichten von Daten, das auch als Datenkomprimierung bezeichnet wird, entweder zur Übertragung oder zur langfristigen Speicherung kann unter Verwendung von Variable-Länge-Codiertechniken erreicht werden. Datenbitzeichenfolgen fester Länge werden in Bitzeichenfolgen variabler Länge codiert, wobei häufiger auftretende Datenbitzeichenfolgen oder Wörter durch Codewörter kürzerer Länge dargestellt werden, wodurch die Übertragungszeit- oder Speichereinrichtungsanforderungen verringert werden. Komprimierte Daten können durch einen Datenprozessor in dieser verdichteten Form nicht verwendet werden. Deshalb werden dieselben in ihre Form fester Länge zurück decodiert, was auch als Dekomprimierung bekannt ist.

[0003] Eine Form von Variable-Länge-Codes mit minimaler Redundanz wurde durch D. A. Huffman in einem Artikel mit dem Titel „A Method for the Construction of Minimum Redundancy Codes," Protokoll IEEE, IRE, Bd. 40 (9), S. 1098–1101, copr. 1952 vorgeschlagen.

[0004] Bei einer Huffman-Codierung und -Decodierung erhalten Symbole mit einer höheren Auftrittswahrscheinlichkeit Codes kürzerer Bitlänge. Die Huffman-Codierung ist weit verbreitet, da dieselbe eine einfache Binärbaumkonstruktion und eine optimale Durchschnittscodewortlänge liefert.

[0005] Verschiedene Techniken wurden entwickelt, um Huffman-Codes zu decodieren. Im Grunde wird eine Decodierung eines Stroms von Huffman-Codes vorgenommen, indem einem Binärdecodiererbaum gefolgt wird. Im Allgemeinen decodieren diese Dekomprimierungsmaschinen einen Huffman-Code entweder iterativ, immer ein oder zwei Bits gleichzeitig unter Verwendung einer sequenziellen Logik, oder parallel, wobei der gesamte Code in einem Taktzyklus unter Verwendung einer kombinatorischen Logik decodiert wird.

[0006] In der Technik der digitalen Druckkopiewiedergabe wurde ein Standard übernommen, der Huffman-Codes zur Datenkomprimierung verwendet. Dieser Standard wurde vorgeschlagen durch die Joint Picture Expert Group (JPEG – Verbund der Gruppe fotografischer Experten) des ANSI X3L2.8-Ausschusses, veröffentlicht als: "JPEG Digital Compression and Coding of Continuous-tone Still Images," Entwurf ISO 10918, 1991. Ein ähnlicher Standard wurde vorgeschlagen durch die Moving Picture Expert Group (MPEG – Bewegtbildexpertengruppe), veröffentlicht als: ISO/IEC JTC1 CD 11172, „Coding of Moving Pictures and Associated Audio for Digital Storage Media Up to 1.5 Mbits/second," Internationale Standardisierungsorganisation, 1992 (bekannt als „MPEG-1").

[0007] Bei Hochbitratenanwendungen, wie zum Beispiel Vollfarbdruckkopiedruckern, Farbbildübertragung bei Faxen oder einer digitalen Videoübertragung und dergleichen, ist ein Aufbau eines geeigneten Huffman-Decodierers entscheidend. Sowohl ein Pipeline- als auch ein Paralleldecodieren ist schwierig aufgrund der Notwendigkeit einer Rückkopplungsschleife, die erforderlich ist, um den Bitstrom wieder auszurichten. Der Variable-Länge-Code muss in seine ursprüngliche Datenzeichenfolge fester Länge decodiert werden unter Verwendung einer programmierbaren Codenachschlagetabelle. Diese Operation muss bevorzugt in einem Zyklus abgeschlossen sein, bevor das nächste Codewort decodiert werden kann, wobei diese Aufgabe durch die Beschaffenheit variabler Länge jedes aufeinander folgenden Codeworts kompliziert wird. Außerdem muss, um bei hohen Taktfrequenzen wirksam zu sein, die Logikmenge auf dem kritischen Weg minimiert sein.

[0008] Bei der spezifischen Anwendung von JPEG/MPEG-Daten ergeben sich zusätzlich zu der eigentlichen Huffman-Decodierungsoperation weitere Komplexitäten. Huffman-Codewörter – die im JPEG-Standard auf ein Maximum von 16 Bits beschränkt sind – werden verwendet, um ein „Begrenzte-Lauflänge-(RLL-)/Koeffizient"-Datenpaar darzustellen. Die RLL stellt die Anzahl von aufeinander folgenden Nullkoeffizienten dar, die dem aktuellen Koeffizienten unmittelbar vorausgehen, der eine Größe aufweist, die die Anzahl von Bits darstellt, die verwendet werden, um die aktuellen Koeffizientendaten zu codieren. Wie es somit in Fig. 1 gezeigt ist, folgt jedem Huffman-Codewort variabler Länge **103** ein codierter Koeffizientenwert variabler Länge **105**, wobei das Huffman-Codewort **103** angibt, wie viele Bits verwendet sind, um den Koeffizientenwert zu codieren. Zum Beispiel kann es sich bei den dargestellten Daten um einen mit Vorzeichen versehenen Zwölf-Bit-Koeffizientenwert handeln, wie zum Beispiel einen roten, grünen oder blauen („RGB") oder cyanfarbenen, magentafarbenen, gelben, schwarzen („CMYK") Farbstimulusvektor zur Farbumwandlung eines Bildpixels, 8 Bit mal 8 Bit. Jedes Codewort ist ein Koeffizient, der von einem Pixel in dem Bild trans-

formierte. Die Grundlagen von dreidimensionalen Strukturen für RGB- oder andere Dreifarbsysteme werden in der Literatur erörtert, wie zum Beispiel Principles of Color Technology, Billmeyer und Saltzman, John Wiley & Sons publishers, NY, copr. 1981 (2. Aufl.) und Color Science: Concepts and Methods, Quantitative Data and Formulae, von Wyszecki und Stiles, John Wiley & Sons publishers, NY, 1982 (2. Aufl.). Eine weitergehende Erläuterung ist jedoch für einen Fachmann für ein Verständnis der vorliegenden Erfindung nicht nötig.

[0009] Um noch zusätzlich Komplexität hinzuzufügen, sind im Allgemeinen Anfangsblöcke und Steuermarkierungen **107** in den komprimierten JPEG-/MPEG-Datenstrom eingebettet. Diese Markierungen sind reservierte Datenmuster; das heißt reservierte, bestimmte Zeichenfolgen von Einsen und Nullen, die an Bytegrenzen in dem Datenstrom eingebettet sind. Beispiele wären Markierungen für ABTASTUNGSBEGINN, BILDBEGINN, BILDENDE und NEUSTART. Somit muss ein JPEG-Huffman-Decodierer das Vorhandensein einer Markierung erfassen und die Markierung sowie jegliche Auffüllbits, die eventuell hinzugefügt worden sind, um die Markierung mit einer Bytegrenze auszurichten, entfernen.

[0010] Bei einem Standard-JPEG-Huffman-Decodierer, der eine Nachschlagetabelle von 2^{16} Speicherorten verwendet, werden die maximalen 16 Bits als eine Adresse zu der Nachschlagetabelle verwendet, die den Lauf und die Größe für das bestimmte empfangene Codewort liefern würde. Dies erfordert jedoch einen großen, langsamen Speicher. In anderen Worten müssen der Ort und die Länge des Huffman-Codes bestimmt werden, gefolgt von einer Nachschlagetabelle- oder anderen Decodieroperation, die die Lauflänge und Größe des folgenden Koeffizienten angibt, gefolgt von einem Extrahieren dieser Daten und einer nachfolgenden Verschiebung, um das nächste Codewort zu finden, um die Operation zu wiederholen.

[0011] Um den Adressraum zu verringern, entwickelt Tong et al. in der U.S.-Patentschrift Nr. 5,208,593 ein Verfahren, das eine Reihe von führenden Einsen in dem Codewort verwendet, um einen kleineren Speicher zu indexieren. Ein Hinzufügen von Bits beeinträchtigt das Komprimierungsverhältnis. Um die Geschwindigkeit zu erhöhen, entwickeln Retter et al. in der U.S.-Patentschrift Nr. 5,379,070 eine Methodologie einer parallelen Verarbeitung. Dieses Verfahren umfasst zusätzliche teure Hardwarekomponenten.

[0012] Somit besteht ein Bedarf nach einem Huffman-Decodierverfahren und einer -Architektur hoher Bitrate, die für eine JPEG-/MPEG-Datendekomprimierung eines Bitdatenstromes, wie es in Fig. 1 gezeigt ist, geeignet sind.

[0013] Die US-A-5,343,195 beschreibt einen Variable-Länge-Decodierer, der einen Pufferspeicher, einen Bitstellenverschieber und eine Mehrzahl von ODER-Gattern aufweist, die einem Zwischenspeicher Signale liefern, dessen Ausgang mit dem Adresseingang einer Nachschlagetabelle gekoppelt ist. Diese Kombination liefert den Angaben gemäß eine schnelle Anlegung aufeinander folgender Codewörter variabler Länge an die Nachschlagetabelle.

Zusammenfassung der Erfindung

[0014] In ihren Grundaspekten liefert die vorliegende Erfindung eine Huffman-Decodierervorrichtung, wie dieselbe in Anspruch 1 definiert ist.

[0015] In einem weiteren Grundaspekt der vorliegenden Erfindung wird ein Verfahren zum Decodieren eines codierten Datenstroms geliefert, wie dasselbe in Anspruch 5 definiert ist.

[0016] Es ist ein Vorteil der vorliegenden Erfindung, dass die Logik, die in dem kritischen Decodierweg benötigt wird, minimiert ist.

[0017] Es ist ein weiterer Vorteil der vorliegenden Erfindung, dass ein Codewortkoeffizientenpaar bei jedem Taktzyklus decodiert wird.

[0018] Es ist ein weiterer Vorteil der vorliegenden Erfindung, dass Direktzugriffsspeicheranforderungen für eine kommerzielle Implementierung annehmbar sind.

[0019] Es ist ein weiterer Vorteil der vorliegenden Erfindung, dass Steuermarkierungen richtig erfasst werden und auf dieselben reagiert wird, wie es für eine Konformität mit JPEG- und MPEG-Standards notwendig ist.

[0020] Andere Merkmale und Vorteile der vorliegenden Erfindung werden bei einer Betrachtung der folgenden Erläuterung und der beiliegenden Zeichnungen ersichtlich, in denen gleiche Bezugszeichen gleiche Merkmale in den Zeichnungen darstellen.

Kurze Beschreibung der Zeichnungen

[0021] Fig. 1 ist eine schematische Darstellung eines komprimierten JPEG-Datenstroms.

[0022] Fig. 2 ist ein schematisches Blockdiagramm eines Huffman-Decodierers gemäß der vorliegenden Erfindung.

[0023] Fig. 3A und Fig. 3B sind ein detailliertes schematisches Logikdiagramm des Huffman-Decodierers gemäß der vorliegenden Erfindung, wie in Fig. 2 gezeigt.

[0024] Die Zeichnungen, auf die in dieser Beschreibung Bezug genommen wird, sollen nicht so verstanden werden, dass dieselben im richtigen Maßstab gezeichnet sind, es sei denn, es wird speziell darauf hingewiesen.

Beschreibung des bevorzugten Ausführungsbeispiels

[0025] Es wird nun im Detail Bezug genommen auf ein spezifisches Ausführungsbeispiel der vorliegenden Erfindung, das die beste Ausführung veranschaulicht, die derzeit von den Erfindern zum Praktizieren der Erfindung betrachtet wird. Alternative Ausführungsbeispiele werden gegebenenfalls ebenfalls kurz beschrieben, Es wird ein exemplarisches Ausführungsbeispiel mit Hinblick auf den JPEG-Standard bereitgestellt. Ein Fachmann wird jedoch erkennen, dass die Erfindung auch bei MPEG oder einer anderen Datendecodierung angewendet werden kann. Die Verwendung des exemplarischen Ausführungsbeispiels soll keine Einschränkung des Schutzbereiches der Erfindung, wie derselbe durch die Ansprüche dargelegt ist, sein und eine derartige Einschränkung soll auch nicht gefolgert werden.

[0026] Ein spezifisches exemplarisches Ausführungsbeispiel der vorliegenden Erfindung ist in Fig. 2 gezeigt.

Allgemeine Operation

[0027] Im Überblick wird der komprimierte JPEG-Datenstrom in Wörtern einer Zeichenfolge von 32 Bits – einem einzigen Huffman-Codewortkoeffizientenpaar maximaler Länge – an einem Bus **200** in einen Koeffizientendecodierer **201** eingegeben. Bytegrenzeninformationen, ein Wort einer Zeichenfolge von 4 Bits, werden über einen Bus **202** in einen RLL-Detektor **203** eingegeben. Anfangsblock-/Markierungscode-4-Bit-Zeichenfolge-Wörter werden über einen Bus **204** zum Decodieren in einem Markierungsdetektor **205** eingegeben. Im Allgemeinen wird nur ein Eingangsschieberegister für die Eingangsdatenzeichenfolge verwendet.

[0028] Am Beginn eines Bildes werden zwei 32-Bit-Eingangswörter geladen. Der zu verschiebende Betrag wird am Beginn eines Bildes auf Null gelöscht, sodass bei dem ersten Zyklus keine Schiebeoperation durchgeführt wird. Bei jedem nachfolgenden Taktzyklus werden die Daten um die Anzahl von Bits in einem vorhergehenden Huffman-Codewortkoeffizientenpaar verschoben.

[0029] Eine logische ODER-Operation wird dann zum Decodieren verwendet. Dabei handelt es sich um eine relativ schnelle Operation, da es dieselbe überflüssig macht, in dem Datenteilsatz anzuzeigen, wo das nächste Codewort beginnt, wodurch der kritische Weg verkürzt wird. Das Problem besteht jedoch darin, wie die Bytegrenzeninformationen zurückgehalten werden, da die Position dieser Informationen jedes Mal verloren geht, wenn eine Schiebe- und ODER-Operation mit den vorliegenden Eingangsdaten durchgeführt wird. Die Bytegrenzeninformationen sind offensichtlich entscheidend, wenn ein/eine Datenstromanfangsblock/-Markierung angetroffen wird, wie zum Beispiel eine JPEG-Markierung NEUSTART oder BILDENDE. Dies wird deshalb bei der Handhabungsimplementierung des parallelen RLL-Detektors **202** und des Markierungsdetektors **204** berücksichtigt.

Eingangsdaten – Laden

[0030] Mit Bezugnahme auf die Fig. 3A und Fig. 3B wird ein spezifisches Ausführungsbeispiel eines Hochgeschwindigkeits-Huffman-Decodierers **300** gemäß der vorliegenden Erfindung gezeigt, einschließlich einer Lösung für das Dilemma, Bytegrenzen- und Datenstrommarkierungsinformationen in richtiger Ausrichtung aufrechtzuerhalten. Eine Standardlogik ist vorgestellt, wie es für einen Fachmann zu erkennen ist. Die Huffman-Decodierungstechnik selbst verwendet ein Schema zum Zugreifen auf eine Nachschlagetabelle, basierend auf der Anzahl von führenden Einsen in dem Huffman-Codewort. Dieses Decodierungsverfahren ist in der U.S.-Patentschrift Nr. 4,899,149 (Kahan) und in der U.S.-Patentschrift Nr. 5,208,593 (Tong) beschrieben. Somit konzentriert sich die folgende Beschreibung auf die Beschreibung der Erfindung, die hier beansprucht wird.

[0031] Das 32-Bit-Eingangsdatenwort, das an dem Bus **200** empfangen wird, wird in ein Datenregister „Q" **301** geladen, ein 75-Bit-Datenregister für das Segment des Eingangsdatenstroms, das derzeit decodiert wird. 75 Bits sind notwendig, da nach einem Verschieben die elf Bits ganz links den Koeffizienten ohne Vorzeichen von dem vorhergehenden Codewortkoeffizientenpaar halten. Zusätzlich zu diesem Satz von elf Bits werden zwei volle 32-Bit-Wörter von dem aktuellen Eingangsdatenstrom in die Warteschlange gestellt. Zum Decodieren eines „nächsten" Huffman-Codeworts ist es notwendig, das Codewort aus dem Datenstrom in das Q_Register **301** zu extrahieren. Bei dem JPEG-Standard kann dieses Codewort bis zu 16 Bits lang sein. Das vorhergehende Codewortkoeffizientenpaar kann bis zu 27 Bits lang sein (d. h. 16 Bits für das Codewort und 11 Bits für den codierten Koeffizienten). Somit ist es bei jedem beliebigen gegebenen Zyklus notwendig, ein Minimum von 43 Bits (16 Bits + 27 Bits) in dem Q_Register **301** zu haben. Da dreiundvierzig mehr ist als eine 32-Bit-Eingangsdatenwortlänge, wurde bestimmt, dass es eine rasche Implementierung ist, zwei 32-Bit-Wörter in die Warteschlange zu stellen.

Anfangsblock/Markierungen/Bytegrenzeninformationsbits – Laden

[0032] Anfangsblock/Markierungen und Bytegrenzeninformationen in dem Eingangsdatenstrom unter den 32-Bit-Wörtern werden auf eine standardmäßige Weise erfasst, wie es in der Technik bekannt ist, und als 4-Bit-Wörter jeweils auf die Busse **204**, **202** auseinander getrennt. Sowohl der RLL-Detektor **203** als auch der Anfangsblock-/Markierungsdetektor **205** erweitert dann jedes jeweilige 4-Bit-Eingangswort auf 32 Bits durch ein Einfügen von sieben Nullen zwischen jedem der vier Bits jeweils an einer „Nulleinfüge"-Hardware **303**, **305**.

[0033] Bei dem exemplarischen Ausführungsbeispiel werden im JPEG-Standard definierte Markierungen NEUSTART und BILDENDE („EOI") an Bytegrenzen in den komprimierten JPEG-Datenstrom eingefügt. Diese JPEG-Markierungen werden durch einen Markierungsdetektor **205** erfasst, und vier Bits werden ausgegeben, um anzuzeigen, an welcher Bytegrenze in dem 32-Bit-Eingangswort die Markierung erfasst wurde. Die Markierung selbst wird aus dem Eingangsdatenstrom entfernt. Diese 4-Bit-Markierungsidentifikation wird auf volle 32 Bits erweitert, um mit dem Eingangsdatenwort übereinzustimmen, durch das Einfügen von sieben Nullen zwischen jedem der vier Bits. Die Bitposition der Markierungsidentifiziererbits stimmt somit genau mit dem Punkt in dem Eingangsdatenwort überein, wo die Markierung erfasst und entfernt wurde.

[0034] Ein ähnlich erweitertes 32-Bit-RLL-Wort wird in ein Register R **307** geladen. Das erweiterte 32-Bit-NEUSTART- oder EOI-Markierungswort wird in ein Register „E" **309** geladen. Nun können die Register Q, R und E **301**, **307**, **309** synchron derart verschoben werden, dass Bytegrenzen- und Anfangsblock-/Markierungsorte in der gleichen Reihenfolge bewahrt werden, wie es ursprünglich in den Eingangsdaten codiert war. Das heißt, wenn nachfolgende Schiebeoperationen stattfinden, die im Folgenden erläutert sind, werden das Eingangsdatenstrom-Q_Register **301**, ein RLL-Wort in dem R_Register **307** und ein NEUSTART- oder EOI-Markierungswort in dem E_Register **309** zusammen verschoben und bleiben miteinander in Synchronisation.

Eingangsdaten – Decodierung

[0035] Ein „nächstes" nachfolgendes 32-Bit-Datenwort des Eingangsdatenstroms von dem Bus **200** wird um einen Betrag, der gleich der Anzahl von Datenbits ist, die in dem 75-Bit-Datenregister **301** vorhanden sein werden, nachdem die „aktuelle" Linksschiebeoperation verarbeitet ist, nach rechts verschoben **313**. Dabei handelt es sich um eine Vorbereitung für eine bitweise ODER-Operation **315** in das Q_Register **301**. Die Rechtsschiebeoperation **313** plus die bitweise ODER-Operation **315** haben den Nettoeffekt, das neue 32-Bit-Datenwort in die richtige Position unmittelbar rechts von allen Datenbits, die sich momentan in dem Q_Register **301** befinden, zu laden.

[0036] Das „nächste" Wort wird dann bitweise in einer ODER-Operation **315** in das Q_Register **301** geladen.

[0037] Das mit 75 Bits gefüllte Q_Register **310** wird dann in einem Register **317** um die Größe der Summe des ersten Codeworts plus Koeffizient nach links verschoben.

[0038] Die 27 Bits ganz links in dem nach links verschobenen Datenstrom werden dann untersucht. Von diesen 27 Bits sind die 11 Bits ganz links in dem Register **319** der Koeffizient des ersten Codewortkoeffizientenpaars. Die Bits 63:48 enthalten den Variable-Länge-Huffman-Code, der gemäß dem JPEG-Standard bis zu 16 Bits lang sein kann, wobei angenommen wird, dass das Bit ganz links des Registers Bit_{74} ist und das Bit ganz rechts Bit_0 ist.

[0039] Ein Decodieren des Huffman-Codes unter Verwendung der Nachschlagetabelle ergibt die Länge des kombinierten Huffman-Codeworts und des Koeffizienten ohne Vorzeichen. Bei dem nächsten Taktzyklus wird das Q_Register **301** um diesen Betrag nach links verschoben.

[0040] Nach der Linksverschiebung enthalten die 11 Bits ganz links des Q_Registers **301** den Koeffizienten variabler Länge ohne Vorzeichen von dem vorhergehenden Codewortkoeffizientenpaar. Der Koeffizient variabler Länge ohne Vorzeichen kann eine beliebige Länge von null bis elf Bits aufweisen, das KOEFF_GRÖßE_A-Register **327** enthält die Länge dieses Koeffizienten. Die KOEFFIZIENT_WIEDERHERSTELLEN-Einheit **331** nimmt den Koeffizienten ohne Vorzeichen und die Koeffizientenlänge und erzeugt den wiederhergestellten Koeffizientenwert als eine mit einem Vorzeichen versehene 12-Bit-Größenausgabe an die CODIER_DECODIER_HUFFMAN-Einheit **333**.

[0041] Es sei darauf hingewiesen, dass, falls der Koeffizient eine geringere Länge als 11 Bits aufweist, die Informationen in den Bits ganz rechts der 11 Bits ganz links des Q_Registers **301** enthalten sind. Somit werden über eine standardmäßige Boolesche Logik Codierter-Koeffizient- und Koeffizientengröße-Daten verwendet, um den codierten Koeffizienten zu einer mit einem Vorzeichen versehenen 12-Bit-Größe **331** wiederherzustellen, die an die nächste Stufe des JPEG-Decodierers (nicht gezeigt) ausgegeben wird **333**.

[0042] Die 16 Bits ganz rechts sind das aktuelle Huf-

fman-Codewort, das an einem Adressgenerator **321** gesendet wird, um die Adresse zu berechnen, um auf die Huffman-Nachschlagetabelle **323** zuzugreifen. Die Nachschlagetabelle **323** liefert dann die Nulllauflänge **325** (die Anzahl von Nullen, die in den Koeffizienten eingefügt sind, die wieder in die decodierten Daten eingefügt werden müssen) und die Koeffizientengrößendaten **327** sowie die Gesamtlänge des aktuellen Codewortkoeffizientenpaares, gleich dem Betrag der nächsten Linksverschiebung **329**. Das heißt, die Anzahl von Positionen, um die das Linksschieberegister **317** für den nächsten Untersuchungszyklus zu verschieben ist.

[0043] Die Nachschlagetabelle decodiert das Huffman-Codewort von dem Eingangsdatenstrom und liefert drei Werte:

(1) die Lauflänge von Nullkoeffizienten,

(2) die Länge des codierten Koeffizienten variabler Länge, und

(3) die Summe der Länge des Huffman-Codeworts und der Länge des codierten Koeffizienten.

[0044] Der letzte Wert wird unmittelbar verwendet, um das Q_Register **301** nach links zu verschieben. Die Länge des codierten Koeffizienten variabler Länge wird verwendet, um den Koeffizienten zu seinem mit einem Vorzeichen versehenen 12-Bit-Wert wiederherzustellen. Die Lauflängeninformationen werden direkt an den Lauflängendecodierer **363** ausgegeben, verzögert durch das LAUF_LÄNGE_A-Register **325** und das LAUF_LÄNGE_B-Register **326**, da bei der pipelineartigen Implementierung die Lauflängen- und Koeffizientengrößeninformationen verzögert werden müssen, um mit dem richtigen Koeffizienten von dem CODIER_DECODIER_HUFFMAN-Register **333** übereinzustimmen.

[0045] Da die Eingangsdaten von variabler Länge sind, wird ein Gültige_Bits-Register **311** verwendet, um zu verfolgen, wie viele gültige Bits in jedem der Q_Register-**301**-Daten, R_Register-**307**-Bytegrenzeninformationen und E_Register-**309**-Kopfblock/Markierung vorhanden sind. Wenn Bits aus den Registern **301**, **307**, **309** herausgeschoben werden, muss der Wert der Verschiebung – der in jedem Zyklus durch die Nachschlagetabelle geliefert wird – von dem Gültiges_Bit-Zählwert abgezogen werden.

[0046] Wenn ein neues 32-Bit-Wort von Eingangsdaten geladen wird, wird bei **312** dann 32 zu dem Gültige_Bits-Zählwert addiert. Die Ladesteuerung **314** überwacht die Anzahl von gültigen Datenbits in dem Q_Register **301** und den Betrag an Linksverschiebung, der bei dem nächsten Taktzyklus angewendet wird. Wenn die Ladesteuerung **314** berechnet, dass die Anzahl von gültigen Datenbits in dem Q_Register **301** unter 43 fallen wird, gibt dieselbe ein Signal aus, um ein neues Datenwort in das Q_Register zu laden (d. h. falls Anzahl von gültigen Datenbits minus Verschiebungsbetrag ist gleich weniger als dreiundvierzig, neues Datenwort laden).

Anfangsblock/Markierungen/Bytegrenzeninformationsbits – Decodierung

[0047] Es sei nun aus der vorhergehenden Beschreibung ins Gedächtnis zurückgerufen, dass erweiterte 32-Bit-Kopfblock/Markierungen und -Bytegrenzenwörter in dem E_Register **309** bzw. den R_Register **307** erzeugt wurden. Diese 32-Bit-Größen werden jeweils in Schieberegistern **345**, **343** nach rechts verschoben und bitweise jeweils einer ODER-Operation **347**, **349** unterzogen, in exakt der gleichen Weise und dem gleichen Taktzyklus wie ihr zugehöriges Eingangsdatenwort in dem Q_Register **301**. Dann werden, wenn die Daten nach links verschoben werden **317**, die Kopfblock-/Steuermarkierungen und Bytegrenzenwörter durch jeweilige Register **351**, **353** zur gleichen Zeit nach links verschoben, wobei die relative Position damit aufrechterhalten wird. Durch ein Untersuchen der Bits ganz links dieser Register **351**, **353** können Nullen, die in dem E_Register **309** und dem R_Register **307** eingefügt wurden, erfasst **355** und nach links aus dem Datenstrom heraus verschoben werden **357**, und die Kopfblock/Markierungen 365 und Grenzwörter **363** können dementsprechend ausgegeben werden.

[0048] Zusammenfassend codiert die vorliegende Erfindung somit den Lauf, die Größe des Koeffizienten und die Addition des Koeffizienten und der Codewortlänge. Dieser Wert ergibt sich als der Schiebebetrag, der den Betrag der Verschiebung angibt, um zu dem nächsten Codewort zu gelangen. Dies stellt eine schnellere Methodologie zum Decodieren von codierten JPEG-Huffman-Bit-Strömen dar, da die Decodierung erfolgt, während neue Daten empfangen werden. Gleichzeitig werden Bytegrenzen und Kopfblock/Markierungen ordnungsgemäß durch Schaltungsanordnung **203**, **205** verfolgt, die die Datenhandhabung in dem Huffman-Decodierer **201** nachahmt. Wenn eine EOI-Markierung erfasst ist, fährt der Decodierer fort zu verarbeiten, bis die empfangenen Daten erschöpft sind.

[0049] Die vorhergehende Beschreibung des bevorzugten Ausführungsbeispiels der vorliegenden Erfindung wurde zu Zwecken der Veranschaulichung und Beschreibung präsentiert. Dieselbe soll nicht erschöpfend sein oder die Erfindung auf die genaue offenbarte Form beschränken. Es ist offensichtlich, dass viele Modifizierungen und Variationen für praktizierende Fachleute ersichtlich sein werden. In ähnlicher Weise können alle beschriebenen Prozessschritte mit anderen Schritten austauschbar sein, um das gleiche Ergebnis zu erzielen. Das Ausführungsbeispiel wurde ausgewählt und beschrieben, um die Grundsätze der Erfindung und ihre praktische An-

wendung in der besten Ausführung bestmöglich zu erläutern, um es dadurch anderen Fachleuten zu ermöglichen, die Erfindung für verschiedene Ausführungsbeispiele und mit verschiedenen Modifizierungen, wie dieselben für die bestimmte beabsichtigte Verwendung geeignet sind, zu verstehen.

Patentansprüche

1. Eine Huffman-Decodierervorrichtung zum Decodieren eines JPEG-Standard-Datenstroms (**103**, **105**, **107**), der Huffman-Codewörter (**103**) von maximal 16 Bit in einem Huffman-Codewortkoeffizientenpaar-Datenformat aufweist, wobei der Datenstrom ferner Bytegrenzeninformationscodes (**107**) und Anfangsblock-/Markierungsinformationscodes (**107**) umfasst, wobei die Vorrichtung folgende Merkmale umfasst:
einen Eingangsbus (**200**);
ein Rechtsschieberegister (**313**), das mit dem Eingangsbus (**200**) zum Empfangen von zumindest zwei Codewortkoeffizientenpaaren von demselben verbunden ist;
eine bitweise logische ODER-Einrichtung (**315**), die mit dem Rechtsschieberegister (**313**) zum Empfangen einer Rechtsschieberegisterdatenausgabe von demselben verbunden ist;
ein Eingangsdatenregister (**301**), das mit der logischen ODER-Einrichtung (**315**) zum Empfangen von Logische-ODER-Einrichtung-Ausgangsdaten von derselben verbunden ist;
ein Linksschieberegister (**317**), das mit dem Eingangsdatenregister (**301**) zum Empfangen von Ausgangsdaten von demselben verbunden ist;
eine Einrichtung (**331**, **333**, **363**) zum Extrahieren eines aktuellen Codeworts und eines zu decodierenden Koeffizienten von einem Codewortkoeffizientenpaar aus dem Linksschieberegister (**317**), derart, dass eine Lauflänge von Koeffizienten, eine Länge eines codierten Koeffizienten variabler Länge und eine Summe der Länge eines Codeworts plus der Länge eines codierten Koeffizienten erhalten wird; und
eine Rückkopplungseinrichtung zum Linksverschieben der Daten des Linksschieberegisters (**317**) um den Betrag der Summe, um ein nächstes aktuelles Codewort und einen zu decodierenden Koeffizienten zu erhalten.

2. Die Vorrichtung gemäß Anspruch 1, die ferner folgende Merkmale aufweist:
eine Einrichtung (**203**) zum Empfangen und zum Verfolgen der Bytegrenzeninformationscodes, die folgende Merkmale umfasst:
eine Einrichtung (**303**) zum Auffüllen der Bytegrenzeninformationscodes auf die gleiche Bitlänge wie das Codewortkoeffizientenpaar, und
eine Einrichtung (**353**) zum Verschieben des Bytegrenzeninformationscodes synchron zu den Codewortkoeffizientenpaaren.

3. Die Vorrichtung gemäß Anspruch 1, die ferner folgende Merkmale aufweist:
eine Einrichtung (**205**) zum Empfangen und zum Verfolgen der Anfangsblock-/Markierungsinformationscodes, die folgende Merkmale umfasst:
eine Einrichtung (**305**) zum Auffüllen der Anfangsblock-/Markierungsinformationscodes auf die gleiche Bitlänge wie das Codewortkoeffizientenpaar, und
eine Einrichtung (**351**) zum Verschieben der Anfangsblock-/Markierungsinformationscodes synchron zu den Codewortkoeffizientenpaaren.

4. Die Vorrichtung gemäß Anspruch 1, bei der das Eingangsregister folgendes Merkmal aufweist:
ein Register, das eine Datenkapazität von zumindest der doppelten Bitbreite des Eingangsbusses aufweist.

5. Ein Verfahren zum Durchführen eines Huffman-Decodierens eines codierten Datenstroms, der Codewortkoeffizientenpaare einer gegebenen maximalen Länge aufweist, durch ein Empfangen und Speichern von Eingangsdatensätzen, wobei jeder Datensatz eine gegebene Anzahl von Bits aufweist, wobei das Verfahren folgende Schritte aufweist:
Empfangen eines ersten und eines zweiten sequentiellen Codewortkoeffizientenpaars des Datenstroms in einem Schieberegister (**313**);
Rechtsverschieben des zweiten Codewortkoeffizientenpaars um die Anzahl von Bits in dem ersten Codewortkoeffizientenpaar;
Durchführen einer logischen ODER-Operation bezüglich des ersten Codewortkoeffizientenpaars und des zweiten Codewortkoeffizientenpaars und Speichern des Ergebnisses in einem Datenregister (**301**), das eine Datenkapazität von zumindest dem Doppelten der gegebenen maximalen Länge aufweist;
Linksverschieben des gespeicherten Ergebnisses in einem Linksschieberegister (**317**);
Extrahieren einer ersten vorbestimmten Anzahl von höchstwertigen Bits des Ergebnisses, das in dem Links-schieberegister (**317**) gespeichert ist, als einen Koeffizientenwert und Extrahieren einer zweiten vorbestimmten Anzahl von wertigen Bits des Ergebnisses, das in dem Linksschieberegister (**317**) gespeichert ist, als ein Codewort, wobei die erste und die zweite vorbestimmte Anzahl in Bezug auf die gegebene maximale Länge bestimmt sind.

6. Das Verfahren gemäß Anspruch 5, das ferner folgenden Schritt aufweist:
Wiederholen der Schritte gemäß Anspruch 5 für jedes nächste sequentielle Eingangscodewortkoeffizientenpaar des Datenstroms, bis der Strom beendet ist.

7. Das Verfahren gemäß Anspruch 5 oder 6, das ferner folgende Schritte aufweist:
Empfangen von Bytegrenzeninformationen, die in dem Datenstrom eingebettet sind, mit einem zugehö-

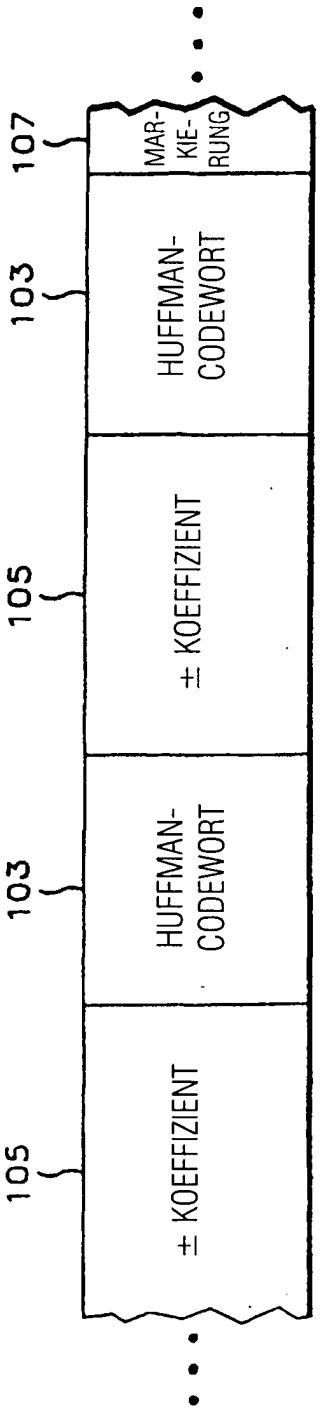
rigen Datensatz;
Auffüllen jedes Bits der Bytegrenzeninformationen mit einer Zeichenfolge von Nullen zwischen den Bits, derart, dass die Bytegrenzeninformationen eine Bitlänge aufweisen, die gleich der Eingangsdatensatzlänge ist;
Rechtsverschieben der Bytegrenzeninformationen gleichzeitig mit dem zugehörigen Datensatz;
Extrahieren jeder der Zeichenfolge von Nullen, um die Bytegrenzeninformationen in Position mit dem decodierten Datensatz neu zu formulieren.

8. Das Verfahren gemäß Anspruch 5, 6 oder 7, das ferner folgende Schritte aufweist:
Empfangen von Steuermarkierungsinformationen, die in dem Datenstrom eingebettet sind, mit einem zugehörigen Datensatz;
Auffüllen jedes Bits der Steuermarkierungsinformationen mit einer Zeichenfolge von Nullen zwischen den Bits, derart, dass die Steuermarkierungsinformationen eine Bitlänge aufweisen, die gleich der Eingangsdatensatzlänge ist;
Rechtsverschieben der Steuermarkierungsinformationen gleichzeitig mit dem zugehörigen Datensatz;
Extrahieren jeder der Zeichenfolge von Nullen, um die Steuerungsmarkierungsinformationen in Position mit dem decodierten Datensatz neu zu formulieren.

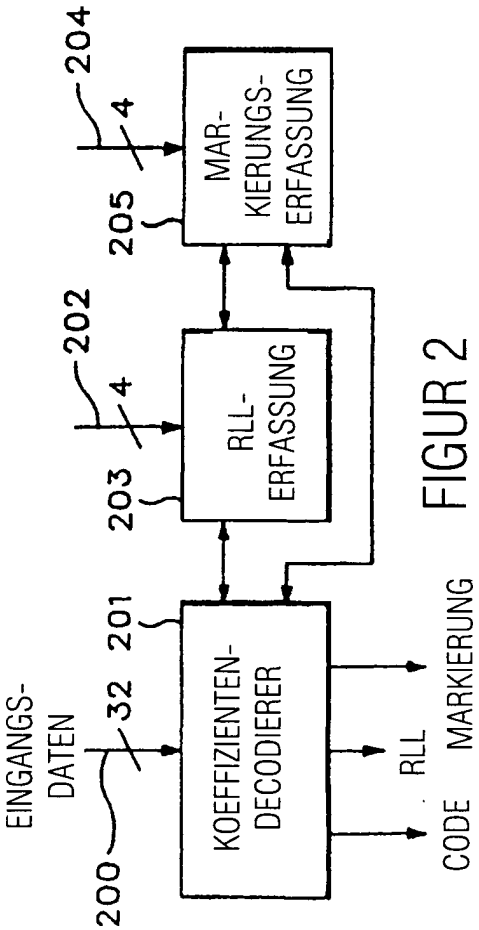
9. Das Verfahren gemäß Anspruch 5, 6, 7 oder 8, das ferner folgenden Schritt aufweist:
bei jedem Taktzyklus, Verschieben der Daten um die Gesamtanzahl von Bits in dem vorhergehenden Codewortkoeffizientenpaar.

Es folgen 3 Blatt Zeichnungen

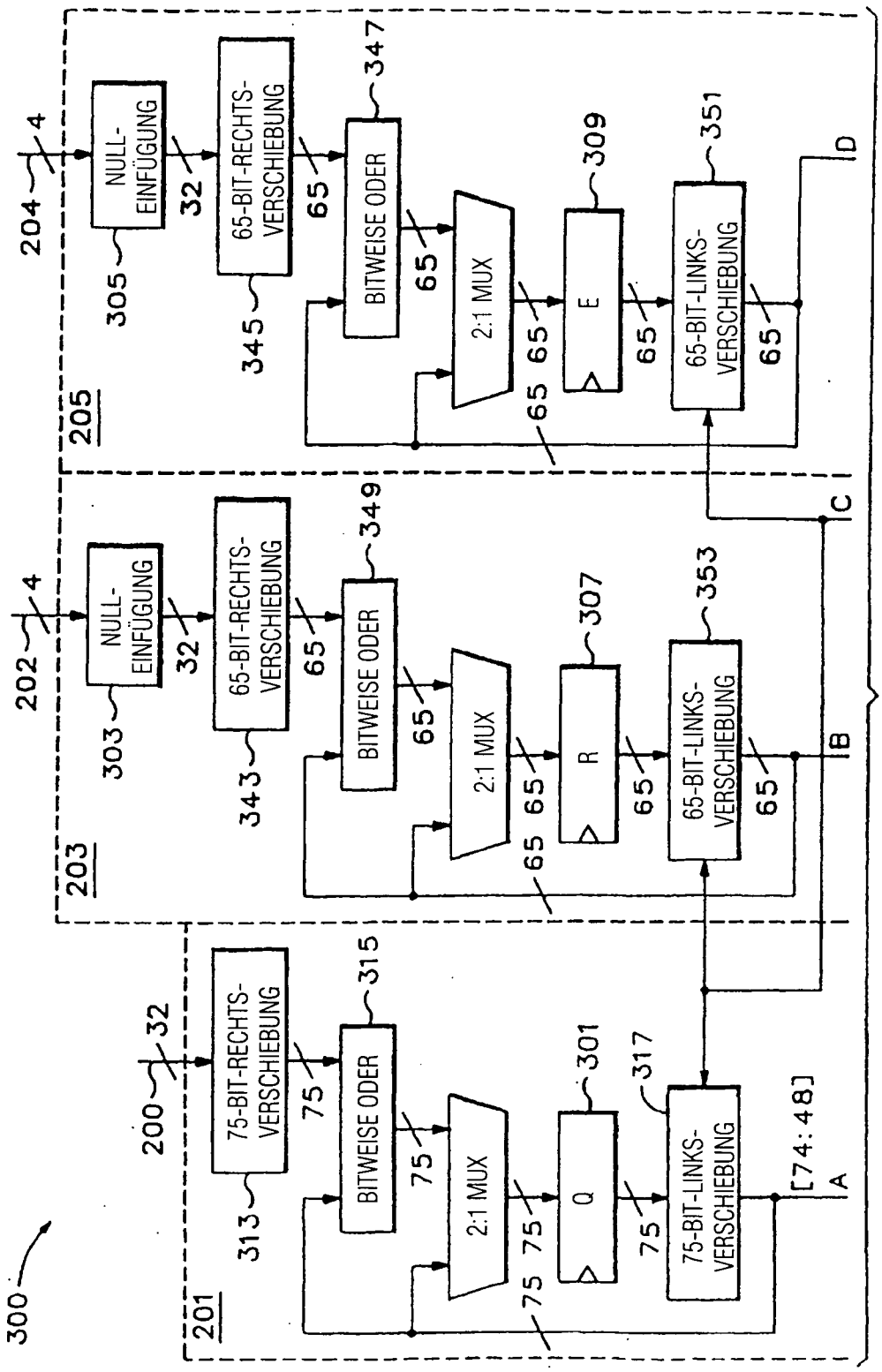
Anhängende Zeichnungen



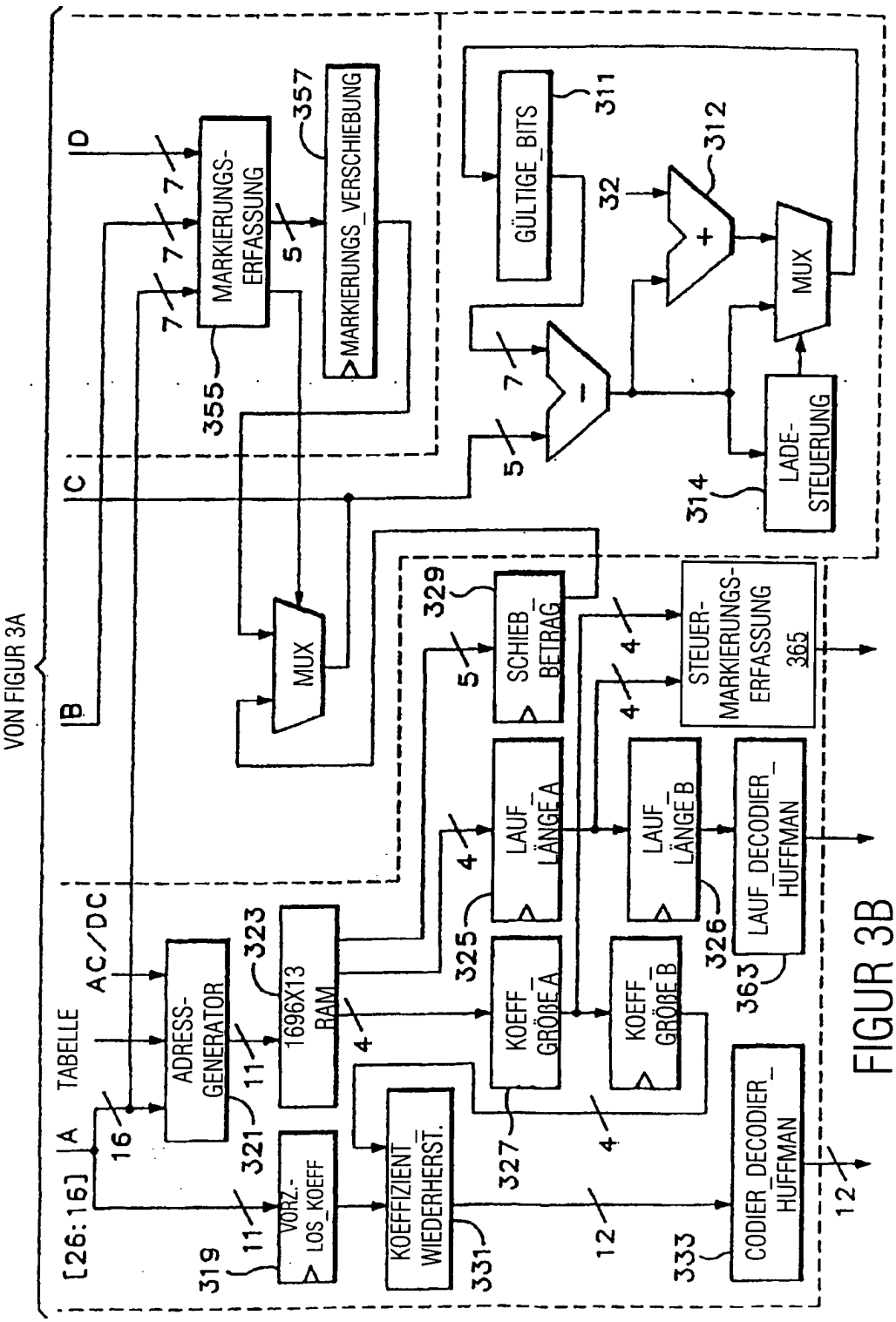
FIGUR 1



FIGUR 2


300
201
203
205
200
32
75-BIT-RECHTS-VERSCHIEBUNG
313
75
BITWEISE ODER
315
75
2:1 MUX
75 75
Q
301
317
75-BIT-LINKS-VERSCHIEBUNG
75
[74:48]
A
202
4
NULL-EINFÜGUNG
303
32
65-BIT-RECHTS-VERSCHIEBUNG
343
65
BITWEISE ODER
349
65
2:1 MUX
65 65
R
307
65
65-BIT-LINKS-VERSCHIEBUNG
353
65
B
204
4
NULL-EINFÜGUNG
305
32
65-BIT-RECHTS-VERSCHIEBUNG
345
65
BITWEISE ODER
347
65
2:1 MUX
65 65
E
309
65
65-BIT-LINKS-VERSCHIEBUNG
351
65
D
C
ZU FIGUR 3B

FIGUR 3A


VON FIGUR 3A
A
B
C
D
[26:16]
TABELLE
AC/DC
ADRESS-GENERATOR
1696X13 RAM
VORZ.-LOS_KOEFF
KOEFFIZIENT WIEDERHERST.
MARKIERUNGS-ERFASSUNG
MARKIERUNGS_VERSCHIEBUNG
MUX
GÜLTIGE_BITS
LADE-STEUERUNG
SCHIEB_BETRAG
LAUF_LÄNGE A
LAUF_LÄNGE B
KOEFF_GRÖßE A
KOEFF_GRÖßE B
STEUER-MARKIERUNGS-ERFASSUNG 365
LAUF_DECODIER_HUFFMAN
CODIER_DECODIER_HUFFMAN
319
321
323
325
326
327
329
331
333
355
357
311
312
314
363

FIGUR 3B