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

(10) **DE 695 34 890 T2** 2006.08.17

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 0 752 694 B1**
(21) Deutsches Aktenzeichen: **695 34 890.6**
(96) Europäisches Aktenzeichen: **95 304 644.8**
(96) Europäischer Anmeldetag: **03.07.1995**
(97) Erstveröffentlichung durch das EPA: **08.01.1997**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **22.03.2006**
(47) Veröffentlichungstag im Patentblatt: **17.08.2006**

(51) Int Cl.[8]: ***G09G 1/16*** (2006.01)

(30) Unionspriorität:
270194 **01.07.1994** **US**

(84) Benannte Vertragsstaaten:
DE, FR, GB, IT, NL

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

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

(72) Erfinder:
Seiler, Larry D., Boylston, Massachusetts 01505, US; Gianos, Christopher C., Sterling, Massachusetts 01564, US; McNamara, Robert S., Bolton, Massachusetts 01740, US; McCormack, Joel J., Woodside, California 94162, US

(54) Bezeichnung: **Verfahren zum schnellen Ausmalen und Kopieren von Bildelementen kurzer Wortlänge in einem breiteren Rasterpufferspeicher**

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

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

Beschreibung

GEBIET DER ERFINDUNG

[0001] Diese Erfindung bezieht sich allgemein auf das Gebiet von Computersystemen und im Einzelnen auf ein Verfahren zum Speichern von Graphikinformationen in einem Computersystem.

HINTERGRUND DER ERFINDUNG

[0002] Wie man in der Technik weiß, umfasst ein Computerverarbeitungssystem allgemein eine Zentralverarbeitungseinheit zum Verarbeiten eines Anweisungsstroms, der in einem Speicher oder auf einer Platte gespeichert ist. Eine Vielzahl von Softwareanwendungen können durch das Computerverarbeitungssystem gleichzeitig ausgeführt werden. Eine der Anwendungen steuert die Bilder, die auf einem mit dem Computerverarbeitungssystem gekoppelten Monitor angezeigt werden. Das Computerverarbeitungssystem umfasst oft eine spezialisierte Graphikhardware und -software, um die auf dem Monitor angezeigten Bildinformationen zu steuern.

[0003] Graphikhardware umfasst üblicherweise eine Graphiksteuerung und einen Videorahmenpuffer. Die Graphiksteuerung empfängt Befehle von dem Computerverarbeitungssystem, die Manipulation von Daten in dem Rahmenpuffer zu steuern. Die Graphiksteuerung kann eine Logik umfassen, um die Leistungsfähigkeit bzw. das Verhalten einer Vielzahl typischer Graphikfunktionen, z.B. ein Kopieren von Daten aus einem Speicher in den Rahmenpuffer, ein Ziehen von Linien oder ein Punktieren von Daten, zu steigern.

[0004] Wenn der Computerprozessor eine Operation durchführt, aktualisiert er die Pixeldaten in den Video-RAMs. Die Graphik-Leistungsfähigkeit wird üblicherweise daran gemessen, wie schnell die Graphiksteuerung Daten in den Video-RAMs aktualisieren oder wiedergewinnen kann.

[0005] Üblicherweise weisen alle Pixel in dem Rahmenpuffer, die auf dem Monitor angezeigt werden, dieselbe physische Größe auf, d.h. umfassen die gleiche Anzahl von Bits pro Pixel. Eine Graphiksteuerung, die derart konfiguriert ist, dass jedem angezeigten Pixel 8 Bits zugewiesen sind, wird hiernach als „8Bit-Graphiksystem" bezeichnet. Die Bitadressen aufeinander folgender angezeigter Pixel in einem 8Bit-Graphiksystem lauten p, p + 8, p + 16 usw. Eine Graphiksteuerung, die derart konfiguriert ist, dass jedem angezeigten Pixel 32 Bits zugewiesen sind, wird hiernach als „32Bit-Graphiksystem" bezeichnet. Die Bitadressen aufeinander folgender angezeigter Pixel in einem 32Bit-Graphiksystem lauten p, p + 32, p + 64 usw.

[0006] Jedem angezeigten Pixel können Steuerbits zugeordnet sein, um zu bestimmen, wie die Datenbits dieses Pixels interpretiert werden sollten. Diese Steuerbits können in den Pixeldaten selbst oder in einem separaten Speicherbereich enthalten sein. Bei einem 32Bit-Graphiksystem kann ein Wert für die Steuerbits festlegen, dass Bits 23:16 des Pixels die Intensität der roten Farbe festlegen, Bits 15:8 die Intensität der grünen Farbe festlegen und Bits 7:0 die Intensität der blauen Farbe festlegen. Ein anderer Wert für die Steuerbits kann festlegen, dass Bits 7:0 verwendet werden sollten, um eine Tabelle von 256 Einträgen mal 24 Bits zu indexieren. Die in der Tabelle zu findenden 24 Bits werden dann verwendet, um jeweils 8 Bits für die Intensität roter, grüner und blauer Farbe festzulegen.

[0007] Durch die Verwendung derartiger Steuerbits kann ein 32Bit-Graphiksystem gleichzeitig Anwendungen anzeigen, die für ein 8Bit-Graphiksystem geschrieben werden, sowie Anwendungen, die Pixel mit mehr als 8 Bits an Informationen erfordern. Jedoch lässt die Leistungsfähigkeit der für 8Bit-Pixel geschriebenen Anwendung nach, wenn sie an einem 32Bit-Graphiksystem betrieben wird, da jedem Pixel physisch 32 Bits zugeordnet sind, oder das Vierfache der Speicherung, die bei dem 8Bit-Graphiksystem erforderlich ist. Da viele Graphikoperationen durch die Busbandbreite auf den Videospeicher beschränkt sind, können manche Operationen für 8Bit-Pixel-Anwendungen an einem 32Bit-Pixel-Graphiksystem bis zu viermal langsamer laufen.

[0008] Unter kurzer Bezugnahme auf Fig. 1 ist veranschaulichenderweise ein Videospeicher **75** gezeigt, der 4 Speicherbänke, Bänke **80**–**83**, aufweist, wobei jede Bank 4 RAM-Vorrichtungen umfasst. Jede der RAM-Vorrichtungen speichert 2 Bytes von Pixeldaten. Wie in Fig. 1 zu sehen ist, speichert der Videospeicher **75** 8 Pixel von 32Bit-Graphikdaten.

[0009] Wenn eine visuelle 8Bit-Anwendung in einem 32Bit-Graphiksystem läuft, umfasst üblicherweise lediglich ein Byte jedes 32Bit-Pixels Daten, die durch die meisten Graphikoperationen verändert werden. Wenn die Steuerbits separat von den Pixeldaten gespeichert sind, sind die anderen drei Bytes jedes Pixels vollständig ungenutzt. Wenn die Steuerbits als Teil der Pixeldaten gespeichert sind, dann werden die Steuerbits durch die Anzeigehardware verwendet, um die Interpretation der restlichen Pixel zu steuern. Diese Steuerbits werden relativ selten verändert, z.B. wenn eine Anwendung ein Fenster einblendet, ausblendet oder bewegt. Die Steuerbits werden üblicherweise nicht durch gewöhnliche Zeichenoperationen, die an dem Fenster vorgenommen werden, verändert. Folglich müssen gewöhnliche Zeichenoperationen nur 8 Bits jedes 32Bit-Pixels lesen oder schreiben, auch wenn die Pixeldaten Steuerbits umfassen.

[0010] Wie in Fig. 1 gezeigt ist, kann in einem 32Bit-Graphiksystem bei einem Speicherzugriff lediglich auf Byte 0 von Pixel 0 und Pixel 1 zugegriffen werden, wenn angenommen wird, dass das Byte 0 das relevante Byte von Daten für die 8Bit-Anwendung ist. Somit werden während jeder Speichertransaktion lediglich 16 Bits des 64Bit-Busses **65** verwendet, während 48 Bits des Busses ungenutzt sind. Folglich werden vier Speicherzugriffe auf die Bänke **83** benötigt, um acht 8Bit-Pixel zu lesen oder zu schreiben.

[0011] Im Gegensatz dazu kann in einem 8Bit-Graphiksystem eine 8Bit-Graphikanwendung acht 8Bit-Pixel in lediglich einem Speicherzugriff lesen oder schreiben. Somit kann dieselbe Graphikhardware für dieselbe Graphikanwendung zwei unterschiedliche Leistungsfähigkeitsergebnisse liefern, je nachdem, wie viele Bits jedem angezeigten Pixel zugewiesen sind.

[0012] Es wäre wünschenswert, ein Graphiksystem zu liefern, das ermöglichen würde, dass Anwendungen weniger Bits pro Pixel benötigen als gemäß der Konfiguration des auszuführenden Graphiksystems erforderlich sind, ohne die Leistungsfähigkeit der Anwendungen zu verringern.

[0013] In der WO 9202922 ist ein System zum Speichern und Verarbeiten eines Arrays von Datenelementen beschrieben, die als Mehrzahl von Seiten von Datenelementen formatiert sind. Das System umfasst einen Speicher, bei dem jede Speicherstelle eine Kapazität von 32 Bits aufweist, und eine Verarbeitungseinrichtung für Speicherlese-/-schreiboperationen. Um den Speicher beispielsweise beim Verarbeiten von Datenelementen, die entweder 16 oder 8 Bits aufweisen, vollständig zu nutzen, ist eine Mehrzahl derartiger Datenelemente in jeder Speicherstelle auf unterschiedlichen Bitpegel gespeichert, so dass an keiner Speicherstelle Datenelemente von mehr als einer Seite gespeichert sind.

ZUSAMMENFASSUNG DER ERFINDUNG

[0014] Gemäß einem Aspekt der vorliegenden Erfindung sind ein Verfahren gemäß demjenigen, das in Patentanspruch 1 beansprucht ist, und eine Vorrichtung gemäß derjenigen, die in Patentanspruch 5 beansprucht ist, vorgesehen. Bei einer derartigen Anordnung kann für Anwendungen, die weniger Bits pro Pixel verwenden als in dem Graphiksystem zur Verfügung stehen, eine maximale Graphikleistungsfähigkeit erzielt werden, indem eine volle Nutzung des Videospeicherbusses erzielt wird.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

[0015] Fig. 1 veranschaulicht ein Layout des Standes der Technik von 8Bit-Graphikpixeln in einem 32Bit-Graphikvideospeicher;

[0016] Fig. 2 veranschaulicht ein Computersystem zum Betrieb mit der Erfindung;

[0017] Fig. 3 ist ein Blockdiagramm einer Graphiksteuerung zur Verwendung bei dem Computersystem der Fig. 2; und

[0018] Fig. 4 veranschaulicht eine Zuweisung von 8Bit-Graphikpixeln in einer 32Bit-Graphikanwendung, die durch die Graphiksteuerung der Fig. 3 ausgeführt wird.

BESCHREIBUNG DES BEVORZUGTEN AUSFÜHRUNGSBEISPIELS

[0019] Unter Bezugnahme auf Fig. 2 ist ein Computersystem **20** gemäß der Erfindung gezeigt, das eine Zentralverarbeitungseinheit (CPU) **22** umfasst, die über einen Systembus **24** gekoppelt ist, um mit einem Speicher **26** zu kommunizieren. Die CPU **22** ist ferner über einen Eingangs-/Ausgangsbus (I/O-Bus) **28** gekoppelt, um mit externen Vorrichtungen wie z.B. einer Plattensteuerung **30** oder einer Graphiksteuerung **32** zu kommunizieren. Die Graphiksteuerung **32** ist gekoppelt, um einem Kathodenstrahlröhrenmonitor (CRT-Monitor, CRT = cathode ray tube) Bilddaten zu liefern.

[0020] Während des Betriebs des Computersystems **20** arbeitet die CPU **22** unter Verwendung eines Anweisungsstroms, der in dem Speicher **26** gespeichert ist, mit Anwendungen. Viele der Anwendungen laufen auf der CPU **22** und liefern Bilddaten oder Zeichnungsanforderungen, die auf der CRT **34** angezeigt werden sollen. Allgemein steuert ein Softwareprogramm, das in der Technik als Graphiktreiber bekannt ist, die Anzeige der Bilddaten oder Zeichnungsanforderungen, die durch verschiedene Anwendungen auf der CRT geliefert werden, indem es entsprechende Adress-, Daten- und Zeichnungsbefehle über den I/O-Bus **28** an die Graphiksteuerung **32** liefert. Die Befehle können Befehle, Daten von dem Speicher **26** in einen Speicher in der Graphikvorrichtung **32** zu kopieren, oder Befehle wie z.B. ein Zeichnen von Linien oder ein Punktieren von Graphikdaten, um eine Schattierung zu liefern, umfassen.

[0021] Der I/O-Bus **28** ist ein 32Bit-Bus, der dazu ausgelegt ist, durch ein definiertes Protokoll mit externen Vorrichtungen, z.B. einer Platte, Konsole usw. zu kommunizieren. Auf dem Markt steht derzeit eine Vielzahl von I/O-Bussen zur Verfügung, von denen jeder sein eigenes definiertes Protokoll aufweist. Der bei einem Ausführungsbeispiel der Erfindung verwendete I/O-Bus **28** arbeitet gemäß einem TURBOChannel™-Protokoll, und somit ist eine Schnittstelle der Graphikvorrichtung **32** gemäß dem TURBOChannel™-Protokoll entworfen. Der TUR-

BOChannel™-Bus ist ein Hochleistungsbus mit einer maximalen Bandbreite, die 100 MBytes/Sek beträgt. Es versteht sich, dass diese Erfindung durch Fachleute an eine Systemanordnung angepasst werden könnte, indem sie ein anderes I/O-Bus-Protokoll verwenden oder indem die Graphiksteuerung **32** mit dem Systembus verbunden wird.

[0022] Unter Bezugnahme auf Fig. 3 zeigt die Graphiksteuerung **32** der vorliegenden Erfindung in der Darstellung eine Steuerlogik **40** zum Decodieren der auf dem I/O-Bus empfangenen Befehle. Die Steuerlogik **40** ist gekoppelt, um Steuersignale an eine Registerlogik **42** zu liefern. Die Registerlogik **42** umfasst eine Mehrzahl von Registern zum Speichern von Informationen über den Betriebsmodus, die Länge der Abtastzeile der Anzeige und andere, ähnliche Informationen, die für die Graphiksteuerung von Nutzen sind. Eines der Register in der Registerlogik **42** ist ein Ebenenmaskenregister **47**. Das Ebenenmaskenregister **47** speichert Informationen, die angeben, welche Datenbits für jede Schreibaktion in den Speicher gelesen oder modifiziert werden sollten.

[0023] Die Registerlogik **42** liefert Informationen über Steuerdatenleitungen **43** an einen Adressgenerator **44**. Ein Datengenerator **46** ist gekoppelt, um Daten von dem I/O-Bus **28** sowie Daten von den Registern **42** zu empfangen. Der Datengenerator **46** liefert Daten an eine Neuanordnungslogik **50**, die Daten in einem in einem Quellenverschiebungsregister **52** angegebenen Umfang zyklisch verschieben kann. Die Neuanordnungslogik **50** leitet Daten über einen Bus **50a** an einen Mischpuffer **58**.

[0024] Der Mischpuffer **58** ist ein 64Bit-Puffer, der die Anzahl von Schreibvorgängen in den Videospeicher verringert, indem er mehrere aufeinander folgende Lese- oder Schreibanforderungen zu einem Speicherzugriff kombiniert, wenn dies möglich ist. Ein Mischpufferadressregister **57** speichert die aktuelle Videospeicheradresse der in dem Mischpuffer **58** gespeicherten Daten. Das Mischpufferadressregister **57** ist gekoppelt, um eine Adresse von einer Adresserzeugungslogik **44** zu empfangen.

[0025] Die Adresserzeugungslogik **44** liefert ferner ein Ausgangssignal an ein Bytefreigaberegister **61**. Das Bytefreigaberegister **61** speichert Freigaben, die angeben, welche Datenbytes auf dem Bus **50a** in einen Videospeicher **70** geschrieben werden sollten. Das Bytefreigaberegister **61** ist gekoppelt, um ein Eingangssignal an eine Schreibsteuerlogik **63** zu liefern. Die Schreibsteuerlogik **63** steuert Aktualisierungen der Daten in dem Mischpuffer **58**. Daten werden entweder dann, wenn der Mischpuffer **58** „voll" ist, wenn die Adresserzeugungslogik **44** eine Adresse liefert, die sich nicht auf die Adresse in dem Mischadressregister **57** bezieht, oder wenn die Adresserzeugungslogik im Leerlauf ist, von dem Mischpuffer **58** an einen Schreibpuffer **60** geleitet.

[0026] Der Schreibpuffer **60** weist 4 16Bit-Puffer **60a**–**60d** auf, von denen jeder mit einer entsprechenden Speichersteuerung **62**, **64**, **66** und **68** gekoppelt ist. Die Speichersteuerungen **62**–**68** steuern den Datentransfer zwischen dem Schreibpuffer 60, einem Videobus **65** und einem Videospeicher **70**.

[0027] Daten von dem Videospeicher **70** werden periodisch an ein Videoschieberegister **72** transferiert und seriell an einen Digital/Analog-Wandler (RAMDAC™) **74** hinausgeschoben. Die dem RAMDAC™ bereitgestellten Pixeldaten werden verwendet, um auf eine Farb-Nachschlagtabelle (LUT – look up table) **76** zuzugreifen, die Digital/Analog-Wandlern **77a**, **77b** und **77c** Ausgangsdaten bereitstellt. Die Form der Ausgangsdaten hängt von dem Modus ab, in dem der RAMDAC™ gerade arbeitet. Die Analog/Digital-Wandler senden jeweils drei analoge Signale, R, G und B auf Leitungen **78a**, **78b** bzw. **78c** an die CRT.

[0028] Daten in dem Videospeicher **70** werden über den Bus **65** durch die CPU gelesen (Fig. 3). Die Daten auf dem Bus **65** werden der Neuanordnungslogik **50** geliefert, die die wiedergewonnenen Bytes in der entsprechenden Reihenfolge zur Ausgabe auf den I/O-Bus **28** anordnet.

[0029] Die Graphiksteuerung **32** ist in der Lage, in einer Vielzahl von Konfigurationen, einschließlich 32Bit- und 8Bit-Graphiksystemen, zu arbeiten. Bei einem 32-Bit-Graphiksystem beträgt jedes angezeigte Pixel **32** Bits. Die 32 Bits umfassen drei Felder: ein Überlagerungsfeld, ein Steuerfeld und ein Farbdatenfeld. Das Überlagerungsfeld umfasst 4 Bits an Überlagerungsinformationen. Ein Steuerfeld umfasst 4 Bits an Steuerinformationen, die der Graphiksteuerung gegenüber angeben, wie die verbleibenden 24 Bits an Farbdaten interpretiert werden sollten (z.B. 8 Bits an Rot-Informationen, 8 Bits an Grün-Informationen, 8 Bits an Blau-Informationen). Alternativ dazu könnten Teilsätze der Bits des Farbdatenfeldes verwendet werden, um die Farb-Nachschlagtabelle in dem RAMDAC™ direkt zu adressieren.

[0030] Bei einem 8Bit-Graphiksystem sind angezeigte Pixel 8 Bits, die auf verschiedene Weisen interpretiert werden können. Beispielsweise können die 8 Bits 8 Bits von Grauskalainformationen liefern, oder die 8 Bits können verwendet werden, um Farbinformationen zu liefern, wobei 3 Bits Rot-, 2 Bits Blau-, 3 Bits Grün-Informationen liefern.

[0031] Ein Videospeicher (auch als „Rahmenpuffer" bezeichnet), der dazu konfiguriert ist, 32 Bits an Pixelinformationen zu speichern, wird als „tiefer" Rahmenpuffer bezeichnet. Dadurch, dass er 32 Bits pro Pixel an Farbinformationen aufweist, liegt eine höhe-

re Abstufung zwischen den Farben vor, die in dem Graphiksystem zur Verfügung stehen. Bei einer Graphikanwendung, die lediglich 8 Bits pro Pixel zur Farbabstufung verwendet und die auf einem Graphikteilsystem ausgeführt wird, das für 32 Bits pro Pixel konfiguriert ist, spricht man davon, dass sie „flache" Pixel verwendet.

[0032] Die internen Datenpfade der Graphiksteuerung sind 64 Bits breit. Der Datenpfad kann entweder zwei 32Bit-Pixel, vier 16Bit-Pixel oder acht 8Bit-Pixel liefern. Jedoch ändert der Videospeicher **70** die Zuweisung eventuell nicht, wenn eine Flachpixelanwendung derzeit auf dem Bildschirm läuft. Um also in der Lage zu sein, 8Bit- und 16Bit-Graphikanwendungen zu unterstützen, muss die Position der Pixel, wenn eine Anwendung in einem 32Bit-Graphiksystem ausgeführt wird, für 8Bit-Pixel-, 16Bit-Pixel- und 32Bit-Pixel-Anwendungen an derselben Position des Video-RAM **70** angeordnet sein.

[0033] Der Adressgenerator **44** und die Neuanordnungslogik **50** arbeiten, um zu gewährleisten, dass für jede Graphikanwendung die Pixeldaten in der erwarteten RAM-Vorrichtung gespeichert werden. Die Neuanordnungslogik **50** wird verwendet, um ein rasches Färben und Kopieren von „flachen" Pixeln in einem „tiefen" Rahmenpuffer zu ermöglichen, d.h. wenn durch eine Anwendung, die in einem 32Bit-Graphiksystem ausgeführt wird, nur 8 Bits an Pixelinformationen oder 16 Bits an Pixelinformationen verwendet werden.

[0034] Unter erneuter kurzer Bezugnahme auf den bekannten Videospeicher der Fig. 1 sind vier Bänke von Video-RAMs **80**–**83** gezeigt, die 4 einzelne RAM-Vorrichtungen umfassen, von denen jede 2 Bytes an Pixeldaten speichert. Somit werden 16 einzelne RAM-Vorrichtungen verwendet, um einem 64Bit-Bus Daten zu liefern.

[0035] Wenn eine 8Bit-Graphikanwendung in einem 32Bit-Graphiksystem ausgeführt wird, umfasst üblicherweise lediglich ein Byte jedes 32Bit-Pixels Pixelfarbdaten. In der Veranschaulichung umfassen 4 Speicherbänke, Bänke **80**–**83**, jeweils zwei Bytes an Pixeldaten. Angenommen, Byte 0 ist das interessierende Byte, so sind bei einem 8Bit-Graphiksystem auf Grund der Anordnung von Pixeln in dem Videospeicher während eines Speicherzugriffs lediglich Byte 0 des Pixels Null (P0.B0) und Pixel Eins (P1.B0) zugänglich. Folglich sind sechs der Bytes des Videospeicherbusses **65** ungenutzt. Um acht Pixel von Byte0-Daten zu erhalten, sind vier Speicherzugriffe auf die Bank **83** erforderlich.

[0036] Unter Bezugnahme auf Fig. 4 ist ein Videospeicher **85** gezeigt, der 4 Speicherscheiben **90**–**93** aufweist, wobei jede der Scheiben zudem 4 RAM-Vorrichtungen aufweist. Jede RAM-Vorrichtung speichert zwei Bytes an Pixeldaten, die mit P(Nr.).B(Nr.) bezeichnet sind, die die Pixelnummer und die Bytenummer in dem Pixel darstellen. Da 16 einzelne RAM-Vorrichtungen vorliegen, können einzelne Bytes jedes Pixels zweckgebundenen Abschnitten einer RAM-Vorrichtung zugewiesen werden, so dass entsprechende Bytes unterschiedlicher Pixel unterschiedlichen „Bytebahnen" des Videobusses **65** zugewiesen sind. Eine „Bytebahn" bezieht sich auf die Spalten eines Videospeichers bezüglich des Videobusses **65**, und bei dem vorliegenden System, das Pixel, die 8 Bits pro Pixel aufweisen, und einen Ausgangsbus, der 64 Gesamtbits aufweist, umfasst, liegen 8 gesonderte „Bytebahnen" in dem Videospeicher **85** vor.

[0037] Durch ein Neuanordnen der Bytepositionen in aufeinander folgenden Speicherzeilen ist die Anordnung von Pixeln in einem Speicher so organisiert, dass die Position eines gegebenen Bytes (z.B. Byte 0) in einer zugänglichen Position einer RAM-Vorrichtung für jedes Pixel (0–7) von Farbdaten vorliegt.

[0038] Somit kann auf alle vier Speicherscheiben **90**–**93** gleichzeitig zugegriffen werden, um 64 Bits von 8Bit-Pixel-Daten bei lediglich einem Zugriff des Videospeichers **70** an den Videobus **65** zu liefern, im Gegensatz zu den vier Zugriffen, die bei dem in Fig. 1 gezeigten bekannten Speichersystem erforderlich sind. Bei einer derartigen Anordnung wird der Speicherbus voll genutzt, und somit ist die Leistungsfähigkeit vieler 8Bit-Graphikanwendungsoperationen in einem 32Bit-Graphiksystem etwa vervierfacht.

[0039] Außerdem kann auf alle vier Speicherscheiben **90**–**93** gleichzeitig zugegriffen werden, um dem Videobus **65** unter Verwendung lediglich eines Zugriffs auf den Videospeicher **70** 64 Bits an 16Bit-Pixel-Daten zu liefern, im Gegensatz zu den zwei Zugriffen, die bei dem in Fig. 1 gezeigten bekannten Speichersystem nötig sind. Bei einer derartigen Anordnung ist die Leistungsfähigkeit vieler 16Bit-Graphikanwendungsoperationen in einem 32Bit-Graphiksystem etwa verdoppelt.

[0040] Die Bytes (0–3) der jeweiligen Pixel (0–7) sind so organisiert, dass eine maximale Nutzung des Videobusses **65** erreicht wird. Dies wird bewerkstelligt, indem gewährleistet wird, dass die einzelnen Pixel derart neu angeordnet werden, dass ein Byte „n" eines Paares der 32Bit-Pixel nicht in derselben Videospeicherscheibe gespeichert wird wie ein Byte „n" eines anderen Pixelpaares.

[0041] Bei dem Graphiksystem der Fig. 3 greift jede Speichersteuerung **62**–**68** auf eine Videospeicherscheibe **90**–**93** zu, um zwei Bytes pro Speicherreferenz zu liefern. Eine Anordnung, wie sie in Fig. 4 vorgesehen ist, erfüllt die obigen Entwurfskriterien, indem sie gewährleistet, dass keine Fälle vorliegen, bei

denen Byte 0 (oder Byte 1–3) jeglicher der Pixel (0–7) in derselben Bytebahn des Busses **65** angeordnet ist, und indem sie somit separate Speicherzugriffe erforderlich macht. Da jede Speichersteuerung unabhängig auf ihre Scheibe zugreifen kann und da jede Scheibe lediglich ein Pixelpaar mit Byte-n-Daten umfasst, können die Speichersteuerungen dahingehend betrieben werden, bei einem Speicherzugriff 64 Bits an Byte-n-Daten zu liefern, und somit ist eine maximale Nutzung des Busses **65** erreicht.

[0042] Die Organisation von Pixelbytes, die in Fig. 4 vorgesehen ist, ermöglicht, dass 8Bit-, 16Bit- und 32Bit-Anwendungen in einem 32Bit-Graphiksystem unterstützt werden. Beim Zugreifen auf den Speicher oder beim Wiedergewinnen von Daten aus dem Speicher müssen die Pixel und Bytes auf dem Bus **65** abhängig von der Art (d.h. der Anzahl von Bits) der Anwendung neu angeordnet werden, um zu gewährleisten, dass die richtigen Speichervorrichtungen mit Daten aktualisiert werden.

[0043] Wenn bei einem 32Bit-Graphiksystem in einem 32Bit-Graphikmodus gearbeitet wird, wird nachfolgend ein Beispiel dafür gezeigt, wie Daten auf dem Bus vorgesehen wären.

[0044] Die Bytes 0–3 des Pixels 0 und des Pixels 1 sind in der folgenden Reihenfolge auf dem Bus **65** vorgesehen:
P1.B3, P0.B3, P1.B2, P0.B2, P1.B1, P0.B1, P1.B0, P0.B0.

[0045] Auf die Bytes 0–3 des Pixels 2 und des Pixels 3 wird in der folgenden Reihenfolge zugegriffen:
P3.B1, P2.B1, P3.B0, P2.B0, P3.B3, P2.B3, P3.B2, P2.B2.

[0046] Auf die Bytes 0–3 des Pixels 4 und des Pixels 5 wird in der folgenden Reihenfolge zugegriffen:
P5.B2, P4.B2, P5.B1, P4.B1, P5.B0, P4.B0, P5.B3, P4.B3.

[0047] Auf die Bytes 0–3 des Pixels 6 und des Pixels 7 wird in der folgenden Reihenfolge zugegriffen:
P7.B0, P6.B0, P7.B3, P6.B3, P7.B2, P6.B2, P7.B1, P6.B1.

[0048] Ein Beispiel dessen, wie Daten für eine 16Bit-Anwendung, die in einem 32Bit-Graphiksystem arbeitet, auf dem Bus vorgesehen wären, wird nachfolgend gezeigt. Die Bytes 0–1 der Pixel 0 bis Pixel 3 wären in der folgenden Reihenfolge auf dem Bus **65** vorgesehen:
P3.B1, P2.B1, P3.B0, P2.B0, P1.B1, P0.B1, P1.B0, P0.B0.

[0049] Die Bytes 0–1 der Pixel 4 bis Pixel 7 wären in der folgenden Reihenfolge auf dem Bus **65** vorgesehen:
P7.B0, P6.B0, P5.B1, P4.B1, P5.B0, P4.B0, P7.B1, P6.B1.

[0050] Es ist zu beachten, dass Byte2-3-Informationen für die Pixel 0–7 auf ähnliche Weise in nur zwei Speicherzugriffen erhalten werden können, wenn mit einer 16Bit-Anwendung in einem 32Bit-Graphiksystem gearbeitet wird.

[0051] Ein Beispiel dafür, wie Daten für eine 8Bit-Anwendung, die Byte0-Informationen wiedergewinnt und in einem 32Bit-Graphiksystem arbeitet, auf dem Bus vorgesehen wären, wird nachfolgend gezeigt:
P7.B0, P6.B0, P3.B0, P2.B0, P5.B0, P4.B0, P1.B0, P0.B0.

[0052] Obwohl die Byte0-Informationen in Fig. 4 hervorgehoben sind, ermöglicht die Organisation des Videospeichers **85**, dass alle Byte3-Informationen der 8 Pixel zu einem Zeitpunkt bereitgestellt werden, oder die Byte2-Informationen oder die Bytel-Informationen. Bei einer derartigen Anordnung kann eine 8Bit-Graphikanwendung jegliche der Bytes der 32Bit-Pixel modifizieren. Das Byte, das ausgewählt wird, wird in dem Quellenbyteregister **52** und dem Zielbyteregister (dest byte register) gespeichert. Dadurch, dass sie das entsprechende Byte, das die Graphikanwendung wünscht, kennt, kann die Neuanordnungslogik den Videospeichereingang oder -ausgang um den richtigen Betrag zyklisch verschieben, um die Pixeldaten in der entsprechenden Reihenfolge zu liefern.

[0053] Dadurch, dass die Reihenfolge der Bytes und Pixel der Daten auf dem Bus **65** je nach der Anzahl der Bits der Graphikanwendung zu einer entsprechenden Reihenfolge neu angeordnet wird, wird gewährleistet, dass jeder RAM-Vorrichtung für jeden Speicherzugriff die entsprechenden Daten bereitgestellt werden.

[0054] Das Verfahren des Organisierens eines Videospeichers dahingehend, Graphikanwendungen zu unterstützen, die weniger Bits aufweisen als das konfigurierte Graphiksystem, ist nicht auf eine oder zwei Sorten von „flachen Pixeln" für einen „tiefen Rahmenpuffer" beschränkt. Vielmehr können für einen Rahmenpuffer, der dazu konfiguriert ist, eine Tiefe von 2^n Bits, in 2^m Scheiben geteilt, zu unterstützen, alle Anwendungen, die eine Pixelbreite in der Bandbreite von $2^{(n-m)}$ bis 2^n aufweisen, auf die obige Weise manipuliert werden, um eine volle Nutzung des Videobusses zu liefern.

[0055] Es gibt eine Vielzahl möglicher Neuanordnungen der Bytes und Pixel in dem Videospeicher, die die oben beschriebenen Vorteile liefern. Je nach der Frequenz gegebener Anwendungen kann es erwünscht sein, eine Bevorzugung eines Anwen-

dungstyps (8- oder 16Bit-Graphik) vor einem anderen in dem Speicherlayout zu liefern, indem die Bytes und Pixel derart angeordnet werden, dass an den wiedergewonnenen/gespeicherten Pixeln für jeden Zugriff lediglich eine einfache Zyklische-Verschiebung-Funktion durchgeführt werden muss.

[0056] Es sollte offensichtlich sein, dass diese Technik auf jede wünschenswerte Speichergröße erweitert werden kann. Jeder 32Byte-Speicherblock sollte auf dieselbe Weise wie die in Fig. 4 gezeigte Anordnung organisiert sein, wobei die Pixelnummer entsprechend ummarkiert wird. Die Neuanordnungslogik **50** verwendet mehrere der Bits der niedrigen Ordnung der Speicheradresse, um zu bestimmen, wie Daten, die den Speichersteuerungen zugehen und von denselben kommen, neu angeordnet werden sollten; höhere Adressbits werden durch die Neuanordnungslogik **50** ignoriert.

[0057] Unter erneuter Bezugnahme auf Fig. 3 arbeitet die Graphiksteuerung **32** wie folgt, wenn sie mit einer 8Bit-Anwendung in einem 32Bit-Graphikmodus arbeitet. Ankommende Pixelaktualisierungsbefehle werden auf dem I/O-Bus **28** empfangen, und Adressbits von der Adresserzeugungslogik **44** werden der Neuanordnungslogik **50** zugeführt. Da die Pixel lediglich 8 Bits relevanter Pixelfarbdateninformationen umfassen, umfasst lediglich ein Byte jedes 32Bit-Pixels Farbdaten. Je nach der Adresse des Pixels werden die Pixeldaten durch die Neuanordnungslogik **50** an der entsprechenden Position für einen 64Bit-Bus neu angeordnet.

[0058] In dem Fall, dass das gewünschte In-Eine-Reihenfolge-Bringen von Pixeln/Bytes durch bloßes zyklisches Verschieben der von dem Video-RAM wiedergewonnenen Ausgangsdaten erhalten werden kann, beruht der Umfang der zyklischen Verschiebung auf einem Teilsatz von Bits in der Adresse des 32Bit-Pixels und dem Datenbyte, das in dem Quellenbyteregister **52** ausgewählt wird. Andernfalls kann die Hardware der Neuanordnungslogik **50** dazu entworfen sein, die gewünschte Organisation von Pixel- und Bytedaten zu berücksichtigen.

[0059] Die Adresse der aktuellen Operation ist in dem Mischadressregister **57** gespeichert. Während die ankommenden Daten auf dem Bus **50a** bereitgestellt werden, liefert das Bytefreigaberegister **61** Informationen, die angeben, welche Datenbytes auf dem Bus **50a** gültig sind und aus dem Speicher geschrieben/gelesen werden sollten. Jede Bytefreigabe entspricht einem der Bytes des 64Bit-Busses. Das Bytefreigaberegister wird der Neuanordnungslogik **50** bereitgestellt.

[0060] Zusätzlich zu Daten von dem Bytefreigaberegister **61** werden der Neuanordnungslogik **50** auch Daten von dem Ebenenmaskenregister **47** bereitgestellt. Das Ebenenmaskenregister **47** speichert eine Ebenenmaske, die bestimmt, welche Bits jedes angezeigten Pixels durch eine Graphikanwendung modifiziert werden können. Somit ist die durch die Neuanordnungslogik **50** bereitgestellte und in einem Mischpuffermaskenregister **55** gespeicherte Bytefreigabe eine Kombination des Inhalts des Bytefreigaberegisters **61**, des Ebenenmaskenregisters **47**, des Betriebsmodus und der Betriebsart.

[0061] Wenn die Adresse der Daten auf dem Bus **50a** der Vierfachwortadresse in dem Mischpufferadressregister **57** entspricht und die Mischpufferbytemaske **55** angibt, dass die Mischpufferposition frei ist, werden die Daten auf dem Bus **50a** mit den Daten in dem Mischpuffer **58** gemischt.

[0062] Wenn der Mischpuffer **58** „voll" ist, d.h. wenn entweder kein Raum in dem Mischpuffer ist, wenn die ankommende Adresse nicht der in dem Mischadressregister **57** gespeicherten Vierfachwortadresse entspricht, oder wenn sich die Adresserzeugungslogik **44** im Leerlauf befindet, werden die Daten von dem Mischpuffer **58** zu dem Schreibpuffer **60** verschoben. Wenn zumindest eine der Bytefreigaben, die den 16 Bits entsprechen, die in einem Schreibteilpuffer gespeichert werden sollen, eingestellt wird, werden sowohl die 16Bit-Daten als auch die Bytefreigabebits in dem entsprechenden Teilpuffer gespeichert. Folglich speichert jeder Teilpuffer **60a**–**60d** 16 Datenbits aus dem Mischpuffer und 2 Bits einer Bytefreigabe, und jeder Teilpuffer **60a**–**60d** wird durch eine der Speichersteuerungen **62**, **64**, **66** bzw. **68** gesteuert.

[0063] Jedoch empfangen lediglich Teilpuffer, bei denen zumindest eines ihrer entsprechenden Bytefreigabebits eingestellt ist, Daten von dem Mischpuffer. Wenn beispielsweise die in dem Mischpufferbytemaskenregister **55** gespeicherten Daten **00** 00 01 00 betrügen, würde lediglich der Teilpuffer **60c** mit Daten von dem Bus **58a** und Bytefreigabebits 01 beladen. Nachdem der Teilpuffer **60c** mit Daten beladen ist, würde die Speichersteuerung **66** die den geladenen Daten entsprechenden Bytefreigabebits verwenden, um die freigegebenen Datenbytes aus dem Teilpuffer **60c** an die entsprechende Stelle in dem Videospeicher zu schreiben.

[0064] Da während des Speicherzugriffs lediglich die Speichersteuerung **66** verwendet wird, sind die verbleibenden Speichersteuerungen dafür frei, andere Pixel an ankommenden Daten über den Bus **58a** in ihre Teilpuffer aufzunehmen.

[0065] Indem man ermöglicht, dass die Speichersteuerungen völlig unabhängig arbeiten, ist die Leistungsfähigkeit im Vergleich zu bekannten Konfigurationen, bei denen Speichersteuerungen synchron agieren, enorm verbessert. Bei einem System synchroner Speichersteuerungen ist es denkbar, dass 4

Speichersteuerungen verwendet werden, um lediglich ein Datenbyte in den Speicher zu schreiben, wobei sich 3 Speichersteuerungen im Leerlauf befinden und einen Abschluss des Schreibvorgangs durch die andere Speichersteuerung abwarten. Bei Systemen synchroner Speichersteuerungen ist die Leistungsfähigkeit des Systems durch die Anzahl physischer Bytes pro Pixel eingeschränkt.

[0066] Wenn die Speichersteuerungen jedoch dazu ausgelegt sind, absolut unabhängig zu arbeiten, sind während des Speichersteuerungszugriffszeitraums lediglich diejenigen Steuerungen belegt, die tatsächlich auf den Videospeicher zugreifen. Demgemäß ist die Leistungsfähigkeit eines derartigen Graphiksystems lediglich durch die Anzahl von Pixeln eingeschränkt, die tatsächlich geschrieben werden, nicht durch die eigentliche Anzahl physischer Bytes pro Pixel.

[0067] Unter erneuter Bezugnahme auf Fig. 4 sei beispielsweise angenommen, dass eine Graphikanwendung, die 24 Bits pro Pixel verwendet, auf einem 32Bit-Graphiksystem ausgeführt wird und dass die Speichersteuerungen **62**, **64**, **66** und **68** mit Byte0-, Byte1-, Byte2- bzw. Byte3-Daten arbeiten. Unter Verwendung eines Systems synchroner Speichersteuerungen würden, um Bytes 0–2 jedes Pixels zu schreiben, Daten aus dem Mischpuffer **60** und Bytefreigaben aus dem Bytefreigaberegister **61** zu den jeweiligen Teilpuffern **60a**–**60d** verschoben. Die Speichersteuerung **68** würde eine Bytefreigabe von 00 von dem Teilpuffer **60d** empfangen und sich während des Speicherzugriffs durch die anderen Speichersteuerungen **62**, **64** und **66** im Leerlauf befinden. Die Speichersteuerung **68** ist nicht frei dafür, weitere Pixeldaten anzunehmen, bis die anderen Steuerungen das Aktualisieren der 3 Bytes an Pixeldaten abgeschlossen haben. Um also acht 24Bit-Pixel zu schreiben, müssen vier Speicherzugriffe durchgeführt werden.

[0068] Das Erfordernis von 4 Speicherreferenzen ist unter Bezugnahme auf Fig. 4 ersichtlich. Bei dem ersten Speicherzugriffszyklus würde auf die Bytes 0–2 des Pixels 0 und des Pixels 1 zugegriffen. Somit würden bei dem ersten Speicherzugriff keine Aktualisierungen der Speicherscheibe **90** vorliegen. Bei dem zweiten Speicherzugriffszyklus würde auf die Bytes 0–2 des Pixels 2 und des Pixels **3** zugegriffen. Es würden keine Aktualisierungen der Speicherscheibe **92** erfolgen. Bei dem dritten Speicherzugriffszyklus würde auf die Bytes 0–2 des Pixels 4 und des Pixels 5 zugegriffen. Es würden keine Aktualisierungen der Speicherscheibe **93** erfolgen. Bei dem vierten Speicherzugriffszyklus würde auf die Bytes 0–2 des Pixels 6 und des Pixels 7 zugegriffen. Es würden keine Aktualisierungen der Speicherscheibe **91** erfolgen.

[0069] Bei dem bevorzugten Ausführungsbeispiel jedoch, bei dem alle Speichersteuerungen unabhängig agieren, muss zu keiner Zeit eine der Speichersteuerungen im Leerlauf bleiben. Dies ist darauf zurückzuführen, dass, wenn die Bytefreigabe für die gegebene Speichersteuerung gleich 00 ist, die Speichersteuerung keinen Eintrag für die gegebenen Daten empfängt, sondern frei dafür ist, von dem Mischpuffer **58** neue Daten in ihren Schreibpuffer aufzunehmen. Folglich kann ein Schreiben von acht 32Bit-Pixeln, wo in der Tat lediglich 24 Bits jedes 32Bit-Pixels modifiziert werden, in einem Äquivalent zu 3 Speicherzugriffen bewerkstelligt werden.

[0070] Unter erneuter Bezugnahme auf Fig. 4 würde beispielsweise während des ersten Speicherzugriffs auf die Bytes 0–2 des Pixels 0 und des Pixels 1 zugegriffen. Außerdem würde auf das Byte 1 des Pixels 2 und des Pixels 3 zugegriffen. Bei dem zweiten Speicherzugriff würde auf das Byte 0 und Byte 2 des Pixels 2 und des Pixels 3 zugegriffen, und es würde auf das Byte 0 und das Byte 2 des Pixels 4 und des Pixels 5 zugegriffen. Bei dem dritten Speicherzugriff würde auf das Byte 1 des Pixels 4 und des Pixels 5 zugegriffen, und es würde auf die Bytes 0–2 des Pixels 6 und des Pixels 7 zugegriffen.

[0071] Ein Zugriff auf weniger als alle vier Bytes jedes 32Bit-Pixels ist ziemlich üblich. Anwendungen, die 24 Bits an Farbinformationen verwenden, modifizieren selten die Steuerbits, und somit lesen oder schreiben die meisten Operationen lediglich drei Bits jedes 32Bit-Pixels. Manche RAM-DACs™ ermöglichen, dass die 24 Bits an Farbinformationen in zwei 12Bit-Puffer aufgeteilt werden; die Steuerbits bestimmen, ob die 12 oberen oder unteren Bits angezeigt werden sollten. Die beste Leistungsfähigkeit würde erhalten, wenn diese zwei 12Bit-Felder so zugewiesen werden könnten, dass auf sie als zwei separate 16Bits-Pixel zugegriffen werden könnte, jedoch wird diese Anordnung durch manche RAMDACs™ von vornherein ausgeschlossen. Trotzdem gilt, dass, wenn der RAMDAC ermöglicht, dass ein 12Bit-Feld den Bits 0–11 zugewiesen wird und das andere 12Bit-Feld den Bits 12–23 zugewiesen wird, ein Zugriff auf eines der 12Bit-Felder einen Zugriff auf lediglich zwei Bytes jedes 32Bit-Pixels erfordert. Somit kann ein Zugriff von 8 Pixeln an 12Bit-Informationen mit lediglich zwei Speicherzugriffen bewerkstelligt werden, wie unter Bezugnahme auf die oben beschriebenen 16Bit-Anwendungen beschrieben wurde.

[0072] Dreidimensionale Anwendungen verwenden Z Puffer und Schablonenpuffer. Bei der in Fig. 3 gezeigten Graphiksteuerung umfasst jedes 32Bit-Z-/-Schablonenpufferpixel zwei Felder: ein 8Bit-Schablonenfeld und ein 24Bit-Z-Wert-Feld. Die Mehrzahl an dreidimensionalen Operationen lesen und schreiben lediglich das Z-Wert-Feld und nicht das Schablonenfeld, um lediglich mit drei Bytes jedes

32Bit-Pixels zu arbeiten. Demgemäß kann ein Zugriff von 8 Pixeln an Z-Informationen in 3 Speicherzugriffszyklen bewerkstelligt werden, wie oben unter Bezugnahme auf Fig. 4 beschrieben wurde.

[0073] Das beschriebene Graphiksystem nutzt die Vorteile einer atypischen Anordnung von Pixeldaten in einem Videospeicher, um die Nutzung des Videobusses zu erhöhen und somit die Gesamtleistungsfähigkeit des Graphiksystems zu erhöhen. Überdies kann die Leistungsfähigkeit der Graphik für manche Anwendungen weiter verbessert werden, indem eine völlige Unabhängigkeit der Speichersteuerungen, die die jeweiligen Videospeicherscheiben steuern, ermöglicht wird. Nachdem ein bevorzugtes Ausführungsbeispiel der Erfindung beschrieben wurde, wird Fachleuten nun einleuchten, dass andere Ausführungsbeispiele, die deren Konzepte beinhalten, verwendet werden können, die in den Schutzumfang der beigefügten Patentansprüche fallen.

Patentansprüche

1. Ein Verfahren zum Verbessern des Verhaltens einer Graphikanwendung, die auf einem Graphikteilsystem (**32**) ausgeführt wird, das einen Videospeicher (**85**) aufweist, der in eine Mehrzahl von Scheiben (**90**–**93**) zum Speichern von Graphikdaten aufgeteilt ist, wobei das Graphikteilsystem konfiguriert ist, um Anwendungen zu unterstützen, die eine erste Anzahl von Bits pro Pixel aufweisen, und wobei die Graphikanwendung unter Verwendung einer Mehrzahl zweiter Anzahlen von Bits pro Pixel ausgeführt wird, wobei die zweiten Anzahlen von Bits pro Pixel kleiner als die erste Anzahl von Bits pro Pixel sind, wobei das Verfahren folgende Schritte umfasst:
Speichern einer Mehrzahl von Pixeldaten (P1–P7) in dem Videospeicher, derart, dass entsprechende Bytes (B0–B3) von Graphikdaten verschiedener Pixel in unterschiedlichen, gleichzeitig zugänglichen Scheiben (**90**–**93**) des Videospeichers gespeichert werden, und derart, dass benachbarte Bytes von Graphikdaten derselben Pixel in unterschiedlichen Scheiben des Videospeichers gespeichert werden; und
Kombinieren, in einem Mischpuffer (**58**), mehrerer aufeinander folgender Lese- oder Schreibanforderungen in einzelne Speicherzugriffe, um die Anzahl von Zugriffen auf den Videospeicher zu verringern.

2. Das Verfahren gemäß Anspruch 1, bei dem der Speicherschritt folgende Schritte umfasst:
Anordnen des Videospeichers, um einen ersten Abschnitt (B0) jedes der Mehrzahl von Pixeln (P1–P7) und einen zweiten Abschnitt (B1) jedes der Mehrzahl von Pixeln (P1–P7) zu speichern, wobei jede der Scheiben (**90**–**93**) eine vorbestimmte Anzahl von Bytes von Pixeldaten speichert;
Anordnen der ersten Abschnitte (B0) und der zweiten Abschnitte (B1) in jeweilige Gruppen von Pixeldaten, wobei jede der Gruppen von Pixeldaten die vorbestimmte Anzahl von Datenbytes umfasst und wobei jede der Gruppen von Pixeldaten entsprechende Datenbytes von verschiedenen Pixeln umfasst; und
Zuweisen jeder der Gruppen von Pixeldaten zu der Mehrzahl von Scheiben, derart, dass jeder Abschnitt von Pixeldaten in einer anderen der Scheiben gespeichert ist.

3. Das Verfahren gemäß Anspruch 2, das ferner folgenden Schritt umfasst:
Zuweisen jeder der Gruppen von Pixeldaten zu den Scheiben (**90**–**93**), derart, dass auf einen Lesezugriff auf den Videospeicher (**85**) hin die Datenabschnitte von den verschiedenen Pixeln in einer Reihenfolge einem Videobus (**65**) bereitgestellt werden.

4. Das Verfahren gemäß Anspruch 2, das ferner folgenden Schritt umfasst:
erneutes Anordnen der Gruppen von Pixeldaten derart, dass auf einen Schreibzugriff auf den Videospeicher (**85**) hin Datenabschnitte von den verschiedenen Pixeln in einer Reihenfolge dem Videospeicher bereitgestellt werden.

5. Eine Vorrichtung zum Verbessern des Verhaltens einer Graphikanwendung, die auf einem Graphikteilsystem (**32**) ausgeführt wird, wobei das Graphikteilsystem konfiguriert ist, um Anwendungen zu unterstützen, die eine erste Anzahl von Bits pro Pixel aufweisen, und wobei die Graphikanwendung unter Verwendung einer Mehrzahl zweiter Anzahlen von Bits pro Pixel ausgeführt wird, wobei die zweiten Anzahlen von Bits pro Pixel kleiner als die erste Anzahl von Bits pro Pixel sind, wobei die Vorrichtung besonders angepasst ist, um das Verfahren gemäß Anspruch 1 zu implementieren, und folgende Merkmale aufweist:
einen Graphikprozessor (**32**);
einen mit dem Graphikprozessor gekoppelten Videobus (**65**);
eine Einrichtung (**46**) zum Bereitstellen von Pixeldaten auf dem Videobus;
einen Videospeicher (**85**), der in eine Mehrzahl von Scheiben (**90**–**93**) zum Speichern einer Mehrzahl von Pixeldaten aufgeteilt ist;
einen Mischpuffer (**58**), der mit dem Graphikprozessor gekoppelt ist, um mehrere aufeinander folgende Lese- oder Schreibzugriffe auf den Videospeicher zu einem einzelnen Zugriff zu kombinieren;
eine Mehrzahl von Speichersteuerungen (**62**, **64**, **66**, **68**), die mit dem Mischpuffer gekoppelt sind, wobei jede derselben einen zweckgebundenen Puffer zum Speichern eines Abschnitts der Pixeldaten umfasst, wobei die Mehrzahl von Speichersteuerungen jeweils unabhängig voneinander ein Speichern von Daten in einer entsprechenden der Mehrzahl von Scheiben des Videospeichers steuert.

6. Die Vorrichtung gemäß Anspruch 5, die ferner

folgende Merkmale aufweist:
eine Einrichtung (**57**) zum Speichern der aktuellen Videospeicheradresse der in dem Mischpuffer (**58**) gespeicherten Daten; und
eine Einrichtung (**63**) zum Steuern von Aktualisierungen der Daten in dem Mischpuffer.

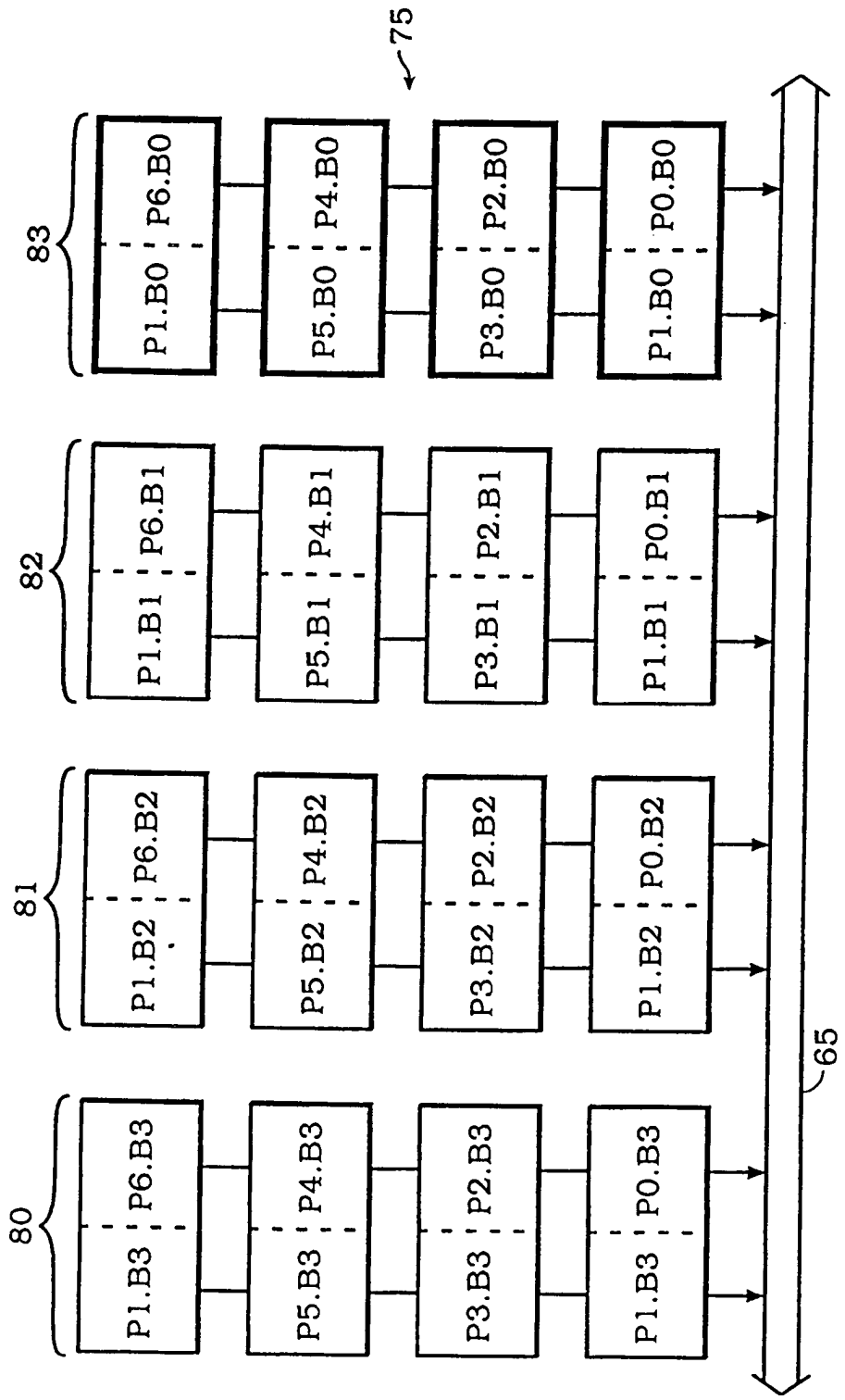
7. Die Vorrichtung gemäß Anspruch 6, die ferner folgendes Merkmal aufweist:
ein mit der Einrichtung (**46**) zum Bereitstellen von Pixeldaten gekoppeltes Bytemaskenregister (**47**), wobei das Bytemaskenregister, das eine Mehrzahl von Bytefreigaben, die der Anzahl von Pixeldatenbytes auf dem Videobus (**65**) entsprechen, wobei die Mehrzahl von Bytefreigaben in eine Mehrzahl von Gruppen von Bytefreigaben ansprechend auf die Anzahl von Scheiben des Videospeichers (**90**–**93**) aufgeteilt ist, wobei jede der Gruppen von Bytefreigaben in einem entsprechenden der zweckgebundenen Puffer gespeichert ist, um eine zugeordnete Speichersteuerung (**62**, **64**, **66**, **68**) freizugeben, um mit Daten in dem entsprechenden Puffer zu arbeiten.

8. Die Vorrichtung gemäß Anspruch 7, die ferner folgendes Merkmal aufweist:
eine Einrichtung, die auf das Bytemaskenregister (**47**) anspricht, zum selektiven Speichern von Abschnitten der Pixeldaten in den zweckgebundenen Puffern.

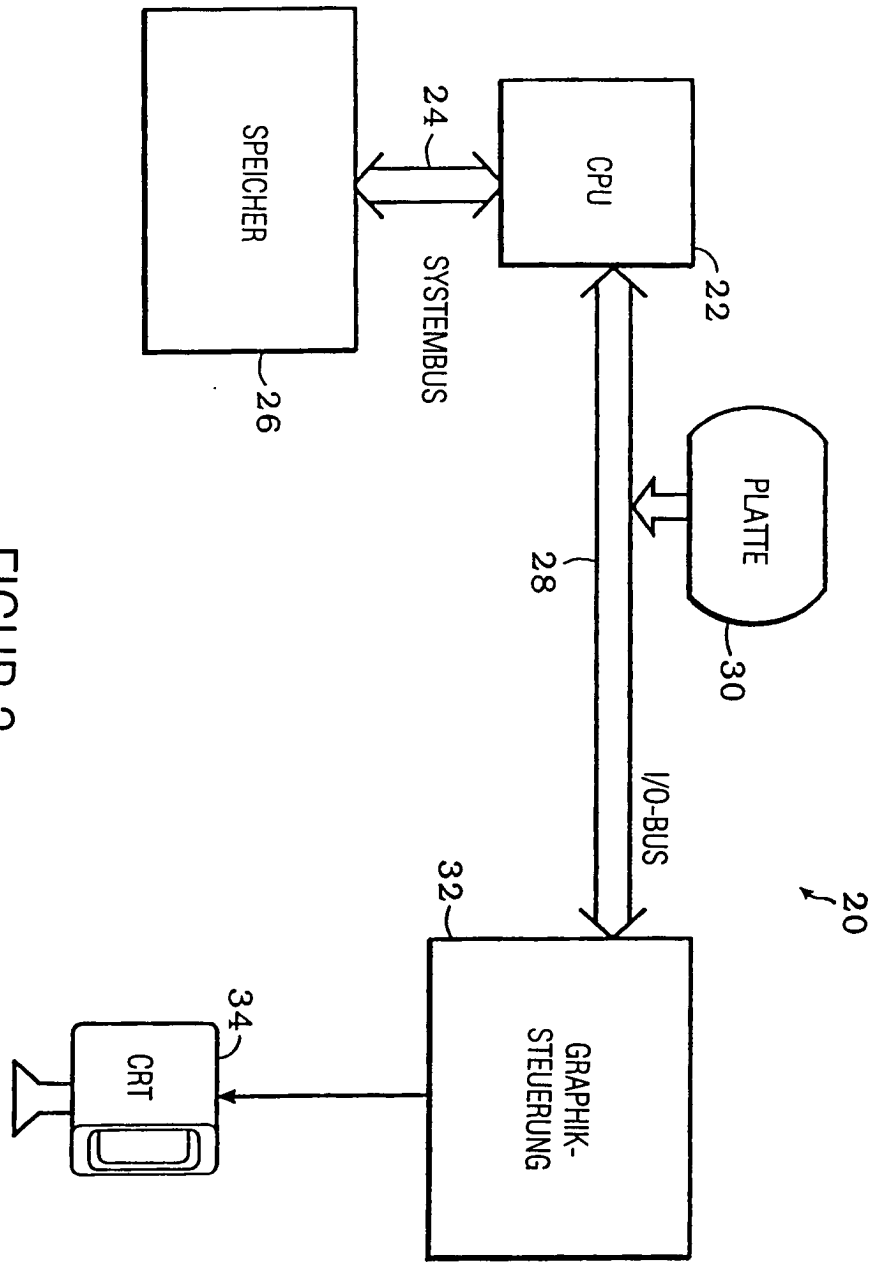
9. Die Vorrichtung gemäß Anspruch 8, die ferner folgendes Merkmal aufweist:
eine mit der Einrichtung (**46**) zum Bereitstellen der Pixeldaten gekoppelte Einrichtung (**50**) zum erneuten Anordnen einzelner Pixel auf dem Videobus (**65**) derart, dass die Pixeldaten derart in dem Videospeicher (**85**) gespeichert sind, dass entsprechende Bytes der Pixeldaten an gleichzeitig zugänglichen Stellen des Videospeichers gespeichert sind.

Es folgen 4 Blatt Zeichnungen

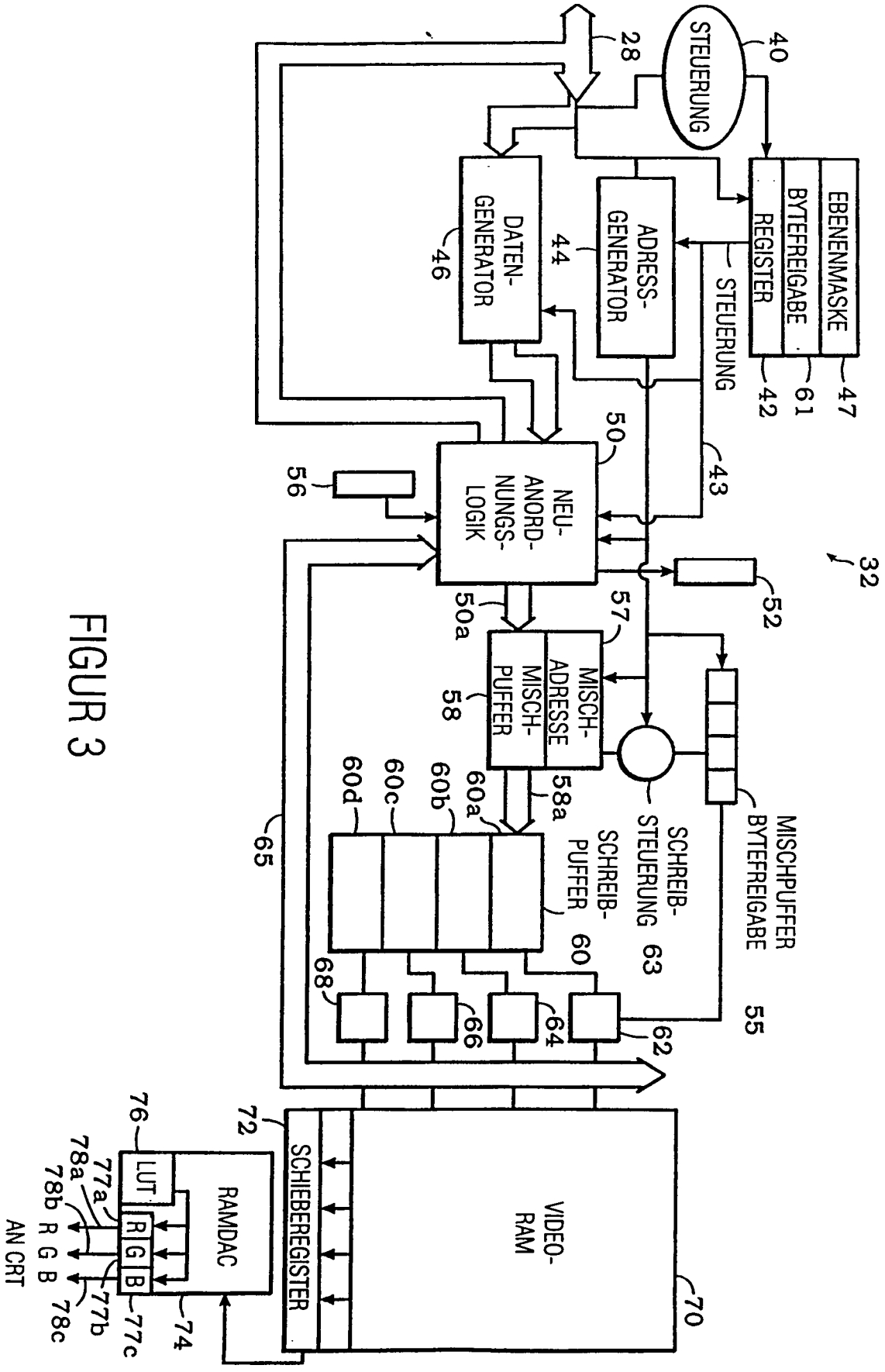
Anhängende Zeichnungen



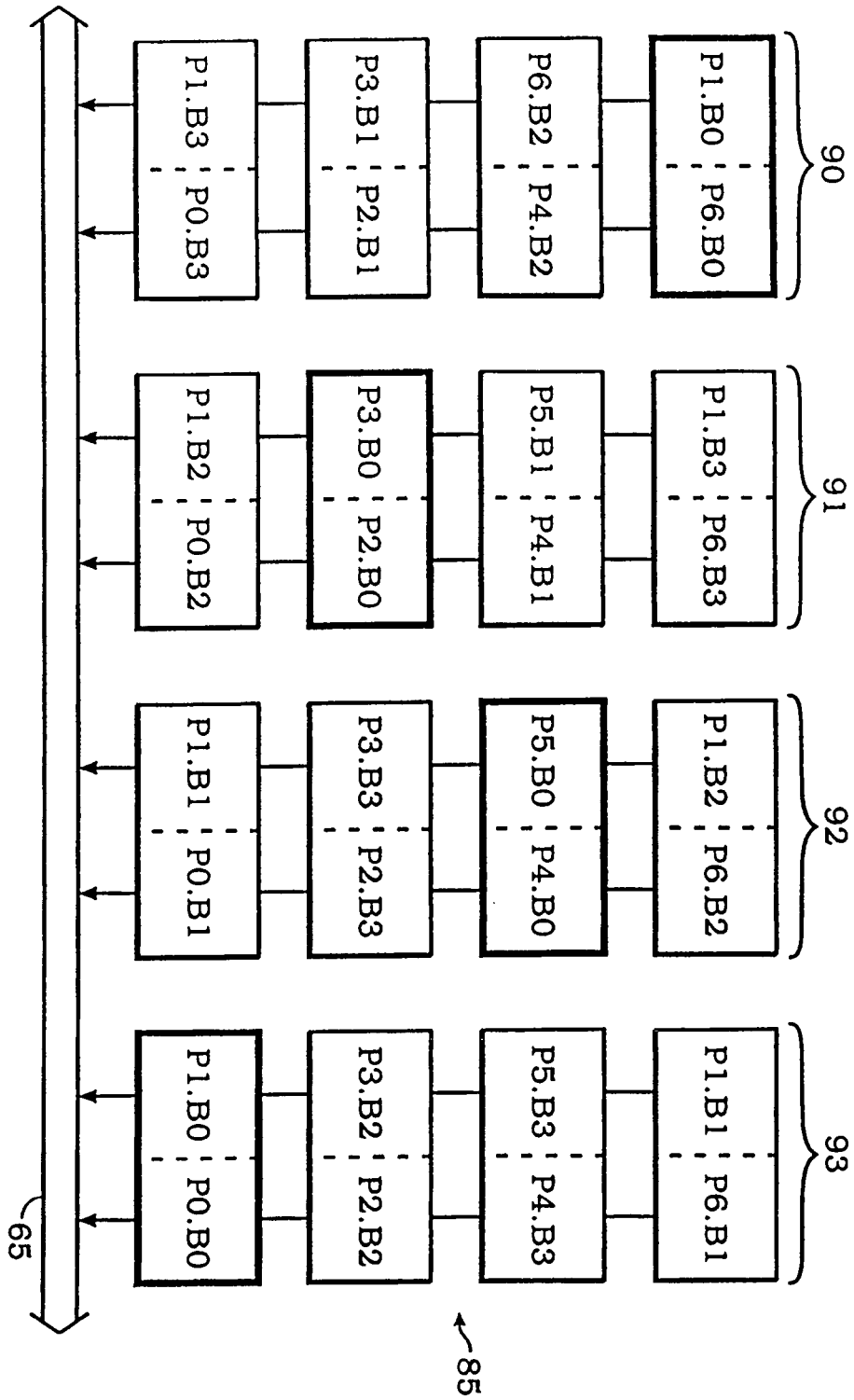
FIGUR 1
(STAND DER TECHNIK)


SPEICHER
24
CPU
SYSTEMBUS
22
26
PLATTE
28
30
I/O-BUS
20
32
GRAPHIK-
STEUERUNG
34
CRT

FIGUR 2


STEUERUNG
40
28
EBENENMASKE
BYTEFREIGABE
REGISTER
47
61
42
DATEN-GENERATOR
46
ADRESS-GENERATOR
44
STEUERUNG
43
50
NEU-ANORD-NUNGS-LOGIK
56
32
52
57
50a
MISCH-ADRESSE
MISCH-PUFFER
58
58a
60a
60b
60c
60d
65
SCHREIB-PUFFER
60
SCHREIB-STEUERUNG
63
MISCHPUFFER BYTEFREIGABE
55
68
66
64
62
72
SCHIEBEREGISTER
VIDEO-RAM
70
76
LUT
R
G
B
RAMDAC
74
77a
77b
77c
78a
78b
78c
R G B
AN CRT

FIGUR 3


P1.B0
P6.B0
P6.B2
P4.B2
P3.B1
P2.B1
P1.B3
P0.B3
90
P1.B3
P6.B3
P5.B1
P4.B1
P3.B0
P2.B0
P1.B2
P0.B2
91
P1.B2
P6.B2
P5.B0
P4.B0
P3.B3
P2.B3
P1.B1
P0.B1
92
P1.B1
P6.B1
P5.B3
P4.B3
P3.B2
P2.B2
P1.B0
P0.B0
93
65
85

FIGUR 4