



(19)
Bundesrepublik Deutschland
Deutsches Patent- und Markenamt

(10) **DE 699 20 582 T2** 2005.09.08

(12)

Übersetzung der europäischen Patentschrift

(97) **EP 0 967 543 B1**

(21) Deutsches Aktenzeichen: **699 20 582.4**

(96) Europäisches Aktenzeichen: **99 202 084.2**

(96) Europäischer Anmeldetag: **25.06.1999**

(97) Erstveröffentlichung durch das EPA: **29.12.1999**

(97) Veröffentlichungstag

der Patenterteilung beim EPA: **29.09.2004**

(47) Veröffentlichungstag im Patentblatt: **08.09.2005**

(51) Int Cl.⁷: **G06F 7/52**
G06F 7/544

(30) Unionspriorität:

90589 P 25.06.1998 US

(73) Patentinhaber:

Texas Instruments Inc., Dallas, Tex., US

(74) Vertreter:

Prinz und Partner GbR, 81241 München

(84) Benannte Vertragsstaaten:

DE, FR, GB, IT, NL

(72) Erfinder:

**Henderson, Alva, Sherman, Texas 75092, US;
Cavaliere, Francesco, Richardson, Texas 75082,
US**

(54) Bezeichnung: **Verfahren zum Multiplizieren von Zahlen in Mehrwortkettendarstellung**

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] Die vorliegende Erfindung betrifft das Gebiet der elektronischen Datenverarbeitung und insbesondere einen Mikroprozessor mit einem integrierten Multipliziererblock, der für eine "Ketten-" oder Mehrwort-Multiplikation geeignet ist.

HINTERGRUND

[0002] Mikroprozessoren sind oftmals erforderlich, um binäre Daten zu verarbeiten, die weite Bereiche von Bitlängen haben, beispielsweise Daten, die von einem einzigen logischen Bit bis zu hochpräzisen Arithmetikoperationen, die mit Daten einhergehen, deren Länge mehr als 128 Bits aufweisen kann, reichen.

[0003] Hardware-Arithmetik-Logik-Einheiten (ALUs) in Mikroprozessoren sind im Allgemeinen dafür entwickelt und ausgelegt, dass sie mit festen Operandenlängen umgehen. Dadurch erfordern hochpräzise Arithmetikoperationen mehrere Programmschritte und mehrere Prozessorzyklen. Diese Datenverarbeitungsbedingungen führen zu Programmen, die hinsichtlich der Ausführungszeit unzulänglich sind, da die Mikroprozessor-Hardware und der unterstützende Pogrammbehlssatz nicht für die Verarbeitung von Daten, die einen weiten Bereich von Operandenlängen aufweisen, optimiert sind.

[0004] Diese Unzulänglichkeit ergibt sich größtenteils sowohl aus wiederholten Speicher- und Ladevorgängen als auch aus einem Softwareschleifen-Steueraufwand (Vergleiche, Verzweigungen usw.). Für komplizierte Operationen, wie etwa die Multiplikation, und für Operationen, die mit Algorithmen erweiterter Genauigkeit einhergehen, ist der Aufwand noch ausgeprägter. Außerdem müssen für Mehrwort-Berechnungen das Vorzeichen (negativ oder positiv) und der Nullzustand eines Arithmetikergebnisses gesondert behandelt werden, was noch mehr Prozessorzeit erfordert, um die Operation durchzuführen.

[0005] Digitale Signalprozessoren (DSPs) sind schnelle Mikroprozessoren, die dafür optimiert sind, eine große Anzahl von Arithmetikoperationen in einem kurzen Zeitraum auszuführen. Wie der Name andeutet, wurden DSPs entwickelt, um Echtzeitverarbeitungsalgorithmen (z. B. Filterung, Komprimierung, Verschlüsselung) an digitalen Signalen auszuführen. DSPs enthalten hinsichtlich der Leistungsfähigkeit optimierte Arithmetikstrukturen, die bei herkömmlichen Mikroprozessoren nicht zu finden sind. Von den Einrichtungen, die in DSPs integriert sind, um die Leistungsfähigkeit hinsichtlich der Verarbeitung von Zahlen zu verbessern, sind die zwei wich-

tigsten der Hardware-Multipliziererblock und das zyklische Ausgangsschieberegister, der Barrelshifter.

[0006] Wie die meisten Strukturen in einem DSP sind die Hardware-Multipliziererböcke hochleistungsfähige, geschwindigkeitsoptimierte Strukturen. Leider benötigen DSP-Hardware-Multipliziererböcke große Flächenanteile der Chipoberfläche für spezialisierte Signalverarbeitungsschaltungsanordnungen. Außerdem erfordert eine Größenzunahme des Multipliziererböcks im Allgemeinen damit einhergehend eine Größenzunahme eines CPU-Datenbusses und der Arithmetik-Logik-Einheiten (ALUs). Beispielsweise ergibt die Multiplikation von zwei 16-Bit-Zahlen ein 32-Bit-Ergebnis. Ein Multipliziererblock, der in der Lage ist, diese 16-Bit-Multiplikation auszuführen, würde im Allgemeinen 32-Bit-Datenbusse, – Arithmetik-Logik-Einheiten (ALUs) und Akkumulatorregister, die das Ergebnis aufnehmen, erfordern. Dies erhöht die Komplexität und folglich die Kosten eines Mikroprozessors um einiges. Außerdem ist es wünschenswert, besonders bei Anwendungen der Signalverarbeitung, Zahlen mit einer Länge von mehr als einem Wort multiplizieren zu können. Leider nehmen die Komplexität und die Größe des Multipliziererböcks schnell und schwer steuerbar zu, wenn die Größe seiner Operanden zunimmt.

[0007] Herkömmliche Mikroprozessoren enthalten oftmals keinen Hardware-Multiplikationsblock oder Barrelshifter, da die Größe und die Kosten dies vor allem im preisgünstigen Käufermarkt verhindern. In einem herkömmlichen einfachen Prozessor werden Zahlen durch wiederholtes Addieren in einem Mehrbitaddierer multipliziert, was eine verhältnismäßig langsame Verarbeitung darstellt. Eine typische Multiplikation, die in der Software ausgeführt wird, erfordert eine lange Folge von Verschiebungen und Additionen, die einen großen Anteil der Prozessorzeit erfordern. Vom Standpunkt der Geschwindigkeit aus wäre höchst wünschenswert, über die Leistungsfähigkeit und Funktionalität eines in einen preiswerten Mikroprozessor integrierten Hardware-Multipliziererböcks zu verfügen, jedoch verbieten normalerweise die Komplexität und der Platzbedarf der zugehörigen Hardware eine derartige Ausführung aus Kostengründen.

[0008] Ein Barrelshifter ist eine weitere geschwindigkeitsoptimierte DSP-Struktur, die für umfangreiche Berechnungen äußerst nützlich ist. Barrelshifter sind so beschaffen, dass sie eine Zahl während einer einzigen Operation um mehrere Bitpositionen verschieben. Obwohl ein Barrelshifter in Funktion und Aufbau einem Multipliziererblock ähnlich ist, ist er herkömmlich als eine völlig separate Struktur auf dem Chip verwirklicht. In der Binärarithmetik kommt eine Linksverschiebung um eine Bitposition einer Multiplikation mit zwei gleich. Eine Linksverschiebung um N Bits ist einer Multiplikation mit 2^N äquiva-

lent.

[0009] Herkömmliche einfache Prozessoren enthalten keine Barrelshifter, sondern führen eine zyklische Schiebeoperation durch eine Reihe von Verschiebungen um eine Position aus, wobei jede Verschiebung eine einzige Softwareoperation umfasst. Eine Barrelshift-Funktion ist in einem Prozessor höchst wünschenswert, jedoch nimmt die spezialisierte Schaltungsanordnung, die erforderlich ist, um diese Funktion auszuführen, zu viel Platz auf einem Chip ein, um in preiswerte Prozessoren integriert zu werden. Die Barrelshift-Funktion könnte von einem Multipliziererblock ausgeführt werden, wenn die Zahlen besonders codiert sind; wenn jedoch die Codierung softwaremäßig ausgeführt werden muss, vermindert der Aufwand deutlich jeden Zugewinn an Leistungsfähigkeit durch einen Hardware-Barrelshift.

[0010] Es ist bekannt, dass frühere Mikroprozessoren fähig waren, Ketten zu verarbeiten, beispielsweise indem ein bestimmter Befehl eine vorgeschriebene Anzahl von Malen wiederholt wurde. Es ist bekannt, dass eine wiederholte Addition mit Übertrag eine Kettenoperation ausführen wird, wobei die Datenspeicheradressen der Operanden und des Ergebnisses nach jeder Operation automatisch inkrementiert werden. Außerdem ist bekannt, dass andere unveränderliche Hardware-Multiplizierer verwendet haben, um Multiplikationen mit erweiterter Genauigkeit unter Verwendung eines Multipliziere-mit-Teilen-Algorithmus vorzunehmen, was eine komplizierte und relativ leistungsschwache Lösung darstellt.

[0011] Wie erwähnt worden ist, müssen Mikroprozessoren typisch mit Operanden mit verschiedenen, mitunter sehr verschiedenen Längen umgehen. Die Operandenlängen können zwischen einem einzigen logischen Bit und bis zu 512 Bits oder mehr variieren. Andererseits haben Arithmetik-Logik-Einheiten (ALUs) eine feste Breite. Wenn eine hochpräzise Arithmetik notwendig ist, die Operanden erfordert, die länger als die ALU sind, muss der Prozessor die Operation in mehreren Schritten ausführen. Die Programme werden hinsichtlich der Ausführungsdauer und des Programmierungscodes ineffizient, da die zu Grunde liegende Hardware und der unterstützende Befehlssatz nicht für die Verarbeitung von hochpräzisen Daten, die durch eine sequentielle Verkettung von Datenwörtern repräsentiert werden, optimiert sind.

[0012] Wenn eine Zahl, die eine Länge von einem Wort aufweist, mit einer Zahl, deren Länge größer als ein Wort ist, multipliziert werden muss (Kettenmultiplikation), kann das Verfahren selbst in Ausführungen, die einen Multipliziererblock enthalten, sehr schwerfällig werden. Eine derartige Multiplikation wird durch eine Reihe von Einzelwort-Teilmultiplikationen ausgeführt, wobei mit dem niedrigstwertigen Wort be-

gonnen wird. Für jede Teilmultiplikation muss der Code den Prozessor anweisen, (1) die Einzelwort-Multiplikation (entweder hardware- oder softwaremäßig) auszuführen, (2) das niedrigstwertige Wort des Ergebnisses im Speicher zu speichern, (3) das höchstwertige Wort des Ergebnisses zu dem richtigen Operandenregister für die nächste Teilmultiplikation zu befördern, (4) zu prüfen, ob die Operandenkette vollständig ist, und wenn nicht, (5) an den Anfang der Schleife zurückzugehen, um das Verfahren noch einmal von vorn zu beginnen. Dieses Verfahren geht mit einem hohen Befehlsdecodierungs- und Datenverschiebungsaufwand einher, wodurch sich der Mehrwort-Multiplikationsvorgang erheblich verlangsamt.

[0013] Im US-Patent 5 657262 ist eine Recheneinrichtung für die Multiplikation von zwei Zahlen mit doppelter Genauigkeit mit einem Schieberegister und einem Multiplizierer beschrieben, die für eine Multiplikations- und Akkumulations-Operation ausgelegt sind.

[0014] Im Fachgebiet besteht weiterhin ein Bedarf an einem verbesserten Multiplizierer mit einer spezialisierten Architektur, um das Problem der Verarbeitung von langen, mehrere Wortlängen umfassenden Daten in einer effizienten, konsistenten und einheitlichen Weise anzugehen.

[0015] Gemäß der vorliegenden Erfindung wird ein Mikroprozessor geschaffen, wie er durch die Ansprüche definiert ist.

[0016] Diese Patentanmeldung offenbart einen Multipliziererblock, der von einer "Verkettungsvorrichtung" und einer integrierten Barrelshift-Schaltungsanordnung Gebrauch macht, um den Arithmetik-Durchsatz zu erhöhen, während gleichzeitig viele der Belastungen der DSP-Logik vermieden werden. Um die Aufnahme einer doppelt breiten (32 Bits) ALU und eines doppelt breiten Akkumulators zu vermeiden und dennoch das doppelt genaue Ergebnis (32 Bits) des Multiplizierers unterzubringen, wird das 32-Bit-Produkt des statischen Multipliziererblocks in zwei Teile geteilt. Die niedrigeren 16 Bits werden für eine unmittelbare Verwendung (Übergeben, Akkumulieren usw.) der ALU direkt zugeführt. Die höheren 16 Bits werden für eine spätere Verwendung in einem 16-Bit-Register zwischengespeichert (bekannt als Product High). Das Register für den höherwertigen Teil des Produkts kann für eine kontinuierliche Mehrwort-(Ketten-) Multiplikation direkt zu dem Multipliziererblock zurückübertragen werden. Diese Optimierung ist im völligen Einklang mit nur einem Datenspeicher und nur einem Datenbus, Ausführungen, wie sie in den meisten Mikroprozessoren anzutreffen sind, die zwei Zyklen benötigen, um zwei Multiplikationsoperatoren anzuordnen.

[0017] Diese Erfindung optimiert die Ausführung des Multipliziererblocks, um die Kosten zu minimieren und die Flexibilität zu erhöhen, so dass er in sehr preiswerte Mikroprozessoren und Einzelchip-Mikrocomputer integriert werden kann. Die Hauptfunktion des Multipliziererblocks ist die Ausführung einer Einzelzyklus-17-Bit $\times$ 17-Bit-Multiplikation, die ein 34-Bit-Produkt ergibt. In der bevorzugten Ausführungsform kann der Operand entweder als vorzeichenbehaftet oder als vorzeichenlos interpretiert werden. Der Multiplizierer ist fähig, eine Multiplikation eines Einzelwortes (16 Bits) mit einer N-Wort-Kette (bis zu 496 Bits) auszuführen, die ein 512-Bit-Ergebnis hervorbringt.

[0018] Außerdem kann der Multiplizierer eine universelle Barrelshift-Funktion aufweisen. Ein 4-Bit-Verschiebungswertregister mit einem 4-bis-16-Bit-Decodierer ermöglicht dem Multipliziererbblock, eine 1 bis 16 Bits umfassende Barrelshift-Operation entweder an einem 16-Bit-Operanden oder an einem N-16-Bit-Ketten-Operanden durchzuführen. Die langkettige Barrelshift ist einmalig für Multipliziererbblockfunktionen.

[0019] Die verhältnismäßig geringe Fläche, die infolge der Beseitigung spezieller DSP-Strukturen wie etwa einer Vorgriffslogik- und einer Fließbandverarbeitungs-Schaltungsanordnung für den Multipliziererbblock und die Barrelshift-Schaltungsanordnung benötigt wird, ermöglicht dem Prozessor, einen großen Teil der Funktionalität und Leistungsfähigkeit eines digitalen Signalprozessors in ein preiswertes Erzeugnis zu integrieren.

[0020] Der in dieser Patentanmeldung offenbarte Multipliziererbblock ist in einen preiswerten Mikroprozessor integriert, der sich für einen Einsatz in Handelsartikeln eignet. Der Mikroprozessor und sein spezieller Befehlssatz sorgen für eine leistungsfähige Datenverarbeitung von Datentypen, deren Länge von einem Bit bis zu 512 Bit reicht. Die bevorzugte Ausführungsform enthält spezielle Verbesserungen an Hardware-Strukturen und Befehlssatz, die Operationen an Daten mit einer mehrfachen Wortbreite in einer effizienten, konsistenten und einheitlichen Weise vereinfachen. Jedes Befehlswort, das Daten verarbeitet, weist einen reservierten Bitschalter auf, der bewirken wird, dass der auszuführende Befehl entweder nur einmal (auf ein Einwort-Datum einwirkend) oder als eine wiederholte Ausführung desselben Befehls (auf eine Kette oder Liste von sequentiellen Daten einwirkend) ausgeführt wird.

[0021] Gemäß der bevorzugten Ausführungsform sind verschiedene Hardware-Strukturen erforderlich, um diese Definition eines Befehlssatzes mit erweiterter Genauigkeit zu unterstützen. Erstens wurde ein Hardware-Kettenregister mit einem Zähler hinzugefügt, um die Wiederholungszählung und die Anzahl

der Wörter in der Kette zu steuern. Zweitens wurde eine Registerreihe implementiert, um die Akkumulationsfunktion für die ALU zu schaffen. Drittens war eine spezielle Adresssteuerung notwendig, um die sequentielle Gewinnung von bis zu zwei Eingangsoperanden-Ketten und einer Ausgangsoperandenkette zu steuern. Viertens wurde eine zusätzliche ALU-Zustandslogik einbezogen, um mit arithmetischen und logischen Zuständen unabhängig von der Datenbreite einheitlich umzugehen. Fünftens wurde das Register für den höherwertigen Teil des Produkts auf den Partialsummeneingang des Hardware-Multiplizierers gelegt, um eine konsistente Kettenmultiplikationsfunktion zu ermöglichen.

[0022] Die bevorzugte Ausführungsform schafft eine ALU mit einer Länge von einem Wort (**31** in [Fig. 1](#)), liefert das Konzept von Wortketten und sorgt in der bevorzugten Ausführungsform dafür, dass vier verschiedene Speicherbereiche spezifiziert werden können, die jeweils ein Paar Kettenwerte speichern, wodurch die Nachteile von früheren Mikroprozessoren überwunden werden.

[0023] Die offenbarten Ausführungsformen gehen mit Daten in einer konsistenten Art und Weise um, wobei es sehr wenige Einschränkungen hinsichtlich der Anzahl der Bits gibt, die verwendet werden, um Daten darzustellen. Die Software-Befehle für die Verarbeitung entweder einzelner 16-Bit-Wörter oder erweiterter 512-Bit-Wortketten sind im Wesentlichen gleich. Die Ausführungszeit nimmt mit der Anzahl der Wörter in den Operanden bei geringem oder keinem Software-Aufwand linear zu. Der Code ist kompakt, logisch und leicht verständlich.

[0024] Diese und weitere Merkmale und Vorteile der neuartigen Prozessor-Hardware werden einem Fachmann auf dem Gebiet durch Heranziehen der folgenden ausführlichen Beschreibung bevorzugter Ausführungsformen der Erfindung, wobei die Beschreibung auf die Zeichnung Bezug nimmt, offensichtlich sein.

KURZBESCHREIBUNG DER ZEICHNUNG

[0025] Die offenbarten Ausführungsformen der Erfindung werden anhand der beigefügten Zeichnung beschrieben, die wichtige beispielhafte Ausführungsformen der Erfindung zeigt, die durch die Bezugnahme Bestandteil der Beschreibung sind, wobei

[0026] [Fig. 1](#) ein Schaltplan ist, der das offenbarte Multiplikationsschema zeigt;

[0027] [Fig. 2](#) ein detaillierter Plan des 17 $\times$ 17-Multipliziererbblocks ist;

[0028] [Fig. 3](#) ein Blockdiagramm eines Mischsignal-Prozessors zeigt;

[0029] [Fig. 4](#) einen Telefonanrufbeantworter zeigt, der diese Speicheranordnung verwendet;

[0030] [Fig. 5](#) eine Darstellung ist, die im Großen und Ganzen [Fig. 1](#) ähnlich ist, jedoch den Umgang mit Datenketten zeigt;

[0031] [Fig. 6](#) den Aufbau und die Anordnung der Datenspeicher-Adressiereinheit von [Fig. 1](#) genauer zeigt;

[0032] [Fig. 7](#) den Aufbau und die Anordnung der Programmzählereinheit von [Fig. 1](#) genauer zeigt.

AUSFÜHRLICHE BESCHREIBUNG DER BEVORZUGTEN AUSFÜHRUNGSFORMEN

[0033] Die neuartigen Lösungsansätze der vorliegenden Anmeldung werden mit besonderer Bezugnahme auf eine derzeit bevorzugte Ausführungsform beschrieben. Es sollte jedoch klar sein, dass diese Klasse von Ausführungsformen nur wenige Beispiele der vielen vorteilhaften Anwendungen der hier offenbarten neuartigen Lösungsansätze liefert. Im Allgemeinen beschränken Aussagen, die bei der Beschreibung der vorliegenden Ausführung getroffen werden, nicht zwangsläufig irgendwelche der verschiedenen Ausführungsformen der vorliegenden Erfindung. Außerdem können einige Aussagen auf einige neuartige Merkmale zutreffen, jedoch nicht auf andere.

[0034] Der Multiplizierblock gemäß einer bevorzugten Ausführungsform ist von allgemeinem Nutzen und findet eine besondere Gebrauchseignung bei solchen Anwendungen wie etwa schnellen Rechnern und mäßig schnellen digitalen Signalprozessorsystemen (DSP-Systemen). Der Prozessor, in den er integriert ist, kann mit vielen verschiedenen Speicherkonfigurationen und mit vielen verschiedenen Peripheriegeräten verwendet werden, um eine breite Vielfalt von anwendungsspezifischen Handelsartikeln zu schaffen.

[0035] Die hier offenbarte bevorzugte Ausführungsform des Ein-Zyklus-Multiplizierers ist so beschaffen und ausgelegt, dass sie einen 16-Bit-Multiplikator mit einem Multiplikanden multipliziert, der ein Wort aus einer Kette aus mehreren Wörtern umfassen kann. Das resultierende Produkt ist ein Wort länger als die eingegebene Wortkette.

[0036] Die Funktionsweise der Recheneinheit (CU), von welcher der Multiplizierer einen Hauptbestandteil darstellt, ist in [Fig. 1](#) gezeigt. Die Recheneinheit ist aus einem 17-Bit $\times$ 17-Bit-Booths-Algorithmus-Multiplizierblock **30** und einer 16-Bit-Arithmetik-Logik-Einheit (ALU) gebildet. Der Multiplizierblock **30** wird von vier Systemregistern bedient: einem 16-Bit-Multiplikatorregister (MR) **32**, einem

16-Bit-Nurschreib-Multiplikanten-Eingaberegister **33**, einem 16-Bit-höherwertiges-Wort-Register (PH) **34**, das die höchstwertigen 16 Bits einer 32-Bit-Multiplizierer-Ausgabe hält, einem Niedrigerwertiges-Wort-Ausgaberegister **25**, das die niedrigstwertigen 16 Bits einer 32-Bit-Multiplizierer-Ausgabe enthält, und einem 4-Bit-Verschiebungswertregister (SV) **35**.

[0037] Die bevorzugte Ausführungsform ist ein rein statischer 17-Bit $\times$ 17-Bit-Multiplizierblock, der eine 17-Bit $\times$ 17-Bit-Zweierkomplement-Multiplikation und eine Multiplikationsakkumulation in einem einzigen Befehlszyklus ausführt. Die Hardware-Anordnung umfasst neun 17-Bit-Addierer, die eine herkömmliche Booths-Algorithmus-Multiplikationsoperation ausführen. Das siebzehnte Bit ist erforderlich, um sowohl die vorzeichenbehaftete als auch die vorzeichenlose Interpretation der 16-Bit-Eingangsoperandendaten zu vereinfachen.

[0038] Einer oder beide, der Multiplikator und der Multiplikand, kann bzw. können zwangsweise das siebzehnte Bit auf einen positiven Wert gesetzt haben (d. h. ein vorzeichenbehafteter Operand), oder einer oder beide dieser Operanden können zwangsweise ein siebzehntes Bit haben, das einen Binärwert aufweist, der gleich dem Binärwert des sechzehnten Bits ist (d. h. ein vorzeichenloser Operand).

[0039] Die niedrigstwertigen 16 Bits des resultierenden 32-Bit-Produkts werden während desselben Befehlszyklus zu der ALU **31** multiplexiert, und dieses 16-Bit-Produkt wird in ein Wortregister AC(0)-AC(31), das sich in der Akkumulatorregisterreihe **36** befindet, geladen oder mit diesem arithmetisch kombiniert. Die höchstwertigen 16 Bits des 32-Bit-Produkts werden am Ende des Befehlszyklus in PH **34** zwischengespeichert.

[0040] Der Multiplikand für den Multiplizierer **30**, der aus dem Datenspeicher erhalten wird, ist ein Registerwort AC(0)-AC(15) vom Akkumulator **36** oder ein Versatzregisterwort AC(16)-AC(31). Der Multiplikator für den Multiplizierer **30** ist entweder vom 16-Bit-Register MR **32** oder vom 4-Bit-Register SV **35**. MR **32** speichert diesen Multiplikator.

[0041] Die bevorzugte Ausführungsform ist in [Fig. 2](#) genauer gezeigt. Jeder Block **202** ist ein Volladdierer und umcodierender Multiplexer. Die zwei Summanden sind umcodierte multiplexierte Werte von dem Multiplikanden-Bus (MCND) und der verschobenen vorhergehenden Partialsumme von dem Addiererausgang **204** darüber. Die umcodierten Eingaben **206**, die die neun 17-Bit-Schnellübertrag-Addiererböcke **208** steuern, kommen von der Multiplikatorquelle. Die anfängliche Partialsumme in dem ersten 17-Bit-Addiererböck **210** wird mit dem PH-Register bedingt verknüpft, um eine Kettenmultiplikation zu

unterstützen. Die siebzehnte Leitung des Multiplikanden-Busses **212** ist entweder das erweiterte 16-te Bit von MCND oder 0, je nachdem, ob der Multiplizierer in der vorzeichenbehafteten oder in der vorzeichenlosen Multiplikationsbetriebsart ist. Die Addierer **202** verwenden eine Schnellübertragsweitergabe, wobei der Übertrag von dem Addierer auf der rechten Seite genommen wird und nach links weitergegeben wird. Die Bits PO bis P15 der Ausgabe (niedrigwertiger Teil des Produkts) gelangen in den Arithmetik-Logik-Einheit-Multiplexer **214**. Die Bits P16 bis P31 gehen zur Verwendung in einer Kettenmultiplikation in das Register für den höherwertigen Teil des Produkts ein.

[0042] In der bevorzugten Ausführungsform wird das resultierende Produkt ein Wort länger als der Multiplikand sein. Da die Länge des Multiplikanden durch den Kettenregister-Zähler gesteuert wird, ist für alle Kettenmultiplikationen ein zusätzlicher Zyklus erforderlich. Folglich werden in der bevorzugten Ausführungsform, obwohl 16×16 -Bit-Operationen keinen zusätzlichen Zyklus benötigen, Kettenmultiplikationen und Barrelshift-Operationen einen zusätzlichen Zyklus für die abschließende Akkumulation erfordern.

[0043] Wie weiter oben beschrieben ist, multipliziert die bevorzugte Ausführungsform zwei 17-Bit-zweierkomplementcodierte Zahlen, woraus sich ein Zweierkomplement-34-Bit-Produkt ergibt. Wenn die 16-Bit-Operanden als vorzeichenlos interpretiert werden sollen, werden sie einfach mit einem siebzehnten positiven Vorzeichenbit erweitert, die niedrigstwertigen 32 Bits des Produkts werden dann das korrekte vorzeichenlose Ergebnis angeben. Wenn die Daten als vorzeichenbehaftet interpretiert werden sollen, wird das siebzehnte Bit gleich dem sechzehnten Bit gesetzt (d. h. das Vorzeichenbit wird erweitert), so dass die niedrigstwertigen 32 Bits des Produkts das korrekte vorzeichenbehaftete Ergebnis enthalten werden.

[0044] Beide, der Multiplikator wie auch der Multiplikand, können unabhängig voneinander gezwungen werden, das siebzehnte Bit zwangsläufig auf einen positiven Wert (0) oder auf einen Wert, der dem höchstwertigen Bit des geladenen Operanden gleich ist (sechzehntes Bit), zu haben. Diese Auswahl ermöglicht sowohl eine vorzeichenbehaftete als auch eine vorzeichenlose Multiplikation. Die niedrigstwertigen 16 Bits des resultierenden Produkts werden während desselben Befehlszyklus zu der ALU multiplexiert und entweder in ein Akkumulatorregister geladen oder mit diesem arithmetisch kombiniert.

[0045] Da der Multiplizierer die Operanden entweder als vorzeichenbehaftet oder als vorzeichenlos interpretieren kann, erfordert eine Kettenmultiplikation, dass alle Wortmultiplikationen außer der letzten einer vorzeichenlosen Betriebsart unterworfen werden.

Um für eine korrekte und einheitliche Arithmetik und Zustandsgewinnung in Bezug auf Einzeldatenwortoperationen zu sorgen, wird das Bit des ausgehenden Übertrags mit dem Bit des eingehenden Übertrags der nächsten Wortberechnung verknüpft, und das Wort-Zustandsbit (gleich oder null) wird logisch UND-verknüpft. Um Additionen und Subtraktionen korrekt vorzunehmen, spiegeln außerdem Vergleiche der langen Ketten den korrekten Zustand wider.

[0046] Während einer Kettenmultiplikation (Multiplikation eines mehrere Wörter umfassenden Operanden) werden die Inhalte des PH-Registers auf den Partialsummeneingang des ersten Addierers in dem Multipliziererblock zurückgeführt. In diesem Zustand ist der Multiplizierer vollständig dafür gerüstet, die Multiplikation des nächsten Wortes in der Kette auszuführen. Diese Ausführung ermöglicht, dass der Multipliziererblock eine Multiplikation von Operanden, die mehrere Wörter lang sind, mit einem Minimum an Ausführungszeitaufwand ausführt.

[0047] Zu Beginn jeder Multiplikation wird das Register für den höherwertigen Teil des Produkts (PH-Register) auf null gesetzt. Für den zweiten bis n-ten Zyklus werden die höchstwertigen 16 Bits der letzten Wortmultiplikation in das PH-Register geleitet. Dies ermöglicht ohne weiteren Software-Aufwand $N \times 1$ -Wortmultiplikationen, ein $N + 1$ -Wortprodukt in $N + 1$ Befehlszyklen zu erzeugen. Durch die Erweiterung der Elementkette mit einem erweiterten Vorzeichenwort können sehr leistungsfähige N -Wortketten $\times$ M -Wortketten-Multiplikationen softwaremäßig implementiert werden.

[0048] In der bevorzugten Ausführungsform kann die Quelle des Multiplikanden entweder ein Datenspeicher, ein Akkumulator oder ein Akkumulator-Versatz sein. Die Quelle des Multiplikators kann entweder das 16-Bit-Multiplikatorregister (MR) für Multiplikationsoperationen oder das Schiebewertregister (SV-Register) für Barrelshift-Operationen sein. Bei allen Multiplikationsoperationen speichert das MR-Register den Multiplikator.

[0049] Alle drei Multiplikationsregister (PH, SV und MR) können vom Speicher geladen und in den Speicher gespeichert werden. Außerdem können Daten von einem Akkumulatorregister zu PH oder umgekehrt übertragen werden. Sowohl lange als auch kurze Konstanten können aus dem Programmspeicher direkt in das MR-Register geladen werden. Der Wert des Multiplikanden wird von dem internen Datenbus (DB) in einem Nurchreib-Register zwischengespeichert, wobei dieser Wert weder gelesen noch in einen Speicher noch in andere Systemregister übertragen werden kann.

[0050] Dieser Multiplizierer führt Barrelshift-Operationen während der Benutzung als integrierter

4-bis-16-Bit-Decodierer unter Verwendung des 4-Bit-Verschiebewertregisters (SV) als Eingang aus. Wenn beispielsweise ein Wert von 7 (binär 0111) in das SV-Register geladen ist, wird ein Multiplikator 0000 0000 1000 0000 (binär) in den Multipliziererbblock geladen. Die Multipliziererbblockausgabe wird dann den Wert des Multiplikanden, um 7 Bitpositionen nach links verschoben, widerspiegeln. Wenn das Zustandsbit FM (Multiplizierer-Verschiebungsbetriebsart) in dem Zustandsregister (STAT) auf eins gesetzt ist, wird der Multiplikator um ein Bit nach links verschoben, um einen Operanden mit 17 signifikanten Bits zu bilden. Diese Betriebsart ist enthalten, um eine Division durch 2 des Produkts zu vermeiden, wenn die eingegebenen Operanden als vorzeichenbehaftete Binärbrüche interpretiert werden.

[0051] Das weiter oben beschriebene Konzept von Ketten ist nicht neu, und andere Prozessoren haben Mechanismen, um einen bestimmten Befehl eine vorgegebene Anzahl von Malen zu wiederholen. Die bevorzugte Ausführungsform ist umfassender und liefert einen sehr leistungsstarken Code für hochpräzise Berechnungen. Sie ist konsistent für alle Multiplikations- und Verschiebeoperationen, wobei sie den korrekten Zustand für Datenbreiten von einem Bit bis zu Ketten von 512 Bits bewahrt. Der Code für die Ausführung einer Einzeloperation oder einer Kettenoperation ist völlig gleich und kompakt. Die Hardware-Multiplizierer-Operation sorgt für eine sehr effiziente und einfache Ausführung von Multiplikationen von Produkten aus bis zu 256×256 Bits = 512 Bits. Dies bietet deutliche Vorteile gegenüber aktuellen Ausführungen, die die Kettenmultiplikation nicht angehen. Ferner verwenden viele bekannte Systeme einen unveränderlichen Hardware-Multiplizierer, um Multiplikationen mit erweiterter Präzision unter Verwendung eines "Multiplikation-in-Teilen"-Algorithmus auszuführen, der kompliziert und viel weniger leistungstark ist.

[0052] Dieser Multipliziererbblock ist so beschaffen, dass er in einen preiswerten Mischsignal-Prozessor integriert werden kann. Die zu Grunde liegende Architektur des Prozessors, der die derzeit bevorzugte Ausführungsform verkörpert, enthält eine schnelle Recheneinheit (CU) und eine vollständig ausgebildete Datenspeicher-Adressiereinheit (DMAU). In der bevorzugten Ausführungsform verkörpert der Prozessor eine Harvard-Architektur (getrennte Daten- und Programmspeicherräume), die folglich einen parallelen Zugriff und einen maximalen rechnerischen Durchsatz ermöglicht. Um eine minimale Leistungsaufnahme zu erzielen, bilden statische Logikimplementierungen die funktionalen Blöcke des Prozessors, wobei die wichtigsten funktionalen Blöcke deaktiviert werden, wenn sie nicht gebraucht werden. Eine Anzahl interner Speicher verschiedener Größen und Typen kann mit aufgabenspezifischen Peripheriegeräten und einer Unterbrechungslogik kombiniert wer-

den, um eine breite Vielfalt von aufgabenspezifischen Vorrichtungen zu schaffen, die jeweils eine andere Gebrauchseignung haben.

[0053] In der derzeit bevorzugten Ausführungsform, jedoch ohne Beschränkung auf diese, sind der Programmspeicher und der Datenspeicher auf interne Speicherblöcke eingeschränkt, die nicht durch die Verwendung eines externen Speichers erweitert werden können. Der Programmspeicher umfasst ROM, EPROM oder OTP, beispielsweise mit der Kapazität von 64 K 17-Bit-Wörtern. Der Datenspeicher ist RAM. Gemäß der Idee und dem Umfang der vorliegenden Anmeldung besitzen verschiedene Ausführungsformen verschiedene Kombinationen von Programm- und Datenspeichergrößen, wobei der Programmspeicher im Allgemeinen wenigstens 16 K 17-Bit-Wörter umfasst. Wenn ein Massendatenspeicher erforderlich ist, kann eine zusätzliche Datenspeicher-Peripherie-Schnittstelle vorgesehen werden.

[0054] Wie aus **Fig. 3** ersichtlich ist, umfasst ein Prozessor **10**, der einen Multipliziererbblock gemäß der bevorzugten Ausführungsform enthält, eine Anzahl wichtiger Unterblöcke, einschließlich eines Programmdaten-Speicherblocks **11** und eines Daten-Speicherblocks **12**. Die wichtigen Unterblöcke umfassen die oben erwähnte Recheneinheit (CU) **13** und die oben erwähnte Datenspeicher-Adressiereinheit (DMAU) **14**, eine Programmzählereinheit (PCU) **15** und einen Befehlsdecodierer **16**. Weitere Funktionen werden durch ein Wiederholungs- oder Ketten-Zählerregister **17**, ein Zustandsregister **18**, zwei Zeitgeber **19** und **20**, eine Unterbrechungslogik **21** und eine Peripherie-Erweiterungsschnittstelle **22** bereitgestellt.

[0055] Ein 17-Bit-Datenbus (DB) **23** sorgt für eine Nachrichtenverbindung zwischen den funktionalen Blöcken innerhalb des Prozessors **10**. Die meisten der Register in dem Prozessor **10** haben einen Lese- und Schreibzugriff auf DB **23**. Die (nicht gezeigten) Bustreiber sind statische Vorrichtungen, um eine unnötige Leistungsaufnahme zu vermeiden und um für eine größtmögliche Logik-Durchlaufzeit zu sorgen. Die kleinstmögliche Befehlsperiode des Prozessors **10** beträgt bei einem vorgesehenen Prozessortakt von 10 MHz (nicht gezeigt) ungefähr 100 ns.

[0056] Der Datenspeicher **12** von **Fig. 3** ist als eine Menge von parallelen 17-Bit-Wörtern organisiert. Die Anzahl der Wörter variiert entsprechend der Anwendung, für welche der Prozessor **10** eingesetzt wird; aber der Bereich von 256 bis 2048 Wörtern ist beispielhaft, wobei in **Fig. 3** 1152 Wörter angegeben sind. Jede Adresse **51**, die von der Datenspeicher-Adressiereinheit (DMAU) **14** geliefert wird, bewirkt die Adressierung von 17 Bits Daten. Diese **17** Bits werden in Abhängigkeit von dem Befehl, der aus-

geführt wird, auf verschiedene Weise verarbeitet. Bei den meisten Befehlen werden diese Daten als ein 16-Bit-Wort-Format interpretiert. Zwei Byte-Befehle wie etwa LACB und SACB bewirken, dass der Prozessor **10** Daten in einem 8-Bit-Wort-Format, das auch als Byteformat bezeichnet wird, liest oder schreibt. Diese Byteformat-Betriebsart veranlasst die Prozessor-Hardware, entweder das höherwertige oder das niedrigerwertige Byte des adressierten 16-Bit-Wortes zu lesen oder zu schreiben, wobei das abgerufene Byte auf dem Datenbus **23** rechts ausgerichtet wird.

[0057] In einer Merkerdaten-Betriebsart verarbeitet ein Befehl nur das siebzehnte Bit des abgerufenen Wortes. Das siebzehnte Bit wird bei allen Lesezugriffen auf den Datenspeicher **12** stets gelesen und dann in das MTAG-Bit des Zustandsregisters **18** von **Fig. 3** geladen und liefert folglich die Kennzeichnung entweder für Wort- oder Bytedaten. Bei der Byte-Betriebsart ist zwei aufeinander folgenden Bytes die gleiche Kennzeichnung zugeordnet. Gekennzeichnete Daten werden von den Befehlen FIR, FIRK, COR und CORK verwendet, um einen Rundpuffer zu emulieren. Außerdem kann das MTAG-Bit des Zustandsregisters als eine Bedingung für Verzweigungs-/Aufruf-Befehl getestet werden oder es kann mit anderen Testbedingungen und anderen Merkern kombiniert werden, um neue Bedingungen zu erzeugen.

[0058] **Fig. 1** zeigt die CU **13** von **Fig. 3** genauer. Die CU **13** enthält einen 17-Bit $\times$ 17-Bit-Booths-Algorithmus-Multiplizierer **30** und eine 16-Bit-ALU **31**.

[0059] Wie weiter oben erläutert worden ist, wird der Multiplizierer **30** von vier Systemregistern bedient, d. h. einem 16-Bit-Multiplikatorregister (MR) **32**, einem 16-Bit-Nurschreib-Multiplikanten-Eingaberegister **33**, einem 16-Bit-höherwertiges-Wort-Register (PH) **34**, das die höchstwertigen 16 Bits einer 32-Bit-Multiplizierer-Ausgabe hält, einem Niedrigerwertiges-Wort-Ausgaberegister **25**, das die niedrigstwertigen 16 Bits einer 32-Bit-Multiplizierer-Ausgabe enthält, und einem 4-Bit-Verschiebungswertregister (SV) **35**.

[0060] Die 16-Bit-Ausgabe **50** der ALU **31** wird an einer 16-Bit-Position einer 32-Wort-Positionen-Akkumulatorregisterreihe **36** gespeichert. Die Akkumulatorregisterreihe **36** wird wirksam, um der ALU **31** einen 16-Bit-Operanden zu liefern (wobei dieser eine Operand eines der adressierten 32 Wörter in der Akkumulatorregisterreihe oder im Versatz dazu ist), oder die Akkumulatorregisterreihe **36** wird wirksam, um der ALU **31** zwei 16-Bit-Operanden zu liefern (wobei diese zwei Operanden das adressierte Akkumulatorregisterreihen-Wort und sein Versatz sind).

[0061] Wie weiter oben erörtert worden ist, führt der Multiplizierer **30** eine 17-Bit $\times$ 17-Bit-Zweierkomple-

ment-Multiplikation in einem einzigen Befehlszyklus des Prozessors **10** aus. Einer oder beide der Multiplikationsoperanden und der Multiplikand können zwangsweise das siebzehnte Bit auf einen positiven Wert gesetzt haben (d. h. ein vorzeichenbehafteter Operand), oder einer oder beide dieser Operanden kann bzw. können zwangsweise ein siebzehntes Bit haben, dass einen Binärwert aufweist, der gleich dem Binärwert des sechzehnten Bits ist (d. h. ein vorzeichenloser Operand).

[0062] Die niedrigstwertigen 16 Bits des resultierenden 32-Bit-Produkts werden während desselben Befehlszyklus zur ALU **31** multiplexiert, und dieses 16-Bit-Produkt wird in ein Wortregister AC(0)-AC(31), das sich in der Akkumulatorregisterreihe **36** befindet, geladen oder mit diesem kombiniert. Die höchstwertigen 16 Bits des 32-Bit-Produkts werden am Ende des Befehlszyklus in PH **34** zwischengespeichert.

[0063] Der Multiplikand für den Multiplizierer **30**, der aus dem Datenspeicher **12** von **Fig. 3** erhalten wird, ist ein Registerwort AC(0)-AC(15) vom Akkumulator **36** oder ist sein Versatzregisterwort AC(16)-AC(31). Der Multiplikator für den Multiplizierer **30** ist entweder vom 16-Bit-MR **32** oder vom 4-Bit-SV **35**. MR **32** speichert diesen Multiplikator.

[0064] Für Barrelshift-Befehle ist der Multiplikator ein 4-bis-16-Bit-decodierter Wert, der das 4-Bit-SV **35** als Eingabe verwendet. Beispielsweise führt ein hexadezimaler Wert von 7H in dem 4-Bit-SV **35** zu dem binären Multiplikator 0000 0000 1000 0000, bewirkt folglich eine Linksverschiebung um 7 Bits zwischen dem 16-Bit-Multiplikanden und dem 32-Bit-Produkt.

[0065] Alle drei Register PH **34**, MR **32** und SV **35** können vom Datenspeicher **12** der **Fig. 3** geladen werden, und der Inhalt dieser Register kann im Speicher **12** gespeichert werden. Außerdem können zwischen einem Register AC(0)-AC(31) der Akkumulatorregisterreihe **36** und PH **34** Daten direkt ausgetauscht werden. Sowohl lange als auch kurze Konstanten können aus dem Programmspeicher **11** direkt in MR **32** geladen werden. Ein Wert eines Multiplikanden wird von dem Datenbus (DB) **23** in einem Nurschreib-Multiplikanden-Register **33** zwischengespeichert, wobei dieser Multiplikandenwert weder gelesen noch in einen Speicher oder in Systemregister übertragen werden kann.

[0066] Die ALU **31** von **Fig. 1** ist insofern das Kernstück der CU **13**, als die ALU **31** der Ort in der CU **13** ist, an dem Daten arithmetisch addiert, subtrahiert oder verglichen werden. Außerdem können in der ALU **31** logische Operationen an Daten vorgenommen werden. Die Wortlänge der ALU **31** beträgt 16 Bits. Die meisten Arithmetik-/Logik-Befehle können jedoch auch eine Wortkette verarbeiten, die eine An-

zahl von 16-Bit-Wörtern umfasst. Die Akkumulatorregisterreihe **36**, die zweiunddreißig 16-Bit-Speicherregister AC(0)-AC(31) umfasst, verbessert die Leistungsfähigkeit der ALU **31** hinsichtlich einer minimalen Ausführungszeit und eines minimalen Programmsteuercodes.

[0067] Die ALU **31** hat zwei Eingänge **37**, **38**, die in [Fig. 1](#) mit A und B bezeichnet sind. Der Eingang **37** (A) enthält einen ersten Multiplexer **39**, während der Eingang **38** (B) einen zweiten Multiplexer **40** enthält.

[0068] Der Multiplexer **39** wählt eine von fünf Eingaben aus, d. h. er wählt zwischen einer Eingabe "alles Nullen" **41**, einer Eingabe **42**, die von PH **34** geliefert wird, einer Eingabe **43**, die von DB **23** geliefert wird, einer Eingabe **44**, die vom PL-Ausgang **25** des Multiplizierers **30** geliefert wird, oder einer Eingabe **45**, die von dem Versatz-Ausgang **46** oder AA der Akkumulatorregisterreihe **36** geliefert wird, aus. Der Multiplexer **40** wählt zwischen der Eingabe "alles Nullen" **41** oder der Ausgabe AA **46**, die von der Akkumulatorregisterreihe **36** geliefert wird, aus.

[0069] Die Eingabe "alles Nullen" **41** ist für Transfer- und Unitaritätsoperationen erforderlich und dient außerdem als Standardwert, um die Leistungsaufnahme zu minimieren.

[0070] Die DB 23-Eingabe **43** in den Multiplexer **39** wird verwendet, um Speicherwerte aus dem Daten-RAM **12** und konstante Werte, wie etwa Programmspeicher-Operanden, an die ALU **31** zu übertragen. Die PH 34-Eingabe **42** und die PL 25-Eingabe **44** in den Multiplexer **39** dienen Multiplikations-Akkumulations-Operationen der ALU **31**.

[0071] Die Akkumulatorregisterreihe **36** weist zwei Ausgänge **46**, **47** auf, die in [Fig. 1](#) mit AA und AB bezeichnet sind. Diese zwei Ausgänge **46** (AA) und **47** (AB) sind an die ALU **31** angeschlossen, um der ALU **31** eine Akkumulator-Referenz und eine Versatz-Akkumulator-Referenz zu liefern.

[0072] Von der ALU **31** unterstützte Operationen schließen die Arithmetikoperationen Addition, Subtraktion und Laden (zu null addieren), logische Operationen wie etwa UND, ODER, exklusives ODER, NICHT, Linksverschiebung, Rechtsverschiebung und Vergleichsoperationen, die einen Vergleich mit einer Konstanten oder mit Speicherwerten liefern, ein, wobei die Werte, die in der Akkumulatorregisterreihe **36** sind, nicht vernichtet werden.

[0073] Die bevorzugte Ausführungsform schafft eine gepaarte 32-Wort- (16 Bits pro Wort) Akkumulatorregisterreihe (AC(0)-AC(31) in [Fig. 3](#)), statt wie in früheren Ausführungen einen einfachen oder doppelten Akkumulator zu verwenden. Außerdem werden einmalige Merkmale in der Adressiereinheit (**14** in

[Fig. 3](#)) des offenbaren Prozessors geschaffen, die zur Wirkung kommen, um diese Akkumulatorregisterreihe zu adressieren.

[0074] Der Akkumulatorblock der Recheneinheit besteht aus **32** Akkumulatoren mit einer Größe von jeweils 16 Bit. Die **32** Register der Akkumulatorregisterreihe **36** sind mit AC(0) bis AC(31) bezeichnet, wobei jedes Register ein 16-Bit-Register ist. Vier mit ACP(0) bis ACP(3) bezeichnete 5-Bit-Zeigerregister **49** sind für die Speicherung von Adresszeigern für die Akkumulatorregisterreihe **36** vorgesehen.

[0075] Der gespeicherte Inhalt jedes ACP-Zeigerregisters **49** wird als ein direkter Verweis auf oder eine direkte Adresse eines der sechzehn AC-Register AC(0) bis AC(15) oder als ein indirekter Verweis auf eines der Versatz-AC-Register AC(16)-AC(31), das einen direkten Verweis hat, der um **16** versetzt ist, verwendet. Das heißt das Register AC(0) hat ein versetztes Register bei der Adresse **16**, der Versatz des Registers AC(1) ergibt die Registeradresse **17** usw. Da der Registerblock, der die Register AC(0) bis AC(31) enthält, zirkular ist, d. h. das Inkrementieren der Adresse **31** zur Adresse **32** die Adresse-0 ergibt, ergibt der Versatz der Register **16-31** jeweils die Register **0-15**.

[0076] In Abhängigkeit vom Programmbefehl wird einer oder werden beide der Ausgänge **46** (AA) und **47** (BB) der Akkumulatorregisterreihe **36** verwendet. Einige Befehle beziehen sich nur auf den Ausgang **46** (AA) und können den Versatz-Ausgang **47** (AB), der gleichzeitig mit dem Ausgang **46** (AA) abgerufen wird, nicht verwenden oder modifizieren. Andere Befehle bieten ein Auswahlfeld in dem Befehlswort, wofür ein Beispiel das Operationscode-Bit "oa" ist. Diese letzteren Befehle können die Spaltenadressierungsrichtung und folglich die Quelle oder die Reihenfolge der Ausgabe- und der Versatz-Ausgaberegister austauschen. Außerdem können einige Befehle steuern, dass die ALU-Ausgabe **50** entweder in ein Akkumulatorregister AC(0)-AC(15) oder in ihr Versatz-Akkumulatorregister AC(16)-AC(31) geschrieben wird.

[0077] Die Akkumulatorregisterreihe **36** sorgt für einen Arbeitsbereich, der die Notwendigkeit von vielen Zwischenspeicher-Ladeoperationen und -Speicherooperationen beseitigt. Außerdem enthält der Befehlsatz für den Prozessor **10** ein 2-Bit-Feld oder einen Bitschalter für alle Akkumulatorregisterreihen-Verweisbefehle, so dass ein direkter Verweis auf eines der vier Zeigerregister **49** erfolgen kann, wodurch die Speicherüberlastung, die häufig bei Ausführungen mit einem einzigen Akkumulatorregister auftritt, beseitigt wird.

[0078] Eine Adresse einer Akkumulatorregisterreihe **36** ist in einem der vier ACP-Zeigerregister **49** ge-

speichert, und ein direkter Verweis auf ein Zeigerregister ACP(0)-ACP(3) erfolgt in dem 2-Bit-"ap"-Feld aller auf den Akkumulator bezogenen Befehle. SAPn-Befehle speichern die Inhalte der ACP(n)-Register in den Speicher **12**, wohingegen LAPn-Befehle aus dem Speicher **12** in das ACP(n)-Register laden. Ein AKACP-Befehl bzw. ein LKACP-Befehl addiert bzw. lädt 5-Bit-Konstanten zu dem aktuellen ACP-Wert.

[0079] Die ACP-Register **49** werden von einem 5-Bit-Prozessor (**80** in **Fig. 5**) bedient, der für eine effiziente Aufeinanderfolge der Akkumulatoradressen bei sich wiederholenden Operationen sorgt, die mit langen Datenketten verbunden sind oder die mit Umadressierungsoperationen auf einer Liste von Daten verbunden sind. Zur Verarbeitung einer Mehrwort-Datenkette wird eine Adresse aus einem ACP-Register kopiert, um das niedrigstwertige Wort in der Kette abzurufen. Diese kopierte Adresse wird dann wiederholt inkrementiert, um die übrigen Wörter in der Kette abzurufen. Dadurch wird die in diesem besonderen ACP-Register **49** gespeicherte Adresse unverändert gelassen, wobei der Wert in diesem ACP-Register noch immer auf den Speicherort des niedrigstwertigen Wortes der Wortkette zeigt. Folglich ist dieses ACP-Register für eine Folgeoperation an dieser Wortkette bereit.

[0080] Der Prozessor **10** kann einen Wert in einem ACP-Register **49** um den Wert +1 oder um den Wert -1 vorweg-ändern, bevor das ACP-Register bei einem Befehl mit Bezug auf einen Akkumulator durch die Akkumulatorregisterreihe **36** verwendet wird. Dies minimiert den Software-Aufwand bei der Handhabung einer Adresse der Akkumulatorregisterreihe **36**, beispielsweise durch Vermeiden einer Software-Überlappungswirkung, die nachträgliche Änderungen verursacht.

[0081] **Fig. 6** zeigt eine Datenspeicher-Adressiereinheit (DMAU) **14**, die wirksam wird, um eine Adresse **51** an den Datenspeicher **12** zu liefern. Die Datenspeicher-Adressiereinheit **14** enthält eine zugeordnete ALU **52**, die Additions-, Subtraktions- und Vergleichsfunktionen ausführt, drei spezielle Adressregister **53** (R5 INDEX), **54** (R6 SEITE) und **55** (R7 PGSTK) sowie fünf universelle Register **56-60** (R0 bis R4). Die Datenspeicher-Adressiereinheit **14** erzeugt eine Adresse **51** parallel zur Recheneinheit **13**, wodurch für einen hohen Prozessor-Durchsatz gesorgt wird.

[0082] Die Datenspeicher-Adressiereinheit **14** kann den Datenspeicher **12** mittels fünf Adressierungsbetriebsarten adressieren, d. h. mittels einer direkten Adressierungsbetriebsart, einer indirekten Adressierungsbetriebsart, wobei eine nachträgliche Modifikation möglich ist, und dreier relativer Adressierungsbetriebsarten. Die drei relativen Betriebsarten sind (1)

unmittelbar-kurz mit Bezug auf R6-SEITE (R6-SEITE + 7-Bit (128 Bytes oder 64 Wörter)), (2) R5-INDEX mit Bezug auf die Registerbasis (Indirektes Register + R5-INDEX), und (3) langzwischenzeitlich mit Bezug auf die Registerbasis (indirektes Register + 16-Bit-Versatz).

[0083] Bei der Ausführung von Wortketten-Befehlen wird bei allen Adressierungsbetriebsarten mit Ausnahme der indirekten mit nachträglicher Modifikation eine temporäre Kopie der Adresse **51** verwendet, um das niedrigstwertige Wort der Wortkette abzurufen. Über der nächsten Sequenz der Befehlszyklen wird diese temporäre Adressenkopie selbsttätig inkrementiert, um die übrigen Wörter in der Wortkette abzurufen. Da diese sequentielle Erhöhung der Adresse **51** temporär ist, bleiben alle Register **55-60** unverändert und verweisen weiterhin auf das niedrigstwertige Wort im Speicher **12**.

[0084] Kettenabrufe, die die indirekte Adressierungsbetriebsart mit nachträglicher Modifikation verwenden, werden wirksam, um die modifizierte Adresse **51** für jeden Prozessorzyklus, an dem die Wortkette beteiligt ist, in ein indirektes Register zurückzuschreiben. Dies hat zur Folge, dass die Adresse in dem indirekten Register auf das Datenwort zeigt, dessen Adresse eins über dem höchstwertigen Wort in der Wortkette ist.

[0085] **Fig. 7** zeigt die Programmmählereinheit (PCU) **15**, die wirksam wird, um die Adresse **61** für den Programmspeicher **11** von **Fig. 3** zu erzeugen. Es ist eine 16-Bit-ALU **62** vorgesehen, um die Adressen **61** des ROM **11** zu inkrementieren und zu laden. Zwei Adressregister **63** (PC) und **64** (DP) werden wirksam, um die ROM-Programm-Adresse **61** in PC **63** zu speichern und eine Datenzeigeradresse in DP **64** zu speichern.

[0086] Der Inhalt von PC **63** wird bei Abrufen und Unterbrechungen in den Programmmähler-Stapel gerettet. Ein Programmmähler-Stapel umfasst ein (nicht gezeigtes) Hardware-Register, das sich am oberen Ende des Stapels befindet, gefolgt von einem Software-Stapel, der durch den Datenspeicher **12** geliefert wird, wobei das Register R7-PCSTK von **Fig. 6** in der indirekten Betriebsart als Stapelzeiger verwendet wird. Folglich wird ein tiefer Stapel, der softwaremäßig definiert ist, mit geringem Ausführungsaufwand geschaffen.

[0087] Der Inhalt von DP **64** wird bei Nachschlagtabellen-Befehlen wie etwa LUAA und LUAM aus der Akkumulatorregisterreihe **36** geladen und beim Abruf von langkettigen Konstanten von PB **24** über DB **23** geladen. Der Inhalt von DP **64** kann in den Datenspeicher **12** gespeichert werden. In der Programmmählereinheit **15** ist eine Programmerrfassungslogik **66** enthalten, um die Programmadressbereiche zu

erfassen, die vor einem externen Zugriff geschützt werden sollen.

[0088] Fig. 5 ist im Großen und Ganzen [Fig. 1](#) ähnlich, zeigt jedoch eine Datenkettenverarbeitung. Die Verarbeitung von Datenketten, d. h. die Verarbeitung von Daten als Liste oder Folge einer Anzahl von 16-Bit-Wörtern ist eine wertvolle Gebrauchseignung des Mikroprozessors mit einem Aufbau und einer Anordnung wie weiter oben anhand [Fig. 1](#) bis [Fig. 4](#) beschrieben. Die Kettenverarbeitung von Daten umfasst alle arithmetischen Befehle, logische Befehle und Transferanweisungen. Kettenbefehle wirken auf eine Kette mit dem niedrigstwertigen Wort der Kette beginnend und mit dem höchstwertigen Wort der Kette endend ein. Siehe beispielsweise die Akkumulatorregisterreihe **36** von [Fig. 5](#), wo von einer Kette mit neun Wörtern ihr niedrigstwertiges Wort im Register AC(3) gespeichert ist, während ihr höchstwertiges Wort im Register AC(11) gespeichert ist.

[0089] Nicht weniger als drei Operanden sind an Datenkettenoperationen beteiligt, beispielsweise zwei Eingangsoperanden und ein Ausgangsoperand. Der Prozessor **10** von [Fig. 3](#) ist in der Lage, drei verschiedene Speicherorte für jeden dieser drei Operanden zu adressieren.

[0090] Beispielsweise kann ein Multiplikand **72** unter Verwendung der DMAU **14** und ihrer ALU **81**, um eine Adresse oder Zeigersteuerung **51** zu liefern, im Datenspeicher **12** adressiert werden. Der zweite eingehende Operand oder zweite Summand und der Ausgangsoperand von der ALU **31** können durch die in [Fig. 5](#) gezeigte Akkumulatorzeigereinheit **70** und ihre ALU **80**, die eine indirekte Adressierung verwendet und Zeiger-ACP-Register **49** enthält, adressiert werden. Der zweite eingehende Operand und der Ausgangsoperand können durch ACP **70** als Paar adressiert werden, d. h. als ein Akkumulatorregister AC(0)-AC(15) und sein gepaarter Versatz-Akkumulator AC(16)-AC(31), siehe beispielsweise das Registerpaar AC(13) und AC(29) von [Fig. 5](#).

[0091] Außerdem werden die Kettenregisterzähler **17** von [Fig. 3](#) genutzt, um die Länge einer Kette von 16-Bit-Wörtern als eine ganze Zahl zu speichern und zu steuern. Für Kettenbefehle oder -operationen inkrementieren DMAU **14** von [Fig. 3](#) und ACP **70** von [Fig. 5](#) selbsttätig die Kettenadressen für den Datenspeicher **12** und für die Akkumulatorregisterreihe **36**, jeweils mit dem niedrigstwertigen Wort beginnend, um dadurch alle Wörter oder Werte, welche die zwei Wortketten enthalten, die im Speicher **12** und im Akkumulator **36** enthalten sind, sequentiell zu lesen. Kettenbefehle, die eine Kette mit N Wörtern verarbeiten, erfordern das Hinzufügen eines zusätzlichen Zykluszählers zu der Programmsequenz, um das N + 1-te Wort, das den Produkt-Austrag des Hardware-Multiplizierers **30** umfasst, zu akkumulieren

oder zu übergeben.

[0092] Die ALU **31** und der Multiplizierer **30**, die in [Fig. 1](#), 2 und 5 gezeigt sind, sorgen für eine Steuerung, um die Verarbeitung der 16-Bit-Wort-Ketten anzupassen. Die ALU **31** schickt selbsttätig einen ausgehenden Übertrag ihrer sechzehnten Stufe aus und benutzt ihn dann im nächsten Prozessorzyklus für die nächste signifikante Kettenwortoperation als eingehenden Übertrag in die ALU **31**. Außerdem wird eine "GLEICH-ZU"-Zustandsbedingung festgehalten und zu der nächsten signifikanten Wortoperation geschickt, um mit dem nächsten Wortzustand UND-verknüpft zu werden. Der Endzustand des Vorzeichenüberlaufs wird nicht eher festgehalten, als das letzte oder höchstwertige Wort verarbeitet ist.

[0093] Der Hardware-Multiplizierer **30** weist einen Anschluss **71** für ein anfängliches Teilprodukt auf. Zu Beginn jeder Multiplikationsoperation wird der Eingang **71** auf null gesetzt. Bei dem zweiten bis N-ten Zyklus werden jedoch die höchstwertigen 16 Bits der letzten Wortmultiplikation zum Anschluss **71** geleitet. Dies ermöglicht ohne Software-Aufwand, dass $N \times 1$ -Wortmultiplikationen ein $N + 1$ -Wortprodukt in $N + 1$ Befehlszyklen erzeugen.

[0094] Es wird nun ein Beispiel für die Funktionsweise der bevorzugten Ausführungsform anhand der mathematischen Operation $(A) \cdot (B) + (C)$ gegeben, wobei sich auf [Fig. 5](#) bezogen wird. Bei dieser Operation umfasst der Term (B) einen 8-Wort-Ketten-Multiplikanden (128 Bits), der mit dem Term (A), einem 1-Wort-Multiplikator (16 Bit) multipliziert wird, wobei das sich ergebende Produkt $(A) \cdot (B)$ einem Term (C), einem Summanden, hinzugefügt wird, der eine 9-Wort-Kette (144 Bits) umfasst.

[0095] Gemäß der bevorzugten Ausführungsform sind die drei Datenterme (A), (B) und (C) in [Fig. 5](#) folgendermaßen gespeichert: Der acht Wörter umfassende Term (B), der Multiplikand, ist im Datenspeicher **12** gespeichert, wobei sein niedrigstwertiges Wort an der Speicheradresse MCND ist und sein höchstwertiges Wort an der Speicheradresse MCND + 7 ist; (2) der neun Wörter umfassende Term (C), der Summand, ist in der Akkumulatorregisterreihe **36** in den Registern AC(03) bis AC(11) gespeichert, wobei auf die Adresse **03** seines niedrigstwertigen Wortes durch den Inhalt von ACP(2) indirekt gezeigt wird; und (3) der ein Wort umfassende Term (A), der Multiplikator, ist im Multiplikatorregister **32** gespeichert. Außerdem wird in dem Kettenregister **17** von [Fig. 3](#) für den acht Wörter umfassenden Term (B), den Multiplikanden, eine Kettenlänge von 8 gesetzt.

[0096] Die Befehlsmnemonik für diese mathematische Operation ist MPAL.A2S.MCND, worin zwei Wörter zum Codieren, ein Wort Operationscode und außerdem eine 1-Wort MCND-Referenz enthalten

sind.

[0097] Der Abschnitt MOND dieses Befehls spezifiziert, dass der Speicher **12** mit einem Multiplikatorregister **32** zu multiplizieren ist und dass das Produkt davon zu einem Wert zu addieren ist, auf den durch das Register ACP(2) gezeigt wird. Der Abschnitt A2S des Befehls gibt den Kettenbetriebszustand der Operation an und spezifiziert, dass das Register der Akkumulatorregisterreihe **36**, auf das durch die Adresse gezeigt wird, die in dem Zeigerregister ACP(2) gespeichert ist, benutzt werden soll, um den neun Wörter umfassenden Term (C), den Summanden, zu speichern und das Endergebnis der Operation $(A) \cdot (B) + (C)$ zu speichern. Der Abschnitt MCND des Befehls spezifiziert einen direkten Speicherverweis (Speicher **12**) auf das niedrigstwertige Wort der 8-Wort-Kette des Terms (B).

[0098] Die Ausführung des oben angegebenen mnemonischen Codes findet in neun Befehlszyklen statt, wobei ein zusätzlicher Zyklus für Multiplikations- oder Barrelshift-Operationen aktiv ist, um von einer N-Wort-Eingabekette eine $N + 1$ -Wort-Ausgabekette zu erzeugen. Für Additionen, Transferoperationen und logische Operationen sind nur acht Befehlszyklen erforderlich, um eine 8-Wort-Kette zu verarbeiten.

[0099] Wenn der Term (A), der Multiplikator, als MPR bestimmt wird, wenn der Term (B), der Multiplikand, als M7-M0 bestimmt wird, wobei sein höchstwertiges Wort M7 im Datenspeicher **12** an der Adresse MCND + 7 gespeichert ist, und wenn der Term (C), der Summand, als A8-A0 bestimmt wird, wobei sein höchstwertiges Wort A8 in der Akkumulatorregisterreihe **36** unter der Adresse **11** gespeichert ist, dann kann die Ausführung dieser neun Befehlszyklen dargestellt werden als:

$$\text{MPR} \times \text{M7}::\text{M6}::\text{M5}::\text{M4}::\text{M3}::\text{M2}::\text{M1}::\text{M0} = \text{P8}::\text{P7}::\text{---} \text{P1}::\text{P0}, \text{A8-A0} = (\text{P8-P0}) + (\text{A8+A0})$$

[0100] Die Aspekte des Aufbaus und der Anordnung, die in **Fig. 5** gezeigt sind und Kettenoperationen des oben angegebenen Typs unterstützen, berücksichtigen, dass (1) der Akkumulator, der die Register AC(0-15) der Akkumulatorregisterreihe **36** umfasst, eine Dualport-Lese/Schreib-Konstruktion mit einem verzögerten Zeigerverweis ist, um der Befehlszyklusverzögerung durch die ALU **80** Rechnung zu tragen; (2) die Akkumulatorregister AC(0-15) und ihre entsprechenden Versatz-Akkumulatorregister AC(16-31) die Adressierung von zwei Operanden mit einem einzigen Zeigerregister ACP(0)-ACP(3) ermöglichen, wobei ein Akkumulatorregister und sein Versatzregister als Eingaberegister oder als Ausgaberegister oder sowohl als Eingaberegister und Ausgaberegister für die ALU-Operanden genutzt werden kann; und (3) das selbsttätige Inkrementieren der

Zeigerregister ACP(0)-ACP(3), um nacheinander die Wörter einer Kette zu lesen, ein temporärer Vorgang ist, der die ACP-Register für einen korrekten zukünftigen Verweis auf das niedrigstwertige Wort der Kette bei zukünftigen Kettenoperationen unverändert lässt.

[0101] Die Fähigkeit der vorliegenden Ausführungsform, eine Kette mit einem erweiterten Vorzeichenwort zu erweitern, ermöglicht die softwaremäßige Implementierung sehr leistungsstarker N-Wort-Ketten $\times$ M-Wort-Ketten-Multiplikationen. Die Programmierung, die erforderlich ist, um diese neuartige Hardware und Architektur vorteilhaft zu nutzen, ist angesichts der obigen Offenlegung im Rahmen der Fähigkeiten eines Fachmanns auf dem Gebiet.

[0102] Die derzeit bevorzugte Ausführungsform entspricht dem MSP58P70 von Texas Instruments Inc. Der MSP58P70 verfügt über einen leistungsstarken Befehlssatz. Die Befehle können ein Bit, ein Byte, ein Wort, oder eine Kette von Wörtern oder Bytes einzeln adressieren. Der Programmspeicher ist siebzehn Bit breit, wobei die gesamte 17-Bit-Breite für die Befehlssatzcodierung genutzt wird. Programme werden von einem internen Programmspeicher aus ausgeführt. Eine Ausführung aus einem externen Speicher heraus ist nicht möglich. Sowohl der Programmspeicher als auch der Datenspeicher des MSP58P70 sind auf interne Blöcke beschränkt und können nicht extern erweitert werden. In der derzeit bevorzugten Ausführungsform ist der Programmspeicher einmalig programmierbarer (OTP-) ROM und auf 32 K 17-Bit-Wörter beschränkt, wovon 2 K für einen internen Testcode reserviert sind und nicht für Anwenderprogramme zugänglich sind. Außerdem ist der Datenspeicher in der bevorzugten Ausführungsform statischer RAM, der auf 1024 17-Bit-Wörter beschränkt ist, wovon sechzehn Bits ein arithmetischer Wert sind, während das siebzehnte Bit als Merker oder Kennzeichnung verwendet wird.

[0103] Dieser Multiplizierer und der Mikroprozessor, in den er integriert ist, sind zur Verwendung in Konsumelektronik wie etwa Telephonanrufbeantwortern entwickelt worden. Ein Blockdiagramm eines Anrufbeantworters, der diese Erfindung verkörpert, ist in **Fig. 4** gezeigt. In dieser Vorrichtung ist der Prozessor **402** mit einer Telephonleitungsschnittstelle **404**, einem Mikrophon **406** und einem Lautsprecher **408** funktional verbunden. Der Mikroprozessor **402** empfängt und sendet Schalldaten mittels der Telephonleitungsschnittstelle **404** über die Telephonleitung **410**. Der Mikroprozessor ist außerdem in der Lage, mittels des Lautsprechers **408** Schall in den Umgebungsbereich zu senden und mittels des Mikrophons **406** Schalldaten aus dem Umgebungsbereich zu empfangen.

[0104] Weitere Merkmale und Einzelheiten, die für eine Verwendung in alternativen Ausführungsformen

der Erfindung, aber nicht notwendig für die praktische Umsetzung der hier offenbarten Ausführungsformen der vorliegenden Erfindung außerdem in Erwägung zu ziehen sind, sind in den folgenden gleichzeitig anhängigen Anmeldungen offenbart:

US-Patent lfd. Nr. 60/090 668 – "Method for Insuring Security of Program Data in One-Time Programmable Memory";

US-Patent lfd. Nr. 09/305 891 – "Variable Word Length Data Memory"; und

US-Patent lfd. Nr. 60/090 671 – "Flexible Accumulator Register File for Use in High Performance Microprocessors".

[0105] All diese werden gemeinsam mit der vorliegenden Anmeldung gehalten und besitzen wirksame Einreichungsdaten, die gleichzeitig zum Datum der vorliegenden Anmeldung sind, und sind hiermit durch die Bezugnahme Bestandteil dieses Patents.

[0106] Wie ein Fachmann auf dem Gebiet erkennt, können die neuartigen Konzepte, die in der vorliegenden Anmeldung beschrieben sind, über einen enorm großen Anwendungsbereich modifiziert und variiert werden, und dementsprechend soll der Geltungsbereich des Gegenstandes nicht durch irgendeinen der gegebenen besonderen, beispielhaften Lösungsansätze eingeschränkt sein.

[0107] Außerdem sollte beachtet werden, dass mit der Zeit tendenziell eine wachsende Zahl von Funktionen in einem einzigen Chip vereint wird. Die offenbarten Ausführungsformen der vorliegenden Erfindung können selbst bei verschiedenen Zuordnungen von Funktionen zu Chips vorteilhaft sein, solange die oben beschriebenen Funktionsprinzipien des Betriebs noch eingehalten werden.

[0108] In weiteren Ausführungen könnten die Schnellübertrags-Addierer in der Multiplizierer-Anordnung durch übertragerhaltende Addierer ersetzt werden, so lange ein 32-Bit-Addierer-Endblock integriert ist, um zum Schluss die Überträge auszugeben.

Patentansprüche

1. Mikroprozessor, mit:
einem Programmzähler (PC), der mit einem Befehls-decodierer funktional verbunden ist;
einer Arithmetik-Logik-Einheit (**31**) (ALU), die mit dem Befehlsdecodierer funktional verbunden ist, um Arithmetikoperationen auszuführen; und
einem Multipliziererblock, der eine Anordnung von Addierern (**30**) besitzt, die mit der ALU funktional verbunden sind und miteinander verbunden sind, um eine Binärzahl mit Einzelwortlänge mit einer weiteren Zahl mit einer Länge von zwei oder mehr Wörtern zu multiplizieren, indem die Zahlen wortweise multipliziert werden; und **dadurch gekennzeichnet**, dass:

die ALU 16 Bit besitzt und die Addierer 17 Bits besitzen und dass er ferner eine Anordnung aus gepaarten Akkumulatoren (AC0, AC16; AC1, AC17; ..., AC15, AC31), die mit der ALU funktional verbunden sind, umfasst.

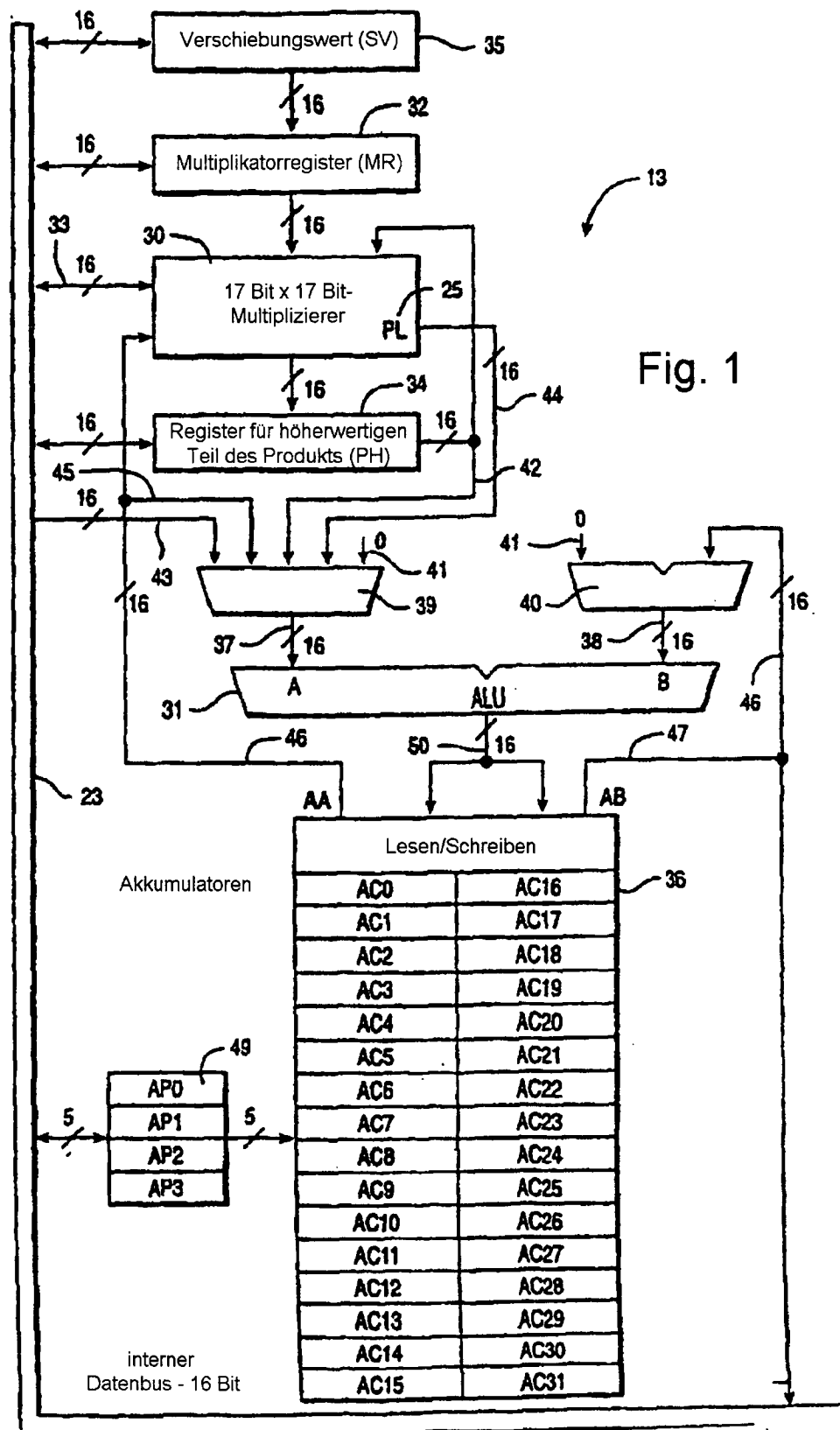
2. Mikroprozessor nach Anspruch 1, bei dem auf gespeicherten Inhalt eines Akkumulatorpaars durch einen einzigen Adresszähler (**49**) Bezug genommen wird.

3. Mikroprozessor nach einem vorhergehenden Anspruch, der einen 4-Bit-nach-16-Bit-Decodierer umfasst, der eine Barrelshift-Funktion in dem Multipliziererblock erleichtert.

4. Telephonanrufbeantworter, mit:
einem Mikroprozessor nach einem vorhergehenden Anspruch;
einer Schnittstelle, die mit einer Telephonleitung funktional verbunden ist, um Nachrichten zu empfangen und zu senden;
einem Mikrofon, das mit dem Steuerabschnitt funktional verbunden ist, um Schall aufzunehmen, der in dem Anrufbeantworter gespeichert wird; und
einem Lautsprecher, der mit dem Steuerabschnitt funktional verbunden ist, um Schall, der in dem Anrufbeantworter gespeichert ist, wiederzugeben.

Es folgen 9 Blatt Zeichnungen

Anhängende Zeichnungen



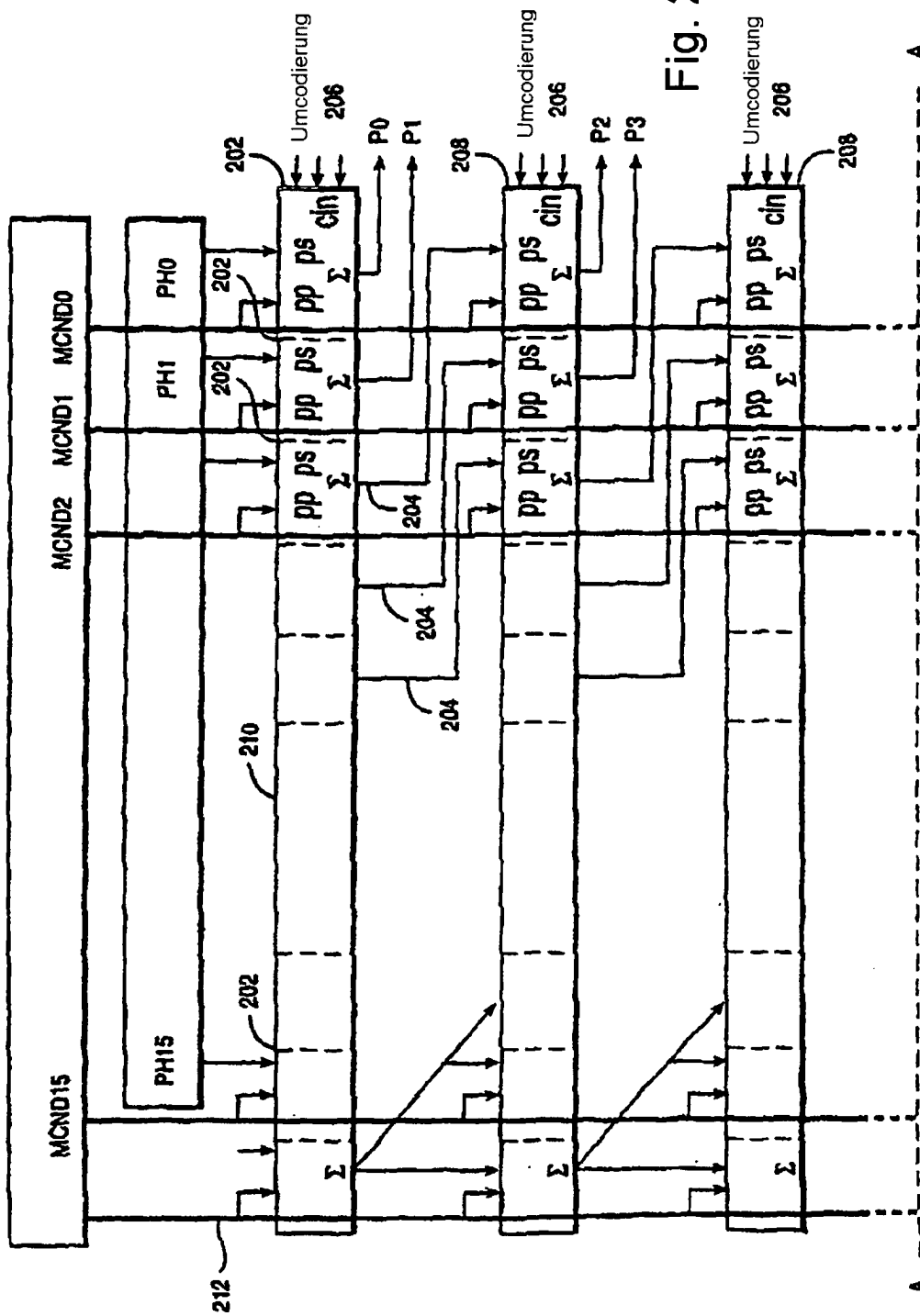


Fig. 2A

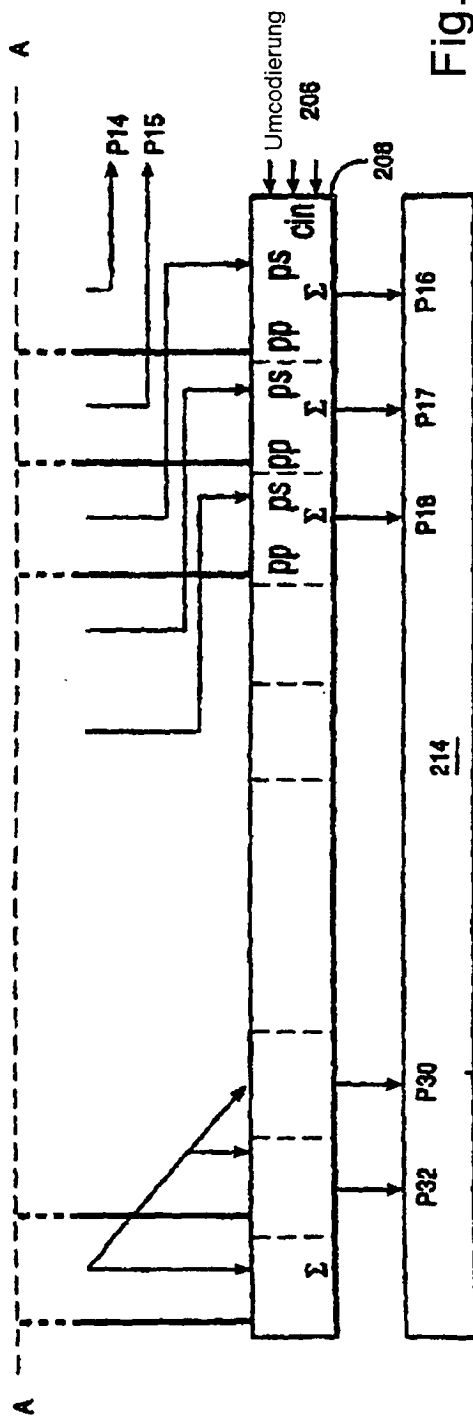


Fig. 2B

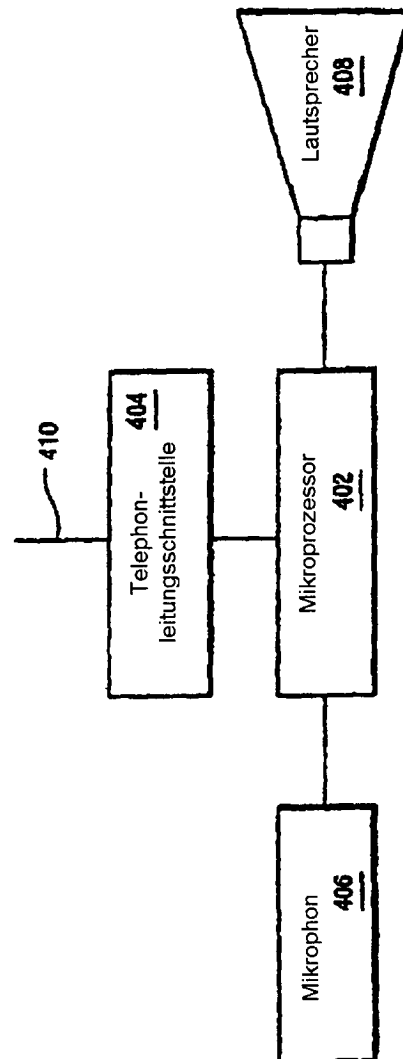


Fig. 4

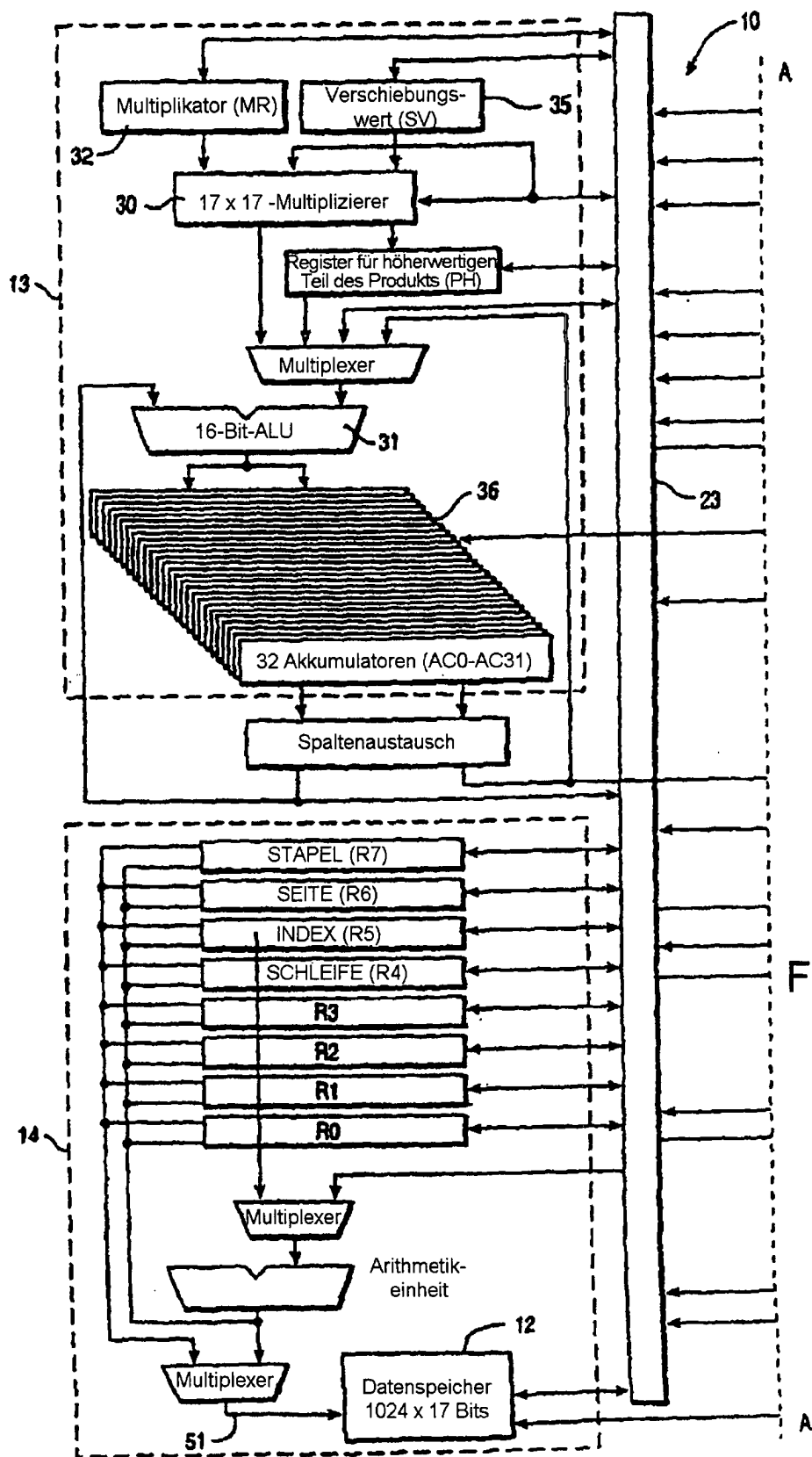
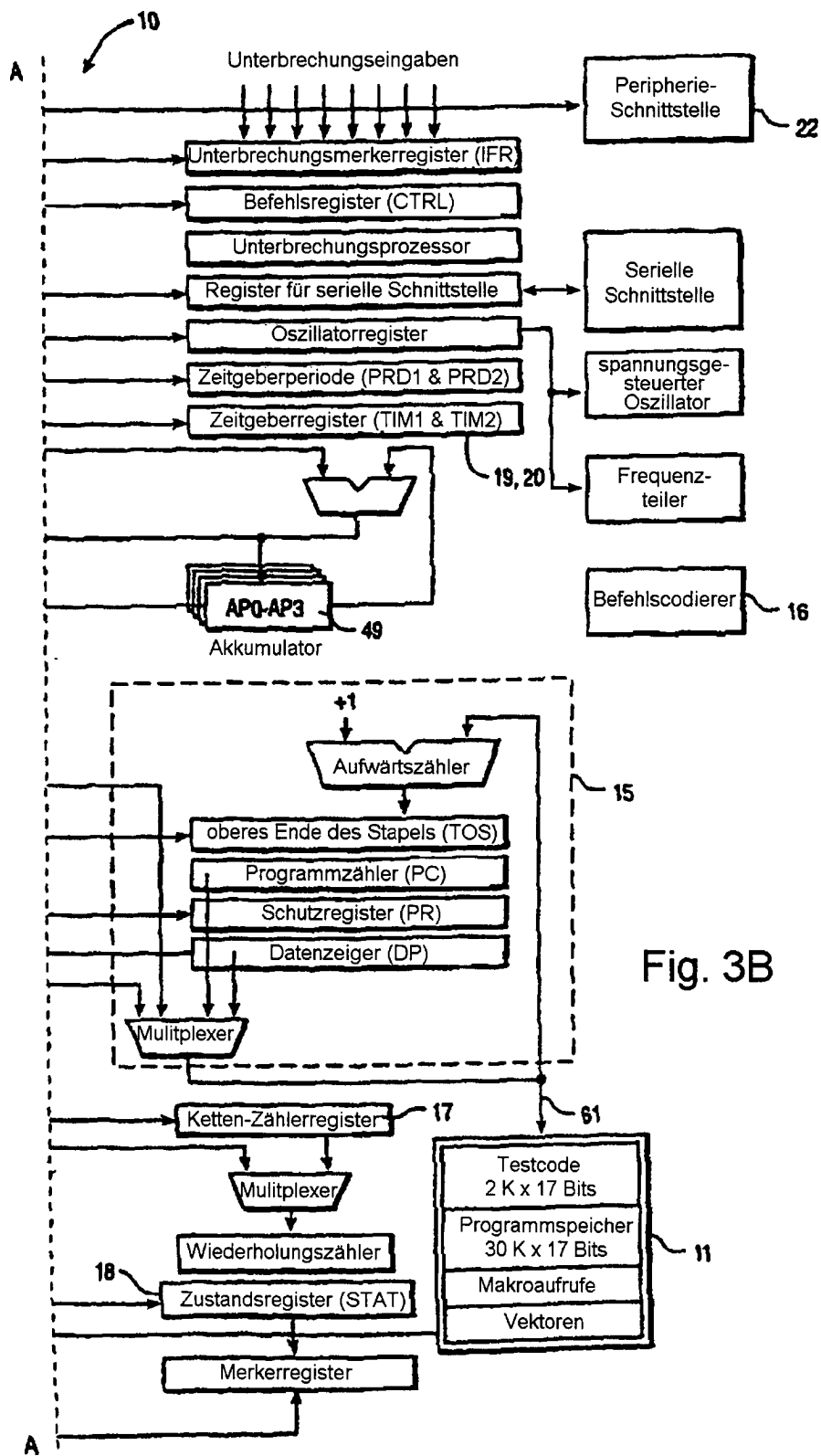


Fig. 3A



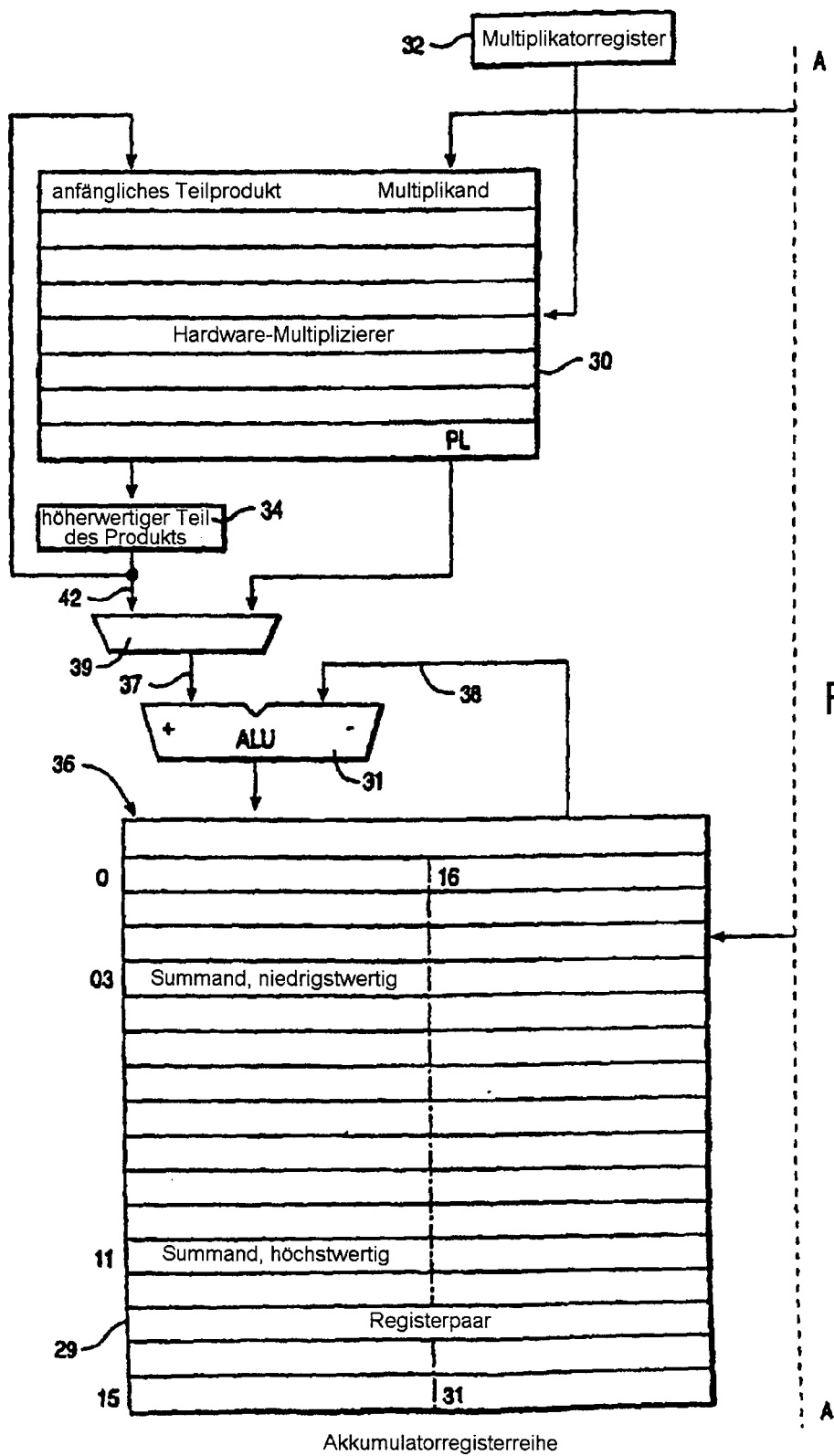


Fig. 5A

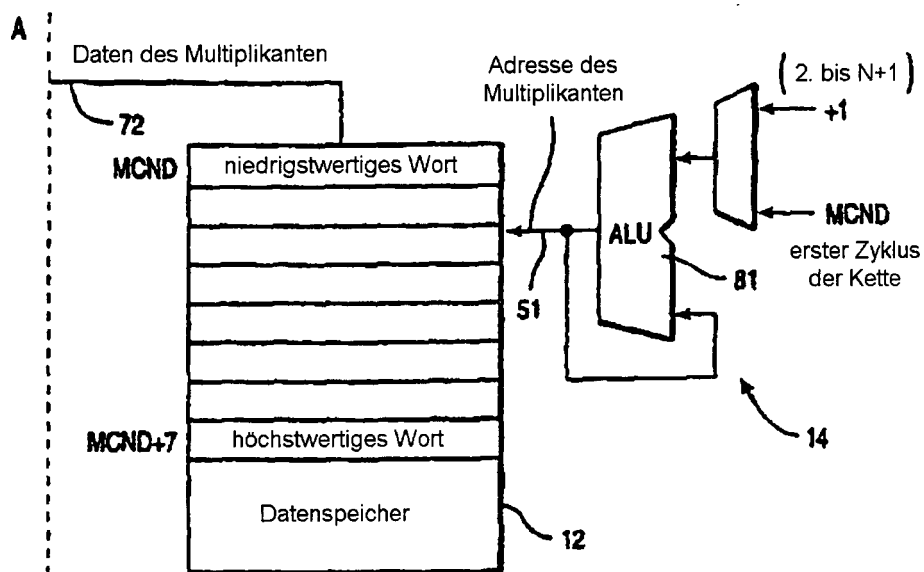
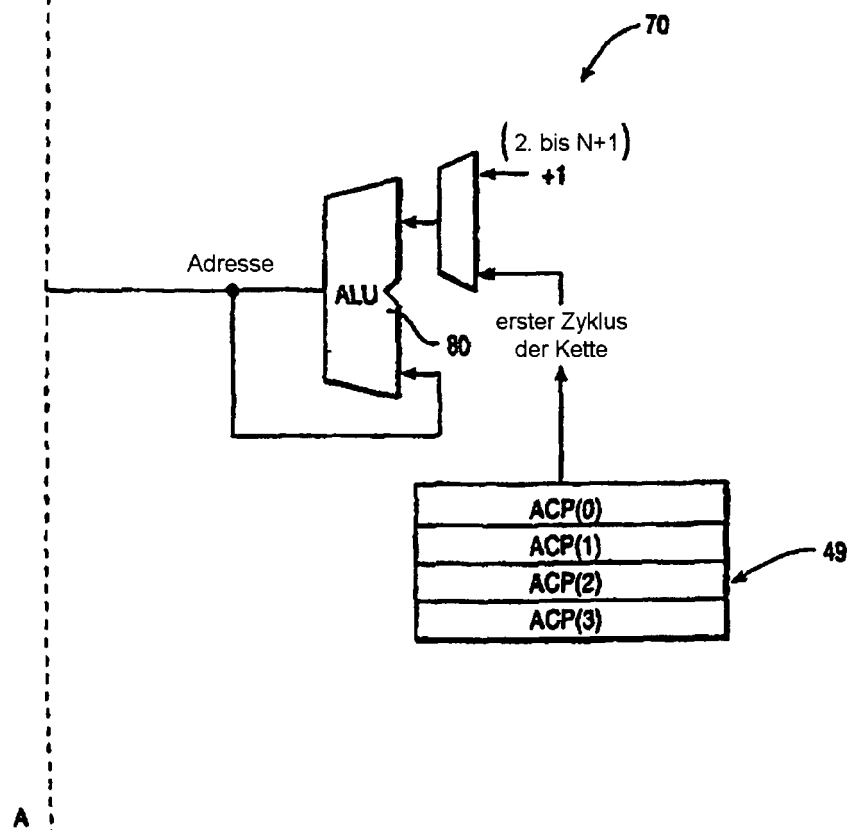


Fig. 5B



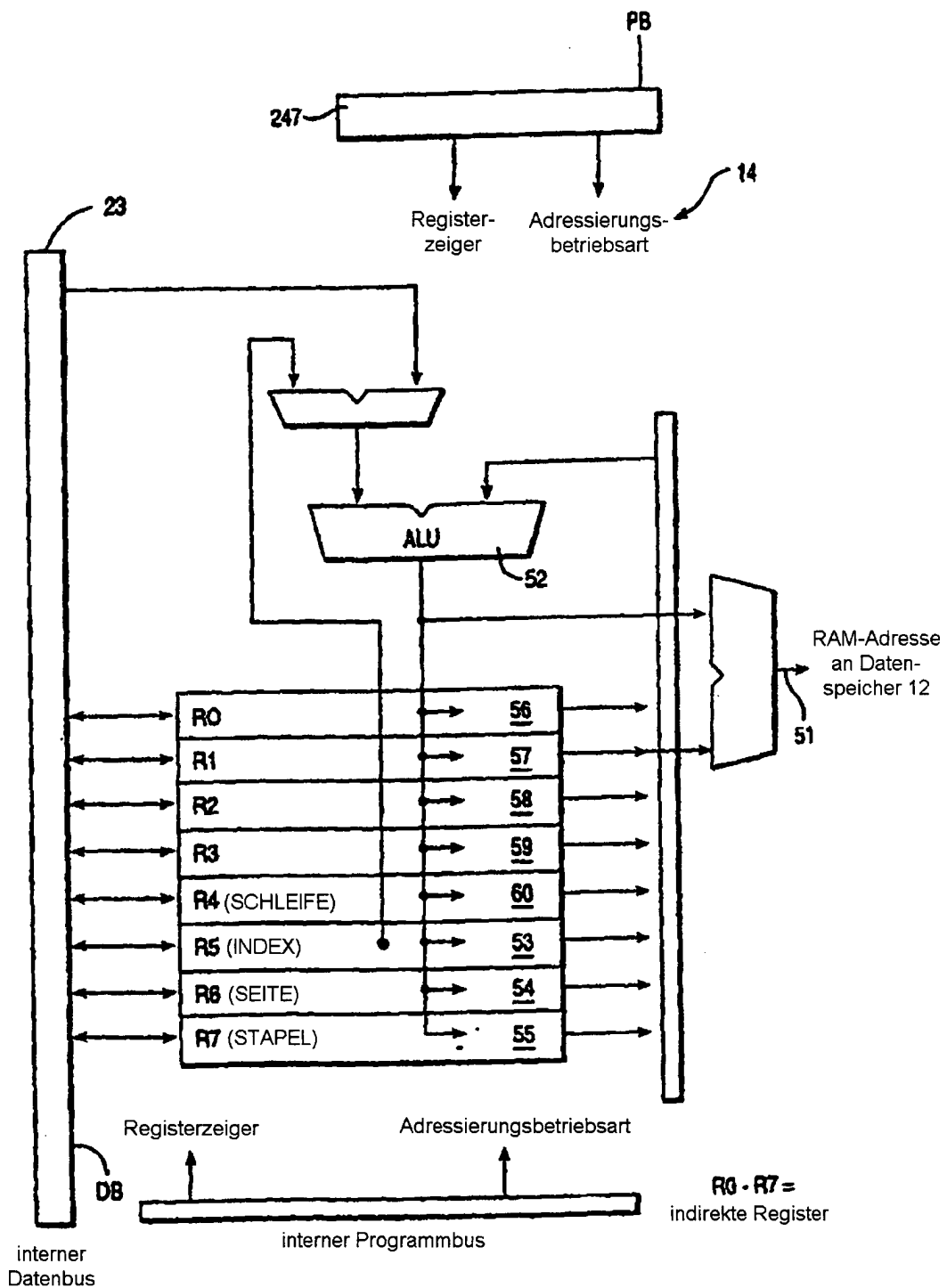


Fig. 6

