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

(10) **DE 696 26 282 T2** 2004.04.08

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 0 829 048 B1**
(21) Deutsches Aktenzeichen: **696 26 282.7**
(86) PCT-Aktenzeichen: **PCT/US96/08256**
(96) Europäisches Aktenzeichen: **96 916 908.5**
(87) PCT-Veröffentlichungs-Nr.: **WO 96/038789**
(86) PCT-Anmeldetag: **31.05.1996**
(87) Veröffentlichungstag der PCT-Anmeldung: **05.12.1996**
(97) Erstveröffentlichung durch das EPA: **18.03.1998**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **19.02.2003**
(47) Veröffentlichungstag im Patentblatt: **08.04.2004**

(51) Int Cl.[7]: **G06F 11/00**

(30) Unionspriorität:
456746 **01.06.1995** **US**

(84) Benannte Vertragsstaaten:
DE, FR, GB

(73) Patentinhaber:
Fujitsu Ltd., Kawasaki, Kanagawa, JP

(74) Vertreter:
W. Seeger und Kollegen, 81369 München

(72) Erfinder:
SAVKAR, Sunil, New York, US; SHEN, Gene, Mountain View, US; SAJJADIAN, Farnad, Sunnyvale, US; SHEBANOW, Mike, Plano, US

(54) Bezeichnung: **PROGRAMMIERBARE VORRICHTUNG UND VERFAHREN ZUM BEFEHLSAUFFANG**

Beschreibung

1. Gebiet der Erfindung

[0001] Diese Erfindung bezieht sich allgemein auf Datenprozessoren. Insbesondere bezieht sich diese Erfindung auf ein System und Verfahren zum Liefern von programmierbaren Hardware-Befehlsfangstellen in einem Mikroprozessor.

2. Beschreibung der Hintergrundtechnik

[0002] Wenn ein Datenprozessor entworfen wird, ist es wichtig, den Entwurf vor einem Tape-Out rigoros zu testen. Im Allgemeinen versucht eine solche Prüfung, Fehler in funktionalen Charakteristiken des Datenprozessors zu identifizieren. Aufgrund von Zeitbeschränkungen ist es jedoch nicht immer durchführbar, alle funktionalen Charakteristiken des Datenprozessorentwurfs vor einem Tape-Out vollständig zu testen. Ein Tape-Out erfolgt, wenn der Entwurf einer physischen Datenbank zum Herstellen der Teile, zum Beispiel des Datenprozessors, freigegeben wird.

[0003] Gelegentlich werden funktionale Fehler im Entwurf des Datenprozessors bis nach einem Tape-Out nicht entdeckt. Unter diesen Umständen ist es teuer, den Entwurf zu korrigieren und neue Datenprozessorteile herzustellen. Funktionale Fehler können auch aufgrund eines Mangels in der Herstellung auftreten, der bestimmte Befehle beeinflusst.

[0004] Eine damit in Zusammenhang stehende Beschränkung tritt mit superskalaren zentralen Verarbeitungseinheiten (CPUs) auf. Viele Softwareanwendungsprogramme sind nicht für die Verwendung in einer superskalaren Umgebung entwickelt. Wenn eine superskalare CPU Befehle in einem solchen Befehlsprogramm ausführt, können einige Befehle nicht in der Art und Weise ablaufen, in der sie entworfen wurden.

[0005] Was benötigt wird, ist ein System und Verfahren zum Liefern einer programmierbaren Hardwarevorrichtung innerhalb einer CPU, die gestattet, dass mehrere Befehle abgefangen (eng. trap) werden, bevor sie ausgeführt werden. Die Befehle, die abgefangen werden sollen, sind programmierbar, um Flexibilität während einer CPU-Programmfehlerbeseitigung zu liefern und sicherzustellen, dass eine Vielzahl von Anwendungsprogrammen durch die CPU geeignet ausgeführt werden kann. Das System muss auch ein Mittel vorsehen, um zu gestatten, dass ein abgefangener Befehl emuliert und/oder seriell ausgeführt wird.

[0006] EP-A-0 271 910 offenbart einen Einchip-Mikroprozessor mit einer Unterbrechungsfunktion. Unterbrechungspunktregister sind vorgesehen, um vom Benutzer festgelegte Adressen oder Datenwerte zu halten. Komparatoren sind vorgesehen, um die vom Benutzer festgelegten Adressen/Datenwerte in den Unterbrechungspunktregistern mit Adressen, auf die zugegriffen werden soll, oder Daten, die verarbeitet werden sollen, zu vergleichen. Falls ein Vergleich von Unterbrechungspunktregistern eine Übereinstimmung liefert, kann eine Unterbrechungsfunktion ausgeführt werden. Vorgesehen sind Mittel, um eine Historie von Vergleichen von Unterbrechungspunktregistern zu halten, und eine Bestimmung, ob eine Unterbrechungsbedingung erfüllt ist, kann nicht nur von einem Vergleich von Unterbrechungspunktregistern, sondern auch der Historie anderer Unterbrechungspunktregister abhängen.

ZUSAMMENFASSUNG DER ERFINDUNG

[0007] Gemäß der vorliegenden Erfindung werden ein Computerverarbeitungssystem nach Anspruch 1 und ein Verfahren nach Anspruch 4 geschaffen.

[0008] Die Erfindung stellt ein System und Verfahren zur Verfügung, das eine programmierbare Hardwarevorrichtung innerhalb einer CPU vorsieht. Die programmierbare Hardwarevorrichtung gestattet, dass mehrere Befehle abgefangen werden, bevor sie ausgeführt werden. Die Befehle, die abgefangen werden sollen, sind programmierbar, um Flexibilität während einer CPU-Programmfehlerbeseitigung zu liefern und sicherzustellen, dass eine Vielzahl von Anwendungsprogrammen durch die CPU geeignet ausgeführt werden kann. Das System muss auch ein Mittel vorsehen, um zu gestatten, dass ein abgefangener Befehl emuliert und/oder seriell ausgeführt wird.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

[0009] **Fig.** 1 ist eine Veranschaulichung einer Computerumgebung, in der eine bevorzugte Ausführungsform der vorliegenden Erfindung liegt.

[0010] **Fig.** 2 ist eine ausführlichere Veranschaulichung einer superskalaren zentralen Verarbeitungseinheit der bevorzugten Ausführungsform der vorliegenden Erfindung.

[0011] **Fig.** 3 ist eine ausführlichere Veranschaulichung einer programmierbaren Befehlsfangstelleneinheit gemäß der bevorzugten Ausführungsform der vorliegenden Erfindung.

[0012] **Fig.** 4 ist eine ausführlichere Veranschaulichung einer programmierbaren Befehlsfangstelleneinheit gemäß der bevorzugten Ausführungsform der vorliegenden Erfindung.

[0013] **Fig.** 5 ist eine ausführlichere Veranschaulichung eines Befehlsfangstellenregister-Wertes (ITRV) und einer Befehlsfangstellenregister-Maske (ITRM) gemäß der bevorzugten Ausführungsform der vorliegenden Erfindung.

[0014] **Fig.** 6 ist eine ausführlichere Veranschaulichung eines Bitregisters gemäß der bevorzugten Ausführungsform der vorliegenden Erfindung.

[0015] **Fig.** 7 ist ein Flussdiagramm, das das Verfahren zum Abfangen eines Befehls gemäß der bevorzugten Ausführungsform der vorliegenden Erfindung beschreibt.

[0016] **Fig.** 8 ist ein Flussdiagramm, das das Verfahren zum Testen eines Befehls gemäß der bevorzugten Ausführungsform der vorliegenden Erfindung beschreibt.

AUSFÜHRLICHE BESCHREIBUNG DER BEVORZUGTEN AUSFÜHRUNGSFORMEN

[0017] Eine bevorzugte Ausführungsform der vorliegenden Erfindung wird nun mit Verweis auf die Figuren beschrieben, in denen gleiche Referenzzahlen identische oder funktional ähnliche Elemente angeben. In den Figuren entspricht die äußerste linke Ziffer jeder Referenzzahl auch der Figur, in der die Referenznummer zuerst verwendet wird.

[0018] **Fig.** 1 ist eine Veranschaulichung einer Computerumgebung, in der eine bevorzugte Ausführungsform der vorliegenden Erfindung liegt. Eine herkömmliche Speichereinheit **104** und herkömmliche Eingabe/Ausgabe-(I/O)-Vorrichtungen **106** kommunizieren mit einem Datenprozessor **102**. In der bevorzugten Ausführungsform ist der Prozessor ein von HaL Computer Systems, Campbell, Kalifornien, entwickelter HaL-R1-Prozessor. Der HaL-R1-Prozessor implementiert die Architektur des 64-Bit-Befehlssatzes SPARC V9, die von SPARC International, Inc., Menlo Park, Kalifornien, entwickelt wurde. Der Prozessor **102** enthält eine superskalare CPU **108**, eine Speicherverwaltungseinheit (MMU) und vier Cache-Speichereinheiten **110**, **112**.

[0019] Die Cache-Speichereinheiten **110**, **112** umfassen vorzugsweise zwei herkömmliche Daten-Caches mit 64 Kilobyte (KB) und zwei herkömmliche Befehls-Caches mit 64 KB. Die Daten-Caches **110** liefern Daten an die superskalare CPU **108** und die MMU **114**. Beide Sätze von Cache-Speichereinheiten **110**, **112** sind virtuell indexiert und markiert. Jeder Cache-Chip **110**, **112** enthält 64 KB Datenspeicher, organisiert als vier Sätze. Jeder 64 KB-Cache-Chip **110**, **112** kann zwei unabhängige Anforderungen von der superskalaren CPU **108** bedienen. Die Schnittstelle zwischen der superskalaren CPU **108** und den Caches **110**, **112** ist insofern nicht blockierend, als die superskalare CPU **108** auf den Cache **110**, **112** zu der gleichen Zeit zugreifen kann, zu der eine Cache-Leitung nachgefüllt oder freigeräumt wird. Ein zusätzliches Detail, das die Cache-Speichereinheiten **110**, **112** betrifft, ist in Chen, Chien, Y. Lu, A. Wond; The Microarchitecture of the HaL Cache, Compcon Proceedings 1995, offenbart.

[0020] Die Funktion der MMU **114** umfasst ein Steuern der Speicherverwaltung und Datenkohärenz, eine Schnittstellenbildung mit dem Speicher **104** und den I/O-Vorrichtungen **106** und eine Fehlerbehandlung für den Prozessor **102**. Die MMU **114** enthält drei Ebenen von Adressräumen: (1) einen virtuellen Adressraum für den Prozessor **102**, (2) einen logischen Adressraum für die I/O-Vorrichtungen **106** und einen diagnostischen Prozessor und (3) einen physikalischen Adressraum für den Speicher **104**. Diese hierarchischen Adressräume liefern einen effizienten Mechanismus zum Verwalten des großen Adressraums. Eine ausführlichere Beschreibung der MMU **114**, die in der bevorzugten Ausführungsform verwendet wird, ist in David Chang, D. Lyon, C. Chen, L. Peng, M. Massoumi, M. Hakimi, S. Iyengar, E. Li, R. Remedios, Microarchitecture of HaL's Memory Management Unit, Compcon Proceedings 1995, beschrieben.

[0021] **Fig.** 2 ist eine ausführlichere Veranschaulichung einer superskalaren CPU **108** der bevorzugten Ausführungsform der vorliegenden Erfindung. Der Prozessor **102** implementiert eine vierstufige Pipeline für Festkommabefehle und eine sechsstufige Pipeline für "LADEN"-Befehle. Die superskalare CPU **108** enthält eine Befehlsabrufeinheit **202**, eine programmierbare Befehlsfangstelleneinheit **204**, eine Befehlserteilungseinheit **206**, eine Befehlsausführungseinheit **218**, eine Präziser-Zustand-Einheit **220**, eine Tabelle **208** der Verzweigungshistorie, eine Tabelle **210** zur Vorhersage eines Rücksprungs, einen Vorabrufpuffer **212**, einen Cache **214** für aufgezeichnete Befehle und eine Verzweigungseinheit **216**.

[0022] Die Abrufeinheit **202** fordert vier Befehle von entweder dem Befehls-Cache **112** mit 64 KB außerhalb des Chip, dargestellt als Datenleitungen **201** in **Fig.** 2, einem Vorabrufpuffer **212** oder einem Cache **214** für aufgezeichnete Befehle an und empfängt diese. Vier Befehle werden dann an eine programmierbare Befehlsfangstelleneinheit **204** gesendet, um zu bestimmen, ob irgendeiner der Befehle abgefangen werden sollte. Die programmierbare Befehlsfangstelleneinheit **204** wird im Folgenden mit Verweis auf **Fig.** 3–8 ausführlicher beschrieben. Die Tabelle **208** der Verzweigungshistorie enthält mehrere Zähler, die verwendet werden, um die Richtung von Ausführungszweigen vorherzusagen. Die Tabelle **210** zur Vorhersage eines Rücksprungs wird verwendet, um die Rücksprungadresse ausgeführter Befehle vorherzusagen. Die Verzweigungseinheit **216** verwendet Informationen von der Erteilungseinheit **206**, der Tabelle **208** der Verzweigungshistorie und der Tabelle **210** zur Vorhersage des Rücksprungs, um zu bestimmen, ob ein Steuertransferbefehl innerhalb des aktuellen Erteilungsfensters liegt, und modifiziert ferner den Programmzähler, falls der Steuertransferbefehl genommen wird. Die Erteilungseinheit **206** bestimmt die verfügbaren Computer-Ressourcen und Erteilungsbeschränkungen, z. B. sollte ein Befehl synchronisiert werden. Alle Befehle werden durch die Erteilungseinheit **206** in der Reihenfolge erteilt, in der sie in einem traditionellen seriellen Prozessor erteilt worden wären, und werden an die Befehlsausführungseinheit **218** gesendet. Die Befehlsausführungseinheit **218** führt die Befehle, obgleich nicht notwendigerweise, in der Reihenfolge aus, in der sie erteilt wurden. Die Befehlsausführungseinheit **218** gibt die Ergebnisse der Befehlsausführung auf einem Datenbus **222** aus. Eine ausführlichere Beschreibung der superskalaren CPU **108** der bevorzugten Ausführungsform ist

in Niteen Patkar, A. Katsuno, S. Li, T. Maruyama, S. Savkar, M. Simone, G. Shen, R. Swami, D. Tovey, Microarchitecture of the HaL PM1 CPU, Compcon Proceedings 1995, beschrieben. Die vorliegende Erfindung liegt in der programmierbaren Befehlsfangstelleneinheit **204**, die hierin beschrieben ist.

[0023] **Fig.** 3 ist eine ausführlichere Veranschaulichung einer programmierbaren Befehlsfangstelleneinheit **204** gemäß der bevorzugten Ausführungsform der vorliegenden Erfindung. Die programmierbare Befehlsfangstelleneinheit **204** empfängt vorzugsweise vier Befehle von der Abrufeinheit **202**. Die Befehle sind in **Fig.** 3 als INST00, INST01, INST10, INST11 dargestellt. Jeder Befehl wird an eine Befehlsfangstellen-Logikeinheit (ITLU) **302** gesendet. Die Befehlsfangstellen-Logikeinheit empfängt auch Signale von vorzugsweise vier Befehlsfangstellenregistern (ITRV) **304**, vier Befehlsfangstellenregister-Masken (ITRM) **306** und vier Steuerregistern (CR) **308**. Eine ausführlichere Beschreibung der Befehlsfangstellen-Logikeinheit **302**, des ITRV **304**, der ITRM **306** und des CR **308** wird im Folgenden bezüglich der **Fig.** 4–8 dargelegt.

[0024] **Fig.** 4 ist eine ausführlichere Veranschaulichung einer programmierbaren Befehlsfangstelleneinheit gemäß der bevorzugten Ausführungsform der vorliegenden Erfindung. Die Befehlsfangstellen-Logikeinheit **302** enthält vorzugsweise vier Exklusiv-NOR-Logikvorrichtungen **404A–D**, wo ein Exklusiv-NOR mit je einem ITRV **304** verbunden ist. Jede Exklusiv-NOR-Vorrichtung **404** vergleicht jedes invertierte Bit des Befehls mit einem entsprechenden Bit von dem zugeordneten ITRV, z. B. ITRV0 **410**, und für jedes Bit gibt die NOR-Vorrichtung **404** einen wert aus, der angibt, ob eine Übereinstimmung aufgetreten ist. Es wird ins Auge gefasst, dass andere Techniken verwendet werden können, um jedes Bit des Befehls INST00 mit jedem Bit in einem ITRV **304** zu vergleichen. Das Verfahren zum Durchführen dieses Vergleichs wird im Folgenden bezüglich der **Fig.** 7–8 ausführlicher beschrieben.

[0025] Die Befehlsfangstellen-Logikeinheit **302** enthält vorzugsweise vier Vorrichtungen **406A–D**, die jedes von einer zugeordneten Exklusiv-NOR-Vorrichtung **404** ausgegebene Bit mit einem Bitwert in einer ITRM **306** vergleichen. Jede ITRM **306** enthält mehrere Bits, die einen Wert "Beachten" oder "Nicht Beachten" enthalten. Jede der vier Vorrichtungen **406A–D** vergleicht ein Bit, das von ihrer zugeordneten Exklusiv-NOR-Vorrichtung ausgegeben wird, und ein Bit in ihrer zugeordneten ITRM **306**. Jede Vorrichtung **406** gibt einen Bitwert aus, der darstellt, ob alle Befehlsbits zu dem zugeordneten ITRV-ITRM-Registerpaar passen. Jede Vorrichtung **406** ist vorzugsweise eine AO32x2-Vorrichtung. Diese Vorrichtung hat zwei Sätze von 32-Bit-Eingängen, einen Satz, der mit jeder Bitleitung des Ausgangs eines zugeordneten Exklusiv-NOR **404** gekoppelt ist, während der andere Satz mit der zugeordneten ITRM **306** gekoppelt ist. Die AO32x2 enthält 32 logische UND-Gatter, deren Ausgaben in ein logisches ODER-Gatter eingegeben werden. Das Verfahren zum Durchführen dieses Vergleichs wird im Folgenden bezüglich der **Fig.** 7–8 ausführlicher beschrieben.

[0026] Die CRs **308** sind mit einem einzelnen ITRV-ITRM-Paar verbunden. CR0 ist zum Beispiel mit ITRV0 und ITRM0 verbunden. Jedes CR enthält Daten, die die Aktion darstellen, die durchzuführen ist, falls der Befehl zu seinem zugeordneten ITRV-ITRM-Registerpaar passt. Obgleich jedes Steuerregister eine beliebige Anzahl Bits enthalten kann, enthält jedes CR 308 in der bevorzugten Ausführungsform zwei Bits. Ein Bit repräsentiert eine Synchronisierungsaktion, das andere Bit repräsentiert eine Befehlsfangstelle. Das heißt, falls die Bits in einem Steuerregister "00" sind und falls der Befehl zu dem zugeordneten ITRV-ITRM-Registerpaar **304**, **306** passt, wird dann keine spezielle Aktion unternommen. Falls die Bits in einem Steuerregister "01" sind und falls der Befehl zum zugeordneten ITRV-ITRM-Registerpaar **304**, **306** passt, wird dann die programmierbare Befehlsfangstelleneinheit **204** ein Signal abgeben, das den zusammenpassenden Befehl synchron zur Ausführung gelangen lässt, d. h. alle Befehle, die vor dem passenden Befehl erteilt wurden, werden eine Ausführung beendet haben, und die Maschine wird in einem festgeschriebenen, d. h. bekannten Zustand sein, bevor der passende Befehl zur Ausführung gelangt. Außerdem wird ein Synchronisierungsbefehl erteilt, ausgeführt, abgeschlossen und festgeschrieben, bevor irgendwelche nachfolgenden Befehle erteilt werden. Falls die Bits in einem Steuerregister "10" sind und falls der Befehl zu dem zugeordneten ITRV-ITRM-Registerpaar **304**, **306** passt, wird dann die programmierbare Befehlsfangstelleneinheit **204** eine nicht-synchronisierende Befehlserteilungsfangstelle erzeugen, d. h. der Befehl wird die Semantik der Befehlsfangstelle übernehmen. Falls die Bits in einem Steuerregister "11" sind und falls der Befehl mit dem zugeordneten ITRV-ITRM-Registerpaar **304**, **306** zusammenpasst, wird dann die programmierbare Befehlsfangstelleneinheit **204** eine synchronisierende Befehlserteilungsfangstelle erzeugen, d. h. der Befehl wird abgefangen und synchron ausgeführt.

[0027] Die Befehlsfangstellen-Logikeinheit **302** enthält eine Vorrichtung zum Erzeugen der Werte, die in dem CR **308** gespeichert werden, verbunden mit einem ITRV-ITRM-Registerpaar **304**, **306**, das zum empfangenen Befehl, zum Beispiel INST00, passt. Eine Vielzahl von Techniken kann genutzt werden, um diesen Wert zu erzeugen. Vorzugsweise empfangen zwei Sätze von vier logischen ODER-Gattern (eines jedem CR **308** zugeordnet) eine Eingabe von CR **308**. Jedes ODER-Gatter des ersten Satzes von ODER-Gattern empfängt zwei Eingaben: (1) das erste Bit eines CR **308**, z. B. CR0 und (2) die Ausgabe eines Bit von der mit dem CR **308** verbundenen AO32x2. Jedes ODER-Gatter des zweiten Satzes von ODER-Gattern empfängt ähnlich zwei Eingaben:

(1) das zweite Bit eines CR **308**, zum Beispiel CR0 und (2) ein von der der CR **308** zugeordneten AO32x2 ausgegebenes Bit. Die Ausgabe aller ODER-Gatter für jeden Satz von ODER-Gattern wird an ein logisches NAND-Gatter **408A–B** geliefert. Das Verfahren zum Erzeugen der Ausgaben der Befehlsfangstellen-Logikeinheit **302** wird im Folgenden bezüglich der **Fig.** 7–8 ausführlicher beschrieben.

[0028] Die Werte in den ITRVs **304**, den ITRMs **306** und den CRs **308** sind programmierbar. Diese Werte können von einem Nutzer programmiert werden, um Befehle abzufangen und zu veranlassen, dass bestimmte Befehle synchron ausgeführt werden, wie oben beschrieben wurde. Das Merkmal einer Programmierbarkeit der vorliegenden Erfindung liefert eine effiziente Technik zum Abfangen und Synchronisieren eines Befehls, um funktionale Fehler in Prozessorbefehle zu korrigieren und den korrekten Betrieb von Anwendungsprogrammen sicherzustellen, indem bestimmte Befehle synchron ausgeführt werden, d. h. diese Befehle in dem ITRV **304**, ITRM **306** gespeichert werden. Eine Vielzahl von Techniken kann verwendet werden, um die ITRVs **304**, die ITRMs **306** und CRs **308** (die zusammen als programmierbare Register bezeichnet werden) zu programmieren. Die programmierbaren Register können zum Beispiel mit einem Datenbus und mit einer Steuerlogik gekoppelt sein, die gestattet, dass die programmierbaren Register vom Datenbus Daten empfangen. In der bevorzugten Ausführungsform werden die Daten während der Urladen-Operation des Computers "eingescannt". Die programmierbaren Register werden unter Verwendung einer Technik ähnlich Schieberegistern programmiert.

[0029] **Fig.** 5 ist eine ausführlichere Veranschaulichung eines ITRV **304** und ITRM **306** gemäß der bevorzugten Ausführungsform der vorliegenden Erfindung. Jedes programmierbare Register **410** enthält mehrere Bitregister **502**. Jedes ITRV **304** und jede ITRM **306** enthalten vorzugsweise 27 Bitregister **502**, während jedes CR **306** 2 Bitregister **502** enthält. **Fig.** 5 ist eine repräsentative ITRV **304** oder ITRM **306**. CRs **308** arbeiten in der gleichen Weise mit der Ausnahme, dass jedes der CR **308** im Gegensatz zu 27 Bitregistern **502** wie oben diskutiert vorzugsweise zwei Bitregister **502** enthält. Jedes programmierbare Register **410** empfängt ein Scan-Takt-(SC)-Signal, ein Scan-Freigabe-(SE)-Signal und ein Scan-Eingabe(SI)-Signal und erzeugt ein Scan-Ausgabe-(SO)-Signal und ein Q-Ausgangssignal.

[0030] **Fig.** 6 ist eine ausführlichere Veranschaulichung eines Bitregisters **502** gemäß der bevorzugten Ausführungsform der vorliegenden Erfindung. Ein einen Bitwert enthaltendes Signal wird auf der SI-Leitung empfangen. Wenn die SC- und SE-Signale hoch sind, d. h. das Signal einen binären Wert "1" repräsentiert, erzeugt das logische UND-Gatter **602** ein hohes Signal. Dieses Signal wird mit einem Gate eines Transistors **604** gekoppelt. Als Folge leitet der Transistor **604**, wodurch ermöglicht wird, dass das Signal auf der SI-Leitung durch den Transistor **604** gelangt und durch zwei Inverter **606** seinen logischen Wert beibehält. Wenn das SC von H auf L schaltet, während das SE-Signal hoch bleibt, schaltet der Transistor **604** ab und das logische UND-Gatter **608** erzeugt ein hohes Signal. Das UND-Gatter **608** ist mit dem Gate eines Transistors **612** gekoppelt. Als Folge leitet der Transistor **612**, und das invertierte SI-Signal wird wieder durch einen der Inverter **610** invertiert und auf der SO-Leitung abgegeben. Das Signal wird ebenfalls auf der Q-Leitung abgegeben, die mit entweder einer Vorrichtung **404A** gekoppelt ist, falls das Bitregister Teil eines ITRV **304** ist, einem Inverter und einer Vorrichtung **406A**, falls das Bitregister Teil einer ITRM **306** ist, oder einem Inverter und einem ODER-Gatter, falls das Bitregister Teil eines CR **308** ist. Die SO-Leitung ist mit der SI-Leitung des nächsten Bitregisters gekoppelt. Die SO-Leitung des letzten Bitregisters von ITRV **304** ist mit dem ersten Bitregister der ersten ITRM **306** gekoppelt. Das letzte Bitregister der ITRM **306** ist mit dem ersten Bitregister der ersten CR **308** gekoppelt. Während eines Prozesses zur Initialisierung eines Computers werden die Werte programmierbarer Register repräsentierende Prozessdaten systematisch in die erste SI-Leitung **412** eingegeben und durch alle Bitregister geschoben, bis jedes Bitregister die gewünschten Werte enthält, d. h. die Bitregisterwerte werden eingescannt.

[0031] Das Verfahren der vorliegenden Erfindung wird nun beschrieben. **Fig.** 7 ist ein Flussdiagramm, das das Verfahren zum Abfangen eines Befehls gemäß der bevorzugten Ausführungsform der vorliegenden Erfindung beschreibt. Zu Anfang werden die ITRVs **304**, ITRMs **306** und die CRs **308** programmiert. Wie oben beschrieben wurde, können diese programmierbaren Register unter Verwendung eines Datenbusses und von Steuersignalen individuell programmiert werden **704**, **706**, **708**. Die programmierbaren Register werden vorzugsweise während einer Initialisierung eines Computers unter Verwendung der Einscan-, das heißt Bit-Schiebe-Technik programmiert, die oben mit Verweis auf **Fig.** 5–6 beschrieben wurde. Nachdem die programmierbaren Register programmiert sind **702**, werden Befehle getestet **710**. Das Verfahren zum Testen von Befehlen wird im Folgenden bezüglich **Fig.** 8 dargelegt.

[0032] **Fig.** 8 ist ein Flussdiagramm, das das Verfahren zum Testen eines Befehls gemäß der bevorzugten Ausführungsform der vorliegenden Erfindung beschreibt. Von der Abrufeinheit **202** werden vorzugsweise vier Befehle durch die programmierbare Befehlsfangstelleneinheit **204** wie in **Fig.** 2 veranschaulicht empfangen, **802**. Jeder Befehl wird an eine separate Befehlsfangstellen-Logikeinheit **302** gesendet. Ein Befehl INST00 wird zum Beispiel an die Befehlsfangstellen-Logikeinheit **302** gesendet. Die Befehlsfangstellen-Logikeinheit **302** invertiert jedes Bit in dem Befehl unter Verwendung eines Inverters **402**. Die invertierten Befehlsbits werden von vier

Exklusiv-NOR-Vorrichtungen **404A–D** empfangen, die oben mit Verweis auf **Fig.** 4 beschrieben wurden. Jede Exklusiv-NOR-Vorrichtung **404** vergleicht, **804**, jedes invertierte Befehlsbit mit einem entsprechenden Bit in einem der ITRVs **304**, die vorher programmiert wurde. Zum Beispiel vergleicht die Exklusiv-NOR-Vorrichtung **404A** ein Befehlsbit 0 mit einem Bitregister 0 von ITRV0, wie in **Fig.** 4–6 veranschaulicht ist. Falls das invertierte Befehlsbit zum zugeordneten Bit in einem der ITRVs **304** passt, erzeugt dann das exklusive NOR ein Signal, das eine logische "1" repräsentiert. Ansonsten erzeugt das exklusive NOR ein Signal, das eine logische "0" repräsentiert. Da die Exklusiv-NOR-Vorrichtung das zugeordnete ITRV-Bitregister **502** und ein invertiertes Bit des Befehls vergleicht, passt, falls das exklusive NOR ein eine logische "0" repräsentierendes Signal erzeugt, dann das ITRV-Bit zum Befehlsbit. Jedes invertierte Bit des Befehls INST00 wird mit jedem zugeordneten Bit jedes ITRV **304** verglichen. Das Ergebnis dieser Vergleiche sind vorzugsweise vier 27-Bit-Signale, die das Ergebnis des Exklusiv-NOR-Vergleichs der Vorrichtungen **404A–D** repräsentieren.

[0033] Jedes Bit in jedem Satz von 27-Bit-Signalen wird vorzugsweise mit einem Bitregister **502** in einem zugeordneten ITRM **306** unter Verwendung einer AO32x2-Vorrichtung **406A–D** verglichen, **806**. Jede AO32x2-Vorrichtung **406A–D** besteht aus mindestens 27 logischen UND-Gattern, deren Ausgaben in ein logisches ODER-Gatter eingegeben werden. Das niedrigstwertige Bit oder das Null-Bit, das von einer Vorrichtung **404A** ausgegeben wird, wird z. B. zusammen mit dem invertierten Null-Bit-Register von ITRM0 in ein UND-Gatter eingegeben. Die ITRMs **306** sind eine Maske. Manchmal ist es wünschenswert, jeden Befehl mit einem Satz verwandter Befehlsmuster zu vergleichen. Es kann zum Beispiel wünschenswert sein, alle Befehle mit einer bestimmten Kombination von Bits 5–9 passend zuzuordnen oder zu vergleichen. In dieser Situation sollten die verbleibenden Bits, zum Beispiel Bits 0–4 und 10–26 maskiert werden, weil ihr Wert für diesen besonderen Abgleich irrelevant ist. Dieser Abgleich wird vollzogen, indem im richtigen ITRM-Bitregister ein Wert "Nicht Beachten" platziert wird. In der vorliegenden Erfindung ist der Wert "Nicht Beachten" eine logische "1". Wie in **Fig.** 4 veranschaulicht ist, werden die ITRM-Werte invertiert, bevor sie von der AO32x2 empfangen werden. Falls ein Bit von der Vorrichtung **404A** eine Null ist, weil das Befehlsbit mit dem zugeordneten ITRV-Bitregister zusammenpasste, wird dementsprechend die Ausgabe des zugeordneten UND-Gatters in der AO32x2 **406A** Null sein. Falls ein Bit von der Vorrichtung **404A** eine logische Eins ist, wird ähnlich dann die Ausgabe des UND Null sein, nur falls das zugeordnete Bitregister im ITRM0 maskiert ist, d. h. ihr invertierter Wert gleich Null ist.

[0034] Wie oben diskutiert wurde, sind alle Ausgänge der UND-Gatter mit einem ODER-Gatter gekoppelt. Falls irgendeine Ausgabe eines UND-Gatters eine logische Eins ist, ist dann die Ausgabe des ODER-Gatters eine logische Eins. Die Ausgabe des ODER-Gatters ist die Ausgabe der AO32x2 **406**. Daher erzeugt die AO32x2 **406** ein Signal eines Befehlsübereinstimmungswertes (IMV), das eine logische Null repräsentiert, nur falls alle Bits von dem Befehl INST00 entweder mit den zugeordneten ITRV-Bits übereinstimmen oder maskiert sind, d. h. falls der Befehl INST00 zum ITRV-ITRM-Registerpaar **304**, **306** passt.

[0035] Die programmierbare Befehlsfangstelleneinheit **204** bestimmt dann, **810**, ob irgendwelche Übereinstimmungen aufgetreten sind, und falls dies der Fall ist, erzeugt, **812**, sie den Wert im zugeordneten CR **308**. Der Prozess, um dies zu erreichen, wird nun beschrieben. Die Ausgabe von jeder AO32x2-Vorrichtung **406** wird von zwei logischen ODER-Gattern, zum Beispiel den ODER-Gattern **418**, **420** empfangen, wie in **Fig.** 4 veranschaulicht ist. Die zweite Eingabe in das erste ODER-Gatter **418** ist der Wert des ersten Bitregisters des zugeordneten CR **308**. Die zweite Eingabe in das zweite ODER-Gatter **420** ist der Wert des zweiten Bitregisters des zugeordneten CR **308**. Falls zum Beispiel ein Befehl INST00 nur mit dem Registerpaar ITRV0-ITRM0 zusammenpasst, erzeugt dann AO32x2 **406A** ein eine logische Null repräsentierendes Signal an ihrem Ausgang, und die verbleibenden AO32x2 **406B–D** erzeugen an ihrem Ausgang ein eine logische Eins erzeugendes Signal. Die mit AO32x2s **406B–D** verbundenen drei ODER-Gatter empfangen eine binäre Eins als eine ihrer Eingaben. Dementsprechend geben diese drei ODER-Gatter ein Signal ab, das eine binäre Eins repräsentiert. Ein NAND-Gatter **408A** erzeugt eine Ausgabe einer logischen Null, nur falls die Ausgabe des ODER-Gatters **418** eine logische Eins erzeugt. Da die Ausgabe der Vorrichtung **406A** eine logische Null ist, ist die Ausgabe des ODER-Gatters **418** eine logische Eins, nur falls das erste Bitregister von CR0 eine logische Null enthält. Jeder Satz aus vier ODER-Gattern und einem NAND-Gatter **408** ergibt entweder (1) den Wert des CR **308**, das mit einem zusammenpassenden Registerpaar verbunden ist, um von den NAND-Gattern **408A–B** ausgegeben zu werden, oder (2) falls der Befehl INST00 mit keinem Registerpaar zusammenpasst, geben die NAND-Gatter **408A–B** einen Wert "00" aus, der darstellt, dass keine spezielle Aktion unternommen werden soll, wie oben beschrieben wurde.

[0036] Alle vier empfangenen Befehle, d. h. INST00, INST01, INST10 und INST11, werden ähnlich mit jedem der vier ITRV-Register und der ITRM-Register verglichen.

[0037] Es ist möglich, dass mehr als ein Registerpaar zu einem einzelnen Befehl passt. Zum Beispiel können sowohl das Registerpaar ITRV1-ITRM1 als auch das Registerpaar ITRV2-ITRM2 zu einem bestimmten Befehl, zum Beispiel INST00, passen. Unter solchen Umständen hat ein Bitwert "1" in einem CR-Bitregister für ein CR, zum Beispiel CR2, zur Fol-

ge, dass das zugeordnete NAND-Gatter ein Signal erzeugt, das eine logische Eins repräsentiert.

[0038] Nachdem alle vier Befehle gleichzeitig getestet sind, geht das Verfahren weiter, **814**. Falls mehr Befehle empfangen werden, **712**, wiederholt sich der Prozess. Nachdem alle Befehle getestet worden sind, können neue Werte für die ITRVs **304**, ITRMs **306** und CRs **308** unter Verwendung eines Datenbusses programmiert werden, **714**, oder indem vorzugsweise der Computer neu gestartet und neue programmierbare Registerwerte, wie oben beschrieben, gescannt werden, und der Prozess **702**–**714** wiederholt sich dann.

Patentansprüche

1. Computerverarbeitungssystem, aufweisend:

eine Befehlsabrufeinheit (**202**), die angeordnet ist, um mehrere Computerbefehle zu empfangen, zum Wiedergewinnen der mehreren Computerbefehle;

eine programmierbare Befehlsfangstelleneinheit (**204**), die mit der Befehlsabrufeinheit (**202**) gekoppelt ist, mit:

mehreren Registerpaaren (**304** & **306**, ITRV0 & ITRM0– ITRV3 & ITRM3), jedes Registerpaar aufweisend

ein Werteregister (**304**, ITRV0–ITRV3) zum Speichern

eines Fang-Wertes, der einen oder mehrere Fang-Befehle repräsentiert, die ein Benutzer abfangen möchte, welcher Fang-Wert vom Nutzer programmierbar ist, und

ein Maskenregister (**306**, ITRM0–ITRM3) zum Speichern von Maskeninformation, die dem Werteregister des Paars zugeordnet ist;

mehreren Komparatoren (**404A**–**404D**, XNOR2), die jeweils angeordnet sind, um einen Computerbefehl (INST0, INST1, INST2, INST3) und einen Fang-Wert von einem der Registerpaare zu empfangen und den Computerbefehl und den betreffenden Fang-Wert zu vergleichen, um ein Vergleichsergebnis zu liefern;

mehreren Maskiervorrichtungen (**406A**–**406D**, AO32X2), die jeweils programmierbar sind, um die Maskeninformation von einem der Registerpaare auf das Vergleichsergebnis anzuwenden, das von demjenigen Komparator geliefert wurde, der angeordnet ist, um den Fang-Wert vom gleichen Registerpaar zu empfangen, um ein maskiertes Vergleichsergebnis zu liefern, auf dessen Basis bestimmt wird, ob der Computerbefehl und der Fang-Wert als zusammenpassend betrachtet werden,

mehreren programmierbaren Steuerregistern (**308**, CR0 –CR3), die jeweils einem der Registerpaare zugeordnet sind, um Daten zu enthalten, die eine Funktion darstellen, die ausgeführt werden soll, falls bestimmt wird, dass ein Computerbefehl und der Fang-Wert des zugeordneten Registerpaares als zusammenpassend betrachtet werden, und

einer mit den Steuerregistern (**308**, CR0–CR3) gekoppelten Steuereinheit (**418**, **420**, **408A**, **408B**), die betreibbar ist, um Befehlsfangstellen-Steuersignale zu erzeugen, die die Funktion repräsentieren, die durch die in einem Steuerregister enthaltenen Daten repräsentiert wird, falls bestimmt wird, dass ein Computerbefehl und der Fang-Wert des dem Steuerregister zugeordneten Registerpaares als zusammenpassend betrachtet werden.

2. Computerverarbeitungssystem nach Anspruch 1, worin jeder der Komparatoren (**404A**–**404D**, XNOR2) betreibbar ist, um jedes Bit eines Computerbefehls (INST0, INST1, INST2, INST3) mit einem entsprechenden Bit des betreffenden Fang-Wertes zu vergleichen, und das Vergleichsergebnis Bits aufweist, die jeweils angeben, ob verglichene entsprechende Bits zusammenpassen oder nicht zusammenpassen, und die auf ein Vergleichsergebnis angewendete Maskierinformation Bits aufweist, die entweder "Beachten" oder "Nicht Beachten" in Bezug auf jedes Bit des Vergleichsergebnisses angeben, welche Maskiervorrichtungen (**406A** –**406D**, AO32X2) betreibbar sind, um die Maskierinformation derart anzuwenden, dass jedes Bit "Nicht Beachten" des Vergleichsergebnisses, das ein Nicht-zusammenpassen angibt, maskiert wird, so dass ein Zusammenpassen im maskierten Vergleichsergebnis angegeben wird.

3. Computerverarbeitungssystem nach Anspruch 1, worin vier Befehle (INST0, INST1, INST2, INST3) gleichzeitig einem Abfangen in der Befehlsfangstelleneinheit unterzogen werden.

4. Verfahren zum Identifizieren abzufangender Befehle, mit den folgenden Schritten:

Programmieren in mehrere Registerpaare, wobei jedes Registerpaar ein Werteregister (**304**, ITRV0–ITRV3) und einem Maskenregister (**306**, ITRM0–ITRM3) aufweist, wie folgt

(a) in die Werteregister (**304**, ITRV0–ITRV3), Programmieren jeweiliger Fang-Werte, die einen oder mehrere Fang-Befehle repräsentieren, die ein Nutzer abfangen möchte,

(b) in die Maskenregister (**306**, ITRM0–ITRM3), Programmieren jeweiliger Elemente einer Maskeninformation, die dem Fang-Wert zugeordnet ist, der in das Werteregister des gleichen Registerpaares programmiert wurde;

Programmieren in mehrere Steuerregister (**308**, CR0– CR3), die jeweils mit einem der Registerpaare verbunden sind, von Daten, die eine Funktion repräsentieren, die ausgeführt werden soll, falls bestimmt wird, dass ein Computerbefehl und der Fang-Wert des zugeordneten Registerpaares als zusammenpassend betrachtet werden,

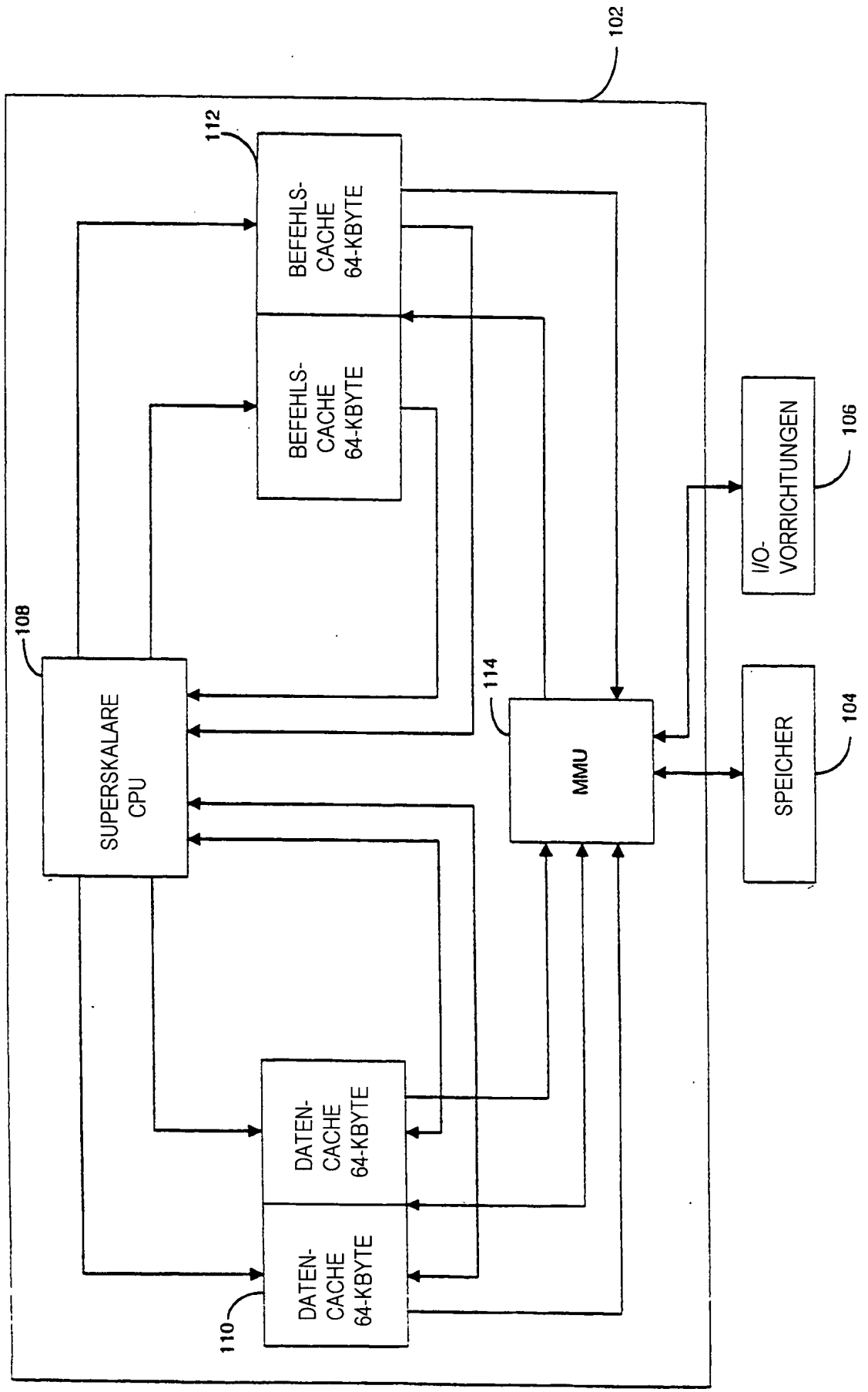
Empfangen von Computerbefehlen (INST0, INST1, INST2, INST3) und Durchführen von Vergleichen (**404A**–**404D**, XNOR2) der Computerbefehle (INST0, INST1, INST2, INST3) und der in die Registerpaare programmierten Fang-Werte, um Vergleichsergeb-

nisse zu liefern;
auf jedes Vergleichsergebnis, Anwenden der Maskeninformation vom gleichen Registerpaar (**406A**–**406D**, AO32X2), wie durch den Fang-Wert vorgesehen, der im Vergleich verwendet wurde, um ein maskiertes Vergleichsergebnis zu liefern, auf dessen Basis bestimmt wird, ob der Computerbefehl und der Fang-Wert als zusammenpassend betrachtet werden, und
Erzeugen von Befehlsfangstellen-Steuersignale (**418**, **420**, **408A**, **408B**), die die Daten darstellen, die in das einem Registerpaar zugeordnete Steuerregister (**308**, CR0 –CR3) programmiert wurden, falls bestimmt wird, dass ein Computerbefehl und der in das betreffende Registerpaar programmierte Fang-Wert als zusammenpassend betrachtet werden.

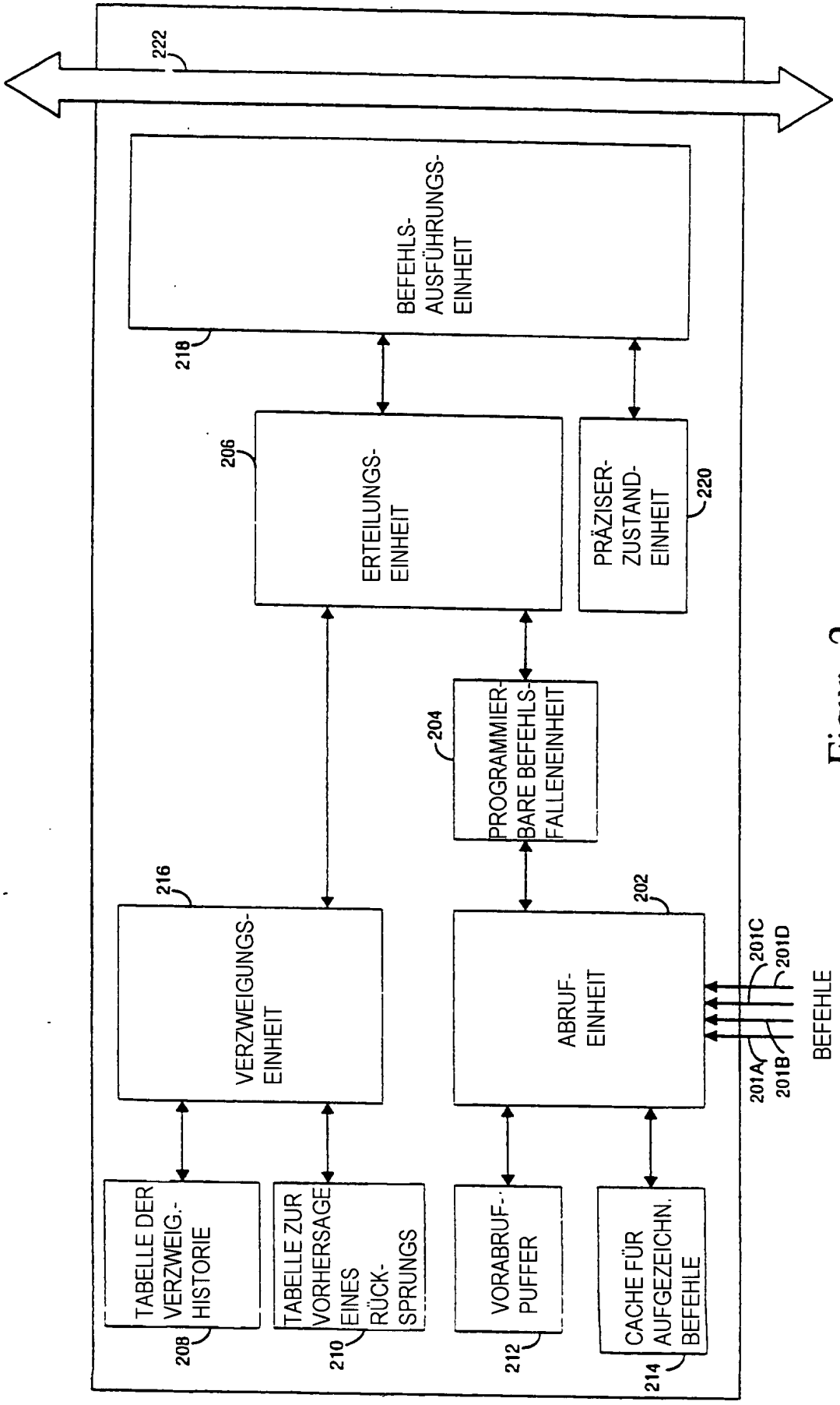
5. Verfahren nach Anspruch 4, worin in jedem Vergleich (**404A**–**404D**, XNOR2) jedes Bit eines Computerbefehls mit einem entsprechenden Bit des betreffenden Fang-Wertes verglichen wird und das Vergleichsergebnis Bits aufweist, die jeweils angeben, ob verglichene entsprechende Bits zusammenpassen oder nicht zusammenpassen, und die auf ein Vergleichsergebnis angewendete Maskierinformation Bits aufweist, die entweder "Beachten" oder "Nicht Beachten" in Bezug auf jedes Bit des Vergleichsergebnisses angeben, welche Maskierinformation angewendet wird (**406A**–**406D**, AO32X2), so dass jedes Bit "Nicht Beachten" des Vergleichsergebnisses, das ein Nicht-Zusammenpassen angibt, maskiert wird, so dass ein Zusammenpassen im maskierten Vergleichsergebnis angegeben wird.

Es folgen 7 Blatt Zeichnungen

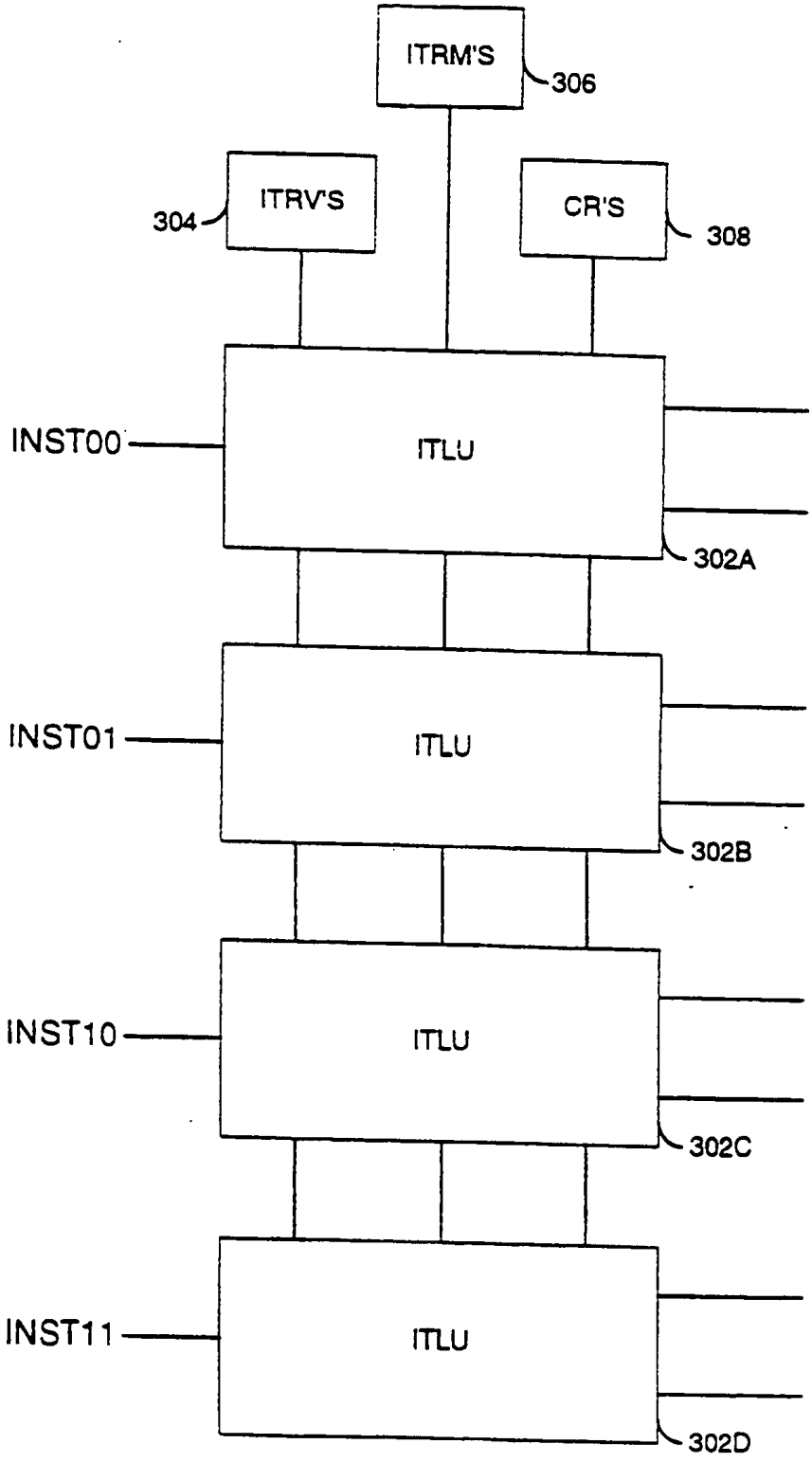
Anhängende Zeichnungen



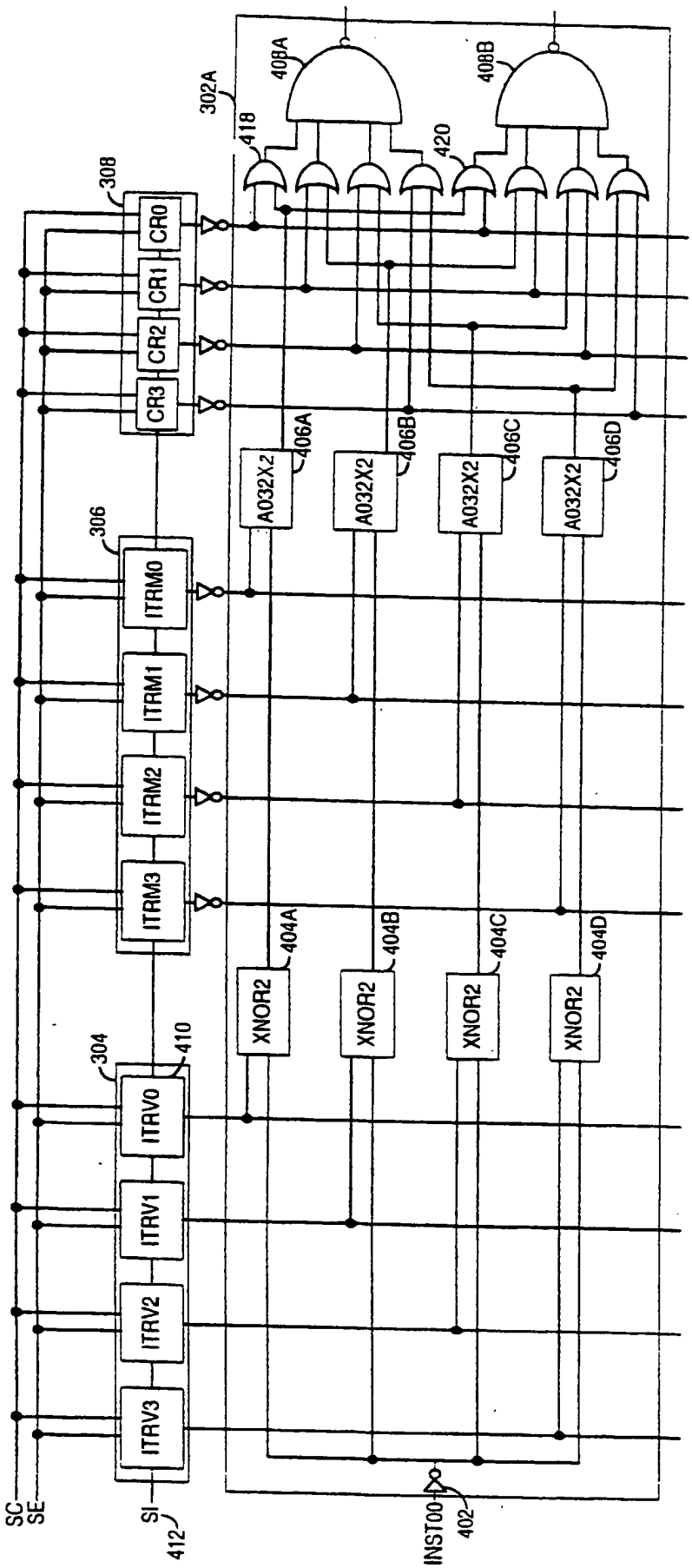
Figur 1


222
BEFEHLS-AUSFÜHRUNGS-EINHEIT
218
206
ERTEILUNGS-EINHEIT
PRÄZISER-ZUSTAND-EINHEIT
220
204
PROGRAMMIER-BARE BEFEHLS-FALLENEINHEIT
216
VERZWEIGUNGS-EINHEIT
202
ABRUF-EINHEIT
201C
201D
201A
201B
BEFEHLE
TABELLE DER VERZWEIG.-HISTORIE
208
TABELLE ZUR VORHERSAGE EINES RÜCK-SPRUNGS
210
VORABRUF-PUFFER
212
CACHE FÜR AUFGEZEICHN. BEFEHLE
214

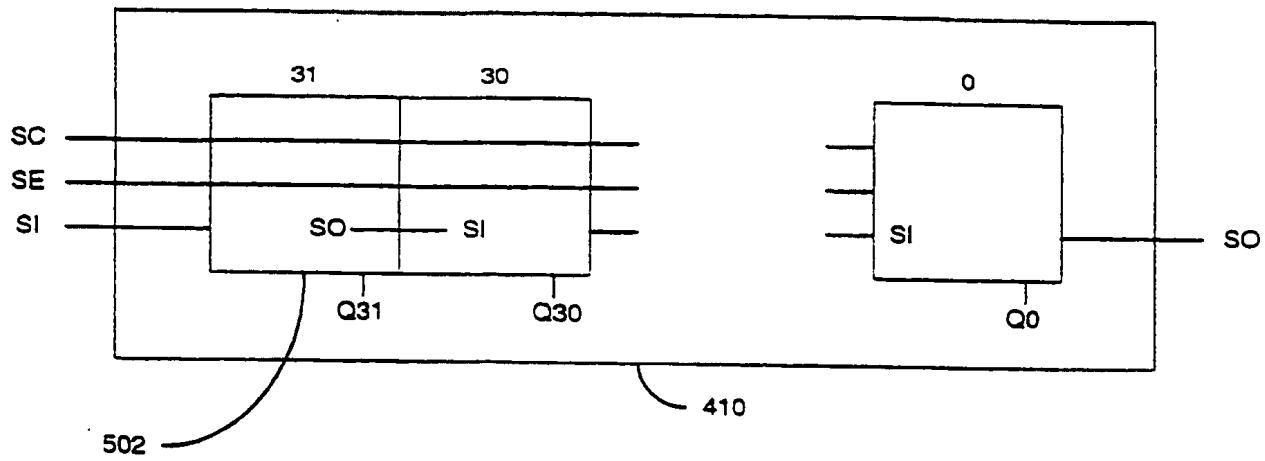
Figur 2


ITRM'S
306
304
ITRV'S
CR'S
308
INST00
ITLU
302A
INST01
ITLU
302B
INST10
ITLU
302C
INST11
ITLU
302D

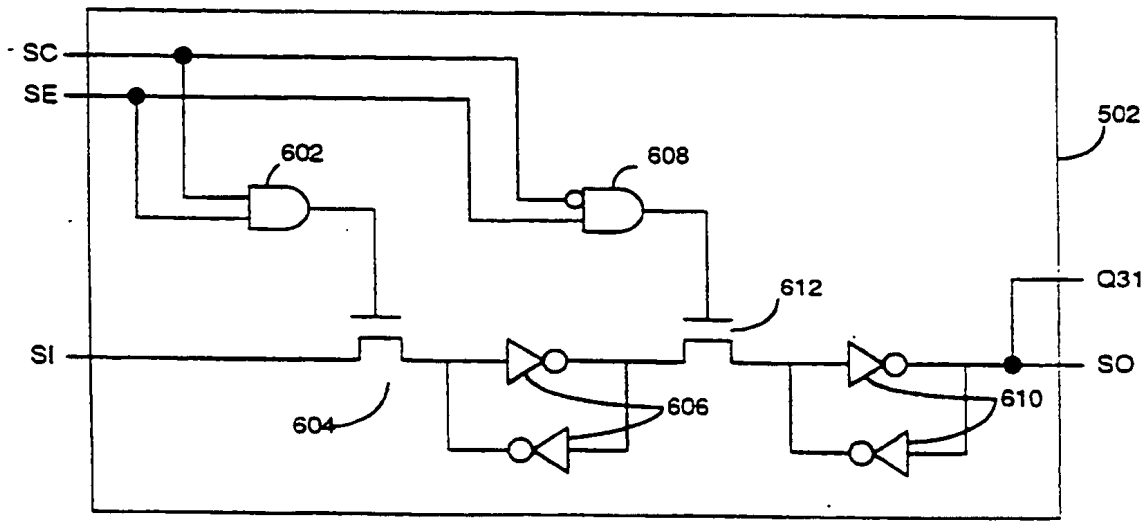
Figur 3 204


302A
418
420
408A
408B
308
CR0
CR1
CR2
CR3
306
ITRM0
ITRM1
ITRM2
ITRM3
304
410
ITRV0
ITRV1
ITRV2
ITRV3
A032X2
406A
406B
406C
406D
XNOR2
404A
404B
404C
404D
SC
SE
SI
412
INST00
402

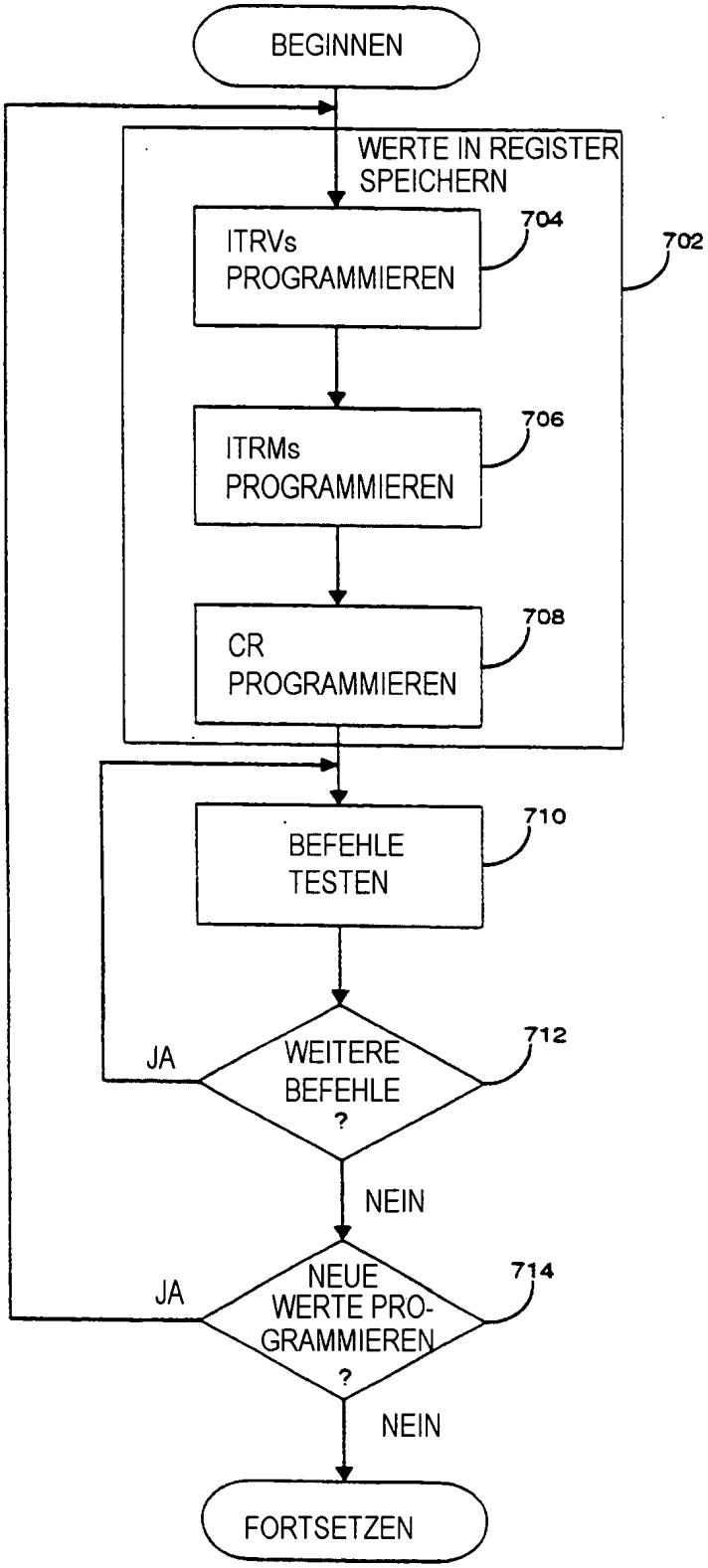
Figur 4



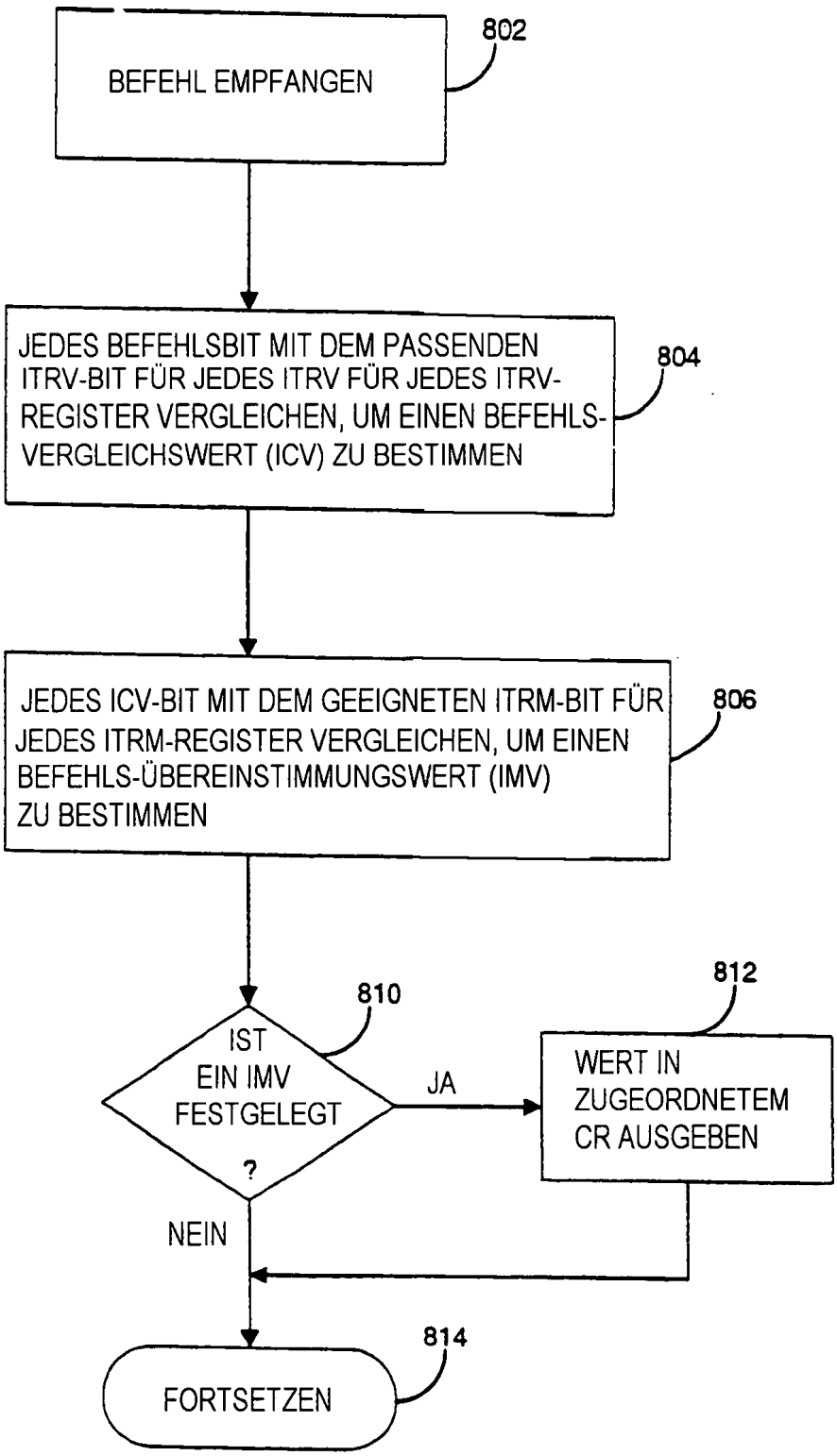
Figur 5



Figur 6


BEGINNEN
WERTE IN REGISTER SPEICHERN
ITRVs PROGRAMMIEREN
704
702
ITRMs PROGRAMMIEREN
706
CR PROGRAMMIEREN
708
BEFEHLE TESTEN
710
JA
WEITERE BEFEHLE ?
712
NEIN
JA
NEUE WERTE PRO-GRAMMIEREN ?
714
NEIN
FORTSETZEN

Figur 7


BEFEHL EMPFANGEN
802
JEDES BEFEHLSBIT MIT DEM PASSENDEN ITRV-BIT FÜR JEDES ITRV FÜR JEDES ITRV-REGISTER VERGLEICHEN, UM EINEN BEFEHLS-VERGLEICHSWERT (ICV) ZU BESTIMMEN
804
JEDES ICV-BIT MIT DEM GEEIGNETEN ITRM-BIT FÜR JEDES ITRM-REGISTER VERGLEICHEN, UM EINEN BEFEHLS-ÜBEREINSTIMMUNGSWERT (IMV) ZU BESTIMMEN
806
IST EIN IMV FESTGELEGT ?
810
JA
WERT IN ZUGEORDNETEM CR AUSGEBEN
812
NEIN
FORTSETZEN
814

Figur 8