(19) Bundesrepublik Deutschland
Deutsches Patent- und Markenamt

(10) **DE 44 30 195 B4** 2004.09.23

(12) **Patentschrift**

(21) Aktenzeichen: **P 44 30 195.2**
(22) Anmeldetag: **25.08.1994**
(43) Offenlegungstag: **14.06.1995**
(45) Veröffentlichungstag der Patenterteilung: **23.09.2004**

(51) Int Cl.[7]: **G06F 9/38**
G06F 9/305

Innerhalb von 3 Monaten nach Veröffentlichung der Erteilung kann Einspruch erhoben werden.

(30) Unionspriorität:
166582 13.12.1993 US

(71) Patentinhaber:
Hewlett-Packard Co. (n.d.Ges.d.Staates Delaware), Palo Alto, Calif., US

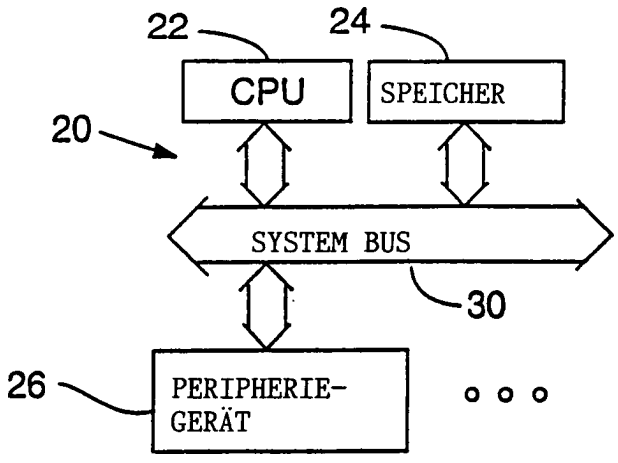
(74) Vertreter:
Schoppe, F., Dipl.-Ing.Univ., Pat.-Anw., 82049 Pullach

(72) Erfinder:
Schlansker, Michael, Los Altos, Calif., US; Rau, B. Ramakrishna, Los Altos, Calif., US; Kathail, Vinod, Cupertino, Calif., US

(56) Für die Beurteilung der Patentfähigkeit in Betracht gezogene Druckschriften:
US 49 41 085
US-Z.: IEEE Transaction on Computers, Vol. 39, März 1990, S. 349-359;
MILLS, J.W.: A pipelined architecture for logic programming with a complex but single-cycle instruction set. In: Engineering Applications of Artificial Intelligence, Vol. 3, No. 1, 1990, S. 30-42;

(54) Bezeichnung: **Verfahren zur Auswertung von Booleschen Ausdrücken**

(57) Hauptanspruch: Verfahren zur Auswertung von Booleschen Ausdrücken bestehend aus einer Mehrzahl von Termen, die durch logische Operatoren verknüpft sind, in einem Computersystem (20), in dem Operationen entsprechend Befehlen eines Befehlssatzes durch eine oder mehrere Funktionseinheiten (36-39) ausgeführt werden,
wobei die Operationen eine oder mehrere Reduktionsoperationen in einer Form einschließen, die einen Wert aus einem Satz von einem oder mehreren Eingangswerten zum Speichern in einem bestimmten Register (54) bestimmt, wobei für jeden Typ der logischen Operatoren für jeden Term während eines Zyklus folgende Schritte durchlaufen werden:
Durchführen einer Ergebnisfunktion (F_{aus}) in der Funktionseinheit (36-39) bezüglich eines oder mehrerer Eingangswerte der Operation, um einen Ergebniswert zu erzeugen;
Durchführen einer Aktivierungsfunktion (F_{en}) in der Funktionseinheit bezüglich eines oder mehrerer Eingangswerte, um einen Booleschen Aktivierungswert zu erzeugen, wobei der Boolesche Aktivierungswert für zumindest eine Kombination der Eingangswerte falsch ist; und
Schreiben des Ergebniswertes an einen bestimmten Registerort (58), wenn der...



Beschreibung

[0001] Die vorliegende Erfindung bezieht sich auf ein Verfahren zur Auswertung von Booleschen Ausdrücken bestehend aus einer Mehrzahl von Termen, die durch logische Operationen verknüpft sind.
[0002] Programme schließen typischerweise viele Boolesche Ausdrücke ein. Werte, die durch diese Ausdrücke erzeugt werden, haben eine Anzahl von Anwendungen, einschließlich als Datenwerte, als Verzweigungsbedingungen und bei einigen Computern als Vorhersagen für bedingte Operationsausführungen.
[0003] Eine herkömmliche Anwendung für Boolesche Ausdrücke ist es, eine Boolesche Reduktion durchzuführen. D.h. eine Mehrzahl von Werten auf einen einzelnen Wert unter Verwendung von Boolleschen Funktionen, z.B. UND und ODER, zu reduzieren. Ein Beispiel eines solchen Ausdruckes zum Durchführen einer Booleschen Reduktion lautet wie folgt:

$$r = u \wedge \neg v \wedge w \wedge x \tag{1}$$

[0004] Der obige Ausdruck reduziert den Booleschen Wert u, das Komplement von v (dargestellt durch ¬v), w und x auf einen einzelnen Booleschen Wert r unter Verwendung der UND-Funktion (dargestellt durch ^). Im allgemeinen kann jeder Boolesche Ausdruck in der Form der Summe von Produkten ausgedrückt werden, die aus einer Anzahl von UND-Reduktionen, denen eine ODER-Reduktion folgt, besteht.
[0005] Typischerweise ergeben sich Boolesche Werte, die in Programmen verwendet werden, aus Vergleichen oder relationalen Operationen, wie z.B. kleiner als (<) und größer als (>). Folglich ist der folgende Boolesche Ausdruck für UND-Reduktionsausdrücke, die in Computerprogrammen angetroffen werden, typisch.

$$r = (a < b) \wedge \neg(c > d) \wedge (a < c) \wedge (b > d) \tag{2}$$

[0006] Die Geschwindigkeit, mit der die Reduktionsausdrücke dieser Art ausgewertet werden können, ist für eine effiziente Programmausführung außerordentlich wichtig. Dies ist besonders im Zusammenhang mit Computersystemarchitekturen wichtig, die einen Befehlsebenenparallelismus verwenden, wie z.B. Architekturen mit sehr groben Befehlsworten (VLIW = Very Large Instruction Word) oder sogenannten Supermaßstabarchitekturen. Boolesche Ausdrücke werden typischerweise bei der Berechnung von Verzweigungsbedingungen verwendet. Folglich ist die Auswertung von Booleschen Ausdrücken häufig ein wichtiger Weg eines Programms. Einige Programmtransformationen zur Ausführung eines Parallelismus (d.h. Operationen, die gleichzeitig ausgeführt werden können) verwenden ebenfalls Boolesche Reduktionen. Die Effektivität solcher Transformationen hängt davon ab, wie schnell diese Ausdrücke ausgewertet werden können. Einige Beispiele solcher Transformationen schließen die Kombination von mehreren Zweigen, die aus einer ausgebreiteten Schleife in einen einzelnen Zweig existieren, und die Höhenreduktion der Steuerungsabhängigkeit in einer ausgebreiteten "While"-Schleife ein.
[0007] Bei Computern, denen die Fähigkeit fehlt, Operationen parallel auszuführen, werden Boolesche Ausdrücke im allgemeinen auf eine serielle Art ausgewertet. D.h. jede Operation in dem Reduktionsausdruck wird zu einem Zeitpunkt ausgeführt.
[0008] Im allgemeinen dauert eine rein sequentielle Auswertung eines Reduktionsausdrucks, der n Vergleichsoperationen enthält, $(2n - 1)\alpha$ Maschinenzyklen, wobei α die Anzahl der Zyklen pro Vergleich oder UND-Operation in dem Computersystem ist. Der Reduktionsausdruck (**2**), der oben dargestellt ist, würde z.B. 7α Zyklen dauern.
[0009] Im Zusammenhang mit Architekturen mit Befehlsebenenparallelismus werden oft Übersetzungszeit-Höhenreduktionstechniken verwendet, um die Auswertung von Booleschen Ausdrücken zu beschleunigen. Eine einfache Technik zum Beschleunigen von Auswertungen ist es, die Vergleichsoperationen parallel auszugeben, und dann die Reduktion unter Verwendung eines binären Baums von UND-Operationen durchzuführen. Es sei z.B. der oben gegebene Ausdruck (**2**) angenommen. Unter der Annahme, daß der Prozessor in dem Computersystem zumindest vier Funktionseinheiten aufweist, kann der Ausdruck ausgewertet werden, wie es in der folgenden Tabelle gezeigt ist:

Tabelle 1, Beispiel der Höhenreduktionstechnik

Zyklus	Befehl
0	r_1=a<b; r_2=¬(c>d); r_3=a<c; r_4=b>d;
α	$r_5=r_1 \wedge r_2$; $r_6=r_3 \wedge r_4$;
2α	$r=r_5 \wedge r_6$
3α	der Wert r ist zur Verwendung verfügbar

[0010] Bei dem Beispiel aus Tabelle 1 wird der Ausdruck (**2**) mittels einer einfachen Höhenreduktionstechnik in nur 3α-Zyklen ausgewertet. Mit dieser Höhenreduktionstechnik kann ein Reduktionsausdruck, der n Vergleichsoperationen enthält, im allgemeinen in $(1 + (\log_2 n))\alpha$ Zyklen ausgewertet werden, vorausgesetzt, das Computersystem kann n Operationen parallel ausführen. Sogar wenn Quellen verführbar sind,die einen zusätzlichen Parallelismus bereitstellen, benötigt diese Technik jedoch immer noch zumindest $(1 + (\log_2 n))\alpha$ Zyklen.

Stand der Technik

[0011] Aus der Veröffentlichung MILLS J. W.: A pipelined architecture for logic programming with a complex but single-cycleinstruction set. In: Engineering Applications of Artificial Intelligence, Band 3, Nr. 1, 1990, S. 30 – 42 ist eine Architektur bekannt, welche logische Programme ausführt, für deren Ausführung weniger Befehlszyklen erforderlich sind als bei bekannten Ausführungen. Es wird speziell auf die Ausführung von Bedingungsbefehlen und auf ein Speicher-Untersystem bezog genommen. Durch die Ausführung bedingter Befehle kann die Anzahl von kurzen Verzweigungen dadurch reduziert werden, dass kurze Sequenzen eines Verzweigungs-Codes in sequentielle Befehle umgewandelt werden. Diese Schrift befaßt sich somit nicht mit einem Verfahren zur Auswertung von Booleschen Ausdrücken, bei dem eine Reduzierungs-Operation dadurch durchgeführt wird, daß eine Ergebnis-Funktion und eine Aktivierungs-Funktion durchgeführt werden.

Aufgabenstellung

[0012] Ausgehend von diesem Stand der Technik liegt der vorliegenden Erfindung die Aufgabe zugrunde, ein Verfahren zur Auswertung von Booleschen Ausdrücken zu schaffen, bei dem Operationen entsprechend Befehlen eines Befehlssatzes durch eine oder mehrere Funktionseinheiten ausgeführt werden, bei dem die Auswertung in einer reduzierten Anzahl von Zyklen erfolgt.

[0013] Diese Aufgabe wird durch ein Verfahren gemäß Anspruch 1 gelöst. Das Vehalten dieses Verfahrens und dieser Technik ist lediglich durch die Anzahl von Funktionseinheiten (Quellen) begrenzt, die Operationen parallel ausführen können, und nicht durch irgendwelche Abhängigkeiten zwischen einzelnen Operationen in dem Ausdruck. Ferner können gemäß der Erfindung Operationen gleichzeitig oder in irgendeiner erwünschten Reihenfolge ausgeführt werden. Folglich ist ein Compiler z.B. frei, die Ausführung von Vergleichsoperationen mit anderen Operationen im Programm zu überlappen, was für Computersysteme mit begrenzten Quellen wichtig ist.

[0014] Die Erfindung hat zwei Aspekte.

[0015] Gemäß einem ersten Aspekt der Erfindung ermöglichen ein oder mehrere Register in einem Computersystem mehreren Operationen, einen Wert gleichzeitig in dieses Register zu schreiben, vorausgesetzt, alle durch diese Operationen geschriebenen Werte sind identisch. In diesem Fall ist das Ergebnis, das in dem Register gespeichert ist, gut definiert und gleicht jedem der Werte, die in das Register geschrieben sind. Wenn mehrere Operationen jedoch gleichzeitig unterschiedliche Werte in ein Register schreiben, ist der sich ergebende gespeicherte Wert undefiniert. Die geschriebenen Werte können Boolesche, Integer-, Gleitkomma-, oder andere Werte sein. Das Register kann ein 1-Bit-Register oder ein Bit-Ort in einem Bedingungs- oder Statusregister sein. Das Register kann ebenfalls ein Gletkommaregister oder ein Register für allgemeine Zwecke

sein. Nicht alle Register in dem Computersystem müssen diese Fähigkeit des gleichzeitigen mehrfachen Schreibens aufweisen.

[0016] Gemäß einem zweiten Aspekt der Erfindung schafft ein Computersystem einen Satz von Reduktionsoperationen. Die Ausführung jeder Reduktionsoperation ist im allgemeinen durch zwei Funktionen der Eingangswerte der Operation definiert, eine Ergebnisfunktion und eine Aktivierungsfunktion. Die Ergebnisfunktion bestimmt, welcher Wert oder Werte, wenn welche vorhanden sind, durch die Operation gespeichert werden. Die Aktivierungsfunktion bestimmt, ob diese Werte geschrieben werden oder in einem Zielort oder Register gespeichert werden oder nicht. Dementsprechend kann das Ergebnis r_{aus} durch eine Reduktionsoperation wie folgt ausgedrückt werden:

$$r_{aus} = F_{aus} (r_{ein1}, r_{ein2}, ..., r_{ein\,n}) \quad (3)$$

wenn $F_{en} (r_{ein1}, r_{ein2}, ..., r_{ein\,n})$

[0017] Wobei F_{aus} die Ergebnisfunktion ist, und F_{en} die Aktivierungsfunktion ist. Die Ergebnis- und Aktivierungsfunktion kann zumindest die folgenden Booleschen Funktionen einschließen: Vergleiche von ganzzahligen oder Gleitkomma-Eingangswerten, und Funktionen von Booleschen Eingangswerten, wie z.B. UND, ODER, Invertierung und Identitätsfunktion. Das bestimmte Zielregister ist bevorzugterweise von der Art, das mehrere gleichzeitige Schreibvorgänge handhabt. Bei einem Computersystem gemäß dieser zwei Aspekte der Erfindung kann jeder Reduktionsausdruck effektiv in nur α Zyklen ausgewertet werden. Weiterhin kann jeder allgemeine Boolesche Ausdruck, der in der Form einer Summe von Produkten ausgedrückt ist, in effektiv 2α Zyklen ausgewertet werden. Der obige Reduktionsausdruck (**2**) kann z.B. durch gleichzeitiges Ausführen von vier Operationen ausgewertet werden, die abhängig von dem Ergebnis eines Vergleichs einen Booleschen Nullwert konditional in ein Register schreiben, das auf Eins voreingestellt ist, wie es in der folgenden Tabelle dargestellt ist.

Tabelle 2: Beispiel des Auswertungsausdrucks (2) gemäß der Erfindung

Zyklus	Befehl
	r=1; (überlappt mit vorhergehenden Operationen)

0	r=UND -< (a,b);
	r=UND$_c$-> (c,d);
	r=UND -< (a,c); und
	r=UND -> (b,d).

α	Der Wert r ist für die Verwendung verfügbar.

[0018] In der obigen Tabelle ist das Register r auf Eins voreingestellt. Die Operationen bestimmtes-UND schreiben bedingt einen Booleschen Nullwert in das Register r, wenn das Ergebnis einer bestimmten Vergleichsoperation (< oder >) Null ist. Die Operation bestimmtes-UND$_c$ schreibt einen Booleschen Nullwert in das Register r, wenn das Komplement des Ergebnisses der bestimmten; Vergleichsoperation Null ist. Obwohl mehr als eine der Operationen gleichzeitig in das Register schreiben kann, sind alle Werte, die geschrieben werden können, Null.

[0019] Die vier gleichzeitig ausgeführten Operationen werten den Reduktionsausdruck (**2**) effektiv in α Zyklen aus. (Dies setzt voraus, daß die Operation zur Voreinstellung des Registers mit vorhergehenden Operationen überlappt sein kann, wie z.B. während Zyklen, wenn eine der Funktionseinheiten ansonsten leerlaufen würde. Folglich benötigt die Voreinstellungsoperation effektiv keine zusätzliche Ausführungszeit.) Wenn das Ergebnis irgendeiner der durch die UND-Operationen durchgeführten Vergleiche oder das Komplement des durch die UND$_c$-Operation durchgeführten Vergleichs Null ist, wird das Register auf Null eingestellt. Wenn das Register r dem ersten Aspekt der Erfindung entspricht, werden mehrere Operationen, die das Register gleichzeitig auf Null einstellen, zu einem Register führen, das auf einen definierten Wert von Null eingestellt ist. Das Register

bleibt ansonsten auf 1 eingestellt.

[0020] Zusätzlich kann jeder allgemeine Boolesche Ausdruck, der in der Form der Summe von Produkten ausgedrückt ist, in 2α Zyklen ausgewertet werden. (Dies setzt wiederum voraus, daß die Voreinstellungsoperationen sich mit vorhergehenden Operationen überlappen, so daß diese keine zusätzlichen Zyklen benötigen.) Bei den ersten α Zyklen wird jeder UND-Ausdruck des Ausdrucks als eine getrennte UND-Reduktion durchgeführt. Bei den nächsten α Zyklen wird eine ODER-Reduktion bezüglich der Ergebnisse der UND-Reduktionen durchgeführt, um den Wert des Ausdrucks zu erhalten.

[0021] Die Werte der Booleschen Ausdrücke sind für eine vorhersagbare Ausführung besonders nützlich. Eine vorhersagbare Ausführung von Befehlen bezieht sich auf eine Ausführung, die für einen Wert einer Eingabe, normalerweise ein Boolescher Wert, konditioniert ist. Auf einer Maschine, die eine vorhersagbare Ausführung unterstützt, kann ein Befehl, der zwei Eingangswerte addiert, z.B. diesbezüglich konditioniert sein, ob eine dritte, vorhersagbare Eingabe einen bestimmten Wert hat. Die Addition von zwei Eingangswerten oder zumindest das Schreiben des Ergebnisses der Additionsfunktion findet nur statt, wenn die vorhersagbare Eingabe den bestimmten Wert hat. Folglich können gemäß einem weiteren Merkmal und einem weiteren Vorteil der Erfindung Befehle bezüglich Reduktionsausdrücken, die gemäß der Erfindung ausgewertet werden, vorhergesagt werden.

Ausführungsbeispiel

[0022] Bevorzugte Ausführungsbeispiele der vorliegenden Erfindung werden nachfolgend unter Bezugnahme auf die beiliegenden Zeichnungen näher erläutert. Es zeigen:

[0023] **Fig.** 1 ein verallgemeinertes Blockdiagramm eines Computersystems

[0024] **Fig.** 2 ein verallgemeinertes Blockdiagramm einer zentralen Verarbeitungseinheit in dem Computersystem aus **Fig.** 1 mit mehreren Funktionseinheiten für die Befehlsebenenparallelität; und

[0025] **Fig.** 3 ein verallgemeinertes Blockdiagramm eines Registers in der zentralen Verarbeitungseinheit aus **Fig.** 2.

[0026] Anhand von **Fig.** 1 wird ein Computersystem **20** beschrieben, das im allgemeinen eine zentrale Verarbeitungseinheit (CPU) **22** umfaßt, die über einen Mehrsignalsystembus **30** mit einem Speicher **24** und peripheren Geräten **26** verbunden ist. Der Speicher **24** wird bevorzugterweise durch dynamische Speicherchips mit wahlfreiem Zugriff ausgeführt, kann aber mit Nur-Lese-Speicherchips oder anderen elektronischen Speichergeräten ausgeführt sein. Der Speicher **24** speichert Daten und Befehle für ein oder mehrere Programme, die ein erwünschtes Ziel oder eine erwünschte Berechnung auf dem Computersystem **20** ausführen.

[0027] Die CPU **22** in dem Computersystem **20** schafft bevorzugterweise eine Befehlsebenenparallelität, wie in **Fig.** 2 gezeigt ist. (Bei alternativen Ausführungsbeispielen der Erfindung kann das Computersystem eine CPU von der Art aufweisen, die das Befehls-Pipeline-Verfahren verwendet.) Genauer gesagt umfaßt der Prozessor **22** eine Befehlseinheit **34** und mehrere Funktionaleinheiten **3639**, die Operationen gleichzeitig entsprechend den Befehlen eines Programms zur Ausführung der erwünschten Aufgabe oder Berechnung ausführen. Die Befehle, die in dem Speicher **24** (**Fig.** 1) gespeichert sind, werden in die Befehlseinheit **34** eingelesen. Eine Busschnittstellenschaltung **42** erzeugt auf dem Systembus **30** die Signale, die notwendig sind, um Daten und Befehle aus dem Speicher zu lesen und Daten in den Speicher **24** zu schreiben. Die Befehlseinheit **34** dekodiert die Befehle und erzeugt Steuerungssignale, die einen Satz von Funktionseinheiten **36-39** anweisen, Operationen in Übereinstimmung mit den Befehlen auszuführen.

[0028] Die Befehlseinheit **34** ist aufgebaut, um auf einen vorbestimmten Satz von Befehlen (den "Befehlssatz") zu reagieren, und die Steuerungssignale zu bilden, die notwendig sind, um die entsprechenden Operationen mit den Funktionseinheiten **36-39** auszuführen. Der Befehlssatz umfaßt im allgemeinen Logikoperationsbefehle, Arithmetikoperationsbefehle und Gleitkommaoperationsbefehle. Der Befehlssatz umfaßt ebenfalls Datenübertragungsbefehle zum Übertragen von Daten zwischen einem Speicher und einem Satz von Registern **46** und Zwischenregistern. In Übereinstimmung mit der Erfindung sind einige der Operationen, die die Befehlseinheit **34** an die Funktionseinheiten zur Durchführung weiterleitet, Operationen für eine vorhersagbare Programmausführung und für die Auswertung von Booleschen Reduktionen, die im folgenden genauer beschrieben werden.

[0029] Die Funktionseinheiten **36-39** führen Operationen durch, um die Befehle auszuführen, wie es durch die Befehlseinheit **34** angewiesen ist. Die Funktionseinheiten **36-39** lesen im allgemeinen einen oder zwei Operanden, die für eine Operation verwendet werden aus Registern ein, die aus den Registern **46** ausgewählt sind. Die Funktionseinheit **36-39** schreibt ebenfalls eines oder mehrere Ergebnisse der Operation in ein Register, das aus den Registern **46** ausgewählt ist.

[0030] Die Befehlseinheit **34** und die Funktionseinheiten **36-39** sind bei dem bevorzugten Ausführungsbeispiel Logikschaltungen, die mit herkömmlichen Logikgattern ausgeführt sind, unter Verwendung von herkömmlichen Logikschaltungsentwurfstechniken, um die im folgenden beschriebenen Operationen bereitzustellen.

[0031] In **Fig.** 3 speichert zumindest eines der Register **46** (**Fig.** 2) Boolesche Operanden, und ist fähig, meh-

rere gleichzeitige Schreibvorgänge handzuhaben. Ein solches Register **54** umfaßt bevorzugterweise ein Speichergerät oder eine Schaltung ("Speicher") **58** und eine Anzahl von Anschlüssen **60**-**63**. (Andere der Register **46** können für andere Aufgaben bestimmt sein, wie z.B. das Speichern von ganzzahligen oder Gleitkommawerten.)

[0032] Im allgemeinen umfaßt der Speicher **58** einen oder mehrere Zwischenspeicher, Flip-Flops oder äquivalente statische oder dynamische Speicherschaltungen. Der Speicher **58** kann ausgeführt sein, um lediglich ein einzelnes Bit zu speichern, er kann aber ebenfalls ausgeführt sein, um Mehrbit-Werte, wie z.B. ganzzahlige oder Gleitkommawerte, zu speichern. Im Fall eines Mehrbit-Speichers kann der Boolesche Operand durch Zuordnen von willkürlichen Operationen zu jedem der Booleschen Zustände (z.B. alle Nullen stellen eine Boolesche Null dar, und jede andere Bit-Kombination stellt eine Boolesche Eins dar) dargestellt sein. Die Mehrbit-Speicherausführung kann verwendet werden, um einen Booleschen Operanden und ganzzahlige oder Gleitkomma-Mehrbit-Werte, die für die Programmausführung erforderlich sein können, zu speichern. Der Speicher **58** kann ebenfalls ausgeführt sein, um einen Mehrbit-Vorhersagevektor zu speichern, der eine Mehrzahl von Booleschen Einzelbit-Vorhersagewerten umfaßt, die einzeln oder gemeinsam geschrieben werden können.

[0033] Die Anschlüsse **60-63** empfangen Werte, die von den Funktionseinheiten **36-39** in das Register **54** geschrieben werden. Die Anschlüsse **60-63** sind im allgemeinen mit den Funktionseinheiten **36-39** in einer Eins-zu-Eins-Beziehung sowohl mit einem Datenbus **66-69** als auch mit einer Schreibaktivierungsleitung **70-73** verbunden. Um einen Wert in das Register **54** zu schreiben, wie z.B. das Ergebnis einer Operation, legen die Funktionseinheiten **36-39** den zu schreibenden Wert an ihren Datenbus **66-69** an, und geben ein Schreibaktivierungssignal an ihre Schreibaktivierungsleitung **70-73** aus. Als Reaktion auf das Schreibaktivierungssignal schalten die Anschlüsse **60-63** den Wert auf ihrem jeweiligen Datenbus **66-69** in den Speicher **58** durch. Wenn die Funktionseinheit **37** z.B. das Schreibaktivierungssignal vom Anschluß **61** ausgibt, wird der Wert auf dem Datenbus **67** für diesen Anschluß in den Speicher **58** geschrieben. Wenn das Schreibaktivierungssignal für den Anschluß **61** nicht ausgegeben ist, wird der Wert nicht in den Speicher **58** geschrieben. (In dem Fall der Ausführung zur Speicherung von Vorhersagevektoren können für jeden der Vorhersagewerte in dem Vektor getrennte Aktivierungsleitungen und Signale verwendet werden.)

[0034] Wenn mehr als eine der Funktionseinheiten **36-39** ihre Schreibaktivierungssignale während gleichzeitig ausgeführter Operationen ausgibt, werden mehrere Werte gleichzeitig in das Register **54** geschrieben. Bei bekannten herkömmlichen Registern sind solche mehrfachen gleichzeitigen Schreibvorgänge nicht möglich und können verursachen, daß der in dem Speicher **58** gespeicherte Wert undefiniert ist. Das Register **54** ist jedoch wirksam, um einen definierten Wert in dem Speicher **58** zu speichern, wenn alle Werte, die gleichzeitig in das Register **54** geschrieben werden, identisch sind. Der gespeicherte Wert ist in diesem Fall bevorzugterweise identisch mit dem geschrieben Wert.

[0035] Der zu speichernde Wert wird, wenn mehrere Werte gleichzeitig geschrieben werden, durch eine Auswahlschaltung **76** abgeleitet, die zwischen den Anschlüssen **60-63** und dem Speicher **58** in dem Register **54** geschaltet ist. Die Auswahlschaltung **76** kann den gespeicherten Wert auf verschiedene Arten ableiten. Die Auswahlschaltung kann z.B. eine UND- oder ODER-Funktion bzgl. der mehrfach geschriebenen Werte durchführen, oder kann einen dieser willkürlich auswählen, um den gespeicherten Wert abzuleiten. Wenn die mehreren gleichzeitig geschriebenen Werte sich unterscheiden, ist das Ergebnis undefiniert. Folglich ist der durch die Auswahlschaltung **76** gespeicherte Wert für den Fall von unterschiedlichen gleichzeitig geschriebenen Werten nicht wichtig.

[0036] In **Fig.** 2 sind die Funktionseinheiten **36-39** durch die Steuerung der Befehlseinheit 34 wirksam, um bestimmte Operationen auszuführen, die dem Befehl eines Programms entsprechen. Die Operationen, die die Funktionseinheiten **36-39** unter der Steuerung der Befehlseinheit ausführen, schließen einen Satz von Reduktionsoperationen ein, die für eine vorhersagbare Programmausführung und die Auswertung von Booleschen Reduktionsausdrücken nützlich sind. Die Reduktionsoperationen haben jeweils die folgende Form:

$$r_{aus} = \text{Operation}\ (r_{ein1}, r_{ein2}, ..., r_{ein\,n}) \tag{4}$$

[0037] Mit anderen Worten agiert jede Reduktionsoperation auf einen oder mehrere Eingaben (r_{ein1}, r_{ein2}, ..., $r_{ein\,n}$), um einen neuen Wert eines Ergebnisses (r_{aus}) zu bestimmen, der in einem bestimmten Register gespeichert wird. Genauer gesagt ist die Ausführung jeder Operation durch zwei Funktionen, eine Ergebnisfunktion (F_{aus}) und eine Aktivierungsfunktion (F_{en}), definiert, wobei das Ergebnis (raus) wie folgt lautet:

$$r_{aus} = F_{aus}\ (r_{ein1}, r_{ein2}, ..., r_{ein\,n}) \tag{5}$$

wenn

$$F_{en}\ (r_{ein1}, r_{ein2}, ..., r_{ein\,n})$$

[0038] Die Ergebnisfunktion (F_{aus}) bestimmt den neuen Wert eines Ergebnisses, das in einem bestimmten Register gespeichert ist, wenn ein Ergebnis gespeichert ist. Die Aktivierungsfunktion (F_{en}) berechnet einen Booleschen Wert, der bestimmt, ob das Ergebnis in dem bestimmten Ausgabespeicher gespeichert ist oder nicht.
[0039] Bei einem Ausführungsbeispiel hat die Ergebnisfunktion einen Booleschen Wert. Für jede Reduktionsoperation ist eine Ergebnisfunktion mit Booleschem Wert und eine Aktivierungsfunktion mit Booleschem Wert typisch. Jede Reduktionsoperation wird in einer Funktionseinheit durch (1) Durchführen der bestimmten Ergebnisfunktion der Operation auf einen Satz von Eingaben, um einen sich ergebenden Booleschen Wert zu erzeugen, und durch (2) Durchführen der Aktivierungsfunktion der Operation, die bestimmt, ob der Boolesche Wert in ein bestimmtes Zielregister geschrieben wird, durchgeführt. Beispiele für die Arten von Funktionen mit Booleschem Wert, die für eine Ergebnis- oder Aktivierungsfunktion typisch sein können, schließen im allgemeinen Vergleichsfunktionen (z.B. größer als, kleiner als, gleich, deren Komplemente und ähnliches) bezüglich ganzzahliger Eingangswerte, Vergleichsfunktionen bezüglich Gleitkommaeingangswerte, und Funktionen von Booleschen Eingangswerten (z.B. UND, ODER, EXKLUSIV- ODER, Identität, deren Komplemente und ähnliches) ein.
[0040] Für jede Reduktionsoperation, die die Befehlseinheit **34** an eine der Funktionseinheiten **36-39** zur Ausführung richtet, bestimmt die Befehlseinheit **34** die Eingangswerte (oder Register, aus denen die Eingangswerte entnommen werden) für die bestimmte Ergebnisfunktion mit Booleschem Wert der Operation, und ein Zielregister, um jeglichen Wert, der durch die Operation geschrieben wird, zu speichern. Der bestimmte Eingangswert muß für die bestimmte Ergebnisfunktion der Operation geeignet sein. Das bestimmte Zielregister muß fähig sein, einen Booleschen Wert zu speichern, und ist bei dem bevorzugten Ausführungsbeispiel von dem Typ (Register **54**), das in **Fig.** 3 dargestellt ist und oben beschrieben wurde, zur Handhabung mehrerer gleichzeitiger Schreibvorgänge.
[0041] Die Aktivierungsfunktionen bestimmen, ob der Ergebniswert, der durch die Ergebnisfunktion erzeugt wird, in das Zielregister geschrieben wird. Dementsprechend benötigt jede Reduktionsoperation drei Aktionen, die von den verbundenen Ergebnissen der Ergebnis- und Aktivierungsfunktion abhängen. Die Aktionen schließen das Schreiben eines Booleschen Eins-Wertes in das Zielregister, das Schreiben eines Null-Wertes in das Zielregister und das Nichtschreiben in das Zielregister ein.
[0042] Die folgende Tabelle 3 stellt eine Wahrheitstabelle für vier Klassen von Reduktionsoperationen bei dem bevorzugten Ausführungsbeispiel der Erfindung dar.

Tabelle 3. Wahrheitstabelle für UND- oder ODER-Reduktionsoperationen

Bedingung	bei Bedingung		bei Komplement der Bedingung	
	ODER	UND	$ODER_C$	UND_C
0	--	0	1	--
1	1	--	--	0

[0043] In Tabelle 3 stellt die linkeste Spalte die Zustände eines "Zustands"-Wertes dar, der als ein Ergebnis einer Vergleichsfunktion mit Booleschem Wert erzeugt wird. Die anderen Spalten stellen die Aktionen dar, die durch vier Klassen von Reduktionsoperationen, ODER, UND, $ODER_c$ und UND_c für jeden Zustand der Bedingung durchgeführt werden, wobei "1" das Schreiben einer Booleschen Eins in das Zielregister darstellt, "0" stellt das Schreiben einer Booleschen Null in das Zielregister dar, und "--" stellt dar, daß der Wert in dem Zielregister unverändert bleibt.
[0044] Wie in **Fig.** 3 dargestellt ist, führen die Funktionseinheiten **36-39** die oben beschriebenen UND- und ODER-Reduktionsoperationen durch Anordnen des Wertes, der in der Wertespalte der folgenden Tabelle 4 gezeigt ist, auf ihrem jeweiligen Datenbus **66-69** durch, und steuern ihre jeweiligen Schreibaktivierungsleitungen an, wie es in der Schreibaktivierungssignalspalte gezeigt ist.

Tabelle 4. Ausführung von UND- und ODER-Reduktionsoperationen

Reduktions-operation	Ergebnis-Wert	Schreibaktivierungs-signal
UND	0	-Bedingung
UND_C	0	Bedingung
ODER	1	Bedingung
$ODER_C$	1	-Bedingung

[0045] Die obige Tabelle 4 zeigt die Werte, die durch die Ergebnis- und Aktivierungs-Funktion für jede der vier Klassen von Operationen aus Tabelle 3 erzeugt werden. Die Ergebnisfunktionen für diese Operationen sind Konstante. Wie es in der "Wert"-Spalte gezeigt ist, erzeugen die Ergebnisfunktionen für die UND- und die UND_c-Operationsklassen immer ein Ergebnis von Null, während diejenigen der ODER- und $ODER_c$-Operationsklassen immer ein Ergebnis von Eins erzeugen. Die "Schreibaktivierungssignal"-Spalte zeigt, daß die Aktivierungsfunktionen für diese Operationsklassen die Vergleichsfunktion sind, die den Bedingungswert oder dessen Komplement erzeugt. Genauer gesagt ist die Aktivierungsfunktion für die ODER- und UND_c-Reduktionsoperationsklassen eine bestimmte Vergleichsfunktion mit Booleschem Wert (die Vergleichsfunktion, die die Bedingung erzeugt). Die Aktivierungsfunktion für die $ODER_c$- und UND-Reduktionsoperationsklassen ist das Komplement der bestimmten Vergleichsfunktion mit Booleschem Wert.

[0046] Die vorhergehende Tabelle 3 stellt vier Klassen von Reduktionsoperationen dar, die bei dem bevorzugten Ausführungsbeispiel der Erfindung bereitgestellt werden, insbesondere die UND-, UND_c-, ODER- und $ODER_c$-Reduktionsoperationsklassen. Die Reduktionsoperationen in der UND- und UND_c-Klasse sind zur Auswertung von UND-Reduktionsausdrücken nützlich. Die Reduktionsoperationen in der ODER- und $ODER_c$-Klasse sind zur Auswertung von ODER-Reduktionsausdrücken nützlich. Der UND-Reduktionsausdruck (**2**), der wie folgt lautet:

r = (a < b) ^ (c > d) ^ (a < c) ^ (b > d) (2)

kann z.B. durch die folgenden vier UND- und UND_c-Klassen-Reduktionsoperationen ausgewertet werden:

r = UND ≤ (a, b); (6)

$r = UND_c \geq (c, d)$; (7)

r = UND ≤ (a, b); und (8)

r = UND ≥ (b, d) . (9)

[0047] Die obigen Reduktionsoperationen (6 – 9) bestehen jeweils aus einer Operations-spezifischen Vergleichsfunktion (< oder >) bezüglich zwei bestimmten Eingangswerten (in Klammern), und aus einer Operations-spezifischen Reduktionsoperation (UND oder UND_c), die auf ein bestimmtes Register (r) zeigen. Jede Vergleichsfunktion einer Reduktionsoperation erzeugt eine Bedingung mit Booleschem Wert. Wie in den Tabellen 3 und 4 oben gezeigt ist, bestimmt jede Aktivierungsfunktion der Reduktionsoperation dann auf der Grundlage der Bedingung, ob eine Boolesche Null in das Zielregister geschrieben wird. Es wird darauf hingewiesen, daß die Reduktionsoperationen (6) und (8) dieselben Reduktionsoperationen sind, aber unterschiedliche Eingangswerte bestimmen.

[0048] Um den UND-Reduktionsausdruck (**2**) mit den Reduktionsoperationen (6 – 9) auszuwerten, wird das Register (r) zuerst auf einen Booleschen Eins-Wert voreingestellt. Dann werden jeweils die Reduktionsoperationen (6 – 9) ausgeführt. Bevorzugterweise, aber nicht notwendigerweise, werden die Reduktionsoperationen

durch die Funktionseinheiten **36-39** parallel ausgeführt. Wenn die Vergleichsfunktion irgendeiner UND-Reduktionsoperation eine Null erzeugt, dann wird eine Null in das Zielregister geschrieben. Wenn der Vergleichsoperationsabschnitt von UND_c eine Eins erzeugt (das Komplement des Vergleichs ist eine Null), dann wird eine Null in das Zielregister geschrieben. Dementsprechend wird, nachdem alle Reduktionsoperationen (6) bis (9) ausgeführt sind, das Zielregister eine Eins enthalten, wenn alle Terme des UND-Reduktionsausdrucks (**2**) wahr waren, und wird ansonsten eine Null enthalten (wodurch der Ausdruck (**2**) effektiv ausgewertet wird). Wie in Tabelle 2 oben gezeigt ist, wird der Ausdruck (**2**) effektiv in nur α Zyklen auf dem Computersystem **20** ausgewertet, wenn die Reduktionsoperationen (6 – 9) parallel ausgeführt werden.

[0049] Bei einem alternativen Ausführungsbeispiel der Erfindung bestimmen die Reduktionsoperationen den Wert, der in das Zielregister geschrieben wird, auf der Grundlage einer vorhersagbaren Eingabe mit Booleschem Wert, und auf der Grundlage des Bedingungswertes, der durch eine Vergleichsfunktion erzeugt wird. Bei diesem alternativen Ausführungsbeispiel bestimmt die Befehlseinheit **34** einen Vorhersagewert (oder ein Register, das den Vorhersagewert enthält) für jede Reduktionsoperation, zusätzlich zu dem Eingangswert oder Werten und dem Zielregister der Operation. Die folgende Tabelle 5 stellt einige exemplarische Klassen von Reduktionsoperationen bei diesem alternativen Ausführungsbeispiel dar:

Tabelle 5 Wahrheitstabelle der Reduktionsoperationen mit vorhersagbarer Eingabe

P-Ein	Beding.	bei Bedingung				bei Komplement der Bed.			
		U	C	ODER	UND	U_C	O_C	$ODER_C$	UND_C
0	0	0	-	-	-	0	-	-	-
0	1	0	-	-	-	0	-	-	-
1	0	0	0	-	0	1	1	1	-
1	1	1	1	1	-	0	0	-	0

[0050] Tabelle 4 stellt acht Klassen von Reduktionsoperationen dar, einschließlich der unkonditionalen (U), der konditionalen (C), der ODER- und der UND-Klasse und deren Komplemente (U_c, C_c, $ODER_c$, UND_c). Die unkonditionalen Klassen (U, U_c) schreiben die Boolesche UND-Funktion der Vorhersageeingabe und die Bedingung (oder deren Komplement) in das Zielregister. Die konditionale Klasse schreibt die Bedingung abhängig von der vorhersagbaren Eingabe in das Zielregister. Abschließend werden die ODER- und UND-Klasse, wie in **Fig.** 3 gezeigt ist, abhängig von der vorhersagbaren Eingabe durchgeführt. Wie bei den Reduktionsoperationen, die in Tabelle 3 gezeigt sind, ist die Ausführung jeder dieser Reduktionsoperationen durch eine Ergebnisfunktion, die einen Ergebniswert erzeugt, und eine Aktivierungsfunktion, die bestimmt, ob der Ergebniswert in ein bestimmtes Zielregister geschrieben wird, definiert.

[0051] Mit den Reduktionsoperationen der Arten, die in Tabelle 5 gezeigt sind, kann das Verhalten eines Satzes von Reduktionsoperationen, die einen Reduktionsausdruck auswerten, auf den Wert einer vorhersagbaren Eingabe konditioniert werden, die in einem bestimmten Register enthalten ist. Ein Satz von Reduktionsoperationen, die eine Operation der Konditional-Klasse zur Voreinstellung eines Zielregisters und zwei oder mehr UND-Klassenoperationen einschließt, kann z.B. eingestellt sein, um einen UND-Reduktionsausdruck (z.B. Ausdruck (**2**)) auszuwerten, wenn eine bestimmte vorhersagbare Eingabe Eins ist, und nichts zu tun, wenn die vorhersagbare Eingabe Null ist.

[0052] Der Boolesche Wert, der aus der Ausführung eines Satzes von Reduktionsoperationen resultiert, der in dem Zielregister der Operation gespeichert ist, nachdem die Operation ausgeführt ist, ist für die vorhersagbare Ausführung von Operationen in einem Programm besonders nützlich. Bei dem bevorzugten Ausführungsbeispiel der Erfindung können die Operationen ein Register bestimmen, in dem ein Boolescher Vorhersagewert gespeichert ist. Die Ausführung der Operation wird dann auf den Vorhersagewert konditioniert. Bei dem bevorzugten Ausführungsbeispiel werden die Operationen nur ausgeführt, wenn ihr bestimmter Vorhersagewert Eins ist.

[0053] Bei einem weiteren alternativen Ausführungsbeispiel ist ein Satz von Reduktionsoperationen vorgesehen, die jeweils mehrere Ergebnisse bei einer gleichen Operation aus einem gegebenen Satz von Eingangswerten erzeugen. Als solche haben die Reduktionsoperationen mit mehreren Ergebnissen die folgende Form:

$$r_{aus,1}, r_{aus,2}, ..., r_{aus,m} = op(r_{aus,1}, r_{aus,2}, ..., r_{aus,n}) \quad (10)$$

[0054] Wobei die mehreren Ergebniswerte ($r_{aus,1}$, $r_{aus,2}$, ..., $r_{aus,m}$) durch Ausführen der Operation auf den Satz von Eingangswerten ($r_{aus,1}$, $r_{aus,2}$, ..., $r_{aus,n}$) erzeugt werden. Die Ausführungen dieser Reduktionsoperationen ist durch zwei Funktionen ($F_{aus,i}$, $F_{en,i}$) für jeden der mehreren Ergebnis-Werte präzise definiert. Genauer gesagt nimmt das Ergebnis ($r_{aus,i}$) für jedes i den folgenden Wert an:

$$r_{aus,i} = F_{aus,i}\ (r_{ein,1}, r_{ein,2}, ..., r_{ein,n}) \quad (11)$$

wenn

$$F_{en,i}\ (r_{ein,1}, r_{ein,2}, ..., r_{ein,n})$$

[0055] Die Reduktionsoperationen von der Art, die mehrere Ergebnisse erzeugen, sind in Situationen besonders sinnvoll, in denen mehr als ein Reduktionsausdruck denselben Satz von Bedingungen, die ausgewertet werden sollen, verwendet. In solchen Situationen kann es z.B. notwendig sein, einen UND-Reduktionsausdruck eines Satzes von vier Vergleichen und einen ODER-Reduktionsausdruck desselben Satzes von Vergleichen auszuwerten. Bei diesem weiteren Ausführungsbeispiel können beide Reduktionsausdrücke gleichzeitig mit nur vier Funktionseinheiten ausgewertet werden. Die vier Funktionseinheiten führen vier der Reduktionsoperationen durch, von denen jede eine UND- und eine ODER-Reduktionsfunktion zum Schreiben in zwei getrennte Zielregister hat. Diese Technik ist besonders zur Berechnung von zwei Ergebnisvorhersagen nützlich, die für die vorhersagbare Ausführung von Operationen von der wahren und falschen Seite eines Bedingungszweigs verwendet werden.

[0056] Nachdem die Prinzipien der vorliegenden Erfindung anhand eines bevorzugten Ausführungsbeispiels beschrieben und dargestellt wurden, ist es offensichtlich, daß die Erfindung bezüglich ihrer Anordnung verändert werden kann, ohne sich von diesen Prinzipien zu entfernen. Obwohl das dargestellte Ausführungsbeispiel ein Computersystem mit paralleler Befehlsebene ist, ist die Erfindung ebenfalls auf Multi-Prozessor-Computersysteme anwendbar (die getrennte Prozessoren zur Ausführung von Operationen haben, und nicht getrennte Funktionseinheiten innerhalb eines Prozessors). Bei einer solchen Ausführung eines Multi-Prozessor-Computersystems werden die Reduktionsausdrücke durch Ausführen eines Satzes von Reduktionsoperationen jeweils auf einem getrennten Prozessor zwischen benachbarten Synchronisationspunkten ausgewertet. Die Reduktionsoperationen bilden den Wert des Ausdrucks in einem Speicherregister, das mehrfache gleichzeitige Schreibvorgänge handhabt.

[0057] Als weiteres Beispiel kann die Vergleichsfunktion mit Booleschem Wert, die einem Reduktionsausdruck zugeordnet ist, getrennt von den Reduktionsoperationen, die den Ausdruck auswerten, durchgeführt werden. Genauer gesagt können die Vergleichsfunktionen mit Booleschem Wert in einem ersten Satz von Operationen durchgeführt werden, um einen Satz von Bedingungen für einen erwünschten Reduktionsausdruck zu bilden. Dann wird ein zweiter Satz von Reduktionsoperationen durchgeführt, um die Bedingungen auf einen abschließenden Wert für den Ausdruck zu reduzieren. Es wird darauf hingewiesen, daß dies immer noch den Vorteil gegenüber den bekannten Höhenreduktionstechniken aufweist, nachdem die Reduktionsfunktionen gleichzeitig durchgeführt werden können und nicht als binärer Baum von zwei Eingangs-UND-Operationen.

[0058] Wiederum als weiteres Beispiel können die Reduktionsoperationen gemäß der Erfindung auf Vektoren wirksam sein, von denen jeder eine Mehrzahl von Booleschen Werten umfaßt. Solche Vektoren können als Eingangswerte, Vorhersagewerte und/oder Ergebnisse von Funktionen einer Reduktionsoperation dienen.

Patentansprüche

1. Verfahren zur Auswertung von Booleschen Ausdrücken bestehend aus einer Mehrzahl von Termen, die durch logische Operatoren verknüpft sind, in einem Computersystem (**20**), in dem Operationen entsprechend Befehlen eines Befehlssatzes durch eine oder mehrere Funktionseinheiten (**36-39**) ausgeführt werden,
wobei die Operationen eine oder mehrere Reduktionsoperationen in einer Form einschließen, die einen Wert aus einem Satz von einem oder mehreren Eingangswerten zum Speichern in einem bestimmten Register (**54**) bestimmt, wobei für jeden Typ der logischen Operatoren für jeden Term während eines Zyklus folgende Schritte durchlaufen werden:
Durchführen einer Ergebnisfunktion (F_{aus}) in der Funktionseinheit (**36-39**) bezüglich eines oder mehrerer Ein-

gangswerte der Operation, um einen Ergebniswert zu erzeugen;
Durchführen einer Aktivierungsfunktion (F_{en}) in der Funktionseinheit bezüglich eines oder mehrerer Eingangswerte, um einen Booleschen Aktivierungswert zu erzeugen, wobei der Boolesche Aktivierungswert für zumindest eine Kombination der Eingangswerte falsch ist; und
Schreiben des Ergebniswertes an einen bestimmten Registerort (**58**), wenn der zugehörige Boolesche Aktivierungswert wahr ist.

2. Verfahren nach Anspruch 1, das ferner folgende Verfahrensschritte aufweist:
Ausführen eines Satzes von Reduktionsoperationen, um einen Booleschen Ausdruck auszuwerten, wobei die Ausführung des Satzes von Reduktionsoperationen folgende Schritte aufweist:
Ausführen einer ersten Reduktionsoperation, um einen ersten Booleschen Wert an einen bestimmten Registerort (**58**) zu schreiben; und
Ausführen eines Satzes von zwei oder mehr Reduktionsoperationen, von denen jede einen zweiten Booleschen Wert an einen bestimmten Registerort (**58**) schreibt, wenn der Boolesche Aktivierungswert, der durch die Aktivierungsfunktion der Operation erzeugt wird, wahr ist, wodurch der an dem bestimmten Registerort (**58**) gespeicherte Wert der Wert des Booleschen Ausdrucks ist.

3. Verfahren nach Anspruch 2, das ferner folgenden Verfahrensschritt aufweist: Nachfolgendes Ausführen einer Operation, die auf den Wert, der in dem bestimmten Register (**58**) gespeichert ist, konditioniert ist.

4. Verfahren nach einem der Ansprüche 1 – 3, das ferner folgenden Verfahrensschritt aufweist:
Ausführen eines Satzes von Reduktionsoperationen, um einen Booleschen Ausdruck auszuwerten, wobei die Ausführung des Satzes von Reduktionsoperationen folgende Verfahrensschritte umfaßt:
Ausführen einer ersten Reduktionsoperation, um einen ersten Booleschen Wert an einen bestimmten Registerort (**58**) zu schreiben; und
Ausführen eines Satzes von zwei oder mehr Reduktionsoperationen, von denen jede einen zweiten Booleschen Wert an einen bestimmten Registerort (**58**) schreibt, wenn der Boolesche Aktivierungswert der durch die Aktivierungsfunktion der Operation erzeugt wird, wahr ist, wobei der an dem bestimmten Registerort (**58**) gespeicherte Wert der Wert des Booleschen Ausdrucks ist.

5. Verfahren nach Anspruch 4, das ferner folgenden Verfahrensschritt aufweist: Aufeinanderfolgendes Ausführen einer Operation, die auf den Wert, der in dem bestimmten Register (**58**) gespeichert ist, konditioniert ist.

6. Verfahren nach Anspruch 1, bei dem die Ausführung von zumindest einer der Reduktionsoperationen in der Funktionseinheit (**36-39**) den folgenden Verfahrensschritt umfaßt: Durchführen einer Ergebnisfunktion und einer Aktivierungsfunktion in der Funktionseinheit (**36-39**), die zumindest eine Eingangsvariable gemeinsam haben, wobei der Boolesche Aktivierungswert, der durch die Durchführung der Aktivierungsfunktion erzeugt wird, für zumindest eine Kombination von Eingangswerten falsch ist.

7. Verfahren nach Anspruch 6, das ferner folgenden Verfahrensschritt aufweist:
Ausführen eines Satzes von Reduktionsoperationen, um einen Booleschen Ausdruck auszuwerten, wobei die Ausführung des Satzes von Reduktionsoperationen folgende Schritte umfaßt:
Ausführen einer ersten Reduktionsoperation, um einen ersten Booleschen Wert an einen bestimmten Registerort (**58**) zu schreiben; und
Ausführen eines Satzes von zwei oder mehr Reduktionsoperationen, von denen jede einen zweiten Booleschen Wert an den bestimmten Registerort (**58**) schreibt, wenn der Boolesche Aktivierungswert, der durch die Aktivierungsfunktion der Operation erzeugt wird, wahr ist, wobei der an dem bestimmten Registerort (**58**) gespeicherte Wert der Wert des Booleschen Ausdrucks ist.

8. Verfahren nach Anspruch 7, das ferner folgenden Verfahrensschritt aufweist: Nachfolgendes Ausführen einer Operation, die auf den Wert, der in dem bestimmten Register (**58**) gespeichert ist, konditioniert ist.

9. Verfahren nach Anspruch 1, das ferner folgende Verfahrenschritte aufweist:
Ausführen mehrerer Ausgangsreduktionsoperationen in den Funktionseinheiten (**36-39**), wobei jede der mehreren Ausgangsreduktionsoperationen eine Form hat, die eine Mehrzahl von Werten in einem oder mehreren bestimmten Registern (**58**) aus einem Satz von einem oder mehreren Eingangswerten bestimmt, wobei das Ausführen jeder der mehreren Ausgangsreduktionsoperationen in einer Funktionseinheit (**36-39**) für jeden der Mehrzahl von Werten, die durch die Operation bestimmt werden, folgende Schritte umfaßt:
Durchführen einer Ergebnisfunktion in der Funktionseinheit (**36-39**) mit einem oder mehreren Eingangswerten

der Operation, um einen Ergebniswert zu erzeugen;
Durchführen einer Aktivierungsfunktion in der Funktionseinheit (**36-39**) mit einem oder mehreren Eingangswerten der Operation, um einen Booleschen Aktivierungswert zu erzeugen; und Schreiben des Ergebniswertes an einen bestimmten Registerort (**58**), wenn der Boolesche Aktivierungswert wahr ist.

10. Verfahren nach Anspruch 9, das ferner folgenden Verfahrensschritt aufweist:
Ausführen eines Satzes von Reduktionsoperationen, um einen Booleschen Ausdruck auszuwerten, wobei zumindest eine Reduktionsoperation in dem Satz eine Reduktionsoperation mit mehreren Ausgaben ist und die Ausführung des Satzes von Reduktionsoperationen folgende Schritte umfaßt:
Ausführen einer ersten Reduktionsoperation, um einen ersten Booleschen Wert an einen bestimmten Registerort (**58**) zu schreiben; und
Ausführen eines Satzes von zwei oder mehr Reduktionsoperationen, von denen jede einen zweiten Booleschen Wert an einen bestimmten Registerort (**58**) schreibt, wenn der Boolesche Aktivierungswert, der durch die Aktivierungsfunktion der Operation erzeugt wird, wahr ist, wobei der an dem bestimmten Registerort (**58**) gespeicherte Wert der Wert des Booleschen Ausdrucks ist.

11. Verfahren nach Anspruch 10, das ferner folgenden Verfahrensschritt aufweist: Nachfolgendes Ausführen einer Operation, die auf den Wert, der in dem bestimmten Register (**58**) gespeichert ist, konditioniert ist.

12. Verfahren nach Anspruch 9, bei dem das Ausführen vor zumindest einer der mehreren Reduktionsoperationen in einer Funktionseinheit (**36-39**) für zumindest einen der Werte, der durch die Operation bestimmt wird, folgende Schritte aufweist: Ausführen einer Aktivierungsfunktion in der Funktionseinheit (**36-39**) mit zwei oder mehr Eingangswerten, um einen Booleschen Aktivierungswert zu erzeugen, wobei der Boolesche Aktivierungswert, der durch die Durchführung der Aktivierungsfunktion erzeugt wird, für zumindest eine Kombination der Eingangswerte falsch ist.

13. Verfahren nach Anspruch 12, das ferner folgenden Verfahrensschritt aufweist:
Ausführen eines Satzes von Reduktionsoperationen, um einen Booleschen Ausdruck auszuwerten, wobei zumindest eine Reduktionsoperation in dem Satz eine Reduktionsoperation mit mehreren Ausgaben ist, und die Ausführung des Satzes von Reduktionsoperationen folgende Schritte aufweist:
Ausführen einer ersten Reduktionsoperation, um einen ersten Booleschen Wert an einen bestimmten Registerort (**58**) zu schreiben; und
Ausführen eines Satzes von zwei oder mehr Reduktionsoperationen, von denen jede einen zweiten Booleschen Wert an den bestimmten Registerort (**58**) schreibt, wenn der Boolesche Aktivierungswert, der durch die Aktivierungsfunktion der Operation erzeugt wird, wahr ist, wobei der an dem bestimmten Registerort (**58**) gespeicherte Wert der Wert des Booleschen Ausdrucks ist; und
nachfolgendes Ausführen einer Operation, die auf den Wert, der in dem bestimmten Register (**58**) gespeichert ist, konditioniert ist.

14. Verfahren nach Anspruch 9, bei dem das Ausführen von zumindest einer der Reduktionsoperationen mit mehreren Ausgaben in einer Funktionseinheit zumindest für einen der durch die Operation erzeugten Werte folgenden Verfahrensschritt aufweist: Durchführen einer Ergebnisfunktion und einer Aktivierungsfunktion in der Funktionseinheit (**36-39**), die zumindest einen Eingangswert gemeinsam haben, wobei der Boolesche Aktivierungswert, der durch die Durchführung der Aktivierungsfunktion erzeugt wird, für zumindest eine Kombination der Eingangswerte falsch ist.

15. Verfahren nach Anspruch 14, das ferner folgenden Verfahrensschritt aufweist:
Ausführen eines Satzes von Reduktionsoperationen, um einen Booleschen Ausdruck auszuwerten, wobei zumindest eine Reduktionsoperation in dem Satz eine Reduktionsoperation mit mehreren Ausgaben ist, und die Ausführung des Satzes von Reduktionsoperationen folgende Schritte umfaßt:
Ausführen einer ersten Reduktionsoperation, um einen ersten Booleschen Wert in einen bestimmten, Registerort (**58**) zu schreiben; und
Ausführen eines Satzes von zwei oder mehr Reduktionsoperationen, von denen jede einen zweiten Booleschen Wert in den bestimmten Registerort (**58**) schreibt, wenn der Boolesche Aktivierungswert, der durch die Aktivierungsfunktion der Operation erzeugt wird, wahr ist, wodurch der Wert, der in einem bestimmten Registerort gespeichert ist, der Wert des Booleschen Ausdrucks ist; und
nachfolgendes Ausführen einer Operation, die auf den Wert, der in dem bestimmten Register (**58**) gespeichert ist, konditioniert ist.

Es folgen 2 Blatt Zeichnungen

Anhängende Zeichnungen

FIG. 1

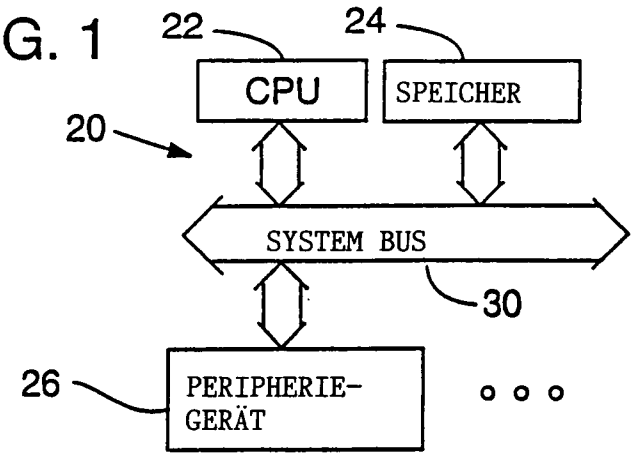
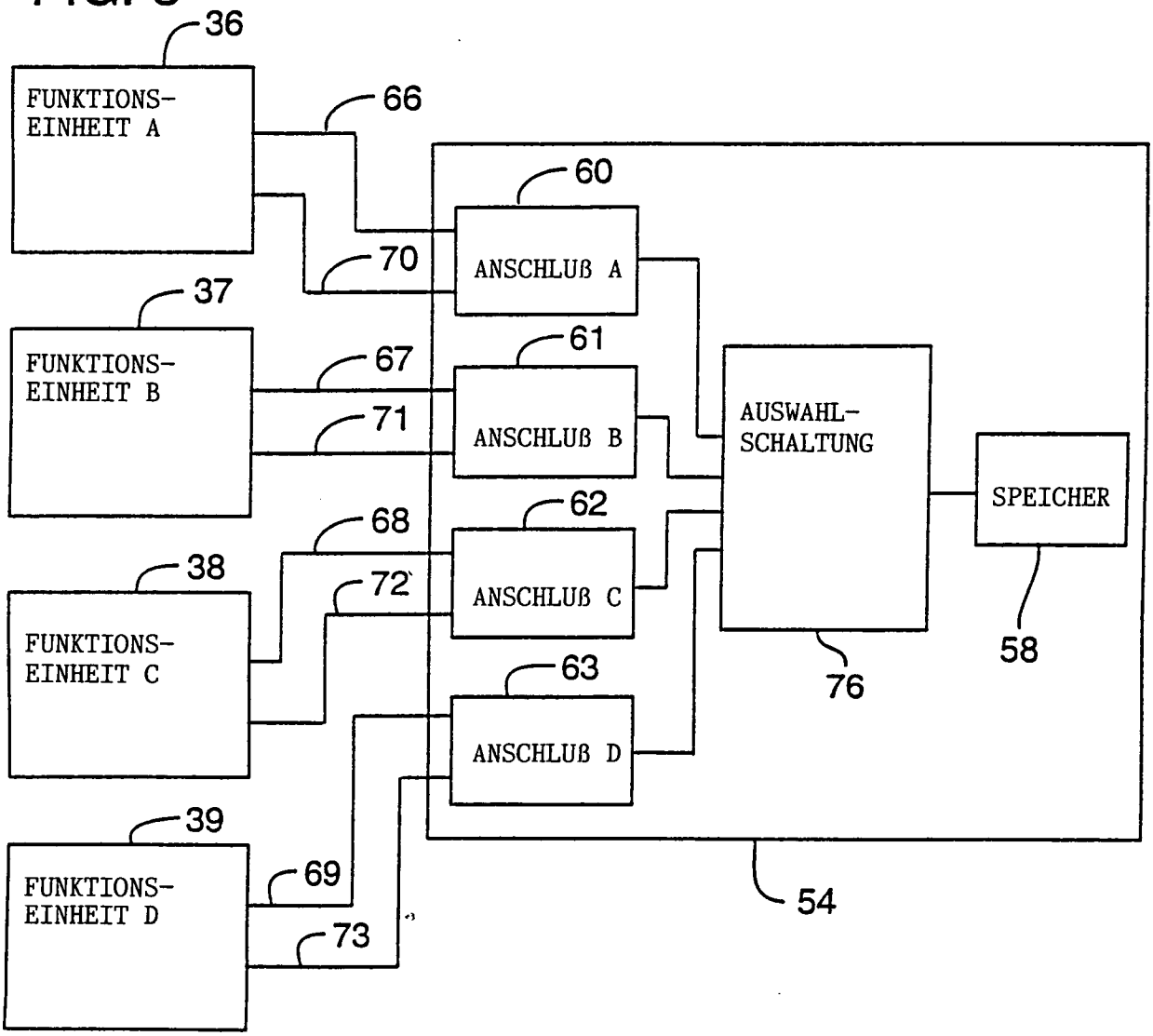


FIG. 3



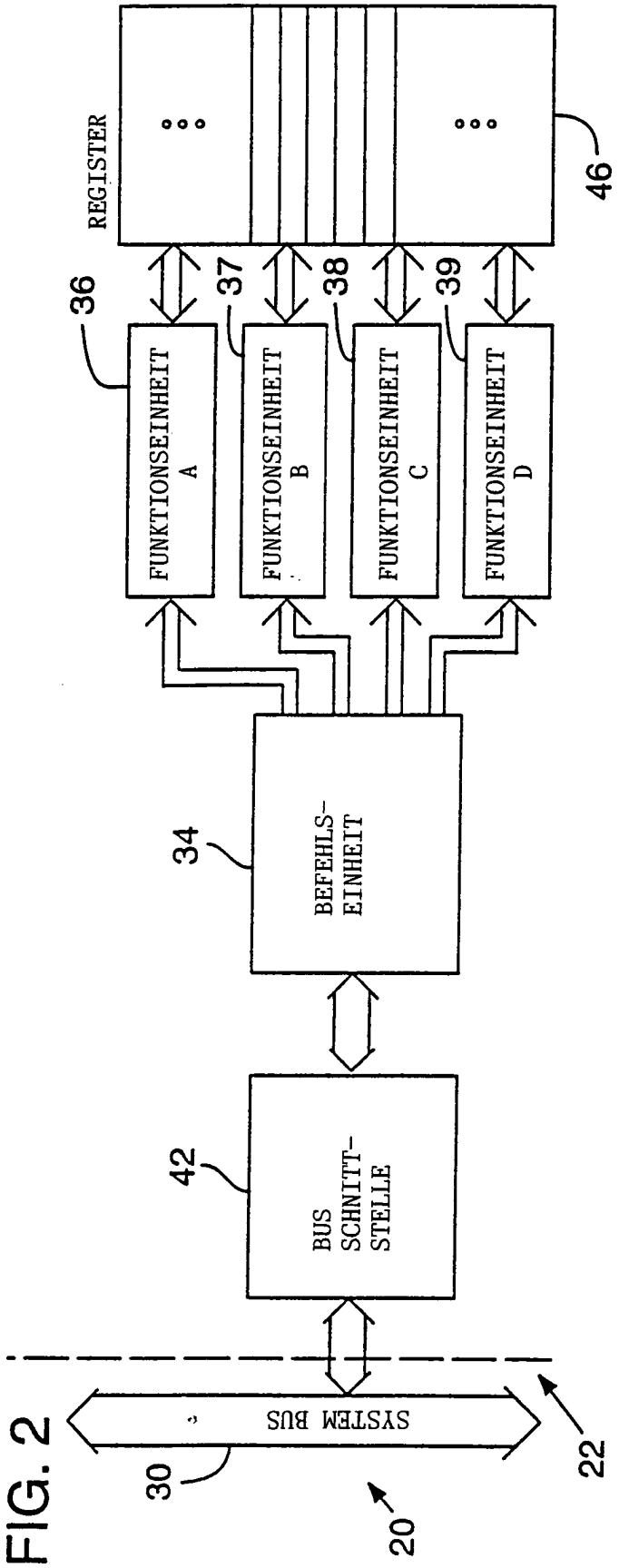

REGISTER
46
36
37
38
39
FUNKTIONSEINHEIT A
FUNKTIONSEINHEIT B
FUNKTIONSEINHEIT C
FUNKTIONSEINHEIT D
34
BEFEHLS-EINHEIT
42
BUS SCHNITT-STELLE
SYSTEM BUS
30
22
20

FIG. 2