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

(10) **DE 600 32 181 T2** 2007.03.29

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 281 118 B1**
(21) Deutsches Aktenzeichen: **600 32 181.9**
(86) PCT-Aktenzeichen: **PCT/US00/27221**
(96) Europäisches Aktenzeichen: **00 965 559.8**
(87) PCT-Veröffentlichungs-Nr.: **WO 2001/088694**
(86) PCT-Anmeldetag: **03.10.2000**
(87) Veröffentlichungstag der PCT-Anmeldung: **22.11.2001**
(97) Erstveröffentlichung durch das EPA: **05.02.2003**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **29.11.2006**
(47) Veröffentlichungstag im Patentblatt: **29.03.2007**

(51) Int Cl.[8]: ***G06F 7/72*** (2006.01)
H03M 7/18 (2006.01)

(30) Unionspriorität:
569944 **12.05.2000** **US**

(84) Benannte Vertragsstaaten:
AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LI, LU, MC, NL, PT, SE

(73) Patentinhaber:
The Athena Group, Inc., Gainesville, Fla., US

(72) Erfinder:
MELLOTT, D., Jonathon, Gainsville, FL 32605, US

(74) Vertreter:
Barz, P., Dipl.-Chem. Dr.rer.nat., Pat.-Anw., 80803 München

(54) Bezeichnung: **VERFAHREN UND ANORDNUNG ZUR AUSFÜHRUNG VON BERECHNUNGEN MIT RESIDUENARITHMETIK**

Beschreibung

[0001] Die vorliegende Erfindung erfolgte mit Unterstützung der Regierung unter einem von dem National Institute Standards and Technology Cooperative Agreement Nr. F0NANB7H3021 unterstützten Forschungsprojekt. Die Regierung kann bestimmte Rechte an der vorliegenden Erfindung besitzen.

Allgemeiner Stand der Technik

[0002] Die vorliegende Erfindung betrifft ein Verfahren und eine Vorrichtung zur Durchführung von Berechnungen unter Verwendung der Restklassenarithmetik/Residuumarithmetik. Das vorliegende Verfahren und die vorliegende Vorrichtung können das Restklassensystem/Residuumsystem (RNS) zur Implementierung von Maschinerie zur automatischen Berechnung verwenden. Die Verwendung des RNS wurde vorgeschlagen in Garner, H. L., „The Residue Number System", IRE Transactions on Electronic Computers, Band EL-8, Nr. 6, Juni 1959, Seiten 140–147, und in Taylor, F. J., „Residue Arithmetic: A Tutorial with Examples", IEEE Computer, Band 17, Nr. 5, Mai 1984, Seiten 50–61. Mit dem RNS implementiert man im Allgemeinen automatische Berechnungsmaschinerie für die digitale Signalbearbeitung. Die digitale Signalverarbeitung (DSP) wird durch die sich wiederholende Berechnung von Summen von Produkten dominiert. Das RNS eignet sich gut für die Durchführung von Berechnungen dieser Art, was in den folgenden Literaturstellen demonstriert wurde: Mellott, J. D., Lewis, M. P., Taylor, F. J., „A 2D DFT VLSI Processor and Architecture", Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, Atlanta, 1996 und Mellott, J. D., Smith, J. C., Taylor, F. J., „The Gauss Machine – A Galois-Enhanced Quadratic Residue Number System Systolic Array", Proceedings of IEEE 11th Symposium on Computer Arithmetic, Windsor Ontario, 1993, Seiten 156–162.

[0003] Die Implementierung von digitalen Signalprozessoren auf großem Maßstab unter Verwendung eines einzigen Halbleiterbausteins war in der Vergangenheit häufig aufgrund der Beschränkungen bezüglich der Menge an Logik, die auf einem solchen Baustein platziert werden kann, nicht praktikabel. Stattdessen wurde die Implementierung von digitalen Signalprozessoren auf großem Maßstab in der Regel unter Verwendung diskreter Logik durchgeführt. Das RNS eignet sich gut für diese Implementierungsmethodologie, da seine Erfordernis von kleinen Addierern und Tabellennachschlagefunktionen der großen Verfügbarkeit diskret gekapselter kleinerer Addierer und kleiner programmierbarer Nurlesespeicher (PROMs) entspricht. Ein Beispiel für diese Implementierungsmethodologie ist die in der oben erwähnten Literaturstelle von Mellott et al. besprochene Gauß-Maschine. Da es möglich wurde, digitale Signalprozessoren auf großem Maßstab auf einen einzigen Halbleiterbaustein zu integrieren, wurde die Methodologie der Verwendung kleiner Addierer und Speicher weitergeführt. Ein Beispiel für einen solchen digitalen Signalprozessor findet sich in Smith, J. C., Taylor, F. J., „The Design of a Fault Tolerant GE-QRNS Processing Element for Linear Systolic Array DSP Applications", Proceedings of IEEE Great Lakes Symposium on VLSI, Notre Dame, Indiana, 1994. Weitere Beispiele für digitale RNS-Signalprozessoren finden sich in dem US-Patent Nr. 5,117,383 (Fujita et al.), ausgegeben am 26.5.1992; in dem US-Patent Nr. 5,008,668 (Takayama, et al.), ausgegeben am 16.4.1991, in dem US-Patent Nr. 4,949,294 (Wambergue), ausgegeben am 14.8.1990; und in dem US-Patent Nr. 4,281,391 (Huang), ausgegeben am 28.7.1981.

[0004] Die obigen Beispiele beschreiben die Verwendung von ROMs zur Implementierung von Tabellennachschlagefunktionen. Für die typischerweise bei digitalen RNS-Signalprozessorimplementierungen anzutreffenden kleinen Tabellennachschlagefunktionen sind ROMs attraktiv, weil sie leicht zu programmieren sind und bekannte Geschwindigkeits-, Flächen- und Stromversorgungskenngrößen aufweisen. Im Gegensatz dazu kann der manuelle Entwurf einer Ansammlung von Logikgattern zur Realisierung einer Tabellennachschlagefunktion eine enorme Aufgabe sein, und die Geschwindigkeits-, Flächen- und Stromversorgungskenngrößen sind im Allgemeinen erst nach dem Entwurf der Schaltung voll bekannt. Ein weiteres mit der vorbekannten Verwendung von ROMs in integrierten im Gegensatz zu diskreten digitalen RNS-Signalprozessorimplementierungen assoziiertes Merkmal besteht darin, dass ROMs im Vergleich zu anderen möglichen Mitteln zur Implementierung kleiner Nachschlagetabellen vorzuziehende Chipfläche bieten.

[0005] Vorbekannte Techniken zur Durchführung von Berechnungen unter Verwendung des RNS haben einen oder mehrere Nachteile in Bezug auf die Verwendung von Speichern (gewöhnlich ROMs) zur Implementierung der Tabellennachschlagefunktionen. Zu diesen Nachteilen sind die folgenden zu zählen: Speicher mit den erforderlichen Eigenschaften für die Verwendung bei RNS-Berechnungen sind nicht in allen ASIC-Implementierungstechnologien in ausreichender Menge verfügbar; Speicher enthalten häufig analoge Schaltkreise, die signifikant viel Strom verbrauchen, auch wenn keine Schaltaktivität in der Schaltung besteht; die in den meisten Speicherbausteinen anzutreffenden analogen Schaltkreise können nicht gut auf tiefe Submikrometer-Halbleiterherstellungstechnologien skaliert werden; da sie von analogen Schaltungen (z.B. Differenzverstärkern) abhängen, können Speicher schwieriger als digitale Logikschaltungen zu prüfen sein, können separate Prüfungen und Prüfmechanismen im Vergleich zu di-

gitalen Logikschaltungen erfordern und sind im Allgemeinen nicht mit Testmethodologien für Leckstrom (I_{DDQ}) kompatibel; es besteht wenig oder keine Flexibilität zur Optimierung eines Speichers mit Bezug auf Geschwindigkeit, Stromversorgung und/oder Fläche; der Pipeline-Betrieb von Speichern kann schwierig sein, und in vielen Implementierungstechnologien besteht keine realistische Möglichkeit für einen Pipeline-Betrieb von Speicher; die Größe des Speichers wird in der Regel durch die Anzahl der Eingänge und Ausgänge festgelegt und ist im Wesentlichen vom Inhalt des Speichers unabhängig; aus Zuverlässigkeitsgründen dürfen nicht mit einem Speicher zusammenhängende Leitungen gewöhnlich nicht über einen Speicher auf einem Halbleiterbaustein verlaufen, sodass die Anwesenheit vieler kleiner Speicher auf einem Speicherbaustein, so wie sie in einer Vorrichtung zur Durchführung von Berechnungen unter Verwendung des RNS verwendet würden, die Möglichkeit zur Verbindung verschiedener Funktionen (sowohl Speicher als auch Nicht-Speicher) auf dem Baustein beeinträchtigen kann.

[0006] Eine vorbekannte Technik, die die Verwendung von Volladdierern anstelle von ROMs oder PLAs zur Durchführung von Berechnungen unter Verwendung des RNS lehrt, findet sich in Seon Wook Kim et al., „New Implementations of Converters for the Residue and Quadratic Residue Number System", Proceedings of the International Symposium on Circuits and Systems, US, New York, IEEE, Band. Symp. 24, 11.6.1991 (1991-06-11), Seiten 2959–2962, XP000299331 ISBN: 0-7803-0050-5.

Kurzfassung der Erfindung

[0007] Die vorliegende Erfindung betrifft ein Verfahren und eine Vorrichtung zum Durchführen von Berechnungen unter Verwendung des Residuumsystems (RNS). Gemäß einem ersten Aspekt der Erfindung wird ein Verfahren zum Durchführen mathematischer Berechnungen nach Anspruch 1 der angefügten Ansprüche bereitgestellt. Entsprechende Vorrichtungen gemäß einem zweiten Aspekt der Erfindung werden nach Anspruch 75 der angefügten Ansprüche bereitgestellt. In einer spezifischen Ausführungsform kann eine Vielzahl von Logikgattern verwendet werden, um Berechnungen unter Verwendung des RNS zu implementieren. Im Hinblick auf neuere Änderungen der Halbleiterbausteinskalierung und Entwurfsmethodologie kann die vorliegende Erfindung gegenüber der Verwendung von ROMs für kleine Tabellennachschlagefunktionen in integrierten digitalen RNS-Signalprozessorimplementierungen Vorteile bieten. Einige dieser Vorteile wären zum Beispiel: Logikgatter können besser als analoge Teile der ROM-Schaltkreise, wie zum Beispiel der Differenzleseverstärker, größenmäßig herunterskaliert werden; für integrierte RNS-Implementierungen erfordern mit Gattern implementierte kleine Tabellennachschlagefunktionen weniger Chipfläche als die selben, mit ROMs implementierten Funktionen; im Allgemeinen sind Logikgatter mit Ruhestrom-Prüfmethodologien kompatibel, während Speicherbausteine mit derzeitigen Prüfmethodologien für Ruhe- oder Leckstrom (die auch als I_{DDQ}-Prüfung bekannt sind) nicht kompatibel sind; Logikgatter sind im Allgemeinen scan-prüfbar, während Speicherbausteine spezielle Prüfstrukturen erfordern können und in der Regel nicht direkt mit Scan-Prüfmethodologien kompatibel sind; und Signalleitungen können über Logikgatter geroutet werden, während die meisten Entwurfsmethodologien nicht erlauben, Signalleitungen über Onchip-Speicher zu routen, sodass die Anwesenheit vieler kleiner Speicher in einem Entwurf die Leitungsführung stauen kann, was potentiell zu größeren Entwurfskosten, langsamerem Schaltungebetrieb, größerem Stromverbrauch, größerem Silizium-Chipflächenverbrauch und somit größeren Herstellungskosten führt.

[0008] Die vorliegende Erfindung kann einen oder mehrere der folgenden Vorteile liefern: Bereitstellung eines Mittels zum Implementieren von Residuumarithmetik-Rechenschaltkreisen mit verringerter Verwendung oder ganz ohne Verwendung von Speichern für Tabellennachschlageoperationen, sodass die Schaltkreise leicht unter Verwendung vielfältiger Technologien implementiert werden können, darunter u.a. angepasste digitale Logik, Standardzellenlogik, auf Zellen basierende Logik-Arrays, Gate-Arrays, am Einsatzort programmierbare Gate-Arrays und programmierbare Logikbausteine; Bereitstellung eines Mittels zur Implementierung von Residuumarithmetik-Rechenschaltkreisen, die bei Abwesenheit von Schaltaktivität in der Schaltung keinen signifikanten Strom verbrauchen; Bereitstellen eines Mittels zum Implementieren von Residuumarithmetik-Rechenschaltkreisen, die direkt auf tiefe Submikrometer-Halbleiterherstellungstechnologien skaliert werden können; Bereitstellen eines Mittels zum Implementieren von Residuumarithmetik-Rechenschaltkreisen, die mit Standard-Logikprüfmethodologien (z.B. Scan, I_{DDQ}) kompatibel sind; Bereitstellen eines Mittels zum Optimieren der mathematischen Funktionen in den Residuumarithmetik-Rechenschaltkreisen für Geschwindigkeit, Strom und/oder Fläche; Bereitstellen eines Mittels zum Implementieren der mathematischen Funktionen in Residuumarithmetik-Rechenschaltkreisen, das Pipelining ermöglicht und völlig mit Methodologien der elektronischen Entwurfsautomatisierung (EDA) für automatisches Pipelining kompatibel ist; Bereitstellen eines Mittels zum Implementieren der mathematischen Funktionen in Residuumarithmetik-Rechenschaltkreisen, das die Struktur der sich aus einer mathematischen Funktion resultierenden Werte ausnutzt, um eine Implementierung zu produzieren, die kleiner und schneller als mit einer beliebigen auf Speicher basierenden Implementierung möglich ist; und Bereitstellen eines Mittels zum Implementieren von mathematischen Funk-

tionen in den Residuumarithmetik-Rechenschaltkreisen, das das Routen von Leitungen auf dem Halbleiterbaustein nicht zu sehr stört.

Kurze Beschreibung der Zeichnungen

[0009] Fig. 1 zeigt ein Blockschaltbild einer Vorrichtung zur digitalen Signalverarbeitung, die Residuumarithmetik zum Operieren an reellen Operanden verwendet und reelle Ergebnisse produziert.

[0010] Fig. 2 zeigt ein Blockschaltbild einer Vorrichtung zur digitalen Signalverarbeitung, die Residuumarithmetik zum Operieren an komplexen Operanden verwendet und reelle Ergebnisse produziert.

[0011] Fig. 3 zeigt ein Blockschaltbild eines modularen Produkttabellennachschlagens für einen Konstanten-Multiplizierer.

[0012] Fig. 4 zeigt eine Tabelle für das Produkt von zwei und einer modulo-5-Variablen modulo 5, der Minimierung der Gleichungen für die Tabelle unter Verwendung von Karnaugh-Abbildungen, wobei eine Vielzahl von Logikgattern die reduzierten Gleichungen implementiert, und die resultierende Tabelle.

[0013] Fig. 5 zeigt eine zahlentheoretische Logarithmus-Nachschlagetabelle.

[0014] Fig. 6 zeigt ein Blockschaltbild eines Multiplizierers, der Produkte unter Verwendung der zahlentheoretischen Logarithmen der Operanden berechnet.

[0015] Fig. 7 zeigt eine Struktur zur Berechnung des Residuums einer vorzeichenlosen N-Bit- oder Zweierkomplementzahl.

[0016] Fig. 8 zeigt ein Blockschaltbild eines Mehrfachoperanden-Modular-Addiererbaums.

[0017] Fig. 9 zeigt eine Struktur zum Umsetzen eines Werts aus der RNS-Darstellung in die Binärdarstellung unter Verwendung des chinesischen Residuumtheorems.

[0018] Fig. 10 zeigt eine Struktur zum Umsetzen eines Werts aus der RNS-Darstellung in die Binärdarstellung unter Verwendung des L-CRT-Algorithmus.

[0019] Fig. 11 zeigt eine Struktur zum Umsetzen eines komplexen RNS-Werts in einen QRNS-Wert.

[0020] Fig. 12 zeigt eine Struktur zum Umsetzen eines QRNS-Werts in einen komplexen RNS-Wert.

Ausführliche Beschreibung der Erfindung

Ermöglichende mathematische Theorie

[0021] Die folgenden Unterabschnitte präsentieren die Mathematik, die für die Funktionsweise der Erfindung relevant ist. Obwohl die Mathematik wohlbekannt ist, wird die Theorie hier präsentiert, um so einen einheitlichen Rahmen der Notation und Symbole bereitzustellen.

Das chinesische Residuumstheorem

[0022] Es sei $S = \{p_0, p_1, p_2, ..., p_{L-1}\}$ mit $\gcd(p_1, p_j) = 1$ für alle $i, j \in \{0, 1, 2, ..., L - 1\}$ und $i \neq j$, wobei gcd für den größten gemeinsamen Nenner steht. Es sei $M = \Pi_{i=0}^{L-1} P_i$, und es sei $X \in Z/MZ$, wobei Z den Ring der ganzen Zahlen bedeutet. Nach dem chinesischen Residuumstheorem existiert ein Isomorphismus

$$\varphi: Z/MZ \rightarrow Z/p_0Z \times Z/p_1Z \times Z/p_2Z \times ... \times Z/p_{L-1}Z.$$

[0023] Die Abbildung φ wird gegeben durch

$$\varphi(X) \rightarrow (x_0, x_1, x_2, ..., x_{L-1})$$

mit $(x_0, x_1, x_2, ..., x_{L-1}) \in Z/p_0Z \times Z/p_1Z \times Z/p_2Z \times ... \times Z/p_{L-1}Z$, und $x_i \equiv X \pmod{p_i}$ für alle $i \in \{0, 1, 2, ..., L - 1\}$. Die Umkehrabbildung wird gegeben durch

$$\varphi^{-1}[(x_0, x_1, x_2, ..., X_{L-1})] \rightarrow X$$

mit

$$X \equiv \left(\sum_{i=0}^{L-1} m_i \langle m_i^{-1} x_i \rangle_{p_i} \right) \pmod{M},$$

$m_i = M/p_i$, $m_i m_i^{-1} \equiv 1 \pmod{p_i}$ und $(x)_p$ bedeutet den Wert in der Menge $\{0, 1, 2, ..., p - 1\}$, der zu x modulo p kongruent ist.

Zahlentheoretische Logarithmen

[0024] Wenn p_i prim ist, existiert ein Generator $a_i \in Z/p_iZ$, sodass

$$\{\alpha_i^k | k = 0, 1, 2, ..., p_i - 2\} = \{1, 2, 3, ..., p_i - 1\}$$

im Ring Z/p_iZ. Im Fall $x_i \in (Z/p_iZ)\backslash\{0\}$ existiert ein eindeutiges

$$l_{x_i} \in \mathbb{Z}/(p_i - 1)\mathbb{Z},$$

sodass

$$x_i \equiv \alpha_i^{l_{x_i}} \pmod{p_i}$$

[0025] Man sagt, dass der Wert

l_{x_i}

der zahlentheoretische Logarithmus von x_i mit der Basis α_i modulo p_i ist.

[0026] Der zahlentheoretische Logarithmus kann ausgenutzt werden, um Produkte in dem Ring Z/p_iZ zu berechnen. Im Fall $x_i, y_i \in (Z/p_iZ)\backslash\{0\}$ existiert ein eindeutiges

$$l_{x_i}, l_{y_i} \in Z/(p_i-1)Z,$$

sodass

$$x_i y_i \equiv \left(\alpha_i^{l_{x_i}}\right)\left(\alpha_i^{l_{y_i}}\right) (\mathrm{mod}\ p_i)$$

$$x_i y_i \equiv \alpha_i^{\left\langle l_{x_i}+l_{y_i}\right\rangle_{p_i-1}} (\mathrm{mod}\ p_i)$$

$$x_i y_i \equiv f_{\alpha_i}\left(\left\langle l_{x_i}+l_{y_i}\right\rangle_{p_i-1}\right) (\mathrm{mod}\ p_i)$$

[0027] Wenn x_i, y_i oder beide null sind, ist das Produkt x_iy_i null.

Komplexe Arithmetik

[0028] Es sei $Z[j]/(j^2 + 1)$ der Ring der Gaußschen ganzen Zahlen unter den gewöhnlichen Operationen von Addition und Multiplikation, und Zahlen der Form $a + jb$ mit $a, b \in Z$ und $j^2 = –1$. Dann bedeutet $(Z[j]/(j^2 + 1))/p_iZ$ den Ring der Gaußschen ganzen Zahlen modulo p_i und im Fall $a + jb \in Z[j]/(j^2 + 1)$ wird die Abbildung φ: $Z[j]/(j^2 + 1) \rightarrow (Z[j]/(j^2 + 1))/p_iZ$ gegeben durch

$$\varphi((a + jb)) \rightarrow a_i + jb_i,$$

mit $a_i \equiv a \pmod{p_i}$ und $b_i \equiv b \pmod{p_i}$. Die Menge $(Z[j]/(j^2 + 1))/p_iZ$ ist ein Ring unter den gewöhnlichen komplexen arithmetischen Operationen der Multiplikation und Addition. Das heißt, im Fall $(a_i + jb_i)$, $(c_i + jd_i) \in (Z[j]/(j^2 + 1))/p_iZ$, gilt

$$(a_i + jb_i) + (c_i + jd_i) = ((a_i + c_i) + j(b_i + d_i))$$

$$(a_i + jb_i) \times (c_i + jd_i) = ((a_ic_i – b_id_i) + j(a_id_i + b_ic_i)).$$

[0029] Man nehme an, dass p_i eine Primzahl ist und $p_i = 4k_i + 1$ mit $k_i \in Z$. Dann existiert ein Isopmorphismus zwischen den Gaußschen ganzen Zahlen modulo p_i unter den gewöhnlichen komplexen arithmetischen Operationen wie oben gezeigt und den Gaußschen ganzen Zahlen modulo p_i unter komponentenweiser Addition und Multiplikation Ψ: $(Z[j]/(j^2 + 1))/p_iZ \rightarrow (Z[j]/(j^2 + 1))/p_iZ$, mit der Abbildung

$$\Psi((a_i + jb_i)) \rightarrow (z_i, z_i^*)$$

mit $z_i = a_i + \hat{\jmath}b_i$, $z_i^* = a_i – \hat{\jmath}b_i$ und $\hat{\jmath}^2 \equiv –1 \pmod{p_i}$.

[0030] Die Umkehrabbildung wird gegeben durch

$$\Psi^{-1}((z_i, z_i^*) \rightarrow (a_i + jb_i)$$

mit $a_i = 2^{-1}(z_i + z_i^*)$, $b_i = \hat{\jmath}2^{-1}(z_i – z_i^*)$ und $2 \cdot 2^{-1} \equiv 1 \pmod{p_i}$.

[0031] Das chinesische Residuumtheorem (CRT) kann ausgenutzt werden, um Addition, Subtraktion und Multiplikation von Werten im Ring der ganzen Zahlen modulo M, Z/MZ durchzuführen, indem man die Berechnung in L unabhängige Berechnungen in Z/p_iZ, für $i \in \{0, 1, 2, ..., L – 1\}$ zerlegt. Wenn jedes $p_i \in S$ prim ist, kann man die zahlentheoretischen Logarithmen ausnutzen, um die Komplexität der Multiplikation zu verringern. Wenn ferner jedes $p_i \in S$ prim und $p_i = 4k_i + 1$ mit $k_i \in Z$ ist, ist es möglich, den Isomorphismus Ψ auszunutzen, um die Anzahl der erforderlichen arithmetischen Operationen zur Implementierung der komplexen Multiplikation von vier reellen Multiplikationen und zwei reellen Additionen auf zwei reelle Multiplikationen zu reduzieren.

[0032] Fig. 1 zeigt eine spezifische Ausführungsform der vorliegenden Erfindung, die zur Durchführung von Summen von Produkten an reellen binären vorzeichenlosen oder Zweierkomplement-, Einerkomplement-, Vorzeichen-Betrag- oder anderen Operanden mit fester Basis oder Fließbasis unter Verwendung von Residuumarithmetik verwendet werden kann. Das in Fig. 1 gezeigte System kann eine Schaltung **1** zum Umsetzen von Daten aus einer herkömmlichen Darstellung wie zum Beispiel, aber ohne Einschränkung, Einerkomplement, Vorzeichen-Betrag, vorzeichenlose Binärdarstellung oder Zweierkomplement, in eine Menge von L-Residuum aufweisen. Wenn Multiplikation benötigt wird, kann das Residuum der Eingangsoperanden durch eine Schaltung **3** mit einem oder mehreren Koeffizienten multipliziert werden. Die Schaltung **3** kann entfernt werden, wenn nur Addition erzielt werden soll. Diese Koeffizienten können feste und/oder programmierte Koeffizienten sein. Die von der Schaltung **3** produzierten modularen Produkte können dann durch eine Schaltung **4** addiert werden, um modulare Summen von Produkten zu produzieren. Die modularen Summen von Produkten können dann durch eine Schaltung **6** in eine herkömmliche Darstellung umgesetzt werden. Die spezifische Anordnung der modularen Produkte und Summen ist von dem Algorithmusentwurf abhängig und kann nach Wunsch optimiert werden.

[0033] Mit Bezug auf Fig. 1 ist eine Ausführungsform gezeigt, die reelle Operanden verarbeiten kann. Datenoperanden zum Beispiel in einem herkömmlichen Format wie etwa Zweierkomplement können in die Schaltung **1** (deren Einzelheiten in der Besprechung von Fig. 7 zusammengefasst werden) eingegeben werden, um die Operanden in RNS-Form um-

zusetzen. Wenn der Algorithmus Multiplikation erfordert, können die Produkte als nächstes durch eine Schaltung **3** berechnet werden, die ein oder mehrere Elemente aus Fig. 3 und/oder Fig. 5 und Fig. 6 umfassen kann. Etwaige erforderliche Summen können als nächstes durch eine Schaltung **4** berechnet werden, die zwei Operanden-Modularaddierer und wahlweise einen oder mehrere Modular-Addierer-Bäume aus Fig. 8 umfasst. Die spezifische Anordnung der arithmetischen Elemente und Zwischenspeicherelemente, darunter, aber ohne Einschränkung, Register, Haltespeicher und Direktzugriffsspeicher (RAMs) kann abhängig von der Situation variiert werden. Zum Beispiel können die arithmetischen Elemente und Zwischenspeicherelemente dafür angeordnet werden, Funktionen wie zum Beispiel, aber ohne Einschränkung, Faltung, Korrelation, nichtrekursive Filter, schnelle Fouriertransformationen, diskrete Kosinustransformationen, Wavelet-Transformationen, Filterbanken, kaskadierte Integrierer-Kammfilter, digitale Empfänger und digitale Sender zu implementieren. Die Ergebnisse der Berechnung können durch eine Schaltung **6**, die zum Beispiel eine CRT-Umsetzung wie in Fig. 9 gezeigt oder eine L-CRT-Umsetzung wie in Fig. 10 gezeigt umfassen kann, in ein herkömmliches Format wie etwa Zweierkomplement umgesetzt werden.

[0034] Fig. 2 zeigt eine weitere spezifische Ausführungsform der vorliegenden Erfindung, die zur Durchführung von Summen von Produkten an komplexen binären vorzeichenlosen oder Zweierkomplement-Operanden unter Verwendung der Residuumarithmetik verwendet werden kann. Das System in Fig. 2 kann eine Schaltung **1** aufweisen, um Daten aus einer herkömmlichen Darstellung wie etwa, aber ohne Einschränkung, Einerkomplement, Vorzeichen-Betrag; vorzeichenlos binär oder Zweierkomplement, in eine Menge von L-Residuum für jede der reellen und imaginären Komponenten jedes Operanden umzusetzen. Das komplexe Residuum kann dann durch eine Schaltung **2** in die quadratische Residuumdarstellung umgesetzt werden. Das quadratische Residuum der Eingangsoperanden kann durch eine Schaltung **3** mit einem oder mehreren Koeffizienten multipliziert werden. Diese Koeffizienten können feste und/oder programmierte Koeffizienten sein. Die von der Schaltung **3** produzierten modularen Produkte können dann durch eine Schaltung **4** addiert werden, um modulare Summen von Produkten zu produzieren. Die quadratischen modularen Summen von Produkten können dann durch eine Schaltung **5** in komplexe Residuen umgesetzt werden. Die komplexen Summen von Produkten können dann durch eine Schaltung **6** in eine herkömmliche Darstellung, wie zum Beispiel komplexvorzeichenlos binär oder Zweierkomplement, umgesetzt werden. Die spezifische Anordnung der modularen Produkte und Summen hängt von dem Algorithmusentwurf ab und kann nach Wunsch optimiert werden. In bestimmten Fällen kann ein Algorithmus dafür ausgelegt werden, reelle Eingaben als Operanden anzunehmen und komplexe Ergebnisse zu produzieren, oder komplexe Eingaben anzunehmen und reelle Ergebnisse zu produzieren. In einem solchen Fall können Schaltung **2** und/oder Schaltung **5** nach Wunsch entfernt werden.

[0035] Mit Bezug auf die in Fig. 2 gezeigte Ausführungsform kann die vorliegende Erfindung komplexe Operanden verarbeiten. Zum Beispiel können Datenoperanden in herkömmlicher Form, wie etwa Zweierkomplement, in die Schaltung **1** (deren Einzelheiten in der Besprechung von Fig. 7 zusammengefasst werden) eingegeben werden, um die Operanden in CRNS-Form umzusetzen. Die CRNS-Operanden können zu einer Schaltung **2** geleitet werden, um die Operanden in das QRNS-Format umzusetzen. Ein Beispiel für eine solche Schaltung **2** ist in Fig. 11 gezeigt. Wenn der Algorithmus Multiplikation erfordert, können die Produkte als nächstes durch eine Schaltung **3** berechnet werden, die ein oder mehrere Elemente aus Fig. 3 und/oder aus Fig. 5 und Fig. 6 umfassen kann. Etwaige erforderliche Summen können als nächstes durch eine Schaltung **4** berechnet werden, die zwei Operanden-Modularaddierer und wahlweise einen oder mehrere modulare Addiererbäume wie in Fig. 8 gezeigt umfassen kann. Die spezifische Anordnung der arithmetischen Elemente und Zwischenspeicherelemente, darunter, aber ohne Einschränkung, Register, Haltespeicher und RAMs, kann abhängig von der Situation variiert werden. Zum Beispiel können die arithmetischen Elemente und Zwischenspeicherelemente dafür angeordnet werden, Funktionen wie etwa, aber ohne Einschränkung, Faltung, Korrelation, nicht rekursive Filter, schnelle Fourier Transformationen, diskrete Kosinustransformationen, Wavelength-Transformationen, Filterbanken, kaskadierte Integrierer-Kammfilter, digitale Empfänger und digitale Sender zu implementieren. Die QRNS-Ergebnisse der Berechnung können dann durch eine Schaltung **5**, wie zum Beispiel in Fig. 12 gezeigt, wieder in die CRNS-Darstellung umgesetzt werden. Die CRNS-Ergebnisse können durch eine Schaltung **6**, die zum Beispiel eine in Fig. 9 gezeigte CRT-Umsetzung oder eine in Fig. 10 gezeigte L-CRT-Umsetzung umfassen kann, in ein herkömmliches Format wie etwa Zweierkomplement umgesetzt werden.

[0036] Fig. 3 zeigt eine Ausführungsform zur Berechnung modularer Produkte einer Konstanten und eines modularen Datenoperanden. Das Produkt kann durch eine Schaltung **7** erzeugt werden, die einen N_i-Bitoperanden annimmt und das Produkt des Operanden und einer Konstanten c_i modulo p_i produziert, wodurch ein N_i-Bit-Ergebnis produziert wird. Fig. 3 zeigt ein Blockschaltbild einer Ausführungsform einer Schaltung **7** zum Produzieren des modularen Produkts eines Operanden und einer Konstanten, wobei eine solche Konstante durch den Entwurf

der Schaltung festgelegt wird. Die Schaltung **7** kann eine Vielzahl von Logikgattern verwenden, die ausgewählt wird, indem man zuerst den Wert einer Funktion der Multiplikation mit einer Konstanten für jeden möglichen modularen Datenoperanden berechnet und die logischen Gleichungen, extrahiert, die die berechneten Werte der Funktion der Multiplikation mit einer Konstanten repräsentieren. Die logischen Gleichungen können dann auf eine Vielzahl von Logikgattern abgebildet werden. Gegebenenfalls können die logischen Gleichungen vor der Abbildung auf eine Vielzahl von Logikgattern zum Beispiel unter Verwendung wohlbekannter Logikminimierungstechniken, die den Umstand ausnutzen, dass für eine beliebige ungültige Eingabe der Wert der Ausgabe ein beliebiger Wert sein darf, minimiert werden. Nachdem die Logikgleichungen auf eine minimierte logische Funktion reduziert wurden, können sie auf eine Implementierung abgebildet werden, die eine Vielzahl von Logikgattern verwendet. Die Abbildung auf eine Vielzahl von Logikgattern kann zum Beispiel manuell durchgeführt werden, oder unter Verwendung von Software, wie etwa DESIGN COMPILER, erhältlich von Synopsys, Inc. in Mountain View, Kalifornien.

[0037] Fig. 4 zeigt ein Beispiel für eine Produktnachschlagetabelle für den Konstanten-Multiplizierer **2** und einen modulo-5-Wert x (die Bit x_2, x_1 und x_0 werden von höchstwertig zu niedrigstwertig geordnet). Eine Wahrheitstabelle **33** zeigt alle möglichen Eingaben in die Tabelle sowie die Ausgabe der Tabelle y (die Bit y_2 y_1 und y_0 werden von höchstwertig zu niedrigstwertig geordnet). Die „x"-Einträge in der Tabelle geben an, dass der Wert der Ausgabe beliebig sein kann. Die Tabelle wird unter Verwendung von Karnaugh-Abbildungen **34A**, **34B** und **34C** auf eine minimale Menge logischer Gleichungen **35A** reduziert. Ein Beispiel für eine Vielzahl von Logikgattern **35B**, womit die logischen Gleichungen **35A** implementiert werden können, ist in Fig. 4 gezeigt. Für größere Moduli und somit größere Tabellen kann die Minimierung der logischen Gleichungen für die Tabelle durch manuelle Mittel impraktikabel sein, sodass ein Computerprogramm verwendet werden kann, um die logischen Gleichungen zu minimieren. Die Ergebnisse der minimierten logischen Gleichungen, wenn alle möglichen Eingaben gegeben sind, sind in einer Wahrheitstabelle **36** gezeigt.

[0038] Fig. 5 zeigt eine Ausführungsform zur Berechnung zahlentheoretischer Logarithmen für eine gegebene Basis α_i und einen Modul p_i. Um zwei Operanden im RNS zu multiplizieren, kann eine Schaltung **8** wie in Fig. 5 gezeigt die Logarithmen der Operanden berechnen. Der Logarithmus kann durch eine Schaltung **8** erzeugt werden, der einen N_i-Bit-Operanden annimmt und den N_i-Bit-Logarithmus des Operanden produziert. Wenn der Eingangsoperand null ist, ist die Ausgabe der Schaltung **8** ein Symbol, das kein gültiger zahlentheoretischer Logarithmus ist.

[0039] Fig. 5 zeigt ein Blockschaltbild einer Ausführungsform einer Schaltung **8** zum Produzieren des zahlentheoretischen Logarithmus eines Residuums oder eines speziellen Nullsymbols, wenn der Eingangsoperand null ist. Für eine gegebene Basis α_i und einen gegebenen Modul p_i ist der zahlentheoretische Logarithmus des Werts in der Menge {1, 2, 3, ..., p_i – 1} in der Menge {1, 2, 3, ..., p_i – 2}. Bei der bevorzugten Ausführungsform der Schaltung **8** ist das spezielle Symbol, das resultiert, wenn die Eingabe null ist, das Binärwort, das aus nur Einsen besteht. Die Tabellennachschlagefunktion **8** kann unter Verwendung der in der Beschreibung von Fig. 3 besprochenen Prozedur auf eine Schaltung reduziert werden.

[0040] Fig. 6 zeigt ein Blockschaltbild einer Ausführungsform einer Schaltung zur Berechnung des Produkts zweier Residuen modulo p_i unter Verwendung der Summe der zahlentheoretischen Logarithmen der Operanden. Die Schaltung von Fig. 6 kann zwei Operanden annehmen, die zahlentheoretischen Logarithmen der Residuen, die multipliziert werden sollen, oder das Symbol für null, das von einer Schaltung **8** produziert wird, wenn sie eine Eingabe von null erhält. Die Operanden können einer modularen Addiererschaltung **9** zugeführt werden, die die Summe der Operanden modulo p_i – 1 produziert, deren Ausgabe nur dann gültig ist, wenn keiner der Operanden das Nullsymbol ist. Die Operanden können auch einer Schaltung **10** zum Detektieren des Symbols für null zugeführt werden. Die durch die Schaltung **9** produzierte Summe der Logarithmen kann dann in eine zahlentheoretische Exponentiierungs-Tabellennachschlageschaltung **11** eingegeben werden. Die Tabellennachschlagefunktion **11** kann unter Verwendung der in der Beschreibung von Fig. 3 besprochenen Prozedur auf eine Schaltung reduziert werden. Die Ausgabe der Nulldetektionsschaltungen **10** kann zum Beispiel durch ein OR-Gatter **12** dann logisch OR-verknüpft werden. Wenn die Ausgabe des OR-Gatters **12** anzeigt, dass einer der Eingangsoperanden das Nullsymbol war, kann die Ausgabe eines Multiplexers **13** auf null gesetzt und andernfalls die Ausgabe der Exponentiierungsschaltung **11** zu dem Ausgang des Multiplexers geleitet werden. Bei den meisten Implementierungen der in Fig. 1 und Fig. 2 gezeigten Systeme ist die zahlentheoretische Exponentiierungstabellen-Nachschlageschaltung **11** der häufigste Tabellennachschlag in dem System. Im allgemeinen gibt es für ein spezifisches $(Z/p_iZ)\backslash 0$ viele mögliche Generatoren. Für einen beliebigen Modul p_i gibt es sogar zwanzig Prozent Variation der Größe der Exponentiierungsschaltung über die gesamte Menge möglicher Generatoren. Folglich können Generatoren auf der Basis eines oder mehrerer Faktoren ausgewählt werden. Bei einer bevorzugten Ausführungsform der vorliegenden Erfindung kann für je-

den Modul p_i ein optimaler Generator α_i auf der Basis eines oder mehrerer Kriterien, wie zum Beispiel Größe, Geschwindigkeit, Strom oder einer bestimmten anderen Kostenfunktion ausgewählt werden. Diese optimale Erzeugung kann dann zur Erzeugung der zahlentheoretischen Exponentiierungsschaltung **11** und/oder der zahlentheoretischen Logarithmusschaltung **8** verwendet werden.

[0041] Bei den in Fig. 6 gezeigten Ausführungsformen werden die Logarithmen der Operanden durch eine Nulldetektionsschaltung **10** geprüft; wenn eine der Logarithmuseingaben das spezielle Symbol für null ist, was durch das logische OR-Gatter **12** bestimmt wird, wird das ausgegebene Produkt durch einen Multiplexer **13** auf null gesetzt. Andernfalls können die Logarithmen modulo $p_i - 1$ durch eine modulare Addiererschaltung **9** addiert werden, deren Ausgabe in eine Exponentiierungsschaltung **11** eingegeben werden kann. Die Ausgabe der Exponentiierungsschaltung **11** kann dann zu dem Multiplexer **13** geleitet werden, und wenn keiner der Operanden das spezielle Nullsymbol war, was durch die Ausgabe des OR-Gatters **12** bestimmt wird, kann die Ausgabe des Multiplexers **13** auf die Ausgabe der Exponentiierungsschaltung **11** gesetzt werden.

[0042] Fig. 7 zeig ein Blockschaltbild einer Ausführungsform zur Reduktion eines N-Bit-Binäroperanden auf sein Residuum modulo p_i. Dieser Binäroperand kann zum Beispiel vorzeichenlos oder Zweierkomplement sein. Eine Nullerweiterung **14** kann die niedrigstwertigen $N_i - 1$ Bit des Eingangsoperanden nehmen und sein N_i-Bit-Residuum modulo p_i produzieren. Der herkömmliche N-Bit-Operand kann in $q_i + 1$ Gruppen von Bit aufgeteilt werden. Die $N_i - 1$ niedrigstwertigen Bit sind bereits modulo p_i reduziert, werden aber durch eine Nullerweiterung **14** auf N_i Bit nullerweitert. Die übrigen $N - N_i + 1$ Bit des Eingangsoperanden können in q_i Gruppen von Bit aufgeteilt werden, die in q_i Tabellennachschläge **15A**, **15B** und **15C** eingegeben werden. Jede Aufteilung von Bit Q_{ij} für $j \in \{0, 1, 2, ..., q_i - 1\}$ kann in eine Tabellennachschlageschaltung **15A**, **15B** und **15C** eingegeben werden. Die Tabellennachschlagungen **15A**, **15B** und **15C** können dann die Residuen der gewichteten Eingaben produzieren. Die von den Tabellennachschlagungen **15A**, **15B** und **15C** durchgeführten mathematischen Funktionen können unter Verwendung der in der Beschreibung von Fig. 3 besprochenen Prozedur auf Schaltungen reduziert werden. Die $q_i + 1$ Residuen können durch einen modularen Addierer **16** mit $q_i + 1$ Operanden addiert werden, um das Residuum des ursprünglichen Eingangsoperanden modulo p_i zu produzieren. Zum Beispiel kann die Ausgabe des Aufteilers **14** und der Tabellennachschlageschaltungen **15A**, **15B**, **15C** durch eine modulare Rddiererschaltung **16** mit $q_i + 1$ Operanden addiert werden, deren Summe der ursprüngliche N-Bit-Operand, reduziert modulo p_i, ist.

[0043] Fig. 8 zeigt ein Blockschaltbild einer Ausführungsform einer Schaltung zur Berechnung der Summe von $L > 2$ Operanden (L Residuum) modulo p_i. Die L Operanden können durch einen Binäraddiererbaum **17** addiert werden, um die volle Summe der L Operanden zu produzieren. Zum Beispiel kann der Binäraddierer **17** die vorzeichenlose Summe mit $N_i + \lceil \log_2 L \rceil$ Bit produzieren. Die $N_i - 1$ niedrigstwertigen Bit können durch einen Aufteiler **20** aus der vollen Summe abgeteilt und durch eine Nullerweiterung **21** auf N_i Bit nullerweitert werden. Wie gezeigt, kann die Ausgabe des Binäraddierers **17** durch einen Busaufteiler **20** aufgeteilt werden und die höchstwertigen $\lceil \log_2 L \rceil + 1$ Bit zu einer modulo-p_i-Tabellennachschlageschaltung **18** geleitet werden, während die niedrigstwertigen $N_i - 1$ Bit zu einer Nullerweiterung **21** geleitet werden. Die Tabellennachschlagefunktion **18** kann unter Verwendung der mit Bezug auf die Ausführungsform von Fig. 3 besprochenen Prozedur auf eine Schaltung reduziert werden. Die Ausgaben der modulo-p_i-Tabellennachschlageschaltung **18** und der Nullerweiterung **21** werden durch einen modulo-p_i-Addierer **19** kombiniert, der die Summe der L Operanden modulo p_i produziert.

[0044] Eine Ausführungsform der vorliegenden Erfindung kann zur Umsetzung eines L-Operanden-RNS-Werts in einen herkömmlichen Wert unter Verwendung des chinesischen Residuumtheorems verwendet werden. Fig. 9 zeigt ein Blockschaltbild einer Ausführungsform einer Schaltung zum Umsetzen der L-Residuumdarstellung eines Werts in seine vorzeichenlose Binärdarstellung durch das chinesische Residuumtheorem. Die L Residuen $\{x_0, x_1, x_2, ..., x_{L-1}\}$ können in L separate CRT-Funktions-Tabellennachschlageschaltungen **22A**, **22B**, **22C** und **22D** eingegeben werden, die L Ergebnisse produzieren. Die Tabellennachschlagefunktionen **22A**, **22B**, **22C** und **22D** können unter Verwendung der in der Beschreibung von Fig. 3 besprochenen Prozedur auf Schaltungen reduziert werden. Diese Ergebnisse (modulare Addiererschaltung **23**) zum Beispiel zum Produzieren der vorzeichenlosen Binärdarstellung des Eingangswerts.

[0045] Eine Ausführungsform der vorliegenden Erfindung kann zur Umsetzung eines L-Operanden-RNS-Werts in einen herkömmlichen Wert unter Verwendung von L-CRT verwendet werden. Fig. 10 zeigt ein Blockschaltbild einer Ausführungsform einer Schaltung zum Umsetzen der L-Residiuumdarstellung eines Werts in eine skalierte vorzeichenlose Binär- oder Zweierkomplementdarstellung unter Verwendung der L-CRT-Umsetzung. Die L Residuen $\{x_0, x_1, x_2, ..., x_{L-1}\}$ können in L separate L-CRT-Funktions-Tabellennachschlageschaltungen **24A**, **24B**, **24C** und **24D** eingegeben werden, die L skalierte Ergebnisse produzieren. Die Tabellennachschlagefunktionen **24A**, **24B**, **24C** und **24D** können unter Verwendung der in der Beschreibung von Fig. 3 besproche-

nen Prozedur auf Schaltungen reduziert werden. Diese von den Tabellennachschlageschaltungen **24A**, **24B**, **24C** und **24D** produzierten Ergebnisse können dann durch eine Binäraddiererschaltung **25** addiert werden, um zum Beispiel die skalierte vorzeichenlose Binär- oder Zweierkomplementdarstellung des Eingangswerts zu produzieren.

[0046] Eine Ausführungsform der vorliegenden Erfindung kann zur Umsetzung von CRNS-Operanden in QRNS-Form verwendet werden. Fig. 11 zeigt ein Blockschaltbild einer Ausführungsform einer Schaltung zum Umsetzen eines Werts des komplexen Residuumsystems (CRNS) in einen Wert des quadratischen Residuumsystems (QRNS). Die imaginäre Komponente der CRNS-Eingabe b_i kann in eine Schaltung **26** zur Konstanten-Multiplikation mit ĵ eingegeben werden. Zum Beispiel kann der imaginäre Residuumoperand b_i in eine Schaltung **26** eingegeben werden, die das Produkt des Operanden mit ĵ nachschlägt. Die Tabellennachschlagefunktion **26** kann unter Verwendung der mit Bezug auf die Ausführungsform von Fig. 3 besprochenen Prozedur auf eine Schaltung reduziert werden. Die Ausgabe der Tabellennachschlagefunktion **26** und der Realteil der CRNS-Eingabe a_i können durch eine modulare Addiererschaltung **27** modulo p_i addiert werden, um die QRNS-Komponente z_i zu produzieren. Die Ausgabe der Tabellennachschlageschaltung **26** kann dann durch eine modulare Subtrahiererschaltung **28** modulo p_i von dem Realteil der CRNS-Eingabe subtrahiert werden, um die QRNS-Komponente z_i* zu produzieren.

[0047] Eine Ausführungsform der vorliegenden Erfindung kann für die Umsetzung von QRNS-Operanden in CRNS-Form verwendet werden. Fig. 12 zeigt ein Blockschaltbild einer Ausführungsform einer Schaltung zum Umsetzen eines Werts des quadratischen Residuumsystems in einen Wert des komplexen Residuumsystems. Die QRNS-Komponenten z_i und z_i* können durch eine modulare Addiererschaltung **29** modulo p_i addiert werden. Die QRNS-Komponente z_i* kann durch eine modulare Subtrahiererschaltung **30** modulo p_i von der Komponente z subtrahiert werden. Die Ausgabe der modularen Addiererschaltung **29** kann in eine Tabellennachschlageschaltung **31** der Konstanten-Multiplikation mit 2^{-1} eingegeben werden, deren Ausgabe die reelle Komponente der CRNS-Darstellung der Daten ist. Die Ausgabe des modularen Addierers **29** kann in eine Schaltung **31** eingegeben werden, die das Produkt der Summe mit 2^{-1} nachschlägt. Die Ausgabe der modularen Subtrahiererschaltung **30** kann in eine Tabellennachschlageschaltung **32** der Konstanten Multiplikation mit $ĵ^{-1}2^{-1}$ eingegeben werden, deren Ausgabe die imaginäre Komponente der CRNS-Darstellung der Daten ist. Die Ausgabe des modularen Subtrahierers **30** kann in eine Schaltung **32** eingegeben werden, die das Produkt der Summe mit $ĵ^{-1}2^{-1}$ nachschlägt. Die Produkttabellennachschlagefunktionen **31** und **32** können unter Verwendung der mit Bezug auf die Ausführungsform von Fig. 3 besprochenen Prozedur auf Schaltungen reduziert werden.

[0048] Die Verwendung von Logikgattern zur Implementierung verschiedener Tabellennachschlageoperationen gemäß der vorliegenden Erfindung kann gegenüber dem vorherigen Verfahren der Verwendung von Speicherbausteinen vielfältige Vorteile ergeben. Die Verwendung von Logikgattern kann eine effiziente Implementierung von RNS-Rechenschaltkreisen in vielfältigen Technologien erlauben, von denen einigen die Verwendung von RNS-Techniken zuvor nicht zugänglich war. Zusätzlich kann die Verwendung von Logikgattern anstelle von Speichern für RNS-Rechenschaltkreise einen oder mehrere der folgenden Vorteile ergeben: Logikgatter, die in die statischer CMOS-Logik (Complimentary Metal Oxide Semiconductor) implementiert werden, können bei Fehlen von Schaltaktivität in der Schaltung sehr viel weniger Strom verbrauchen; Logikgatter können direkt auf Tief-Submikrometer-Halbleiterherstellungstechnologien skaliert werden; Logikgatter können mit Standard-Logikprüfmethodologien kompatibel sein; Gruppen von Logikgattern können auf Geschwindigkeit, Strom und Fläche optimiert werden; Gruppen von Logikgattern können leicht durch manuelle oder automatische Mittel im Pipeline-Verfahren behandelt werden; und Logikgatter können Störungen der Führung von Leitungen auf einem Halbleiterbaustein im Vergleich zu Speichern reduzieren.

[0049] Im Gegensatz zu Speichern, die für eine beliebige gegebene Tabellennachschlagefunktion einer gegebenen Eingangs- und Ausgangsgröße feste Fläche und Geschwindigkeit aufweisen, können Gruppen von Logikgattern für die zu implementierenden spezifischen Tabellennachschlagefunktionen minimiert werden. In vielen Fällen kann die zu minimierende Logikfunktion eine gewisse zugrundeliegende Struktur aufweisen, die aus einer Betrachtung der Tabelle nicht offensichtlich ist. Diese Struktur kann für Gruppen von Logikgattern gegenüber Speichern zu signifikanten Flächen- und Geschwindigkeitsvorteilen führen. Ein Tabellennachschlagen für das Produkt einer Acht-Bit-Eingabe modulo 241 und 2^{-1} modulo 241, die in einem Nurlesespeicher (ROM) in einem 0,2-Mikrometer-Standardzellenprozeß für anwendungsspezifische integrierte Schaltungen (ASIC) produziert wird, erfordert zum Beispiel die äquivalente Fläche von 2250 Gattern und weist bei 100 MHz eine Verlustleistung von 3,6 mW auf, während dieselbe Tabelle, die als Gatter produziert wird, nur die Fläche von 36 Gattern erfordert und bei derselben Geschwindigkeit eine Verlustleistung von 0,23 mW aufweist. Eine weitere Tabelle derselben Größe (eine Exponentiationstabelle modulo 241) erfordert nur eine Fläche von 675 Gattern und weist bei derselben Geschwindigkeit eine Verlustleistung von 1,3 mW

auf.

[0050] Diese Ergebnisse wurden unter Verwendung des zuvor mit Bezug auf die Ausführungsform von Fig. 3 beschriebenen Prozesses erhalten. Der oben erwähnte ROM hat eine minimale Taktperiode von 3,0 ns, während das oben erwähnte, als Gatter implementierte Produktnachschlagen eine maximale Verzögerung vom Eingang zum Ausgang von 1,0 ns aufweist und die Exponentiationsnachschlagung, die als Gatter implementiert wird, eine maximale Verzögerung von 3,0 ns aufweist. Im Fall des Exponentiationsnachschlagens kann eine Verzögerung von 1,2 ns erzielt werden, obwohl die Fläche der Funktion auf 957 Gatter vergrößert wird. Dieses Beispiel ist eine beeindruckende Demonstration der Fähigkeit der vorliegenden Erfindung, die Optimierung des Gleichgewichts zwischen Geschwindigkeit, Fläche und Strom zu erlauben, indem man RNS-Tabellennachschlagevorgänge unter Verwendung von Logikgattern implementiert, anstelle von Speichern wie etwa ROMs. Für eine gegebene Implementierungstechnologie besitzen ROMs die höchste Speicherdichte aller Arten von Speicher. Zum Beispiel erfordert ein statischer RAM, der in derselben Technologie wie der oben erwähnte ROM und mit denselben Größen- und Geschwindigkeitskenngrößen implementiert wird, die äquivalente Fläche von 3660 Gattern. Dieses Beispiel zeigt außerdem, dass durch Verwendung von Logikgattern zur Implementierung von Tabellennachschlagefunktionen ein Kompromiss zwischen Fläche und Geschwindigkeit erzielt werden kann, um den Bedürfnissen eines bestimmten Entwurfs am besten zu genügen.

[0051] Es versteht sich, dass die hier beschriebenen Beispiele und Ausführungsformen lediglich zur Veranschaulichung dienen und dass Fachleuten verschiedene Modifikationen oder Änderungen im Hinblick dieser einfallen werden, die in den Schutzumfang der angefügten Ansprüche aufgenommen werden sollen.

Patentansprüche

1. Verfahren zum Durchführen mathematischer Berechnungen unter Verwendung der Residuumarithmetik, mit einem oder mehreren der folgenden Schritte:
Umsetzen von Daten in Binärcode in Residuen, wobei das Umsetzen von Daten in Binärcode in Residuen folgendes umfasst:
Empfangen von Eingangsdaten in Binärcode; und
Umsetzen der Eingangsdaten in Binärcode in Residuen, wobei die Eingangsdaten in Binärcode binäre Operanden mit einer Vielzahl von Bit umfassen, wobei das Umsetzen der Eingangsdaten in Binärcode in Residuen das Berechnen von Werten des Residuums modulo p_i mindestens einer Untergruppe der Vielzahl von Bit der binären Operanden umfasst,
wobei das Berechnen von Werten des Residuums modulo p_i mindestens einer Untergruppe der Vielzahl von Bit der binären Operanden unter Verwendung einer Vielzahl von Logikgattern implementiert wird, wobei die Vielzahl von Logikgattern durch die folgenden Schritte ausgewählt werden:
Berechnen von Werten des Residuums modulo p_i für die mindestens eine Untergruppe der Vielzahl von Bit der binären Operanden;
Extrahieren von logischen Gleichungen, die die berechneten Werte des Residuums modulo p_i der mindestens einen Untergruppe der Vielzahl von Bit der binären Operanden repräsentieren; und
Abbilden der logischen Gleichungen auf die Vielzahl von Logikgattern;
Umsetzen von Residuen aus dem komplexen Restklassensystem in das quadratische Restklassensystem, wobei das Umsetzen von Residuen aus dem komplexen Restklassensystem in das quadratische Restklassensystem folgendes umfasst:
Empfangen von Eingangsdaten im Residuumformat des komplexen Restklassensystems; und Umsetzen der Residuen aus dem komplexen Restklassensystem in das quadratische Restklassensystem, wobei jedes Residuum des komplexen Restklassensystems einen imaginären Residuumoperanden b_i umfasst, wobei das Umsetzen der Residuen aus dem komplexen Restklassensystem in das quadratische Restklassensystem das Multiplizieren des imaginären Residuumoperanden des eingegebenen komplexen Restklassensystems, b_i, mit ĵ umfasst, wobei das Multiplizieren des imaginären Residuumoperanden des eingegebenen komplexen Restklassensystems, b_i, mit ĵ unter Verwendung einer zweiten Vielzahl von Logikgattern implementiert wird, wobei die zweite Vielzahl von Logikgattern durch die folgenden Schritte ausgewählt wird:
Berechnen eines Werts des Produkts von b_i und ĵ modulo p_i für jeden möglichen modularen Datenoperanden;
Extrahieren von logischen Gleichungen, die berechnete Werte der Multiplizierer mit der ĵ-Funktion repräsentieren; und
Abbilden der logischen Gleichungen auf die zweite Vielzahl von Logikgattern;
Berechnen modularer Produkte von Residuen, wobei das Berechnen modularer Produkte von Residuen folgendes umfasst:
Empfangen von Eingangsdaten im Residuumformat; und Berechnen modularer Produkte der Residuen, wobei das Berechnen modularer Produkte der Residuen folgendes umfasst: Berechnen eines zahlentheoretischen Logarithmus modulo p_i für jede der Residuen, wovon das modulare Produkt berechnet wird, und Berechnen eines zahlentheoretischen Exponenten modulo p_i der Summe der zahlentheoretischen Logarithmen der Residuen,
wobei das Berechnen eines zahlentheoretischen Logarithmus modulo p_i für jede der Residuen, wovon das modulare Produkts berechnet wird, und das Be-

rechnen eines zahlentheoretischen modulo p_i der Summe der zahlentheoretischen Logarithmen der Residuen unter Verwendung einer dritten Vielzahl von Logikgattern implementiert wird,
wobei die dritte Vielzahl von Logikgattern durch die folgenden Schritte ausgewählt wird:
Berechnen des zahlentheoretischen Logarithmus modulo b_i für jedes mögliche Residuum;
Berechnen des zahlentheoretischen Exponenten modulo p_i für jede mögliche Summe;
Extrahieren logischer Gleichungen, die berechnete Werte des zahlentheoretischen Logarithmus und der Exponentenfunktionen repräsentieren; und
Abbilden der logischen Gleichung auf die dritte Vielzahl von Logikgattern;
Berechnen modularer Summen von Residuen, wobei das Berechnen modularer Summen von Residuen folgendes umfasst:
Empfangen von Eingangsdaten im Residuumformat; und
Berechnen modularer Summen der Residuen, wobei das Berechnen modularer Summen der Residuen das Produzieren einer vollen Summe der Produkte von Residuen umfasst, wobei die volle Summe der Residuen eine Vielzahl von Bit aufweist, die der vollen Summe der Residuen entspricht, und Berechnen von Werten mindestens einer Untergruppe der Vielzahl von Bit, die der vollen Summe der Residuen entspricht, modulo p_i,
wobei das Berechnen von Werten mindestens einer Untergruppe der Vielzahl von Bit, die der vollen Summe der Residuen entspricht, modulo p_i unter Verwendung einer vierten Vielzahl von Logikgattern implementiert wird, wobei die vierte Vielzahl von Logikgattern durch die folgenden Schritte ausgewählt wird:
Berechnen der Werte des Residuums modulo p_i für die mindestens eine Untergruppe der Vielzahl von Bit, die der vollen Summe der Residuen entspricht;
Extrahieren logischer Gleichungen, die die berechneten Residuumwerte modulo p_i der mindestens einen Untergruppe der Vielzahl von Bit, die der vollen Summe der Residuen entspricht, repräsentieren; und
Abbilden der logischen Gleichungen auf die vierte Vielzahl von Logikgattern;
Umsetzen von Residuen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem, wobei das Umsetzen von Residuen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem folgendes umfasst:
Empfangen von Eingangsdaten im Residuumformat des quadratischen Restklassensystems; und
Umsetzen der Residuen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem, wobei das Umsetzen der Residuen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem das Berechnen des Produkts von 2^{-1} und der Summe des Residuums z_i des quadratischen Restklassensystems und z_i* umfasst, wobei das Produkt von 2^{-1} und der Summe des Residuums z_i des quadratischen Restklassensystems und z_i* die reelle Komponente des Residuums des komplexen Restklassensystem ist, und Berechnen des Produkts von $\hat{\jmath}^{-1}2^{-1}$ und der Differenz modulo p_i von z_i minus z_i*, wobei das Produkt von $\hat{\jmath}^{-1}2^{-1}$ und der Differenz modulo p_i von z_i minus z_i* die imaginäre Komponente des Residuums des komplexen Restklassensystems ist,
wobei das Berechnen des Produkts von 2^{-1} und der Summe des Residuums des quadratischen Restklassensystems, z_i und z_i*, und das Berechnen des Produkts von $\hat{\jmath}^{-1}2^{-1}$ und der Differenz modulo p_i von z_i minus z_i* unter Verwendung einer fünften Vielzahl von Logikgattern implementiert wird, wobei die fünfte Vielzahl von Logikgattern durch die folgenden Schritte ausgewählt wird:
Berechnen eines Werts des Produkts von z_i minus z_i* und $\hat{\jmath}^{-1}2^{-1}$ modulo p_i für jede mögliche Differenz;
Berechnen eines Werts des Produkts der Summe von z_i und z_i* und 2^{-1} für jede mögliche Summe;
Extrahieren von logischen Gleichungen, die berechnete Werte der Multiplizieren-mit-$\hat{\jmath}^{-1}2^{-1}$ und Multiplizieren-mit-2^{-1}-Funktionen repräsentieren; und
Abbilden der logischen Gleichungen auf die fünfte Vielzahl von Logikgattern; und
Umsetzen von Residuen in Daten in Binärcode, wobei das Umsetzen von Residuen in Daten in Binärcode folgendes umfasst:
Empfangen von Eingangsdaten im Residuumformat; und
Umsetzen der Residuen in Daten in Binärcode, wobei jedes Residuum ein L-Residuum-Restklassensystemwert ist, wobei das Umsetzen der Residuen in Daten in Binärcode das Anwenden der Funktion des chinesischen Restklassentheorems oder der Funktion des chinesischen L-Restklassentheorems auf jede der L Residuen jedes Residuums, um L Ergebnisse zu produzieren, und das modulare Addieren der L Ergebnisse umfasst,
wobei das Anwenden der Funktion des chinesischen Restklassentheorems oder der Funktion des chinesischen L-Restklassentheorems auf jede der L Residuen jedes Residuums unter Verwendung einer sechsten Vielzahl von Logikgattern implementiert wird, wobei die sechste Vielzahl von Gattern durch die folgenden Schritte ausgewählt wird:
Berechnen eines Werts einer Funktion des chinesischen Restklassentheorems oder des chinesischen L-Restklassentheorems für jede mögliche Residuumeingabe;
Extrahieren von logischen Gleichungen, die die berechneten Werte des chinesischen Restklassentheorems oder des chinesischen L-Restklassentheorems repräsentieren; und
Abbilden der logischen Gleichungen auf die sechste Vielzahl von Logikgatter.

2. Verfahren nach Anspruch 1, wobei das Verfahren die folgenden Schritte umfasst:
Empfangen von Eingangsdaten in Binärcode; und
Umsetzen der Eingangsdaten in Binärcode in Resi-

duen, wobei die Eingangsdaten in Binärcode binäre Operanden mit einer Vielzahl von Bit umfassen; wobei das Umsetzen der Eingangsdaten in Binärcode in Residuen das Berechnen von Werten des Residuums modulo p_i mindestens einer Untergruppe der Vielzahl von Bit der binären Operanden umfasst,
wobei das Berechnen von Werten des Residuums modulo p_i mindestens einer Untergruppe der Vielzahl von Bit der binären Operanden unter Verwendung einer Vielzahl von Logikgattern implementiert wird, wobei die Vielzahl von Logikgattern durch die folgenden Schritte ausgewählt wird:
Berechnen der Werte des Residuums modulo p_i für die mindestens eine Untergruppe der Vielzahl von Bit der binären Operanden;
Extrahieren von logischen Gleichungen, die die berechneten Werte des Residuums modulo p_i der mindestens einen Untergruppe der Vielzahl von Bit der binären Operanden repräsentieren; und
Abbilden der logischen Gleichungen auf die Vielzahl von Logikgattern.

3. Verfahren nach Anspruch 2, wobei das Verfahren die folgenden Schritte umfasst:
Umsetzen der Residuen aus dem komplexen Restklassensystem in das quadratische Restklassensystem, wobei jedes Residuum des komplexen Restklassensystems einen imaginären Residuumoperanden b_i umfasst, wobei das Umsetzen der Residuen aus dem komplexen Restklassensystem in das quadratische Restklassensystem das Multiplizieren des imaginären Residuumoperanden des eingegebenen komplexen Restklassensystems, b_i, mit ĵ umfasst,
wobei das Multiplizieren des imaginären Residuumoperanden des eingegebenen komplexen Restklassensystems, b_i, mit ĵ unter Verwendung der zweiten Vielzahl von Logikgattern implementiert wird.

4. Verfahren nach Anspruch 3, wobei das Verfahren die folgenden Schritte umfasst:
Berechnen modularer Produkte der Residuen des quadratischen Restklassensystems, wobei das Berechnen modularer Produkte von Residuen des quadratischen Restklassensystems das Berechnen eines zahlentheoretischen Logarithmus modulo p_i für jede der Residuen des quadratischen Restklassensystems, wovon das modulare Produkt berechnet wird, und das Berechnen eines zahlentheoretischen Exponenten modulo p_i der Summe der zahlentheoretischen Logarithmen der Residuen des quadratischen Restklassensystems umfasst,
wobei das Berechnen eines zahlentheoretischen Logarithmus modulo p_i für jede der Residuen des quadratischen Restklassensystems, wovon das modulare Produkt berechnet wird, und das Berechnen eines zahlentheoretischen Exponenten modulo p_i der Summe der zahlentheoretischen Logarithmen der Residuen des quadratischen Restklassensystems unter Verwendung der dritte Vielzahl von Logikgattern implementiert wird.

5. Verfahren nach Anspruch 4, wobei das Verfahren die folgenden Schritte umfasst:
Berechnen modularer Summen der Produkte von Residuen, wobei das Berechnen modularer Summen der Produkte von Residuen das Produzieren einer vollen Summe der Produkte der Residuen umfasst, wobei die volle Summe der Produkte eine Vielzahl von Bit aufweist, die der vollen Summe der Produkte entspricht, und Berechnen von Werten mindestens einer Untergruppe der Vielzahl von Bit, die der vollen Summe der Produkte entspricht, modulo p_i,
wobei das Berechnen von Werten mindestens einer Untergruppe der Vielzahl von Bit, die der vollen Summe der Produkte entspricht, modulo p_i, unter Verwendung der vierten Vielzahl von Logikgattern implementiert wird.

6. Verfahren nach Anspruch 5, wobei das Verfahren die folgenden Schritte umfasst:
Umsetzen der modularen Summen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem,
wobei das Umsetzen der modularen Summen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem das Berechnen des Produkts von 2^{-1} und der Summe des Residuums z_i des quadratischen Restklassensystems und z_i* umfasst, wobei das Produkt von 2^{-1} und der Summe des Residuums z_i des quadratischen Restklassensystems und z_i* die reelle Komponente des Residuums des komplexen Restklassensystems ist, und Berechnen des Produkts von $ĵ^{-1}2^{-1}$ und der Differenz von z_i minus z_i* modulo p_i, wobei das Produkt von $ĵ^{-1}2^{-1}$ und der Differenz von z_i minus z_i*, modulo p_i, die imaginäre Komponente des Residuums des komplexen Restklassensystems ist,
wobei das Berechnen des Produkts von 2^{-1} und der Summe des Residuums z_i des quadratischen Restklassensystems und z_i* und das Berechnen des Produkts von $ĵ^{-1}2^{-1}$ und der Differenz von z_i minus z_i*, modulo p_i, unter Verwendung der fünften Vielzahl von Logikgattern implementiert wird.

7. Verfahren nach Anspruch 6, wobei das Verfahren die folgenden Schritte umfasst:
Umsetzen der Residuen des komplexen Restklassensystems in Daten in Binärcode,
wobei jedes Residuum des komplexen Restklassensystems ein L-Residuum-Restklassensystemwert ist, wobei das Umsetzen der Residuen des komplexen Restklassensystems in Daten in Binärcode das Anwenden der Funktion des chinesischen Restklassentheorems oder der Funktion des chinesischen L-Restklassensystem auf jede der L Residuen jedes Residuums des komplexen Restklassensystems, um L Ergebnisse zu produzieren, und das modulare Addieren der L Ergebnisse umfasst,
wobei das Anwenden der Funktion des chinesischen Restklassentheorems oder der Funktion des chinesischen L-Restklassentheorems auf jede der L Residu-

en jedes Residuums des komplexen Restklassensystems unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

8. Verfahren nach Anspruch 5, wobei das Verfahren die folgenden Schritte umfasst:
Umsetzen der Residuen des quadratischen Restklassensystems in Daten in Binärcode,
wobei jedes Residuum des komplexen Restklassensystems ein L-Residuum-Restklassensystemwert ist,
wobei das Umsetzen der Residuen des komplexen Restklassensystems in Daten in Binärcode das Anwenden der Funktion des chinesischen Restklassentheorems oder der Funktion des chinesischen L-Restklassentheorems auf jede der L Residuen jedes Residuums des komplexen Restklassensystems, um L Ergebnisse zu produzieren, und das modulare Addieren der L Ergebnisse umfasst,
wobei das Anwenden der Funktion des chinesischen Restklassentheorems oder der Funktion des chinesischen L-Restklassentheorems auf jede der L Residuen jedes Residuums des komplexen Restklassensystems unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

9. Verfahren nach Anspruch 4, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der modularen Produkte aus dem quadratischen Restklassensystem in das komplexe Restklassensystem,
wobei der Schritt des Umsetzens der modularen Produkte aus dem quadratischen Restklassensystem in das komplexe Restklassensystem unter Verwendung der fünften Vielzahl von Logikgattern implementiert wird.

10. Verfahren nach Anspruch 9, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen des komplexen Restklassensystems in Daten in Binärcode,
wobei der Schritt des Umsetzens der Residuen des komplexen Restklassensystems in Daten in Binärcode unter Verwendung einer sechsten Vielzahl von Logikgattern implementiert wird.

11. Verfahren nach Anspruch 4, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der modularen Produkte in Daten in Binärcode,
wobei der Schritt des Umsetzens der modularen Produkte in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

12. Verfahren nach Anspruch 3, wobei das Verfahren den folgenden Schritt umfasst:
Berechnen modularer Summen der Residuen des quadratischen Restklassensystems, wobei der Schritt des Berechnens modularer Summen der Residuen des quadratischen Restklassensystems unter Verwendung der vierten Vielzahl von Logikgattern implementiert wird.

13. Verfahren nach Anspruch 12, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der modularen Summen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem,
wobei der Schritt des Umsetzens der modularen Summen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem unter Verwendung der fünften Vielzahl von Logikgattern implementiert wird.

14. Verfahren nach Anspruch 13, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen des komplexen Restklassensystems in Daten in Binärcode,
wobei der Schritt des Umsetzens der Residuen des komplexen Restklassensystems in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

15. Verfahren nach Anspruch 12, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der modularen Summen in Daten in Binärcode,
wobei der Schritt des Umsetzens der modularen Summen in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

16. Verfahren nach Anspruch 3, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen des quadratischen Restklassensystems aus dem quadratischen Restklassensystem in das komplexe Restklassensystem,
wobei der Schritt des Umsetzens der Residuen des quadratischen Restklassensystems aus dem quadratischen Restklassensystem in das komplexe Restklassensystem unter Verwendung der fünften Vielzahl von Logikgattern implementiert wird.

17. Verfahren nach Anspruch 16, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen des komplexen Restklassensystems in Daten in Binärcode,
wobei der Schritt des Umsetzens der Residuen des komplexen Restklassensystems in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

18. Verfahren nach Anspruch 3, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen des quadratischen Restklassensystems in Daten in Binärcode,
wobei der Schritt des Umsetzens der Residuen des quadratischen Restklassensystems in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

19. Verfahren nach Anspruch 2, wobei das Verfahren den folgenden Schritt umfasst:
Berechnen modularer Produkte der Residuen,
wobei der Schritt des Berechnens modularer Produkte der Residuen unter Verwendung der dritten Vielzahl von Logikgattern implementiert wird.

20. Verfahren nach Anspruch 19, wobei das Verfahren den folgenden Schritt umfasst:
Berechnen modularer Summen der modularen Produkte von Residuen,
wobei der Schritt des Berechnens modularer Summen der modularen Produkte von Residuen unter Verwendung der vierten Vielzahl von Logikgattern implementiert wird.

21. Verfahren nach Anspruch 20, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der modularen Summen aus dem quadratischen Restklassensystems in das komplexe Restklassensystem,
wobei der Schritt des Umsetzens der modularen Summen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem unter Verwendung der fünften Vielzahl von Logikgattern implementiert wird.

22. Verfahren nach Anspruch 21, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen des komplexen Restklassensystems in Daten in Binärcode,
wobei der Schritt des Umsetzens der Residuen des komplexen Restklassensystems in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

23. Verfahren nach Anspruch 20, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der modularen Summen in Daten in Binärcode,
wobei der Schritt des Umsetzens der modularen Summen in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

24. Verfahren nach Anspruch 19, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der modularen Produkte aus dem quadratischen Restklassensystem in das komplexe Restklassensystem,
wobei der Schritt des Umsetzens der modularen Produkte aus dem quadratischen Restklassensystem in das komplexe Restklassensystem unter Verwendung der fünften Vielzahl von Logikgattern implementiert wird.

25. Verfahren nach Anspruch 24, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen des komplexen Restklassensystems in Daten in Binärcode,
wobei der Schritt des Umsetzens der Residuen des komplexen Restklassensystems in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

26. Verfahren nach Anspruch 19, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der modularen Produkte in Daten in Binärcode,
wobei der Schritt des Umsetzens der modularen Produkte in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

27. Verfahren nach Anspruch 2, wobei das Verfahren den folgenden Schritt umfasst:
Berechnen modularer Summen der Residuen,
wobei der Schritt des Berechnens modularer Summen der Residuen unter Verwendung der vierten Vielzahl von Logikgattern implementiert wird.

28. Verfahren nach Anspruch 27, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der modularen Summen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem,
wobei der Schritt des Umsetzens der modularen Summen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem unter Verwendung der fünften Vielzahl von Logikgattern implementiert wird.

29. Verfahren nach Anspruch 28, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen des komplexen Restklassensystems in Daten in Binärcode,
wobei der Schritt des Umsetzens der Residuen des komplexen Restklassensystems in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

30. Verfahren nach Anspruch 27, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der modularen Summen in Daten in Binärcode,
wobei der Schritt des Umsetzens der modularen Summen in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

31. Verfahren nach Anspruch 2, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem,
wobei der Schritt des Umsetzens der Residuen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem unter Verwendung der fünften Vielzahl von Logikgattern implementiert wird.

32. Verfahren nach Anspruch 31, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen des komplexen Restklassensystems in Daten in Binärcode,
wobei der Schritt des Umsetzens der Residuen des komplexen Restklassensystems in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

33. Verfahren nach Anspruch 2, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen in Daten in Binärcode,
wobei der Schritt des Umsetzens der Residuen in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

34. Verfahren nach Anspruch 1, wobei das Verfahren den folgenden Schritt umfasst:
Empfangen von Eingangsdaten im Residuumformat des komplexen Restklassensystems; und
Umsetzen der Residuen aus dem komplexen Restklassensystem in das quadratische Restklassensystem, wobei jedes Residuum des komplexen Restklassensystems einen imaginären Residuumoperanden b_i umfasst, wobei das Umsetzen der Residuen aus dem komplexen Restklassensystem in das quadratische Restklassensystem das Multiplizieren des imaginären Residuumoperanden des eingegebenen komplexen Restklassensystems, b_i, mit ĵ umfasst, wobei das Multiplizieren des imaginären Residuumoperanden des eingegebenen komplexen Restklassensystems, b_i, mit ĵ unter Verwendung der fünften Vielzahl von Logikgattern implementiert wird, wobei die fünfte Vielzahl von Logikgattern durch die folgenden Schritte ausgewählt wird:
Berechnen eines Werts des Produkts von b_i und ĵ, modulo p_i, für jeden möglichen modularen Datenoperanden;
Extrahieren von logischen Gleichungen, die berechnete Werte der Multiplizieren mit-ĵ-Funktion repräsentieren; und
Abbilden der logischen Gleichungen auf die fünfte Vielzahl von Logikgattern.

35. Verfahren nach Anspruch 4, wobei das Verfahren den folgenden Schritt umfasst:
Berechnen modularer Produkte der Residuen des quadratischen Restklassensystems,
wobei der Schritt des Berechnens modularer Produkte der Residuen des quadratischen Restklassensystems unter Verwendung der dritten Vielzahl von Logikgattern implementiert wird.

36. Verfahren nach Anspruch 35, wobei das Verfahren den folgenden Schritt umfasst:
Berechnen modularer Summen der Produkte von Residuen,
wobei der Schritt des Berechnens modularer Summen der Produkte von Residuen unter Verwendung der vierten Vielzahl von Logikgattern implementiert wird.

37. Verfahren nach Anspruch 36, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der modularen Summen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem,
wobei der Schritt des Umsetzens der modularen Summen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem unter Verwendung der fünften Vielzahl von Logikgattern implementiert wird.

38. Verfahren nach Anspruch 37, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen des komplexen Restklassensystems in Daten in Binärcode,
wobei der Schritt des Umsetzens der Residuen des komplexen Restklassensystems in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

39. Verfahren nach Anspruch 36, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen des quadratischen Restklassensystems in Daten in Binärcode,
wobei der Schritt des Umsetzens der Residuen des quadratischen Restklassensystems in Daten in Binärcode unter Verwendung der fünften Vielzahl von Logikgattern implementiert wird.

40. Verfahren nach Anspruch 35, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen modularer Produkte aus dem quadratischen Restklassensystem in das komplexe Restklassensystem,
wobei der Schritt des Umsetzens der Residuen modularer Produkte aus dem quadratischen Restklassensystem in das komplexe Restklassensystem unter Verwendung der fünften Vielzahl von Logikgattern implementiert wird.

41. Verfahren nach Anspruch 40, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen des komplexen Restklassensystems in Daten in Binärcode,
wobei der Schritt des Umsetzens der Residuen des komplexen Restklassensystems in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

42. Verfahren nach Anspruch 35, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der modularen Produkte in Daten in Binärcode,
wobei der Schritt des Umsetzens der modularen Produkte in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

43. Verfahren nach Anspruch 34, wobei das Verfahren den folgenden Schritt umfasst:
Berechnen modularer Summen der Residuen des quadratischen Restklassensystems,
wobei der Schritt des Berechnens modularer Summen der Residuen des quadratischen Restklassensystems unter Verwendung der vierten Vielzahl von Logikgattern implementiert wird.

44. Verfahren nach Anspruch 43, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der modularen Summen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem,
wobei der Schritt des Umsetzens der modularen Summen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem unter Verwendung der fünften Vielzahl von Logikgattern implementiert wird.

45. Verfahren nach Anspruch 44, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen des komplexen Restklassensystems in Daten in Binärcode,
wobei der Schritt des Umsetzens der Residuen des komplexen Restklassensystems in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

46. Verfahren nach Anspruch 43, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der modularen Summen in Daten in Binärcode,
wobei der Schritt des Umsetzens der modularen Summen in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

47. Verfahren nach Anspruch 34, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen des quadratischen Restklassensystems aus dem quadratischen Restklassensystem in das komplexe Restklassensystem,
wobei der Schritt des Umsetzens der Residuen des quadratischen Restklassensystems aus dem quadratischen Restklassensystem in das komplexe Restklassensystem unter Verwendung der fünften Vielzahl von Logikgattern implementiert wird.

48. Verfahren nach Anspruch 47, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen des komplexen Restklassensystems in Daten in Binärcode,
wobei der Schritt des Umsetzens der Residuen des komplexen Restklassensystems in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

49. Verfahren nach Anspruch 34, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen des komplexen Restklassensystems in Daten in Binärcode,
wobei der Schritt des Umsetzens der Residuen des komplexen Restklassensystems in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

50. Verfahren nach Anspruch 1, wobei das Verfahren die folgenden Schritte umfasst:
Empfangen von Eingangsdaten im Residuumformat; und
Berechnen modularer Produkte der Residuen,
wobei das Berechnen modularer Produkte der Residuen das Berechnen eines zahlentheoretischen Logarithmus modulo p_i für jede der Residuen, wovon das modulare Produkt berechnet wird, und das Berechnen eines zahlentheoretischen Exponenten modulo p_i der Summe der zahlentheoretischen Logarithmen der Residuen umfasst,
wobei das Berechnen eines zahlentheoretischen Logarithmus modulo p_i für jede der Residuen, wovon das modulare Produkt berechnet wird, und das Berechnen eines zahlentheoretischen Exponenten modulo p_i der Summe der zahlentheoretischen Logarithmen der Residuen unter Verwendung einer dritten Vielzahl von Logikgattern implementiert wird,
wobei die Vielzahl von Logikgattern durch die folgenden Schritte ausgewählt wird:
Berechnen des zahlentheoretischen Logarithmus modulo p_i für jedes mögliche Residuum;
Berechnen des zahlentheoretischen Exponenten modulo p_i für jede mögliche Summe;
Extrahieren logischer Gleichungen, die berechnete Werte der Funktionen des zahlentheoretischen Logarithmus und des Exponenten repräsentieren; und
Abbilden der logischen Gleichung auf die dritte Vielzahl von Logikgattern.

51. Verfahren nach Anspruch 50, wobei das Verfahren den folgenden Schritt umfasst:
Berechnen modularer Summen der Produkte von Residuen,
wobei der Schritt des Berechnens modularer Summen der Produkte von Residuen unter Verwendung der vierten Vielzahl von Logikgattern implementiert wird.

52. Verfahren nach Anspruch 51, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der modularen Summen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem,
wobei der Schritt des Umsetzens der modularen Summen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem unter Verwendung der vierten Vielzahl von Logikgattern implementiert wird.

53. Verfahren nach Anspruch 52, wobei das Verfahren den folgenden Schritt umfasst:

Umsetzen der Residuen des komplexen Restklassensystems in Daten in Binärcode,
wobei der Schritt des Umsetzens der Residuen des komplexen Restklassensystems in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

54. Verfahren nach Anspruch 51, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der modularen Summen in Daten in Binärcode,
wobei der Schritt des Umsetzens der modularen Summen in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

55. Verfahren nach Anspruch 50, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der modularen Produkte aus dem quadratischen Restklassensystem in das komplexe Restklassensystem,
wobei der Schritt des Umsetzens der modularen Produkte aus dem quadratischen Restklassensystem in das komplexe Restklassensystem unter Verwendung der fünften Vielzahl von Logikgattern implementiert wird.

56. Verfahren nach Anspruch 55, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen des komplexen Restklassensystems in Daten in Binärcode,
wobei der Schritt des Umsetzens der Residuen des komplexen Restklassensystems in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

57. Verfahren nach Anspruch 50, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der modularen Produkte in Daten in Binärcode,
wobei der Schritt des Umsetzens der modularen Produkte in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

58. Verfahren nach Anspruch 1, wobei das Verfahren die folgenden Schritte umfasst:
Empfangen von Eingangsdaten im Residuumformat; und
Berechnen modularer Summen der Residuen,
wobei das Berechnen modularer Summen der Residuen das Produzieren einer vollen Summe der Produkte der Residuen umfasst, wobei die volle Summe der Residuen eine Vielzahl von Bit aufweist, die der vollen Summe der Residuen entspricht, und Berechnen von Werten mindestens einer Untergruppe der Vielzahl von Bit, die der vollen Summe der Residuen entspricht, modulo p_i,
wobei das Berechnen von Werten mindestens einer Untergruppe der Vielzahl von Bit, die der vollen Summe der Residuen entspricht, modulo p_i, unter Verwendung einer Vielzahl von Logikgattern implementiert wird, wobei die Vielzahl von Logikgattern durch die folgenden Schritte ausgewählt wird:
Berechnen der Werte des Residuums modulo p_i für die mindestens eine Untergruppe der Vielzahl von Bit, die der vollen Summe der Residuen entspricht;
Extrahieren von logischen Gleichungen, die die berechneten Werte des Residuums modulo p_i der mindestens einen Untergruppe der Vielzahl von Bit, die der vollen Summe der Residuen entspricht, repräsentieren; und
Abbilden der logischen Gleichungen auf die vierte Vielzahl von Logikgattern.

59. Verfahren nach Anspruch 58, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem,
wobei der Schritt des Umsetzens der Residuen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem unter Verwendung der fünften Vielzahl von Logikgattern implementiert wird.

60. Verfahren nach Anspruch 59, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen des komplexen Restklassensystems in Daten in Binärcode,
wobei der Schritt des Umsetzens der Residuen des komplexen Restklassensystems in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

61. Verfahren nach Anspruch 58, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen in Daten in Binärcode,
wobei der Schritt des Umsetzens der Residuen in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

62. Verfahren nach Anspruch 1, wobei das Verfahren die folgenden Schritte umfasst:
Empfangen von Eingangsdaten im Residuumformat des quadratischen Restklassensystems; und
Umsetzen der Residuen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem, wobei das Umsetzen der Residuen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem das Berechnen des Produkts von 2^{-1} und der Summe des Residuums z_i des quadratischen Restklassensystems und z_i* umfasst, wobei das Produkt von 2^{-1} und der Summe des Residuums z_i des quadratischen Restklassensystems und z_i* die reelle Komponente des Residuums des komplexen Restklassensystems ist, sowie das Berechnen des Produkts von $\hat{\jmath}^{-1}2^{-1}$ und der Differenz von z_i minus z_i*, modulo p_i, wobei das Produkt von $\hat{\jmath}^{-1}2^{-1}$ und der Differenz von z_i minus z_i*, modulo p_i, die imaginäre Komponente des Residuums des komplexen

Restklassensystems ist,
wobei das Berechnen des Produkts von 2^{-1} und der Summe des Residuums z_i des quadratischen Restklassensystems und z_i* und das Berechnen des Produkts von $\hat{\jmath}^{-1}2^{-1}$ und der Differenz von z_i minus z_i*, modulo p_i, unter Verwendung der fünften Vielzahl von Logikgattern implementiert wird, wobei die fünfte Vielzahl von Logikgattern durch die folgenden Schritte ausgewählt wird:
Berechnen eines Werts des Produkts von z_i minus z_i* und $\hat{\jmath}^{-1}2^{-1}$, modulo p_i, für jede mögliche Differenz;
Berechnen eines Werts des Produkts der Summe von z_i und z_i* und 2^{-1} für jede mögliche Summe;
Extrahieren von logischen Gleichungen, die berechnete Werte der Multiplizieren-mit-$\hat{\jmath}^{-1}2^{-1}$- und Multiplizieren-mit-2^{-1}-Funktionen repräsentieren; und
Abbilden der logischen Gleichungen auf die fünfte Vielzahl von Logikgattern.

63. Verfahren nach Anspruch 62, wobei das Verfahren den folgenden Schritt umfasst:
Umsetzen der Residuen des komplexen Restklassensystems in Daten in Binärcode,
wobei der Schritt des Umsetzens der Residuen des komplexen Restklassensystems in Daten in Binärcode unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird.

64. Verfahren nach Anspruch 1, wobei das Verfahren die folgenden Schritte umfasst:
Empfangen von Eingangsdaten im Residuumformat; und
Umsetzen der Residuen in Daten in Binärcode,
wobei jedes Residuum ein L-Residuum-Restklassensystemwert ist, wobei das Umsetzen der Residuen in Daten in Binärcode das Anwenden der Funktion des chinesischen Restklassentheorems oder der Funktion des chinesischen L-Restklassentheorems auf jede der L Residuen jedes Residuums, um L Ergebnisse zu produzieren, und das modulare Addieren der L Ergebnisse umfasst,
wobei das Anwenden der Funktion des chinesischen Restklassentheorems oder der Funktion des chinesischen L-Restklassentheorems auf jede der L Residuen jedes Residuums unter Verwendung der sechsten Vielzahl von Logikgattern implementiert wird, wobei die sechste Vielzahl von Gattern durch die folgenden Schritte ausgewählt wird:
Berechnen eines Werts einer Funktion des chinesischen Restklassentheorems oder des chinesischen L-Restklassentheorems für jede mögliche Residuumeingabe;
Extrahieren von logischen Gleichungen, die die berechneten Werte des chinesischen Restklassentheorems oder des chinesischen L-Restklassentheorems repräsentieren; und
Abbilden der logischen Gleichungen auf die sechste Vielzahl von Logikgattern.

65. Verfahren nach Anspruch 1, wobei die Vielzahl von Logikgattern aus der folgenden Gruppe ausgewählt wird:
angepasste digitale Logik, Standardzellenlogik, auf Zellen basierende Logik-Arrays, Gate-Arrays, am Einsatzort programmierbare Gate-Arrays und programmierbare Logikbausteine.

66. Verfahren nach Anspruch 2, wobei der Binärcode aus der folgenden Gruppe ausgewählt wird:
Einerkomplement, Vorzeichen-Betrag, vorzeichenlos binär, Zweierkomplement, Festbasis und Gleitbasis.

67. Verfahren nach Anspruch 1 zum Berechnen des Produkts zweier Residuen, modulo p_i, mit den folgenden Schritten:
Empfangen eines ersten zahlentheoretischen Logarithmus eines ersten Residuums, die einem ersten Eingangsoperanden entspricht, wobei die erste Zahlentheoretische ein Nullsymbol ist, wenn das erste Residuum null ist;
Empfangen eines zweiten zahlentheoretischen Logarithmus eines zweiten Residuums, die einem zweiten Eingangsoperanden entspricht, wobei die zweite Zahlentheoretische das Nullsymbol ist, wenn des zweiten Residuums null ist;
Addieren des ersten zahlentheoretischen Logarithmus und des zweiten zahlentheoretischen Logarithmus modulo $p_i - 1$ zu dem Produkt einer Summe des ersten Eingangsoperanden und des zweiten Eingangsoperanden modulo $p_i - 1$;
Erzeugen einer Exponentiierung der Summe des ersten Eingangsoperanden und des zweiten Eingangsoperanden modulo $p_i - 1$, wobei der Schritt des Erzeugens einer Exponentiierung durch Eingeben der Summe modulo $p_i - 1$ in die dritte Vielzahl von Logikgattern erzielt wird, wobei die dritte Vielzahl von Logikgattern durch die folgenden Schritte ausgewählt wird:
Berechnen eines Werts einer zahlentheoretischen Expontiierungsfunktion für jede mögliche Eingabe;
Extrahieren von logischen Gleichungen, die die berechneten Werte der zahlentheoretischen Exponentiierungsfunktion repräsentieren;
Abbilden der logischen Gleichung auf die dritte Vielzahl von Logikgattern;
wobei die Ausgabe der dritten Vielzahl von Logikgattern das Produkt des ersten Residuums und des zweiten Residuums, modulo p_i, ist.

68. Verfahren nach Anspruch 67, wobei der Schritt des Addierens des ersten und des zweiten zahlentheoretischen Logarithmus unter Verwendung einer modularen Addiererschaltung erzielt wird.

69. Verfahren nach Anspruch 67, ferner mit dem folgenden Schritt:
Setzen der Ausgabe der dritten Vielzahl von Logikgattern auf Null, wenn der erste zahlentheoretische Logarithmus und/oder der zweite zahlentheoretische Logarithmus ein Nullsymbol ist.

70. Verfahren nach Anspruch 1 zur Reduktion eines N-Bit-Binäroperanden auf ein Residuum, modulo p_i, mit den folgenden Schritten:
Empfangen eines N-Bit-Operanden, der in $q_i + 1$ Gruppen von Bit aufgeteilt wird;
Nullerweitern der $N_i - 1$ niedrigstwertigen Bit;
Eingeben jeder der übrigen q_i Aufteilungen von Bit Q_{ij} für $j \in \{0, 1, 2, ..., q_i - 1\}$ in entsprechende q_i Vielzahlen von Logikgattern, wobei jede der q_i Vielzahlen von Logikgattern ein Residuum der Eingangsaufteilung produziert;
Addieren der nullerweiterten $N_i - 1$ niedrigstwertigen Bit und der q_i Residuen der q_i Aufteilungen, modulo p_i, um ein Residuum modulo p_i des N-Bit-Operanden zu produzieren.

71. Verfahren nach Anspruch 70, wobei die q_i Vielzahlen von Logikgattern durch die folgenden Schritte ausgewählt werden:
Berechnen eines Werts einer modularen Reduktionsfunktion für jede mögliche Eingabe für jede der übrigen q_i Aufteilungen;
Extrahieren von logischen Gleichungen, die die berechneten Werte der modularen Reduktionsfunktion repräsentieren; und
Abbilden der logischen Gleichungen auf q_i Vielzahlen von Logikgattern.

72. Verfahren nach Anspruch 71, ferner mit dem Schritt des Minimierens der logischen Gleichungen vor dem Schritt des Abbildens der logischen Gleichungen auf die q_i Vielzahlen von Logikgattern.

73. Verfahren nach Anspruch 1 zum Umsetzen eines L-Operanden-RNS-Werts in eine binäre Darstellung unter Verwendung des chinesischen Restklassentheorems (CRT), mit den folgenden Schritten:
Eingeben von L Residuen in entsprechende L Vielzahlen von Logikgattern, um L Ergebnisse zu produzieren;
Addieren der L Ergebnisse modulo M, um eine binäre Darstellung eines L-Operanden-RNS-Werts zu produzieren,
wobei die sechste Vielzahl von Logikgattern durch die folgenden Schritte ausgewählt wird:
Berechnen eines Werts einer CRT-Funktion für jede mögliche Eingabe;
Extrahieren von logischen Gleichungen, die die berechneten Werte der CRT-Funktion repräsentieren; und
Abbilden der logischen Gleichung auf die sechste Vielzahl von Logikgattern.

74. Verfahren nach Anspruch 73, ferner mit dem Schritt des Minimierens von logischen Gleichungen vor dem Schritt des Abbildens der logischen Gleichung auf die Vielzahl von Logikgattern.

75. Vorrichtung zur Durchführung mathematischer Berechnungen unter Verwendung von Residuumarithmetik, mit einem oder mehreren der folgenden:
ein Mittel zum Umsetzen von Daten in Binärcode in Residuen, wobei die Eingangsdaten in Binärcode binäre Operanden mit einer Vielzahl von Bit umfassen, wobei das Mittel zum Umsetzen der Daten in Binärcode in Residuen ein Mittel zum Berechnen von Werten des Residuums modulo p_i mindestens einer Residuumuntergruppe der Vielzahl von Bit der binären Operanden umfasst,
wobei das Mittel zum Berechnen von Werten des Residuums modulo p_i mindestens einer Untergruppe der Vielzahl von Bit der binären Operanden eine entsprechende Vielzahl von Logikgattern umfasst, wobei die Vielzahl von Logikgattern durch die folgenden Schritte ausgewählt wird:
Berechnen der Werte des Residuums modulo p_i für die mindestens eine Untergruppe der Vielzahl von Bit der binären Operanden;
Extrahieren von logischen Gleichungen, die die berechneten Werte des Residuums modulo p_i der mindestens einen Untergruppe der Vielzahl von Bit der binären Operanden repräsentieren; und
Abbilden der logischen Gleichungen auf die Vielzahl von Logikgattern;
ein Mittel zum Umsetzen von Residuen aus dem komplexen Restklassensystem in das quadratische Restklassensystem, wobei jedes Residuum des komplexen Restklassensystems einen imaginären Residuumoperanden b_i umfasst, wobei das Mittel zum Umsetzen der Residuen aus dem komplexen Restklassensystem in das quadratische Restklassensystem ein Mittel zum Multiplizieren des imaginären Residuumoperanden des eingegebenen komplexen Restklassensystems, b_i, mit ĵ umfasst, wobei das Mittel zum Multiplizieren des imaginären Residuumoperanden des eingegebenen komplexen Restklassensystems, b_i, mit ĵ eine zweite Vielzahl von Logikgattern umfasst, wobei die zweite Vielzahl von Logikgattern durch die folgenden Schritte ausgewählt wird:
Berechnen eines Werts des Produkts von b_i und ĵ modulo p_i für jeden möglichen modularen Datenoperanden;
Extrahieren von logischen Gleichungen, die berechnete Werte der Multiplizieren-mit-ĵ-Funktion repräsentieren; und
Abbilden der logischen Gleichungen auf die zweite Vielzahl von Logikgattern;
ein Mittel zum Berechnen modularer Produkte von Residuen, wobei das Mittel zum Berechnen modularer Produkte der Residuen des quadratischen Restklassensystems ein Mittel zum Berechnen eines zahlentheoretischen Logarithmus modulo p_i für jede der Residuen des quadratischen Restklassensystems, wovon das modulare Produkt berechnet wird, und ein Mittel zum Berechnen eines zahlentheoretischen Exponenten modulo p_i der Summe der zahlentheoretischen Logarithmen der Residuen des quadratischen Restklassensystems umfasst, wobei das Mittel zum Berechnen eines zahlentheoretischen Logarithmus

modulo p_i für jede der Residuen des quadratischen Restklassensystems, wovon das modulare Produkt berechnet wird, und ein Mittel zum Berechnen eines zahlentheoretischen Exponenten modulo p_i der Summe der zahlentheoretischen Logarithmen der Residuen des quadratischen Restklassensystems eine dritte Vielzahl von Logikgattern umfassen, wobei die dritte Vielzahl von Logikgattern durch die folgenden Schritte ausgewählt wird:

Berechnen des zahlentheoretischen Logarithmus modulo p_i für jedes mögliche Residuum;

Berechnen des zahlentheoretischen Exponenten modulo p_i für jede mögliche Summe;

Extrahieren von logischen Gleichungen, die berechnete Werte der Funktionen des zahlentheoretischen Logarithmus und des Exponenten repräsentieren; und

Abbilden der logischen Gleichungen auf die dritte Vielzahl von Logikgattern;

ein Mittel zum Berechnen modularer Summen von Residuen, wobei das Mittel zum Berechnen modularer Summen der Produkte von Residuen ein Mittel zum Produzieren einer vollen Summe der Produkte von Residuen umfasst, wobei die volle Summe der Produkte eine Vielzahl von Bit aufweist, die der vollen Summe der Produkte entspricht, und ein Mittel zum Berechnen von Werten mindestens einer Untergruppe der Vielzahl von Bit, die der vollen Summe der Produkte entspricht, modulo p_i, wobei das Mittel zum Berechnen von Werten mindestens einer Untergruppe der Vielzahl von Bit, die der vollen Summe der Produkte entspricht, modulo p_i, eine vierte Vielzahl von Logikgattern umfasst, wobei die vierte Vielzahl von Logikgattern durch die folgenden Schritte ausgewählt wird:

Berechnen von Werten des Residuums modulo p_i für die mindestens eine Untergruppe der vierten Vielzahl von Bit, die der vollen Summe der Residuenprodukte entspricht;

Extrahieren von logischen Gleichungen, die die berechneten Werte des Residuums modulo p_i der mindestens einen Untergruppe der Vielzahl von Bit, die der vollen Summe der Residuenprodukte entspricht, repräsentieren und

Abbilden der logischen Gleichungen auf die vierte Vielzahl von Logikgattern;

ein Mittel zum Umsetzen von Residuen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem, wobei das Mittel zum Umsetzen der modularen Summen aus dem quadratischen Restklassensystem in das komplexe Restklassensystem ein Mittel zum Berechnen des Produkts von 2^{-1} und der Summe des Residuums z_i des quadratischen Restklassensystems und z_i^* umfasst, wobei das Produkt von 2^{-1} und der Summe des Residuums z_i des quadratischen Restklassensystems und z_i^* die reelle Komponente des Residuums des komplexen Restklassensystems ist, und Berechnen des Produkts von $\hat{\jmath}^{-1}2^{-1}$ und der Differenz von z_i minus z_i^* modulo p_i, wobei das Produkt von $\hat{\jmath}^{-1}2^{-1}$ und der Differenz von z_i minus z_i^* modulo p_i die imaginäre Komponente des Residuums des komplexen Restklassensystems ist, wobei das Mittel zum Berechnen des Produkts von 2^{-1} und der Summe des Residuums z_i des quadratischen Restklassensystems und z_i^* und das Mittel zum Berechnen des Produkts von $\hat{\jmath}^{-1}2^{-1}$ und der Differenz von z_i minus z_i^* modulo p_i eine fünfte Vielzahl von Logikgattern umfassen, wobei die fünfte Vielzahl von Logikgattern durch die folgenden Schritte ausgewählt wird:

Berechnen eines Werts des Produkts von z_i minus z_i^* und $\hat{\jmath}^{-1}2^{-1}$ modulo p_i für jede mögliche Differenz;

Berechnen eines Werts des Produkts der Summe von z_i und z_i^* und 2^{-1} für jede mögliche Summe;

Extrahieren von logischen Gleichungen, die berechnete Werte der Multiplizieren-mit-$\hat{\jmath}^{-1}2^{-1}$- und Multiplizieren-mit-2^{-1}-Funktionen repräsentieren; und

Abbilden der logischen Gleichungen auf die fünfte Vielzahl von Logikgattern; und

ein Mittel zum Umsetzen von Residuen in Daten in Binärcode, wobei jedes Residuum ein L-Residuum-Restklassensystemwert ist, wobei das Mittel zum Umsetzen der Residuen in Daten in Binärcode ein Mittel zum Anwenden der Funktion des chinesischen Restklassentheorems oder ein Mittel zum Anwenden der Funktion des chinesischen L-Restklassentheorems auf jede der L Residuen jedes Residuums zum Produzieren von L Ergebnissen und zum modularen Addieren der L Ergebnisse umfasst,

wobei das Mittel zum Anwenden der Funktion des chinesischen Restklassentheorems oder das Mittel zum Anwenden der Funktion des chinesischen L-Restklassentheorems auf jede der L Residuen jedes Residuums des komplexen Restklassensystems eine sechste Vielzahl von Logikgattern umfasst, wobei die sechste Vielzahl von Gattern durch die folgenden Schritte ausgewählt wird:

Berechnen eines Werts einer Funktion des chinesischen Restklassentheorems oder des chinesischen L-Restklassentheorems für jede mögliche Residuumeingabe;

Extrahieren von logischen Gleichungen, die die berechneten Werte des chinesischen Restklassentheorems oder des chinesischen L-Restklassentheorems repräsentieren; und

Abbilden der logischen Gleichungen auf die sechste Vielzahl von Logikgattern.

Es folgen 7 Blatt Zeichnungen

Anhängende Zeichnungen

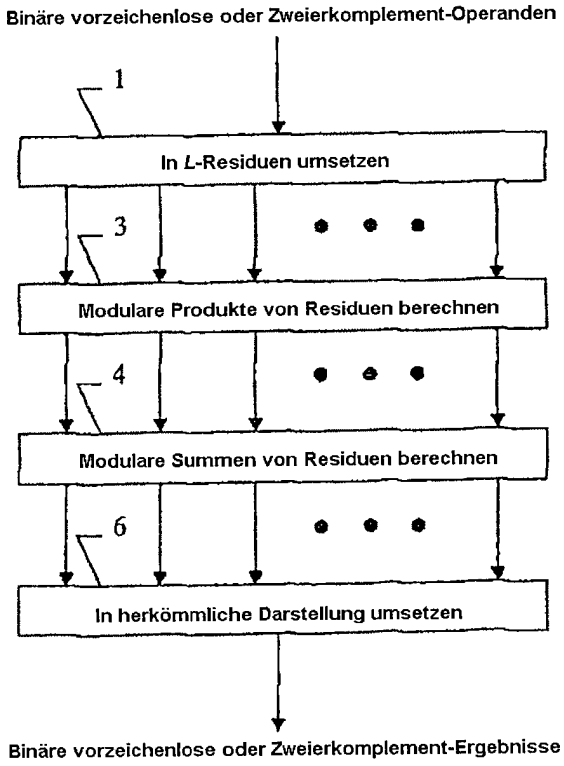

FIG. 1

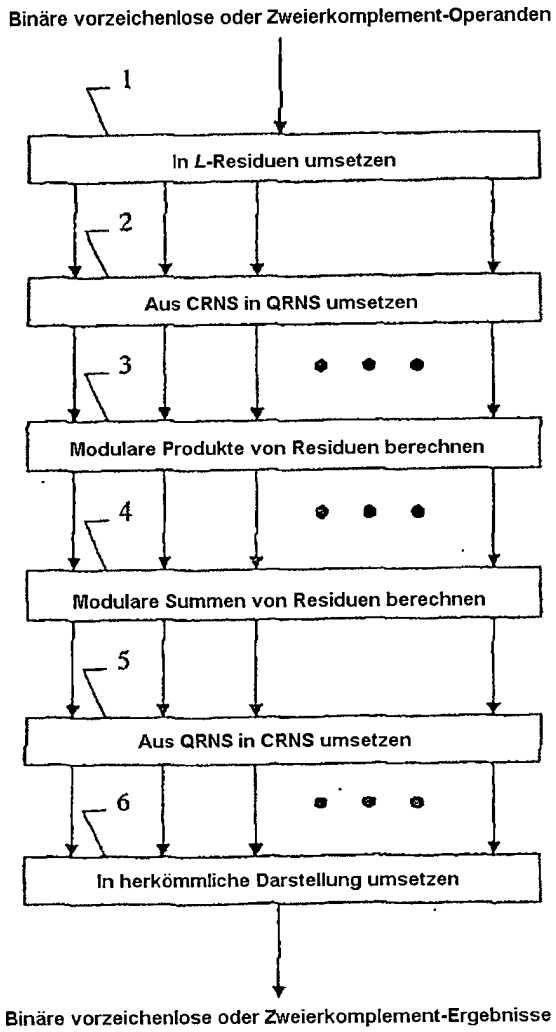
Binäre vorzeichenlose oder Zweierkomplement-Operanden
1
In L-Residuen umsetzen
2
Aus CRNS in QRNS umsetzen
3
Modulare Produkte von Residuen berechnen
4
Modulare Summen von Residuen berechnen
5
Aus QRNS in CRNS umsetzen
6
In herkömmliche Darstellung umsetzen
Binäre vorzeichenlose oder Zweierkomplement-Ergebnisse

FIG. 2

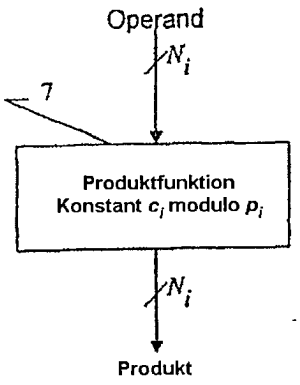

FIG. 3

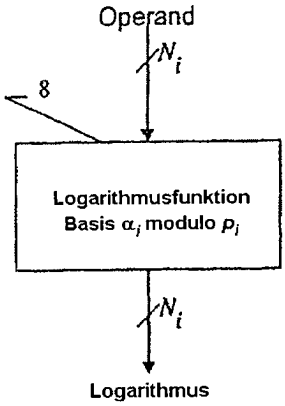

FIG. 5

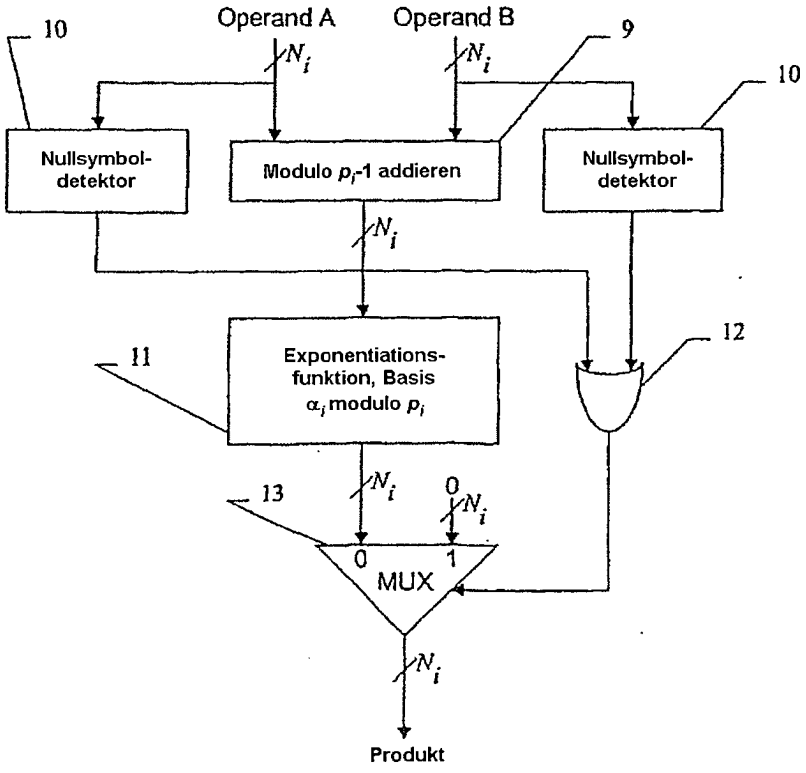

FIG. 6

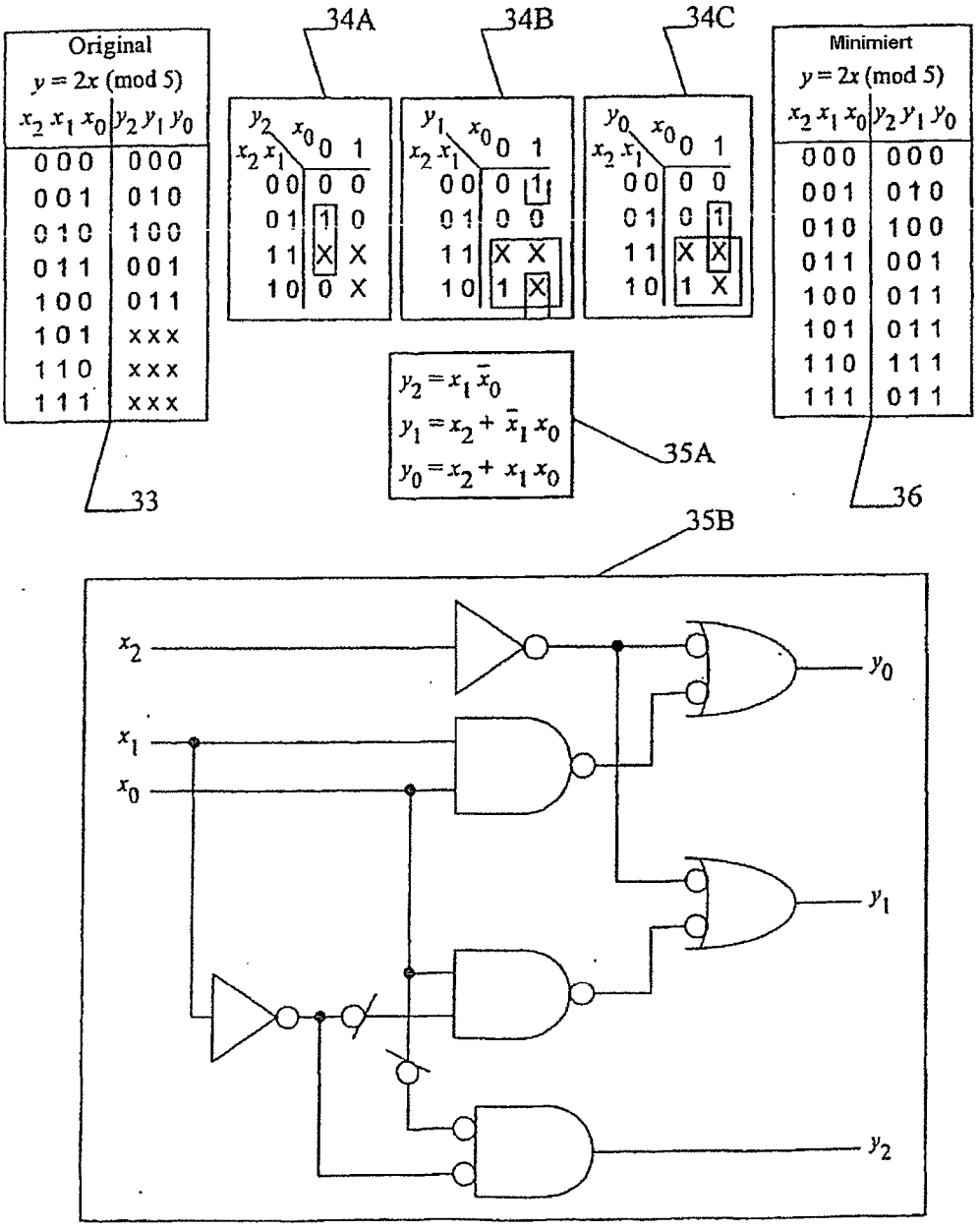
Original
y = 2x (mod 5)
x2 x1 x0 | y2 y1 y0
000 | 000
001 | 010
010 | 100
011 | 001
100 | 011
101 | xxx
110 | xxx
111 | xxx
33
34A
34B
34C
y2 = x1 x̄0
y1 = x2 + x̄1 x0
y0 = x2 + x1 x0
35A
Minimiert
y = 2x (mod 5)
x2 x1 x0 | y2 y1 y0
000 | 000
001 | 010
010 | 100
011 | 001
100 | 011
101 | 011
110 | 111
111 | 011
36
35B
x2
x1
x0
y0
y1
y2

FIG. 4

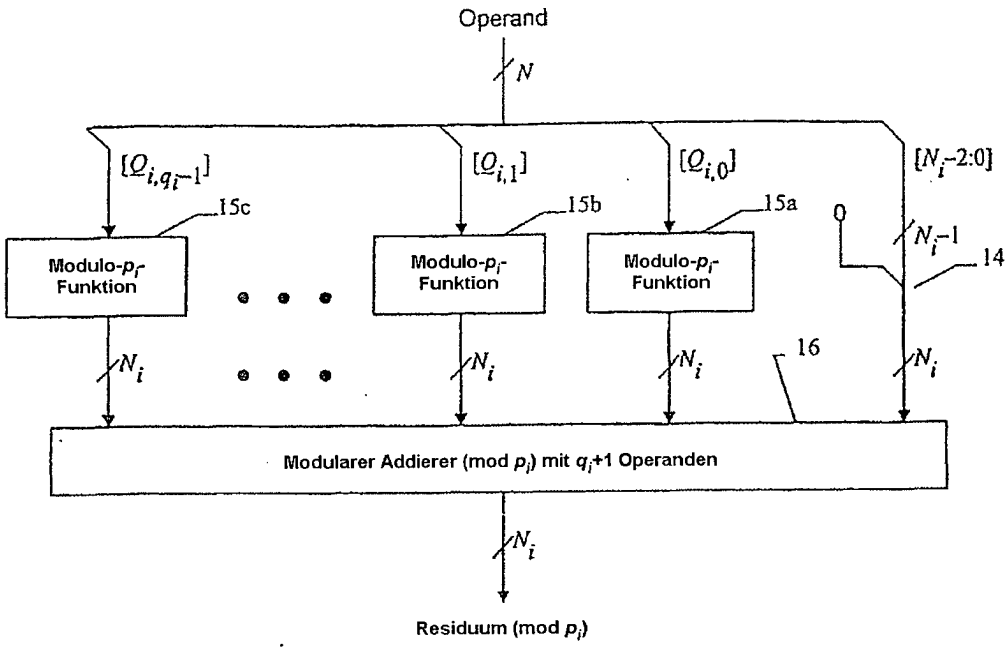
Operand
N
[Q_{i,q_i-1}]
[Q_{i,1}]
[Q_{i,0}]
[N_i-2:0]
15c
15b
15a
Modulo-p_i-Funktion
Modulo-p_i-Funktion
Modulo-p_i-Funktion
0
N_i-1
14
N_i
N_i
N_i
16
N_i
Modularer Addierer (mod p_i) mit q_i+1 Operanden
N_i
Residuum (mod p_i)

FIG. 7

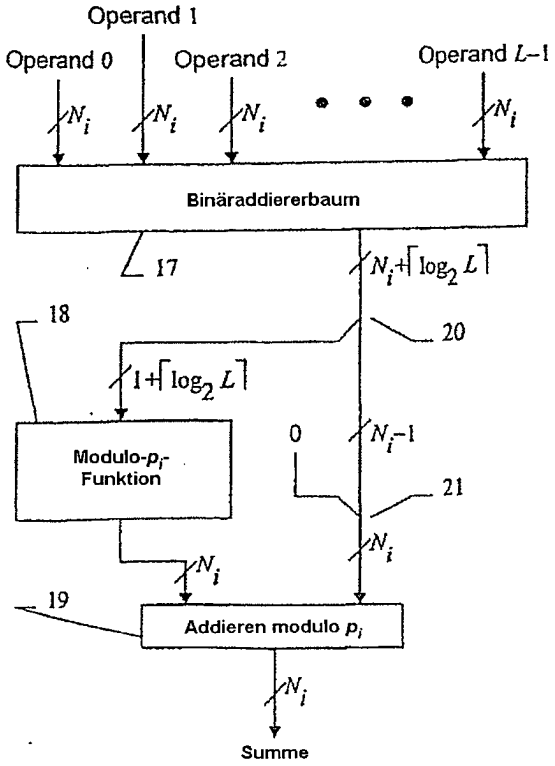
Operand 1
Operand 0
Operand 2
Operand L-1
N_i
N_i
N_i
N_i
Binäraddiererbaum
17
N_i+⌈log_2 L⌉
18
20
1+⌈log_2 L⌉
Modulo-p_i-Funktion
0
N_i-1
21
N_i
N_i
19
Addieren modulo p_i
N_i
Summe

FIG. 8

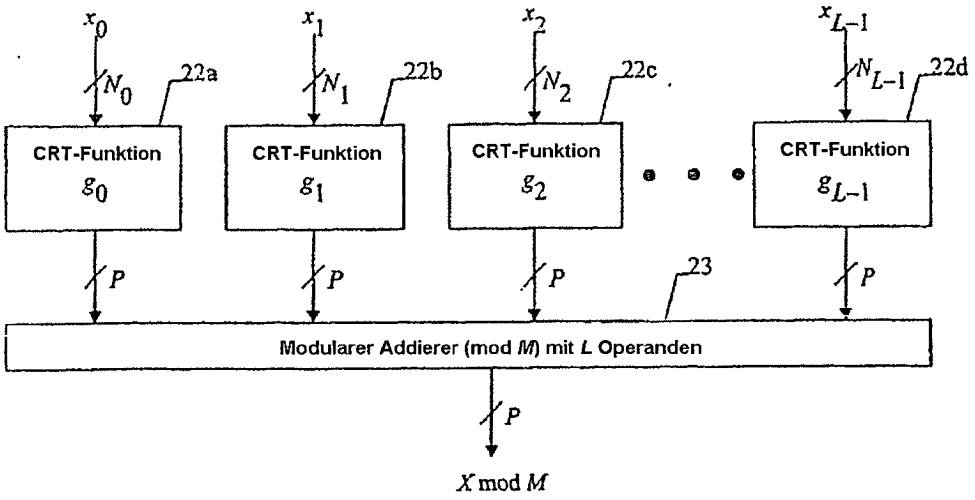
x_0
N_0
22a
x_1
N_1
22b
x_2
N_2
22c
x_{L-1}
N_{L-1}
22d
CRT-Funktion g_0
CRT-Funktion g_1
CRT-Funktion g_2
CRT-Funktion g_{L-1}
P
23
Modularer Addierer (mod M) mit L Operanden
X mod M

FIG. 9

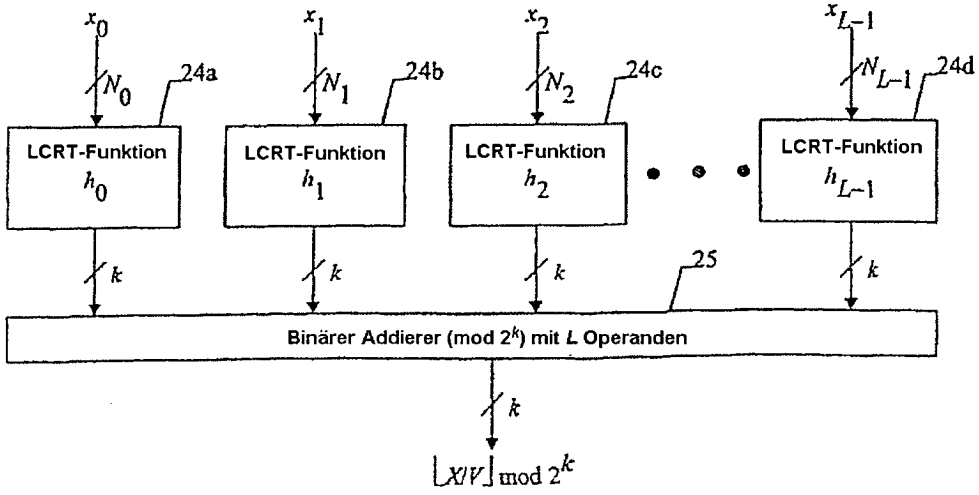
x_0
N_0
24a
x_1
N_1
24b
x_2
N_2
24c
x_{L-1}
N_{L-1}
24d
LCRT-Funktion h_0
LCRT-Funktion h_1
LCRT-Funktion h_2
LCRT-Funktion h_{L-1}
k
25
Binärer Addierer (mod 2^k) mit L Operanden
$\lfloor X/V \rfloor \bmod 2^k$

FIG. 10

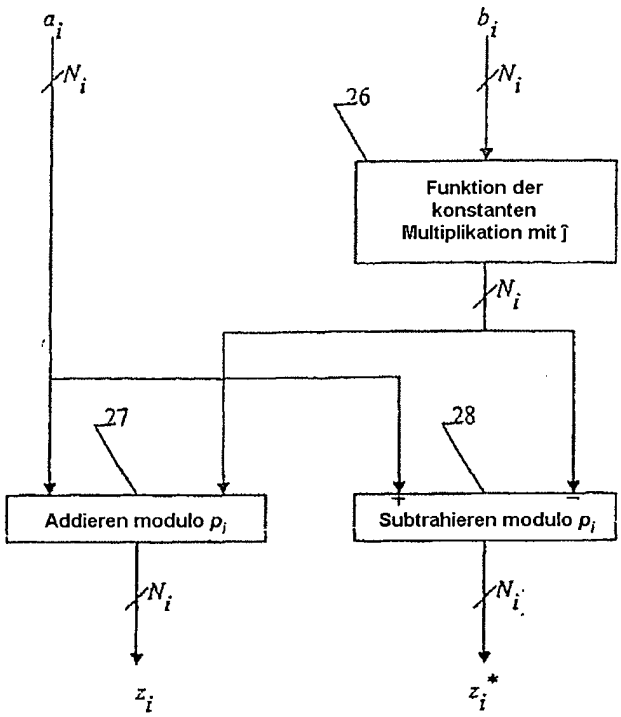
a_i
N_i
b_i
N_i
26
Funktion der konstanten Multiplikation mit $\hat{\jmath}$
N_i
27
28
Addieren modulo p_i
Subtrahieren modulo p_i
N_i
N_i
z_i
z_i^*

FIG. 11

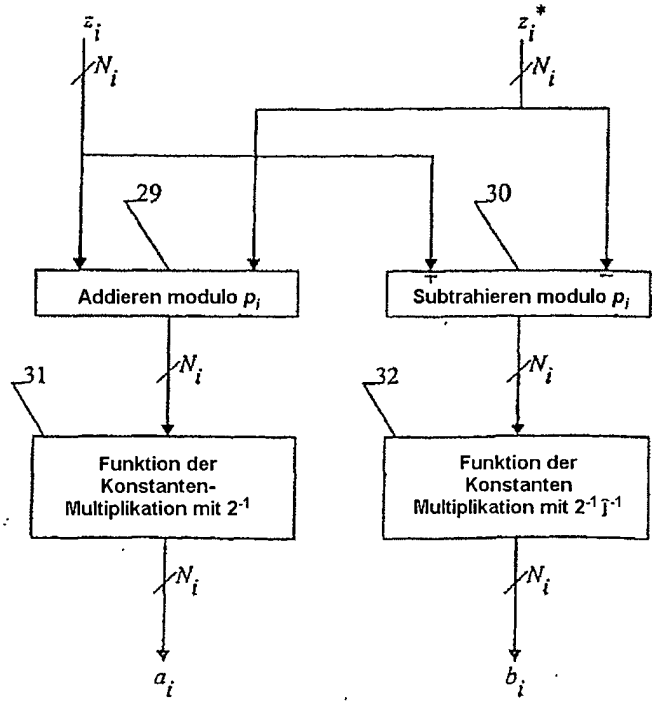
z_i
N_i
z_i^*
N_i
29
30
Addieren modulo p_i
Subtrahieren modulo p_i
N_i
N_i
31
32
Funktion der Konstanten-Multiplikation mit 2^{-1}
Funktion der Konstanten Multiplikation mit $2^{-1}\hat{\jmath}^{-1}$
N_i
N_i
a_i
b_i

FIG. 12