(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4182226号
(P4182226)
(45)発行日　平成20年11月19日(2008.11.19)　(24)登録日　平成20年9月12日(2008.9.12)

(51)Int.Cl.　F I
G09C 1/00 (2006.01)　G09C 1/00 650A

請求項の数 2　(全 14 頁)

(21)出願番号　特願2005-242956(P2005-242956)
(22)出願日　平成17年8月24日(2005.8.24)
(65)公開番号　特開2007-57811(P2007-57811A)
(43)公開日　平成19年3月8日(2007.3.8)
審査請求日　平成19年10月31日(2007.10.31)

特許法第３０条第１項適用　２００５年３月３日　社団法人電子情報通信学会発行の「電子情報通信学会技術研究報告　信学技報Ｖｏｌ．１０４　Ｎｏ．７０８」に発表

早期審査対象出願

前置審査

(73)特許権者　504139662
国立大学法人名古屋大学
愛知県名古屋市千種区不老町１番
(74)代理人　100082500
弁理士　足立　勉
(72)発明者　▲高▼木　直史
名古屋市千種区不老町１番　国立大学法人名古屋大学内
(72)発明者　カイハラ、マルセロ　エミリオ
名古屋市千種区不老町１番　国立大学法人名古屋大学内

審査官　速水　雄太

最終頁に続く

(54)【発明の名称】剰余系の計算方法及び装置並びにプログラム

(57)【特許請求の範囲】
【請求項１】
ｒ進の整数Ｍを法とする剰余系において（ただし、Ｍとｒとは互いに素）、前記法Ｍ、ｒ進でｎ桁の変数Ｙ及び変数Ｘが入力されたときに前記変数Ｙを上位（ｎ－ｍ）桁のY_H及び下位ｍ桁のY_Lに分割する分割手段と、
前記変数Ｙを分割した上位（ｎ－ｍ）桁のY_H、前記変数Ｘ及び前記法Ｍで
$X \cdot Y_H \bmod M$
を計算して出力する第１乗算剰余算器と、
前記変数Ｙを分割した下位ｍ桁のY_L、前記変数Ｘ及び前記法Ｍで
$X \cdot Y_L \cdot r^{-m} \bmod M$
を計算して出力する第２乗算剰余算器と、
前記第１乗算剰余算器の出力及び前記第２乗算剰余算器の出力を加算し、その加算結果を出力する加算器と、
を備えたことを特徴とする乗算剰余演算装置。
【請求項２】
請求項１に記載の乗算剰余演算装置において、
前記加算器の加算結果を入力し、前記法Ｍによる剰余算を行って出力する剰余算器を備えたことを特徴とする乗算剰余演算装置。
【発明の詳細な説明】
【技術分野】

【０００１】
本発明は、剰余系の計算方法及び装置並びにプログラムに関する。
【背景技術】
【０００２】
従来、ネットワーク上で送受信されるデータのセキュリティを確保するために、データを暗号化・復号化するＲＳＡ（Rivest-Shamir-Adleman)等の公開鍵暗号システムが用いられている。
【０００３】
その公開鍵暗号システムでは、使用される公開鍵のサイズ（桁数）を大きくすれば、不正な方法による暗号の解読が困難となり、情報を送受信する際のセキュリティをより強化することができる。
【０００４】
ところが、ＲＳＡ等の公開鍵暗号システムでは、暗号化と復号化のために剰余系指数演算を行う必要があり、その剰余系指数演算の量は公開鍵のサイズにともなって多くなる。
そして、ほとんどの場合、その剰余系指数演算は非常に大きな整数の乗算剰余算（剰余系乗算）を繰返し計算することにより実現されており、その乗算剰余算の繰返し計算には膨大な時間が必要であった。
【０００５】
したがって、この乗算剰余算の繰り返し計算の実行速度を速くすることができれば、使用する鍵のサイズをさらに大きくでき、ネットワーク上で送受信されるデータのセキュリティ強化を図ることができる。
【０００６】
乗算剰余算の繰返し計算を高速化する方法として、変数をモンゴメリの領域に変換して個々の乗算剰余算をモンゴメリ乗算に置き換える方法があり（例えば、非特許文献１参照）、さらに、そのモンゴメリ乗算の高速化法として基数を増す方法がある（例えば、非特許文献２参照）。
【０００７】
また、個々の乗算剰余算の高速化の方法としてインターリーブ法で基数を増す方法がある（例えば、非特許文献３参照）。
【非特許文献１】P.L. Montgomery, "Modular Multiplication without Trial Division," Mathematics of Computation, vol.44,no.170,pp.519-521, Apr.1985.
【非特許文献２】S.E Eldridge and C.D. Walter, "Hardware Implementation of Montgomery's Modular Multiplication Algorithm," IEEE Transactions on Computers, vol.42,no.6,pp.693-699,Jun.1993.
【非特許文献３】N. Takagi, "A Radix-4 Modular Multiplication Hardware Algorithm for Modular Exponentiation," IEEE Transactions on Computers, vol.41, no.8, pp.949-956, Aug. 1992.
【発明の開示】
【発明が解決しようとする課題】
【０００８】
ところで、上記非特許文献２及び非特許文献３に記載されている改良された乗算剰余算方法は、演算の基数を増すことにより、演算に必要なクロックサイクル数を削減するものである。
【０００９】
しかし、基数が増すと演算回路が複雑になり、演算のサイクル時間が大きくなる。従って、基数が増すにつれ、高速化の割合は小さくなるという問題がある。
本発明は、こうした問題に鑑みなされたもので、剰余系における乗算剰余算の繰り返し計算をサイクル時間を増すことなく高速化するための計算方法を提供することを目的とする。
【課題を解決するための手段】

【００１０】
本発明の理解をより明確にするために、特許請求の範囲に記載した課題解決手段を具体的に解説する前に、本発明の技術的思想の創作過程について説明する。
前述したように、ＲＳＡ等の公開鍵暗号システムでは、暗号化と復号化のために非常に大きな整数の乗算剰余算（剰余系乗算）を繰返し計算することにより実現されている。
【００１１】
このような、大きな整数に関する繰返し計算においては、その大きな整数を上位部分と下位部分とに分割して、上位部分の計算と下位部分の計算とを並列処理して高速化することが考えられる。
【００１２】
ところが、乗算剰余算は、被乗数に乗数を掛け、その掛け算の結果を法Ｍで割り、そのときの余りを得るという演算であるため、乗数を分割して並列処理をするメリットが得られない。
【００１３】
これを分かりやすく説明するために、以下に具体的な例により説明する。
例えば、法Ｍ＝９７５３とする乗算剰余算、　５４３２×４３２１　ｍｏｄ　９７５３を考える。
【００１４】
５４３２×４３２１　ｍｏｄ　９７５３をそのまま計算すると、
５４３２×４３２１　ｍｏｄ　９７５３
＝２３４７１６７２　ｍｏｄ　９７５３＝５９５４
となり、８桁を４桁で割る割り算を行う必要がある。
【００１５】
次に、乗数の４３２１を上位２桁の４３と下位２桁の２１とに分割して計算すると、
５４３２×４３２１　ｍｏｄ　９７５３
＝（５４３２×４３００　ｍｏｄ　９７５３
＋５４３２×２１　ｍｏｄ　９７５３）　ｍｏｄ　９７５３　となり、上位部分と下位部分とを並列計算することができるようになる。
【００１６】
実際に計算すると、乗数の上位部分は、２３３５７６００　ｍｏｄ　９７５３＝８９１８となり、下位部分は、１１４０７２　ｍｏｄ　９７５３＝６７８９になる。
このように、乗数４３２１を上位２桁と下位２桁に分割すると、下位部分は、６桁を４桁で割る割り算となり、確かに計算が簡単になる。
【００１７】
ところが、上位部分の計算は、結局８桁を４桁で割る割り算が必要となる。つまり、乗算剰余算においては、乗数を単純に上位部分と下位部分とに分割して、それを並列計算しても、結局、上位部分で分割前と同じ桁数同士の割り算（ここでは、８桁を４桁で割る割り算）を行う必要がある。
【００１８】
このように、乗算剰余算では、乗数を上位部分と下位部分に分割して計算しても、並列処理を行う利点が得られなかった。
そこで、本発明では、剰余系を新たに定義する領域に変換し、乗数を上位部分と下位部分とに分割した場合に、剰余演算の部分で上位部分、下位部分ともに計算の桁数を減らして並列計算ができるようにし、ひいては計算の高速化を可能としたのである。
【００１９】
ここで、乗算剰余算と新たに定義する領域との関係を図１を参照しつつ説明する。
図１に示すように、剰余系における変数Ｕ，Ｖを新たに定義する領域では、各々Ｘ＝Ｕ・Ｒ　ｍｏｄ　Ｍ、Ｙ＝Ｖ・Ｒ　ｍｏｄ　Ｍに変換する。また、剰余系における乗算剰余算Ｕ・Ｖ　ｍｏｄ　ＭをＸ・Ｙ・$Ｒ^{-1}$　ｍｏｄ　Ｍ＝Ｕ・Ｖ・Ｒ　ｍｏｄ　Ｍに置き換える。
【００２０】

ここで、Mがｒ進であるとき、$R=r^m$とすると、図１のＳ１００に示すように、被乗数Ｕは、$X=U\cdot r^m \bmod M$に変換される。また乗数Ｖも同様に、$Y=V\cdot r^m \bmod M$に変換される（図１のＳ１０２を参照）。つまり、Ｕ，Ｖが新たに定義する領域のＸ，Ｙに変換される。

【００２１】

そして、乗算剰余算を、$U\cdot V \bmod M$から、新たに定義する領域における、$X\cdot Y\cdot r^{-m} \bmod M$に置換する（図１のＳ１０４を参照）。

このようにして、パラメータｍを導入することにより、ｎ桁の乗数Ｙを上位（ｎ－ｍ）桁のY_Hと下位ｍ桁のY_Lとの二つの部分に分割し、これらを並列に処理することが可能となる。ここで、０＜ｍ＜ｎ、ｍ：整数とする。

【００２２】

このように、その新たに定義した領域において、変数（乗数）Ｙを上位部分Y_Hと下位部分Y_Lとに分けて、それらを並列に計算し、その計算結果を逆変換して元の領域に戻すことにより最終的に乗算剰余算の結果を得るという剰余系の計算方法がある。

【００２３】

その計算方法とは、以下に記載のように、整数Ｍを法とする剰余系において、変数Ｕ，Ｖを法Ｍと互いに素でＭより小さな定数Ｒを用いて、変数$X=U\cdot R \bmod M$、変数$Y=V\cdot R \bmod M$、に変換し、剰余系における乗算剰余算、$U\cdot V \bmod M$を演算

$$X\cdot Y\cdot R^{-1} \bmod M \quad \cdots \text{式}1$$

に置換し、剰余系における計算と同じ計算を行い、その計算結果Ｚを演算

$$Z\cdot R^{-1} \bmod M \quad \cdots \text{式}2$$

にて逆変換して剰余系における計算結果を得ることを特徴とする。

【００２４】

このような計算方法によれば、各変数Ｕ，Ｖが定数Ｒで、$X=U\cdot R \bmod M$、$Y=V\cdot R \bmod M$、に変換され、剰余系における乗算剰余算$U\cdot V \bmod M$が新たに定義される$X\cdot Y\cdot R^{-1} \bmod M$に置換される。この、変数の変換と乗算剰余算式の置換とによって得られる代数系を「新たに定義した領域」と呼ぶことにすると、剰余系での計算が新たに定義した領域において、同じように計算できる。

【００２５】

この新たに定義した領域における乗算剰余算においては、前述したように乗数Ｙを上位部分Y_Hと下位部分Y_Lとに分割して計算することができる。

したがって、剰余系の演算を新たに定義した領域において計算しようとすると、上記の変換や逆変換が必要となるものの、乗算剰余算において乗数Ｙを上位部分Y_Hと下位部分Y_Lとに分けて計算できるので、例えば公開鍵暗号等の計算のように、剰余系において乗算剰余算を繰返し行わなければならない計算を高速化することができる。

【００２６】

つまり、新たに定義した領域において、上位部分Y_H及び下位部分Y_Lの計算を並列計算によって、剰余系における乗算剰余算と同じ計算結果を得ることができるので、剰余系において乗算剰余算を繰返し行わなければならない計算を高速化することができるのである。

【００２７】

ここで、上記式１及び式２における$\bmod M$は、Ｍを法とする剰余算であり、その計算の結果は通常０から（Ｍ－１）の値をとるが、ここでは計算結果が法Ｍと合同であり、Ｍ以上あるいは負である場合も含むものである。なお、上記式１及び式２における$\bmod M$に限らず、本明細書において、$\bmod M$はすべて同様の意味である。

【００２８】

そして、新たに定義した領域において、以下に記載のように、変数Ｙがｒ進でｎ桁であるとき、定数Ｒを、

$$R=r^m$$

とし、変数Ｙを

$Y = Y_H \cdot r^m + Y_L$・・・式３

によって、上位（ｎ－ｍ）桁のY_Hと下位ｍ桁のY_Lとに分割し、（ｍは、ｍ＜ｎを満たす整数）式１を、

$(X \cdot Y_H \bmod M + X \cdot Y_L \cdot R^{-1} \bmod M) \bmod M$・・・式４

に変換して、式４の

$X \cdot Y_H \bmod M$・・・式４ａ

と、

$X \cdot Y_L \cdot R^{-1} \bmod M$・・・式４ｂ

と、を並列処理で実行するようにするとよい。

【００２９】

このように、式４ａによる計算と式４ｂとによる計算とを並列処理すれば、剰余系における乗算剰余算に対応する新たに定義した領域での計算速度を容易に高速化することができる。

【００３０】

すなわち、式４ａによる計算と式４ｂとによる計算とは各々の計算方法が異なるためその計算速度が異なるので、式４ａによる計算と式４ｂとによる計算とを各々の計算速度に合わせるようにする。

【００３１】

例えば、式４ａによる計算の計算速度が速ければ、乗数Ｙを分割する際の上位部分Y_Hの桁数（ｎ－ｍ）を下位部分Y_Lの桁数ｍよりも多くし、逆に、式４ｂによる計算の計算速度が速ければ、上位部分Y_Hの桁数（ｎ－ｍ）を下位部分Y_Lの桁数ｍよりも少なくするようにする。すると、両方の計算をほぼ同時に終了させることができるので、全体での計算時間を容易に短縮、換言すれば、計算速度を高速化することができるのである。

【００３２】

以上のように、新たに定義した領域において、乗数Ｙを上位部分Y_Hと下位部分Y_Lとに分割して計算することによって、剰余系における乗算剰余算の演算速度を高速化することができるが、分割した上位部分Y_Hを計算するための式４ａによる計算と下位部分Y_Lを計算するための式４ｂによる計算とには、「背景技術」で述べたように種々の計算方法がある。

【００３３】

例えば、式４ａによる計算にインターリーブ法を用い、式４ｂによる計算にモンゴメリ乗算における計算方法を用いると、従来のプログラムや演算回路を利用することができるので、プログラムや演算装置を容易に構成することができる。したがって、プログラム作成や演算装置のコストダウン等が可能になる。

【００３４】

なお、式４ａと式４ｂとによる計算は、必ずしも独立している必要はなく、式４ａと式４ｂとによる計算を行っていく過程で同期をとって、お互いの中間計算結果をやり取りしつつ計算を行うようにしてもよい。

【００３５】

また、ｒ進、ｎ桁の変数Ｙにおいて、ｎ桁とは変数Ｙの桁数として設定された桁数のことであり、例えば、ｒ＝２でｎ＝８（つまり、２進８桁）の場合、Ｙ＝００００１０１０のように、桁の上位桁（この場合上位４桁）が０であるものも含んでいる。

【００３６】

以上に説明した剰余系の計算方法を装置化したものが請求項１に記載の乗算剰余系演算装置である。すなわち、請求項１に記載の乗算剰余演算装置は、ｒ進の整数Ｍを法とする剰余系において（ただし、Ｍとｒとは互いに素）、法Ｍ、ｒ進でｎ桁の変数Ｙ及び変数Ｘが入力されたときに変数Ｙを上位（ｎ－ｍ）桁のY_H及び下位ｍ桁のY_Lに分割する分割手段（１０：この欄においては、発明に対する理解を容易にするため、必要に応じて「発明を実施するための最良の形態」欄において用いた符号を付すが、この符号によって請求の範囲を限定することを意味するものではない。）と、変数Ｙを分割した上位（ｎ－ｍ）桁

のY_H、変数Ｘ及び法
Ｍで
　　$X \cdot Y_H \bmod M$
を計算して出力する第１乗算剰余算器（２０）と、変数Ｙを分割した下位ｍ桁のY_L、変数Ｘ及び法Ｍで
　　$X \cdot Y_L \cdot r^{-m} \bmod M$
を計算して出力する第２乗算剰余算器（３０）と、第１乗算剰余算器（２０）の出力及び第２乗算剰余算器（３０）の出力を加算し、その加算結果を出力する加算器（４０）と、
を備えたことを特徴とする乗算剰余演算装置である。

【００３７】

このように構成された乗算剰余演算装置の出力、つまり加算器（４０）から出力される計算結果は、０から（Ｍ－１）の値以外に、法Ｍと合同であり、かつ、Ｍ以上あるいは負である場合の計算結果を得ることができる。

【００３８】

そして、このように構成された乗算剰余演算装置によれば、第１乗算剰余算器（２０）において実行される上位（ｎ－ｍ）桁の計算と、第２乗算剰余算器（３０）において実行される下記ｍ桁の計算とが並列計算される。そして、変数Y_HとY_Lとは分割される前の変数Ｙに比べ、桁数が減っているので、各々の乗算剰余算器（２０，３０）での計算速度は速くなる。したがって、剰余系における乗算剰余算に対応する新たに定義した領域での計算速度を容易に高速化することができる。

【００３９】

また、上記計算方法にて説明したように、第１乗算剰余算器（２０）で実行される計算（式４ａに相当）と第２乗算剰余算器（３０）で実行される計算（式４ｂ相当）とが並列処理されているので、剰余系における乗算剰余算に対応する新たに定義した領域での計算速度を容易に高速化することができる。

【００４０】

すなわち、第１乗算剰余算器（２０）で実行される計算と第２乗算剰余算器（３０）で実行される計算とは各々の計算方法が異なるため、その計算速度が異なる。そこで、第１乗算剰余算器（２０）で実行される計算と第２乗算剰余算器（３０）で実行される計算とを各々の計算速度に合わせて実行する。

【００４１】

例えば、第１乗算剰余算器（２０）での計算速度が速ければ、乗数Ｙを分割する際の上位部分Y_Hの桁数（ｎ－ｍ）を下位部分Y_Lの桁数ｍよりも多くし、逆に、第２乗算剰余算器（３０）での計算速度が速ければ、上位部分Y_Hの桁数（ｎ－ｍ）を下位部分Y_Lの桁数ｍよりも少なくするようにする。すると、両方の乗算剰余算器（２０，３０）での計算をほぼ同時に終了させることができるので、全体での計算時間を容易に短縮、換言すれば、計算速度を高速化することができるのである。

【００４２】

以上のように、新たに定義した領域において、乗数Ｙを上位部分Y_Hと下位部分Y_Lとに分割して計算することによって、剰余系における乗算剰余算の演算速度を高速化することができるが、分割した上位部分Y_Hを計算するための第１乗算剰余算器（２０）及び下位部分Y_Lを計算するための第２乗算剰余算器（３０）には種々の回路構成がある。

【００４３】

例えば、第１乗算剰余算器（２０）にインターリーブ法による乗算剰余を行う回路を用い、第２乗算剰余算器（３０）にモンゴメリ乗算における計算を行う回路を用いると、従来の回路を利用することができるので、回路を容易に構成することができる。したがって乗算剰余演算装置のコストダウン等が可能になる。

【００４４】

ところで、第１乗算剰余算器（２０）の処理と第２乗算剰余算器（３０）の処理とは必ずしも独立している必要はなく、第１剰余算器（２０）の処理と第２乗算剰余算器（３０

）の処理との過程で同期をとって、お互いの中間処理結果をやり取りしつつ処理を行うように各々を構成してもよい。

【発明を実施するための最良の形態】

【００４８】

以下に、本発明の実施形態を式及び図面とともに説明する。

（新たに定義した領域における計算方法の説明）

Ｒをr^mとする。ただし、０＜ｍ＜ｎで、r^mは整数とする。このとき、Ｒ＝r^mはＭと互いに素である。

【００４９】

この新たに定義した領域における計算を効率よく行う計算方法として、基数をｒとした場合について以下に示す。

法をＭ（r^{n-1}＜Ｍ＜r^n、かつ、Ｍとｒとが互いに素）とし、ｎ桁の被乗算Ｘ、ｎ桁の乗算Ｙ（０≦Ｘ，Ｙ＜Ｍ）を入力とし、$Z=X\cdot Y\cdot r^{-m} \bmod M$を出力する演算であり、具体的には以下の手順で実行される。

【００５０】

ステップ１として、Ｓ，Ｔを０に初期化し、被乗数ＸをＡに代入し、乗数Ｙをパラメータｍにより、上位（ｎ－ｍ）桁のY_Hと下位ｍ桁のY_Lとに分割して、各々をB_HとB_Lとに代入する。

【００５１】

すなわち、下位B_Lはｍ桁、上位B_Hは（ｎ－ｍ）桁となる。

ステップ２としてＡとB_Hに従来のインターリーブ法を適用して、（ｎ－ｍ）桁分の乗算剰余算を行い、その結果としてＳを得る。また、ＡとB_Lに従来のモンゴメリ乗算における計算方法を適用して、ｍ桁分の計算を行い、Ｔを得る。

【００５２】

このステップ２のインターリーブ法による乗算剰余算と、モンゴメリ乗算における計算とは並列処理で実行される。

そして、並列処理が終了して、ＳとＴの両者が得られると、ステップ３へ移行する。

【００５３】

ステップ３として、Ｍを法としてＳとＴとの加算剰余算を行い、出力Ｚを得る。

このようにして最終的に得られたＺが新たに定義した領域での計算結果となる。

以上の手順を式に表すと下記のようになる。

【００５４】

入力：Ｍ：r^{n-1}＜Ｍ＜r^n，ｇｃｄ（Ｍ，ｒ）＝１

０≦Ｘ，Ｙ＜Ｍ

出力：$Z=X\cdot Y\cdot r^{-m} \bmod M$

演算手順：

ステップ１：Ａ：＝Ｘ；Ｍ：＝Ｍ；Ｓ：＝０；Ｔ：＝０；

B_H：＝Y_H；B_L：＝Y_L　（但し、$Y=Y_H r^m+Y_L$）

ステップ２：｛Ｓ：＝Interleaved_modmul（Ａ，B_H，Ｍ）；

Ｔ：＝Montgomery_modmul（Ａ，B_L，Ｍ）；｝

ステップ３：Ｚ：＝Ｓ＋Ｔ ｍｏｄ Ｍ；

なお、ステップ２の中で、Interleaved_modmul（Ａ，B_H，Ｍ）は、前述のインターリーブ法による乗算剰余算を示しており、Ａは被乗数、B_Hは乗数、Ｍは法を示している。

【００５５】

また、Montgomery_modmul（Ａ，B_L，Ｍ）とは、前述のモンゴメリ乗算における計算方法を示しており、Ａは被乗数、B_Lは乗数、Ｍは法を示している。

また、ステップ２における｛　｝内の計算は並列に行う。

【００５６】

次に、以上に説明した新たに定義した領域における計算において、乗数を上位部分Y_Hと下位部分Y_Lの桁数を同じにした場合、つまりｍをｎ／２とした場合の演算方法につい

て図２を参照しつつ説明する。

【００５７】

図２は、新たに定義した領域での計算手順を被乗数Ｘと乗数Ｙとが各々８ビットの場合について模式的に示した図である。

ここでは、基数を２とし、基数２の符号付きディジット表現により、すべての加減算を桁上げの伝搬なしに行うものとする。

【００５８】

ｎビットの被乗数Ｘ、ｎビットの乗数Ｙ（０≦Ｘ、Ｙ＜Ｍ）を入力とし、Ｚ＝Ｘ・Ｙ・$2^{-n/2}$ ｍｏｄ Ｍを出力する演算であり、具体的には以下の手順で実行される。

ステップ１として、Ｓ，Ｔを０に初期化し、被乗数ＸをＡに代入する。さらに、乗数Ｙをパラメータｍ（＝ｎ／２）により、上位のY_Hと下位のY_Lとに分割し、各々をB_HとB_Lとに代入する（図２のＳ１１０参照）。

【００５９】

ステップ２として、ｎビットのＡとｎ／２ビットのB_Hに従来のインターリーブ法を適用し、また、Ａとｎ／２ビットのB_Lに従来のモンゴメリ乗算における計算方法を適用して、各々の計算を並列処理で実行し、各々の計算結果であるＳとＴとを得る（図２のＳ１１２参照）。

【００６０】

ここで、上位部分の計算は以下の手順（Ｈ１）～（Ｈ７）をｎ／２回繰り返すことにより行われる。

（Ｈ１）Ｓを左（上位）へ１桁シフトする。

【００６１】

（Ｈ２）シフト後のＳの（ｎ＋１）番目の桁（シフトした際の桁上がり）、ｎ番目の桁、（ｎ－１）番目の桁をq_1、つまり、$q_1=[s_{n+1}s_ns_{n-1}]$とする。

（Ｈ３）q_1が０よりも大きければＳから法Ｍを減じて、それを新たなＳとし、q_1が０よりも小さければＳに法Ｍを加算したものを新たなＳとする。

【００６２】

（Ｈ４）被乗数Ａと乗数B_Hの最上位ビットb_{n-1}との論理積をとり、その結果にＳを加算したものを新たなＳとする。

（Ｈ５）新たなＳの（ｎ＋１）番目の桁、ｎ番目の桁、（ｎ－１）番目の桁をq_2、つまり、$q_2=[s_{n+1}s_ns_{n-1}]$とする。

【００６３】

（Ｈ６）q_2が０よりも大きければＳから法Ｍを減じて、それを新たなＳとし、q_2が０よりも小さければＳに法Ｍを加算したものを新たなＳとする。

（Ｈ７）B_Hを左へ１ビットシフトする。

【００６４】

また、下位部分の演算は以下の手順（Ｌ１）～（Ｌ４）をｎ／２回繰り返すことにより行われる。

（Ｌ１）被乗数Ａと乗数B_Lの最下位ビットb_0との論理積をとり、その結果にＴを加算したものを新たなＴとする。

【００６５】

（Ｌ２）Ｔの最下位桁t_0をq_3とする。

（Ｌ３）q_3が０でなければ、Ｔに法Ｍを加算して、それを右へ１桁シフトしたものを新たなＴとする。一方、q_3が０であればＴを右へ１桁シフトし、それを新たなＴとする。

【００６６】

（Ｌ４）B_Lを右へ１ビットシフトする。

以上のように、並列処理で上位部分の乗算剰余算結果Ｓと下位部分の乗算剰余算結果Ｔとを得る。

【００６７】

次に、ステップ３として、ＳとＴとを加算し、それを新たなＳとする（図２のＳ１１４参照）。

そして、新たなＳの（ｎ＋１）番目の桁、ｎ番目の桁、（ｎ－１）番目の桁をq_2、つまり、$q_2=[s_{n+1}s_ns_{n-1}]$とする。

【００６８】

q_2が０よりも大きければＳから法Ｍを減じて、それを新たなＳとし、q_2が０よりも小さければＳに法Ｍを加算したものを新たなＳとする。

ステップ４として、Ｓを周知の演算方法により、基数２の符号付きディジット表現から通常の２進数表現に変換し、その結果を出力Ｚとする（図２では省略）。

【００６９】

ステップ５として、出力Ｚが０よりも小さければ、出力Ｚに法Ｍを加算して、それを出力Ｚとする（図２のＳ１１６参照）。

このようにして最終的に得られた出力Ｚが乗数Ｙと被乗数Ｘとの法Ｍに基づく新たに定義した領域における乗算剰余算結果となる。

【００７０】

以上の手順を式に表すと下記のようになる。

入力：$M: 2^{n-1}<M<2^n$、ｇｃｄ（Ｍ、２）＝１
　　Ｘ、Ｙ：$0 \leqq X, Y<M$
出力：$Z=X\cdot Y\cdot 2^{-n/2} \bmod M\ (0 \leqq Z<M)$
演算手順：
ステップ１：$A:=X$；$B_H:=Y_H$；$B_L:=Y_L$；$S:=0$；$T:=0$；$M:=M$；
ステップ２：ｆｏｒ　ｉ：＝１　ｔｏ　ｎ／２
　　ｄｏ　Ｈ　ａｎｄ　Ｌ　ｉｎ　ｐａｒａｌｌｅｌ
　Ｈ：ｄｏ
　　$S:=2\cdot S$；
　　$q_1:=[s_{n+1}s_ns_{n-1}]$
　　ｉｆ　$q_1>0$　ｔｈｅｎ　$S:=S-M$
　　ｅｌｓｅｉｆ　$q_1<0$　ｔｈｅｎ　$S:=S+M$；
　　$S:=S+b_{n-1}\cdot A$；
　　$q_2:=[s_{n+1}s_ns_{n-1}]$；
　　ｉｆ　$q_2>0$　ｔｈｅｎ　$S:=S-M$
　　ｅｌｓｅｉｆ　$q_2<0$　ｔｈｅｎ　$S:=S+M$；
　　$B_H:=B_H<<1$；
　ｅｎｄｄｏ
　Ｌ：ｄｏ
　　$T:=T+b_0\cdot A_i$；
　　$q_3:=t_0$
　　ｉｆ　$q_3 \neq 0$　ｔｈｅｎ　$T:=(T+M)>>1$
　　ｅｌｓｅ　$T:=T>>1$；
　　$B_L:=B_L>>1$；
　ｅｎｄｄｏ
　ｅｎｄｆｏｒ
ステップ３：$S:=S+T$；
　　$q_2:=[s_{n+1}s_ns_{n-1}]$；
　　ｉｆ　$q_2>0$　ｔｈｅｎ　$S:=S-M$
　　ｅｌｓｅｉｆ　$q_2<0$　ｔｈｅｎ　$S:=S+M$；
ステップ４：$Z:=$ＳＤ２_to_Binary（Ｓ）；
ステップ５：ｉｆ　$Z<0$　ｔｈｅｎ　$Z:=Z+M$；

以上に説明した、基数を２とした場合の新たに定義した領域における計算では、ｎビットの乗数が同じビット数（ｎ／２）の上位部分と下位部分とに分割され、上位部分はインターリーブ法で乗算剰余算され、下位部分はモンゴメリ乗算における計算方法で計算されている。さらに、その２つの計算は並列処理で、ほぼ同時に実行されている。

【００７１】

従って、新たに定義した領域での計算によれば、ｎビットの乗数全体をインターリーブ法により乗算剰余算する場合の約１／２の時間で実行することができる。

すなわち、通常、処理するビット数が同じであれば、インターリーブ法よりもモンゴメリ乗算における計算の方が高速であるので、インターリーブ法によりｎ／２ビットの乗算剰余算（つまり、上位部分の乗算剰余算）が終了したときには、モンゴメリ乗算における計算方法によるｎ／２ビットの計算（つまり、下位部分の計算）は終了している。

【００７２】

従って、新たに定義した領域における計算によれば、乗数全体をインターリーブ法によって乗算剰余算を行う場合に比べ、約１／２の演算時間で計算を実行することができるのである。

【００７３】

なお、本実施形態では、乗数のビットを上位と下位のビット数を同じ、つまりｍをｎ／２としたが、上位部分と下位部分のビット数が異なるようにしてもよい。

例えば、インターリーブ法とモンゴメリ法との演算速度の違いを考慮すれば、全体の計算時間をより短縮できる分割方法を決定することができる。つまり、インターリーブ法とモンゴメリ乗算における計算方法との演算速度の比がｐ：ｑのときには、下位部分のビット数ｍをおよそ、ｎ・ｑ／（ｐ＋ｑ）とすれば、上位部分と下位部分の計算時間がほぼ同じになり、全体の計算時間を短縮することができる。

（乗算剰余演算装置の説明）

次に、上記説明した新たに定義した領域における計算を実行するための乗算剰余演算装置１について図３に従って説明する。

【００７４】

図３は、乗算剰余演算装置１の構成を表すブロック図である。

図３に示すように、乗算剰余演算装置１は、主に分割回路１０、乗算剰余算回路２０、モンゴメリ乗算回路３０、加算回路４０、剰余算回路５０とを備えている。

【００７５】

分割回路１０は、入力されたｒ進、ｎ桁の変数Ｙを上位（ｎ－ｍ）桁のY_Hと下位ｍ桁のY_Lとに分割するための回路である。なお、ｎ桁の変数Ｙを上位部分のY_Hと下位部分のY_Lとに分割するためのパラメータｍは、分割回路１０の内部にあらかじめ設定されていてもよいし、外部から入力されるようになっていてもよい。

【００７６】

乗算剰余算回路２０は、分割回路１０で分割された変数Ｙの上位（ｎ－ｍ）桁のY_Hと入力された変数Ｘ及び法Ｍによって

$$S = X \cdot Y_H \bmod M \quad \cdots \text{式１０}$$

を計算して、Ｓを加算回路４０に出力するための公知の回路である。

【００７７】

モンゴメリ乗算回路３０は、分割回路１０で分割された変数Ｙの下位ｍ桁のY_Lと入力された変数Ｘ及び法Ｍによって、モンゴメリ乗算に基づいた計算、

$$T = X \cdot Y_L \cdot r^{-m} \bmod M \quad \cdots \text{式２０}$$

を実行して、Ｔを加算回路４０に出力するための公知の回路である。

【００７８】

加算回路４０は、乗算剰余算回路２０の出力Ｓ、モンゴメリ乗算回路３０の出力Ｔを入力とし、Ｓ＋Ｔを計算して出力するための公知の回路である。

剰余算回路５０は、加算回路４０の出力Ｓ＋Ｔ及び法Ｍを入力として剰余算を行い、Ｚを出力するための回路、すなわち、

Ｚ＝Ｓ＋Ｔ ｍｏｄ Ｍ・・・式３０
によりＺを出力するための公知の回路である。
【００７９】
以上のように構成された乗算剰余演算装置１における計算の流れについて説明する。
乗算剰余演算装置１には、ｒ進、ｎ桁の変数Ｙ、変数Ｘ及び法Ｍが入力される。
入力された変数Ｙは、分割回路１０に入力され、変数Ｘは、乗算剰余算回路２０、モンゴメリ乗算回路３０に入力され、法Ｍは、乗算剰余算回路２０、モンゴメリ乗算回路３０及び剰余算回路５０に入力される。
【００８０】
乗算剰余演算装置１に入力された変数Ｙは、分割回路１０に入力され、（ｎ－ｍ）桁の上位部分Y_Hとｍ桁の下位部分Y_Lとに分割される。
分割された上位部分Y_Hは、乗算剰余算回路２０に入力され、変数Ｘと法Ｍとで上記式１０に従って乗算剰余算が実行され、その結果Ｓが出力される。
【００８１】
一方、分割された下位部分Y_Lは、モンゴメリ乗算回路３０に入力され、変数Ｘと法Ｍとで上記式２０に従ってモンゴメリ乗算に基づく計算が実行され、その結果Ｔが出力される。
【００８２】
そして、乗算剰余算回路２０の出力Ｓ、モンゴメリ乗算回路３０の出力Ｔが加算回路４０に入力されて加算され、その出力と法Ｍとが剰余算回路５０に入力され、上記式３０に従って、剰余算が実行され、その結果Ｚが出力される。
【００８３】
このような構成の乗算剰余演算装置１によれば、乗数Ｙを２分割し、上位部分Y_Hと下位部分Y_Lとに割り当てて各々の乗算剰余算を独立して並列処理で実行している。従って、従来のようにｎ桁の乗数をそのまま乗算剰余算する演算方法に比べ、少ない時間で乗算剰余算を実行することができる。
【００８４】
なお、本実施形態において、分割回路１０が分割手段に、乗算剰余算回路２０が第１乗算剰余算器に、モンゴメリ乗算回路３０が第２乗算剰余器に、加算回路４０が加算器に、剰余算回路５０が剰余算器に各々相当する。
【００８５】
以上、本発明の実施形態について説明したが、本発明は、本実施形態に限定されるものではなく、種々の態様を採ることができる。
例えば、本実施形態では、乗数を分割した上位部分の計算（式４ａによるの計算）にインターリーブ法を用い、下位部分の計算（式４ｂによる計算）にモンゴメリ乗算における計算を用いたが、各計算はそれらに限定されるものではなく、数学的に並列処理できる計算方法であればどのような計算方法であってもよい。
【００８６】
また、並列処理の際、式４ａと式４ｂとによる計算は、必ずしも独立している必要はなく、式４ａと式４ｂとによる計算を行っていく過程で同期をとって、お互いの中間計算結果をやり取りしつつ計算を行うようにしてもよい。
【００８７】
同様に、乗算剰余算回路２０の処理とモンゴメリ乗算回路３０の処理とは必ずしも独立している必要はなく、乗算剰余算回路２０の処理とモンゴメリ乗算回路３０の処理との過程で同期をとって、お互いの中間処理結果をやり取りしつつ処理を行うように各々を構成してもよい。
【００８８】
また、乗算剰余演算装置１では、乗除算器５０の出力をＺとしているが、加算器４０の出力を他の回路や装置、例えば、他の乗算剰余算回路に入力して、加算器４０の計算結果を基に更なる演算を行うようにしてもよい。

【図面の簡単な説明】

【００８９】

【図１】剰余系におけるｒを基数とする各変数と新たに定義した領域における表現との関係を示す図である。

【図２】新たに定義した領域における演算手順を被乗数Ｘと乗数Ｙとが各々８ビットの場合について模式的に示した図である。

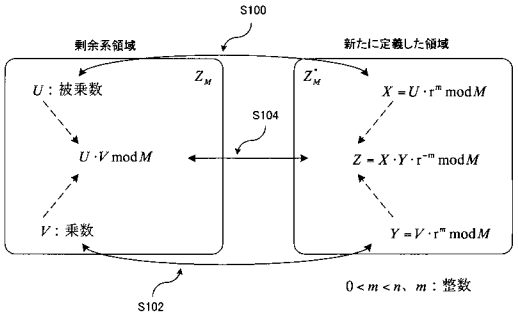
【図３】乗算剰余演算装置１の構成を表すブロック図である。

【符号の説明】

【００９０】

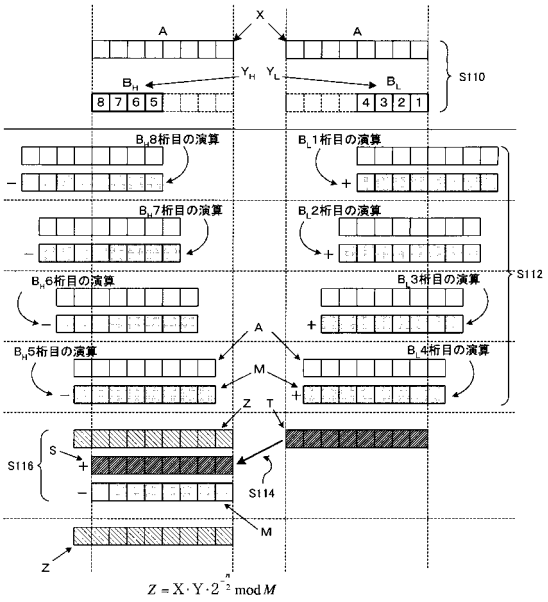
１…乗算剰余演算装置、１０…分割回路、２０…乗算剰余算回路、３０…モンゴメリ乗算回路、４０…加算回路、５０…剰余算回路。

【図１】



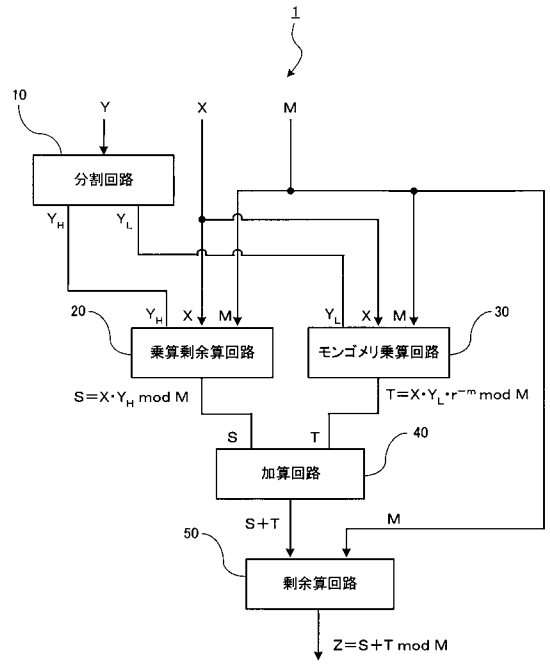
剰余系領域と基新たに定義した領域とにおける剰余表現

【図２】



m=n／2の場合の乗算剰余演算の手順

【図３】


1
10
Y
X
M
分割回路
Y_H
Y_L
20
Y_H X M
乗算剰余算回路
30
Y_L X M
モンゴメリ乗算回路
$S=X\cdot Y_H \bmod M$
$T=X\cdot Y_L\cdot r^{-m} \bmod M$
S
T
40
加算回路
S+T
M
50
剰余算回路
$Z=S+T \bmod M$

フロントページの続き

(56)参考文献　特開平０５－３２４２７７（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０９Ｃ　　１／００