(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3615622号
(P3615622)
(45)発行日　平成17年2月2日(2005.2.2)　(24)登録日　平成16年11月12日(2004.11.12)

(51)Int.Cl.[7]　F I
G06F　7/552　G06F　7/552　A

請求項の数 11　(全 22 頁)

(21)出願番号　特願平8-169322
(22)出願日　平成8年6月28日(1996.6.28)
(65)公開番号　特開平10-21057
(43)公開日　平成10年1月23日(1998.1.23)
審査請求日　平成15年2月28日(2003.2.28)

(73)特許権者　503121103
株式会社ルネサステクノロジ
東京都千代田区丸の内二丁目４番１号
(74)代理人　100089071
弁理士　玉村　静世
(72)発明者　中田　邦彦
東京都小平市上水本町五丁目２０番１号
株式会社日立製作所　半導体事業部内

審査官　山崎　慎一

(56)参考文献　特開平０９－２７４５６０（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.[7]，ＤＢ名)
G06F　7/552

(54)【発明の名称】マイクロコンピュータ

(57)【特許請求の範囲】

【請求項１】

コプロセッサと、マイクロプロセッサと、ＲＯＭと、電気的にデータを書換え可能に保持する不揮発性メモリとを１個の半導体基板に備えたマイクロコンピュータであって、

前記コプロセッサは、ｎを演算値のビット数を表わす正の整数、Ｎを 0＜N＜2n となるnビットの正の整数、Ａｉｎを0≦Ａｉｎ＜2n となるｎビットの正の整数、Ａｏｕｔを0≦Ａｏｕｔ＜2n となるｎビットの正の整数、ｋを０又は正の整数とするとき、演算式がＡｏｕｔ = f(Ａｉｎ) mod N +kN で与えられ、入力値 Ａｉｎ と出力値 Ａｏｕｔ の値の範囲を０以上、2n 未満の整数とする剰余演算を行う演算手段と、この演算手段による上記剰余演算を制御する制御手段とを有し、

前記Ａｏｕｔ = f(Ａｉｎ) mod N +kN で与えられる剰余演算は、 Ａｏｕｔ = Ａｉｎ・Ｂ・Ｒ－１ mod N +kN とＡｏｕｔ = Ａｉｎ・Ｂ mod N +kNの演算式で夫々与えられる剰余演算を含み、Ｒは2n、Ｒ－１はＲ・Ｒ－１ mod N = １ を満たす 0＜Ｒ－１ ≦N となるnビットの正の整数、Ｂは0≦Ｂ＜2n となるnビットの正の整数であり、

前記マイクロプロセッサは、前記コプロセッサに前記演算Ａｏｕｔ = Ａｉｎ・Ｂ・Ｒ－１ mod N +kN に必要な入力値を設定して当該演算処理を指示し、その演算結果を利用するものであり、

前記ＲＯＭは、前記マイクロプロセッサにべき乗剰余演算を実行させるための動作プログラムを保有し、前記演算Ａｏｕｔ = Ａｉｎ・Ｂ・Ｒ－１ mod N +kN は前記べき乗剰余演算に含まれる演算であり、

前記不揮発性メモリは、前記べき乗剰余演算対象とされるデータを電気的な書き換え可能に保持するものであることを特徴とするマイクロコンピュータ。

【請求項２】

前記演算式Ａｏｕｔ ＝Ａｉｎ・Ｂ・Ｒ－１ mod Ｎ +kN で与えられる剰余演算は、ｔ＝(Ａｉｎ・B+M・N)/Rで与えられる演算処理と、これに続いてif t≧R then return t-N else return tで与えられ、ｔ－Ｎ又はｔを前記Ａｏｕｔとする演算処理とを含み、前演算手段は、前記 t≧Rを、ｎビットのｔのオーバーフローによって検出することを特徴とする請求項１記載のデータ処理装置。

【請求項３】

前記演算手段は、演算式Ａｏｕｔ ＝Ａｉｎ・Ｂ・Ｒ－１ mod Ｎ +kN で与えられる剰余演算を実行するための、直列的に接続された積和演算器を備えて成るものであることを特徴とする請求項１記載のデータ処理装置。

【請求項４】

前記演算式Ａｏｕｔ ＝Ａｉｎ・Ｂ・Ｒ－１ mod Ｎ +kN で与えられる剰余演算は、ｔ＝(Ａｉｎ・B+M・N)/Rで与えられる演算処理と、これに続いてif t≧R then return t-N else return tで与えられ、ｔ－Ｎ又はｔを前記Ａｏｕｔとする演算処理とを含み、前記演算手段は、直列的に接続された積和演算器を備え、部分積の和として前記演算処理t＝(Ａｉｎ・B+M・N)/Rを実行するものであることを特徴とする請求項１記載のデータ処理装置。

【請求項５】

前記制御手段は前記直列的に接続された積和演算器を用いて乗数と被乗数が多倍長とされる多倍長乗算を選択的に実行するものであることを特徴とする請求項４記載のデータ処理装置。

【請求項６】

前記演算手段は、前記演算式Ａｏｕｔ ＝Ａｉｎ・Ｂ・Ｒ－１ mod Ｎ +kN におけるBの値を、Ａｉｎ又は1 に選択的に指定する手段を備えて成るものであることを特徴とする請求項３乃至５の何れか１項記載のデータ処理装置。

【請求項７】

前記演算式Ａｏｕｔ ＝Ａｉｎ・Ｂ・Ｒ－１ mod Ｎ +kN で与えられる剰余演算は、ｔ＝(Ａｉｎ・B+M・N)/Rで与えられる演算処理と、これに続いてif t≧R then return t-N else return tで与えられｔ－Ｎ又はｔを前記Ａｏｕｔとする演算処理とを含み、前記演算手段は、前記 t≧Rを、ｎビットのｔのオーバーフローによって検出し、また、直列的に接続された積和演算器を備え部分積の和として前記演算処理t＝(Ａｉｎ・B+M・N)/Rを実行し、また、前記演算式Ａｏｕｔ ＝Ａｉｎ・Ｂ・Ｒ－１ mod Ｎ +kN におけるBの値をＡｉｎ又は1 に選択的に指定する手段を備えて成るものであることを特徴とする請求項１記載のデータ処理装置。

【請求項８】

前記制御手段は前記直列的に接続された積和演算器を用いて乗数と被乗数が多倍長の多倍長乗算を選択的に実行するものであることを特徴とする請求項７記載のデータ処理装置。

【請求項９】

コプロセッサと、マイクロプロセッサと、ＲＯＭと、電気的にデータを書換え可能に保持する不揮発性メモリとを１個の半導体基板に備えたマイクロコンピュータであって、

前記コプロセッサは、演算結果Ａｏｕｔを次の演算の入力値Ａｉｎとし、演算式 Ａｏｕｔ ＝f(Ａｉｎ) で与えられる演算を実行し、Ａｉｎ とAout の内容を各々記憶する記憶手段と、当該記憶手段に対するＡｉｎ とAout の記憶場所の相互入れ替えを指示するフラグ手段と、前記演算式の演算実行終了後に前記フラグ手段の値を反転させることにより、Ａｉｎ とAout の値の物理的な入れ替えに代えて、Ａｉｎ とAout の記憶場所を論理的に入れ換える制御手段とを含み、

前記演算式Ａｏｕｔ ＝ｆ（A_{in}）で与えられる演算は演算式Ａｏｕｔ ＝Ａｉｎ・Ｂ・Ｒ－１ mod N+kN で与えられる剰余演算であり、

前記剰余演算は、ｔ＝(Ａｉｎ・B+M・N)/Rで与えられる演算処理と、これに続いてif t≧R

then return t-N else return tで与えられｔ－Ｎ又はｔを前記Ａｏｕｔとする演算処理とを含み、前記演算手段は、直列的に接続された前記積和演算器により、部分積の和として前記演算処理t = (Ａｉｎ・B+M・N)/Rを実行するものであり、

前記マイクロプロセッサは、前記コプロセッサに前記演算Ａｏｕｔ = Ａｉｎ・B・R－１ mod N +kN に必要な入力値を設定して当該演算処理を指示し、その演算結果を利用するものであり、

前記ＲＯＭは、前記マイクロプロセッサにべき乗剰余演算を実行させるための動作プログラムを保有し、前記演算Ａｏｕｔ = Ａｉｎ・B・R－１ mod N +kN は前記べき乗剰余演算に含まれる演算であり、

前記不揮発性メモリは、前記べき乗剰余演算対象とされるデータを電気的な書き換え可能に保持するものであることを特徴とするマイクロコンピュータ。

【請求項１０】

コプロセッサと、マイクロプロセッサと、ＲＯＭと、電気的にデータを書換え可能に保持する不揮発性メモリとを１個の半導体基板に備えたマイクロコンピュータであって、

前記コプロセッサは、ｎを演算値のビット数を表わす正の整数、Ｎを 0＜N＜2n となるnビットの正の整数、Ａｉｎを0≦Ａｉｎ＜2n となるｎビットの正の整数、Ａｏｕｔを0≦Ａｏｕｔ＜2n となるｎビットの正の整数、kを０又は正の整数とするとき、演算式がＡｏｕｔ = f(Ａｉｎ) mod N+kN で与えられ、入力値 Ａｉｎ と出力値 Ａｏｕｔ の値の範囲をNよりも大きな2n 未満とする剰余演算を行う演算手段と、この演算手段による上記剰余演算を制御する制御手段とを備え、前記Ａｏｕｔ = f(Ａｉｎ) mod N+kN で与えられる剰余演算は、 Ａｏｕｔ = Ａｉｎ・B・R－１ mod N+kN の演算式で与えられる剰余演算を含み、Ｒは2n、R－１はR・R－１ mod N = １ を満たす 0＜R－１ ≦N となるnビットの正の整数、Ｂは0≦B＜2n となるnビットの正の整数であり、前記演算手段は、演算式Ａｏｕｔ = Ａｉｎ・B・R－１ mod N+kN、で与えられる剰余演算を実行するための、直列的に接続された積和演算器を備え、

前記マイクロプロセッサは、前記コプロセッサに前記演算Ａｏｕｔ = Ａｉｎ・B・R－１ mod N +kN に必要な入力値を設定して当該演算処理を指示し、その演算結果を利用するものであり、

前記ＲＯＭは、前記マイクロプロセッサにべき乗剰余演算を実行させるための動作プログラムを保有し、前記演算Ａｏｕｔ = Ａｉｎ・B・R－１ mod N +kN は前記べき乗剰余演算に含まれる演算であり、

前記不揮発性メモリは、前記べき乗剰余演算対象とされるデータを電気的な書き換え可能に保持するものであることを特徴とするマイクロコンピュータ。

【請求項１１】

前記演算式Ａｏｕｔ = Ａｉｎ・B・R－１ mod N+kN で与えられる剰余演算は、t = (Ａｉｎ・B+M・N)/Rで与えられる演算処理と、これに続いてif t≧R then return t-N else return tで与えられｔ－Ｎ又はｔを前記Ａｏｕｔとする演算処理とを含み、前記演算手段は、直列的に接続された前記積和演算器により、部分積の和として前記演算処理t = (Ａｉｎ・B+M・N)/Rを実行するものであることを特徴とする請求項１０記載のマイクロコンピュータ。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、剰余演算機能を備えたデータ処理装置に適用して有効なものに関し、特に、剰余乗算演算やべき乗剰余演算を用いた符号化及び復号化装置に適用して有効な技術に関するものである。

【０００２】

【従来の技術】

暗号技術の一つとして公開鍵暗号技術が有る。例えば、図１４に示されるように、送信者は、メッセージMに対して「Ｃ＝M^{e} ｍｏｄＮ」による符号化で暗号文Ｃを形成し、受信

者は暗号文Ｃに対して、「Ｍ＝Ｃ^{d} ｍｏｄＮ」による復号化でメッセージＭを得る。上記符号化におけるパラメータｅ，Ｎは公開とされ、上記復号化におけるｄは非公開とされる。前記符号化及び復号化のための演算式は、べき乗剰余演算であり、代表的に「Ｘ^{Y} ｍｏｄ Ｎ」（Ｘ， Ｙ， Ｎ は正の整数）と記述する事ができる。「ｍｏｄ」は剰余演算を意味し、べき乗剰余演算「Ｘ^{Y} ｍｏｄ Ｎ」は、Ｘ^{Y} をＮで除算した余りを解とするものである。

【０００３】

符号化／復号化装置において、べき乗剰余演算「Ｘ^{Y} ｍｏｄ Ｎ」（Ｘ， Ｙ， Ｎ は正の整数）を用いることにより、高度なセキュリティが得られることは、一般に知られており、例えば、池野、小山”現代暗号理論”、電子情報通信学会編 （昭和６１年）にその記載がある。

【０００４】

ここで前記Ｘ， Ｙ， Ｎ は、通常１００ビット～２０００ビット程度の非常に大きな数が使用されるため、「Ｘ^{Y} ｍｏｄ Ｎ」をいかにして高速に実行するかが、従来から数学、工学の分野で課題とされていた。

【０００５】

その一つの解法として、次に示す〔アルゴリズム１〕が古典的に知られている。

【０００６】

〔アルゴリズム１〕

```
input X, Y=e_n e_{n-1} ･･･e_1, N        ステップ1
A = X                                    ステップ2
B = X                                    ステップ3
for i = n-1 to 1 step -1 {               ステップ4
        A = A^2 mod N                    ステップ5
        if e_i=1 then A = A･B mod N      ステップ6
}                                        ステップ7
output A                                 ステップ8
```

このアルゴリズムにおいてｎはＹのビット数に対応され、e_n e_{n-1} ・・・e_1 はＹの２進数表現である。このアルゴリズムは、概略的には、自乗の剰余乗算「Ａ^{2} ｍｏｄ Ｎ」（ステップ５）及び剰余乗算「Ａ・Ｂ ｍｏｄ Ｎ」（ステップ６）を組み合わせて実行され、e_n e_{n-1} ・・・e_1 における論理値１の個数をｒ（ｅ）とすると、前記自乗の剰余乗算「Ａ^{2} ｍｏｄ Ｎ」にｎ－１回、前記剰余乗算「Ａ・Ｂ ｍｏｄ Ｎ」にｒ（ｅ）－１回の演算が繰り返し行われることになる。

【０００７】

このように〔アルゴリズム１〕では、べき乗剰余演算「Ｘ^{Y} ｍｏｄ Ｎ」 を剰余乗算「Ａ・Ｂ ｍｏｄ Ｎ」など に分解して実行するため、「Ａ・Ｂ ｍｏｄ Ｎ」 の演算機能を実現する演算装置を用いればよい。

【０００８】

しかしながら、Ａ， Ｂ も非常に大きな数であり、たとえばデータ長を各々５１２ビットとすると、中間結果ＡＢ が１０２４ビットの大きな数となる問題がある。さらに、Ａ・Ｂ をＮ で割った余りを最終結果とするため、１０２４ビット÷５１２ビット という大きな値を扱う除算を実行しなければならない。ここで乗算は乗数と被乗数を分割することにより、マイクロプロセッサ等によって並列高速処理が可能であるが、除算は並列高速処理が困難であり、剰余乗算「Ａ・Ｂ ｍｏｄ Ｎ」 に分解して実行するとしても、やはりその除算処理が高速化を阻むという問題があった。

【０００９】
特に、上記剰余乗算「Ａ・Ｂ　ｍｏｄ　Ｎ」における除算の問題を解決するため、Ｎによる除算を行なわずに剰余乗算「Ａ・Ｂ・Ｒ$^{-1}$　ｍｏｄ　Ｎ」を実行する以下の〔アルゴリズム２〕が知られている。ここで、Ｒ　は２^{n}（ｎは例えばＮのサイズ）であり、Ｒ＞Ｎ　を満たす正の整数である。
【００１０】
このＮによる除算を行なわずに剰余乗算「Ａ・Ｂ・Ｒ$^{-1}$　ｍｏｄ　Ｎ」を実行する〔アルゴリズム２〕に至るまでの論証の詳細については、ここでは説明を詳細するが、例えば、Ｍｏｎｔｇｏｍｅｒｙ，　Ｐ．　Ｌ．：　Ｍｏｄｕｌａｒ　Ｍｕｌｔｉｐｌｉｃａｔｉｏｎ　ｗｉｔｈｏｕｔ　Ｔｒｉａｌ　Ｄｉｖｉｓｉｏｎ，　Ｍａｔｈｅｍａｔｉｃｓ　ｏｆ　Ｃｏｍｐｕｔａｔｉｏｎ，　Ｖｏｌ．　４４，　Ｎｏ．　１７０，　ｐｐ．　５１９－５２１　（１９８５）、Ｄｕｓｓë，　Ｓ．Ｒ．　ａｎｄ　Ｋａｌｉｓｋｉ，　Ｂ．Ｓ．　Ｊｒ．：　Ａ　Ｃｒｙｐｔｏｇｒａｐｈｉｃ　Ｌｉｂｒａｒｙ　ｆｏｒ　ｔｈｅ　Ｍｏｔｏｒｏｌａ　ＤＳＰ５６０００，　Ａｄｖａｎｃｅｓ　ｉｎ　Ｃｒｙｐｔｏｌｏｇｙ　－ＥＵＲＯＣＲＹＰＴＯ　’９０，　Ｌｅｃｔｕｒｅ　Ｎｏｔｅｓ　ｉｎ　Ｃｏｍｐｕｔｅｒ　Ｓｃｉｅｎｃｅ　４７３，　ｐｐ．　２３０－２４４，　Ｓｐｒｉｎｇｅｒ－Ｖｅｒｌａｇ　（１９９１）に記載がある。
【００１１】
〔アルゴリズム２〕

$N' = -N^{-1} \bmod R$　　ステップ1

$M = A \cdot B \cdot N' \bmod R$　　ステップ2

$t = (A \cdot B + M \cdot N)/R$　　ステップ3

if $t \geqq N$ then return $t-N$ else return t　ステップ4

【００１２】
【発明が解決しようとする課題】
第１の問題点
上記〔アルゴリズム２〕では、そのステップ１、ステップ２でＮ’，　Ｍ　をあらかじめ計算しておくという前処理が必要になる。またＭ　の値を一時的に保持する記憶装置も新たに必要になる。
【００１３】
第２の問題点
さらに、上記〔アルゴリズム２〕において、そのステップ４でｔとＮとの大小比較を行なわなければならず、その比較のために、大きな値のｔとＮに対して実際にｔ－Ｎ　の減算処理を必要とするため、そのような減算処理を繰り返し行うことによって全体の演算時間が増大してしまう。
【００１４】
第３の問題点
また、上記〔アルゴリズム２〕のステップ３においては、中間結果Ａ・Ｂ　がＡ，　Ｂのデータ長の合計をデータ長とする大きな数となるため、これらを一時的に保持する記憶装置が新たに必要になる。さらに、〔アルゴリズム２〕では〔アルゴリズム１〕で必要であったＮによる除算が不要になったとは言え、ステップ３ではＡ，　Ｂ，　Ｍ，　Ｎ　の大きな数どうしの乗算が必要であり、これをいかにして効率良く実現するかという課題のあることが本発明者によって見い出された。特にこれをハードウェアで実現する場合、Ａ，　Ｂ，　Ｎ　の値を保持する記憶装置と、乗算を実行する演算器との間でのデータのやりとりを頻繁に行なわなければならず、記憶装置やデータパスの論理回路規模を最小限に抑える必要のあることが本発明者によって見い出された。
【００１５】

第４の問題点
さらに、〔アルゴリズム２〕で計算された剰余乗算「Ａ・Ｂ・Ｒ$^{-1}$　ｍｏｄ　Ｎ」を用いて、べき乗剰余演算　「Ｘ^{Ｙ}　ｍｏｄ　Ｎ」　をいかに効率良く計算するかという課題がある。　一例として、本発明者は〔アルゴリズム２〕を適用して〔アルゴリズム１〕を変形することにより、以下の〔アルゴリズム３〕を得た。
【００１６】
〔アルゴリズム３〕

input X, Y=e_n e_{n-1} ･･･e_1, N, R	ステップ1
B = R^2 mod N	ステップ2
A = X	ステップ3
A = A･B･R^{-1} mod N	ステップ4
B = A	ステップ5
for i = n-1 to 1 step -1 {	ステップ6
A = A^2･R^{-1} mod N	ステップ7
if e_i=1 then A = A･B･R^{-1} mod N	ステップ8
}	ステップ9
A = A･R^{-1} mod N	ステップ10
output A	ステップ11

この〔アルゴリズム３〕では、ステップ２で「Ｒ^{2}　ｍｏｄ　Ｎ」の計算が前処理として必要になる。これを例えば汎用マイクロプロセッサを用いてソフトウエアで計算した場合、計算時間がかかり、全体として〔アルゴリズム３〕の計算時間を増大させてしまうことが本発明者によって見い出された。この問題点は〔アルゴリズム１〕のステップ５，６でも同様に生じる。
【００１７】
第５の問題点
前記〔アルゴリズム３〕のステップ７で「Ａ^{2}・Ｒ$^{-1}$　ｍｏｄ　Ｎ」を計算する際、剰余乗算「Ａ・Ｂ・Ｒ$^{-1}$　ｍｏｄ　Ｎ」を用いて計算するためには、ＢにＡの値を転送するのに時間のかかることが本発明者によって明らかにされた。また、前記〔アルゴリズム３〕のステップ１０では、Ｂの値を　Ｂ＝１　にセットしなければならず、これによっても同様に時間がかかる。
【００１８】
第６の問題点
さらに〔アルゴリズム３〕においてステップ４の「Ａ　＝　Ａ・Ｂ・Ｒ$^{-1}$　ｍｏｄ　Ｎ」、ステップ７の「Ａ　＝　Ａ^{2}・Ｒ$^{-1}$　ｍｏｄ　Ｎ」、ステップ８の「Ａ　＝　Ａ・Ｂ・Ｒ$^{-1}$　ｍｏｄ　Ｎ」では、演算結果を入力値と同じ記憶装置Ａに保存しなければならない。このとき、剰余乗算「Ａ・Ｂ・Ｒ$^{-1}$　ｍｏｄ　Ｎ」は通常瞬時には実行できず、多くのクロック数を費やして入力値Ａ，　Ｂ，　Ｎ　を何度も参照しながら計算するため、途中結果および最終結果を入力値Ａ　とは同じ記憶装置に保存できず、一時記憶値（Ｔｅｍｐ）として別の記憶装置（例えばテンポラリレジスタ）に保存する必要が生じる。そのため、実質的な演算式は「Ｔｅｍｐ　←　Ａ・Ｂ・Ｒ$^{-1}$　ｍｏｄ　Ｎ」となり、たとえばステップ８を実行するためには最後に「Ａ　←　Ｔｅｍｐ」を実行しなければならず、計算時間が増大してしまうという問題があった。この問題は〔アルゴリズム１〕のステップ５、ステップ６でも同様に生じる。
【００１９】

本発明は上記事情に鑑みて成されたものであり、上記〔アルゴリズム１〕、〔アルゴリズム２〕、〔アルゴリズム３〕で代表されるような、上記種々の問題点を解決し、高速にべき乗剰余演算「 X^Y ｍｏｄ Ｎ 」を実現することを目的とする。

【００２０】

特に、上記課題を専用のハードウェアおよび汎用のマイクロプロセッサを用いて解決し、高速なべき乗剰余演算「 X^Y ｍｏｄ Ｎ」 を実現することを目的とする。

【００２１】

また本発明の別の目的は、上記の専用ハードウェアの実現において、その論理回路規模を最小限にすることにある。

【００２２】

本発明のその他の目的は、上記専用ハードウェアをＩＣカード用マイクロコンピュータと同一の半導体チップに搭載し、べき乗剰余演算「 X^Y ｍｏｄ Ｎ」を適用した符号化・復号化のためのマイクロコンピュータを低コストで使い易く実現することにある。

【００２３】

この発明の前記ならびにそのほかの目的と新規な特徴については、本明細書の記述および添付図面から明らかになるであろう。

【００２４】

【課題を解決するための手段】

本願において開示される発明のうち代表的なものの概要を説明すれば、下記の通りである。

【００２５】

すなわち、べき乗剰余演算に利用されるデータ処理装置（６）は、ｎを演算値のビット数を表わす正の整数、Ｎを $0<N<2^n$ となるｎビットの正の整数、A_{in} を $0 \leqq A_{in}<2^n$ となるｎビットの正の整数、A_{out} を $0 \leqq A_{out}<2^n$ となるｎビットの正の整数、ｋを０又は正の整数とするとき、演算式が $A_{out}=f(A_{in})\ mod\ N+kN$ で与えられ、入力値 A_{in} と出力値 A_{out} の値の範囲を０以上、2^n 未満の整数とする剰余演算を行う演算手段（３２）と、この演算手段による上記剰余演算を制御する制御手段（３１）とを１個の半導体基板に備えて構成される。

【００２６】

一般式で与えられる前記 $A_{out}=f(A_{in})\ mod\ N+kN$ の剰余演算は、 $A_{out}=A_{in}\cdot B\cdot R^{-1}\ mod\ N+kN$ と $A_{out}=A_{in}\cdot B\ mod\ N+kN$ の演算式で夫々与えられる剰余演算とすることができ、このとき、Ｒは 2^n、R^{-1} は $R\cdot R^{-1}\ mod\ N=1$ を満たす $0<R^{-1} \leqq N$ となるｎビットの正の整数、Ｂは $0 \leqq B<2n$ となるｎビットの正の整数と表現できる。

【００２７】

前記演算式 $A_{out}=A_{in}\cdot B\cdot R^{-1}\ mod\ N+kN$ で与えられる剰余演算には、ｔ ＝ （A_{in}・Ｂ＋Ｍ・Ｎ）／Ｒで与えられる演算処理と、これに続いてｉｆ ｔ≧Ｒ ｔｈｅｎ ｒｅｔｕｒｎ ｔ－Ｎ ｅｌｓｅ ｒｅｔｕｒｎ ｔで与えられｔ－Ｎ又はｔを前記 A_{out} とする演算処理とを含むことができる。図１に例示されるように、ｔとＲの比較においては、前記 ｔ≧Ｒを、ｎビットのｔのオーバーフローによって検出する。オーバーフローの検出結果に応じてｔ－Ｎの演算を行えばよく、ｔとＮを直接の比較対象とする場合に比べてｔ－Ｎの減算処理の頻度を少なくでき、剰余演算速度の高速化に寄与する。

【００２８】

図１１及び図１２に例示されるように、直列的に接続された積和演算器（３３、３４）を備え、部分積の和として前記演算処理ｔ ＝ （A_{in}・Ｂ＋Ｍ・Ｎ）／Ｒを実行することができる。これにより、部分積の一次記憶用メモリ手段を要さず、演算手段の回路規模の縮小を実現する。

【００２９】

前記制御手段は前記直列的に接続された積和演算器を用いて乗数と被乗数が多倍長とされる多倍長乗算を選択的に実行することができる。これは、べき乗剰余演算のための「R^2 mod N」の処理に必要な多倍長乗算処理に適用できる。

【００３０】

前記演算手段は、図５に例示されるように、前記演算式$A_{out} = A_{in} \cdot B \cdot R^{-1} \bmod N + kN$におけるＢの値を、$A_{in}$又は１に選択的に指定する手段（１６、１７）を備えることができる。これにより、「$A_{in} \cdot B \cdot R^{-1} \bmod N + kN$」、「$A_{in}^2 \cdot R^{-1} \bmod N + kN$」又は「$A_{in} \cdot R^{-1} \bmod N + kN$」を同じように演算することができ、値Ｂを保有するためのレジスタに値Ａを転送したり、値Ｂを保有するためのレジスタに１を設定する処理を要さず、剰余演算処理の高速化に寄与する。

【００３１】

マイクロコンピュータ（ＭＣＵ）は、図３に例示されるように前記データ処理装置（６）と、このデータ処理装置をコプロセッサとして利用するマイクロプロセッサ（５）とを１個の半導体基板に含み、前記マイクロプロセッサは、前記データ処理装置に前記演算$A_{out} = A_{in} \cdot B \cdot R^{-1} \bmod N + kN$に必要な入力値を設定して当該演算処理を指示し、その演算結果を利用する。このマイクロコンピュータによれば、剰余乗算「$A_{out} = A_{in} \cdot B \cdot R^{-1} \bmod N + kN$」、「$A_{out} = A_{in}^2 \cdot R^{-1} \bmod N + kN$」、「$A_{out} = A_{in} \cdot R^{-1} \bmod N + kN$」で与えられる演算を、積和演算器を内蔵したコプロセッサで実行し、さらにそのコプロセッサに前処理「$R^2 \bmod N$」を高速に実行するための乗算機能を備え、これらのコプロセッサの演算機能を用いて、マイクロプロセッサでべき乗剰余演算「$X^Y \bmod N$」を高速に実行することができる。

【００３２】

前記マイクロプロセッサにべき乗剰余演算を実行させるための動作プログラムが格納されたＲＯＭ（８）をマイクロコンピュータのチップに搭載することができる。このとき、前記演算$A_{out} = A_{in} \cdot B \cdot R^{-1} \bmod N + kN$は前記べき乗剰余演算に含まれる演算とされる。これにより、高度なセキュリティーの達成に寄与するマイクロコンピュータを実現できる。

【００３３】

前記べき乗剰余演算対象とされるデータを電気的な書き換え可能に保持する不揮発性メモリ（１０）を更にマイクロコンピュータチップに搭載することができる。これにより、セキュリティーを保つべきデータを取り扱うＩＣカード用マイクロコンピュータとしての利用に最適化することができる。

【００３４】

演算結果A_{out}を次の演算の入力値A_{in}とし、演算式$A_{out} = f(A_{in})$で与えられる演算を実行するデータ処理装置を、図７に例示されるように、A_{in}とA_{out}の内容を各々記憶する記憶手段（２３、２４）と、当該記憶手段に対するA_{in}とA_{out}の記憶場所の相互入れ替えを指示するフラグ手段（２７）と、前記演算式の演算実行終了後に前記フラグ手段の値を反転させることにより、A_{in}とA_{out}の値の物理的な入れ替えに代えて、A_{in}とA_{out}の記憶場所を論理的に入れ換える制御手段（２８、３１）とを含んで構成することができる。これにより、値A_{in}とA_{out}の記憶場所の入れ換えを見かけ上瞬時に実行することができ、演算時間の短縮に寄与する。

【００３５】

前記直列的に配置された積和演算器を利用した処理は以下のデータ処理装置にも適用できる。すなわち、ｎを演算値のビット数を表わす正の整数、Ｎを$0 < N < 2^n$となるｎビットの正の整数、A_{in}を$0 \leqq A_{in} < 2^n$となるｎビットの正の整数、A_{out}を$0 \leqq A_{out} < 2^n$となるｎビットの正の整数とするとき、演算式が$A_{out} = f(A_{in}) \bmod N$で与えられ、入力値$A_{in}$と出力値$A_{out}$の値の範囲

を０以上、2^n 未満の整数とする剰余演算を行う演算手段と、この演算手段による上記剰余演算を制御する制御手段とを１個の半導体基板に備えて成るデータ処理装置である。前記$A_{out} = f(A_{in})$ mod N で与えられる剰余演算は、$A_{out} = A_{in} \cdot B \cdot R^{-1}$ mod Nの演算式で与えられる剰余演算を含み、Rは2^n、R^{-1}はR・R^{-1} mod N ＝ １ を満たす ０＜R^{-1} ≦N となるｎビットの正の整数、Ｂは０≦Ｂ＜2^n となるｎビットの正の整数であり、前記演算手段は、演算式$A_{out} = A_{in} \cdot B \cdot R^{-1}$ mod N、で与えられる剰余演算を実行するための、直列的に接続された積和演算器を備える。これに対する具体的な演算態様として、前記演算式$A_{out} = A_{in} \cdot B \cdot R^{-1}$ mod N で与えられる剰余演算は、ｔ ＝ （A_{in}・Ｂ＋Ｍ・Ｎ）／Ｒで与えられる演算処理と、これに続いてｉｆ ｔ ≧Ｒ ｔｈｅｎ ｒｅｔｕｒｎ ｔ－Ｎ ｅｌｓｅ ｒｅｔｕｒｎ ｔで与えられｔ－Ｎ又はｔを前記A_{out}とする演算処理とを含み、前記演算手段は、直列的に接続された前記積和演算器により、部分積の和として前記演算処理ｔ ＝ （A_{in}・Ｂ＋Ｍ・Ｎ）／Ｒを実行する。これによっても同じく、部分積の一次記憶用メモリ手段などを必要とせず、演算手段の回路規模を縮小できる。

【００３６】

【発明の実施の形態】

（１）第１の実施の形態

上記〔アルゴリズム２〕では、ステップ４でｔ とＮ との大小比較を毎回行なわなければならず、特にｔ とＮ の値が大きな値の時には実際にｔ－Ｎ の減算を必要とするため、全体の計算時間が増大するという上記第２の問題点があった。第１の実施の形態においては、この問題点を解決するため、〔アルゴリズム２〕のステップ４を、ｔとＮとの大小比較ではなく、ｔとＲとの大小比較とするようにする。そのアルゴリズムを以下に示す。

【００３７】

〔アルゴリズム４〕

$N' = -N^{-1} \bmod R$ ステップ1

$M = A \cdot B \cdot N' \bmod R$ ステップ2

$t = (A \cdot B + M \cdot N)/R$ ステップ3

if $t \geqq R$ then return $t-N$ else return t ステップ4

上記〔アルゴリズム４〕において、Ａ，Ｂ，Ｎのビット数はｎビットであり、Ｒ＝2^n（第ｎビットまでの全ビットが論理値０、第ｎ＋１番目のビットが論理値１とされる２進数）とされる。このアルゴリズム４は、後述する演算式「$A_{out} = A_{in} \cdot B \cdot R^{-1}$ mod N ＋ｋＮ」 を実現するものであり、その処理はべき乗剰余演算のための一つのサブルーチンとして位置付けることができる。アルゴリズム４において、ｔ≧Ｒならば、ｔ－ＮがA_{out} とされ、ｔ＜ＲならばｔがA_{out}とされる。

【００３８】

上記〔アルゴリズム４〕のステップ４の大小比較と減算は、ステップ３における「ｔ ＝（Ａ・Ｂ＋Ｍ・Ｎ）／Ｒ」の演算結果ｔが所定の値2^n（＝Ｒ）以上であるか否かをｔのオーバーフローによって検出し、オーバーフローを検出したとき、ステップ３のｔをｔ－Ｎに補正するという、「オーバフロー補正」として特徴付けることができる。この特徴を実現する回路の一例が、図１に示されるオーバフロー補正回路である。図２ にはオーバフロー補正回路の比較例として前記〔アルゴリズム２〕のステップ４を実現する回路のブロック図が示されている。

【００３９】

先ず、図２では、ｔ－Ｎの減算が減算器３Ａで実行され、その演算の結果、ボロー（Ｂｏｒｒｏｗ）が発生したか否かでｔと減算結果ｔ－Ｎのどちらかをセレクタ４Ａで選択して

出力結果とする。従ってこの構成ではｔ－Ｎの減算の実行は必ず必要となる。

【００４０】

これに対して図１の構成では、ｔの値が所定の数ｎで与えられるビット数を越えているかどうかを、オーバフロービットＯＶが０か１かで判断する。これが上記のｔとＲとの大小比較に相当する。すなわち、上記の通り$R=2^n$であるから前記オーバーフロービットＯＶが１であればｔ≧Ｒとされ、オーバフロービットＯＶが０であればｔ＜Ｒとされる。

【００４１】

そこでＯＶ＝１でオーバフローが発生している時だけ、減算器３でｔ－Ｎの減算を実行し、セレクタ３で、その減算結果を選択して出力結果とする。ＯＶ＝０でオーバフローが発生していない時には、減算器３による減算は行なわず、ｔをそのままセレクタ４で選択して出力結果とする。オーバーフロービットＯＶは、演算を制御する制御回路３１にも供給されている。制御回路３１は、ＯＶ＝１又はＯＶ＝０に応じて、例えば減算器３で減算を行うか否かを制御する。減算を行う場合、その減算の完了後にセレクタ４の出力を後段にラッチするように制御する。

【００４２】

従って図１の構成では図２の構成に比べて減算を実行する頻度が低減でき、平均的な演算時間の短縮を図ることができる。

【００４３】

ここで、前記〔アルゴリズム４〕で与えられる最終演算結果ｔはＮよりも大きくなることがある。即ち、Ｒ（$=2^n$）＞Ｎであるから、ステップ３の処理を経て前記オーバーフロー補正で得られるｔの値がＲよりも小さく且つＮよりも大きければ、そのｔの値が、〔アルゴリズム４〕で与えられる演算結果ｔとされる。前記〔アルゴリズム２〕の場合には直接Ｎと比較するため、最終的に得られる演算結果ｔは必ずＮよりも小さくされる。そのため、〔アルゴリズム４〕全体の演算式を「$A\cdot B\cdot R^{-1} \mod N + kN$」で表わし、補正項ｋＮによって前記誤差を解消することが必要になる。ここでｋ は０又は正の整数である。

【００４４】

前記剰余演算式「$A\cdot B\cdot R^{-1} \mod N + kN$」は、その値Ａの入力値（$A_{in}$）と出力値（$A_{out}$）を区別して記述すれば、演算式「$A_{out} = A_{in}\cdot B\cdot R^{-1} \mod N + kN$」 となる。これを、一般式化すれば、「$A_{out} = f(A_{in}) \mod N + kN$」と記述することができ、入力値 A_{in} と出力値 A_{out} の値の範囲はＮよりも大きな2^n 未満とされ、ｎは演算値のビット数を表わす正の整数、Ｎは $0<N<2^n$ となるｎビットの正の整数、A_{in}は$0\leqq A_{in}<2^n$ となるｎビットの正の整数、A_{out}は$0\leqq A_{out}<2^n$ となるｎビットの正の整数、ｋは０又は正の整数とされる。前記Ｒは2^n、R^{-1}はＲ・$R^{-1} \mod N = 1$ を満たす $0<R^{-1}\leqq N$ となるｎビットの正の整数、Ｂは$0\leqq B<2^n$ となるｎビットの正の整数である。

【００４５】

前記〔アルゴリズム４〕を使用した場合、その補正項に伴ってべき乗剰余演算「$X^Y \mod N$」を実行する〔アルゴリズム３〕を修正すると、以下に示す〔アルゴリズム５〕を得る。

【００４６】

〔アルゴリズム５〕

input X, Y=e_n e_{n-1} ･･･e_1, N, R　　ステップ1
B = R^2 mod N　　ステップ2
A = X　　ステップ3
A = A･B･R^{-1} mod N + kN　　ステップ4
B = A　　ステップ5
for i = n-1 to 1 step -1 {　　ステップ6
　　A = A^2･R^{-1} mod N + kN　　ステップ7
　　if e_i=1 then A = A･B･R^{-1} mod N + kN　　ステップ8
　　}　　ステップ9
A = A･R^{-1} mod N + kN　　ステップ10
A = A mod N　　ステップ11
output A　　ステップ12

前記〔アルゴリズム３〕から〔アルゴリズム５〕への変更点は、ステップ４，７，８，１０のそれぞれの剰余乗算の演算式を変更した他は、ステップ１１　を追加しただけである。ステップ１１は演算結果をＮよりも小さくするために追加されている。前記〔アルゴリズム５〕の演算は汎用マイクロプロセッサによって実行することも可能である。本実施例においては、専用ハードウェアとしてのコプロセッサを用いる。

【００４７】

図３には、べき乗剰余演算「X^Y　ｍｏｄ　Ｎ　」を行うための〔アルゴリズム５〕を実行可能なマイクロコンピュータＭＣＵの一例ブロック図が示されている。

【００４８】

図３において、５はマイクロプロセッサ、６はコプロセッサ、７はクロック発生器、８はマイクロプロセッサ５が実行すべきプログラムや固定データが格納されたＲＯＭ（読み出し専用メモリ又はリードオンリメモリ）、９はマイクロプロセッサ５の作業領域を提供するＲＡＭ（随時読み出し書き込み可能なメモリ又はランダムアクセスメモリ）、１０は前記べき乗剰余演算「X^Y　ｍｏｄ　Ｎ　」を適用して符号化された情報等の保持に利用される電気的に書き換え可能なＥＥＰＲＯＭやフラッシュメモリなどの不揮発性メモリ（単にＥＥＰＲＯＭと称する）、１１は入出力ポート（単にＩ／Ｏポートと称する）である。マイクロプロセッサ５、コプロセッサ６、ＲＡＭ８、ＲＯＭ９、ＥＥＰＲＯＭ１０及びＩ／Ｏポート１１は代表的に示されたバス１２、１３に共通接続されている。バス１２はアドレスバスとコントロールバスを総称する。１３はデータバスを意味する。クロック発生器７はクロック端子ＣＬＫから供給されるクロック信号に基づいて内部の動作基準クロック信号を生成してマイクロプロセッサ５及びコプロセッサ６に供給する。Ｉ／Ｏポート１１はデータ入出力外部端子Ｉ／Ｏに結合される。Ｖｃｃ，Ｖｓｓはマイクロコンピュータ ＭＣＵの電源用外部端子、ＲＥＳはマイクロコンピュータＭＣＵのリセット用外部端子である。

【００４９】

マイクロコンピュータＭＣＵは、特に制限されないが、図３に示されている全ての機能ブロックが単結晶シリコン基板のような一個の半導体基板に形成されて成る。

【００５０】

前記コプロセッサ６は、上記〔アルゴリズム５〕のステップ４、ステップ７、ステップ８、ステップ１０で示された「Ａ　＝　Ａ・Ｂ・R^{-1}　ｍｏｄ　Ｎ　＋　ｋＮ」等で記述される「剰余乗算」を実行するための専用ハードウェアとして位置付けられ、演算回路３

２と制御回路３１を含んでいる。前記図１のオーバーフロー補正回路はコプロセッサ６に含まれる。剰余乗算の入力値Ａ，Ｂ，Ｒ，Ｎ及び出力値Ａはコプロセッサ６の中のレジスタ又はＲＡＭなどの記憶装置に保持される。あるいは、前記ＲＡＭ９に保持させるようにしてもよい。

【００５１】

マイクロプロセッサ５は、公知演算命令や条件分岐命令等を実行できる汎用マイクロプロセッサであり、コプロセッサ６が実現する演算機能を使用しながら、〔アルゴリズム５〕による、べき乗剰余演算「 X^Y ｍｏｄ Ｎ」 を実現する。

【００５２】

図４には〔アルゴリズム５〕によってべき乗剰余演算「 X^Y ｍｏｄ Ｎ」 を実行するときのマイクロプロセッサ５とコプロセッサ６による処理のフローチャートが概略的に示されている。図４においてＴ１及びＴ５は〔アルゴリズム５〕においてマクロプロセッサ５が負担すべき演算処理とされ、Ｃ２は〔アルゴリズム５〕においてコプロセッサ６が負担すべき演算処理とされる。Ｃ２で行われる演算処理は、特に制限されないが、〔アルゴリズム５〕のステップ４，７，８，１０の処理とされる。マイクロプロセッサ５は、コプロセッサ６に演算を実行させるとき、それに必要な値（Ａ，Ｂ，Ｎなど）を例えばコプロセッサ６内部のレジスタ等に設定する（Ｔ２）。その後マイクロプロセッサ５はコプロセッサ６に対して演算開始をコマンドなどによって指示する（Ｔ３）。コプロセッサ６はＣ１によって設定された入力値を用い、コマンドなどによって指示された演算処理を行う（Ｃ２）。その間、マイクロプロセッサ５はコプロセッサ６による演算の終了を待ち、或いはその他の処理を行うことができる（Ｔ４）。コプロセッサ６は、Ｃ２による演算を終了すると、それをマイクロプロセッサに通知する（Ｃ３）。マイクロプロセッサ５はコプロセッサ６によるＣ２の演算結果を用いてアルゴリズムの処理を行う（Ｔ５）。更にコプロセッサ６に負担させるべき処理が残っている場合にはＴ６を経て上記処理が繰り返される。

【００５３】

（２）第２の実施の形態

上記〔アルゴリズム５〕において、ステップ７で示された剰余乗算「 $A^2 \cdot R^{-1}$ ｍｏｄ Ｎ ＋ ｋＮ」は、ステップ８で示された剰余乗算「Ａ・Ｂ・R^{-1} ｍｏｄ Ｎ ＋ ｋＮ」においてＢ＝Ａ とすることにより実現できる。しかしながら、実際に、Ａに割り当てられたレジスタの値をＢに割り当てられたレジスタへ転送していたのでは、演算の実行時間が増大してしまうという問題を生じる。またステップ８をｆｏｒ文のループの中で繰り返し実行する場合、同一のＢの値を繰り返し使用できるため、ステップ７でＢの値をＡの値で書き換えない方がより効率的な演算の実行が可能となる。さらにステップ１０で示された剰余乗算「Ａ・R^{-1} ｍｏｄ Ｎ ＋ ｋＮ」でも、同様にＢ＝１ とすることにより実現できるが、Ｂ に１ をセットするために演算の実行時間が増大してしまうという問題が生じる。これらは前記第５の問題点に対応される内容である。

【００５４】

この問題点を解決するため、剰余乗算「Ａ・Ｂ・R^{-1} ｍｏｄ Ｎ ＋ ｋＮ」を実現する回路には、Ｂの値の代わりにＡの値又は”１” を選択できるようにする構成を採用するものである。

【００５５】

図５にはその課題を解決するための演算手段のブロック図が示される。特に制限されないが、図５に示されている演算手段は、図３のコプロセッサ６が実現する一つの回路構成として位置付けることができ、単結晶シリコン基板のような一個の半導体チップに形成されたマイクロコンピュータＭＣＵに含まれている。

【００５６】

図５において１５は演算器、１６はセレクタ、１７はコントロールレジスタ、１８～２０は演算用のレジスタである。セレクタ１６はコントロールレジスタ１７の制御ビットＣＢ０，ＣＢ１の値に従ってレジスタ１８，１９又は固定値”１”を選択して演算器１３に与

える。コントロールレジスタ１７の２ビットＣＢ０，ＣＢ１はその論理値の組合せに応じて演算式を指定する。即ち、その設定値に応じて、剰余乗算「Ａ・Ｂ・R^{-1} ｍｏｄ Ｎ ＋ ｋＮ」におけるＢの値として、レジスタ１８の値Ａ， レジスタ１９の値Ｂ、又は固定値”１” の何れかをセレクタ１６を介して演算器１５に供給するようになっている。コントロールレジスタ１７のビットＣＢ０，ＣＢ１に従ったセレクタの選択動作に従って、演算器１５は、「Ａ・Ｂ・R^{-1} ｍｏｄ Ｎ ＋ ｋＮ」、「A^2・R^{-1} ｍｏｄ Ｎ ＋ ｋＮ」又は「Ａ・R^{-1} ｍｏｄ Ｎ ＋ ｋＮ」を同じように演算することができる。したがって、値Ｂを保有するためのレジスタに値Ａを転送したり、値Ｂを保有するためのレジスタに１を設定する処理を要しない。コントロールレジスタ１５への値の設定は、例えば図３に示されるマイクロプロセッサ５によりバス１２，１３を経由して行われる。

【００５７】

図６には図５の比較例としての演算手段のブロック図が示されている。図６の場合には、剰余乗算「A^2・R^{-1} ｍｏｄ Ｎ ＋ ｋＮ」を実行する場合には、バスを経由してＡ の値をＢ へ転送したり、Ｂの値を１に設定する処理を行わなければならない。

【００５８】

（３）第３の実施の形態

上記〔アルゴリズム５〕において、ステップ８に代表される剰余乗算「Ａ（A_{out}）＝ Ａ（A_{in}）・Ｂ・R^{-1} ｍｏｄ Ｎ ＋ ｋＮ」では、演算結果（A_{out}）を入力値Ａ（A_{in}）と同じ記憶装置もしくはレジスタに保存しなければならない。剰余乗算は通常瞬時には実行できず、多くのクロック数（演算ステップ数）を費やして入力値Ａ，Ｂ，Ｎを何度も参照しながら計算を行うため、計算の途中結果および最終結果を入力値Ａと同じ記憶装置に保存保存することはできず、一次記憶値（Ｔｅｍｐ）用の別の一次記憶装置に保存する必要が生じる。そのため、演算式「Ａ＝ Ａ・Ｂ・R^{-1} ｍｏｄ Ｎ ＋ ｋＮ」は、「Ｔｅｍｐ ← Ａ・Ｂ・R^{-1} ｍｏｄ Ｎ ＋ ｋＮ」となり、ステップ８を実現するためには最後に「Ａ ← Ｔｅｍｐ」を実行しなければならず、計算時間が増大してしまうという問題が本発明者によって見い出されている。これは前記第６の問題点に対応される。この問題点は〔アルゴリズム５〕のステップ４、７、１０でも同様に生じる。

【００５９】

図７には上記問題点の解決に着目した演算手段のブロック図が示される。特に制限されないが、図７に示されている演算手段は図３のコプロセッサ６が実現する一つの回路構成として位置付けることができ、単結晶シリコン基板のような一個の半導体チップに形成されたマイクロコンピュータＭＣＵに含まれる。

【００６０】

図７において、２２は演算器、２３と２４は前記値Ａを格納するレジスタと一次記憶値Ｔｅｍｐ用のテンポラリレジスタに交互に切換え利用されるレジスタ、３５は値Ｂを格納する為に利用されるレジスタ、２６は値Ｎを格納するために利用されるレジスタである。レジスタ２３～２６と演算器２２はコプロセッサ内部のバス４０に共通接続されている。

【００６１】

図７においてＳ１はレジスタ２３の選択信号、Ｓ２はレジスタ２４の選択信号である。Ｒｔｍｐは一次記憶値Ｔｅｍｐに対するアクセス信号、Ｒａは値Ａに対するアクセス信号である。アクセス信号Ｒｔｍｐ，Ｒａは、マイクロプロセッサがコプロセッサに与えたコマンドに応じて制御回路３１が出力する演算制御信号の一種とされる。

【００６２】

２９、３０はセレクタである。セレクタ２９、３０は、フラグ２７の値に従ってアクセス信号Ｒａ又はＲｔｍｐを選択し、選択した信号をレジスタ選択信号Ｓ１，Ｓ２として出力する。フラグ２７が”１”のとき、セレクタ２９はアクセス信号Ｒａをレジスタ選択信号Ｓ１として選択し、セレクタ３０はアクセス信号Ｒｔｍｐをレジスタ選択信号Ｓ２として選択する。フラグ２７が”０”の時はセレクタ２９，３０による選択状態はその逆にされ

る。

【００６３】

前記フラグ２７の設定はフラグ反転制御ロジック２８によって行われる。このフラグ反転制御ロジック２８は、例えば、前記制御回路３１に含まれ、演算の手順を制御するシーケンサが剰余乗算「Ａ・Ｂ・Ｒ$^{-1}$　ｍｏｄ　Ｎ」の終了タイミングを示したとき、それに同期してフラグ２７の状態を反転制御する。

【００６４】

このように、前記レジスタ２３と２４は剰余乗算「Ａ・Ｂ・Ｒ$^{-1}$　ｍｏｄ　Ｎ」の度に交互に値Ａ又は一次記憶値Ｔｅｍｐ　の格納用に割り当てられる。例えば、レジスタ２３が値Ａの格納に使用されている時、レジスタ２４は一次記憶値Ｔｅｍｐの格納に使用され、剰余乗算「Ａ・Ｂ・Ｒ$^{-1}$　ｍｏｄ　Ｎ」の実行終了時に、アクセス信号Ｒａ，Ｒｔｍｐに対する選択信号Ｓ１，Ｓ２の対応が切り換えられるように制御される。このレジスタ機能の交換は、図示されるように、１ビットのフラグ２７を剰余乗算「Ａ・Ｂ・Ｒ$^{-1}$　ｍｏｄ　Ｎ」の実行終了時に反転させることにより制御される。これにより、値Ａ，　Ｔｅｍｐ　の内容の入れ換えを見かけ上瞬時に実行することができ、演算時間の短縮に寄与する。

【００６５】

図８は図７の演算手段に対する比較例を示すものである。図８に示される演算手段において、剰余乗算「Ａ・Ｂ・Ｒ$^{-1}$　ｍｏｄ　Ｎ」の実行結果は、まず一時記憶値Ｔｅｍｐのために固定的に割り当てているテンポラリレジスタに格納され、そのテンポラリレジスタの値Ｔｅｍｐが、値Ａ専用のレジスタに転送されて、値Ａのレジスタが最終結果を保持する。そのようなデータ転送には少なからず時間を要する。

【００６６】

図９には図７のレジスタ２３～２６をＲＡＭ３２で実現する場合のコプロセッサの例が示される。

【００６７】

特に制限されないが、図９に示されているコプロセッサは、図３のマイクロコンピュータＭＣＵのコプロセッサ６に適用される一つの回路構成として位置付けることができ、前記同様に、単結晶シリコン基板のような一個の半導体チップに形成されたマイクロコンピュータＭＣＵに含まれている。

【００６８】

図９において５０はＲＡＭであり、同図には記憶領域２３Ｍ～２６Ｍが代表的に示されている。記憶領域２３Ｍ，２４Ｍは交互に前記値Ａを格納する領域と演算の途中結果である一次記憶値Ｔｅｍｐを格納する領域に切換え利用される。記憶領域２５Ｍは値Ｂを格納する為に利用され、記憶領域２６Ｍは値Ｎを格納するために利用される。各記憶領域２３Ｍ～２６Ｍは演算回路内部のバス２７を介して、代表的に示された演算器２２に接続されている。ＲＡＭ５０に対するアクセス制御はバス５２、５３を介して行われる。アドレス信号としてＡｍｓｂ～Ａ０の所定ビットのアドレス信号によってＲＡＭ５０のアドレシングが行われる。

【００６９】

コントロールレジスタ５１にはマイクロプロセッサ５から供給されたコマンドが設定され、制御回路３１がそれを解読して、演算器２２などに対する制御信号を生成する。

【００７０】

前記記憶領域２３Ｍ～２６Ｍに対する選択情報は、特に制限されないが、アドレス信号の最上位側の２ビットＡｍｓｂ，Ａｍｓｂ－１とされる。ここで、Ａｍｓｂ＝１のとき、フラグ２７の論理値に応じてＡｍｓｂ－１の論理値を強制的に反転させるためにアンドゲート５４及び排他的論理和ゲート５５が設けられ、そのゲート５５の出力Ａ’ｍｓａ－１がＡｍｓａ－１に代えてＲＡＭ５０に供給される。Ａｍｓｂ＝１，Ａ’ｍｓｂ－１＝０で記憶領域２３Ｍ、Ａｍｓｂ＝１，Ａ’ｍｓｂ－１＝１で記憶領域２４Ｍ、Ａｍｓｂ＝０，Ａ’ｍｓｂ－１＝０で記憶領域２５Ｍ、Ａｍｓｂ＝０，Ａ’ｍｓｂ－１＝１で記憶領域２６

Ｍを選択する。コプロセッサの動作プログラム上において、Ａｍｓｂ＝１，Ａｍｓｂ－１＝０は値Ａの格納領域を指し、Ａｍｓｂ＝１，Ａｍｓｂ－１＝１はｔｅｍｐの格納領域を指す。

【００７１】

前記フラグ２７の設定はフラグ反転制御ロジック２８によって行われる。このフラグ反転制御ロジック２８は、例えば、演算の手順を制御する制御回路３１内部のシーケンサが剰余乗算「Ａ・Ｂ・R^{-1}　ｍｏｄ　Ｎ」の終了タイミングを示したとき、それに同期してフラグ２７の状態を反転制御する。

【００７２】

これにより、前記記憶領域２３Ｍ，２４Ｍはそれぞれ排他的に値Ａの格納、又は一次記憶値Ｔｅｍｐの格納に使用される。例えば、フラグ２７が０のとき、記憶領域２３Ｍが値Ａの格納に、記憶領域２４ＭがＴｅｍｐの格納に使用される。剰余乗算の実行終了時にフラグ２７の値が１に反転されると、次の演算では、記憶領域２３Ｍが値ｔｅｍｐの格納に、記憶領域２４Ｍが値Ａの格納に使用される。これにより、レジスタに代えてＲＡＭ５０を用いる場合にも、値Ａ，　Ｔｅｍｐ　の内容の入れ換えを見かけ上瞬時に実行することができる。

【００７３】

（４）第４の実施の形態

上記〔アルゴリズム５〕において、ステップ８に代表される剰余乗算「Ａ＝　Ａ・Ｂ・R^{-1}　ｍｏｄ　Ｎ　＋　ｋＮ」を実行するためのアルゴリズムは、前記〔アルゴリズム４〕で記述されるが、このアルゴリズムにおいて次の点を考慮する必要がある。

【００７４】

前記〔アルゴリズム５〕のステップ１、ステップ２でＮ’，　Ｍ　を予め用意しなければならない。また大きな値を採るＭ　は演算時点まで保持されなければならない。また、前記〔アルゴリズム５〕のステップ３においては、中間結果ＡＢは、値Ａ，Ｂのデータ長の合計をデータ長とする大きな数となるため、これらを一時的に保持するにはそれに応じた記憶装置が必要になる。これらは、前記第１の問題点に対応される。

【００７５】

また、前記〔アルゴリズム５〕のステップ３ではＡ，Ｂ，Ｍ，Ｎの大きな数どうしの乗算が必要であり、これをいかにして効率良く実現するかという課題があった。特にこれをハードウェアで実現する場合、Ａ，Ｂ，Ｎ　の値を保持する記憶装置と、乗算を実行する演算器との間でデータのやりとりを頻繁に行なわなければならず、記憶装置やデータパスの論理回路規模を最小限に抑えるための考慮が必要である。これは前記第３の問題点に対応される。

【００７６】

ここでは、それらの問題点を解決するための手段として、先ず、前記Ａ，Ｂ，Ｍ，Ｎの値の大きな数どうしの乗算を効率良く実現するための原理について説明する。

【００７７】

図１０には値Ｂ，　Ｍ　のブロック分割の例が示されている。図１０においては、ｎビット長のＢ，Ｍを各々Ｌビット長のブロックに分割した状態が図示されている。これにより、Ｂ，Ｍは各々ｎ／Ｌ　個のブロックに分割される。例えばｎは５１２ビット、Ｌは３２ビットのような値とされる。

【００７８】

このブロック分割を用いて、〔アルゴリズム４〕のステップ３で記述された演算式「（Ａ・Ｂ＋Ｍ・Ｎ）／Ｒ」に、部分積に分解する変形を施すと、以下のようになる。下記に示された式の変形結果は専ら部分積の項の形式が把握できる範囲で記載を省略してある。

【００７９】

$$(A\cdot B+M\cdot N)/R = (A\cdot(2^{n-L}\cdot B_{n/L-1} +\cdots+ 2^{L}\cdot B_1 + B_0)+(2^{n-L}\cdot M_{n/L-1} + \cdots + 2^{L}\cdot M_1 + M_0)\cdot N)/R$$

$$= (\cdots((A\cdot B_0+M_0\cdot N)/2^{L} + (A\cdot B_1+M_1\cdot N))/2^{L} +\cdots+ (A\cdot B_{n/L-1}+M_{n/L-1}\cdot N))/2^{L}$$

この変形式より明らかなように、演算「（Ａ・Ｂ$_i$＋Ｍ$_i$・Ｎ）／２^{L}」をｎ／Ｌ回　繰り返して実行すれば、「（Ａ・Ｂ＋Ｍ・Ｎ）／Ｒ」を実現できることになる。

【００８０】

図１１には前記部分積による演算「（Ａ・Ｂ$_i$＋Ｍ$_i$・Ｎ）／２^{L}」を実行する為の演算手段に着目したコプロセッサの更に別の例が示されている。特に制限されないが、図１１に示されているコプロセッサは、図３のコプロセッサ６に適用される一つの回路構成として位置付けることができ、前記同様に、単結晶シリコン基板のような一個の半導体チップに形成されたマイクロコンピュータＭＣＵに含まれる。

【００８１】

図１１において、３３は第１の積和演算器、３４は第２の積和演算器、３５は一次記憶値Ｔｅｍｐを保持するテンポラリレジスタ、３６は値Ａの格納に利用れるレジスタ、３７は値Ｂの格納に利用されるレジスタ、３８は値Ｎの格納に利用されるレジスタである。３９はＭｉ生成ロジック、４０はＭｉ生成ロジック３９で生成された値Ｍ$_i$を保持するラッチ、４１は「÷２^{L}」を行うためのシフト回路である。

【００８２】

図１１に示される回路は図９に示される態様のブロック分割に基づいて演算「（Ａ・Ｂ$_i$＋Ｍ$_i$・Ｎ）／２^{L}」を実行する回路である。先ず、第１の積和演算器３３は、レジスタ３５の値Ｔｅｍｐ、レジスタ３６の値Ａ、レジスタ３７の値Ｂ$_i$　を入力として、積和演算「Ｔｅｍｐ　＋　Ａ・Ｂ$_i$」を実行する。その演算結果は値Ｔｅｍｐ２　として次段の第２の積和演算器３４へ送られる。値Ｔｅｍｐ２　はｎ＋Ｌビット長の整数である。一方、Ｍｉ生成ロジック３９は、Ｌビット長の数Ａ$_0$，　Ｂ$_i$，　Ｎ$_0$　を入力としてＬビットの整数Ｍ$_i$を生成し、この正数Ｍ$_i$はレジスタ４０に一時的に保持される。第２の積和演算器３４は、前記Ｔｅｍｐ２，　Ｎ，　Ｍ$_i$　を入力として、積和演算「Ｔｅｍｐ２　＋　Ｍ$_i$・Ｎ」を実行する。ｎ＋Ｌ　ビット長の演算結果の下位Ｌビットは全て０であり、これをシフタ４１によって消去して（すなわち２^{L}で割って）、ｎビット長の結果が値Ｔｅｍｐ　としてレジスタ３５に送られ保持される。

【００８３】

以上の動作をｎ／Ｌ回　繰り返し実行すれば、演算「（Ａ・Ｂ＋Ｍ・Ｎ）／Ｒ」が実現できる。これによれば、ｎビットの整数Ｍをあらかじめ計算して保持する必要はなく、Ｌビット長のＭ$_i$のみを積和演算器３３の計算中に求めてレジスタ４０に保持すればよく、値Ｍの計算時間の削除、および値Ｍを保持する記憶手段の規模を縮小することができる。さらに、積和演算器３３と積和演算器３４を直列的に接続して連続的に動作させることにより、ｎ＋Ｌビット長の中間結果Ｔｅｍｐ２　を一時的に保持する記憶手段を特別に設けることも必要なくなる。

【００８４】

図１２には前記部分積による演算「（Ａ・Ｂ$_i$＋Ｍ$_i$・Ｎ）／２^{L}」を実行する為の演算部に着目したコプロセッサの更に別の例が示される。特に制限されないが、図１２に示されているコプロセッサは、図３のコプロセッサ６に適用される一つの回路構成として位置付けることができ、前記同様に、単結晶シリコン基板のような一個の半導体チップに形成されているマイクロコンピュータＭＣＵに含まれる。

【００８５】

図１２の例は、図１１のコプロセッサに対し、レジスタ３５～３８を積和演算器３３、３４にバス４０で接続した点が相違される。したがって前記レジスタ３５～３８をＲＡＭ４２で構成することができるようになる。これにより、半導体チップ上のレジスタ面積の低減が可能となる。また、この構成においては、特にバス４０によるデータ転送量が多いた

め、バス幅が大きくなって半導体チップの面積が大きくならないようにする必要が生じるが、図１１の例で示したように積和演算器３３と積和演算器３４を直列的に接続することにより、中間結果Ｔｅｍｐ２　をバスを用いて転送することが不要になるため、バスによるデータ転送量の低減を図ることができる。

【００８６】

（５）第５の実施の形態

図１１又は図１２のコプロセッサの例において、第１の積和演算器３３でＴｅｍｐ＝０、第２の積和演算器３４でM_i・Ｎ＝０、さらにセレクタ４１による「$\div 2^L$」の動作を行なわないことにより、同図に示される演算手段を、「Ａ・B_i」のような多倍長乗算（小さな数B_iとその多倍長に相当する大きな数Ａとの乗算）を実行する回路として使用することができる。これは、前記第４の問題点を解決するための一つの手段とされる。「Ａ・B_i」のような多倍長乗算演算は、例えば〔アルゴリズム５〕のステップ２の演算「R^2　ｍｏｄ　Ｎ」をマイクロプロセッサを用いて実行するときに適用されることにより、その演算の高速化を図ることができる。

【００８７】

即ち、図１３には「R^2　ｍｏｄ　Ｎ」の計算の概念図が示されている。図１３において、$R=2^n$、ｎ＝５１２とされ、Ｎは５１２ビット、R^2は最上位ビットだけが１で下位側１０２４ビット全てが０の値とされる。マイクロプロセッサで演算「R^2　ｍｏｄ　Ｎ」を行うとき、大きな数のR^2を同様に大きな数のＮで直接に除算するのは効率的でないから、被除数を最上位側から６４ビット単位のブロックとして把握し、また、除数を最上位側から３２ビット単位のブロックとして把握し、順次上位側のブロック同士を対象に除算を行い、それによって得られる値を商の概数として把握する。図１３において例えばＱ（＝Ｄａ÷Ｎａ）を商の概数として把握する。概略的には、R^2の上位側に対して「Ｑ・Ｎａ」を減算し、その減算結果の上位側に対して「Ｑ・Ｎｂ」を減算する。「Ｑ・Ｎｂ」の減算結果に対して同様の処理を行い、更にその結果の対して同様の処理を繰り返すという手法によって、「R^2　ｍｏｄ　Ｎ」の結果を得ることができる。実際にはその途上で、余剰ビットを消去するための減算処理が介在される。このとき、図１３の例に従えば、前記演算「Ｑ・Ｎｂ」の処理は、第１回目では３２ビットと４８０ビットという大きな数の乗算処理とされる。しかもそのような大きな数の乗算処理は何回も繰り返される。このとき、図１１や図１２に示されるコプロセッサによって演算可能な前記「Ａ・B_i」のような多倍長乗算演算を利用することにより、換言すれば、そのような多倍長乗算演算をコプロセッサに負担させれば、〔アルゴリズム５〕におけるステップ２の演算「R^2　ｍｏｄ　Ｎ」をマイクロプロセッサを用いて実行するとき、その演算処理の高速化を図ることができる。

【００８８】

以上本発明者によって成された発明を具体的に説明したが、本発明はそれに限定されるものではなく、その要旨を逸脱しない範囲において種々変更可能であることは言うまでもない。図１、５、７、９、１１、１２に基づいて説明した内容は、それぞれに対応される課題の解決手段を理解し易くするために、コプロセッサの別々の回路構成であるかのように説明してきたが、それらの構成は全てを又は選択的に複数種類の構成を一つのコプロセッサで実現できることは言うまでもない。

【００８９】

また、値Ａ，Ｂ，Ｎのビット数は５１２ビットに限定されず、それ以上のビット数を利用可能であることは言うまでもない。また、べき乗剰余演算を実現するためのハードウェア構成は上記の各種実施の形態に限定されず適宜変更可能である。

【００９０】

また、上記説明した演算技術は、特に図示はしないが、前記アルゴリズム２を採用しないところのアルゴリズム１に従ったべき乗剰余演算を行うコプロセッサやマイクロコンピュータ等のデータ処理装置にも適用することができる。

【００９１】

さらに、上記説明では補正項ｋＮを用いる場合を一例としたが、前記直列的に配置された積和演算器３３、３４を利用した処理は、前記アルゴリズム３に基づいてべき乗剰余演算を行う以下のデータ処理装置にも適用できる。すなわち、ｎを演算値のビット数を表わす正の整数、Ｎを　０＜Ｎ＜2^{n}　となるｎビットの正の整数、A_{in}を$0 \leqq A_{in} < 2^{n}$　となるｎビットの正の整数、A_{out}を$0 \leqq A_{out} < 2^{n}$　となるｎビットの正の整数とするとき、演算式が$A_{out} = f(A_{in})$ ｍｏｄ Ｎで与えられ、入力値　A_{in}　と出力値　A_{out}　の値の範囲をＮよりも大きな2^{n}　未満とする剰余演算を行う演算手段と、この演算手段による上記剰余演算を制御する制御手段とを１個の半導体基板に備えて成るデータ処理装置である。前記$A_{out} = f(A_{in})$ ｍｏｄ Ｎで与えられる剰余演算は、$A_{out} = A_{in} \cdot B \cdot R^{-1}$ ｍｏｄ Ｎの演算式で与えられる剰余演算を含み、Ｒは2^{n}、R^{-1}はＲ・R^{-1} ｍｏｄ Ｎ ＝ １ を満たす　０＜R^{-1}　≦Ｎ　となるｎビットの正の整数、Ｂは$0 \leqq B < 2^{n}$　となるｎビットの正の整数であり、前記演算手段は、演算式$A_{out} = A_{in} \cdot B \cdot R^{-1}$ ｍｏｄ Ｎ、で与えられる剰余演算を実行するための、直列的に接続された積和演算器３３、３４を備える。これに対する具体的な演算態様として、前記演算式$A_{out} = A_{in} \cdot B \cdot R^{-1}$ ｍｏｄ Ｎ で与えられる剰余演算は、ｔ ＝ （A_{in}・Ｂ＋Ｍ・Ｎ）／Ｒで与えられる演算処理と、これに続いてｉｆ ｔ≧Ｒ ｔｈｅｎ ｒｅｔｕｒｎ ｔ－Ｎ ｅｌｓｅ ｒｅｔｕｒｎ ｔで与えられｔ－Ｎ又はｔを前記A_{out}とする演算処理を含み、前記演算手段は、直列的に接続された前記積和演算器３３、３４により、部分積の和として、前記演算処理　ｔ＝（A_{in}・Ｂ＋Ｍ・Ｎ）／Ｒを実行する。これによっても同じく、部分積の一次記憶用メモリ手段などを必要とせず、演算手段の回路規模を縮小できる。

【００９２】

本発明はコプロセッサやマイクロコンピュータの他に、ＩＣカード、符号化・復号化装置若しくは暗号化・復号化装置などに広く適用することができる。

【００９３】

【発明の効果】

本願において開示される発明のうち代表的なものによって得られる効果を簡単に説明すれば、下記の通りである。

【００９４】

すなわち、高速にべき乗剰余演算「　X^{Y}　ｍｏｄ　Ｎ　」を実現することができる。

【００９５】

また、べき乗剰余演算に為の上記の専用ハードウェアの実現において、その論理回路規模を最小限にすることができる。

【００９６】

更に、上記専用ハードウェアをＩＣカード用マイクロコンピュータと同一の半導体チップに搭載し、べき乗剰余演算「　X^{Y}　ｍｏｄ　Ｎ」を適用した符号化・復号化のためのマイクロコンピュータを低コストで使い易く実現することができる。

【００９７】

そして、マイクロプロセッサを搭載するマイクロコンピュータにおいて、剰余乗算　Ａ＝　$A \cdot B \cdot R^{-1}$ ｍｏｄ Ｎ ＋ｋＮ、Ａ ＝ $A^{2} \cdot R^{-1}$ ｍｏｄ Ｎ ＋ｋＮ、Ａ ＝ $A \cdot R^{-1}$ ｍｏｄ Ｎ ＋ｋＮ で与えられる演算を、積和演算器を内蔵したコプロセッサで実行し、さらに同一のコプロセッサに前処理　R^{2}　ｍｏｄ Ｎ を高速に実行するための乗算機能を備え、これらのコプロセッサの演算機能を用いて、マイクロプロセッサでべき乗剰余演算　X^{Y}　ｍｏｄ Ｎ　を高速に実行することができる。

【図面の簡単な説明】

【図１】オーバフロー補正回路の一例を示すブロック図である。

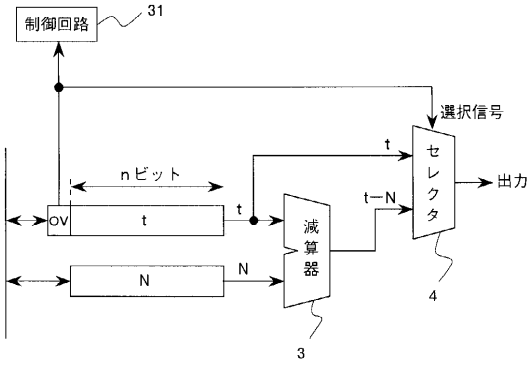
【図２】オーバフロー補正回路に対する比較例のブロック図である。

【図３】べき乗剰余演算「X^{Y}　ｍｏｄ　Ｎ」を行うためのアルゴリズム５を実行可能なマイクロコンピュータの一例ブロック図である。

【図４】３図のマイクロコンピュータにおいてアルゴリズム５に従ってべき乗剰余演算「X^{Y} ｍｏｄ Ｎ」を行うときのマイクロプロセッサとコプロセッサの概略的な処理の流れを示すフローチャートである。
【図５】複数種類の剰余乗算「Ａ・Ｂ・R^{-1} ｍｏｄ Ｎ ＋ ｋＮ」、「A^{2}・R^{-1} ｍｏｄ Ｎ ＋ ｋＮ」又は「Ａ・R^{-1} ｍｏｄ Ｎ ＋ ｋＮ」を選択的に実行できるコプロセッサの演算手段を示すブロック図である。
【図６】図５の演算手段に対する比較例を示すブロック図である。
【図７】Ａ＝Ａ・Ｂ・R^{-1} ｍｏｄ Ｎ ＋ ｋＮなどの演算において入力レジスタとテンポラリレジスタとのレジスタ機能交換を実現する演算手段の一例を示すブロック図である。
【図８】図７の演算手段に対する比較例を示すブロック図である。
【図９】図７のレジスタをＲＡＭに置き換えてレジスタ機能交換を実現する演算手段の一例を示すブロック図である。
【図１０】部分積のために値Ｂ，Ｍをブロック分割した例を示す説明図である。
【図１１】部分積により（Ａ・B_i ＋M_i ・Ｎ）／2^{L} を実行するための演算手段を含むコプロセッサのブロック図である。
【図１２】部分積により（Ａ・B_i ＋M_i ・Ｎ）／2^{L} を実行するための別の演算手段を含むコプロセッサのブロック図である。
【図１３】R^{2} ｍｏｄ Ｎの計算の概念を示す説明図である。
【図１４】べき乗剰余演算を適用する符号化、復号化装置の概略説明図である。
【符号の説明】
３ 演算器
４ セレクタ
ＭＣＵ マイクロコンピュータ
５ マイクロプロセッサ
６ コプロセッサ
１５ 演算器
１６ セレクタ
１７ コントロールレジスタ
ＣＢ，ＣＢ２ 制御ビット
１８～２０ レジスタ
２２ 演算器
２３～２６ レジスタ
２７ フラグ
２８ フラグ反転制御ロジック
２９，３０ セレクタ
Ｒａ，Ｒｔｍｐ アクセス信号
Ｓ１，Ｓ２ レジスタ選択信号
３１ 制御回路
５０ ＲＡＭ
２３Ｍ～２６Ｍ 記憶領域
５１ コントロールレジスタ
３３ 第１の積和演算器
３４ 第２の積和演算器
３５～３８ レジスタ
３９ Ｍｉ生成ロジック
４０ レジスタ
４１ シフタ
４２ ＲＡＭ

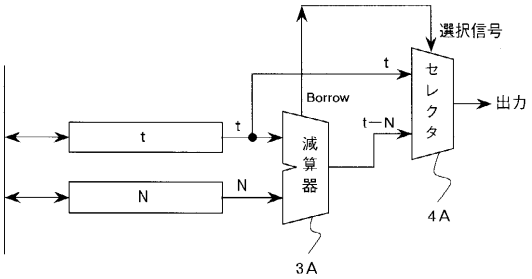
【図１】

【図1】



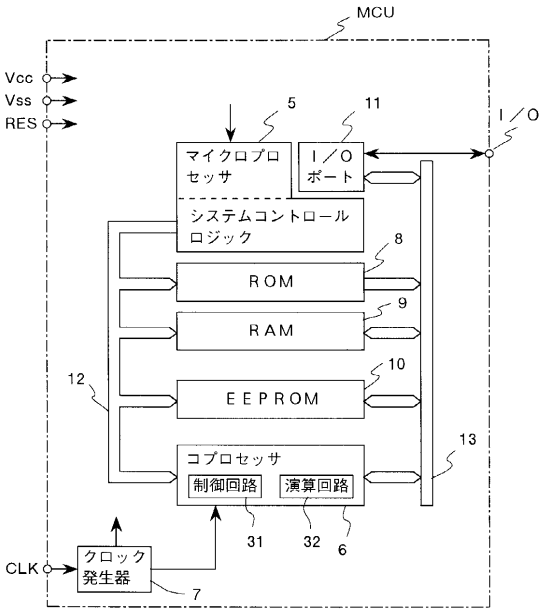
【図２】

【図2】



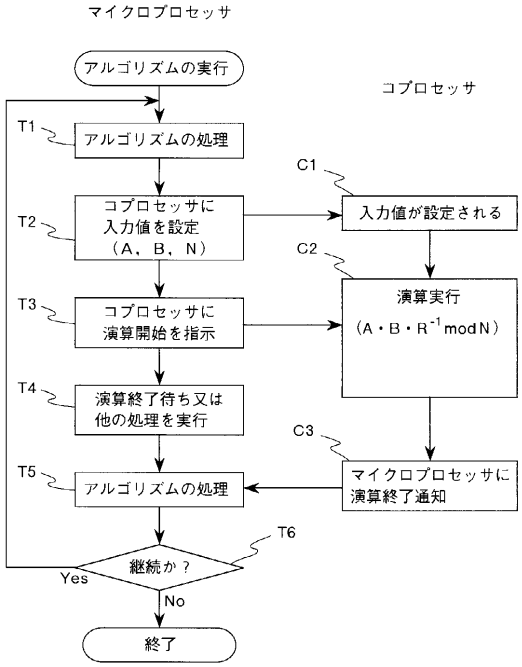
【図３】

【図3】



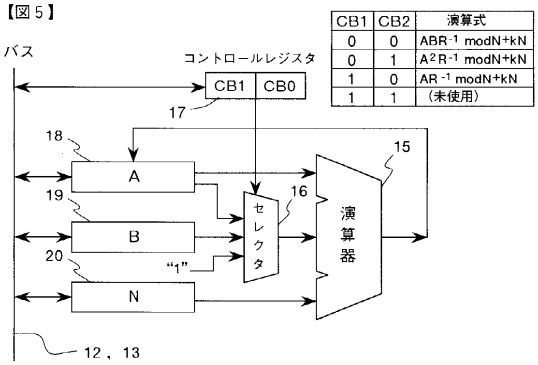
【図４】

【図4】



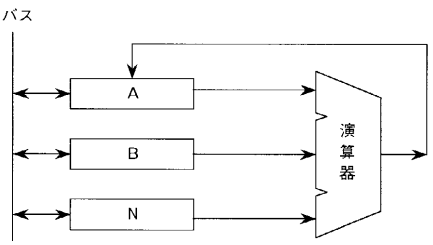
【図５】

【図5】

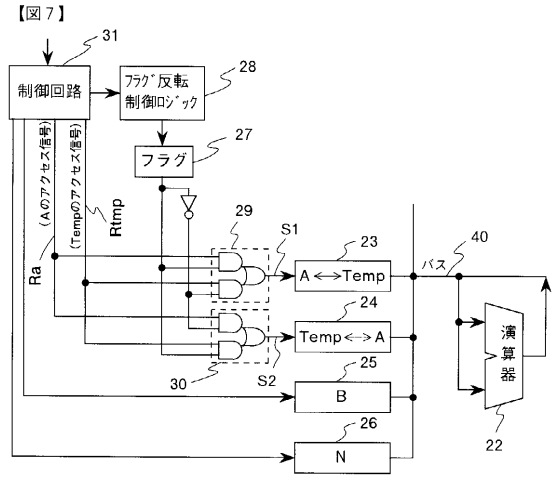


【図６】

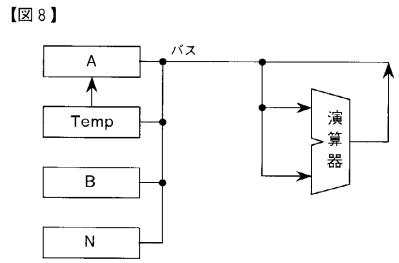
【図6】



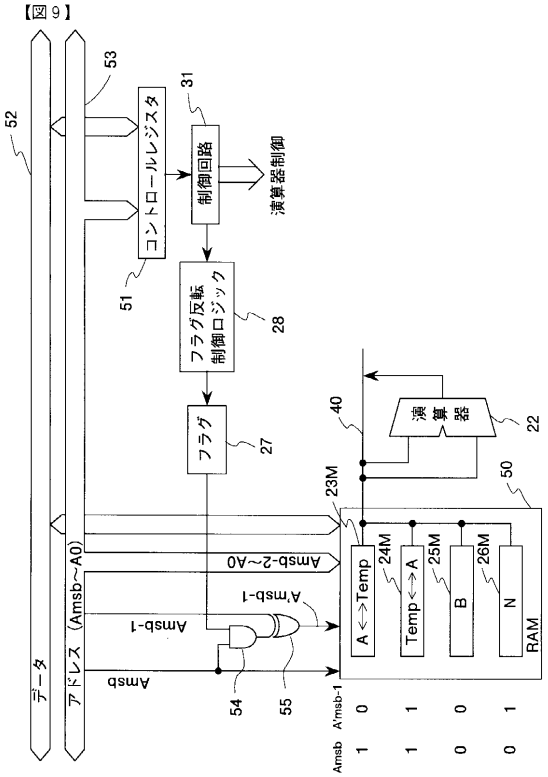
【図７】



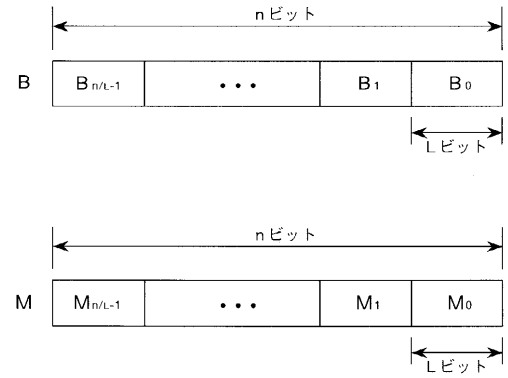
【図８】



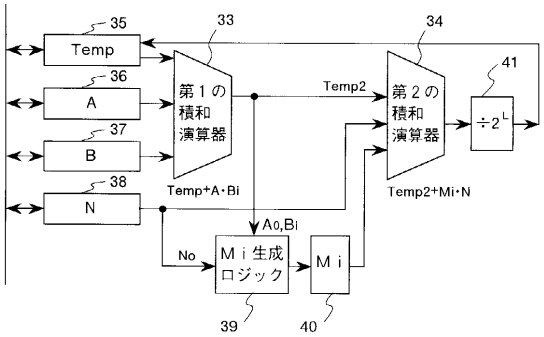
【図９】



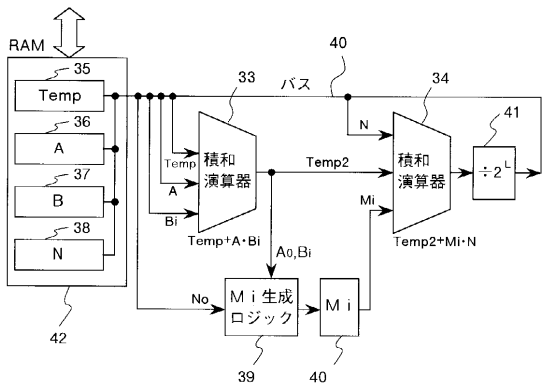
【図１０】



【図１１】

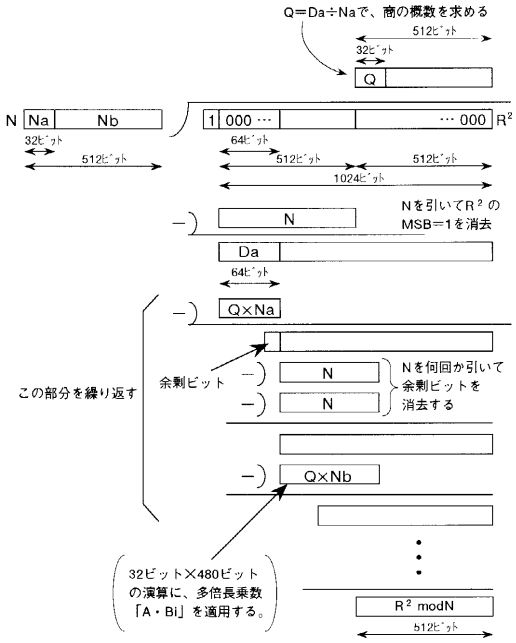


【図１２】



【図１３】

【図１３】



R²modNの計算概念図

【図１４】

【図１４】