

(19) 日本国特許庁(JP)

(12) 公 開 特 許 公 報(A)

(11) 特許出願公開番号  
特開2006-23647  
(P2006-23647A)

(43) 公開日 平成18年1月26日(2006.1.26)

(51) Int.Cl.  
G09C 1/00 (2006.01)  
G06F 7/72 (2006.01)

F I  
G09C 1/00 650A  
G06F 7/72

テーマコード (参考)  
5 J 1 0 4

審査請求 有 請求項の数 17 O L (全 18 頁)

(21) 出願番号	特願2004-203435 (P2004-203435)	(71) 出願人	000232036 NECマイクロシステム株式会社 神奈川県川崎市中原区小杉町1丁目403番53
(22) 出願日	平成16年7月9日(2004.7.9)	(71) 出願人	899000068 学校法人早稲田大学 東京都新宿区戸塚町1丁目104番地
		(74) 代理人	100123788 弁理士 宮崎 昭夫
		(74) 代理人	100120628 弁理士 岩田 慎一
		(74) 代理人	100127454 弁理士 緒方 雅昭
		(74) 代理人	100106138 弁理士 石橋 政幸

最終頁に続く

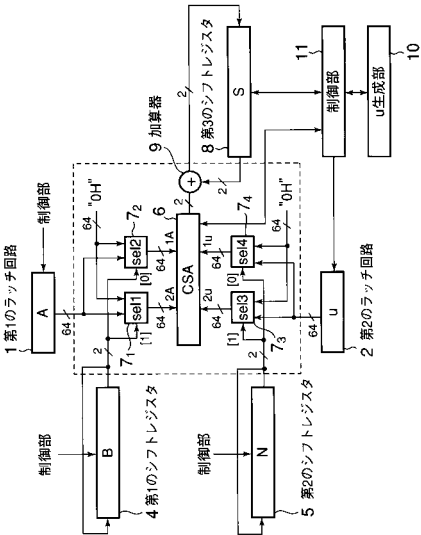
(54) 【発明の名称】 乗算剰余演算器及び情報処理装置

(57) 【要約】

【課題】 回路規模を増大させることなく、演算時間を短縮できる乗算剰余演算器及び情報処理装置を提供する。

【解決手段】  $S = S + A \times B + u \times N$  を算出するための乗算剰余演算器であって、複数のビット数  $q$  単位で供給される乗数  $B$  の値に応じて被乗数  $A$  または  $0$  の値を切換えて出力し、ビット数  $q$  単位で供給される乗数  $N$  の値に応じて被乗数  $u$  または  $0$  の値を切換えて出力するセレクタと、セレクタから順次出力される値を用いて  $A \times B + u \times N$  の演算を実行する桁上げ保存加算器と、桁上げ保存加算器からビット数  $q$  単位で出力される  $A \times B + u \times N$  の演算結果とビット数  $q$  単位で供給される過去の該演算結果とを加算し、該加算結果を乗算剰余演算結果  $S$  として出力する加算器とを有する構成とする。

【選択図】 図 1



## 【特許請求の範囲】

## 【請求項 1】

被乗数を A、u とし、乗数を B、N とし、乗算剰余演算結果を S としたとき、 $S = S + A \times B + u \times N$  を算出するための乗算剰余演算器であって、

複数のビット数 q 単位で供給される前記乗数 B の値に応じて前記被乗数 A または 0 の値を切換えて出力し、前記ビット数 q 単位で供給される前記乗数 N の値に応じて前記被乗数 u または 0 の値を切換えて出力するセクタと、

前記セクタから順次出力される値を用いて  $A \times B + u \times N$  の演算を実行する桁上げ保存加算器と、

前記桁上げ保存加算器から前記ビット数 q 単位で出力される前記  $A \times B + u \times N$  の演算結果と、前記ビット数 q 単位で供給される過去の該演算結果とを加算し、該加算結果を前記乗算剰余演算結果 S として出力する加算器と、  
を有する乗算剰余演算器。 10

## 【請求項 2】

前記被乗数 A を保持し、前記セクタに供給する第 1 の記憶素子と、

前記被乗数 u を保持し、前記セクタに供給する第 2 の記憶素子と、

前記乗数 B を保持し、前記セクタに前記ビット数 q 単位で供給する第 3 の記憶素子と

、

前記乗数 N を保持し、前記セクタに前記ビット数 q 単位で供給する第 4 の記憶素子と

、

前記加算器から出力される前記乗算剰余演算結果 S を保持し、前記ビット数 q 単位で該乗算剰余演算結果 S を前記加算器に供給する第 5 の記憶素子と、  
をさらに有する請求項 1 記載の乗算剰余演算器。 20

## 【請求項 3】

前記桁上げ保存加算器の動作を制御する制御部をさらに有する請求項 1 または 2 記載の乗算剰余演算器。

## 【請求項 4】

前記制御部は、

前記第 1 の記憶素子に前記被乗数 A をセットし、

前記第 2 の記憶素子に前記被乗数 u をセットし、 30

前記第 3 の記憶素子に前記乗数 B をセットし、

前記第 4 の記憶素子に前記乗数 N をセットし、

前記セクタに 0 を供給する請求項 3 記載の乗算剰余演算器。

## 【請求項 5】

予め算出された、前記被乗数 A、前記乗数 B、前記乗数 N、及び前記乗算剰余演算結果 S の値に対する前記被乗数 u の値の関係が格納される u 生成部をさらに有し、

前記制御部は、

前記  $S = S + A \times B + u \times N$  の演算時に前記 u 生成部を参照することで前記被乗数 u の値を決定する請求項 3 または 4 記載の乗算剰余演算器。 40

## 【請求項 6】

前記ビット数 q は 2 である請求項 1 乃至 5 のいずれか 1 項記載の乗算剰余演算器。

## 【請求項 7】

前記ビット数 q は 4 である請求項 1 乃至 5 のいずれか 1 項記載の乗算剰余演算器。

## 【請求項 8】

前記第 1 の記憶素子及び前記第 2 の記憶素子はラッチ回路である請求項 2 乃至 7 のいずれか 1 項記載の乗算剰余演算器。

## 【請求項 9】

前記第 3 の記憶素子、前記第 4 の記憶素子及び前記第 5 の記憶素子はシフトレジスタである請求項 2 乃至 8 のいずれか 1 項記載の乗算剰余演算器。

## 【請求項 10】

請求項 1 に記載の乗算剰余演算器と、

前記被乗数 A を保持し、前記セレクトタに供給する第 1 の記憶素子と、

前記被乗数 u を保持し、前記セレクトタに供給する第 2 の記憶素子と、

前記乗数 B を保持し、前記セレクトタに前記ビット数 q 単位で供給する第 3 の記憶素子と

、  
前記乗数 N を保持し、前記セレクトタに前記ビット数 q 単位で供給する第 4 の記憶素子と

、  
前記加算器から出力される前記乗算剰余演算結果 S を保持し、前記ビット数 q 単位で該乗算剰余演算結果 S を前記加算器に供給する第 5 の記憶素子と、  
を有する情報処理装置。

10

【請求項 1 1】

前記桁上げ保存加算器の動作を制御する制御部をさらに有する請求項 1 0 記載の情報処理装置。

【請求項 1 2】

前記制御部は、

前記第 1 の記憶素子に前記被乗数 A をセットし、

前記第 2 の記憶素子に前記被乗数 u をセットし、

前記第 3 の記憶素子に前記乗数 B をセットし、

前記第 4 の記憶素子に前記乗数 N をセットし、

前記セレクトタに 0 を供給する請求項 1 1 記載の情報処理装置。

20

【請求項 1 3】

予め算出された、前記被乗数 A、前記乗数 B、前記乗数 N、及び前記乗算剰余演算結果 S の値に対する前記被乗数 u の値の関係が格納される u 生成部をさらに有し、

前記制御部は、

前記  $S = S + A \times B + u \times N$  の演算時に前記 u 生成部を参照することで前記被乗数 u の値を決定する請求項 1 1 または 1 2 記載の情報処理装置。

【請求項 1 4】

前記ビット数 q は 2 である請求項 1 0 乃至 1 3 のいずれか 1 項記載の情報処理装置。

【請求項 1 5】

前記ビット数 q は 4 である請求項 1 0 乃至 1 3 のいずれか 1 項記載の情報処理装置。

30

【請求項 1 6】

前記第 1 の記憶素子及び前記第 2 の記憶素子はラッチ回路である請求項 1 0 乃至 1 5 のいずれか 1 項記載の情報処理装置。

【請求項 1 7】

前記第 3 の記憶素子、前記第 4 の記憶素子及び前記第 5 の記憶素子はシフトレジスタである請求項 1 0 乃至 1 6 のいずれか 1 項記載の情報処理装置。

【発明の詳細な説明】

【技術分野】

【0001】

本発明はべき乗剰余演算を効率よく処理するための乗算剰余演算器及びそれを備えた情報処理装置に関する。

40

【背景技術】

【0002】

近年、パーソナルコンピュータや PDA (Personal Digital(Data) Assistants) あるいは携帯電話機等の各種情報処理装置の処理能力が飛躍的に向上し、さらに各種記録メディアの大容量化や通信インフラストラクチャーの整備が進んだことで、個人情報や企業情報等がネットワークや無線手段を介して送受信される機会が増大している。そのため、それらの情報を秘匿化し第三者への漏洩を防ぐ技術が益々重要になってきている。

【0003】

50

送受信データを秘匿化するための一般的な手法としては、データを送受信する端末装置どうしが共通の鍵を用いて該データの暗号化と復号を行う共通鍵暗号方式がよく知られている。さらに、近年ではB to B、B to C等の電子商取引の拡大に伴ってPKI (Public Key Infrastructure) 技術が注目されている。

#### 【0004】

PKIの基本技術である公開鍵暗号方式は、公開鍵を用いて送信データを暗号化し、該公開鍵とペアとなる公開することのない秘密鍵を用いて受信データを復号する方式である。この公開鍵暗号方式は、送信側と受信側で異なる鍵を用い、かつ秘密鍵を通信相手に通知する必要が無いため、上述した共通鍵暗号方式に比べて秘匿化性能が向上する。

#### 【0005】

公開鍵暗号方式では、現在、RSA (Rivest, Shamir Adleman) 暗号が主として用いられている (例えば、非特許文献1参照)。RSA暗号は、任意の2つの素数を乗算した値Nを素因数分解する際の困難性とNを法とする数の世界の性質とを利用する暗号化方式であり、暗号化及び復号化のためにべき乗剰余演算 ( $M^d \bmod N$ ) を実行する。

#### 【0006】

べき乗剰余演算は、通常、以下に示す乗算剰余演算の繰り返し処理に置き換えて実行される。

#### 【0007】

例えば、 $d = 19$  とするとき、 $C = M^d \bmod N$  は、  
 $d = 19 = 1 + 2 \times (1 + 2 \times (0 + 2 \times (0 + 2 \times 1)))$  により、  
 $C = M^{19} \bmod N$   
 $= M^{1+2 \times (1+2 \times (0+2 \times (0+2 \times 1)))} \bmod N$   
 $= ((((M^1)^2 M^0)^2 M^0)^2 M^1)^2 M^1 \bmod N$   
 $= ((M^2)^2 M)^2 M \bmod N$

となる。このようにdを分解すれば、Mを単純にd回掛けるよりも演算回数を低減できるため、演算時間を短縮できる。なお、dの分解方法については様々な方法が知られており、上記はその一例を示している。

#### 【0008】

しかしながら、このような乗算剰余演算も、乗算によって演算桁数が倍になり、さらにその乗算結果をNで除算するため、ハードウェアまたはソフトウェアのいずれを利用して  
 も効率よく処理するのが非常に困難な演算である。そのため、乗算剰余演算を効率化するための様々な手法が検討され、代表的な例としてモンゴメリ (Montgomery) 法と呼ばれるアルゴリズムを応用した演算方法が知られている (例えば、特許文献1参照)。

#### 【0009】

モンゴメリ法を応用すると、除算を実質的に行わずに乗算と加減算で上記乗算剰余演算が実現可能であり、乗算剰余演算  $P(A, B)_N = A \cdot B \cdot r^{-n} \bmod N = S$  は、例えば、以下の(1)～(8)で示す手順で求めることができる。但し、 $0 < N < r^n$ 、Nは奇数 (Nとrは互いに素である)、 $0 < A < N$ 、 $0 < B < N$ 、 $A = A_{n-1} A_{n-2} \dots A_0$  (例えば  $A = A_3 A_2 A_1 A_0 = 1234$ ) である。

$$(1) \quad v = -N^{-1} \bmod r$$

$$(2) \quad S = 0$$

$$(3) \quad \text{for } i = 0 \text{ to } n - 1 \quad \{$$

$$(4) \quad S = S + A_i \cdot B$$

$$(5) \quad u = S \cdot v \bmod r$$

$$(6) \quad S = S + u \cdot N$$

$$(7) \quad S = S / r$$

$$(8) \quad \}$$

乗算剰余演算は、上記アルゴリズムから  $S = S + A_i \times B + u \times N$  ( $i = 0 \sim n - 1$ ) の繰り返し演算処理に置き換え可能であり、この処理を実現するための回路である乗算剰余演算器は、例えば図6に示すような構成になる。

10

20

30

40

50

## 【 0 0 1 0 】

図 6 は従来の乗算剰余演算器の構成を示すブロック図である。

## 【 0 0 1 1 】

図 6 に示すように、従来の乗算剰余演算器は、被乗数である上記 A の値を保持する第 1 のラッチ回路 5 1 と、被乗数である上記 u の値を保持する第 2 のラッチ回路 5 2 と、 $A + u$  の値を保持する第 3 のラッチ回路 5 3 と、1 ビット毎に供給される乗数 B、N の値に応じて被乗数 A、u、 $A + u$ 、または 0 H (全ビット 0) を選択し出力するセレクタ 5 7 と、セレクタ 5 7 から出力される値を用いて  $A \times B + u \times N$  の演算を行う周知の桁上げ保存加算器 (Carry Save Adder: 以下、CSA と称す) 5 6 と、CSA 5 6 から出力される乗算剰余演算結果 S と外部で保持された算出済みの乗算剰余演算結果 S とを加算し、該加算結果を乗算剰余演算結果 S として出力する加算器 5 9 とを有する構成である。なお、A、u、及び  $A + u$  の各値は、例えば不図示の制御部により第 1 のラッチ回路 5 1 ~ 第 3 のラッチ回路 5 3 に供給され、乗数 B、N、及び 0 H の各値は、例えば不図示の制御部によりセレクタ 5 7 に供給される。

## 【 0 0 1 2 】

図 6 に示す乗算剰余演算器では、乗算剰余演算器の処理ビット長 (例えば、512bit) の乗数 B、N がそれぞれ 1 ビット単位でセレクタ 5 7 に供給される。また、被乗数 A、u、 $A + u$  は、CSA 5 6 の処理ビット長 (図 6 では m ビット) に対応して、該ビット長単位でラッチ回路に格納され、CSA 5 6 に供給される。したがって、例えば乗算剰余演算器の処理ビット長が 512bit であり、CSA 5 6 の処理ビット長が 128bit の場合、図 6 に示す構成では、被乗数 A、u、 $A + u$  の選択処理を 5 1 2 回繰り返すことで  $A (128bit) \times B (512bit) + u (128bit) \times N (512bit)$  の演算が完了し、さらにそれを 4 回繰り返すことで  $A (512bit) \times B (512bit) + u (512bit) \times N (512bit)$  の演算処理が完了することになる。

## 【 0 0 1 3 】

セレクタ 5 7 は、1 ビットずつ供給される乗数 B、N の値に応じて、第 1 のラッチ回路 5 1 ~ 第 3 のラッチ回路 5 3 から供給される被乗数 A、u、 $A + u$ 、または 0 H を選択し CSA 5 6 に供給する。CSA 5 6 は、セレクタ 5 7 から順次供給される被乗数 A、u、 $A + u$  または 0 H をシフト加算することで  $A \times B + u \times N$  を算出し、その中間演算結果を保持しつつ乗算剰余演算結果 S を 1 ビット単位で出力する。

【非特許文献 1】三谷政昭著、「やり直しのための工業数学」、第 5 版、CQ 出版社、2003 年 2 月 1 日、p. 115 - 122

【特許文献 1】特表 2001 - 527673 号公報

## 【 発明の開示 】

## 【 発明が解決しようとする課題 】

## 【 0 0 1 4 】

現在、公開鍵暗号方式では、上記べき乗剰余演算の C、M、N、d に 1024 ビットの数値を用いた RSA 暗号が広く利用され、さらにビット数が増えることも予想される。そのため、暗号化及び復号化に膨大な量の乗算剰余演算を実行しなければならない。公開鍵暗号方式は、暗号化及び復号化に要する処理時間が共通鍵暗号方式に比べて長いことが問題であり、乗算剰余演算に要する演算時間の短縮が重要な課題となっている。

## 【 0 0 1 5 】

なお、市場では、携帯電話機、PDA、パーソナルコンピュータやサーバ装置等の情報処理装置の普及に伴い、処理性能が高く、かつ低コストな製品が求められている。したがって、このような要求を満たすためには、乗算剰余演算に要する演算時間を短縮すると共に、回路規模の削減を実現できる乗算剰余演算器が必須となる。

## 【 0 0 1 6 】

本発明は上記したような従来の技術が有する問題点を解決するためになされたものであり、演算時間をより短縮できる乗算剰余演算器及び情報処理装置を提供することを目的とする。

## 【 0 0 1 7 】

10

20

30

40

50

また、本発明のさらなる目的は、回路規模を増大させることなく演算時間を短縮できる乗算剰余演算器及び情報処理装置を提供することにある。

【課題を解決するための手段】

【0018】

上記目的を達成するため本発明の乗算剰余演算器は、被乗数をA、uとし、乗数をB、Nとし、乗算剰余演算結果をSとしたとき、 $S = S + A \times B + u \times N$ を算出するための乗算剰余演算器であって、

複数のビット数q単位で供給される前記乗数Bの値に応じて前記被乗数Aまたは0の値を切換えて出力し、前記ビット数q単位で供給される前記乗数Nの値に応じて前記被乗数uまたは0の値を切換えて出力するセクタと、

前記セクタから順次出力される値を用いて $A \times B + u \times N$ の演算を実行する桁上げ保存加算器と、

前記桁上げ保存加算器から前記ビット数q単位で出力される前記 $A \times B + u \times N$ の演算結果と、前記ビット数q単位で供給される過去の該演算結果とを加算し、該加算結果を前記乗算剰余演算結果Sとして出力する加算器と、  
を有する構成である。

【0019】

一方、本発明の情報処理装置は、上記乗算剰余演算器と、

前記被乗数Aを保持し、前記セクタに供給する第1の記憶素子と、

前記被乗数uを保持し、前記セクタに供給する第2の記憶素子と、

前記乗数Bを保持し、前記セクタに前記ビット数q単位で供給する第3の記憶素子と

、  
前記乗数Nを保持し、前記セクタに前記ビット数q単位で供給する第4の記憶素子と

、  
前記加算器から出力される前記乗算剰余演算結果Sを保持し、前記ビット数q単位で該乗算剰余演算結果Sを前記加算器に供給する第5の記憶素子と、  
を有する構成である。

【0020】

上記のように構成された乗算剰余演算器及び情報処理装置では、乗数を複数のビット数q単位でセクタに供給し、セクタにより該乗数の値に応じて被乗数または0の値を切換えてCSAに供給するため、CSAの処理ビット長をビット数qの値に反比例して短縮できる。

【0021】

また、本発明の乗算剰余演算器及び情報処理装置は、予め算出された、前記被乗数A、前記乗数B、前記乗数N、及び前記乗算剰余演算結果Sの値に対する前記被乗数uの値の関係が格納されるu生成部をさらに有し、

制御部により、前記 $S = S + A \times B + u \times N$ の演算時に前記u生成部を参照することで前記被乗数uの値を決定する構成である。

【0022】

なお、前記ビット数qは2または4であることが望ましい。

【0023】

上記のような乗算剰余演算器は、ビット数qを2または4とすることで、u生成部の回路規模の増大を抑制できる。

【発明の効果】

【0024】

本発明の乗算剰余演算器及び情報処理装置は、CSAの処理ビット長をビット数qの値に反比例して短縮できるため、従来の乗算剰余演算器よりも演算時間を短縮できる。

【0025】

また、CSAの処理ビット長を短縮することで、CSAが備えるフリップフロップ数が低減するため、乗算剰余演算器の回路規模が低減する。特に、ビット数qを2または4と

10

20

30

40

50

すれば、 $u$  生成部の回路規模が増大することがないため、回路規模を増大させることなく演算時間を短縮できる。

【発明を実施するための最良の形態】

【0026】

次に本発明について図面を参照して説明する。

【0027】

図1は本発明の乗算剰余演算器の一構成例を示すブロック図であり、図2は本発明の情報処理装置の一構成例を示すブロック図である。

【0028】

図1に示すように、本発明の乗算剰余演算器は、被乗数 $A$ の値を保持する第1のラッチ回路1と、被乗数 $u$ の値を保持する第2のラッチ回路2と、乗数 $B$ の値を保持する第1のシフトレジスタ4と、乗数 $N$ の値を保持する第2のシフトレジスタ5と、第1のシフトレジスタ4から複数ビット（図1では2bit）毎に供給される乗数 $B$ の値に応じて被乗数 $A$ または0Hを選択しCSA6に供給する第1のセクタ（Sel1） $7_1$ 及び第2のセクタ（Sel2） $7_2$ と、第2のシフトレジスタ5から複数ビット（図1では2bit）毎に供給される乗数 $N$ の値に応じて被乗数 $u$ または0Hを選択しCSA6に供給する第3のセクタ（Sel3） $7_3$ 及び第4のセクタ（Sel4） $7_4$ と、第1～第4のセクタ $7_1 \sim 7_4$ から供給される値を用いて $A \times B + u \times N$ の演算を行う周知のCSA6と、CSA6から複数ビット（図1では2bit）単位で出力される乗算剰余演算結果 $S$ を保持し、複数ビット（図1では2bit）単位で出力する第3のシフトレジスタ8と、CSA6から出力される $A \times B + u \times N$ の演算結果と第3のシフトレジスタ8の出力とを加算し、加算結果を乗算剰余演算結果 $S$ として第3のシフトレジスタ8に再び格納する加算器9と、被乗数 $u$ の値を生成するためのテーブルが格納される $u$ 生成部10と、被乗数 $A$ 、 $u$ の値を第1のラッチ回路1及び第2のラッチ回路2に供給し、乗数 $B$ 、 $N$ の値を第1のシフトレジスタ4及び第2のシフトレジスタ5に供給し、0Hを第1～第4のセクタ $7_1 \sim 7_4$ に供給すると共に、CSA6、 $u$ 生成部10及び第3のシフトレジスタ8の動作を制御する制御部11とを有する構成である。

【0029】

本発明の乗算剰余演算器は、制御部11による被乗数 $A$ 、 $u$ のラッチ回路へのセット、及び乗数 $B$ 、 $N$ のシフトレジスタへのセットを契機に、外部から供給される所定周波数のクロック（CK）にしたがって動作する回路であり、制御部11は、例えばプログラムにしたがって動作するCPU、DSPあるいは論理回路等によって実現される。

【0030】

このような構成において、本発明の乗算剰余演算器では、被乗数 $A$ 、 $u$ が、CSA6の処理ビット長に対応して複数に分割され、制御部11により該分割単位で第1及び第2のラッチ回路1、2に格納される。一方、乗数 $B$ 、 $N$ は、制御部11により、例えば乗算剰余演算器の処理ビット長で第1及び第2のシフトレジスタに格納される。なお、乗数 $B$ 、 $N$ も、所定のビット長毎に複数に分割され、制御部11により該分割単位で第1及び第2のシフトレジスタに格納されてもよい。

【0031】

各セクタには、第1のラッチ回路1及び第2のラッチ回路2から上記分割単位で被乗数 $A$ 、 $u$ が供給され、第1のシフトレジスタ4及び第2のシフトレジスタ5から複数ビット単位で乗数 $B$ 、 $N$ が供給される。図1では、乗数 $B$ 、 $N$ がそれぞれ2bit単位で第1～第4のセクタ $7_1 \sim 7_4$ に供給される例を示しているが、乗数 $B$ 、 $N$ は、4bit単位、あるいはそれ以上のビット数単位でセクタに供給されてもよい。なお、図1では4つのセクタを用いて被乗数 $A$ 、 $u$ または0Hを切換える構成を示しているが、乗数 $B$ 、 $N$ の値に対応する被乗数 $A$ 、 $u$ または0Hを選択できれば、セクタの数はいくつであってもよい。

【0032】

第1のセクタ $7_1$ 及び第2のセクタ $7_2$ は、第1のシフトレジスタ4から複数ビットづつ供給される乗数 $B$ の値に応じて、被乗数 $A$ （1A、2A）または0Hを選択しCSA

10

20

30

40

50

6 に供給する。同様に、第 3 のセレクタ 7<sub>3</sub> 及び第 4 のセレクタ 7<sub>4</sub> は、第 2 のシフトレジスタ 5 から複数ビットずつ供給される乗数 N の値に応じて、被乗数 u ( 1 u、2 u ) または 0 H を選択し C S A 6 に供給する。

【 0 0 3 3 】

ここで、1 A は被乗数 A の 1 倍の値であり、2 A は被乗数 A の 2 倍の値である。また、1 u は被乗数 u の 1 倍の値であり、2 u は被乗数 u の 2 倍の値である。2 A、2 u は、例えば被乗数 A、u の値を 1 ビットシフトさせれば得られるため容易に生成できる。図 1 では第 1 のセレクタ 7<sub>1</sub> で 2 A を生成し、第 3 のセレクタ 7<sub>3</sub> で 2 u を生成する例を示しているが、2 A、2 u は C S A 6 内で生成してもよい。

【 0 0 3 4 】

C S A 6 は、各セレクタから順次供給される被乗数または 0 H をシフト加算することで  $A \times B$ 、及び  $u \times N$  をそれぞれ算出し、それらの加算結果 ( 乗算剰余演算結果 ) S を複数ビット単位で出力する。C S A 6 から出力された演算結果は、第 3 のシフトレジスタ 8 の出力 ( 過去の乗算剰余演算結果 S ) と複数ビット単位で加算され、加算結果は第 3 のシフトレジスタ 8 に再び格納される。

【 0 0 3 5 】

なお、図 1 に示した第 1 のラッチ回路 1、第 2 のラッチ回路 2、第 1 シフトレジスタ 4、第 2 のシフトレジスタ 5、第 3 のシフトレジスタ 8、及び u 生成部 10 は、乗算剰余演算器の内部に備えている必要はなく、乗算剰余演算器を利用する情報処理装置に備えていてもよい。同様に、制御部 11 も乗算剰余演算器の内部に備えている必要はなく、乗算剰余演算器を利用する情報処理装置が備える処理装置 ( C P U ) によって実現してもよい。

【 0 0 3 6 】

また、被乗数 A、u は、ラッチ回路に格納する必要はなく、例えばシフトレジスタや R A M 等のようにデータを一時的に保持できる記憶素子であればどのようなものを用いてもよい。同様に、乗数 B、N、及び演算結果 S は、シフトレジスタに格納する必要はなく、例えば R A M のように格納されたデータを複数ビット単位で出力できる記憶素子であればどのようなものを用いてもよい。

【 0 0 3 7 】

図 2 に示すように、本発明の情報処理装置は、例えばパーソナルコンピュータやサーバ装置等のコンピュータシステムであり、プログラムにしたがって所定の処理を実行する処理装置 20 と、処理装置 20 に対してコマンドや情報等を入力するための入力装置 30 と、処理装置 20 の処理結果をモニタするための出力装置 40 とを有する構成である。

【 0 0 3 8 】

処理装置 20 は、C P U 21 と、C P U 21 の処理に必要な情報を一時的に記憶する主記憶装置 22 と、C P U 21 に上記制御部 11 の処理を実行させるプログラムが記録された記録媒体 23 と、処理に必要なデータ等を蓄積するデータ蓄積装置 24 と、主記憶装置 22、記録媒体 23、及びデータ蓄積装置 24 とのデータ転送を制御するメモリ制御インタフェース部 25 と、入力装置 30 及び出力装置 40 とのインタフェース装置である I / O インタフェース部 26 と、図 1 に示した乗算剰余演算器 27 と、ネットワーク等との通信を制御するインタフェースである通信制御装置 28 とを備え、それらがバス 29 等を介して接続された構成である。なお、処理装置 20 には、乗算剰余演算器 27 の構成に応じて、被乗数 A、u を保持するラッチ回路、及び乗数 B、N、及び演算結果 S を保持するシフトレジスタ等を備えていてもよい。

【 0 0 3 9 】

処理装置 20 は、記録媒体 23 に記録されたプログラムにしたがって C P U 21 により上記制御部 11 の処理を実行し、乗算剰余演算器 27 を用いて  $S = S + A_i \times B + u \times N$  の演算を実行する。なお、記録媒体 23 は、磁気ディスク、半導体メモリ、光ディスクあるいはその他の記録媒体であってもよい。

【 0 0 4 0 】

10

20

30

40

50



次に、本発明の乗算剰余演算器の動作について図面を用いて具体的に説明する。

【 0 0 4 1 】

以下では、A、u、B、Nがそれぞれ512bitであり、処理ビット長が64bitのCSA6を用い、第1及び第2のシフトレジスタ4、5がそれぞれ2bit単位で各セクタに乗数B、Nを供給し、第3のシフトレジスタ8が2bit単位で乗算剰余演算結果Sを入出力する場合を例にして説明する。また、第1及び第2のシフトレジスタ4、5には乗数B、Nがそれぞれ512bit単位で格納され、第1及び第2のラッチ回路1、2には被乗数A、uがCSA6の処理ビット長に合わせて64bit単位で格納されるものとする。

【 0 0 4 2 】

処理ビット長が64bitのCSA6を用い、乗数B、Nを2bit単位で出力する場合、A、u、B、Nがそれぞれ512bitの乗算剰余演算 ( $512\text{bit} \times 512\text{bit} \times 2^{-512} \bmod 512\text{bit}$ ) は、 $64\text{bit} \times 512\text{bit} \times 2^{-64} \bmod 512\text{bit} (A \times B \times 2^{-64} \bmod N)$  の演算を繰り返し実行すればよい。

10

【 0 0 4 3 】

本発明の乗算剰余演算器では、モンゴメリ法による乗算剰余演算の特徴である、下位ビットが0になることを利用して（ここでは、下位64bitが0H）、上記S、A、B、Nの値に対応するuを予め算出し、u生成部10にテーブル形式で格納しておく。

【 0 0 4 4 】

例えば、乗数を2bit単位で出力する場合、uの値は以下のように求める（但し、Nは奇数）。

20

【 0 0 4 5 】

N[1:0]=01, (S+AiB)[1:0]=00のとき、  
 $S=S+AiB+uN=00$ となるuは、 $u[1:0]=00$   
 N[1:0]=01, (S+AiB)[1:0]=01のとき、  
 $S=S+AiB+uN=00$ となるuは、 $u[1:0]=11$   
 N[1:0]=01, (S+AiB)[1:0]=10のとき、  
 $S=S+AiB+uN=00$ となるuは、 $u[1:0]=10$   
 N[1:0]=01, (S+AiB)[1:0]=11のとき、  
 $S=S+AiB+uN=00$ となるuは、 $u[1:0]=01$   
 N[1:0]=11, (S+AiB)[1:0]=00のとき、  
 $S=S+AiB+uN=00$ となるuは、 $u[1:0]=00$   
 N[1:0]=11, (S+AiB)[1:0]=01のとき、  
 $S=S+AiB+uN=00$ となるuは、 $u[1:0]=01$   
 N[1:0]=11, (S+AiB)[1:0]=10のとき、  
 $S=S+AiB+uN=00$ となるuは、 $u[1:0]=10$   
 N[1:0]=11, (S+AiB)[1:0]=11のとき、  
 $S=S+AiB+uN=00$ となるuは、 $u[1:0]=11$

30

以上をまとめると、表1のようになる。

【 0 0 4 6 】

【表 1】

N[1]	S+AiB[1:0]	u
0	00	00
0	01	11
0	10	10
0	11	01
1	00	00
1	01	01
1	10	10
1	11	11

10

## 【0047】

ここで、A、B、Nはいずれも既知の値（第1のラッチ回路1、第1のシフトレジスタ4及び第2のシフトレジスタ5に格納する値）であり、Sは0H（演算開始時）または直前の $64\text{bit} \times 512\text{bit} \times 2^{-64} \bmod 512\text{bit}$ の演算結果を用いるため既知である。なお、Nは奇数であるため、N[1:0]=01または11で固定である。したがって、A、B、及びSの各値を基に算出した被乗数uの値をテーブル形式でu生成部10に格納しておき、制御部11は

20

該テーブルを参照して被乗数uの値を決定する。なお、参考として表2に乗数B、Nを4bit単位で出力する場合に用いるuを生成するためのテーブルを示す。

## 【0048】

【表 2】

N[3:1], S+AB[3:0]	u	N[3:1], S+AB[3:0]	u	N[3:1], S+AB[3:0]	u	N[3:1], S+AB[3:0]	u
0	0	16	0	32	0	48	0
1	15	17	5	33	3	49	9
2	14	18	10	34	6	50	2
3	13	19	15	35	9	51	11
4	12	20	4	36	12	52	4
5	11	21	9	37	15	53	13
6	10	22	14	38	2	54	6
7	9	23	3	39	5	55	15
8	8	24	8	40	8	56	8
9	7	25	13	41	11	57	1
10	6	26	2	42	14	58	10
11	5	27	7	43	1	59	3
12	4	28	12	44	4	60	12
13	3	29	1	45	7	61	5
14	2	30	6	46	10	62	14
15	1	31	11	47	13	63	7

10

N[3:1], S+AB[3:0]	u	N[3:1], S+AB[3:0]	u	N[3:1], S+AB[3:0]	u	N[3:1], S+AB[3:0]	u
64	0	80	0	96	0	112	0
65	7	81	13	97	11	113	1
66	14	82	10	98	6	114	2
67	5	83	7	99	1	115	3
68	12	84	4	100	12	116	4
69	3	85	1	101	7	117	5
70	10	86	14	102	2	118	6
71	1	87	11	103	13	119	7
72	8	88	8	104	8	120	8
73	15	89	5	105	3	121	9
74	6	90	2	106	14	122	10
75	13	91	15	107	9	123	11
76	4	92	12	108	4	124	12
77	11	93	9	109	15	125	13
78	2	94	6	110	10	126	14
79	9	95	3	111	5	127	15

20

30

## 【0049】

本発明の乗算剰余演算器では、まず、制御部11により、第1のラッチ回路1に被乗数A(512bit)の最下位64bitのデータをセットし、第1のシフトレジスタ4に乗数B(512bit)のデータをセットし、第2のシフトレジスタ5に乗数N(512bit)のデータをセットする。

40

## 【0050】

続いて、制御部11は、64bitの被乗数A、64bitの乗数B、64bitの乗数Nからu生成部10に格納されたテーブルを参照してu(64bit分)の値を求め、第2のラッチ回路2に格納する。

## 【0051】

制御部11による第1のラッチ回路1、第2のラッチ回路2、第1のシフトレジスタ4及び第2のシフトレジスタ5に対する被乗数または乗数のセットが完了すると、乗算剰余演算器は $S = S + A \times B + u \times N$ の演算を開始する。

## 【0052】

乗算剰余演算器は、まず、第1のセレクタ7<sub>1</sub>及び第2のセレクタ7<sub>2</sub>にて、第1のシフ

50

トレジスタ 4 から出力される 2bit の乗数 B の値から被乗数 A (64bit) または 0 H を選択し C S A 6 へ供給する。ここでは、第 1 のセクタ 7<sub>1</sub> により 0 H / 2 A を切換え、第 2 のセクタ 7<sub>2</sub> により 0 H / 1 A を切換える。

【 0 0 5 3 】

同様に、乗算剰余演算器は、第 3 のセクタ 7<sub>3</sub> 及び第 4 のセクタ 7<sub>4</sub> にて、第 2 のシフトレジスタ 5 から出力される 2bit の乗数 N の値から被乗数 u (64bit) または 0 H を選択し C S A 6 へ供給する。ここでは、第 3 のセクタ 7<sub>3</sub> により 0 H / 2 u を切換え、第 4 のセクタ 7<sub>4</sub> により 0 H / 1 u を切換える。

【 0 0 5 4 】

C S A 6 は、各セクタから順次供給される被乗数または 0 H の値を、桁合わせを実行しつつ加算することで  $A \times B$ 、及び  $u \times N$  を算出し、それらの加算結果 (乗算剰余演算結果) S を 2bit 単位で出力する。C S A 6 から出力された演算結果は、第 3 のシフトレジスタ 8 の出力と 2bit 単位で加算器 9 にて加算され、加算後の値が第 3 のシフトレジスタ 8 に再び格納される。

【 0 0 5 5 】

図 3 は本発明の乗算剰余演算器が実行する演算処理のうち、 $A \times B$  の演算処理の手順を示している。 $u \times N$  の演算処理も図 3 に示す  $A \times B$  の演算処理と同様である。図 3 に示す  $A \times B$  の演算結果の下位 2 ビット ( 1 ) と、同様にして求めた  $u \times N$  の演算結果の下位 2 ビットの加算結果が、C S A 6 の出力 (2bit) となる。

【 0 0 5 6 】

この処理を第 1 のシフトレジスタ 4 及び第 2 のシフトレジスタ 5 内の全てのビットデータに対して繰り返し実行することで、 $64\text{bit} \times 512\text{bit} \times 2^{-64} \bmod 512\text{bit}$  の演算が終了する。但し、この段階では C S A 6 の内部に部分積の演算結果の上位 64bit が残っているため、このデータを制御部 11 の指示により第 3 のシフトレジスタ 8 に格納する。その結果、第 3 のシフトレジスタ 8 に  $64\text{bit} \times 512\text{bit} \times 2^{-64} \bmod 512\text{bit}$  の演算結果 S が格納される。

【 0 0 5 7 】

乗算剰余演算器は、 $64\text{bit} \times 512\text{bit} \times 2^{-64} \bmod 512\text{bit}$  の演算が完了すると、制御部 11 により第 1 のラッチ回路 1 に被乗数 A (512bit) の次の下位 64bit のデータ (最下位から 65bit 目 ~ 128bit 目のデータ) をセットし、上記と同様に u 生成部 10 のテーブルを参照して被乗数 u の値を求め、求めた値を第 2 のラッチ回路 2 に格納した後、再び  $64\text{bit} \times 512\text{bit} \times 2^{-64} \bmod 512\text{bit}$  の演算を開始する。

【 0 0 5 8 】

以降、第 1 のラッチ回路 1 に格納される被乗数 A (512bit) の全てのビットデータに対して同様の処理を繰り返し実行する。すなわち、上記  $64\text{bit} \times 512\text{bit} \times 2^{-64} \bmod 512\text{bit}$  の演算を 8 回繰り返す。その結果、本発明の乗算剰余演算器による  $512\text{bit} \times 512\text{bit} \times 2^{-512} \bmod 512\text{bit}$  の演算が終了する。

【 0 0 5 9 】

次に、本発明の乗算剰余演算器の効果について図面を用いて説明する。

【 0 0 6 0 】

図 4 は乗数を 1bit 単位で出力する従来の乗算剰余演算器の回路規模及び乗数を 2bit または 4bit 単位で出力する本発明の乗算剰余演算器の回路規模を示すグラフである。また、図 5 は乗数を 1bit 単位で出力する従来の乗算剰余演算器の処理クロック数及び乗数を 2bit または 4bit 単位で出力する本発明の乗算剰余演算器の処理クロック数を示すグラフである。なお、図 5 は、乗算剰余演算器の処理ビット長を 512bit としたときの乗算剰余演算に必要な処理クロック数を示している。

【 0 0 6 1 】

図 4 及び図 5 に示す 1bit 構成とは乗数を 1bit 単位で出力する従来の乗算剰余演算器の構成を示し、2bit 構成とは乗数を 2bit 単位で出力する本発明の乗算剰余演算器の構成を示し、4bit 構成とは乗数を 4bit 単位で出力する本発明の乗算剰余演算器の構成を示している。

以降の説明でも1bit構成、2bit構成、あるいは4bit構成と称した場合は同様の構成を示しているものとする。

【0062】

また、図4及び図5に示すグラフの横軸は、乗算剰余演算器の処理ビット長（128bit、256bit、512bit）を示し、以下では、乗算剰余演算器の処理ビット長が同じである場合、乗数の出力ビット単位に反比例してCSA6の処理ビット長が短縮されるものとする。それらの関係を表3に示す。ここで、表3の各エントリは（CSAの処理ビット長）＊（出力ビット数）を示している。

【0063】

【表3】

10

乗算剰余演算器	4bit 構成	2bit 構成	1bit 構成
128bit	32bit*4bit	64bit*2bit	128bit*1bit
256bit	64bit*4bit	128bit*2bit	256bit*1bit
512bit	128bit*4bit	256bit*2bit	512bit*1bit

【0064】

図4から分かるように、乗算剰余演算器の処理ビット長が同じである場合、乗数を複数ビット単位で出力する本発明の乗算剰余演算器は、乗数を1bit単位で出力する従来の乗算剰余演算器に比べて回路規模が低減する。

20

【0065】

図4を基に1bit構成の従来の乗算剰余演算器と4bit構成の本発明の乗算剰余演算器を比較した場合、本発明の乗算剰余演算器は、従来の約半分の回路規模となる。これは2bit構成とすることでCSA6の処理ビット長を従来の半分にできるためであり、4bit構成とすることでCSA6の演算ビット長を従来の1/4にできるためである。

【0066】

例えば、乗算剰余演算器の処理ビット長を128bitとした場合、従来の乗算剰余演算器では、CSAでSUM（加算結果）の値とCARRY（桁上げ）の値をそれぞれ128個ずつ保持する必要があるため、256個のフリップフロップ（Data-F/F）が必要になる。

30

【0067】

それに対して、2bit構成の本発明の乗算剰余演算器が備えるCSA6では、処理ビット長が従来の半分の64bitで済むため、SUMの値とCARRYの値を保持するフリップフロップも128個で済む。すなわち、本発明のように乗数を複数ビット単位で出力すると、CSA6が備えるフリップフロップの数を大きく削減できるため、回路規模を低減できる。

【0068】

一方、図5から分かるように、乗算剰余演算器の処理ビット長が同じである場合、乗数を複数ビット単位で出力する本発明の乗算剰余演算器は、乗数を1bit単位で出力する従来の乗算剰余演算器に比べて処理クロック数が少なくなる。これはCSA6内に残る演算結果を出力する処理時間の差から生じる結果である。

40

【0069】

本発明の乗算剰余演算器では、上述したようにCSA6の処理ビット長を従来の半分あるいは1/4にできるが、被乗数を分割して処理するため、乗算剰余演算を複数回繰り返すことになる。そのため、本発明の乗算剰余演算器では、従来の乗算剰余演算器よりも繰り返し演算の回数が増え、CSA6内に残る部分積の演算結果を出力する回数も増えてしまう。

【0070】

しかしながら、本発明の乗算剰余演算器では、CSA6の処理ビット長を短縮できることから、CSA6内に残る演算結果を出力する処理時間も2bit構成の場合で従来の半分となり、4bit構成の場合で従来の1/4となる。そのため、僅かではあるが、1つのA、u

50

、B、Nに対する乗算剰余演算の処理時間は従来よりも低減する。

【0071】

本発明の乗算剰余演算器は、処理時間の大幅な低減は実現できないが、多数の数字の配列に対して大きな値のべき乗剰余演算を行うRSAによる暗号化及び復号に本発明の乗算剰余演算器を用いる場合は、この僅かな処理時間の向上が非常に有益となる。

【0072】

ところで、被乗数uは、乗数B、Nの出力ビット数をqとすると、上記モンゴメリ法を応用したアルゴリズムの(1)、(5)から以下の式で算出できる。

【0073】

$$v = -N^{-1} \bmod 2^q$$

$$u = Sv \bmod 2^q$$

ここで、vは演算開始時に一度だけ計算する値である。なお、rに代えて $2^q$ としているのはrを2進数で表したためである。

【0074】

q=1となる従来の乗算剰余演算器では、Nが奇数であることから $v=1$ となるため、 $u = S \bmod 2 = S[0]$ となり、被乗数uはSの下位ビットに等しくなる。したがって、被乗数uを実施的に計算する必要はない。

【0075】

しかしながら、 $q > 1$ となる本発明の乗算剰余演算器では、 $u = S[0]$ が成立しないため、上記2つの演算が必要になる。但し、qの値が小さい場合(例えば、 $q=2, 4$ )は、v、uも2bitまたは4bitであり、その演算に必要なN、Sも2bitまたは4bitである。そのため、本発明ではA、B、S、Nの値から予めuの値を算出してテーブルを作成しておき、該テーブルを参照することで第2のラッチ回路2に格納するuを決定している。このqの値を大きくすれば、CSA6の処理ビット長をさらに短縮できるため、乗算剰余演算の処理時間をさらに短縮することができる。

【0076】

しかしながら、 $q > 4$ の場合、すなわち乗数B、Nを8ビット以上で出力する構成では、被乗数uをテーブル内から選択するために必要な、例えばデコーダ等の回路規模が増大するため、記憶素子を含むu生成部10の回路規模が増大し、上述したCSA6の処理ビット長を短縮することによる乗算剰余演算器の回路規模の低減効果を相殺してしまう。

【0077】

表4にqの値に対するu生成部10のレイアウト面積(単位： $\text{mm}^2$ )を示し、表5にqの値に対するCSAとu生成部とを含む総レイアウト面積(単位： $\text{mm}^2$ )を示す。

【0078】

【表4】

q=1	q=2	q=4	q=8
0	0.003	0.014	0.937

【0079】

【表5】

CSA+u生成部	q=1	q=2	q=4	q=8
32bit	0.103	0.169	0.308	1.371
64bit	0.292	0.423	0.592	1.903
128bit	0.580	0.842	1.171	2.988
256bit	1.153	1.691	2.310	5.135

【0080】

表 4 及び表 5 から分かるように、例えば C S A の処理ビット長を 256bit としたとき、 $q = 1$  のときの総レイアウト面積に対して、C S A の処理ビット長を 128bit にできる  $q = 2$  の場合及び C S A の処理ビット長を 64bit にできる  $q = 4$  の場合の総レイアウト面積は低減する。しかしながら、 $q = 8$  にすると総レイアウト面積が増大してしまう。

【 0 0 8 1 】

したがって、本発明の乗算剰余演算器では、 $q$  の値が 2 または 4 であることが回路規模の増大を抑制しつつ演算時間を短縮できるために望ましい。但し、回路規模よりも演算時間の向上を優先する場合は、 $q$  の値を 8 以上に設定してもよい。その場合、 $q$  の値は  $u$  生成部 10 のレイアウト面積の増大を考慮しつつ最適な値を選択すればよい。

【図面の簡単な説明】

10

【 0 0 8 2 】

【図 1】本発明の乗算剰余演算器の一構成例を示すブロック図である。

【図 2】本発明の情報処理装置の一構成例を示すブロック図である。

【図 3】本発明の乗算剰余演算器が実行する演算処理のうち、 $A \times B$  の演算処理の手順を示す模式図である。

【図 4】本発明の乗算剰余演算器の回路規模を示すグラフである。

【図 5】本発明の乗算剰余演算器の処理クロック数を示すグラフである。

【図 6】従来の乗算剰余演算器の構成を示すブロック図である。

【符号の説明】

【 0 0 8 3 】

20

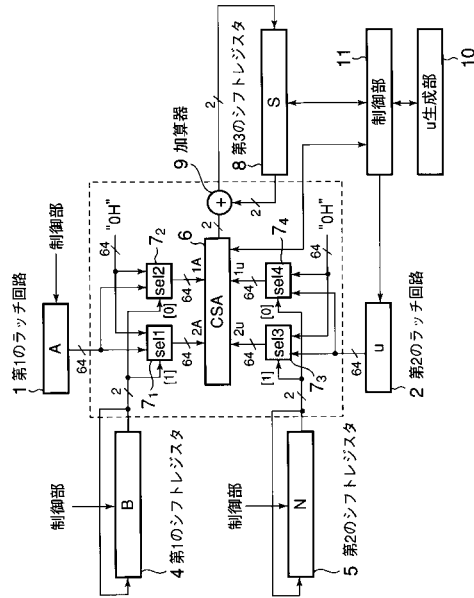
- 1 第 1 のラッチ回路
- 2 第 2 のラッチ回路
- 4 第 1 のシフトレジスタ
- 5 第 2 のシフトレジスタ
- 6 C S A
- 7<sub>1</sub> 第 1 のセレクタ
- 7<sub>2</sub> 第 2 のセレクタ
- 7<sub>3</sub> 第 3 のセレクタ
- 7<sub>4</sub> 第 4 のセレクタ
- 8 第 3 のシフトレジスタ

30

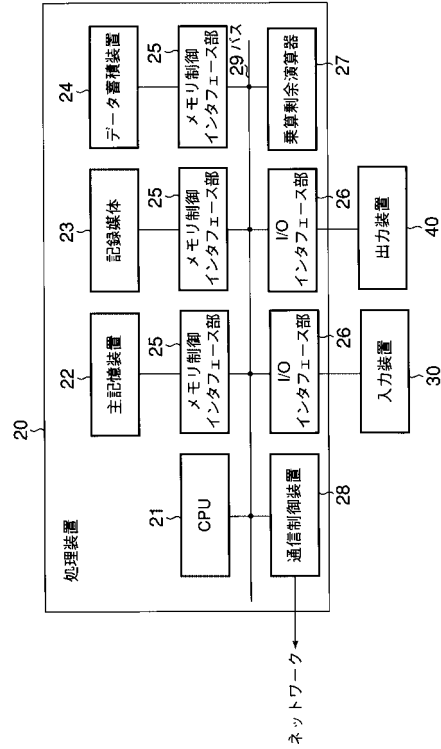
- 9 加算器
- 10  $u$  生成部
- 11 制御部
- 20 処理装置
- 21 C P U
- 22 主記憶装置
- 23 記録媒体
- 24 データ蓄積装置
- 25 メモリ制御インタフェース部
- 26 I / O インタフェース部
- 27 乗算剰余演算器
- 28 通信制御装置
- 29 バス
- 30 入力装置
- 40 出力装置

40

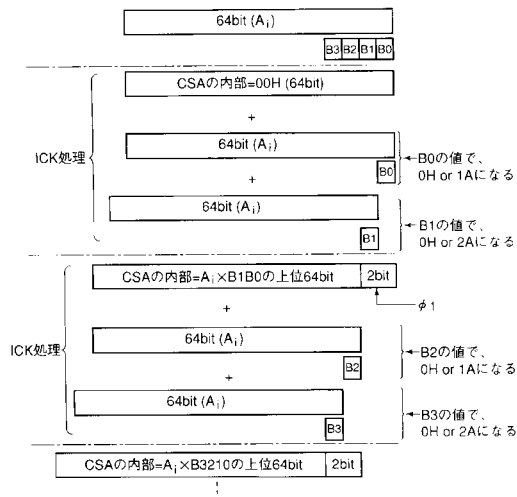
【図1】



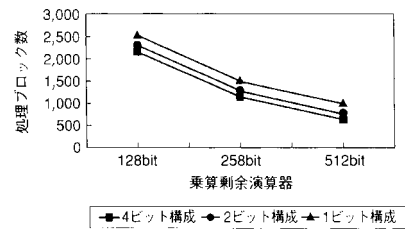
【図2】



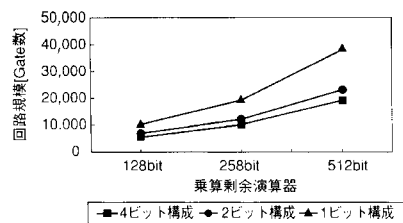
【図3】



【図5】

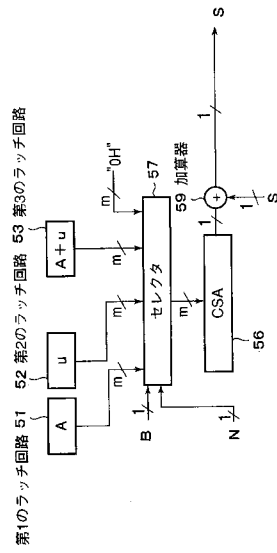


【図4】





【図 6】



---

フロントページの続き

(72)発明者 東 邦彦

神奈川県川崎市中原区小杉町1丁目403番53 NECマイクロシステム株式会社内

(72)発明者 久門 亨

神奈川県川崎市中原区小杉町1丁目403番53 NECマイクロシステム株式会社内

(72)発明者 後藤 敏

東京都新宿区戸塚町一丁目104番地 学校法人 早稲田大学内

(72)発明者 池永 剛

東京都新宿区戸塚町一丁目104番地 学校法人 早稲田大学内

Fターム(参考) 5J104 AA18 AA22 NA18