(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2004-501396
(P2004-501396A)
(43)公表日　平成16年1月15日(2004.1.15)

(51)Int.Cl.[7]	F I			テーマコード（参考）
G09C 1/00	G09C	1/00	620B	5J104
G06F 7/72	G09C	1/00	620Z	
	G09C	1/00	650A	
	G06F	7/72		

審査請求 未請求　予備審査請求 有　（全 114 頁）

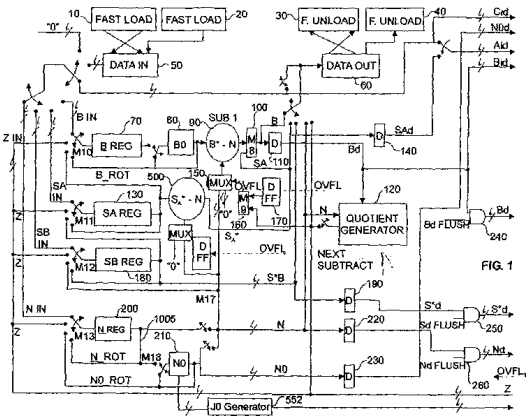
(21)出願番号　特願2001-585437(P2001-585437)
(86)(22)出願日　平成13年5月14日(2001.5.14)
(85)翻訳文提出日　平成14年11月15日(2002.11.15)
(86)国際出願番号　PCT/IL2001/000425
(87)国際公開番号　WO2001/089129
(87)国際公開日　平成13年11月22日(2001.11.22)
(31)優先権主張番号　136151
(32)優先日　平成12年5月15日(2000.5.15)
(33)優先権主張国　イスラエル(IL)
(31)優先権主張番号　139674
(32)優先日　平成12年11月14日(2000.11.14)
(33)優先権主張国　イスラエル(IL)

(71)出願人　502415179
エム－システムズ　フラッシュ　ディスク　パイオニアーズ　リミテッド
イスラエル　オメール　８４９６５、　オメール　インダストリアル　パーク　３Ｂ
(74)代理人　100109955
弁理士　細井　貞行
(74)代理人　100090619
弁理士　長南　満輝男
(74)代理人　100111785
弁理士　石渡　英房
(72)発明者　ドロル、イタイ
イスラエル　オメール　８４９６５、　ハルトジット　ストリート　１３

最終頁に続く

(54)【発明の名称】整数の計算フィールド範囲の拡張

(57)【要約】
出願人によって先に実装されたような同時縮小を有する直列／並列モンゴメリモジュラー乗算法の拡張であって、素体及びＧＦ（2^q）多項式ベースのフィールドの両方において、先のモジュラー乗算手順を拡張する複数の予測関数を実行することによりオペランドのフローを簡略化して実行するのに革新的に適している。



【特許請求の範囲】
【請求項１】
多項式ベースのＧＦ（2^q）及びＧＦ（ｐ）の両フィールドの演算において▲○×▼乗算及び平方を実行するマイクロエレクトロニクス装置であって、
平方及び減算が直列供給される基数$2^{▲l▼}$の乗数Ｂをｋ文字の被乗数セグメントA_i及びｋ文字の▲○＋▼アキュムレーターと共に使用し、限定された合同がモジュラスＮ上で被乗数A_iに乗数Ｂを掛け合わせてシストリックな態様で「即座に」実行され、その結果が多くとも２ｋ＋１文字長さであり、保存されないｋの最初に発生し無視されたゼロ文字を含み、ｋ文字がモジュラスと同等のビットを有する前記演算が二つの段階で行われ、
各々が少なくともｎビット長さのオペランドを保持するように動作し、夫々符号Ｂの乗数値及び符号Ｎの2^nより小さなモジュラスを保存するように動作する第一主メモリレジスタ手段Ｂ及び第二主メモリレジスタ手段Ｎと、
装置から発生する全ての最初のｋ文字がゼロにされるように▲○＋▼アダーアキュムレーターデバイスにおける値にモジュラス値が▲○＋▼加算されるような場合に、「即座に」予測するように動作するデジタル論理検知検出器Y_0と、
ただ一つの少なくともｋ文字長さの▲○＋▼加算器と、ｋ文字の被乗数を受け入れるように動作する▲○＋▼加算装置と、順番に被乗数値を▲○＋▼アキュムレーターデバイス内に入れ替え、順番にＢレジスタから乗数値を受け入れるように動作する▲○×▼乗算装置と、第一段階においてｋの最初に発生するゼロ文字を出力させるように動作する乗数と同時に生成した「即座の」予測値とを有し、各々の有効マシンサイクルで少なくとも一つの指定された被乗数が▲○＋▼加算装置に▲○＋▼加算される、少なくともｋ文字の入力被乗数のためのモジュラー乗算装置と、
全ゼロ文字列の値である第一被乗数、被乗数A_iである第二被乗数、モジュラスのN_0セグメントである第三被乗数のうちの一つ若しくは二つの被乗数からなる、順番に▲○＋▼アキュムレーションデバイス内に入れ替えられる被乗数値と、
▲ｌ▼ビットのｋ文字直列入力Y_0乗数値を予測する装置と、
第一段階において乗算装置に順番に入力されるＢオペランドである乗数値と、同時に、最初に発生するゼロを出力させるための「即座の」予測されたｋ文字列であるY_0からなる第二乗数値と、
▲○＋▼アキュムレーションデバイスであって、被乗数が該アキュムレーションデバイスへ▲○＋▼加算されると同時に値を出力するように動作する▲○＋▼アキュムレーションデバイスと、
第二段階において▲○＋▼アキュムレーションデバイスからの最終のモジュラー▲○×▼乗算の結果を出力するように動作する出力転送機構とからなるマイクロエレクトロニクス装置。
【請求項２】
▲○＋▼アキュムレーションデバイスへの▲○＋▼加算が、各々の新たな直列にロードされた高位乗数により始まる請求項１記載のマイクロエレクトロニクス装置。
【請求項３】
乗数が、入力されたＢ文字と対応する入力されたY_0文字の両方がゼロの場合は、▲○＋▼アキュムレーションデバイスへの▲○＋▼加算はされず、
入力されたＢ文字が１であり、対応するY_0がゼロである場合は、A_i被乗数のみが▲○＋▼加算され、
入力されたＢ文字がゼロであり、対応するY_0が１である場合は、モジュラスＮのみが▲○＋▼加算され、
入力されたＢ文字と対応するY_0文字の両方が１である場合は、被乗数A_iと共にモジュラスＮが▲○＋▼加算されるように動作する請求項１記載のマイクロエレクトロニクス装置。
【請求項４】
被乗数値A_i，Ｎを二つの指定されたプリロードバッファにプリロードし、これらの値を

第三プリロードバッファに▲○＋▼加算するように動作して、各々の被乗数値を別々に▲○＋▼加算する必要をなくす請求項１記載のマイクロエレクトロニクス装置。

【請求項５】

乗数値が入力において直列単一文字であり、▲○＋▼アキュムレーションデバイスの出力が直列単一文字出力であり、Y_0検出装置が一回のクロックで一文字のみ予測するように動作する請求項１記載のマイクロエレクトロニクス装置。

【請求項６】

▲○＋▼アキュムレーションデバイスが２を法としてＸＯＲ加算／減算の計算を実行し、加算成分及び減算成分中の全ての桁上げビットが無視されることにより、計算における桁溢れや更に限定する有益性が排除される請求項１記載のマイクロエレクトロニクス装置。

【請求項７】

全ての桁上げ入力がゼロ（▲Ｓ▼＝０を意味する）まで実行不能とされ、概して、多項式ベースの乗算を実行するように動作する請求項１記載のマイクロエレクトロニクス装置。

【請求項８】

▲Ｓ▼がＧＦ（2^q）において計算する回路方程式中の要素に影響するゼロに等しく、▲Ｓ▼が除かれた回路を示し、▲○＋▼で表される全ての加算器及び減算器が縮小されて、２を法とした加算／減算要素の排他的論理和を求める請求項１記載のマイクロエレクトロニクス装置。

【請求項９】

ｋの最初に発生するゼロが次のY_0文字を予測する際に、

ｉ．　A_iレジスタの右側文字にＢストリームのB_d文字を掛けた▲ｌ▼ビット毎の$2^{▲l▼}$を法とする▲○×▼乗算の結果の▲ｌ▼ビットのS_{out}ビット　$A_0 \cdot B_d \bmod 2^{▲l▼}$、

ｉｉ．　▲○＋▼アキュムレーションデバイスから最初に発生する桁上げ文字▲Ｓ▼（CO_0）、

ｉｉｉ．　▲○＋▼アキュムレーションデバイスの右側の発生セルからの次に発生する桁上げ文字からの▲ｌ▼ビットのS_{out}文字SO_1、

ｉｖ．　N_0モジュラス被乗数レジスタにおける右側文字の負の逆数である▲ｌ▼ビットのJ_0値、

の四つの数量によって調整される装置から発し、

前記$A_0 \cdot B_d \bmod 2^{▲l▼}$、▲Ｓ▼（$CO_0$）、$SO_1$の値が、共に文字に▲○＋▼加算される文字であり、▲ｌ▼ビットのゼロを発する有効なY_0ゼロ強制予測文字を出力するために「即座に」J_0文字によって乗算される請求項１記載のマイクロエレクトロニクス装置。

【請求項１０】

多項式ベースのオペランドでの▲○×▼乗算が逆モードで実行され、右側のＭＳ文字から左側のＬＳ文字を乗算し、モンゴメリ型寄生関数なしでモジュラー縮小された▲○×▼乗算を実行するように動作する請求項１記載のマイクロエレクトロニクス装置。

【請求項１１】

プリロードバッファが直列供給され、被乗数値が複数の記憶装置から即座にプリロードバッファにプリロードされる請求項１記載のマイクロエレクトロニクス装置。

【請求項１２】

プロセスにおいて用いられる有限体に適するように、▲○＋▼が加算を定義し、▲○×▼が乗算を定義し、

Y_0検出器が▲○＋▼アキュムレーションデバイスにおいて▲○＋▼加算にモジュラスを▲○＋▼加算する必要性を検出ように動作する時に、最初に発生する出力文字がゼロであるように、▲ｌ▼ビットの▲○＋▼加算器回路を経由して、更なるｎビットのレジスタＳから発生する先の値が▲○＋▼アキュムレーションデバイスの出力値に▲○＋▼加算され、Y_0検出器が、次に順番に加算した文字$A_0 \cdot B_d \bmod 2$、▲Ｓ▼（CO_0）、SO_1、S_d、▲Ｓ▼（CO_Z）を利用して、▲ｌ▼ビットのJ_0値により即座に▲○×▼

乗算される有限フィールドであるように▲○＋▼加算される文字の合成を検出するように動作する請求項１記載のマイクロエレクトロニクス装置。

【請求項１３】

▲ｌ▼＝１で、ハードウェアを追加することなく、$Ｊ_0$が暗黙的に１であり、$Ｊ_0$▲○×▼乗算が暗黙的である請求項１記載のマイクロエレクトロニクス装置。

【請求項１４】

コンパレータが、最初の右側の発生するｋゼロ文字が無視されるＧＦ（ｐ）において機能し、▲○×▼モジュラー乗算装置からの有限フィールド出力を検知するように動作し、その出力がモジュラスＮより大きいことにより、モジュラー縮小をを調整するように動作し、前記値が、乗算装置からの出力ストリームの行き先であるメモリレジスタから出力されることにより、より小さな積の値に第二の記憶装置を割り当てる必要がなくなる請求項１記載のマイクロエレクトロニクス装置。

【請求項１５】

ＧＦ（2^q）における▲○×▼モジュラー乗算で、前記マイクロエレクトロニクス装置が外部で予め計算された▲ｌ▼ビットより大きなゼロ強制因子なしで乗算するように動作する請求項１記載の装置。

【請求項１６】

Ａオペランド値又はＢオペランド値のいずれかをゼロに再設定し、部分的な結果値$Ｓ_0$を１に設定することにより、$Ｊ_0$定数を計算するように動作する請求項１記載の方法。

【請求項１７】

ＡにＢを掛けてモジュラスがＮとなる出力ストリームを生成するように動作する、整数Ａ及びＢのインタリーブされた有限体▲○×▼モジュラー乗算を実行するマイクロエレクトロニクス装置であって、モジュラスオペランドレジスタにおける文字の数ｎがｋよりも大きく、▲○×▼乗算プロセスが反復で実行され、各々のインタリーブされた反復で▲○×▼乗算装置に入力されるオペランドは、モジュラスＮ、乗数Ｂ、予め計算された部分的な結果Ｓ、被乗数Ａのｋ文字列セグメントからなり、セグメントは$Ａ_0$文字列セグメントから$Ａ_{m-1}$文字列セグメントまで進行し、各々の反復の結果は次の順番の一時的結果Ｓに▲○＋▼加算され、反復結果の最初に発生する文字はゼロであって、

各々がオペランドを保存、出力でき、夫々乗算値、部分的な結果値、Ｎで示されるモジュラスを保存するように動作する第一主メモリレジスタＢ、第二主メモリレジスタＳ、第三主メモリレジスタＮと、

反復▲○×▼乗算プロセスの間、順番に複数の被乗数値の一つ又は二つを▲○＋▼アキュムレーションデバイスに▲○＋▼加算し、順番に第一の値であるＢレジスタからの入力と、各々の反復において最初の右側ゼロを出力させる乗数である第二の値である「即座の」予測値$Ｙ_0$の入力と、Ｎレジスタからの第三の値であるモジュラスの入力とを乗数として受け入れるように動作するモジュラー乗算装置と、

少なくとも、Ａ，Ｂ，Ｎレジスタ資源からの値を順番に受け入れ、続いて被乗数ゼロ強制値$Ｙ_0$をも受け入れるように動作する被乗数並列レジスタと、

第一段階の間乗数となるように動作し、第二段階の間被乗数となるように動作する二進列（ｂｉｎａｒｙ　ｓｔｒｉｎｇ）を生成するように動作する最初に発生するゼロ強制Ｙ０検出装置と、

第一にゼロ、第二に被乗数Ａのｋ文字列セグメントである$Ａ_i$、第三にモジュラスＮの最初に発生するｋ文字である$Ｎ_0$からなる、第一段階で▲○＋▼アキュムレーションデバイスに入れ替える被乗数値と、

次の反復で並列の結果を生成するために、アキュムレーションデバイスから発する値に加算される先の反復の結果である一時的結果値Ｓと、

第一にゼロ、第二に第一段階から残存するオペランド$Ａ_i$、第三に第一段階で予測された$Ｙ_0$値である、第二段階でアキュムレーションデバイスに順番に入力される被乗数値と、Ｂオペランドの最初に発生する文字列セグメントである、最初に発生する文字列$Ｂ_0$であり、第二段階でプリロードされた被乗数バッファに生成すると同時に文字毎にロードされ

る予測されたＹ$_0$文字列からなる第二の乗数値と並行して乗算する第一段階で乗算装置に入力される乗数値と、
夫々、Ｂで表されるＢオペランドからの左側ｎ－ｋ文字の値と、Ｎで表されるＮモジュラスの左側ｎ－ｋ文字である、第二段階の間に装置に入力される二つの乗数値と、
アキュムレーションデバイスに残存する結果値の左側セグメントを結果レジスタに転送するように最終段階で動作する乗算フラッシュアウト装置とからなるマイクロエレクトロニクス装置。

【請求項１８】
ＭＳ文字からＬＳ文字まで乗算する逆モードにおいて多項式ベースのオペランドで乗算が実行され、モンゴメリ型寄生関数なしでモジュラー縮小を実行するように動作する請求項１７記載のマイクロエレクトロニクス装置。

【請求項１９】
被乗数の最初に発生する値、Ｂ乗数の現在の入力、アキュムレーションデバイスからの桁上げ値、アキュムレーションデバイスからの加算値、先に計算された部分的な結果からの現在の値、先の部分的な結果にアキュムレーションデバイスからの結果を加算する加算器からの桁上げ値を用いてＹ$_0$値を予測するように動作する装置。

【請求項２０】
ｋの最初に発生するゼロが、次の順番のＹ$_0$文字を予測する際に、
ｉ．　Ａ$_i$レジスタの右側文字にＢストリームのＢ$_d$文字を掛けた▲ｌ▼ビット毎の$2^{▲l▼}$を法とする乗算の結果の▲ｌ▼ビットのＳ$_{out}$ビット　$A_0 \cdot B_d \bmod 2^{▲l▼}$、
ｉｉ．　アキュムレーションデバイスから最初に発生する桁上げ文字▲Ｓ▼（ＣＯ$_0$）、
ｉｉｉ．　アキュムレーションデバイスの右側の発生セルからの第二からの▲ｌ▼ビットのＳ$_{out}$文字ＳＯ$_1$、
ｉｖ．　Ｓストリームからの次の順番の文字値Ｓ$_d$、
ｖ．　Ｚ出力全加算器からの▲ｌ▼ビット桁上げ文字▲Ｓ▼（ＣＯ$_Z$）、
ｖｉ．　Ｎ$_0$モジュラス被乗数レジスタにおける右側文字の負の逆数である▲ｌ▼ビットのＪ$_0$値、
の六つの数量によって調整される装置から発し、
前記$A_0 \cdot B_d \bmod 2^{▲l▼}$、▲Ｓ▼（ＣＯ$_0$）、ＳＯ$_1$、Ｓ$_d$の値が、共に文字に▲○＋▼加算される文字であり、▲ｌ▼ビットのゼロ文字列を発するように有効なＹ$_0$ゼロ強制予測文字を出力するために「即座に」Ｊ$_0$文字によって▲○×▼乗算される請求項１９記載の装置。

【請求項２１】
出力結果をモジュラスＮと比較するように動作する少なくとも一つのセンサからなり、その機構が、結果レジスタの出力に第二の減算器を動作させることにより、出力された結果値と限定された合同であるモジュラー縮小された値を出力するように動作し、より小さな結果に第二の記憶装置を割り当てる必要を排除する請求項１７記載のマイクロエレクトロニクス装置。

【請求項２２】
最初の値の一つがプリロードバッファにロードされるのと並行して、二つの被乗数の加算である値が、少なくとも一つのｋ文字メモリ手段レジスタでプリロード文字バッファにロードされる請求項１７記載のマイクロエレクトロニクス装置。

【請求項２３】
一連のインタリーブされた▲○×▼モジュラー乗算及び平方を実行するように動作し、三つの自然数乗算演算と同等のものを並行して実行する、結果がべき算である単一のアキュムレーションデバイス及び予測ゼロ強制機構を有する装置。

【請求項２４】
次の順番に用いられる被乗数が、即座にプリロードレジスタバッファ手段にプリロードされる請求項１７記載のマイクロエレクトロニクス装置。

【請求項２５】
二つの被乗数の合計である値が、その一つの値がそのプリロードバッファにロードされるのと並行して少なくとも一つのｋ文字レジスタに加算される請求項１７記載のマイクロエレクトロニクス装置。
【請求項２６】
装置のバッファ及びレジスタは外部メモリー資源からの値をロードされるように動作し、前記バッファ及びレジスタは計算の間に外部メモリー資源にアンロードされるように動作し、オペランドの最大サイズが利用できるメモリー手段に依存する請求項１７記載のマイクロエレクトロニクス装置。
【請求項２７】
メモリーレジスタ手段が、概して、直列単一文字入力／直列単一文字出力、並列少なくともｋ文字入力／並列少なくともｋ文字出力、直列単一文字入力／並列少なくともｋ文字出力、並列少なくともｋ文字入力／直列単一文字出力である請求項１７記載のマイクロエレクトロニクス装置。
【請求項２８】
乗算型反復の最終段階の間、乗数入力が桁上げ保存累算メモリーの左側セグメントをフラッシュアウトするように動作するゼロ文字である請求項１７記載のマイクロエレクトロニクス装置。
【請求項２９】
次の順番の被乗数が、即座にプリロードメモリーバッファにプリロードされる請求項１７記載のマイクロエレクトロニクス装置。
【請求項３０】
被乗数値が、中央記憶装置から即座にプリロードバッファにプリロードされる請求項１７記載のマイクロエレクトロニクス装置。
【発明の詳細な説明】
【０００１】
（技術分野）
本発明は暗号の共同処理用周辺装置（ｃｒｙｐｔｏｇｒａｐｈｉｃ ｃｏ－ｐｒｏｃｅｓｓｉｎｇ ｐｅｒｉｐｈｅｒａｌｓ）を速めるための演算装置に関するものであり、更に、楕円曲線及びＲＳＡ型の計算のために設計されたモジュラー演算公開鍵暗号コプロセッサ（ｍｏｄｕｌａｒ ａｒｉｔｈｍｅｔｉｃ ｐｕｂｌｉｃ ｋｅｙ ｃｒｙｐｔｏｇｒａｐｈｉｃ ｃｏｐｒｏｃｅｓｓｏｒ）における整数体の計算範囲及び直列の入力オペランド幅を拡張する多項式ベースで素数フィールド（ｐｒｉｍｅ ｎｕｍｂｅｒ ｆｉｅｌｄ；素体）の演算のための加速された演算装置に関するものである。
【０００２】
（背景技術）
計算装置のセキュリティ強化及び機能高速化は、出願人による米国特許第５７４２５３０号明細書（以下Ｐ１と称す）、第５５１３１３３号明細書、第５４４８６３９号明細書、第５２６１００１号明細書、第５２０６８２４号明細書、公開された国際出願第ＰＣＴ／ＩＬ９８／００１４８号（国際公開第９８／５０８５１号パンフレット）、対応する米国特許出願公開第０９／０５０９８号明細書（以下Ｐ２と称す）、オニスチャック（Ｏｎｙｓｚｃｈｕｋ）らの米国特許第４７４５５６８号明細書、オムラらの米国特許第４５８７７６２７号明細書、及び出願人の米国特許出願公開第０９／４８０１０２号明細書に記載されているが、本願はこれらの開示を参考文献として取り込んでいる。
出願人の米国特許第５２０６８２４号明細書には、多項式ベースの乗算及び平方を実行する初期の演算装置が示されているが、素数フィールドにおける演算（ｏｐｅｒａｔｉｏｎ）を行うことはできず、多項式ベースの計算においてインタリーブする（ｉｎｔｅｒｌｅａｖｅ）ように設計されていない。
更なる解析が、パール・シー（Ｐａａｒ， Ｃ．），エフ・フライシュマン（Ｆ． Ｆｌｅｉｓｃｈｍａｎｎ），ピー・ソリアーロドリゲス（Ｐ． Ｓｏｒｉａ－Ｒｏｄｒｉｇｕｅ

ｚ）著、「素数ではない数（合成数）を有するガロア体における公開鍵アルゴリズムのための高速演算（Ｆａｓｔ Ａｒｉｔｈｍｅｔｉｃ ｆｏｒ Ｐｕｂｌｉｃ－Ｋｅｙ Ａｌｇｏｒｉｔｈｍｓ ｉｎ Ｇａｌｏｉｓ Ｆｉｅｌｄｓ ｗｉｔｈ Ｃｏｍｐｏｓｉｔｅ Ｅｘｐｏｎｅｎｔｓ）」、コンピュータに関するＩＥＥＥ論文集、第４８巻、第１０号、１９９９年１０月（ＩＥＥＥ Ｔｒａｎｓａｃｔｉｏｎｓ ｏｎ Ｃｏｍｐｕｔｅｒｓ，ｖｏｌ．４８，Ｎｏ．１０，Ｏｃｔｏｂｅｒ １９９９）（以下パールと称す）で、多項式ベースの演算における拡大体を利用するアプローチによりなされている。
ダブリュー・ウエズレー・ピーターソン（Ｗ．Ｗｅｓｌｅｙ Ｐｅｔｅｒｓｏｎ），イー・ジェー・ウェルドン・ジュニア（Ｅ．Ｊ．Ｗｅｌｄｏｎ Ｊｒ．）著、「エラー訂正コード（Ｅｒｒｏｒ－Ｃｏｒｒｅｃｔｉｎｇ Ｃｏｄｅｓ）」、第二版、エムアイティープレス（ＭＩＴ Ｐｒｅｓｓ）、ケンブリッジ、マサチューセッツ州（Ｍａｓｓ．）、１９７２年、ｐ．１７４－１７９（以下ピーターソンと称す）、では、多項式ベースの剰余表現ＧＦ（2^q）における除算実行回路を示している。
ピーターソンの回路は、乗数がまさにモジュラス（ｍｏｄｕｌｕｓ；剰余）の長さである装置においてのみ利用され得る。
典型的には、それは本装置より２倍の長さがある装置を必要とし、コンパクトなインプリメンテーション（ｉｍｐｌｅｍｅｎｔａｔｉｏｎ；実装）には経済的ではない。
また、マルチビット文字（ｍｕｌｔｉｂｉｔ ｃｈａｒａｃｔｅｒ；マルチビットキャラクター）のＹ０を決定する従来装置を備えていないので、インタリーブされた（ｉｎｔｅｒｌｅａｖｅｄ）インプレメンテーションにも使用できず、▲ｌ▼が１より長い場合は有効でない。

【０００３】
クヌース『ディー・クヌース（Ｄ．Ｋｎｕｔｈ）著、「コンピュータプログラミングの技法（Ｔｈｅ ａｒｔ ｏｆ ｃｏｍｐｕｔｅｒ ｐｒｏｇｒａｍｍｉｎｇ）」、第２巻、準数値算法（Ｓｅｍｉｎｕｍｅｒｉｃａｌ ａｌｇｏｒｉｔｈｍｓ）、アディソン・ウェズリー（Ａｄｄｉｓｏｎ－Ｗｅｓｌｅｙ）、リーディング、マサチューセッツ州、１９８１年、ｐ．４０７』は、多項式ベースの除算において、単一の▲ｌ▼ビット文字に通常の除算プロセスを使用する旨を暗示しているのに対し、我々は商における次の文字を予測する方法を仮定でき、本発明が論理構成（ｌｏｇｉｃ ｃｏｎｆｉｇｕｒａｔｉｏｎ）を利用して決定論的に商の次の文字を予測する方法を開示する。

【０００４】
（発明の開示）
本発明の目的は、様々な基数での乗算器においてインタリーブされたモジュラー乗算（ｉｎｔｅｒｌｅａｖｅｄ ｍｏｄｕｌａｒ ｍｕｌｔｉｐｌｉｃａｔｉｏｎ）及び縮小（ｒｅｄｕｃｔｉｏｎ）を同時に実行するために同じ予測法（ａｎｔｉｃｉｐａｔｉｎｇ ｍｅｔｈｏｄ）を利用して、多項式ベース（ｐｏｌｙｎｏｍｉａｌ ｂａｓｅｄ）及び素数ベースの数値体フィールド（ｎｕｍｂｅｒ ｆｉｅｌｄ）における大数計算（ｌａｒｇｅ ｎｕｍｂｅｒ ｃｏｍｐｕｔａｔｉｏｎｓ）を実行するように機能するマイクロエレクトロニクス専用の算術演算装置を提供することである。

【０００５】
本発明の更なる目的は、非常に大きな整数でのモジュラー及びノーマルな（自然数であって負でない整数の体）乗算、除算、加算、減算、べき算を実行するためのコンパクトなマイクロエレクトロニクス専用の算術論理演算装置に関するものである。
簡略化された多項式ベースの乗算及び平方のためにモンゴメリ法及び逆フォーマット法（ｒｅｖｅｒｓｅｄ ｆｏｒｍａｔ ｍｅｔｈｏｄ）の両方を利用するモジュラー乗算及び平方に言及する場合、スーパースカラ（ｓｕｐｅｒｓｃａｌａｒ）モジュラー算術コプロセッサ、ＳＭＡＰ、ＭＡＰ、若しくはＳｕｐｅｒＭＡＰ（商標）のような装置の特定部分が参照され、また、出願人の１９９８年３月３１日に提出された米国特許出願公開第０９／０５０９５８号明細書及び２０００年１月１０日に提出された米国一部継続出願公開第０９／４８０１０２号明細書に存在する改良に関するものが参照される。

【０００６】
本願明細書において記載される本発明の好ましい実施例は、携帯型スマートカードで、概して一般に普及している磁気帯付クレジット及びバンクカードと形状及びサイズが同一の公開鍵暗号アプリケーションのためのモジュラー計算演算子を提供する。
出願人の米国特許第５５１３１３３号、５７４２５３０号明細書の技術及び出願人の上述した出願の技術による同様のスマートカードは、コンピュータ、データベース、商取引、軍事用取引、家庭内取引におけるデータフローを調整し保全するため及びスクランブルされた有料テレビプログラムを復号する等のための重要な装置、同様の応用のための端末装置としての重要な装置へのアクセスを制御するための新世代の公開鍵暗号装置において利用される。
典型的には、これらの装置は、コンピューターファクシミリ端末、ドアロック、自動販売機等にも組み込まれている。
【０００７】
好ましい構成（ａｒｃｈｉｔｅｃｔｕｒｅ；アーキテクチャ）は、多数のマイクロコントローラ及びデジタル信号処理装置並びに減少した命令セット計算設計（ｒｅｄｕｃｅｄ ｉｎｓｔｒｕｃｔｉｏｎ ｓｅｔ ｃｏｍｐｕｔａｔｉｏｎａｌ ｄｅｓｉｇｎ）に集積されて動作する装置である一方で、該装置はホストの処理装置と並列に演算する。
【０００８】
この実施例は、好ましくは、本質的に２～３つの乗算装置の役割を果たすたった一つの乗算装置を利用するものであり、該乗算装置は、出願人の米国特許第５５１３１３３号明細書に記載され、米国特許出願公開第０９／０５０９５８号明細書及び国際出願ＰＣＴ／ＩＬ９８／００４８で改良された構成に、基本的には近似している。
従来のマイクロエレクトロニクス技術を利用して、本発明の装置は、スマートカード超小形電子回路上へ記憶装置と共に制御ユニットを集積化される。
【０００９】
多項式ベースの体（ｆｉｅｌｄ）におけるハードウェア実装と素数からなる体（ｆｉｅｌｄ：フィールド）におけるハードウェア実装との間の主な差異は、多項式ベースの加算及び減算が、ＬＳからＭＳへの桁上げ信号の伝搬のない単純なＸＯＲ論理演算であるということである。
従って、ハードウェア実装における隣接するセル間で相互作用がなく、減算と加算が同一の手順（ｐｒｏｃｅｄｕｒｅ）となる。
著者が知っている最初の公表は、１９９４年にイタリアのペルージャで開催されたユーロクリプト・コンフェレンス・ランプ・セッション（Ｅｕｒｏｃｒｙｐｔ Ｃｏｎｆｅｒｅｎｃｅ Ｒｕｍｐ Ｓｅｓｓｉｏｎ）でフォンダツィオーネ・ウーゴ・ボルドーニ（Ｆｏｎｄａｚｉｏｎｅ Ｕｇｏ Ｂｏｒｄｏｎｉ）のマルコ・ブッチ（Ｍａｒｃｏ Ｂｕｃｃｉ）による短い講演であるが、この時でさえ当業者であるエンジニアの間にこの構成はよく知られていた。
【００１０】
Ｐ１，Ｐ２に記載された出願人の先の装置は、概して、ＧＦ（ｐ） 体における楕円曲線暗号プロトコルを効率的に計算するために備えられたものである。
本発明においては、ＧＦ（2^q） 体における使用のために、我々は、多項式体における隣接する二進数ビット間で相互作用がないので、モンゴメリ関数やモンゴメリ寄生要素（Ｍｏｎｔｇｏｍｅｒｙ ｐａｒａｓｉｔｅ）を導入することなくスーパースカラ乗算装置で縮小（Ｒｅｄｕｃｔｉｏｎ）及び乗算を同時に実行して、計算が効率的に行われることを示す。
計算機が好ましくは最上位の部分積から始めて、ＧＦ（2^q）における乗算が実行される。
縮小は、ＭＳの１をゼロに再設定するのに必要なだけモジュラスを加算することによって実行される。
これらの加算において、桁上げはないので、結果は自動的にモジュラー方式で（ｍｏｄｕ

ｌａｒｌｙ）縮小される。
本発明において、多項式計算は同じ構成を利用して実行されるが、そこにおいて、ＧＦ（2^q）ではオペランドは最初にＭＳ文字において供給され、全ての内部桁上げ信号はゼロにされる。
ＧＦ（ｐ）計算は好ましくはＰ１及びＰ２に記載のように実行され、そこにおいて、ＬＳ文字は最初に処理され、ＭＳ文字は最後に処理される。
【００１１】
該構成は、直列乗算器が▲ｌ▼ビットの幅広文字（▲ｌ▼ ｂｉｔ ｗｉｄｅ ｃｈａｒａｃｔｅｒｓ）であり、各々のクロックで、▲ｌ▼ビット文字が桁上げ保存累算器ＣＳＡから発生される（ｂｅ ｅｍｉｔｔｅｄ；エミットされる）という点で、潜在的により速い進行（ｐｒｏｇｒｅｓｓｉｏｎ；数列）を可能にするように拡張されてきた。
このことは、単一ビットの広さのバスで2^xを法とした奇数の反転（ｉｎｖｅｒｓｉｏｎ；逆転）も奇数であるという点で予測プロセス（Y_0）をいくらか複雑にし、被乗数（J_0）の最下位ビットが常に１であった。
しかし、両方の数値体で、我々が、ｋ文字がゼロの文字列を出力することのみが目的であることを覚えており、Y_0関数をゼロ強制ベクトル（ｚｅｒｏ ｆｏｒｃｉｎｇ ｖｅｃｔｏｒ；ゼロフォーシングベクトル）としてのみ見なす場合は、桁上げをなくする（制限する）と仮定すると、縮小プロセスは同一である。
【００１２】
本発明は、米国特許第５５１３１３３号明細書に記載されたプロセスに関連する計算的、論理的、算術的な新規な特徴を有する従来のデジタルプロセッサの周辺機器であるデジタル装置の構成を提供しようとするものでもある。
【００１３】
好ましくは、概して各々の演算で大スケールの乗算及び除算を実行する古典的な乗算／除算装置により実行されるのと同様の演算数で除算することなくモジュラーべき算を実行するために、並列プロセス及びハードウェアアーキテクチャが提供される。
本発明の好ましい実施例の特別の特徴は、より大きなスケールの予測ゼロ強制関数（ａｎｔｉｃｉｐａｔｏｒｙ ｚｅｒｏ ｆｏｒｃｉｎｇ ｆｕｎｃｔｉｏｎ）の並列化（ｃｏｎｃｕｒｒｅｎｃｙ；同時進行）、数値体の拡張、及び安全な通信のためのこの種のユニットを集積化する能力である。
【００１４】
本発明の好ましい実施例によって認識される利点は、直列プロセスの同期シーケンスから生じる。
これらのプロセスは、ｎの有効クロックサイクル（ｅｆｆｅｃｔｉｖｅ ｃｌｏｃｋ ｃｙｃｌｅ）において一つの多重化されたｋ文字の直列／並列乗算器を用いて、同時に（並列的に）ｎ文字のオペランドで三つの乗算演算を成し遂げるように組み合わされ、結果の左側の最後のｋ文字は乗算装置の出力バッファに存在する。
この手順は、モンゴメリによって記載されるように、素数からなる体（ｆｉｅｌｄ：フィールド）で両体における三つの乗算計算と同等のものを得ることができ、ＧＦ（2^q）において二つの乗算及び一つの除算プロセスと同等のものを得ることができる。
【００１５】
ＳｕｐｅｒＭＡＰへのオペランドのロード及びオペランド数値の即座の（ｏｎ ｔｈｅ ｆｌｙ；高速の）検出と、即座のプリローディング（ｐｒｅｌｏａｄｉｎｇ）及び使用されたオペランドの隣の同時の加算とを同期することにより、装置は決定論的に（ｉｎ ｄｅｔｅｒｍｉｎｉｓｔｉｃ ｆａｓｈｉｏｎ）計算を実行するように機能する。
次の反復の平方シーケンスで三つの初めのｋ文字変数を即座にプリロードする全ての乗算及びべき算回路が好ましくは加わる。
好ましくは検出装置が備えられ、ｋの有効クロックサイクル待ち状態を排除して、三つのオペランドのうちの二つだけが次の反復の被乗数に選択される。
条件付き分岐（ｃｏｎｄｉｔｉｏｎａｌ ｂｒａｎｃｈ）が、局所検出及び補正装置（ｌ

ｏｃａｌ ｄｅｔｅｃｔｉｏｎ ａｎｄ ｃｏｍｐｅｎｓａｔｉｏｎ ｄｅｖｉｃｅ）と置換されることにより、単純な制御機構での基準を備えることとなる。
ここに記載された基本的な演算は、典型的には、グレッセル（Ｇｒｅｓｓｅｌ）らに付与された米国特許第５５１３１３３号明細書に記載された装置又はフランスのルーセ（Ｒｏｕｓｓｅｔ）にあるエスティマイクロエレクトロニクス社（ＳＴＭｉｃｒｏｅｌｅｃｔｒｏｎｉｃｓ）による商品名ＳＴ１９－ＣＦ５８の装置を用いて、ＧＦ（ｐ）において決定論的な時間で実行される。

【００１６】
オペランドが全演算に亘って外部の揮発性記憶装置へロードされ装置内に保存されるので、本発明の装置は、ほとんどの演算について外部の揮発性記憶装置に特に依存する。
装置は、好ましくは、ＭＡＰがその大数計算を実行する間に、単純なロード及びアンロード並びに装置へのコマンドのシーケンスを実行するために、装置が追加されたＣＰＵを利用する。
現在、スマートカードに応用されている大数は、１２８ビットから２０４８ビットまでの自然数の応用に及ぶ。
べき算処理時間は仮想的にＣＰＵから独立しており、これによりＣＰＵを制御する。
実際には、ＣＰＵに装置を追加する時に、設計上の変更は典型的には不必要である。
ハードウェアデバイスは自己内蔵型であって、好ましくはいかなるＣＰＵバスにも追加される。

【００１７】
一般的に、本発明は、大きな整数の演算処理にも関するものである。
これらの大数は典型的には、（ゼロ以上の）整数の自然数体（ｎａｔｕｒａｌ ｆｉｅｌｄ：自然数フィールド）におけるものか、又は、素数のガロア体ＧＦ（ｐ）、合成数モジュラス（ｃｏｍｐｏｓｉｔｅ ｐｒｉｍｅ ｍｏｄｕｌｕｓ）、多項式ベースの数のガロア体ＧＦ（2^{q}）におけるものである。
とりわけ、本発明の好ましい実施例は、大数のモジュラー演算及びべき算を実行することができる装置を提供しようとするものである。
このような装置は、素数フィールドにおいて次第に大きなオペランドで機能し、現存のモジュラー演算コプロセッサでは効率的に実行できず、ソフトウェア構築でも、しっかりと実行できないような公開鍵暗号認証（Ｐｕｂｌｉｃ Ｋｅｙ Ｃｒｙｐｔｏｇｒａｐｈｉｃ ａｕｔｈｅｎｔｉｃａｔｉｏｎ）及び暗号化プロトコルの演算を実行することに適している。
好ましくは、より小さい桁である整数に関して、同じ全体的な構成が楕円曲線の実装（ｉｍｐｌｅｍｅｎｔａｔｉｏｎ）において利用される。
新規な乗算の逆モード法（ｒｅｖｅｒｓｅ ｍｏｄｅ ｍｅｔｈｏｄ；リバースモード法）を利用して、ゼロを生成することにより計算の際に寄生的な2^{-n}因子の負担がないので、多項式演算は有利である。

【００１８】
該構成は、直列単一文字のバス（ｓｅｒｉａｌ ｓｉｎｇｌｅ ｃｈａｒａｃｔｅｒ ｂｕｓ）を広げることによって、即ちより大きな基数の使用によって可能になった標準的な小さなオペランドの演算を可能する一方で、大きなオペランド整数の演算のモジュラー実行をも可能にする。
概して、これは、計算を高速化し、ＳｕｐｅｒＭＡＰを実行するためのシリコン領域を減少し、一般に普及しているデジタル信号プロセッサ（ＤＳＰ）で互換性がある長さの装置を生成することに有効である。

【００１９】
奇数の素体及び合成数体（ｃｏｍｐｏｓｉｔｅ ｆｉｅｌｄ；合成体）におけるモジュラー乗算について、Ａ及びＢは、夫々被乗数及び乗数と定義され、Ｎはモジュラー演算におけるモジュラスと定義される。
Ｎは、Ａ又はＢより概して大きい。

Ｎは、モジュラスの値が保存される合成レジスタ（ｃｏｍｐｏｓｉｔｅ ｒｅｇｉｓｔｅｒ；複合レジスタ）を意味する。
Ｎは、若干の場合において、概してＡより小さい。
Ａ，Ｂ，Ｎは概してｎ文字長さであり、その文字は概して１～８ビットの長さである。
ｋは、乗算装置のサイズ（セル数）により定められる群のサイズにおける▲ｌ▼ビットの文字の数である。
同様に、多項式ベースのＧＦ（２^{q}）計算において、モジュラスＮは、ＭＳビットが１である（モニック（ｍｏｎｉｃ）である）ｎビットの長さであり、Ａ，Ｓ，Ｂのオペランドも適切に縮小された場合はｎビットの長さである。
ＧＦ（２^{q}）計算の結果がモニックである場合、前記結果の値からモジュラスで排他的論理和を求める（ＸＯＲ）ことにより、好ましくはＭＳゼロを有する値へ「縮小される（ｂｅ ｒｅｄｕｃｅｄ）」。
好ましい実施例において、ＧＦ（２^{q}）の最初の有効ビットが逆モードにおいて形成されるので、ＭＡＰはそのビットが１であるか否かを検知する（ｓｅｎｓｅ）ことができ、好ましい縮小を実行できる。
【００２０】
素体において、≡、或いは、若干の場合の＝は、例えば１６ ≡ ２ ｍｏｄ ７のように、モジュラー数の合同を意味するように利用される。
１６を７で割った場合、２が剰余であるので、１６は７を法として（ｍｏｄｕｌｏ ７）２と「合同である」と言う。
Ｙ ｍｏｄ Ｎ ≡ Ｘ ｍｏｄ Ｎの場合、Ｙ，Ｘの両方がＮより大きくてもよいが、正のＸ，Ｙについては、剰余は同一である。
負の整数Ｙの合同がＹ＋ｕ・Ｎ（Ｎはモジュラス）であることにも留意する必要があり、Ｙの合同がＮ未満となるような場合は、ｕは正の結果を与える最も小さい整数である。
【００２１】
ＧＦ（２^{q}）において、加算及び減算が同一であり、通常の計算は、概して実質的な桁溢れを残さないので、合同はより単純である。
Ｎ＝１１０１，Ａ＝１００１について、左側のＡのＭＳビットが１であるので、我々は２を法とした演算、Ａ ＸＯＲ Ｎ＝１００１ ＸＯＲ １１０１＝０１００を利用することにより、ＡからＮを縮小（「減算」）しなければならない。
【００２２】
円記号¥は、ＧＦ（ｐ）において特に有用な限定された意味（ｌｉｍｉｔｅｄ ｓｅｎｓｅ）における合同を意味するように用いられる。
ここに記載されるプロセスの間、値はしばしば所望の値か、若しくは所望の値にモジュラスを加えた値と等しいかのいずれかである。
例えばＸ ¥ ２ ｍｏｄ ７の場合である。
Ｘは、２又は９と等しくなり得る。
Ｘは、７を法として２が限定された合同（ｌｉｍｉｔｅｄ ｃｏｎｇｒｕｅｎｃｅ ｔｏ ２ ｍｏｄ ７）を有すると定義される。
円記号が$B^{¥}$のように上付文字として用いられる場合、０≦$B^{¥}$＜２Ｎであり、別の表現をすれば、$B^{¥}$は$B^{¥}$と合同である最小の正のＢと等しいか、又は最小の正のＢにモジュラスＮを加えたものと合同のものと等しいかのいずれかである。
本発明に特有のその他の記号は、後でまとめて説明する。
【００２３】
Ｘ＝Ａ ｍｏｄ Ｎの場合、ＸはＡをＮで割った剰余として定義され、即ち、３＝４５ ｍｏｄ ７であり、ＧＦ（２^{q}）ではより単純であり、即ち、１１１１ ｍｏｄ １００１＝０１１０である。
【００２４】
数論において、Ｘのモジュラー逆数（ｍｏｄｕｌａｒ ｍｕｌｔｉｐｌｉｃａｔｉｖｅ ｉｎｖｅｒｓｅ）はX^{-1}と記述され、Ｘ・X^{-1} ｍｏｄ Ｎ＝１によって定義される。

Ｘ＝３、Ｎ＝１３の場合、X^{-1}＝９であり、即ち、ＧＦ（ｐ）において３・９を１３で割った剰余は１である。

【００２５】

両方の数値からなる体で、我々は概して、指数関数を利用して、Ａの逆数を計算する方を選択するので、例えば、A^{-1} ｍｏｄ ｑ≡A^{q-2} ｍｏｄ ｑである。

【００２６】

略語ＭＳ及びＬＳは、デジタル用語で一般に用いられているように、ビット、文字、全オペランド値に言及する場合に夫々「最上位」及び「最下位」を意味するように用いられるが、逆モードの多項式ベースにおいて、オペランドは最初にＭＳデータをロードされ、最後にＬＳデータをロードされ、そこにおいて、データ語（ｄｅｔａ ｗｏｒｄ）のビットオーダー（ｂｉｔ ｏｒｄｅｒ）は、ロード時に逆にされる（ｂｅ ｒｅｖｅｒｓｅｄ）。

【００２７】

この明細書の全体に亘って、Ｎは値Ｎ、及び、Ｎを保存する送りレジスタ（ｓｈｉｆｔ ｒｅｇｉｓｔｅｒ）の名称の両方を示す。

値上のアスタリスクの上付き文字は、その値が潜在的に不完全であるか変化する可能性があることを意味する。

Ａは累乗されるべき数値であり、ｎはＮオペランドのビット長さである。

初期状態設定の後、ＡがA^{*}に「モンゴメリＰ体に標準化された（Ｍｏｎｔｇｏｍｅｒｙ Ｐ ｆｉｅｌｄｎｏｒｍａｌｉｚｅｄ）」（Ｐ１に記載したA^{*}＝2^{n}・Ａ）場合、A^{*}及びＮは、概してべき算における中間段階を通じて一定値である。

計算がビットの標準的な逆にされていない位置（ｕｎｒｅｖｅｒｓｅｄ ｐｏｓｉｔｉｏｎｉｎｇ）によって実行されるＧＦ（2^{q}）計算において、我々はこの同じプロトコルによって結合される（ｂｅ ｂｏｕｎｄ）。

しかし、桁上げがないので、逆フォーマットを利用して、我々の計算は最上位ゼロを生成し、それを無視し、乗算のシフト（ｓｈｉｆｔ；桁送り）を表さない。

【００２８】

第一の反復の間、べき算の初期状態設定の後、ＢはA^{*}に等しい。

Ｂは、最終的にべき算の所望の結果と等しくなる累算された値が存在するレジスタの名称でもある。

Ｓ又はS^{*}は一時的な値を示し、Ｓは、ＧＦ（ｐ）のＳにおける数の単一のＭＳビット以外の全てを保存するレジスタを示す。

（このＭＳビットに連結されるS^{*}は、Ｓと同一である。）

Ｓ（ｉ－１）は、ｉ番目の反復の最初でのＳの値を意味する。

これらの多項式計算において、Ｓ上のモジュラー縮小を実行する必要はない。

【００２９】

概して、素数フィールドにおけるＸ，Ｙのモンゴメリ乗算は、実際に（Ｘ・Ｙ・2^{-n}）ｍｏｄ Ｎの実行であり、ｎは概してモジュラスにおける文字数である。

これは

【外１】

$\mathcal{P}(A\cdot B)N$ （本明細書において▲Ｐ▼(Ａ・Ｂ)Ｎと表す。）

と記述され、Ｐ体におけるＭＭ又は乗算を意味する。

モンゴメリ数学のコンテクスト（ｃｏｎｔｅｘｔ）において、我々は、Ｐ体及び多項式ベースの体における乗算及び平方を乗算及び平方演算と称す。

【００３０】

我々は、ＧＦ（2^{q}）におけるモンゴメリ型演算のこの革新的な拡張を、逆フォーマットのデータ順序（ｄａｔａ ｏｒｄｅｒ）を意味するように再定義するが、ＭＳゼロ強制（ＭＳ ｚｅｒｏ ｆｏｒｃｉｎｇ）は合同を変更させず、若しくは、厄介な寄生要素（ｐａｒａｓｉｔｉｃ ｆａｃｔｏｒ）を開始させない。

従って、我々は、演算の拡張を受け入れるため及びより広い直列乗算器バスを有する構成を可能にするために、新たな記号のセットを導入する。
自然数のスーパースカラ乗算器を可能にするために、１ビットを越える直列乗算器ストリームが好ましい。
このような乗算器装置は、同時にモジュラー算術乗算及び縮小を実行することができる装置に、３２ビット被乗数及び４ビット乗数を受け入れることができる。
【００３１】
直列／並列のスーパースカラモジュラー乗算器の拡張における記号
【外２】

／ （本明細書において▲l▼と表す。）：

文字のビット数（桁）
ｒ：乗数（ｍｕｌｔｉｐｌｉｅｒ ｃｈａｒａｃｔｅｒ）の基数 $r=2^{\blacktriangle l \blacktriangledown}$
ｎ：文字におけるオペランド（乗数、被乗数、モジュラス）のサイズ。
モンゴメリ演算のＧＦ（ｐ） 体における計算の実証において、▲ｌ▼は１に等しく、ｎはモジュラスのオペランドのビット長である。
ｋ：文字における直並列の乗数の長さ
ｍ：被乗数のインタリーブされたスライス（ｓｌｉｃｅ）（セグメント）の数で、ｍ＝ｎ／ｋ
S_i：ｉ番目のＭＭ反復の部分積の結果で、$0 \leqq i \leqq m-1$，$S0=0$
$S_{i\,0}$：Ｚの第一のｋ文字の右側ゼロを無視した後の、ｉ番目の反復結果の右側の文字$\underline{S_i}$：ｉ番目の結果の左側のｎ－ｋ文字
$S_{i\,j}$：S_iのｊ番目の文字
Ａ：ｍ・ｋ文字からなる並列の被乗数
A_i：Ａのｉ番目のｋ文字のスライス（及び／又は文字列A_i）
$A_{i\,t}$：A_iのｔ番目の文字
Ｂ：直列乗数（及び／又はＢのレジスタ記憶装置）
B_0：Ｂの最初の右側のｋ文字
$\underline{B}$：Ｂの最後の左側の（ｎ－ｋ）文字
$B_{0\,j}$：B_0のｊ番目の文字
$\underline{B_j}$：$\underline{B}$のｊ番目の文字
Ｎ：モジュラスのオペランド。｛及び／又は前記乗数を保存するレジスタ。｝
N_0：Ｎの右側のｋ文字。｛ＧＦ（ｐ）におけるＬＳ文字，ＧＦ（2^q）におけるＭＳ文字｝
$\underline{N}$：Ｎの左側の（ｎ－ｋ）文字。｛ＧＦ（ｐ）におけるＭＳ文字，ＧＦ（2^q）におけるＬＳ文字｝
$N_{0\,j}$：N_0のｊ番目の文字
$\underline{N_j}$：$\underline{N}$のｊ番目の文字
Y_0：ＧＦ（ｐ）におけるモンゴメリ乗算及び縮小の両方に必要なゼロ強制変数。
Y_0は、ｋ文字長さである。
$Y_{0\,j}$：Y_0のｊ番目の文字
Ｒ：桁上げ保存アキュムレーターに存在する値の総和（未解の内部桁上げ挿入（ｃａｒｒｙ ｉｎｓ）を含む）及び最終の直列加算器（ｓｅｒｉａｌ ｓｕｍｍａｔｏｒ；直列総和器）４６０からの桁上げビット（ｃａｒｒｙ ｏｕｔ ｂｉｔ）。
$J_{0\,0}$：「即座の」有限体乗算及び縮小のためのモジュラスＮのゼロ強制文字関数。 ▲ｌ▼＝１では$J_{0\,0}$は常に１に等しい。
$Carry_j$：基数ｒの直並列乗算器のｊ番目の内部桁上げ文字
$Carry_a$：ＧＦ（ｐ）計算のための出力直列加算器の基数ｒの桁上げ
Sum_j：基数ｒの直並列乗算器のｊ番目の内部の和の文字。

ＬＳ：最下位
ＭＳ：最上位
‖：連結（ｃｏｎｃａｔｅｎａｔｉｏｎ） 例えばＡ＝１１０，Ｂ＝１１０１の時、Ａ‖Ｂ＝１１０１１０１
右側：全てのＧＦ（ｐ）計算データブロックの最下位の部分と逆にされたＧＦ（2^q）フォーマットのＭＳ部分
左側：全てのＧＦ（ｐ）計算データブロックの最上位の部分と逆にされたＧＦ（2^q）フォーマットのＬＳ部分
ＧＦ（ｐ）：ガロア体。厳密に言うと、加算、減算、乗算、擬除算（ｐｓｅｕｄｏ－ｄｉｖｉｓｉｏｎ）を許容する合成数（二つの非常に大きな素数の積）を使用する素数での有限体。
ＧＦ（2^q）：ガロア体で２を法とする演算を利用する有限体。
【外３】

⊕ （本明細書において▲○＋▼と表す。）：

特定の記数法に適するように、桁上げの有無に関わらず整数を加算又は減算するために外部で切り換えられる演算子又は装置。
【外４】

⊗ （本明細書において▲○×▼と表す。）：

ＧＦ（ｐ）又はＧＦ（2^q）のいずれかで乗算を実行するために切り換えられる演算子又は装置。
【外５】

$\mathcal{S}$ （本明細書において▲Ｓ▼と表す。）：

数値体スイッチであり、▲Ｓ▼＝１の場合、スイッチはＧＦ（ｐ）計算で全ての桁上げの入出力を可能にするように機能し、▲Ｓ▼＝０の場合、スイッチはＧＦ（2^q） 計算で全ての桁上げ入出力を不可能にするように機能する。
ＳｕｐｅｒＭＡＰ：本発明の目的である登録商標のスーパースカラモジュラー演算プロセッサ系統（Ｓｕｐｅｒｓｃａｌａｒ Ｍｏｄｕｌａｒ Ａｒｉｔｈｍｅｔｉｃ Ｐｒｏｃｅｓｓｏｒ ｆａｍｉｌｙ）の構成要素の一つ。
ＳｕｐｅｒＭＡＰの商標は、ヨーロッパにおいて登録され、米国においては係属中である。
【００３２】
本発明の第一の態様によると、多項式ベースのＧＦ（2^q）及びＧＦ（ｐ）の両体の演算において▲○×▼乗算及び平方を実行するマイクロエレクトロニクス装置であって、平方及び減算が直列供給される基数$2^{▲l▼}$の乗数Ｂをｋ文字の被乗数セグメントA_i及びｋ文字の▲○＋▼アキュムレーターと共に使用し、限定された合同がモジュラスＮ上で被乗数A_iに乗数Ｂを掛け合わせてシストリックな（ｓｙｓｔｏｌｉｃ）態様で「即座に」実行され、その結果が多くとも２ｎ＋１文字長さであり、保存されないｋの最初に発生し無視されたゼロ文字を含み、ｋ文字がモジュラスと同等のビットを有し、前記演算が二つの段階で行われ、各々が少なくともｎビット長さのオペランドを保持するように動作し、夫々符号Ｂの乗数値及び符号Ｎの2^nより小さなモジュラスを保存するように動作する第一主メモリレジスタ手段Ｂ及び第二主メモリレジスタ手段Ｎと、装置から発生する全ての最初のｋ文字がゼロにされるように▲○＋▼アダーアキュムレーターデバイスにおける値にモジュラス値が▲○＋▼加算されるような場合に、「即座に」予測するように動作するデジタル論理検知検出器（ｄｉｇｉｔａｌ ｌｏｇｉｃ ｓｅｎｓｉｎｇ ｄｅｔｅｃｔｏｒ）Y_0と、ただ一つの少なくともｋ文字長さの▲○＋▼加算器と、ｋ文字の被乗数を受け

入れるように動作する▲○＋▼加算装置（▲○＋▼ ｓｕｍｍａｔｉｏｎ ｄｅｖｉｃｅ）と、順番に被乗数値を▲○＋▼アキュムレーターデバイス内に入れ替え、順番にＢレジスタから乗数値を受け入れるように動作する▲○×▼乗算装置と、第一段階においてｋの最初に発生するゼロ文字を出力させるように動作する乗数と同時に生成した「即座の」予測値とを有し、各々の有効マシンサイクル（ｅｆｆｅｃｔｉｖｅ ｍａｃｈｉｎｅ ｃｙｃｌｅ）で少なくとも一つの指定された被乗数が▲○＋▼アキュムレーターデバイスに▲○＋▼加算される、少なくともｋ文字の入力被乗数のためのモジュラー乗算装置と、全ゼロ文字列の値である第一被乗数、被乗数A_iである第二被乗数、モジュラスのN_0セグメントである第三被乗数のうちの一つ若しくは二つの被乗数からなる、順番に▲○＋▼アキュムレーションデバイス内に入れ替えられる被乗数値と、▲ｌ▼ビットのｋ文字直列入力Y_0乗数値を予測するアンティシペーター（ａｎｔｉｃｉｐａｔｏｒ；予測回路）とからなるマイクロエレクトロニクス装置が提供され、該装置は乗数値を乗算装置に順番に入力するように動作でき、前記乗数値は第一段階においてＢオペランドであり、同時に、第二乗数値は最初に発生するゼロを出力させるための「即座の」予測されたｋ文字列であるY_0からなり、該装置は更に、アキュムレーションデバイス▲○＋▼からなり、該アキュムレーションデバイスは被乗数が該アキュムレーションデバイスへの▲○＋▼であると同時に値を出力するように動作し、第二段階において▲○＋▼アキュムレーションデバイスからの最終のモジュラー▲○×▼乗算の結果を出力するように動作する出力転送機構（ｏｕｔｐｕｔ ｔｒａｎｓｆｅｒ ｍｅｃｈａｎｉｓｍ）からなる。

【００３３】

好ましい実施例によると、▲○＋▼アキュムレーションデバイスへの▲○＋▼加算は、各々の新たな直列にロードされた高位乗数（ｈｉｇｈ ｏｒｄｅｒ ｍｕｌｔｉｐｌｉｅｒ ｃｈａｒａｃｔｅｒ）により始まる。

【００３４】

好ましくは、乗数は、入力されたＢ文字と対応する入力されたY_0文字の両方がゼロの場合は、▲○＋▼アキュムレーションデバイスへの▲○＋▼加算はされず、入力されたＢ文字が１であり、対応するY_0がゼロである場合は、A_i被乗数のみが▲○＋▼加算され、入力されたＢ文字がゼロであり、対応するY_0が１である場合は、モジュラスＮのみが▲○＋▼加算され、入力されたＢ文字と対応するY_0文字の両方が１である場合は、被乗数A_iと共にモジュラスＮが▲○＋▼加算されるように動作する。

【００３５】

好ましくは、該装置は被乗数値A_i，Ｎを二つの指定されたプリロードバッファ（ｐｒｅｒｏａｄ ｂｕｆｆｅｒ）にプリロードし、これらの値を第三プリロードバッファに▲○＋▼加算するように動作して、各々の被乗数値を別々に▲○＋▼加算する必要をなくす。

【００３６】

好ましくは、乗数値は直列単一文字の形式での入力用に調整され、Y_0検出装置は一回のクロックで一文字のみ予測するように動作する。

【００３７】

▲○＋▼アキュムレーションデバイスが２を法としてＸＯＲ加算／減算の計算を実行する好まし実施例において、加算成分及び減算成分中の全ての桁上げビットが無視されることにより、計算における桁溢れや更に限定する有益性は排除される。

【００３８】

好ましくは、桁上げ入力はゼロ（▲Ｓ▼＝０を意味する）まで実行不能とされ、概して、多項式ベースの乗算を実行するように動作する。

【００３９】

好ましくは、該装置は桁上げ回路を除くことにより桁上げなしの算術を提供するように動作するので、▲Ｓ▼はＧＦ（2^q）において計算する回路方程式（ｃｉｒｃｕｉｔ ｅｑｕａｔｉｏｎ）中の要素に影響するゼロに等しく、▲Ｓ▼は除かれた回路を示し、▲○＋▼で表される全ての加算器及び減算器は２を法とした加算／減算要素の排他的論理和を求める。

【００４０】
好ましい実施例は、動作ユニットから発生した最初のｋ文字セグメントがゼロであって、次のＹ$_0$文字を予測する際に以下の四つの数量によって調整されるように適応する。
ｉ．　Ａ$_i$レジスタの右側文字にＢストリームのＢ$_d$文字を掛けた▲ｌ▼ビット毎の$2^{▲l▼}$を法とする▲○×▼乗算の結果の▲ｌ▼ビットのＳ$_{out}$ビット　Ａ$_0$・▲○×▼Ｂ$_d$ ｍｏｄ $2^{▲l▼}$、
ｉｉ．　▲○＋▼アキュムレーションデバイスから最初に発生する桁上げ文字▲Ｓ▼（ＣＯ$_0$）、
ｉｉｉ．　▲○＋▼アキュムレーションデバイスの右側の発生セル（ｅｍｉｔｔｉｎｇ ｃｅｌｌ）からの次（ｔｈｅ ｓｅｃｏｎｄ）に発生する桁上げ文字からの▲ｌ▼ビットのＳ$_{out}$文字ＳＯ$_1$、
ｉｖ．　Ｎ$_0$モジュラス被乗数レジスタにおける右側文字の負の逆数である▲ｌ▼ビットのＪ$_0$値。
ここで、Ａ$_0$ ▲○×▼ Ｂ$_d$ ｍｏｄ $2^{▲l▼}$、▲Ｓ▼（ＣＯ$_0$）、ＳＯ$_1$の値は、共に文字に▲○＋▼加算される文字であり、▲ｌ▼ビットのゼロを発する有効なＹ$_0$ゼロ強制予測文字（Ｙ$_0$ ｚｅｒｏ－ｆｏｒｃｉｎｇ ａｎｔｉｃｉｐａｔｏｒｙ ｃｈａｒａｃｔｅｒ）を出力するために「即座に」Ｊ$_0$文字によって乗算される。
【００４１】
該装置は好ましくは逆モードで実行される多項式ベースのオペランドで乗算を実行するように動作でき、右側のＭＳ文字から左側のＬＳ文字を乗算し、モンゴメリ型寄生関数（Ｍｏｎｔｇｏｍｅｒｙ ｔｙｐｅ ｐａｒａｓｉｔｉｃ ｆｕｎｃｔｉｏｎ）なしでモジュラー縮小された▲○×▼乗算を実行するように動作する。
【００４２】
好ましくは、該装置は、更に直列供給されるプリロードバッファからなり、被乗数値は一以上の記憶装置から即座にプリロードバッファにプリロードされる。
【００４３】
該装置は、好ましくは、Ｙ$_0$検出器が▲○＋▼アキュムレーションデバイスにおいて▲○＋▼加算にモジュラスを▲○＋▼加算する必要性を検出ように動作する時に、最初に発生する出力文字がゼロであるように、▲ｌ▼ビットの▲○＋▼加算器回路を経由して、更なるｎビットのＳレジスタから発生する先の値に乗算ストリームを▲○＋▼加算するように動作し、Ｙ$_0$検出器は、次に順番に加算した文字Ａ$_0$ ▲○×▼ Ｂ$_d$ ｍｏｄ ２、▲Ｓ▼（ＣＯ$_0$）、ＳＯ$_1$、Ｓ$_d$、▲Ｓ▼（ＣＯ$_Z$）を利用して、▲ｌ▼ビットのＪ$_0$値により即座に▲○×▼乗算される有限体であるように▲○＋▼加算される文字の合成（ｃｏｍｐｏｓｉｔｅ）を検出するように動作するが、ここで、プロセスにおいて用いられる有限体に適するように、▲○＋▼は加算を定義し、▲○×▼は乗算を定義する。
【００４４】
好ましくは、▲ｌ▼＝１で、ハードウェアを追加することなく、Ｊ$_0$は暗黙的に（ｉｍｐｌｉｃｉｔｌｙ）１であり、Ｊ$_0$▲○×▼乗算は暗黙的（ｉｍｐｌｉｃｉｔ）である（ｉｍｐｌｉｃｉｔ）。
【００４５】
好ましくは、コンパレータ（ｃｏｍｐａｒａｔｏｒ）は、最初の右側の発生するｋゼロ文字が無視されるＧＦ（ｐ）において機能している間、▲○×▼モジュラー乗算装置からの有限体出力を検知するように動作し、その出力がモジュラスＮより大きいことにより、モジュラー縮小をを調整するように動作し、前記値は、乗算装置からの出力ストリームの行き先であるメモリレジスタから出力されることにより、より小さな積の値に第二の記憶装置を割り当てる必要がなくなる。
【００４６】
好ましくは、ＧＦ（2^q）における▲○×▼モジュラー乗算で、該装置は外部で予め計算された▲ｌ▼ビットより大きなゼロ強制因子なしで乗算するように動作する。
【００４７】

好ましい実施例は、Ａオペランド値又はＢオペランド値のいずれかをゼロに再設定し、部分的な結果値S_0を１に設定することにより、J_0定数を計算するように動作する。

【００４８】

本発明の第二の態様によると、ＡにＢを掛けてモジュラスがＮ（Ａ ｔｉｍｅｓ Ｂ ｍｏｄｕｌｕｓ Ｎ）となる出力ストリームを生成するために、整数Ａ及びＢのインタリーブされた有限体モジュラー乗算を実行するマイクロエレクトロニクス装置であって、モジュラスオペランドレジスタにおける文字の数ｎがｋ文字のセグメント長さよりも大きく、▲○×▼乗算プロセスが複数のインタリーブされた反復で実行され、各々のインタリーブされた反復で▲○×▼乗算装置に入力されるオペランドは、モジュラスＮ、乗数Ｂ、予め計算された部分的な結果Ｓ、被乗数Ａのｋ文字列セグメントからなり、セグメントはA_0文字列セグメントからA_{m-1}文字列セグメントまで進行し、各々の反復の結果は次の順番の一時的結果Ｓに▲○＋▼加算され、反復結果の最初に発生する文字はゼロであって、各々が夫々乗算値、部分的な結果値、モジュラスを保存するように動作する第一主メモリレジスタＢ、第二主メモリレジスタＳ、第三主メモリレジスタＮと、順番に前記Ｂレジスタからの入力と、各々の反復において最初の右側ゼロを出力させる乗数として有用な「即座の」予測値Y_0の入力と、前記Ｎレジスタと、少なくとも、Ａ，Ｂ，Ｎレジスタ資源からの値を順番に受け入れ、続いて被乗数ゼロ強制値Y_0をも受け入れるように動作する被乗数並列レジスタとからの入力で、複数の反復▲○×▼乗算プロセスの各々の間、順番に複数の被乗数値の一つ又は二つを▲○＋▼アキュムレーションデバイスに▲○＋▼加算し動作するモジュラー乗算装置と、該装置は、第一段階の間乗数となるように動作し、第二段階の間被乗数となるように動作する二進列（ｂｉｎａｒｙ ｓｔｒｉｎｇ）を生成するように動作するY_0検出装置を更に利用し、該装置は、第一にゼロ、第二に被乗数Ａのｋ文字列セグメントであるA_i、第三にモジュラスＮの最初に発生するｋ文字であるN_0からなる、第一段階で▲○＋▼アキュムレーションデバイスに入れ替えるのに適した被乗数値を得るように動作し、該装置は更に、次の反復で並列の結果を生成するために、▲○＋▼アキュムレーションデバイスから発する値に▲○＋▼加算される先の反復の結果である一時的結果値Ｓを利用するように動作し、該装置は更に、第一にゼロ、第二に第一段階から残存するオペランドA_i、第三に第一段階で予測されたY_0値からなる乗算の第二段階で▲○＋▼アキュムレーションデバイスに順番に入力される被乗数値を利用するように動作し、第一段階で乗算装置に入力される乗数値は、最初に発生する文字列B_0であり、前記乗算装置は、第二段階でプリロードされた被乗数バッファに生成すると同時に文字毎にロードされる予測されたY_0文字列からなる第二の▲○×▼乗数値と▲○×▼乗算するのと並行して前記文字列セグメントを乗算するように動作し、第二段階の間、装置に入力されるように動作する二つの乗数値は夫々、Ｂで表されるＢオペランドからの左側ｎ－ｋ文字の値と、Ｎで表されるＮモジュラスの左側ｎ－ｋ文字であり、前記装置は更に、▲○＋▼アキュムレーションデバイスに残存する結果値の左側セグメントを結果レジスタに転送するように最終段階で動作する乗算フラッシュアウト装置（ｍｕｌｔｉｐｌｙｉｎｇ ｆｌｕｓｈ ｏｕｔ ｄｅｖｉｃｅ）からなる。

【００４９】

好ましくは、該装置は、ＭＳ文字からＬＳ文字まで乗算する逆モードにおいて実行される多項式ベースのオペランドで▲○×▼乗算を実行するように動作でき、出願人の米国特許第５７４２５３０号明細書に記載しているように、モンゴメリ型寄生関数なしでモジュラー縮小を実行するように動作する。

【００５０】

本発明の第三の態様によると、被乗数の最初に発生する値、Ｂ乗数の現在の入力、▲○＋▼アキュムレーションデバイスからの桁上げ値、▲○＋▼アキュムレーションデバイスからの▲○＋▼加算値、先に計算された部分的な結果からの現在の値、先の部分的な結果に▲○＋▼アキュムレーションデバイスからの結果を▲○＋▼加算する▲○＋▼加算器からの桁上げ値を用いてY_0値を予測するように動作する装置が提供される。

【００５１】

好ましくは、該装置は、装置からのｋの最初に発生する値がゼロ文字であることを保証するのに適しており、前記適していることが以下の数量を用いる次の順番のY_0文字の予測からなる。

ｉ．　A_iレジスタの右側文字にＢストリームのB_d文字を掛けた▲ｌ▼ビット毎の$2^{▲l▼}$を法とする▲○×▼乗算の結果の▲ｌ▼ビットのS_{out}ビット　$A_0 \cdot B_d \bmod 2^{▲l▼}$、

ｉｉ．　▲○＋▼アキュムレーションデバイスから最初に発生する桁上げ文字▲Ｓ▼（CO_0）、

ｉｉｉ．　▲○＋▼アキュムレーションデバイスの右側の発生セル（ｅｍｉｔｔｉｎｇ　ｃｅｌｌ）からの第二からの▲ｌ▼ビットのS_{out}文字SO_1、

ｉｖ．　Ｓストリームからの次の順番の文字値S_d、

ｖ．　Ｚ出力全加算器からの▲ｌ▼ビット桁上げ文字▲Ｓ▼（CO_Z）、

ｖｉ．　N_0モジュラス被乗数レジスタにおける右側文字の負の逆数である▲ｌ▼ビットのJ_0値。

ここで、$A_0 \cdot B_d \bmod 2^{▲l▼}$、▲Ｓ▼（$CO_0$）、$SO_1$、$S_d$の値は、共に文字に▲○＋▼加算される文字であり、有効なY_0ゼロ強制予測文字を出力するために「即座に」J_0文字によって▲○×▼乗算される。

【００５２】

更なる実施例において、出力結果をモジュラスＮと比較するように動作する少なくとも一つのセンサをも提供され、その機構は、結果レジスタの出力に第二（ｔｈｅ　ｓｅｃｏｎｄ）の減算器を動作させることにより、出力された結果値と限定された合同であるモジュラー縮小された値を出力するように動作し、より小さな結果に第二の記憶装置を割り当てる必要を排除する。

【００５３】

なおも更なる実施例において、最初の値の一つが別のプリロードバッファにロードされるのと並行して、二つの被乗数の▲○＋▼加算である値が、少なくとも一つのｋ文字メモリ手段レジスタでプリロード文字バッファにロードされる。

【００５４】

本発明の第四の態様によると、一連のインタリーブされた▲○×▼モジュラー乗算及び平方を実行するように動作し、三つの自然数乗算演算と同等のものを並行して実行するのに適し、結果がべき算である単一の▲○＋▼アキュムレーションデバイス及び予測ゼロ強制機構（ａｎｔｉｃｉｐａｔｉｎｇ　ｚｅｒｏ　ｆｏｒｃｉｎｇ　ｍｅｃｈａｎｉｓｍ）を有する装置が提供される。

【００５５】

一つの実施例において、次の順番に用いられる被乗数は、即座にプリロードレジスタバッファ（ｐｒｅｌｏａｄ　ｒｅｇｉｓｔｅｒ　ｂｕｆｆｅｒ）にプリロードされる。

【００５６】

更なる実施例において、装置のバッファ及びレジスタは外部メモリー資源からの値をロードされるように動作し、前記バッファ及びレジスタは計算の間に外部メモリー資源にアンロードされるように動作するので、オペランドの最大サイズが利用できるメモリー手段に依存する。

【００５７】

なおも更なる実施例において、メモリーレジスタ手段をも提供され、前記メモリー手段は、概して、直列単一文字入力／直列単一文字出力、並列少なくともｋ文字入力／並列少なくともｋ文字出力、直列単一文字入力／並列少なくともｋ文字出力、並列少なくともｋ文字入力／直列単一文字出力である。

【００５８】

好ましくは、該装置は、乗算型反復の最終段階の間、乗数入力が桁上げ保存▲○＋▼累算メモリーの左側セグメントをフラッシュアウトするように動作するゼロ文字であることを提供するように動作できる。

【００５９】
好ましくは、該装置は、次の順番の被乗数を、必要とされるよりも前に、即座にプリロードメモリーバッファにプリロードするように動作できる。
【００６０】
好ましくは、該装置は、被乗数値を中央記憶装置から即座にプリロードバッファにプリロードするように動作できる。
【００６１】
同装置は、好ましくはＡ，Ｂの両方をゼロに再設定し、S_0＝１に設定することにより、モジュラスの右側ｋ文字セグメントに関するｋ文字モンゴメリ定数J_0を計算するように動作でき、続いてｋビット乗算を実行する。
結果はY_0レジスタに存在する。
【００６２】
モンゴメリ型演算を利用するモジュラー乗算シーケンス
ｋ文字桁上げ保存加算器ＣＳＡは、多項式体及び素数フィールドの両方における直列／並列のスーパースカラモジュラー乗算の基礎である。
多項式ＧＦ（2^q）ベースの計算は、好ましくは全ての桁上げ機構のスイッチを切られた状態で実行される。
【００６３】
直並列スーパースカラモンゴメリ乗算器は三段階のモンゴメリモジュラー積を計算するが、その一つの好ましい実施例において、最終段階はＣＳＡの桁上げにより左側のｋ文字セグメント全体の単一クロックのダンプであってもよく（ＭＳは通常の乗算で、ＬＳは逆モードの多項式計算）、よりコンパクトな実施例では、最終段階はＣＳＡの内容のｋ有効クロック直列フラッシュアウト（ｋ ｅｆｆｅｃｔｉｖｅ ｃｌｏｃｋ ｓｅｒｉａｌ ｆｌｕｓｈ ｏｕｔ）であってもよい。
【００６４】
先のＰ２の開示において、Y_0因子はビット毎に計算されるので、J_0の右側のビット、即ち定義では一つのビットのみが重要であって、モジュラスの右側ビットの関数である。この拡張された装置において、装置は文字直列（ｃｈａｒａｃｔｅｒ ｓｅｒｉａｌ）であり、▲ｌ▼ビット文字Y_0は各々のクロックサイクルで生成される。
先のＰ１の開示において、Y_0は累算された結果に必要な回数だけモジュラス値を加算する第一段階のゼロ強制関数であるので、関連した解が合同であってｍｋ＋１文字より決して長くなく、最初に右側で発生した文字は全てゼロであった。
ＱＮ≡０ ｍｏｄ ＮなのでＸ＋ＱＮ≡Ｘである。
【００６５】
モジュラー乗算シーケンス
計算を開始する前に、我々は装置に過去の一時的又はランダムな値がなく、オペランドＮ，Ｂ及び少なくとも最初のＡのセグメント値が装置のレジスタにおいて利用できると仮定する。
S_0＝０で初めの部分積は概してゼロである。
典型的には、モジュラー演算は、概して三つの明確な段階において二以上のｋ文字セグメントからなるオペランドで実行される。
ｍセグメントのモジュラスがある通常の全乗算では、概して、ｍスーパースカラ乗算インタリーブされた反復（ｍ ｓｕｐｅｒｓｃａｌａｒ ｍｕｌｔｉｐｌｉｃａｔｉｏｎ ｉｎｔｅｒｌｅａｖｅｄ ｉｔｅｒａｔｉｏｎｓ）があり、被乗数の各々のセグメントが、典型的にはＢである全ての乗数によって掛けられる。
【００６６】
各々のインタリーブされたスーパースカラ乗算のｉ＝０番目のセグメントにおける第一段階のプロセスは、一般的なスーパースカラ乗算累算間の相互作用である。
即ち、S_i ▲○＋▼ A_i・B_0 ▲○＋▼ Y_0・N_0
（B_0，Y_0は直列的に文字毎にオペランドの最初のセグメントから乗算器に供給され、

A_i，N_0は並列的な単一のスライスのオペランドであり、S_iは先の反復／計算からの部分積である。０番目の最初の反復では$S_i = 0$。）

【００６７】

第一段階のプロセスは、先の結果の右側セグメントへの二つのスーパースカラ積の▲○＋▼加算を実行する。

ゼロのｋ文字列は、乗算装置から発生して、無視されるが、最初のセグメントの部分積は装置バッファに存在し、第二段階の結果に加算される。

【００６８】

第一段階の結果は典型的にはＲからなり、ＣＳＡの内容が全ゼロの直列出力された右側セグメントに連結される。（ＧＦ（ｐ）計算において、Ｒへの更なるＬＳ桁上げビットがある。）

【００６９】

第二段階のプロセスは、一般的なスーパースカラ乗算累積間相互作用である。

即ち、Ｒ ▲○＋▼ S_i ▲○＋▼ A_i・Ｂ ▲○＋▼ Y_0・Ｎ

（下線を引いた変数、例えばBはオペランドの残存する左側の値であることに留意する。それは、典型的には一つ以上のセグメント、即ち、ｍ－１セグメントである。B，Nは直列的に文字毎に乗算器に供給され、第一段階から残存するA_iと、第一段階において乗数であり次の反復において被乗数となるように第一段階においてマシンへロードされるY_0とは並列なオペランドである。）

【００７０】

概してｍ－１回の反復からなる第二段階の終わりに、S_iの左側セグメントが転送可能な状態でＣＳＡ内に残存し、右側のスライス（ｋ文字セグメント）は、装置から典型的にはＳレジスタに発せられる。

素数フィールドにおける乗算が、従来の桁上げ保存加算法で実行されることに留意すべきである。

ＧＦ（2^q）の逆フォーマットモードにおける乗算は、最上位文字から最下位文字まで進行した。

モジュラス値がアキュムレーターに「加算され」なければならない時、Y_0関数が予測した。

装置において使用できない桁上げビットを除いて、機械的処理は二つの記数法で典型的には同一である。

【００７１】

我々は、有限体におけるゼロ強制ベクトルの$Y_{0\,j}$文字を誘導するために利用される方法を開示する。

【００７２】

$J_{0\,j} \equiv -N_{0\,0}^{-1} \mod 2^{▲\,l\,▼}$を計算する。

モジュラスに対し互いに素である全ての自然数は、両数値体において逆数を有する。

$N_{0\,0}$は奇数であるので、２の因数を有しない。

$\mod 2^{▲\,l\,▼}$の全ての因数は２であるので、最下位１を有する数及び唯一の因数が２であるモジュラスは、互いに素であり、$J_{0\,j}$は常に存在する。

正式には、奇数$N_{0\,0}$及び$2^{▲\,l\,▼}$について、ｇｃｄ（$N_{0\,0}$，$2^{▲\,l\,▼}$）＝１となる。

【００７３】

関数のこの単一文字は、ランダムロジック、単純回路、又は単純なルックアップテーブル（ｌｏｏｋ ｕｐ ｔａｂｌｅ）で配線実装され得る。

ルックアップテーブルにおいて誘導されなければならない2^{k-1}の異なる値がある。

逆モードフォーマットにおいて、多項式のモジュラスは、右詰めで、名目上奇数でなければならない。

典型的な指数関数（ｅｘｐｏｎｅｎｔｉａｔｉｏｎａｌ ｆｕｎｃｔｉｏｎ；べき算関数）において、モジュラスビットの右側ビットは一つであり、名目上奇数であり、2^kを法とする奇数の逆数は常に奇数である。

【００７４】
▲ｌ▼＝１である場合、Ｊ$_{00}$乗数は明らかに１に等しく、計算される必要はない。
第一段階の間、Ｙ$_{0}$関数によって出される文字出力の結果は常に０であるので、ＳｕｐｅｒＭＡＰのｊ番目の文字出力Ｚ$_{ij}$は、０＝（２$^{▲l▼}$Ｒ ▲○＋▼ Ｓ$_{ij}$ ▲○＋▼ Ａ$_{i0}$・Ｂ$_{0j}$ ▲○＋▼ Ｙ$_{0j}$・Ｎ$_{00}$）ｍｏｄ ２$^{▲l▼}$＝Ｚ$_{ij}$となり、従って、（Ｒ ▲○＋▼ Ｓ$_{ij}$ ▲○＋▼ Ａ$_{i0}$・Ｂ$_{0j}$）≡－Ｙ$_{0j}$・Ｎ$_{00}$であり、Ｙ$_{0j}$≡－Ｎ$_{00}^{-1}$（Ｒ ▲○＋▼ Ｓ$_{ij}$ ▲○＋▼ Ａ$_{i0}$・Ｂ$_{0j}$）ｍｏｄ ｒである。
【００７５】
上記の式から、Ｊ$_{00}$が好ましくは両記数法でモジュラスの右側のｋ文字のモジュラー逆数の負の値であることが分かるが、２を法とする演算において、正及び負の値は同じであることに留意すべきである。
【００７６】
Ｒは、図２における最終直列加算器４６０からの桁上げビットに加算されるＣＳＡに残存する値の総和である。
Ｓ$_{ij}$は、ｉ番目の反復での部分積のｊ番目のビットである。
Ａ$_{i0}$は、Ａのｉ番目のスライスの右側文字（ＧＦ（ｐ）におけるＬＳ）である。
Ｂ$_{0j}$は、Ｂのｊ番目の文字である。
Ｂ$_{0}$は、モンゴメリ乗算の全反復の間の定数（乗数）である。
Ｙ$_{0}$は、各々の（ｉ番目の）反復で生成されるｋ文字ベクトルである。
Ｙ$_{0j}$は、反復の第一段階のｊ番目のクロックで生成されるｊ番目の文字である。
Ｎは、ｍスライスされた（ｍ ｓｌｉｃｅｄ）モジュラスである。
Ｎ$_{0}$は、モジュラスの右側スライスである。
Ｎ$_{00}$は、Ｎ$_{0}$の右側文字である。
【００７７】
両体についてスーパースカラモジュラー乗算法の公式化（Ｆｏｒｍａｌｉｚｉｎｇ）
Ｓ$_{0}$＝０
ｉ＝０～ｍ－１（インタリーブ反復）で、
（各々のインタリーブの）第一段階
Ｒ＝０
ｊ＝０～ｋ－１（第一段階の各々の文字）で
Ｙ$_{0j}$＝（Ｊ$_{00}$（Ｒ ▲○＋▼ Ｓ$_{0j}$ ▲○＋▼ Ａ$_{i0}$ ▲○×▼ Ｂ$_{0j}$））ｍｏｄ ２$^{▲l▼}$
Ｚ$_{ij}$＝（Ｒ ▲○＋▼ Ｓ$_{ij}$ ▲○＋▼ Ａ$_{i0}$ ▲○×▼ Ｂ$_{0j}$ ▲○＋▼ Ｙ$_{0j}$ ▲○×▼ Ｎ$_{0}$）ｍｏｄ ２$^{▲l▼}$
Ｒ＝［（２$^{▲l▼}$Ｒ ▲○＋▼ Ｓ$_{ij}$ ▲○＋▼ Ａ$_{i0}$ ▲○×▼ Ｂ$_{0j}$ ▲○＋▼ Ｙ$_{0j}$ ▲○×▼ Ｎ$_{0}$）］／２$^{▲l▼}$
【００７８】
ｋ有効クロックサイクルの後、Ｚストリームの最初のセグメントは全てゼロであって無視され、関連したＹ$_{0}$（ｋ文字ベクトル）は次の段階において被乗数となるように用意され、加算されたＲの値は次の段階において利用される。
【００７９】
第二段階
ｊ＝ｋ～ｎ－１で、
Ｚ$_{ij}$＝（Ｒ ▲○＋▼ Ｓ$_{ij}$ ▲○＋▼ Ａ$_{i0}$ ▲○×▼ Ｂ$_{0j}$ ▲○＋▼ Ｙ$_{0j}$ ▲○×▼ Ｎ$_{0j}$）ｍｏｄ ２$^{▲l▼}$
Ｒ＝［２$^{▲l▼}$Ｒ ▲○＋▼ Ｓ$_{ij}$ ▲○＋▼ Ａ$_{i0}$・Ｂ$_{0j}$ ▲○＋▼ Ｙ$_{0j}$・Ｎ$_{0j}$］／２$^{▲l▼}$
【００８０】
文字ベースの直並列乗算器での上記アルゴリズムのインプリメンテーションは、上記のプロトコルの単純な拡張である。

（Ｑｕｏｔｉｅｎｔ（商）（ｘ，ｙ）は、剰余のない整数の除算関数である。例えば、ｘ＝１０１０１$_b$，ｙ＝１００００$_b$である場合は、Ｑｕｏｔｉｅｎｔ（ｘ，ｙ）＝１）。

【００８１】

$S_0 = 0$

ｉ＝０～ｍ－１（インタリーブのループ）で、

第一段階

ｊ＝０～ｋ－１で、

$Y_{0j} = (J_{00} \cdot S_{i0}$ ▲○＋▼ A_{i0} ▲○×▼ B_{0j} ▲○＋▼ ▲Ｓ▼・$Carry_0$ ▲○＋▼ Sum_1 ▲○＋▼ Ｑｕｏｔｉｅｎｔ（S_{i0} ▲○＋▼ Sum_0，ｒ））） ｍｏｄ ｒ

ｔ＝０～ｋ－１（一つのクロックパルスを有する全ループ）で、

$Sum_t = (Sum_{t+1}$ ▲○＋▼ ▲Ｓ▼・$Carry_t$ ▲○＋▼A_{it} ▲○×▼ B_{0j} ▲○＋▼ Y_{0j} ▲○×▼ N_{0t}）ｍｏｄ ｒ

$Carry_t = ($Ｑｕｏｔｉｅｎｔ（（Sum_{t+1} ▲○＋▼ $Carry_t$ ▲○＋▼ A_{it} ▲○×▼ B_{0j} ▲○＋▼ Y_{0j} ▲○×▼ N_{0t}），ｒ）

（この段階の乗算器装置の出力は、『０』である。）

【００８２】

第二段階

主要部

$Carry_a = 0$

ｊ＝ｋ～ｎ－１，ｔ＝０～ｋ－１（一つのクロックパルスを有する全ループ）で、

$Sum_t = (Sum_{t+1}$ ▲○＋▼ ▲Ｓ▼・$Carry_t$ ▲○＋▼ A_{it} ▲○×▼ B_j ▲○＋▼ Y_{0t} ▲○×▼ N_j）ｍｏｄ ｒ

$Carry_t =$Ｑｕｏｔｉｅｎｔ（（Sum_{t+1} ▲○＋▼ $Carry_t$ ▲○＋▼ A_{it} ▲○×▼ B_j ▲○＋▼ Y_{0t} ▲○×▼ N_j），ｒ）

$S_{i,j-k} = (S_{i,j-2k}$ ▲○＋▼ Sum_0 ▲○＋▼ ▲Ｓ▼・$Carry_a$）ｍｏｄ ｒ

$Carry_a =$Ｑｕｏｔｉｅｎｔ（（$S_{i,j-2k}$ ▲○＋▼ Sum_0 ▲○＋▼ $Carry_a$），ｒ）

乗算器のフラッシング

ｊ＝ｎ～（ｎ＋ｋ－１），ｔ＝０～ｋ－１（一つのクロックパルスを有する全ループ）で、

$Sum_t = (Sum_{t+1}$ ▲○＋▼ ▲Ｓ▼・$Carry_t$）ｍｏｄ ｒ

$Carry_t =$Ｑｕｏｔｉｅｎｔ（（Sum_{t+1} ▲○＋▼ $Carry_t$），ｒ）

$S_{i,j-k} = (S_{i,j-2k}$ ▲○＋▼ Sum_0 ▲○＋▼ ▲Ｓ▼・$Carry_a$）ｍｏｄ ｒ

$Carry_a =$Ｑｕｏｔｉｅｎｔ（（$S_{i,j-2k}$ ▲○＋▼ Sum_0 ▲○＋▼ $Carry_a$），ｒ）

【００８３】

ＧＦ（ｐ）体における▲１▼＝１の特定の場合の例での正式な説明については、Ｐ１を参照する。

【００８４】

上記は、ＡにＢを掛けてモジュラスがＮ（Ａ ｔｉｍｅｓ Ｂ ｍｏｄｕｌｕｓ Ｎ）となる出力ストリームを生成するために、整数Ａ及びＢのインタリーブされた有限体モジュラー乗算を実行するマイクロエレクトロニクスの方法及び装置であって、ｎがモジュラスオペランドレジスタにおける文字であり、ｋ文字のセグメント長さよりも大きく、▲○×▼乗算プロセスが複数の反復で実行され、各々のインタリーブされた反復で▲○×▼乗算装置に入力されるオペランドは、モジュラスＮ、乗数Ｂ、予め計算された部分的な結果Ｓ、被乗数Ａのｋ文字列セグメントからなり、セグメントはA_0文字列セグメントからA_{m-1}文字列セグメントまで進行し、各々の反復の結果は次の順番の一時的結果Ｓに▲○＋▼加

算され、反復結果の最初に発生する文字はゼロであるマイクロエレクトロニクスの方法及び装置を説明している。

【００８５】

典型的には、乗算装置に供給される四つの直列▲１▼ビット文字レジスタ、第一にＢ、第二にＳ、第三にＮ、好ましくはＡがあり、乗算器に効率的にロードされるように構成される。

【００８６】

典型的には、ＭＡＰの内部レジスタに収容されない長いオペランドでの計算のために、ＣＰＵは、そのアクセス可能なメモリーからオペランドをロードすることができる。

【００８７】

典型的には、これらの主メモリレジスタは、夫々、乗数値、部分的な結果値、モジュラスＮを保存するように動作するオペランドを保存し出力する。

モジュラー乗算装置は、反復▲○×▼乗算プロセスの間、順番に複数の被乗数値の一つ又は二つを▲○＋▼アキュムレーションデバイスに▲○＋▼加算し、順番に前記Ｂレジスタの第一の値からの入力と、各々の反復において最初の右側ゼロ文字を出力させる乗数としての「即座の」予測値Y_0の第二の値からの入力と、前記モジュラスＮレジスタの第三の値からの入力とを乗数として受け入れるように動作する。

【００８８】

被乗数並列レジスタは、Ａ，Ｂ，Ｎレジスタ資源からの値を順番に受け入れ、続いて被乗数ゼロ強制値Y_0をも受け入れるように動作する。

【００８９】

ゼロ強制Y_0検出装置は、演算の第一段階の間、乗数となるように動作し、各々の反復乗算の第二段階の間、被乗数となるように動作する二進列を生成するように動作する。

【００９０】

第一段階で▲○＋▼アキュムレーションデバイスに入れ替えられる被乗数値は、第一のゼロ値、被乗数Ａのｋ文字列セグメントである第二の値A_i、モジュラスＮの最初に発生するｋ文字である第三の値N_0からなり、四つの値のうちの一つになり得る。

図６のように、四番目のプリロードバッファがある場合は、N_0値は概して、乗算の開始時に入れ替えられる。

そして、Ａのｋ文字スライスが入力される際は、A_i値はN_0値に直列的に加算され、四番目のバッファに保存される。

【００９１】

概して単一のｋ文字モジュラスでの計算の場合、Ｓレジスタ若しくは一時的結果値Ｓの必要性はない。

オペランドが２ｋ文字又はそれより長い場合、A_iスライスを進行させつつ、処理が反復的でなければならない。

平方演算で、Ｂのスライスは、概してＢストリームから即座に回収され（ｂｅ ｓｎａｒｅｄ）、A_iプリロードバッファにプリロードされる。

【００９２】

乗算操作の第一の反復で、一時的結果はゼロである。

【００９３】

先の反復からのその後の一時的結果は、次の順番の反復で部分的な結果を生成するために、▲○＋▼アキュムレーションデバイスから発生する値に▲○＋▼加算されるように動作する。

【００９４】

第二段階で▲○＋▼アキュムレーションデバイスに順番に入力される被乗数値は、第一に擬レジスタ値（ｐｓｅｕｄｏ ｒｅｇｉｓｔｅｒ ｖａｌｕｅ）であるゼロ、第二に第一段階から残存するオペランドA_i、第三にＮモジュラスの残存する文字を乗算し続けるように動作する第一段階で予測されたY_0値である。

【００９５】

第一段階で乗算装置に入力される乗数値は、最初に発生する文字列B_0であり、最初に発生するＢオペランドの文字列セグメントであり、第二段階でプリロードされた被乗数バッファに生成すると同時に文字毎にロードされる予測されたY_0文字列からなる第二の▲○×▼乗数値と▲○×▼乗算するのと並行して乗算する。

【００９６】

第二段階の間、装置に入力される二つの乗数値は夫々、Ｂで表されるＢオペランドからの左側ｎ－ｋ文字の値と、Ｎで表されるＮモジュラスの左側ｎ－ｋ文字である。

【００９７】

第三段階は、▲○＋▼アキュムレーションデバイスに残存する結果値の左側セグメントを転送するように動作する装置のフラッシュアウトである。

これは、乗数入力において供給されるゼロ文字によってなされる単一クロックのデータダンプか、又はまたは単純な直列アンロードのいずれかである。

【００９８】

ダンプが並列のダンプである場合、その結果（ｒｅｓｕｌｔ）がモジュラスによって更なる縮小を必要とするか否かを決定するために比較する手段がある。

【００９９】

本発明におけるより革新的な拡張のうちの一つは、ＧＦ（2^q）における逆モード乗算である。

この演算において加算器セル間の相互作用がないため、積のＭＳ側から始めて乗算及び縮小を実行することができることによって、従来のモンゴメリ乗算における右側シフトを実行することに相当する無視されたゼロにより発生する厄介な寄生（ｐａｒａｓｉｔｅ）なしでモジュラー縮小された解である積を有する。

【０１００】

自動ゼロ強制を可能にする第二の革新点は、ただ一つのビットが一度に予測される装置を記載する特許出願Ｐ２のY_0関数の拡張である。

そこで$J_{0\ 0}$ビットは、単一ビットの排他的論理和を求められた値に掛けられるだけであった。

奇数の逆数及びその負の値は、奇数を発生させる。

これは、▲ｌ▼＝１でのJ_0値を計算するために、ルックアップテーブル又はランダム論理回路を実装して保存した。

J_0は、近似していない記数法においては異なる数量であることに留意すべきである。

我々は、両方の関連した数値体で、この拡張において、Y_0値がどのように誘導され得るかについて示した。

【０１０１】

以下の記載は、被乗数の最初に発生する値、Ｂ乗数の現在の入力、▲○＋▼アキュムレーションデバイスからの桁上げ値、▲○＋▼アキュムレーションデバイスからの▲○＋▼加算値、先に計算された部分的な結果からの現在の値、先の部分的な結果に▲○＋▼アキュムレーションデバイスからの結果を▲○＋▼加算する▲○＋▼加算器からの桁上げ値を用いてY_0値を予測するように動作する回路要素を記載する。

【０１０２】

別の表現をすれば、ゼロ強制関数を制御するように動作する六つの値は、

ｉ．　Ａｉレジスタの右側文字にＢストリームのB_d文字を掛けた▲ｌ▼ビット毎の$2^{▲l▼}$を法とする▲○×▼乗算の結果の▲ｌ▼ビットのS_{out}ビット　A_0　▲○×▼　B_d　ｍｏｄ　$2^{▲l▼}$、

ｉｉ．　▲○＋▼アキュムレーションデバイスから最初に発生する桁上げ文字▲Ｓ▼（CO_0）、

ｉｉｉ．　▲○＋▼アキュムレーションデバイスの右側の発生セルからの第二からの▲ｌ▼ビットのS_{out}文字SO_1、

ｉｖ．　Ｓストリームからの次の順番の文字値S_d、

ｖ．　Ｚ出力全加算器からの▲ｌ▼ビット桁上げ文字▲Ｓ▼（CO_Z）、

ｖｉ．　N_0モジュラス被乗数レジスタにおける右側文字の負の逆数である▲ｌ▼ビットのJ_0値、である。
ここでA_0 ▲○×▼ B_d ｍｏｄ $2^{▲l▼}$、▲Ｓ▼（CO_0）、SO_1、S_dの値は、▲ｌ▼ビットのゼロ文字列を発生させるための共に文字に▲○＋▼加算される文字であり、有効なY_0ゼロ強制予測文字を出力するために「即座に」J_0文字によって▲○×▼乗算される。
Ｐ１の記載にあるように、出力がモジュラー縮小されなければならないか否かを決定するためには、センサが出力結果をモジュラスＮと比較するように動作し、その機構は、結果レジスタの出力に第二の減算器を動作させることにより、出力された結果値と限定された合同であるモジュラー縮小された値を出力するように動作し、より小さな結果に第二の記憶装置を割り当てる必要を排除する。
【０１０３】
乗算を実行するように構成される単独の▲○＋▼アキュムレーションデバイス及び予測ゼロ強制機構は共に、一連のインタリーブされた▲○×▼モジュラー乗算及び平方を実行するように動作する。
全装置は、従来のモンゴメリ法における三つの整数乗算と同等のものを実行し、J_0はB_0▲○×▼A_i及びS_iの最初のｋ文字加算を乗算し、最後にＮを乗算すべくY_0を予測するｋ文字装置である。
【０１０４】
ＳｕｐｅｒＭＡＰが乗算の最後の反復を計算している間、次の乗算の第一のスライスは、即座にプリロードレジスタバッファ手段にプリロードされ得る。
この値は、図１若しくは図５のレジスタバンクにおけるレジスタセグメントの一つに残存する先の乗算の結果か、又は被乗数のスライスである。
【０１０５】
二つの被乗数の▲○＋▼加算であるプリロードされた値は、ＧＦ（2^q）計算でのみ、ｋ文字レジスタに▲○＋▼加算される。
ＧＦ（ｐ）計算においては、更なる桁上げビットを備えなければならない。
【０１０６】
特に非常に長いモジュラスでは、ＳｕｐｅｒＭＡＰに隣接したバッファ及びレジスタは、概して不十分なメモリ資源を有する。
プリロードバッファにオペランドを直接ロードする手段が提供され、ＣＰＵのメモリマップ（ｍｅｍｏｒｙ ｍａｐ）にオペランドを保存するように動作する。
逆フォーマット乗算では、ＣＰＵから入力された語（ｗｏｒｄ）のビットオーダーは、概してデータ入出力装置において逆にされる。
【０１０７】
図面において、太線は、ｋ文字（ｋ▲ｌ▼ビット）の広い並列バスラインを示す。
より薄い隣接した信号ラインは、▲ｌ▼ビットの広いラインを表す。
ほとんどの制御ラインは表示されていないが、ここに含まれるものは、概して手順を理解するのに必要であり、概して一点鎖線で示されている。
【０１０８】
（発明を実施するための最良な形態）
図面において、太線は、ｋ文字（ｋ▲ｌ▼ビット）の広い並列バスラインを示す。
より薄い隣接した接続信号ラインは、▲ｌ▼ビットの広いラインを表す。
概して、制御ラインは表示されていないが、手順を理解するのに好ましくは必要なものは、概して一点鎖線で示されている。
【０１０９】
図１，２は共に、本発明の好ましい実施例によって構成され作動する直並列算術論理演算ユニット（ＡＬＵ）のブロック略図である。
図１，２の装置は、好ましくは以下の部品を含む。
単一マルチプレクサ（ｓｉｎｇｌｅ ｍｕｌｔｉｐｌｅｘｅｒ）－一つの信号又は文字の

ストリームを多数の信号入力から選択し、この選択された信号を単一の出力に当てる制御切換素子。
マルチプレクサは、Ｍ１～１３の符号を付され、より大きな素子が本来備える素子である。
【０１１０】
マルチプレクサ及びプリアッダー（ｐｒｅ－ａｄｄｅｒ；予加算器）３９０は、ｋ▲ｌ▼＋１のマルチプレクサの配列であり、四つのｋ若しくはｋ＋１文字入力のいずれがＣＳＡ（ｃａｒｒｙ ｓａｖｅ ａｃｃｕｍｕｌａｔｏｒ；桁上げ保存アキュムレーター）４１０に加えられるかを選択する。
【０１１１】
Ｂ（７０，８０）、S_A（１３０）、S_B（１８０）、Ｎ（２００，２１０）は、好ましい実施例における四つの主要な直列レジスタである。
S_Aは、概念的に及び実際に冗長であるが、非常に長い番号の計算をかなり速めることができ、特にモジュラスの長さが２・ｋ・ｍ文字長さである場合には、揮発性記憶装置で保存する。
【０１１２】
直列加算器及び直列減算器は、二つの直列文字入力及び一つの直列文字出力を有し、二つの長い文字列を加算若しくは減算する論理素子である。
部品９０及び５００は減算器であり、３３０及び４６０は直列加算器である。入力から出力への伝搬時間は、非常に短い。
直列減算器９０及び５００は概して、B^*がＮ以上である場合、B^*をＢに減少させ、及び／又は、S^*がＮ以上である場合、S^*をＳに減少させる。
直列減算器４８０は、B^*がＮ以上であるか否かを検知するコンパレータ部品の一部として利用される。
全加算器３３０は、二つの文字ストリームを加算し、ロードバッファ（ｌｏａｄ ｂｕｆｆｅｒ）３４０に、ロードバッファ２９０及び３２０の値の合計と同じ値を供給する。
【０１１３】
高速ローダ及びアンローダ（１０，２０及び３０，４０）は夫々、ＣＰＵ制御装置からのデータフローを速める装置である。
これらの装置は概して、他の直接メモリアクセス部品の必要性を排除する。
符号２０，４０は、逆フォーマット（ｒｅｖｅｒｓｅ ｆｏｒｍａｔ）のＧＦ（2^q）の乗算のためのデータ語を逆にする必要がある際に、データ語を逆にするためのものである。
【０１１４】
本算術論理演算ユニット装置は直列供給されるシストリックプロセッサ（ｓｙｓｔｏｌｉｃ ｐｒｏｃｅｓｓｏｒ）であるので、データ入力（Ｄａｔａ ｉｎ）５０はパラレルイン－シリアルアウト方式の装置であり、データが並列に供給され、直列に処理される。
【０１１５】
データ出力（Ｄａｔａ ｏｕｔ）６０は、結果をコプロセッサから出力するためにシリアルイン－パラレルアウト方式の装置である。
商生成器（ｑｕｏｔｉｅｎｔ ｇｅｎｅｒａｔｏｒ）は、図２のその部分であるが、それは分割機構の各々の反復で商文字を生成する。
【０１１６】
Ｂｄ上のフラッシュ信号（ｆｌｕｓｈ ｓｉｇｎａｌ）２４０、S^*d上のフラッシュ信号２５０、Ｎｄ上のフラッシュ信号２６０は、直前のｋ＋１文字が、確実にＣＳＡをフラッシュアウトできる（ｆｌｕｓｈ ｏｕｔ）ような信号である。
第二の実施例は、第二段階の終わりにＲデータを一致させ、ＣＳＡをフラッシュアウトするために単一のパラレルデータのダンプ（ｄｕｍｐ）を行う。
【０１１７】
ロードバッファＲ１，Ｒ２，Ｒ３（２９０，３２０，３４０）は、三つの可能な０以上の

被乗数の組合せを受信するのに適したシリアルイン－パラレルアウト方式の送りレジスタ（ｓｈｉｆｔ　ｒｅｇｉｓｔｅｒ；シフトレジスタ）である。

【０１１８】

ラッチＬ１，Ｌ２，Ｌ３（３６０，３７０，３８０）は、ロードバッファからの出力を受信し、それにより、次段階のデータが好ましくはＬ１，Ｌ２，Ｌ３にラッチされる前に、ロードバッファがこのデータを処理することが時間的に可能となる。

ラッチＬ０は概して３９０への「仮想」の一定の全ゼロの入力であり、概してラッチされた論理（ｌａｔｃｈｅｄ　ｌｏｇｉｃ）では実行されない。

【０１１９】

Y_0検知（Y_0　ｓｅｎｓｅ）４３０は論理回路であり、ＬＳ（ｌｅａｓｔ　ｓｉｇｎｉｆｉｃａｎｔ；最下位）ゼロのｋ文字列が▲○×▼乗算のＺで終了するために、モジュラスが累算される回数を決定する。

【０１２０】

一文字遅延装置（Ｏｎｅ　ｃｈａｒａｃｔｅｒ　ｄｅｌａｙ　ｄｅｖｉｃｅ）１００，２２０，２３０は、計算の同期のために図１のデータ作成装置と図１のデータ処理装置との間に収容されるように、夫々のデータストリームに挿入される。

【０１２１】

ｋ文字遅延送りレジスタ４７０はＮを同期させ、減算器は、Ｎより大きな比較のために右側の出力のゼロ文字列を無視した後の結果からＮを減算する。

【０１２２】

ＣＳＡは、従来のような直列／並列乗算器の入力にラッチされるような単一の値の代わりに、三つの異なるゼロより大きな値が加算されるということを除いては、直列／並列乗算器とほとんど同一である。

多項式ベースの計算において利用される場合には、「全ての桁上げに依存する」関数は使用禁止となる。

【０１２３】

最後の桁上げ挿入（Ｉｎｓｅｒｔ　Ｌａｓｔ　Ｃａｒｒｙ）４４０は、Ｓレジスタがｍ・ｋ文字長さしかないので、Ｓストリームの（ｍ・ｋ・▲ｌ▼＋１）番目のビットを挿入するために利用される。

【０１２４】

借り／桁溢れ検出（ｂｏｒｒｏｗ／ｏｖｅｒｆｌｏｗ　ｄｅｔｅｃｔ）４９０は、概して、結果が（Ｎからの）モジュラス以上であるか、又はＧＦ（ｐ）計算内であるかを検出する。

多項式ベースの計算において、第一の重要な結果のビットが一つである場合、桁溢れが検出される。

【０１２５】

制御機構は記載されていないが、好ましくは、ＧＦ（ｐ）及びＧＦ（2^q）の両方のシストリック・データ・フロー用に設定されるスイッチを有する特殊関数のための有限状態機械を有する一組のカスケード方式の計数装置であると理解される。

【０１２６】

数の素体及び合成素体（ｃｏｍｐｏｓｉｔｅ　ｐｒｉｍｅ　ｆｉｅｌｄ）におけるモジュラー型乗算のため、我々はＡ及びＢを被乗数及び乗数と定義し、ＮをＡ若しくはＢよりも概して大きなモジュラスと定義する。

Ｎはモジュラスの値が保存されるレジスタをも意味する。

Ｎは、若干の場合において、Ａより小さい。

我々は、Ａ，Ｂ，Ｎをｍ・ｋ＝ｎの文字長さのオペランドとして定義する。

各々のｋの文字群（ｋ　ｃｈａｒａｃｔｅｒ　ｇｒｏｕｐ）はセグメント（ｓｅｇｍｅｎｔ）と呼ばれ、その群のサイズは乗算装置のサイズによって定められる。

Ａ，Ｂ，Ｎは各々ｍ文字長さである。

順次手順を説明していく際の平易のために、Ａ，Ｂ，Ｎは５１２ビットの長さ（ｎ＝５１

２）と仮定し、乗数の経済的な長さ及び簡単なＣＰＵのデータ操作速度を考慮して、ｋは６４文字長さと仮定し、ｍ＝８がオペランドにおけるセグメントの数及び５１２ビットのオペランドでの平方若しくは乗算ループにおける反復の数である。
全てのオペランドは、自然数である。
より一般的には、Ａ，Ｂ，Ｎ，ｎ，ｋ，ｍは、いかなる適切な値をも仮定することができる。

【０１２７】
非モジュラー関数において、Ｎ及びＳレジスタは、他の算術オペランドの一時記憶のために利用されることができる。

【０１２８】
我々は、記号≡をモジュラー数の合同を意味するものとし、例えば、２は１６を７で割った剰余となるので、１６≡２ ｍｏｄ ７ の場合、１６は、７を法として（ｍｏｄｕｌｏ７）２と合同であると言う。
我々がＹ ｍｏｄ Ｎ≡Ｘ ｍｏｄ Ｎと記述する場合は、Ｙ，Ｘの両方ともＮより大きくてもよいが、正数のＸ，Ｙでは、剰余は同一である。
負の整数Ｙの合同がＹ＋ｕ・Ｎ（Ｎはモジュラス）であることにも留意する必要があり、Ｙの合同がＮ未満となる場合は、ｕは正の結果を与える最も小さい整数である。

【０１２９】
我々は、記号¥を、より限定された意味における合同を意味するものとして利用する。
ここに記載されるプロセスの間、値はしばしば所望の値か、若しくは所望の値にモジュラスを加えた値と等しいかのいずれかである。
例えばＸ¥２ ｍｏｄ ７の場合である。
Ｘは、２又は９と等しい。
この場合、我々は、Ｘが７を法として限定された合同（ｌｉｍｉｔｅｄ ｃｏｎｇｒｕｅｎｃｅ ｔｏ ２ ｍｏｄ ７）を有すると言う。
多項式ベースの体において、アナログは、我々がＮより大きいというところのモニック（ｍｏｎｉｃ）の値であり、モジュラスに対する排他的論理和（ＸＯＲ）を求めることにより減少される。
ＧＦ（2^q）においては、桁溢れがないので、¥の値は概して無視される。

【０１３０】
我々がＸ＝Ａ ｍｏｄ Ｎ と記述した場合、ＸはＡをＮで割った剰余と定義され、例えば、３＝４５ ｍｏｄ ７となる。

【０１３１】
数論において、モジュラー型逆数は基礎概念である。
例えば、Ｘのモジュラー型逆数はX^{-1}と記述され、Ｘ・X^{-1} ｍｏｄ Ｎ＝１により定義される。
Ｘ＝３及びＮ＝１３の場合、X^{-1}＝９であるが、これを言い換えれば、３・９を１３で割った剰余は１ということである。

【０１３２】
略語ＭＳ及びＬＳは、デジタル用語で一般に用いられているように、ビット、文字、セグメント、全オペランド値に言及する場合に最上位及び最下位を意味するように用いられる。

【０１３３】
この明細書の全体に亘って、Ｎは値Ｎ、及び、Ｎを含む送りレジスタの名称の両方を示す。
値上のアスタリスクの上付き文字は、その値が潜在的に不完全であるか変化する可能性があることを意味する。
Ａは累乗されるべき数値であり、ｎはＮオペランドの文字長さである。
初期状態設定の後、ＡがA^*に「モンゴメリ標準化された（Ｍｏｎｔｇｏｍｅｒｙ ｎｏｒｍａｌｉｚｅｄ）」（後述する$A^*=2^n \cdot A$）場合、A^*及びＮは、べき算における

中間段階を通じて一定値である。
第一の反復の間、べき算の初期状態設定の後、ＢはＡ＊に等しい。
Ｂは、最終的にべき算の所望の結果と等しくなる累算された値が存在するレジスタの名称でもある。
Ｓ＊は一時的な値を示し、Ｓ，S_A，S_Bは、Ｓの単一のＭＳビット以外の全てを保存するレジスタを示す。
（このＭＳビットに連結されるＳ＊は、Ｓと同一である。）
Ｓ（ｉ－１）は、ｉ番目の反復の最初でのＳの値を意味し、S_0は、Ｓ（ｉ）番目の値のＬＳセグメントを意味する。
【０１３４】
我々は、ＧＦ（ｐ）体（後に定義する）におけるプロセス▲Ｐ▼（Ａ・Ｂ）Ｎを、Ｐ体における乗算と、又は時には単に乗算演算と称する。
【０１３５】
我々は、並列／直列乗算器の標準的な構造を複動式の直列並列乗算器を構成するための基礎として利用してきたので、（かなり複雑な桁上げ先見加算器（ｃａｒｒｙ　ｌｏｏｋ　ａｈｅａｄ　ａｄｄｅｒ）若しくは非常に遅いリップル加算器とは対照的に）、桁上げ保存累算を基礎とする乗算器の加算部との差異を認め、桁上げ保存加算器若しくは桁上げ保存アキュムレーターと呼び、プリロード機構（ｐｒｅｌｏａｄｉｎｇ　ｍｅｃｈａｎｉｓｍ）とマルチプレクサ及びラッチを別々に取り扱うが、これらは、例えばＡ・Ｂ＋Ｃ・Ｄ＋Ｓのように、ＡとＢ及びＣとＤを同時に掛けることができ、連続的に両方の結果と先の結果であるＳとを加算でき、このアキュムレーターをより汎用的なエンジンに変換する。
更なるロジックが、モジュラー縮小（ｍｏｄｕｌａｒ　ｒｅｄｕｃｔｉｏｎ）に必要な予測された検知演算（ａｎｔｉｃｉｐａｔｅｄ　ｓｅｎｓｅ　ｏｐｅｒａｔｉｏｎ）、及び、極めて大きな数におけるモジュラー演算と通常の整数演算とを提供するのに必要な直列総和（ｓｅｒｉａｌ　ｓｕｍｍａｔｉｏｎ）を提供するために、この乗算器に加えられる。
【０１３６】
ＧＦ（ｐ）におけるモンゴメリモジュラー型乗算
以下の説明は、数のＧＦ（ｐ）におけるモンゴメリ演算について言及する。
本装置はＧＦ（2^q）における多項式ベースの数上のモンゴメリ演算に利用され得るが、計算が、全ての実行可能なオペランドが要素2^nを乗ぜられるＰ体におけるものであるので、実行においては機能が低下する。
【０１３７】
モジュラー型乗算を計算するための古典的な方法において、Ａ・Ｂ ｍｏｄ Ｎ の場合、積Ａ・Ｂの剰余は除算によって算出される。
大きなオペランドで従来の除算を実行することは、直列／並列乗算よりも機能することが困難である。
【０１３８】
モンゴメリのモジュラー縮小法（ｍｏｄｕｌａｒ　ｒｅｄｕｃｔｉｏｎ　ｍｅｔｈｏｄ）を利用して、除算は二つの予め計算された定数を利用する乗算により本質的に置換される。
本願明細書で示される手順において、一つだけ予め計算された定数があり、それはモジュラス関数である。
この定数は、算術論理演算ユニット装置を利用して計算されるか、若しくは計算され得る。
【０１３９】
この装置において利用されるようなモンゴメリプロセスは、まず簡略的に示され、その後に完全な好ましい説明がなされる。
【０１４０】
我々が、例えば１０１０００１（＝81_{10}）のような奇数（ＬＳビットが１）を有する場合、我々は常に、その奇数に他の固定補正（ｆｉｘｉｎｇ，　ｃｏｍｅｐｎｓａｔｉｏｎｇ）奇数、例えば１１１１（＝15_{10}）を加算することにより、１１１１＋１０１０

００１＝１１０００００（96_{10}）として、この奇数を偶数（単一のＬＳビットが０）に変換することができる。
この具体的事例において、我々は前もって全文字列である８１を知っており、８１に加算する二進数を容易に決定でき、我々が必要とするＬＳゼロの数を有する新たな二進数を生成できたので、五つのＬＳゼロを生成する数を見つけることができた。
この固定数は右側に１を有していなければならず、さもなければ、結果として出るＬＳ文字に影響を及ぼさない。
【０１４１】
我々のプロセスが、連続したＬＳゼロの数を有することを望み、各々のクロックサイクルで次のビットを固定しさえすればよいクロックされた直列／並列桁上げ保存プロセスである場合、次のビットが１となるようなら各々のクロックでその固定数を加えれば十分であり、予測されるビットがゼロとなるようならその固定数を加えなければ十分である。
しかし、関連した乗数のビットが１であり、Ｙ検知も１と予測する場合、ビット間の桁溢れ（倍の桁上げ）を発生させないために、この固定数は好ましくは前もって被乗数に加算され、アキュムレーターに加えられる。
【０１４２】
ここで、モジュラー演算において、我々はモジュラスによって除算される値の剰余にのみ興味があるので、モジュラスを何回でも値に加算することができ、同じ剰余を有する値をなお有することができることを知っている。
これは、$Y \cdot N = \Sigma y_i \cdot r^i \cdot N$をあらゆる整数に加えることができ、なおも同じ剰余を有することを意味する。
ここで、Ｙは、必要なｋ▲ｌ▼の右側ゼロを生成するために、モジュラスＮにおいて加算した回数である。
上記したように、我々が加えるモジュラスは奇数のみである。
（偶数モジュラスが、r^iが、ｉが偶数におけるＬＳゼロの数である場合に得られる奇数との積として定義される方法は存在する。）
【０１４３】
モンゴメリのインタリーブされた縮小は、概して保存必要量を減少させ、乗算装置の経済的なサイズを小さくする。
このことは、例えばｎ＝１０２４ビットのような一つの大きな整数に、別の同じ長さの整数を掛けるような、通常倍の長さの整数を生成するプロセスである公開鍵暗号機能を実行する際に、特に有効である。
【０１４４】
我々は、ｎのＬＳゼロと、多くともｎ＋１のＭＳ文字を有する数Ｚを有するように、乗算（若しくは平方）プロセスの間、Ｎｓ（モジュラス）においてＡ・Ｂ＝Ｘ若しくはＡ・Ｂ＋Ｓ＝Ｘに十分な回数加算することができる。
【０１４５】
これらのゼロを無視することにより、r^nで所望の結果を除算したことを留意していれば、このような数を利用し続け、ＬＳのｎ文字を無視し続けることができる。
【０１４６】
ＬＳのｎ文字を無視し、最上位のｎ（若しくはｎ＋１）文字のみを使用すると、その結果にr^nの逆数であるr^{-n}を効果的に乗算した。
我々が続けてこの結果にr^n ｍｏｄ Ｎ（若しくはr^n）を再乗算する場合、Ａ・Ｂ＋Ｓ ｍｏｄ Ｎとして、（同じ剰余を有する）所望の結果と合同な値を得る。
ここで見られるように、ＭＭ（Ｍｏｎｔｇｏｍｅｒｙ ｍｕｌｔｉｐｌｉｃａｔｉｏｎ）を利用した場合、その結果は、ＭＭによって再導入されたr^{-n}寄生要素（ｐａｒａｓｉｔｉｃ ｆａｃｔｏｒ）を克服するために、好ましくはr^{2n}を乗算される。
【０１４７】
例
Ａ・Ｂ＋Ｓ ｍｏｄ Ｎ＝（１２・１１＋１０） ｍｏｄ １３＝（１１００・１０１１＋１

０１０）$_2$ ｍｏｄ １０１１$_2$
▲ｌ▼＝１，ｒ＝２
【０１４８】
我々は、固定数がｎのＬＳビットのうち一つで必要な場合はいつでも2^i Ｎを加算する。
Ｂ　　１０１１
×　Ａ　　１１００
Ｓ　　１０１０　を加算する。
Ａ（０）・Ｂ　　００００　を加算する。
ＬＳビットの合計＝０なのでＮを加算しない。
2^0（Ｎ・０）　　００００　を加算する。
合計しシフトする　０１０１　→０のＬＳビットが桁上げ保存加算器を離れる。
Ａ（１）・Ｂ　　００００　を加算する。
ＬＳビットの合計＝０なのでＮを加算する。
2^1（Ｎ・１）　　１１０１　を加算する。
合計しシフトする　１００１　→０のＬＳビットが桁上げ保存加算器を離れる。
Ａ（２）・Ｂ　　１０１１　を加算する。
ＬＳビットの合計＝０なのでＮを加算しない。
2^2（Ｎ・０）　　００００　を加算する。
合計しシフトする　１０１０　→０のＬＳビットが桁上げ保存加算器を離れる。
Ａ（３）・Ｂ　　１０１１　を加算する。
ＬＳビットの合計＝１なのでＮを加算する。
2^3（Ｎ・１）　　１１０１　を加算する。
合計しシフトする　１０００１　→０のＬＳビットが桁上げ保存加算器を離れる。
結果は１０００１　０００$0_2$ ｍｏｄ １３＝１７・2^4 ｍｏｄ １３となる。
【０１４９】
１７は１３より大きいので、１３を減じて、１７・2^4 ≡４・2^4 ｍｏｄ １３となり、正式には2^{-n}（Ａ・Ｂ＋Ｓ） ｍｏｄ Ｎ＝９（１２・１１＋１０） ｍｏｄ １３≡４である。
【０１５０】
モンゴメリ演算において、我々はＭＳゼロでない結果である４のみを利用し、実際の結果が2^nで割られｎのゼロがＭＭの結果に押し込められたことを効果的に留意する。
【０１５１】
我々は、2^4 ｍｏｄ １３≡３により結果を効果的に乗算した（８＋２）・１３＝１０・１３において追加した。
結局、我々は不必要なゼロを利用したので、一プロセスでＡ・Ｂ＋Ｙ・Ｎ＋Ｓ－（１２・１１＋１０・１３＋１０）を実行したと言うことができ、好ましい実施例でできるだけ後述する。
【０１５２】
チェックすると、（１２・１１＋１０） ｍｏｄ １３＝１２、４・３＝１２
【０１５３】
結局、モンゴメリ乗算の結果は、2^{-n}によって乗算される所望の結果である。
【０１５４】
各々のＭＭは2^{-n}の寄生要素を残すので、前の結果を同じ乗算法を利用する所望の結果へ戻すために、我々は2^{2n}により前の結果をモンゴメリ乗算する必要があり、これをＨと呼ぶ。
【０１５５】
モンゴメリ乗算関数▲Ｐ▼（Ａ・Ｂ）Ｎは、Ｐ体で積Ａ・ＢのＮを法とする乗算を実行する。（上記の例では４を誘導した部分である。）
Ｐフィールドから通常のモジュラー体へ戻すことは、予め計算された定数Ｈを利用して、▲Ｐ▼（Ａ・Ｂ）Ｎの結果におけるＰを定めることにより実行される。

ここでＰ＝▲Ｐ▼（Ａ・Ｂ）Ｎである場合、▲Ｐ▼（Ｐ・Ｈ）Ｎ≡Ａ・Ｂ ｍｏｄ Ｎに従うことにより、二つのＰ体の乗算において通常のモジュラー乗算を行う。

【０１５６】

モンゴメリモジュラー縮小（Ｍｏｎｔｇｏｍｅｒｙ ｍｏｄｕｌａｒ ｒｅｄｕｃｔｉｏｎ）は、ｎ若しくはｎ＋１文字長さであるオペランドでの一連の乗算、加算、減算を実行することにより、ｎ若しくは２ｎ文字長さであるオペランドでの一連の乗算、除算を回避する。

全体のプロセスは、Ｎより小さいかＮと同じ結果となる。

与えられたＡ，Ｂ，奇数Ｎについては、Ａ・Ｂ＋Ｑ・ＮがｎのＬＳ文字がゼロである数となるか、又は、$Ｐ・２^{ｎ}＝Ａ・Ｂ＋Ｑ・Ｎ$となるようなＱが常にある。

【０１５７】

これは、我々がｎのＬＳ文字がゼロである（可能な１ビットの桁溢れを有する）２ｎ文字長さである表現を有することを意味する。

【０１５８】

ここで、基数$ｒ＝２^{▲ｌ▼}$で、$Ｉ・ｒ^{ｎ}≡１$ ｍｏｄ Ｎとする（Ｉは全ての奇数Ｎで存在する）。

上式の両側にＩを乗算することで、以下の合同が得られる。

式の左側からは$Ｐ・Ｉ・ｒ^{ｎ}≡Ｐ$ ｍｏｄ Ｎ（$Ｉ・ｒ^{ｎ}≡１$ ｍｏｄ Ｎより）、式の右側からはＡ・Ｂ・Ｉ＋Ｑ・Ｎ・Ｉ≡Ａ・Ｂ・Ｉ ｍｏｄ Ｎ（Ｑ・Ｎ・Ｉ≡０ ｍｏｄ Ｎより）、

従って、Ｐ≡Ａ・Ｂ・Ｉ ｍｏｄ Ｎ

これは、Ｐ体乗算が実行される毎に、寄生要素$Ｉ＝ｒ^{－ｎ}$ ｍｏｄ Ｎが導入されることをも意味する。

▲Ｐ▼演算子は、Ｐ≡Ａ・Ｂ・Ｉ ｍｏｄ Ｎ≡▲Ｐ▼（Ａ・Ｂ）Ｎとなるように定義され、これを「Ｐ体におけるＡ×Ｂの乗算」又はモンゴメリ乗算と呼ぶ。

Ｐ体からの戻しは、Ｐ・Ｈで▲Ｐ▼を演算することによって計算でき、▲Ｐ▼（Ｐ・Ｈ）Ｎ≡Ａ・Ｂ ｍｏｄ Ｎとする。

我々は、前出の合同式のＰを置換することによってＨの値を誘導することができる。

我々は、▲Ｐ▼（Ｐ・Ｈ）Ｎ≡（Ａ・Ｂ・Ｉ）（Ｈ）（Ｉ） ｍｏｄ Ｎを見出した。

（Ａ・Ｂ・Ｉ←Ｐ、Ｈ←Ｈ、Ｉ←であり、あらゆる乗算演算が寄生要素Ｉを導入することを参照）

【０１５９】

Ｈが$Ｉ^{２}$の逆数と合同である場合、合同式は有効であるので、$Ｈ＝Ｉ^{－２}$ ｍｏｄ $Ｎ≡ｒ^{２ｎ}$ ｍｏｄ Ｎ（Ｈは、Ｎの関数であり、Ｈパラメーターと呼ぶ。）

【０１６０】

従来のモンゴメリ法において、Ａ・Ｂでの▲Ｐ▼演算子を定めるためには、予め計算された定数Ｊを用いて以下のプロセスを行う。

１）Ｘ＝Ａ・Ｂ

２）$Ｙ＝（Ｘ・Ｊ）$ ｍｏｄ $ｒ^{ｎ}$（ｎのＬＳ文字のみが必要）

３）Ｚ＝Ｘ＋Ｙ・Ｎ

４）$Ｓ＝Ｚ／ｒ^{ｎ}$（Ｊの必要条件は、Ｚを$ｒ^{ｎ}$によって割り切れるようにすることである。）

５）Ｐ￥Ｓ ｍｏｄ Ｎ（Ｓ≧Ｎの場合、ＮはＳから減算される。）

最後に第５段階で、

Ｐ￥▲Ｐ▼（Ａ・Ｂ）Ｎ

［Ｎの減算の後、必要ならばＰ＝▲Ｐ▼（Ａ・Ｂ）Ｎ］

上記に続いて、$Ｙ＝Ａ・Ｂ・Ｊ$ ｍｏｄ $ｒ^{ｎ}$（ｎのＬＳ文字のみを使用）

$Ｚ＝Ａ・Ｂ＋（Ａ・Ｂ・Ｊ$ ｍｏｄ $ｒ^{ｎ}）・Ｎ$

【０１６１】

Ｚを$ｒ^{ｎ}$で割り切れるようにするため（ＺのｎのＬＳ文字は好ましくはゼロである）、以

下の合同式がある。
［Ａ・Ｂ＋（Ａ・Ｂ・Ｊ ｍｏｄ ｒ^{n}）・Ｎ］ ｍｏｄ ｒ^{n} ≡０
【０１６２】
この合同式が存在するために、Ｎ・Ｊ ｍｏｄ ｒ^{n} は－１と合同であるか、又は、
Ｊ≡－Ｎ$^{-1}$ ｍｏｄ ｒ^{n}
我々は、定数Ｊを見つけた。
【０１６３】
従って、ＪはＮのみの関数である予め計算された定数である。
しかし、ＭＭの結果を出力するマシンにおいて、ＬＳ文字列で出力される文字が別にゼロであった各々の場合で、文字毎にＮに加えられる用意がなされているべきであり、それにより、Ｙが配線論理を利用して文字毎に検出されるので、Ｊを予め計算し、続いてＹ＝Ａ・Ｂ・Ｊ ｍｏｄ ｒ^{n} を計算する必要性はなくなる。
我々は、この方法が奇数のＮｓについてのみ機能できることを述べた。
【０１６４】
従って、明らかなように、上述したプロセスは、▲Ｐ▼（Ａ・Ｂ）Ｎを得るために、与えられたＡ，Ｂ，Ｎ及び予め計算された定数で三つの乗算、一つの加算、多くても一つの減算を用いる。
この結果、同じプロセス及び予め計算された定数Ｈ（モジュールＮの関数）を利用して、我々はＡ・Ｂ ｍｏｄ Ｎを見つけることができる。
ＡはＢと等しくてもよいので、この基本演算器は、モジュラー演算において平方又は乗算する装置として使用できる。
【０１６５】
インタリーブされたモンゴメリモジュラー型乗算
上記では、全てｎ文字長さであるオペランド及び２ｎ＋１文字の保存空間を必要とした結果の乗算を含んだモジュラー型乗算の方法を説明している。
【０１６６】
Ｐ１に記載したモンゴメリのインタリーブされた縮小を利用して、より短いオペランド、レジスタ及びハードウェア乗算器で乗算演算を実行することが可能であり、比較的少ない論理ゲートでの電子装置のインプリメンテーション（ｉｍｐｌｅｍｅｎｔａｔｉｏｎ）を可能にする。
【０１６７】
まず、我々は、インタリーブの各々の反復で、Ｊ$_{0}$ 定数を利用してＮが加算される回数を計算する場合に、装置がどのように機能するかについて説明する。
後で我々は、Ｙ$_{0}$ の配線誘導（ｈａｒｄｗｉｒｅ ｄｅｒｉｖａｔｉｏｎ）を利用したインタリーブする方法を説明するが、これは各々の乗算のＪ$_{0}$＋段階を排除し（後述の例の（２）参照）、二つの別々の直列／乗算器を、同様のシリコン資源を利用して、倍速以上の速度でＡ・Ｂ＋Ｃ・Ｎ＋Ｓを実行できる新たな単一の汎用乗算器に組み込むことを可能にする。
【０１６８】
ｋ文字の乗数を利用して、ｎ文字にｍのセグメントを有する、即ちｍ・ｋ＝ｎとなるようにｋ文字長さのセグメントを定めることは便利である。
Ｊ$_{0}$ は、ＪのＬＳセグメントである。
従って、Ｊ$_{0}$ ≡－Ｎ$_{0}$$^{-1}$ ｍｏｄ ｒ^{k} である（Ｎとして存在するＪ$_{0}$ は奇数である）。
【０１６９】
Ｊ及びＪ$_{0}$ 定数は、所定数の最下位ゼロを有するように、減少していない出力を定める場合に、モジュラスを加える回数を伝える補正数（ｃｏｍｐｅｎｓａｔｉｎｇ）であることに留意すべきである。
次の出力の直列ビットを容易に決めることができるように、次の中間結果に常にモジュラス（常に奇数）を加算するので、我々は本発明に係るシリアルデバイスの更なる利点を後述する。

これは、この加算がなければＣＳＡに存在するＬＳ直列ビットである出力文字が「１」であった場合であり、それにより、先の中間結果にさえモジュラスを加算し、出力文字列に別のＬＳゼロを約束する。
どんなにモジュラスが結果に加算されようとも、剰余が一定であるので、合同が維持されることを思い出すべきである。
【０１７０】
モンゴメリのインタリーブされた縮小の従来の使用において、▲Ｐ▼（Ａ・Ｂ）Ｎは、（１）～（５）の段階で説明されるように、ｍ回の反復で定められる。
まずＳ（０）＝０とする（第一の反復の最初でのＳの￥の値）。
ｉ＝１，２では、ｍ：
１）$X = S(i-1) + A_{i-1} \cdot B$（$A_{i-1}$はＡのｉ－１番目の文字であり、Ｓ（ｉ－１）は、ｉ番目の反復の最初でのＳの値である。）
２）$Y_0 = X_0 \cdot J_0 \bmod r^k$（$X_0 \cdot J_0$の積のＬＳｋ文字）
（このプロセスは、ｋのＬＳ文字のみ、例えば最下位６４文字のみを使用し計算する。）
直列マシンにおいて、Y_0は文字毎に予測されるから、好ましいインプリメンテーションにおいて、この段階は不要である。
３）$Z = X + Y_0 \cdot N$
４）$S(i) = Z / r^k$
（ＺのｋのＬＳ文字は常にゼロであるので、Ｚは常にｒで割り切れる。
ＺのＬＳｋ文字は全てのゼロであるので、この除算はｋ文字の右側へのシフト（ｓｈｉｆｔ；桁送り）に相当し、または、回路に示されるように、ＺのＬＳｋ文字は単に無視される。
５）$S(i) = S(i) \bmod N$（Ｎは、Ｎより大きなそれらのＳ（ｉ）から減算されることになる）。
最後に、（必要な場合はＮの減算の後）最後の反復で、$C = S(m) = ▲P▼(A \cdot B)N$
【０１７１】
$F = A \cdot B \bmod N$を誘導するために、Ｐ体計算▲Ｐ▼（Ｃ・Ｈ）Ｎが実行される。
【０１７２】
好ましい実施例において、全てのＳ（ｉ）でＳ（ｉ）が２Ｎより小さくなることを知ることが望ましい。
これは、最後の結果（Ｓ（ｍ））が常に、多くとも一回のＮの減算で、Ｎより少ない数量に減算され得ることをも意味する。
【０１７３】
我々は、以下のプロセスにおいて利用されるオペランドでそのことを観察する。
即ち、$S(i-1) < r^{n+1}$
（一時レジスタは、Ｂ又はＮレジスタよりも１ビット長くてもよい）、
$B < N < r^n$、及び、$A_{i-1} < r^k$
定義により、$S(i) = Z / r^k$（可能な減算前のプロセスの終わりのＳの値）
全てのＺで、$Z(i) < r^{n+k+1}$
$X_{max} = S_{max} + A_i \cdot B < r^{n+1} - 1 + (r^k - 1)(r^n - 1)$
$Q_{max} = Y_0 N < (r^k - 1)(r^n - 1)$
従って、$Z_{max} < r^{k+n+1} - r^{k+1} + 1 < r^{k+n+1} - 1$
Z_{max}はr^kで割り切れるので、
$S(m) < r^{n+1} < -r^1$
$N_{min} > r^n - r$なので、$S(m)_{max}$は常に２・N_{min}よりも小さく、従って、一回の減算が最後の結果で必要なだけである。
$S(m)_{max} - N_{min} = (r^{n+1} - r^1 - 1) - (r^n - 1) = r^n - 4 < N_{min}$
【０１７４】

モンゴメリのインタリーブされたモジュラー型乗算の例
十六進数での以下の計算は、インタリーブされた方法の意味を明らかにする。
Ｎ＝ａ５９（モジュロ（ｍｏｄｕｌｏ））、Ａ＝９９ｂ（乗数）、Ｂ＝５ｃ３（被乗数）、ｎ＝１２、 ｒ＝２）（Ｎの文字長さ）、ｋ＝４（乗数の文字におけるサイズ、更にはセグメントのサイズ）、ｎ＝ｋ・ｍなのでｍ＝３。
７・９≡－１ ｍｏｄ １６及びＨ≡$2^{2\cdot 12}$ ｍｏｄ ａ５９≡４４ｂなのでＪ$_0$＝７。
【０１７５】
期待される結果は、Ｆ≡Ａ・Ｂ ｍｏｄ Ｎ≡９９ｂ・５ｃ３ ｍｏｄ ａ５９≡３７５８１１ ｍｏｄ ａ５９≡２２０$_{16}$である。
【０１７６】
最初にＳ（０）＝０で、
段階１
Ｘ＝Ｓ（０）＋Ａ$_0$・Ｂ＝０＋ｂ・５ｃ３＝３ｆ６１
Ｙ０＝Ｘ$_0$・Ｊ$_0$ ｍｏｄ r^k＝７（Ｙ$_0$は、ＳｕｐｅｒＭＡＰにおいて予測される配線）
Ｚ＝Ｘ＋Ｙ$_0$・Ｎ＝３ｆ６１＋７・ａ５９＝８７ｄ０
Ｓ（１）＝Ｚ／r^k＝８７ｄ
段階２
Ｘ＝Ｓ（１）＋Ａ$_1$・Ｂ＝８７ｄ＋９・５ｃ３＝３ｃ５８
Ｙ$_0$＝Ｘ$_0$・Ｊ$_0$ ｍｏｄ r^k＝８・７ ｍｏｄ 2^4＝８（予測される配線）
Ｚ＝Ｘ＋Ｙ$_0$・Ｎ＝３ｃ５８＋５２ｃ８＝８ｆ２０
Ｓ（２）＝Ｚ／r^k＝８ｆ２
段階３
Ｘ＝Ｓ（２）＋Ａ$_2$・Ｂ＝８ｆ２＋９・５ｃ３＝３ｃｃｄ
Ｙ$_0$＝ｄ・７ ｍｏｄ 2^4＝ｂ（予測される配線）
Ｚ＝Ｘ＋Ｙ$_0$・Ｎ＝３ｃｃｄ＋ｂ・ａ５９＝ａｅａ０
Ｓ（３）＝Ｚ／r^k＝ａｅａ
Ｓ（３）＞Ｎなので、
Ｓ（ｍ）＝Ｓ（３）－Ｎ＝ａｅａ－ａ５９＝９１
従って、Ｃ＝▲Ｐ▼（Ａ・Ｂ）Ｎ＝９１$_{16}$
Ｐ体からの戻しは、▲Ｐ▼（Ｃ・Ｈ）Ｎを計算することによって実行される。
再び最初にＳ（０）＝０で、
段階１
Ｘ＝Ｓ（０）＋Ｃ０・Ｈ＝０＋１・４４ｂ＝４４ｂ
Ｙ$_0$＝ｄ（スーパーマップにおいて予測される配線）
Ｚ＝Ｘ＋Ｙ$_0$・Ｎ＝４４ｂ＋８６８５＝８ａｄ０
Ｓ（１）＝Ｚ／r^k＝８ａｄ
段階２
Ｘ＝Ｓ（１）＋Ｃ$_1$・Ｈ＝８ａｄ＋９・４４ｂ＝２ｆ５０
Ｙ$_0$＝０（スーパーマップにおいて予測される配線）
Ｚ＝Ｘ＋Ｙ$_0$・Ｎ＝２ｆ５０＋０＝２ｆ５０
Ｓ（２）＝Ｚ／r^k＝２ｆ５０
段階３
Ｘ＝Ｓ（２）＋Ｃ$_2$・Ｈ＝２ｆ５＋０・４４ｂ＝２ｆ５
Ｙ$_0$＝３（スーパーマップにおいて予測される配線）
Ｚ＝Ｘ＋Ｙ$_0$・Ｎ＝２ｆ５＋３・ａ５９＝２２００
Ｓ（３）＝Ｚ／r^k＝２２０$_{16}$
それは、９９ｂ・５ｃ３ ｍｏｄ ａ５９の期待値である。
【０１７７】
各々の段階で我々がｋのＬＳを無視したとすれば、本質的にｎのＭＳ文字をr^kによって

乗算することとなる。
同様に、各々の段階で、乗数のｉ番目のセグメントはr^{ik}によって乗算される数であり、それにＳ（ｉ）と同位（ｓａｍｅ　ｒａｎｋ）を与える。
【０１７８】
J_0定数を知るための潜在的な値がある別の好ましい実施例において、$A_i \cdot B + S = 1$である場合、$Y_0 = -N_0^{-1} = J_0$である。
【０１７９】
べき算（Ｅｘｐｏｎｅｎｔｉａｔｉｏｎ）
『ディー・クヌース（Ｄ．Ｋｎｕｔｈ）著、「コンピュータプログラミングの技法（Ｔｈｅ　ａｒｔ　ｏｆ　ｃｏｍｐｕｔｅｒ　ｐｒｏｇｒａｍｍｉｎｇ）」、第２巻、準数値算法（Ｓｅｍｉｎｕｍｅｒｉｃａｌ　ａｌｇｏｒｉｔｈｍｓ）、アディソン・ウェズリー（Ａｄｄｉｓｏｎ－Ｗｅｓｌｅｙ）、リーディング、マサチューセッツ州、１９８１年、ｐ．４０７』（以下クヌースと称す）で、シーケンスの以下の導出は、平方及び乗算のシーケンスを説明しており、べき剰余（ｍｏｄｕｌａｒ　ｅｘｐｏｎｅｎｔｉａｔｉｏｎ）を実行する。
【０１８０】
この装置はＰ体において平方及び乗算することができるので、モンゴメリ定数を予め計算した後（$H = 2^{2n}$）、我々は$C = A^E$ ｍｏｄ Ｎを計算する。
Ｅ（ｊ）はべき指数Ｅの二進法におけるｊビットを意味し、指数１のＭＳビットから始まり、指数ｑのＬＳビットで終わるとすると、奇数のべき指数で以下のように累算できる。
A^* ￥ ▲Ｐ▼（Ａ・Ｈ）Ｎ
A^*は、$A \cdot 2^n$に等しい。
$B = A^*$
ｊ＝２～ｑ－１で（ＦＯＲ ｊ＝２ ＴＯ ｑ－１）
Ｂ ￥ ▲Ｐ▼（Ｂ・Ｂ）Ｎ
Ｅ（ｊ）＝１ならば、Ｂ ￥ ▲Ｐ▼（Ｂ・A^*）Ｎである
（もし Ｅ（ｊ）＝１ならば Ｂ ￥ ▲Ｐ▼（Ｂ・A^*）Ｎ
ＦＯＲループの終わり（ＥＮＤＦＯＲ）
Ｂ ￥ ▲Ｐ▼（Ｂ・Ａ）Ｎ　　Ｅ（０）＝１
Ｂは、2^nを掛けられる最後の一時的な値であり、Ａは元々のＡである。
$C = B$
もしＣ≧Ｎならば、Ｃ＝Ｃ－Ｎ
最後の反復の後、値ＢはA^E ｍｏｄ Ｎに対する￥であり、Ｃは最終の値である。
【０１８１】
明らかにするために、以下の例を利用することとする。
Ｅ＝１０１１ → Ｅ（１）＝１、Ｅ（２）＝０、Ｅ（３）＝１、Ｅ（４）＝１
ｑ＝４でA^{1011} ｍｏｄ Ｎを見つけるために、
$A^* =$▲Ｐ▼（Ａ・Ｈ）Ｎ＝$A \cdot I^{-2} I = A \cdot I^{-1}$ ｍｏｄ Ｎ
$B = A^*$
ｊ＝２～ｑで（ＦＯＲ ｊ＝２ ｔｏ ｑ）
Ｂ＝▲Ｐ▼（Ｂ・Ｂ）Ｎより、$A^2 (I^{-1})^2 \cdot I = A^2 \cdot I^{-1}$
Ｅ（２）＝０　　$B = A^2 \cdot I^{-1}$
ｊ＝３　　$B =$▲Ｐ▼$(B \cdot B) N = A^2 (I^{-1})^2 \cdot I = A^4 \cdot I^{-1}$
Ｅ（３）＝１　　$B =$▲Ｐ▼$(B \cdot A^*) N = (A^4 \cdot I^{-1})(A \cdot I^{-1}) \cdot I = A^5 \cdot I^{-1}$
ｊ＝４　　$B =$▲Ｐ▼$(B \cdot B) N = A^{10} \cdot I^{-2} \cdot I = A^{10} \cdot I^{-1}$
Ｅ（４）が奇数だったので、寄生要素I^{-1}を取り除くために、最後の乗算はＡによるものである。
$B =$▲Ｐ▼$(B \cdot A) N = A^{10} \cdot I^{-1} \cdot A \cdot I = A^{11}$
$C = B$

【０１８２】
逆数プロセスによってＨパラメータを計算する方法は、米国特許第５５１３１３３号明細書に記載されている。
【０１８３】
次に図３を参照するが、これは本発明がスマートカード及び他のセキュリティ装置に実装される方法を示すブロック略図である。
内部バス５００は、ＣＰＵ５０２、ＲＡＭ５０４、不揮発性メモリ５０６、制御アクセスＥＥＰＲＯＭ（ｃｏｎｔｒｏｌｌｅｄ ａｃｃｅｓｓ ＥＥＰＲＯＭ）５０８、モジュラー演算コプロセッサ５１０を含む部品と連結する。
ここに示すように、コプロセッサ５１０は、データレジスタ５１２及び制御レジスタ５１４を経由して、内部バス５００に接続される。
制御アクセスＥＥＰＲＯＭ５０８は、アドレス及びデータラッチ手段５１６、並びに、制御及びテストレジスタ５１８を経由して接続される。
例えば物理シーケンスランダム生成器（ｐｈｙｓｉｃａｌ ｓｅｑｕｅｎｃｅ ｒａｎｄｏｍ ｇｅｎｅｒａｔｏｒ）５２０、セキュリティ論理（ｓｅｃｕｒｉｔｙ ｌｏｇｉｃ）５２２、スマートカードインターフェース回路５２４、外部ポートインターフェース回路５２６のような多様な他の装置が夫々、バスに接続されてもよい。
【０１８４】
ＲＳＡデジタル署名を確認するような暗号プログラムが実行される場合に、べき剰余のようなモジュラー演算機能を必要とする。
暗号関数を呼び出す暗号プログラムは、好ましくはＣＰＵ５０２上で実行される。
【０１８５】
次に図４を参照するが、これはスマートカードにおいて利用するための本発明の実装の別のブロック略図である。
図３において示されるものと同じ部品は、同じ符号を与えられているが、本実施例の理解に必要なもの以外は、ここでは説明しない。
図４において、ＣＰＵ５０２は外部アキュムレーター７３５０を備えて示される。
演算コプロセッサからのデータをアンロードすることによりＳＭＡＰから記憶装置への直接データ転送を可能にする一方で、データ使用禁止スイッチ７３４０が、データバス５００からＣＰＵアキュムレーターを分離する。
【０１８６】
図５は、J_0生成器と共に、図２，６，７のコプロセッサにて図示されるようなコプロセッサ６０７５内でのデータレジスタバンク（ｄａｔａ ｒｅｇｉｓｔｅｒ ｂａｎｋ）６２０５の好ましい実施例のブロック略図であり、J_0生成器は概して▲ｌ▼ビットの一次ゼロ（ｐｒｉｍａｒｙ ｚｅｒｏ）強制関数をコンパイルする（ｃｏｍｐｉｌｅ）。
【０１８７】
コプロセッサ６０７５は、前出の図にあるように、ＣＰＵと共にデータバスに接続される。
レジスタバンク６２０５は、Ｂレジスタ６０７０、Ａレジスタ６１３０、Ｓレジスタ６１８０、Ｎレジスタ６２００からなる。
各々のレジスタの出力は、直列データスイッチ及び直列プロセス調節器（ｓｅｒｉａｌ ｄａｔａ ｓｗｉｔｃｈ ａｎｄ ｓｅｒｉａｌ ｐｒｏｃｅｓｓ ｃｏｎｄｉｔｉｏｎｅｒ）６０２０に接続し、次いで演算ユニット６２０６に接続され、そこでモジュラー演算を行う。
Ｎレジスタ６２００と演算ユニット６２０６との間には、J_0生成器５５２が接続される。
【０１８８】
本実施例において、J_0生成器は、上記したモジュラー演算機能において利用するために、▲ｌ▼ビットの一次ゼロ強制関数をコンパイルする。
【０１８９】

図６は、図５の演算ユニットの内部ブロック略図である。
このユニットは、更なるY_0 B_0シリアルバッファ（ｓｅｒｉａｌ ｂｕｆｆｅｒ）が第一段階でY_0を受け入れ、第二段階で次の平方演算のためにモジュラー縮小されたB_0を受け入れ、そこでＢがＮより長いことが明らかになるという点において、好ましくは高速化平方演算（ａｃｃｅｌｅｒａｔｅｄ ｓｑｕａｒｉｎｇ ｏｐｅｒａｔｉｏｎ）を支援する。

【０１９０】
次に図７Ａを参照するが、これは図６の演算ユニットの主計算部のブロック図である。
丸数字で表した数字は図７Ｂ，Ｄのシーケンス図に関するものである。

【０１９１】
次に図７Ｂを参照するが、これは平方演算の第一の反復（ｆｉｒｓｔ ｉｔｅｒａｔｉｏｎ）へ向かい、第一の反復をも含むプロセスを経時的に示した事象タイミングをポインタで示す図である。

【０１９２】
次に図７Ｃを参照するが、これは平方シーケンス（ｓｑｕａｒｉｎｇ ｓｅｑｕｅｎｃｅ）の第一の反復における次のモンゴメリ平方（Ｎｅｘｔ Ｍｏｎｔｇｏｍｅｒｙ Ｓｑｕａｒｉｎｇ）を排除する事象シーケンスを示す図である。
丸数字は図７Ａ，Ｂ，Ｄを参照する。

【０１９３】
次に図７Ｄを参照するが、これは、平方演算の第一の反復の計算の出力のタイミングを示す事象タイミングをポインタで示す図である。

【０１９４】
次に図８Ａを参照するが、これは概してJ_0の選択を示す一組のルックアップテーブルであり、N_0の右側の文字のモジュラス$2^{▲l▼}$での負の逆数（ｍｕｌｔｉｐｌｉｃａｔｉｖｅ ｉｎｖｅｒｓｅ；乗法逆元）である。
N_0は常に、ＧＦ（2^q）についてモニックである（ｍｏｎｉｃ）か、又はＧＦ（ｐ）について奇数であるので、J_0は常に存在する。

【０１９５】
図８Ａ及び８Ｂにおいて、我々はこのモジュラスの右側の文字をN_0と呼ぶ。
我々は、$N_{0\,j}$を、局所定義された（ｌｏｃａｌｌｙ ｄｅｆｉｎｅｄ）N_0文字のｊ番目のビットとする。

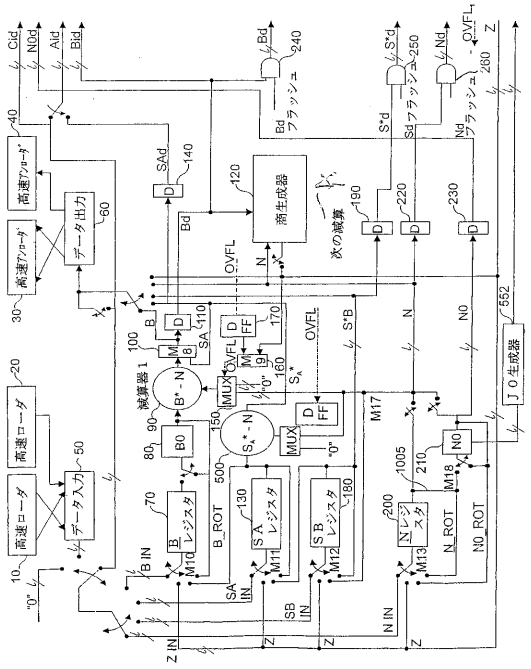
【０１９６】
図８Ｂは、４ビット若しくは２ビットのY_0ゼロ強制関数文字のいずれかの設計回路図である。
強制関数への変数入力は、N_0ビット（乗算の間一定である）、▲ｌ▼、S_0ビット、▲ｌ▼乗数及び被乗数ビットの積の▲ｌ▼の右側のビット$A_{i\,0}$及び$B_{0\,j}$、ＧＦ（2^q）又はＧＦ（ｐ）においてどの関数が機能するかを決定する桁上げスイッチ▲Ｓ▼である。
Ａ及びＢビットは、▲○×▼乗算器及びS_0に加算された▲○＋▼に入力される。　▲Ｓ▼＝０の場合、全ての桁上げができなくなる。

【０１９７】
本発明の内容を明らかにするために別の実施例の文脈において記載される本発明の多様な特徴が、単一の実施例との組合せにより提供され得ることは理解される。
逆に、説明の容易化のために単一の実施例の文脈において記載される本発明の多様な特徴が、別々に、若しくは適切なサブコンビネーションにおいても提供され得る。

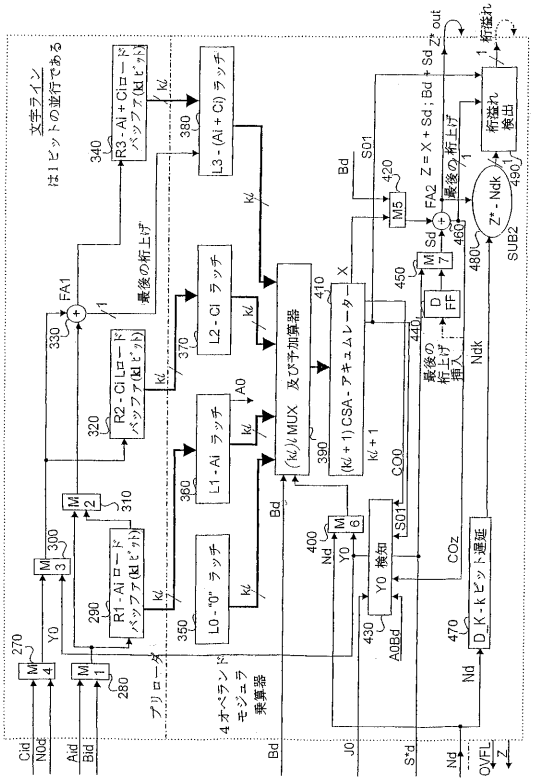
【０１９８】
本発明が、本明細書に特に示され記載されたものに限定されないことは当業者ならば理解できるだろう。
むしろ、本発明の範囲は、本明細書に記載された多様な特徴の変更及び修正と同様に、その多様な特徴の組合せ及びサブコンビネーションをも含み、従来技術にない前述の記載を読めば、当業者ならば思いつくものである。

【０１９９】
特許請求の範囲において、符号や記号は上記で与えられた意味を有する。
【図面の簡単な説明】
【図１】
本発明の一実施例に係る装置のブロック図であり、四つの主要なレジスタが記載され、演算ユニットへのシリアルデータのフロー経路、及び図３のホストＣＰＵへの入出力データの経路が示される。
【図２】
図１からのデータを演算するように機能する演算ユニットの実施例のブロック図である。
【図３】
典型的にはスマートカード内にある完全なシングルチップのモノリシック暗号コンピュータ（ｍｏｎｏｌｉｔｈｉｃ ｃｒｙｐｔｏｃｏｍｐｕｔｅｒ））の好ましい実施例のブロック略図である。
【図４】
データ使用禁止スイッチ（ｄａｔａ ｄｉｓａｂｌｅ ｓｗｉｔｃｈ）が演算ユニットからのデータのアンロードを速めるように作動する完全なシングルチップのモノリシック暗号コンピュータの好ましい実施例のブロック略図である。
【図５】
J_0を生成するように作動するデータレジスタバンク（ｄａｔａ ｒｅｇｉｓｔｅｒ ｂａｎｋ）のブロック略図である。
【図６】
Y_0検知（Y_0 ｓｅｎｓｅ；Y_0センス）が第一段階の出力を０にする演算装置である演算ユニットのブロック略図である。
【図７Ａ】
図７Ｂ～Ｄのタイミング図及びフロー図に係る丸数字を付したシーケンスアイコンを有する図６の主計算部のブロック図である。
【図７Ｂ】
平方演算の第一の反復（ｆｉｒｓｔ ｉｔｅｒａｔｉｏｎ）へ向かい、第一の反復をも含むプロセスを経時的に示した事象タイミングをポインタで示す図である。
【図７Ｃ】
図７Ａ，Ｂ，Ｄに係る平方シーケンス（ｓｑｕａｒｉｎｇ ｓｅｑｕｅｎｃｅ）のアイコン化されたポインタの第一の反復における「次のモンゴメリ平方（Ｎｅｘｔ Ｍｏｎｔｇｏｍｅｒｙ Ｓｑｕａｒｉｎｇ）」を排除する詳細な事象シーケンスを示す図である。
【図７Ｄ】
図７Ａ～Ｃに係る計算の出力のタイミングを示す図である。
【図８Ａ】
ＧＦ（2^q）及びＧＦ（ｐ）におけるY_0ベクトルの生成について記載した図であって、▲ｌ▼＝２及び▲ｌ▼＝４の場合で、右側のN_0の負の逆数を決定する一組のルックアップテーブル（ｌｏｏｋ ｕｐ ｔａｂｌｅ；参照表）である。
【図８Ｂ】
ＧＦ（2^q）及びＧＦ（ｐ）におけるY_0ベクトルの生成について記載した図であって、▲ｌ▼＝２及び▲ｌ▼＝４の場合の両数値フィールドにおけるY_0関数を生成する信号を記載するブロック略図である。

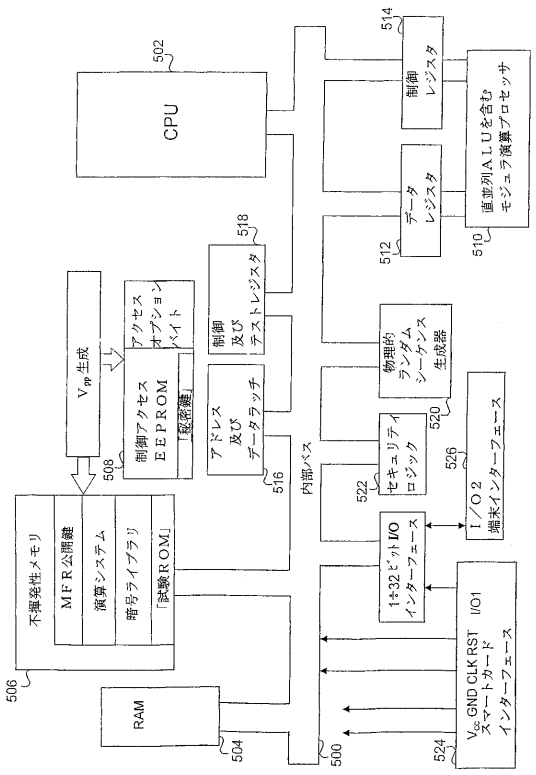
【図１】



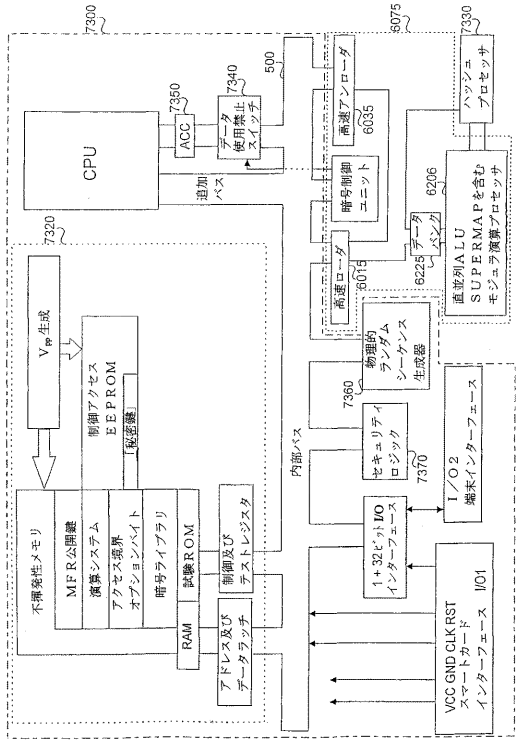
【図２】



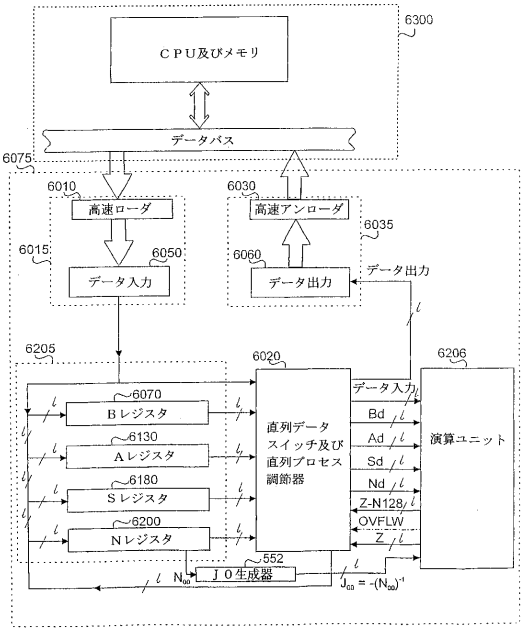
【図３】



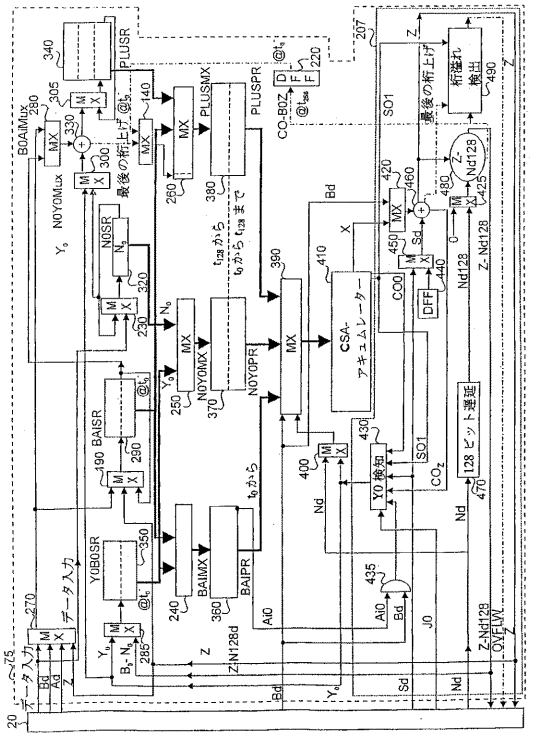
【図４】



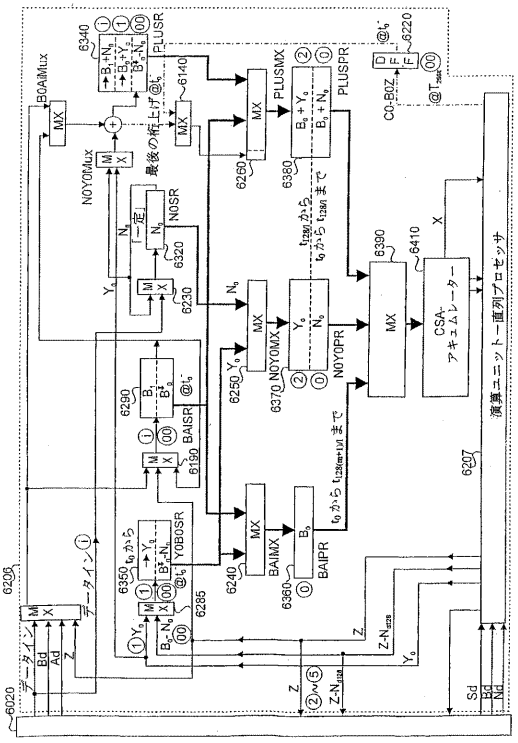
【図５】


CPU及びメモリ
データバス
高速ローダ
高速アンローダ
データ入力
データ出力
Bレジスタ
Aレジスタ
Sレジスタ
Nレジスタ
直列データスイッチ及び直列プロセス調節器
演算ユニット
J0生成器

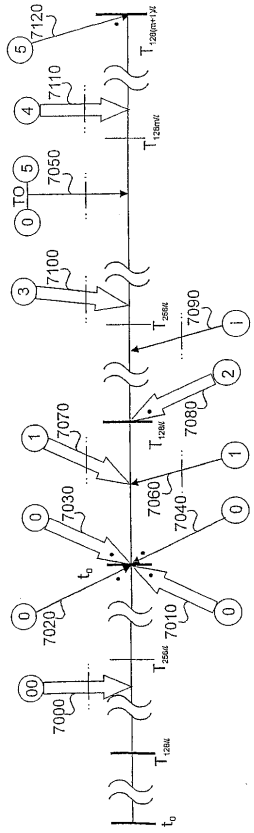
【図６】



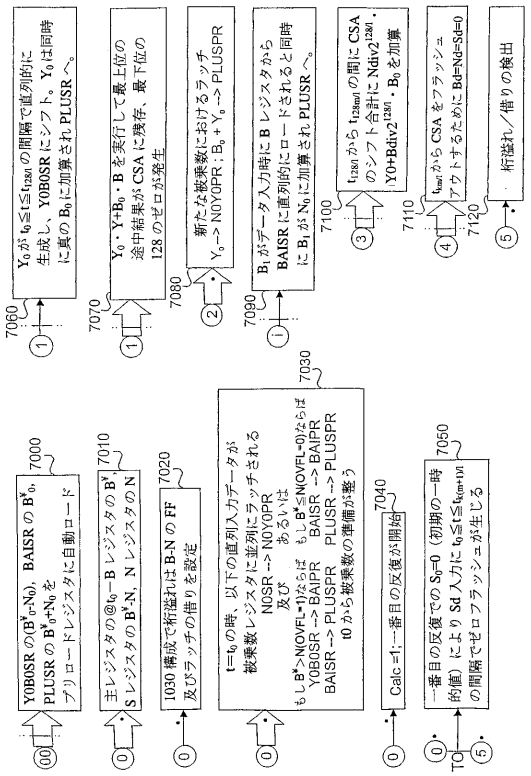
【図７Ａ】



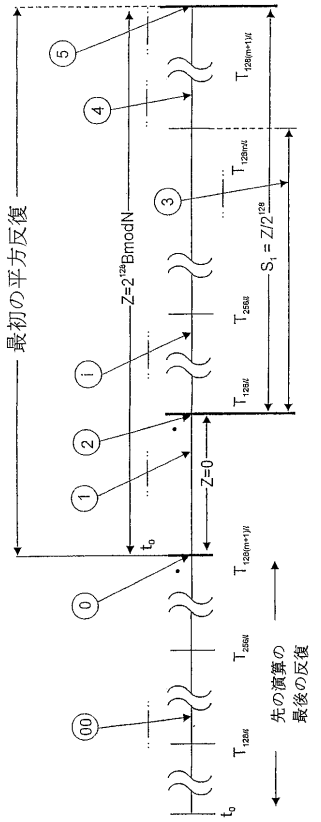
【図７Ｂ】



【図７Ｃ】



【図７Ｄ】



【図８Ａ】

	N_0	N_0^{-1}	$-(N_0^{-1})$
1	0001	0001	1111
3	0011	1011	0101
5	0101	1101	0011
7	0111	0111	1001
9	1001	1001	0111
11	1011	0011	1101
13	1101	0101	1011
15	1111	1111	0001

$\mathcal{G}$ = 1
GF(p)
l = 4

N_0	N_0^{-1}	$-N_0^{-1}$
01	01	11
11	11	01

$\mathcal{G}$ = 1
GF(p)
l = 2

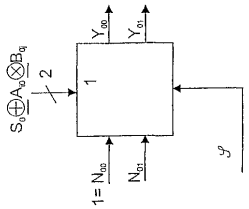
	N_0	$-N_0^{-1} = N_0^{-1}$
1	0001	0001
3	0011	1111
5	0101	0101
7	0111	1011
9	1001	1001
11	1011	0111
13	1101	1101
15	1111	0011

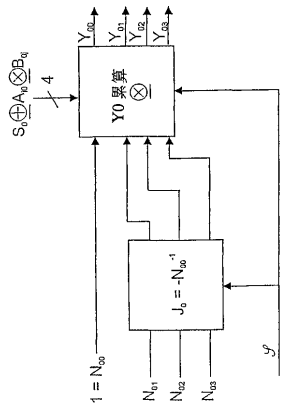
$\mathcal{G}$ = 0
桁上げなし
GF(2^n)
l = 4

N_0	N_0^{-1}
01	01
11	11

$\mathcal{G}$ = 0
桁上げなし
GF(2^n)
l = 2

【図８Ｂ】





【国際公開パンフレット】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 November 2001 (22.11.2001)

PCT

(10) International Publication Number
WO 01/89129 A2

(51) International Patent Classification[7]: H04L

(21) International Application Number: PCT/IL01/00425

(22) International Filing Date: 14 May 2001 (14.05.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
136151 15 May 2000 (15.05.2000) IL
139674 14 November 2000 (14.11.2000) IL

(71) Applicant *(for all designated States except US)*: **M-SYSTEMS FLASH DISK PIONEERS LTD.** [IL/IL]; 3B Omer Industrial Park, 84965 Omer (IL).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **DROR, Itai** [IL/IL]; 13 Hartzit Street, 84965 Omer (IL). **GRESSEL, Carmi,** **David** [IL/IL]; Kibbutz Urim, 85530 Mobile Post Negev (IL). **MOSTOVOY, Michael** [IL/IL]; 3 Michael Hazani Street, 84480 Beer Sheva (IL). **MOLCHANOV, Alexey** [IL/IL]; 12 Jabotinsky Street, 84000 Beer Sheva (IL).

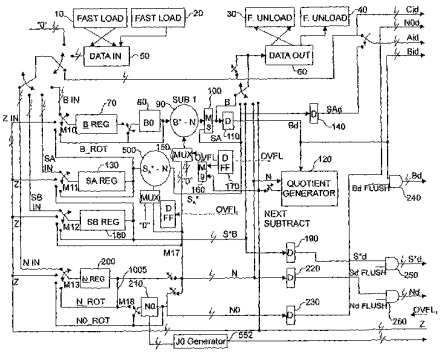
(74) Agents: **COLB, Sanford, T Sanford T. Colb & CO.** et al.; P.O. Box 2273, 76122 Rehovot (IL).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: EXTENDING THE RANGE OF COMPUTATIONAL FIELDS OF INTEGERS



(57) Abstract: An extension of the serial/parallel Montgomery modular multiplication method with simultaneous reduction as previously implemented by the applicants, adapted innovatively to perform both in the prime number and in the GF(2ⁿ) polynomial based number field, in such a way as to simplify the flow of operands, by performing a multiple anticipatory function to enhance the previous modular multiplication procedures.

WO 01/89129 A2

WO 01/89129 A2

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

EXTENDING THE RANGE OF COMPUTATIONAL FIELDS OF INTEGERS

FIELD OF THE INVENTION

The present invention relates to apparatus operative to accelerate cryptographic co-processing peripherals and additionally but not exclusively to an accelerated processing apparatus for polynomial based and prime number field arithmetic, extending the range of computational fields of integers and width of serial input operands in modular arithmetic public key cryptographic coprocessors designed for elliptic curve and RSA type computations.

BACKGROUND OF THE INVENTION

Security enhancements and performance accelerations for computational devices are described in Applicant's U.S. Patents 5,742,530, hereinafter "P1", 5,513,133, 5,448,639, 5,261,001; and 5,206,824 and published PCT patent application PCT/IL98/00148 (WO98/50851); and corresponding U.S. Patent application 09/050958, hereinafter "P2", Onyszchuk et al's U.S. Patent 4,745,568; Omura et al's U.S. Patent 4,5877,627, and applicant's U.S. Patent Application 09/480,102; the disclosures of which are hereby incorporated by reference. Applicant's U.S. Patent 5,206,824 shows an early apparatus operative to implement polynomial based multiplication and squaring, which cannot perform operations in the prime number field, and is not designed for interleaving in polynomial based computations. An additional analysis is made of an approach to use the extension field in polynomial based arithmetic in Paar, C., F. Fleischmann and P. Soria-Rodriguez, "Fast Arithmetic for Public-Key Algorithms in Galois Fields with Composite Exponents", IEEE Transactions on Computers, vol. 48, No. 10, October 1999, henceforth "Paar". W. Wesley Peterson and E.J. Weldon Jr., in the second edition of *"Error-Correcting Codes"*, published by the MIT Press, Cambridge, Mass., 1972, pages 174-179, demonstrated circuits for performing division in the polynomial based residual number system $GF(2^q)$, hereinafter, "Peterson". Peterson's circuit can only be used in a device where the multiplier is exactly the length of the modulus. Typically, that would demand a device that would be more than twice as long as present devices, and would not be cost effective for compact

implementations. It could not be used in interleaved implementations, and could not be useful where ℓ is longer than 1, as he has not provided an anticipatory device for determining the Y_0 of a multibit character.

Whereas, Knuth [D. Knuth, *The art of computer programming*, vol. 2: *Seminumerical algorithms*, Addison-Wesley, Reading Mass., 1981] page 407, implies that using an ordinary division process on a single ℓ bit character in polynomial based division, we can assume a method to anticipate the next character in the quotient, this invention discloses a method for anticipating the next character of a quotient deterministically using a logic configuration.

SUMMARY OF THE INVENTION

It is an aim of the present invention to provide a microelectronic specialized arithmetic unit operative to perform large number computations in the polynomial based and prime integer based number fields, using the same anticipating methods for simultaneously performing interleaved modular multiplication and reduction on varied radix multipliers.

A further aim of the invention also relates to a compact microelectronic specialized arithmetic logic unit, for performing modular and normal (natural, non-negative field of integers) multiplication, division, addition, subtraction and exponentiation over very large integers. When referring to modular multiplication and squaring using both Montgomery methods and a reversed format method for simplified polynomial based multiplication and squaring, reference is made to the specific parts of the device as a superscalar modular arithmetic coprocessor, or SMAP, or MAP or SuperMAP™, also as relates to enhancements existing in the applicant's U.S. Patent pending 09/050,958 filed March 31, 1998 and a continuation in part, 09/480,102 filed on January 10, 2000.

Preferred embodiments of the invention described herein provide a modular computational operator for public key cryptographic applications on portable Smart Cards, typically identical in shape and size to the popular magnetic stripe credit and bank cards. Similar Smart Cards as per applicant's technology of US Patent 5,513,133 and 5,742,530, and applicants above mentioned pending applications are being used in the new generation of public key cryptographic devices for controlling access to

computers, databases, and critical installations; to regulate and secure data flow in commercial, military and domestic transactions; to decrypt scrambled pay television programs, etc. and as terminal devices for similar applications. Typically, these devices are also incorporated in computer and fax terminals, door locks, vending machines, etc.

The preferred architecture is of an apparatus operative to be integrated to a multiplicity of microcontroller and digital signal processing devices, and also to reduced instruction set computational designs while the apparatus operates in parallel with the host's processing unit.

This embodiment preferably uses only one multiplying device which inherently serves the function of two or three multiplying devices, basically similar to the architecture described in applicant's 5,513,133 and further enhanced in U.S. Patent application 09/050,958 and PCT application PCT/IL98/0048. Using present conventional microelectronic technologies, the apparatus of the present invention may be integrated with a controlling unit with memories onto a smart card microelectronic circuit.

The main difference between hardware implementations in the polynomial based field, and in the prime number field, is that polynomial based additions and subtractions are simple XOR logic operations, without carry signals propagating from LS to MS. Consequently, there is no interaction between adjacent cells in the hardware implementation, and subtraction and addition are identical procedures. The earliest public notice that the authors are aware of was a short lecture by Marco Bucci of the Fondazione Ugo Bordoni, at the Eurocrypt Conference Rump Session in Perugia, Italy, in 1994, though, even then, this configuration was well known to engineers practiced in the art.

Previous applicant's apparati, described in P1 and P2, were typically prepared to efficiently compute elliptic curve cryptographic protocols in the GF(p) field. For use in the GF(2^q) field, in this invention, we show that as there is no interaction between adjacent binary bits in the polynomial field, computations can be processed efficiently, simultaneously performing reduction and multiplication on a superscalar multiplication device without introducing *Montgomery functions* and Montgomery parasites. Multiplication in GF(2^q) is performed as the computation machine preferably starts from the most significant partial products. Reduction is performed by adding as many

moduli as are necessary to reset MS ones to zeroes. As there is no carry out, in these additions, results are automatically modularly reduced. In this invention polynomial computations are performed using the same architecture, wherein, in $GF(2^p)$ the operands are fed in MS characters first, wherein all internal carry signals are forced to zero. $GF(p)$ computations are preferably executed as in P1 and P2, wherein LS characters are processed first and MS characters last.

The architecture has been extended to allow for a potentially faster progression, in that serial multipliers are l bit wide characters, and at each clock, an l bit character is emitted from the carry save accumulator, the CSA. This has somewhat complicated the anticipation process (Y_0), in that for single bit wide busses, an inversion of an odd number over a mod 2^x base is also an odd number, and the least significant bit of the multiplicand (J_0) was always a one. For both number fields, however the reduction process is identical, assuming the switch out of (suppression of) carries, if we remember that our only aim is to output a k character zero string, and we regard the Y_0 function only as a zero forcing vector.

The present invention also seeks to provide an architecture for a digital device, which is a peripheral to a conventional digital processor, with computational, logical and architectural novel features relative to the processes described in US Patent 5,513,133.

A concurrent process and a hardware architecture are provided, to perform modular exponentiation without division preferably with the same number of operations as are typically performed with a classic multiplication/division device, wherein a classic device typically performs both a large scale multiplication and a division on each operation. A particular feature of a preferred embodiment of the present invention is the concurrency of larger scale anticipatory zero forcing functions, the extension of number fields, and the ability to integrate this type unit for safe communications.

The advantages realized by a preferred embodiment of this invention result from a synchronized sequence of serial processes. These processes are merged to simultaneously (in parallel) achieve three multiplication operations on n character operands, using one multiplexed k character serial/parallel multiplier in n effective clock cycles, where the left hand final k characters of the result reside in the output buffer of the multiplication device. This procedure accomplishes the equivalent of three

multiplication computations in both fields, as described by Montgomery, for the prime number field and the equivalent of two multiplications and a division process in GF(2^q).

By synchronizing loading of operands into the SuperMAP and on the fly detecting values of operands, and on the fly preloading and simultaneous addition of next to be used operands, the apparatus is operative to execute computations in a deterministic fashion. All multiplication and exponentiation circuitry is preferably added which on the fly preloads, three first k character variables for a next iteration squaring sequence. A detection device is preferably provided where only two of the three operands are chosen as next iteration multiplicands, eliminating k effective clock cycle wait states. Conditional branches are replaced with local detection and compensation devices, thereby providing a basis for a simple control mechanism. The basic operations herein described are typically executed in deterministic time in GF(p) using a device described in US Patent 5,513,133 to Gressel et al or devices as by STMicroelectronics in Rousset, France, under the trade name ST19-CF58.

The apparatus of the present invention has particularly lean demands on external volatile memory for most operations, as operands are loaded into and stored in the device for the total length of the operation. The apparatus preferably exploits the CPU onto which it is appended, to execute simple loads and unloads, and sequencing of commands to the apparatus, whilst the MAP performs its large number computations. Large numbers presently being implemented on smart card applications range from 128 bit to 2048 bit natural applications. The exponentiation processing time is virtually independent of the CPU, which controls it. In practice, architectural changes are typically unnecessary when appending the apparatus to any CPU. The hardware device is self-contained, and is preferably appended to any CPU bus.

In general, the present invention also relates to arithmetic processing of large integers. These large numbers are typically in the natural field of (non-negative) integers or in the Galois field of prime numbers, GF(p), composite prime moduli, and polynomial based numbers in GF(2^q). More specifically, preferred embodiments of the present invention seek to provide devices that can implement modular arithmetic and exponentiation of large numbers. Such devices are suitable for performing the operations of Public Key Cryptographic authentication and encryption protocols, which, in the prime number field work over increasingly large operands and which cannot be

executed efficiently with present generation modular arithmetic coprocessors, and cannot be executed securely in software implementations. Preferably, the same general architecture is used in elliptic curve implementations, on integers, which are orders of magnitude smaller. Using the novel reverse mode method of multiplication, polynomial arithmetic is advantageous as generating zeroes does not encumber computations with the parasitic 2^{-n} factor.

The architecture offers a modular implementation of large operand integer arithmetic, while allowing for normal and smaller operand arithmetic enabled by widening the serial single character bus, i.e., use of a larger radix. Typically, this is useful for accelerating computations, for reducing silicon area for implementing the SuperMAP, and for generating a device of length compatible with popular Digital Signal Processors (DSP).

For modular multiplication in the prime and composite field of odd numbers, A and B are defined as the multiplicand and the multiplier, respectively, and N is defined as the modulus in modular arithmetic. N, is typically larger than A or B. N also denotes the composite register where the value of the modulus is stored. N, is, in some instances, typically smaller than A. A, B, and N are typically n characters long, where characters are typically one to 8 bits long. k the number of l bit characters in the size of the group defined by the size (number of cells) of the multiplying device. Similarly, in polynomial based GF(2^q) computations, the modulus, N, is n bits long wherein the MS bit is a one (a monic), and the A, S and B operands are also, when properly reduced, n bits long. If a result of a GF(2^q), computation is monic it is preferably "reduced" to a value with an MS zero, by XORing said result value with the modulus. In a preferred embodiment, as the first significant bit of a GF(2^q) is formed in the reverse mode, the MAP can sense if the bit is a one, and perform the preferred reduction.

In the prime field, $\equiv$, or in some instances $=$, is used to denote congruence of modular numbers, for example $16 \equiv 2 \bmod 7$. 16 is termed "congruent" to 2 modulo 7 as 2 is the remainder when 16 is divided by 7. When $Y \bmod N \equiv X \bmod N$; both Y and X may be larger than N; however, for positive X and Y, the remainders are identical. Note also that the congruence of a negative integer Y, is $Y + u \cdot N$, where N is the modulus, and if the congruence of Y is to be less than N, u is the smallest integer which gives a positive result.

In GF(2^q) congruence is much simpler, as addition and subtraction are identical, and normal computations typically do not leave a substantial overflow. For $N = 1101$ and $A = 1001$, as the left hand MS bit of A is 1, we must reduce ("subtract") N from A by using modulo 2 arithmetic, A XOR N = 1001 XOR 1101 = 0100.

The Yen symbol, ¥, is used to denote congruence in a limited sense, especially useful in GF(p). During the processes described herein, a value is often either the desired value, or equal to the desired value plus the modulus. For example X ¥ 2 mod 7. X can be equal to 2 or 9. X is defined to have limited congruence to 2 mod 7. When the Yen symbol is used as a superscript, as in $B^{¥}$, then $0 \le B^{¥} < 2N$, or stated differently, $B^{¥}$ is either equal to the smallest positive B which is congruent to $B^{¥}$, or is equal to the smallest positive congruent B plus N, the modulus. Other symbols, specific to this invention appear later in this summary.

When $X = A \bmod N$, X is defined as the remainder of A divided by N; e.g., 3 = 45 mod 7, and much simpler in GF(2^q) – 1111 mod 1001 ≡ 0110.

In number theory, the modular multiplicative inverse of X is written as X^{-1}, which is defined by $X \cdot X^{-1} \bmod N = 1$. If $X = 3$, and $N = 13$, then $X^{-1} = 9$, i.e., the remainder of $3 \cdot 9$ divided by 13 is 1 in GF(p).

For both number fields, we typically choose to compute the multiplicative inverse of A using the exponential function, e.g., $A^{-1} \bmod q \equiv A^{q-2} \bmod q$.

The acronyms MS and LS are used to signify "most significant" and "least significant", respectively, when referencing bits, characters, and full operand values, as is conventional in digital nomenclature, but in the reversed mode polynomial base, operands are loaded MS data first and LS last, wherein the bit order of the data word is reversed when loaded.

Throughout this specification N designates both the value N, and the name of the shift register which stores N. An asterisk superscript on a value, denotes that the value, as stands, is potentially incomplete or subject to change. A is the value of the number which is to be exponentiated, and n is the bit length of the N operand. After initialization when A is "Montgomery P field normalized" to A^* ($A^* = 2^n A$ - explained in P1) A^* and N are typically constant values throughout the intermediate step in the exponentiation. In GF(2^q) computations where computations might be performed with the normal unreversed positioning of bits we would be bound by this same protocol. However,

using the reversed format, our computations generate most significant zeroes, which are disregarded, and do not represent a multiplication shift, as there is no carry out.

During the first iteration, after initialization of an exponentiation, B is equal to A^*. B is also the name of the register wherein the accumulated value that finally equals the desired result of exponentiation resides. S or S^* designates a temporary value; and S designates the register or registers in which all but the single MS bit of a number in GF(p) S is stored. (S^* concatenated with this MS bit is identical to S.) S(i-1) denotes the value of S at the outset of the i'th iteration. In these polynomial computations there is no need to perform modular reduction on S.

Typically, Montgomery multiplication of X and Y in the prime number field is actually an execution of $(X \cdot Y \cdot 2^{-n}) \bmod N$, where n is typically the number of characters in a modulus. This is written, $\mathcal{P}(A \cdot B)_N$, and denotes MM or multiplication in the P field. In the context of Montgomery mathematics, we refer to multiplication and squaring in the P field and in the polynomial based field as multiplication and squaring operations.

We will redefine this innovative extension of a Montgomery type arithmetic in the GF(2^q) to mean a reversed format data order, wherein MS zero forcing does not change congruence, or initiate a burdensome parasitic factor. We may thus introduce a new set of symbols to accommodate the arithmetic extension, and to enable an architecture with wider serial multiplier buses. Such a more than one bit serial multiplier stream is preferable for enabling a natural integer superscalar multiplier.Such multiplier device may accept 32 bit multiplicands and 4 bit multipliers, into an apparatus that can perform modular arithmetic multiplications and reductions simultaneously.

Symbols in the Serial/Parallel Super Scalar Modular Multiplier Enhancement

ℓ Number of bits in a character (digit).

r Radix of multiplier character, $r = 2^{\ell}$.

n Size of operands (multiplier, multiplicand and modulus) in characters. In the demonstration of the computations in the GF(p) field of Montgomery arithmetic, ℓ is equal to one, and n is the bit length of the modulus operand.

k Length of serial-parallel multiplier in characters.

m Number of interleaved slices (segments) of multiplicand: $m = n/k$.

S_i Partial product result of i 'th MM iteration; $0 \le i \le m - 1$; $S_0 = 0$.

S_{i0} Right hand character of the i 'th iteration result, after **disregarding** the first k character right hand zeroes of Z.

$\underline{S_i}$ The left hand $n - k$ characters of the i 'th result.

S_{ij} j 'th character of S_i.

A Parallel multiplicand consists of $m \cdot k$ characters.

A_i The i 'th k character slice of A, (and/or register storing A_i)

A_{it} The t 'th character of A_i.

B A serial multiplier (and/or register store of B).

B_0 First right hand k characters of B.

$\underline{B}$ Last left hand $(n - k)$ characters of B.

B_{0j} j 'th character of B_0.

$\underline{B_j}$ j 'th character of $\underline{B}$.

N Modulus operand. (and/or register storing said multiplier).

N_0 The Right Hand k characters of N.
{the LS characters in GF(p); MS characters in GF(2^q) }

$\underline{N}$ ($n - k$) Left Hand characters of N.
{ MS characters in GF(p); LS characters in GF(2^q) }

N_{0j} j 'th character of N_0.

$\underline{N_j}$ j 'th character of $\underline{N}$.

Y_0 Zero forcing variable required for both Montgomery multiplication and reduction in GF(p). Y_0 is k characters long.

Y_{0j} j 'th character of Y_0.

R A summation of the value residing in the Carry Save Accumulator, (includes unresolved internal carry ins) and the carry out bit from the final serial summator, 460.

J_{00} Zero forcing character function of the modulus, N, for "on the fly" finite field multiplication and reduction. For $i = 1$, J_{00} is always equal to 1.

$Carry_j$ j 'th internal carry character of radix r serial-parallel multiplier.

$Carry_\alpha$ radix r carry of output serial adder for GF(p) computations.

Sum_j j 'th internal sum character of radix r serial-parallel multiplier.

LS Least Significant.

MS Most Significant.

|| Concatenation, e.g. $A = 110$, $B = 1101$; $A \| B = 1101101$.

Right Hand A Least Significant portion of all GF(p) computation data blocks and an MS portion of the reversed GF(2^q) format.

Left Hand A Most Significant portion of all GF(p) computation data blocks and an LS portion of the reversed GF(2^q) format.

GF(p) Galois Field, strictly speaking finite fields over prime numbers where we also use composite integers (product of two very large prime numbers) that allow for addition, subtraction, multiplication and pseudo-division.

GF(2^q) Galois Fields finite fields using modulo 2 arithmetic.

$\underline{\oplus}$ An operator or device which may be externally switched to add or subtract integers with or without carries, as befits the specific number system.

$\underline{\otimes}$ An operator or device which may be switched to either execute multiplication over GF(p) or multiplication over GF(2^q).

$\mathcal{S}$ The number field switch, where if:

$\mathcal{S} = 1$, the switch is operative to enable all carry in/outputs for GF(p) computations;

$\mathcal{S} = 0$, the switch is operative to disable all carry in/outputs for GF(2^q) computations.

SuperMAP Any one of a member of the proprietary Superscalar Modular Arithmetic Processor family that is the subject of this invention. The SuperMAP trademark is registered in Europe and is pending in the United States.

According to a first aspect of the present invention, there is therefore provided a microelectronic apparatus for performing $\underline{\otimes}$ multiplication and squaring in both polynomial based GF(2^q) and GF(p) field arithmetic, squaring and reduction using a serial fed radix 2^d multiplier, B, with k character multiplicand segments, A_i, and a k character $\underline{\oplus}$ accumulator wherein reduction to a limited congruence is performed "on the fly", in a systolic manner, with A_i, a multiplicand, times B, a multiplier, over a modulus, N, and a result being at most $2n + 1$ characters long, including the k first emitting disregarded zero characters, which are not saved, where k characters have no less bits than the modulus, wherein said operations are carried out in two phases, the apparatus comprising;

a first (B), and second (N) main memory register means, each register operative to hold at least n bit long operands, respectively operative to store a multiplier value designated B, and a modulus, denoted N, wherein the modulus is smaller than 2^n;

a digital logic sensing detector, Y_0, operative to anticipate "on the fly" when a modulus value is to be $\underline{\oplus}$ added to the value in the $\underline{\oplus}$ adder accumulator device such that all first k characters emitting from the device are forced to zero;

a modular multiplying device for at least k character input multiplicands, with only one, at least k characters long $\underline{\oplus}$ adder, $\underline{\oplus}$ summation device operative to accept k character multiplicands, the $\underline{\otimes}$ multiplication device operative to switch into the $\underline{\oplus}$ accumulator device, in turn, multiplicand values, and in turn to receive multiplier values from a B register, and an "on the fly" simultaneously generated anticipated value as a multiplier which is operative to force k first emitting zero output characters in the first phase, wherein at each effective machine cycle at least one designated multiplicand is $\underline{\oplus}$ added into the $\underline{\oplus}$ accumulation device;

the multiplicand values to be switched in turn into the $\underline{\oplus}$ accumulation device consisting of one or two of the following three multiplicands, the first multiplicand being an all-zero string value, a second value, being the multiplicand A_i, and a third value, the N_0 segment of the modulus;

an anticipator to anticipate the ℓ bit k character serial input Y_0 multiplier values;

the apparatus being operable to input in turn multiplier values into the multiplying device said values being first the B operand in the first phase, and concurrently, the second multiplier value consisting of the Y_0, "on the fly" anticipated k character string, to force first emitting zeroes in the output;

the apparatus further comprising an accumulation device, $\underline{\oplus}$, operative to output values simultaneously as multiplicands are $\underline{\oplus}$ into the $\underline{\oplus}$ accumulation device;

an output transfer mechanism, in the second phase operative to output a final modular $\underline{\otimes}$ multiplication result from the $\underline{\oplus}$ accumulation device.

According to a preferred embodiment $\underline{\oplus}$ summations into the $\underline{\oplus}$ accumulation device are activated by each new serially loaded higher order multiplier characters.

Preferably, the multiplier characters are operative to cause no $\oplus$ summation into the $\oplus$ accumulation device if both the input B character and the corresponding input Y_0 character are zeroes;

are operative to $\oplus$ add in only the A_i multiplicand if the input B character is a one and the corresponding Y_0 character is a zero;

are operative to $\oplus$ add in only the N, modulus, if the B character is a zero, and the corresponding Y_0 character is a one; and

are operative to $\oplus$ add in the $\oplus$ summation of the modulus, N, with the multiplicand A_i if both the B input character and the corresponding Y_0 character are ones.

Preferably, the apparatus is operative to preload multiplicand values A_i and N, into two designated preload buffers, and to $\oplus$ summate these values into a third multiplicand preload buffer, obviating the necessity of $\oplus$ adding in each multiplicand value separately.

Preferably, the multiplier character values are arranged for input in serial single character form, and wherein the Y_0 detect device is operative to anticipate only one character in a clocked turn.

In a preferred embodiment wherein the $\oplus$ accumulation device performs modulo 2 computations, XOR addition/subtraction, wherein all carry bits in addition and subtraction components are disregarded, thereby precluding provisions for overflow and further limiting convergence in computations.

Preferably, carry inputs are disabled to zero, denoted, $\mathcal{S}=0$, typically operative to perform polynomial based multiplication.

Preferably the apparatus is operative to provide non-carry arithmetic by omitting carry circuitry, such that an $\mathcal{S}$ equal to zero acting on an element in a circuit equation computing in GF(2^q), $\mathcal{S}$ designates omitted circuitry and all adders and subtractors, designated $\oplus$ to XOR, modulo 2 addition/subtraction elements.

A preferred embodiment is adapted such that the first k character segments emitted from the operational unit are zeroes, being controlled by the following four quantities in anticipating the next in turn Y_0 character:

i. the ℓbit S_{out} bits of the result of the ℓbit by ℓbit mod 2^{ℓ} $\underline{\otimes}$ multiplication of the right-hand character of the A_i register times the B_d character of the B Stream, $A_0 \cdot \underline{\otimes} B_d \bmod 2^{\ell}$;

ii. the first emitting carry out character from the $\underline{\oplus}$ accumulation device, $\mathcal{S}(CO_0)$;

iii. the ℓbit S_{out} character from the second from the right character emitting cell of the $\underline{\oplus}$ accumulation device, SO_1;

iv. the ℓbit J_0 value, which is the negative multiplicative inverse of the right-hand character in the N_0 modulus multiplicand register,

wherein values, $A_0 \underline{\otimes} B_d \bmod 2^{\ell}$, $\mathcal{S}(CO_0)$, and SO_1 are $\underline{\oplus}$ added character to character together and "on the fly" multiplied by the J_0 character to output a valid Y_0 zero-forcing anticipatory character to force an ℓbit egressing string of zeroes.

The apparatus is preferably operable to perform multiplication on polynomial based operands performed in a reverse mode, multiplying from right hand MS characters to left hand LS characters, operative to perform modular reduced $\underline{\otimes}$ multiplication without Montgomery type parasitic functions.

Preferably, the apparatus further comprises preload buffers which are serially fed and where multiplicand values are preloaded into the preload buffers on the fly from one or more memory devices.

The apparatus is preferably operative to $\underline{\oplus}$ summate into a multiplication stream a previous value, emitting from an additional n bit S register, via an ℓ bit $\underline{\oplus}$ adder circuit such that first emitting output characters are zeroes when the Y_0 detector is operative to detect the necessity of $\underline{\oplus}$ adding moduli to the $\underline{\oplus}$ summation in the $\underline{\oplus}$ accumulation device, wherein the Y_0 detector operates to detect, utilizing the next in turn $\underline{\oplus}$ added characters $A_0 \underline{\otimes} B_d \bmod 2^{\ell}$, $\mathcal{S}(CO_0)$, SO_1, S_0 and $\mathcal{S}(CO_z)$, the composite of $\underline{\oplus}$ added characters to be finite field $\underline{\otimes}$ multiplied on the fly by the ℓbit J_0 value, where $\underline{\oplus}$ defines the addition and $\underline{\otimes}$ defines the multiplication as befits the finite field used in the process.

Preferably, for $\ell = 1$, J_0 is implicitly 1, and the $J_0 \otimes$ multiplication is implicit, without additional hardware.

Preferably, a comparator is operative to sense a finite field output from the $\otimes$ modular multiplication device, whilst working in GF(p), where the first right hand emitting k zero characters are disregarded, where the output is larger than the modulus, N, thereby operative to control a modular reduction whence said value is output from the memory register to which the output stream from the multiplier device is destined, and thereby precluding allotting a second memory storage device for the smaller product values.

Preferably, for $\otimes$ modular multiplication in the GF(2^q), the apparatus is operative to multiply without an externally precomputed more than ℓ bit zero-forcing factor.

A preferred embodiment is operative to compute a J_0 constant by resetting either the A operand value or the B operand value to zero and setting the partial result value, S_0, to 1.

According to a second aspect of the present invention there is provided a microelectronic apparatus for performing interleaved finite field modular multiplication of integers A and B, so as to generate an output stream of A times B modulus N, wherein a number of characters in a modulus operand register, n, is larger than a segment length of k characters, wherein the $\otimes$ multiplication process is performed in a plurality of interleaved iterations, wherein at each interleaved iteration with operands input into a $\otimes$ multiplying device, consisting of N, the modulus, B, a multiplier, a previously computed partial result, S, and a k character string segment of A, a multiplicand, the segments progressing from the A_0 string segment to the A_{m-1} string segment, wherein each iterative result is $\oplus$ summated into a next in turn S, temporary result, in turn, wherein first emitting characters of iterative results are zeroes, the apparatus comprising:

first (B), second (S) and third (N) main memory registers, each register respectively operative to store a multiplier value, a partial result value and a modulus;

a modular multiplying device operative to $\oplus$ summate into the $\oplus$ accumulation device, in turn one or two of a plurality of multiplicand values, during each one of a plurality of phases of the iterative $\otimes$ multiplication process, and in turn to receive as multipliers, in turn, inputs from:

said B register,

an "on the fly" anticipating value, Y_0, being usable as a multiplier to force first emitting right-hand zero output characters in each iteration, and

said N, register;

the multiplicand parallel registers operative at least to receive in turn, values from the A, B, and N register sources, and in turn, also a multiplicand zero forcing Y_0, value;

the apparatus further utilizing the Y_0 detect device operative to generate a binary string operative to be a multiplier during the first phase and operative to be a multiplicand in the second phase;

the apparatus being operable to obtain multiplicand values suitable for switching into the $\underline{\oplus}$ accumulation device for the first phase consisting of a first zero value, a second value, A_i, which is a k character string segment of a multiplicand, A, and a third value N_0, being the first emitting k characters of the modulus, N;

the apparatus further being operable to utilize a temporary result value, S, resulting from a previous iteration, operative to be $\underline{\oplus}$ summated with the value emanating from the $\underline{\oplus}$ accumulation device, to generate a partial result for the next-in-turn iteration;

the apparatus further being operable to utilize multiplicand values to be input, in turn, into the $\underline{\oplus}$ accumulation device for a second multiplication phase being, comprising firstly a first zero value, a second A_i operand, remaining in place from the first phase, and thirdly a Y_0 value having been anticipated in the first phase;

multiplier values input into the multiplying device in the first phase being firstly an emitting string, B_0, said multiplication device being operable to multiply said string segment concurrently $\underline{\otimes}$ multiplying with a second $\underline{\otimes}$ multiplier value consisting of the anticipated Y_0 string which is simultaneously loaded character by character as it is generated into a preload multiplicand buffer for the second phase;

the two multiplier values operative to be input into the apparatus during a second phase being left hand n- k character values from the B operand, designated $\underline{B}$, and the left hand n - k characters of the N modulus, designated $\underline{N}$, respectively; and

wherein said apparatus further comprises a multiplying flush out device operative in a last phase to transfer the left hand segment of a result value remaining in the $\underline{\oplus}$ accumulation device into a result register.

Preferably, the apparatus is operable to perform $\underline{\otimes}$ multiplication on polynomial based operands is performed in a reverse mode, multiplying from MS characters to LS characters, operative to perform modular reduction without the Montgomery type parasitic functions, as in applicant's patent US 5,742,530.

According to a third aspect of the present invention there is provided apparatus operative to anticipate a Y_0 value using first emitting values of the multiplicand, and present inputs of the B multiplier, carry out values from the $\underline{\oplus}$ accumulation device, $\underline{\oplus}$ summation values from the $\underline{\oplus}$ accumulation device, the present values from the previously computed partial result, and carry out values from the $\underline{\oplus}$ adder which $\underline{\oplus}$ summates the result from the $\underline{\oplus}$ accumulation device with the previous partial result.

Preferably, the apparatus is adapted to insure that k first emitting values from the device are zero characters said adaptation comprising anticipation of a next in turn Y_0 character using the following quantities:

i. the ℓ bit S_{out} bits of the result of the ℓ bit by ℓ bit mod 2^{ℓ} $\underline{\otimes}$ multiplication of the right-hand character of the A_i register times the B_d character of the B Stream, $A_0 \cdot B_d \bmod 2^{\ell}$;

ii. the first emitting carry out character from the $\underline{\oplus}$ accumulation device, $\mathcal{S}(CO_0)$;

iii. the ℓ bit S_{out} character from the second from the right-hand character emitting cell of the $\underline{\oplus}$ accumulation device, SO_1;

iv. the next in turn character value from the S stream, S_d;

v. the ℓ bit carry out character from the Z output full adder, $\mathcal{S}(CO_2)$;

vi. the ℓ bit J_0 value, which is the negative multiplicative inverse of the right-hand character in the N_0 modulus multiplicand register;

wherein values, $A_0 \cdot B_d \bmod 2^{\ell}$, $\mathcal{S}(CO_0)$, SO_1, S_d are $\underline{\oplus}$ added character to character together and "on the fly" $\underline{\otimes}$ multiplied by the J_0 character to output a valid Y_0 zero-forcing anticipatory character.

In a further embodiment there is also provided at least one sensor operative to compare the output result to N, the modulus, the mechanism operative to actuate a second subtractor on the output of the result register, thereby to output a modular reduced value which is of limited congruence to the output result value precluding the necessity to allot a second memory storage for a smaller result.

In a yet further embodiment, a value which is a $\underline{\oplus}$ summation of two multiplicands is loaded into a preload character buffer with at least a k characters memory means register concurrently whilst one of the first values is loaded into another preload buffer.

According to a fourth aspect of the present invention there is provided apparatus with one $\underline{\oplus}$ accumulation device, and an anticipating zero forcing mechanism operative to perform a series of interleaved $\underline{\otimes}$ modular multiplications and squarings and being adapted to perform concurrently the equivalent of three natural integer multiplication operations, such that a result is an exponentiation.

In an embodiment, next-in-turn used multiplicands are preloaded into a preload register buffer on the fly.

In a further embodiment, apparatus buffers and registers are operative to be loaded with values from external memory sources and said buffers and registers are operative to be unloaded into the external memory source during computations, such that the maximum size of the operands is dependent on available memory means.

In a yet further embodiment there is also provided memory register means, said memory means are typically serial single character in/ serial single character out, parallel at least k characters in/parallel at least k characters out, serial single character in/parallel at least k characters out, and parallel k characters in/serial single character out.

Preferably the apparatus is operable to provide, during a final phase of a multiplication type iteration, the multiplier inputs are zero characters which are operative to flush out the left hand segment of the carry save $\underline{\oplus}$ accumulator memory.

Preferably, the apparatus is operable to preload next in turn multiplicands into preload memory buffers on the fly, prior to their being required in an iteration.

Preferably, the apparatus is operable to preload multiplicand values into preload buffers on the fly from central storage memory means.

The same device is operable to compute the k character Montgomery constant J_0, related to the right hand k character segment of a modulus preferably by resetting both A and B to zero and setting $S_0 = 1$, whilst subsequently performing a k bit multiplication. The result will reside in the Y_0 register.

Modular Multiplication Sequences using Montgomery type Arithmetic

The k character carry save adder, the CSA, is the basis for the serial/parallel superscalar modular multiplication in both the polynomial field and in the prime number field. Polynomial GF(2^e) based computations are executed preferably with all carry mechanisms switched off.

The Serial - Parallel Super Scalar Montgomery Multiplier computes Montgomery modular product in three phases, wherein in one preferred embodiment, the last phase may be a single clock dump of the whole left hand k character segment with a carry of the CSA (MS for normal multiplications, LS for reverse mode polynomial computations), and in a more compact embodiment, the last phase may be a k effective clock serial flush out of the contents of the CSA.

In previous P2 disclosures, the Y_0 factor was computed bit by bit, consequently, only the right hand bit of J_0 was consequential, and by definition a one, and a function of the right hand bit of the modulus. In this enhancement device, the device is character serial, and an ℓ bit character Y_0 is generated at each clock cycle. As in previous P1 disclosures, Y_0 is the first phase zero forcing function which adds in the value of modulus a necessary number of times into the accumulated result, so that the relevant answer is congruent and never longer than $mk+1$ characters, and the first right hand emitted characters were all zeroes. $X + QN \equiv X$, as $QN \equiv 0 \bmod N$.

Modular Multiplication Sequences

Prior to initiating computation, we assume that there are no previous temporary or random values in the device; that the operands N, B, and at least the first segment value of A are available in the registers of the device. $S_0 = 0$; the partial product at initiation is typically zero. Typically, modular arithmetic is executed on operands consisting of two or more k character segments, typically in three distinct phases. For a normal full multiplication, where there are m segments of the modulus, there are typically m

superscalar multiplication interleaved iterations whence each segment of the multiplicand is multiplied by the total multiplier, typically, B.

The process of the first phase, on the (i = 0'th segment) of each interleaved superscalar multiplication is the generic superscalar multiplication accumulation interaction:

$$S_i \underline{\oplus} A_i \cdot B_0 \underline{\oplus} Y_0 \cdot N_0$$

(B_0 and Y_0 are serially character by character fed from the first segment of the operand into the multiplier, A_i and N_0 are parallel single slice operands, S_i is the partial product from a previous iteration/computation. $S_i = 0$ on the 0'th first iteration.)

The first phase process implements a $\underline{\oplus}$ summation of the two superscalar products with the right hand segment of the previous result. A k character string of zeroes has emitted from the multiplying device, and is disregarded; a partial first segment result resides in the device buffer, which is summated into the second phase result.

The first phase result consists typically of R, the contents of the CSA concatenated with the serial outputted right hand segment of all zeroes. (In GF(p) computations there is an additional LS carry-in bit to R.)

The process of the second phase is the generic superscalar multiplication accumulation interaction:

$$R \underline{\oplus} S_i \underline{\oplus} A_i \cdot \underline{B} \underline{\oplus} Y_0 \cdot \underline{N}$$

(Remember, an underlined variable, e.g., $\underline{B}$, is the remaining left hand value of an operand. It is typically one or more segments, i.e., $m - 1$ segments. $\underline{B}$ and $\underline{N}$ are serially fed character by character into the multiplier, and both A_i, remaining from the first phase, and Y_0, which was a multiplier in the first phase and was loaded into the machine in the first phase to be a multiplicand in subsequent iterations, are parallel operands).

At the end of the second phase, typically consisting of m-1 iterations, the left hand segment of S_i remains in the CSA - ready to be transferred – and the right hand slice (k character segment) has emanated from the device, typically into an S register. Note that multiplication in the prime number field has been performed in a conventional carry save summation method. Multiplication in the GF(2^g) reversed format mode has progressed from most to least significant characters. The Y_0 function has anticipated when a modulus value must be "added" into the accumulator. Except for the disabled

carry bits in the device, the mechanical process is typically identical for the two number systems.

We disclose the method used for deriving the Y_{0j} characters of the zero-forcing vector in finite fields.

Compute: $J_{0j} \equiv -N_{00}^{-1} \bmod 2^l$.

All natural integers which are relatively prime to a modulus have multiplicative inverses in both number fields. N_{00} is odd and, therefore, has no factor of 2. All factors of mod 2^l are 2, so that any number with a least significant 1, and a modulus whose only factor is 2, are relatively prime, and J_{0j} always exists. Formally, for odd N_{00} and 2^l, $\gcd(N_{00},2^l) = 1$.

This single character of the function can be hardwire implemented with random logic, with simple circuitry, or with a simple look up table. There are only 2^{k-1} different values that must be derived in a look up table. In the reverse mode format, the polynomial modulus must be right justified, nominally odd. In typical exponentiational functions, both number fields, the right hand bit of the modulus bit is a one, nominally odd, and the multiplicative inverse of an odd number mod 2^k must always be odd.

If $l = 1$, the J_{00} multiplier is explicitly equal to 1, and need not be computed.

The result of a character output forced by the Y_0 function during the first phase is always 0, consequently the j'th character output, Zij, of the SuperMAP is:

$0 = (2^l R \oplus S_{ij} \oplus A_{i0} \cdot B_{0j} \oplus Y_{0j} \cdot N_{00}) \bmod 2^l = Z_{ij}$; therefore,

$(R \oplus S_{ij} \oplus A_{i0} \cdot B_{0j}) \equiv -Y_{0j} \cdot N_{00}$; and

$Y_{0j} \equiv -N_{00}^{-1} (R \oplus S_{ij} \oplus A_{i0} \cdot B_{0j}) \bmod r$.

From the above equation, we learn that J_{00} is preferably the negative value of the modular multiplicative inverse of the right hand k character of the modulus for both number systems; noting that in modulo 2 arithmetic, positive and negative values are the same.

R is the summation of the value remaining in the CSA summated to the carry out bit from the final serial adder, 460, in Fig. 2. S_{ij} is the j'th bit of the partial product at the i'th iteration. A_{i0} is the right hand character (LS in GF(p)) of the i'th slice of A. B_{0j} is the j'th character of B. B_0 is a constant (multiplier) during all iterations of a Montgomery multiplication. Y_0 is a k character vector generated at each (i'th) iteration.

Y_{0j} is j'th character generated at the j'th clock of the first phase of an iteration. N, is an m sliced modulus. N_0 is the right hand slice of the modulus. N_{00} is the right hand character of N_0.

Formalizing the Super scalar modular multiplication method for both fields:

$S_0 = 0$;

For $i = 0$ to m - 1 (interleave iterations)

First phase: (of each interleave)

$R = 0$

For $j = 0$ to $k - 1$ (each character of first phase)

$Y_{0j} = (J_{00} \ (R \oplus S_{0j} \oplus A_{i0} \otimes B_{0j})) \bmod 2^l$

$Z_{ij} = (R \oplus S_{ij} \oplus A_{i0} \otimes B_{0j} \oplus Y_{0j} \otimes N_0) \bmod 2^l$; and

$R = [(2^l R \oplus S_{ij} \oplus A_{i0} \otimes B_{0j} \oplus Y_{0j} \otimes N_0)] / 2^l$

After k effective clock cycles, the first segment of the Z stream was all zeroes, and was disregarded; the relevant Y_0, k character vector, is now prepared to be a multiplicand in the next phase, and the summated R value will be used in the next phase.

Second phase:

For $j = k$ to n - 1

$Z_{ij} = (R \oplus S_{ij} \oplus A_{i0} \otimes B_{0j} \oplus Y_0 \otimes N_{0j}) \bmod 2^l$;

$R = [2^l R \oplus S_{ij} \oplus A_{i0} \cdot B_{0j} \oplus Y_0 \cdot N_{0j}] / 2^l$

Implementation of the above algorithm with a character based serial-parallel multiplier is a simple extension of the above protocol:

(*Quotient*(x, y) is the integer division function without remainder. For example, if $x = 10101_b$, and $y = 10000_b$, then *Quotient*(x, y) = 1) .

$S_0 = 0$

For $i = 0$ to m - 1 (Interleaved loop)

First phase:

For $j = 0$ to k - 1

$Y_{0j} =$

$(J_{00} \cdot S_{i0} \oplus A_{i0} \otimes B_{0j} \oplus \mathcal{S} Carry_0 \oplus Sum_1 \oplus Quotient(S_{i0} \oplus Sum_0, r))) \bmod r$

For $i = 0$ to k - 1 (Whole loop with 1 clock pulse)

$Sum_t = (Sum_{t+1} \oplus \mathcal{S}^k Carry_t \oplus A_{it} \otimes B_{0j} \oplus Y_{0j} \otimes N_{0t}) \bmod r$

$Carry_t = (Quotient((Sum_{t+1} \oplus Carry_t \oplus A_{it} \otimes B_{0j} \oplus Y_{0j} \otimes N_{0t}), r)$

(Output of multiplier device in this stage is '0' s)

Second phase:

Main part

$Carry_a = 0$

For $j = k$ to $n - 1$

For $t = 0$ to $k - 1$ (Whole loop with 1 clock pulse)

$Sum_t = (Sum_{t+1} \oplus \mathcal{S}^k Carry_t \oplus A_{it} \otimes B_j \oplus Y_{0t} \otimes N_j) \bmod r$

$Carry_t = Quotient((Sum_{t+1} \oplus Carry_t \oplus A_{it} \otimes B_j \oplus Y_{0t} \otimes N_j), r)$

$S_{i,j-k} = (S_{i,j-2k} \oplus Sum_0 \oplus \mathcal{S}^k Carry_a) \bmod r$

$Carry_a = Quotient((S_{i,j-2k} \oplus Sum_0 \oplus Carry_a), r)$

Flushing of the multiplier

For $j = n$ to ($n + k - 1$)

For $t = 0$ to $k - 1$ (Whole loop with 1 clock pulse)

$Sum_t = (Sum_{t+1} \oplus \mathcal{S}^k Carry_t) \bmod r$

$Carry_t = Quotient((Sum_{t+1} \oplus Carry_t), r)$

$S_{i,j-k} = (S_{i,j-2k} \oplus Sum_0 \oplus \mathcal{S}^k Carry_a) \bmod r$

$Carry_a = Quotient((S_{i,j-2k} \oplus Sum_0 \oplus Carry_a), r)$

For a formal explanation with examples of the particular case where $l = 1$ in the GF(p) field, see P1.

The above describes a microelectronic method and apparatus for performing interleaved finite field modular multiplication of integers A and B operative to generate an output stream of A times B modulus N wherein n is the number of characters in the modulus operand register and is larger than k, wherein the $\otimes$ multiplication process is performed in iterations, wherein at each interleaved iteration with operands input into a $\otimes$ multiplying device, consisting of N, the modulus, B, a multiplier, a previously computed partial result, S, and a k character string segment of A, a multiplicand, the segments progressing from the A_0 string segment to the A_{m-1} string segment, wherein each iterative result is $\oplus$ summated into a next in turn S, temporary result, in turn,

wherein first emitting characters of iterative results are zeroes, the apparatus comprising:

Typically, there may be four serial l bit character registers feeding the multiplying device, first (B), second (S) and third (N) and preferably (A), configured to efficiently load the multiplier. For computations on long operands which typically are not accommodated in the MAP's internal registers, the CPU can load operands from its accessible memory.

Typically, these main memory registers store and output operands, respectively operative to store a multiplier value, a partial result value and a modulus, N.

The modular multiplying device operative to $\underline{\oplus}$ summate into the $\underline{\oplus}$ accumulation device, in turn one or two of a plurality of multiplicand values, in turn, during the phases of the iterative $\underline{\otimes}$ multiplication process, and in turn to receive as multipliers, in turn, inputs from a first value B register, second, from an "on the fly" anticipating value, Y_0, as a multiplier to force first emitting right-hand zero output characters in each iteration, and third values from the modulus, N, register.

The multiplicand parallel registers are operative to receive in turn, values from the A, B, and N register sources, and in turn, also a multiplicand zero forcing Y_0, value.

The zero forcing Y_0 detect device is operative to generate a binary string operative to be a multiplier during the first phase of operation and is operative to be a multiplicand in the second phase of each iterative multiplication.

The multiplicand values to be switched into the $\underline{\oplus}$ accumulation device for the first phase consist can be on of four values, a first zero value, a second value, A_i, which is a k character string segment of a multiplicand, A, and a third value N_0, being the first emitting k characters of the modulus, N. The N_0 value is typically switched in at the start of a multiplication, if there is a fourth preload buffer as in Fig. 6. Then when a k character slice of A is input, the A_i value is serially summated with the N_0 value and stored in the fourth buffer.

If a computation is typically on a single k character modulus, then there is no need for the S register, or for the temporary result value, S. If the operand is $2k$ characters or longer, then the manipulations must be iterative, with progressing A_i slices. For squaring operations slices of B are typically snared from the B stream on the fly and preloaded into the A_i preload buffer.

At the first iteration of a multiplication procedure, the temporary result is zero.

Subsequent temporary results from previous iterations, are operative to be $\underline{\oplus}$ summated with the value emanating from the $\underline{\oplus}$ accumulation device, to generate a partial result for the next in turn iteration;

The multiplicand values to be input, in turn, into the $\underline{\oplus}$ accumulation device for the second phase being, a first zero value, which is a pseudo register value, a second A_i operand, remaining in place from the first phase, and a third Y_0 value having been anticipated in the first phase operative to continue multiplying the remaining characters of the N modulus.

The multiplier values input into the multiplying device in the first phase being a first emitting string, B_0, being the first emitting string segment of the B operand, concurrently $\underline{\otimes}$ multiplying with the second $\otimes$ multiplier value consisting of the anticipated Y_0 string which is simultaneously loaded character by character as it is generated into a preload multiplicand buffer for the second phase;

The two multiplier values input into the apparatus during the second phase being the left hand n - k character values from the B operand, designated $\underline{B}$, and the left hand n - k characters of the N modulus, designated $\underline{N}$, respectively.

The third phase is a flush out of the device operative to transfer the left hand segment of a result value remaining in the $\underline{\oplus}$ accumulation device. This can either be a single clock data dump, or a simple serial unload, driven by zero characters fed in the multiplier inputs.

If the dump is a parallel dump, some means for comparing to decide if the result demands an additional reduction by the modulus.

One of the more innovative enhancements in the invention, is the reverse mode multiplication in the GF(2^g). Because of the lack of interaction between adder cells in this arithmetic, it is possible to perform multiplication and reduction starting from the MS end of the product, thereby having a product that is the modular reduced answer, without a burdensome parasite, caused by disregarded zeroes which, are tantamount to performing a right shift in conventional Montgomery multiplication.

The second innovation that allows for automatic zero forcing is an extension of the Y_0 function of patent application P2, which describes a device wherein only one bit was anticipated at a time. There the J_{00} bit, only, multiplied the single bit XORed values. Both the multiplicative inverse of an odd number and its negative value produce an odd

number. This saved implementing a look up table or a random logic circuit to compute the J_0 value for $\ell = 1$. Note, J_0 is a different quantity in non-alike number systems. We have shown in this extension how a Y_0 value can be derived, for both relevant number fields.

The following describes the elements of the circuitry operative to anticipate the Y_0 value using first emitting values of the multiplicand, and present inputs of the B multiplier, carry out values from the $\oplus$ accumulation device, $\oplus$ summation values from the $\oplus$ accumulation device, the present values from the previously computed partial result, and carry out values from the $\oplus$ adder which $\oplus$ summates the result from the $\oplus$ accumulation device with the previous partial result.

Stated differently, the six values operative to control the zero forcing function, are:

i. the ℓbit S_{out} bits of the result of the ℓbit by ℓ bit mod 2^{ℓ} $\otimes$ multiplication of the right-hand character of the A_i register times the B_d character of the B Stream, $A_0 \otimes B_d \bmod 2^{\ell}$;

ii. the first emitting carry out character from the $\oplus$ accumulation device, $\mathcal{S}(CO_0)$;

iii. the ℓ bit S_{out} character from the second from the right-hand character emitting cell of the $\oplus$ accumulation device, SO_1;

iv. the next in turn character value from the S stream, S_d;

v. the ℓbit carry out character from the Z output full adder, $\mathcal{S}(CO_z)$;

vi. the ℓbit J_0 value, which is the negative multiplicative inverse of the right-hand character in the N_0 modulus multiplicand register;

wherein values, $A_0 \otimes B_d \bmod 2^{\ell}$, $\mathcal{S}(CO_0)$, SO_1, S_d are $\oplus$ added character to character together and "on the fly" $\otimes$ multiplied by the J_0 character to output a valid Y_0 zero-forcing anticipatory character to force an ℓbit egressing character string of zeroes.

Just as in P1, in order to determine if an output must be modular reduced, a sensor operative to compare the output result to N, the modulus, the mechanism operative to actuate a second subtractor on the output of the result register, thereby to output a

modular reduced value which is limited congruent to the output result value precluding the necessity to allot a second memory storage for a smaller result.

The single $\oplus$ accumulation device, configured to perform multiplication, and an anticipating zero forcing mechanism together are operative to perform a series of interleaved $\otimes$ modular multiplications and squarings. The total device performs the equivalent of three integer multiplications, as in a conventional Montgomery method, J_0 is a k character device multiplying the first k character summation of $B_0 \otimes A_i$ and S_i, and in finally using the Y_0 to multiply N.

Whilst the SuperMAP is computing the last iteration of a multiplication, the first slice of a next multiplication can be preloaded into a preload register buffer means on the fly. This value may be the result of a previous multiplication or a slice of a multiplicand residing in one of the register segments in the register bank of Fig. 1 or Fig. 5.

The preloaded value which is a $\oplus$ summation of two multiplicands is $\oplus$ summated into a k character register, only, for GF(2^g) computations. In GF(p) computations, provision must be made for an additional carry bit.

Especially for very long moduli, buffers and registers adjacent to the SuperMAP typically have insufficient memory resources. Means for loading operands directly into preload buffers is provided, operative to store operands in the CPU's memory map. For reverse format multiplication, bit order of input words from the CPU are typically reversed in the Data In and Data Out devices.

BRIEF DESCRIPTION OF THE DRAWINGS

In the drawings:

Thick lines designate k character (kl bit) wide parallel bus lines. Thinner contiguous signal lines depict l bit wide lines. Most control lines are not depicted; those that are included are typically necessary to understand procedures and are typically drawn as dash-dot-dash lines

Fig. 1 is a block diagram of the apparatus according to an embodiment of the invention where four main registers are depicted and the serial data flow path to the operational unit is shown and the input and output data path to the host CPU of Fig. 3;

Fig. 2 is a block diagram of an embodiment of an operational unit operative to manipulate data from Fig. 1;

Fig. 3 is a simplified block diagram of a preferred embodiment of a complete single chip, monolithic cryptocomputer, typically in smart cards;

Fig. 4 is a simplified block diagram of a preferred embodiment of a complete single chip monolithic cryptocomputer wherein a data disable switch is operative to provide for accelerated unloading of data from the operational unit;

Fig. 5 is a simplified block diagram of a data register bank, operative to generate J_0;

Fig. 6 is a simplified block diagram of an operational unit, wherein the Y_0 sense is a device operative to force a zero first phase output;

Fig. 7A is a block diagram of the main computational part of Fig. 6, with circled numbered sequence icons relating to the timing diagrams and flow charts of Figs. 7B, 7C, and Fig. 7D;

Fig. 7B is an event timing pointer diagram showing progressively the process leading to and including the first iteration of a squaring operation;

Fig. 7C is a detailed event sequence to eliminate the "Next Montgomery Squaring" delays in the first iteration of a squaring sequence iconed pointers relating to Fig. 7A, Fig. 7B, and Fig. 7D;

Fig. 7D illustrates the timing of the computational output, relating to Fig. 7A, 7B, and Fig. 7C; and

Figs. 8A and 8B, taken together describe generation of the Y_0 vector in $GF(2^q)$ and in $GF(p)$. Fig. 8A is a set of look up tables for determining the negative multiplicative inverse of the right hand character of N_0, for $\ell = 2$ and $\ell = 4$ and Fig. 8B describes, in simplified block form the signals that generate the Y_0 function for $\ell = 2$ and $\ell = 4$ in both number fields.

DESCRIPTION OF PREFERRED EMBODIMENTS

In the drawings:

Thick lines designate k character ($k\ell$ bit) wide parallel bus lines. Thinner contiguous connecting signal lines depict ℓ bit wide lines. Typically, control lines are

not depicted; those that are preferably necessary to understand procedures, are typically drawn as dash-dot-dash lines

Figs. 1 - 2, taken together, form a simplified block diagram of a serial-parallel arithmetic logic unit (ALU) constructed and operative in accordance with a preferred embodiment of the present invention. The apparatus of Figs. 1 - 2, preferably include the following components:

Single Multiplexers - Controlled Switching Elements which select one signal or character stream from a multiplicity of inputs of signals and direct it this chosen signal to a single output. Multiplexers are marked M1 to M13, and are intrinsic parts of larger elements.

The Multiplexer and pre-adder, 390, is an array of k $l+1$ multiplexers, and chooses which of the four k or $k+1$ character inputs are to be added into the CSA, 410.

The B (70) and 80), S_A (130), S_B (180), and N (200) and (210) are the four main serial main registers in a preferred embodiment. The S_A is conceptually and practically redundant, but can considerably accelerate very long number computations, and save volatile memory resources, especially in the case where the length of the modulus is $2 \cdot k \cdot m$ characters long.

Serial Adders and Serial Subtractors are logic elements that have two serial character inputs and one serial character output, and summate or perform subtraction on two long strings of characters. Components 90 and 500 are subtractors, 330, and 460 are serial adders. The propagation time from input to output is very small. Serial subtractors 90 and 500 typically reduce B^* to B if B^* is larger than or equal to N and/or S* to S if S* is larger than or equal to N. Serial Subtractor 480, is used, as part of a comparator component to detect if B^* will be larger than or equal to N. Full Adder 330, adds the two character streams which feed the Load Buffer 340, with a value that is equal to the sum of the values in the 290 and 320 Load Buffers.

Fast Loaders and Unloaders, 10 and 20, and 30 and 40, respectively, are devices to accelerate the data flow from the CPU controller. Typically, these devices eliminate the necessity for other direct memory access components. 20 and 40 are for reversing the data word, as is necessary for reversing the data words for reverse format GF(2^g) multiplications.

Data In, 50, is a parallel in serial out device, as the present ALU device is a serial fed systolic processor, and data is fed in, in parallel, and processed in serial.

Data Out, 60, is a serial in parallel out device, for outputting results from the coprocessor. The quotient generator is that part of Fig. 2, which generates a quotient character at each iteration of the dividing mechanism.

Flush Signals on Bd, 240; on S*d, 250; and on Nd, 260, are made to assure that the last $k+1$ characters can flush out the CSA. A second embodiment would reconcile the R data at the end of the second phase, and would perform a single parallel data dump to flush out the CSA.

Load Buffers R1, 290; R2, 320; and R3, 340 are serial in parallel out shift registers adapted to receive the three possible more than zero multiplicand combinations.

Latches L1, 360; L2, 370; and L3, 380; are made to receive the outputs from the load buffers, thereby allowing the load buffers, the temporal enablement to process the next phase of data before this data is preferably latched into L1, L2, and L3. Latch L0 is typically a "virtual" constant all zero input into 390, which typically is not implemented in latched logic.

Y_0 Sense, 430, is the logic device, which determines the number of times the modulus is accumulated, in order that a k character string of LS zeros will exit at Z in $\otimes$ multiplications.

One character delay devices 100, 220 and 230 are inserted in the respective data streams to accommodate for computation synchronization between the data preparation devices in Fig. 1, and the data processing devices in Fig. 1.

The k character delay shift register 470, synchronizes N and the subtractor subtracts N from the result after disregarding the right hand output zero character string for the larger than N comparison.

The Carry Save Accumulator is almost identical to a serial/parallel multiplier, excepting for the fact that three different larger than zero values can be summated, instead of the single value as conventionally is latched onto the input of the s/p multiplier. When used in polynomial based computations "all carry dependent" functions are disabled.

The Insert Last Carry, 440, is used to insert the $(m \cdot k \cdot l+1)$'th bit of the S stream, as the S register is only $m \cdot k$ characters long.

The borrow/overflow detect, 490, typically detects if a result is larger than or equal to the modulus (from N), or in GF(p) computations. In polynomial based computations the overflow is detected if the first significant result bit is a one.

The control mechanism is not depicted, but is preferably understood to be a set of cascaded counting devices with finite state machines for specific functions with switches set for systolic data flow in both GF(p) and GF(2^q).

For modular multiplication in the prime and composite prime field of numbers, we define A and B to be the multiplicand and the multiplier, and N to be the modulus which is typically larger than A or B. N also denotes the register where the value of the modulus is stored. N, may, in some instances, be smaller than A. We define A, B, and N as $m \cdot k = n$ character long operands. Each k character group will be called a segment, the size of the group defined by the size of the multiplying device. Then A, B, and N are each m characters long. For ease in following the step by step procedural explanations, assume that A, B, and N are 512 bits long, ($n = 512$); assume that k is 64 characters long because of the present cost effective length of such a multiplier, and data manipulation speeds of simple CPUs; and $m = 8$ is the number of segments in an operand and also the number of iterations in a squaring or multiplying loop with a 512 bit operand. All operands are positive integers. More generally, A, B, N, n, k and m may assume any suitable values.

In non-modular functions, the N and S registers can be used for temporary storage of other arithmetic operands.

We use the symbol, $\equiv$, to denote congruence of modular numbers, for example $16 \equiv 2 \bmod 7$, and we say 16 is congruent to 2 modulo 7 as 2 is the remainder when 16 is divided by 7. When we write $Y \bmod N \equiv X \bmod N$; both Y and X may be larger than N; however, for positive X and Y, the remainders will be identical. Note also that the congruence of a negative integer Y, is $Y + u \cdot N$, where N is the modulus, and if the congruence of Y is to be less than N, u will be the smallest integer which will give a positive result.

We use the symbol, ¥, to denote congruence in a more limited sense. During the processes described herein, a value is often either the desired value, or equal to the

desired value plus the modulus. For example X¥2 mod 7. X can be equal to 2 or 9. We say X has limited congruence to 2 mod 7. In the polynomial based field, the analog is a monic value, which we say is larger than N, and is reduced by XORing to the modulus. As in GF(2^n), there is no overflow, this Yen value is typically disregarded.

When we write $X = A \bmod N$, we define X as the remainder of A divided by N; e.g., 3 = 45 mod 7.

In number theory the modular multiplicative inverse is a basic concept. For example, the modular multiplicative inverse of X is written as X^{-1}, which is defined by $X \cdot X^{-1} \bmod N = 1$. If $X = 3$, and $N = 13$, then $X^{-1} = 9$, i.e., the remainder of $3 \cdot 9$ divided by 13 is 1.

The acronyms MS and LS are used to signify most significant and least significant when referencing bits, characters, segments, and full operand values, as is conventional in digital nomenclature.

Throughout this specification N designates both the value N, and the name of the shift register which contains N. An asterisk superscript on a value, denotes that the value, as stands, is potentially incomplete or subject to change. A is the value of the number which is to be exponentiated, and n is the character length of the N operand. After initialization when A is "Montgomery normalized" to A^* ($A^* = 2^n \cdot A$ - to be explained later) A^* and N are constant values throughout the intermediate step in the exponentiation. During the first iteration, after initialization of an exponentiation, B is equal to A^*. B is also the name of the register wherein the accumulated value, which finally equals the desired result of exponentiation resides. S^* designates a temporary value, and S, S_A and S_B designate, also, the register or registers in which all but the single MS bit of S is stored. (S^* concatenated with this MS bit is identical to S.) $S(i\text{-}1)$ denotes the value of S at the outset of the i'th iteration; S_0 denotes the LS segment of an $S(i)$'th value.

We refer to the process in the GF(p) field (defined later) $\mathcal{P}(A \cdot B)N$ as multiplication in the P field, or sometimes, simply, a multiplication operation.

As we have used the standard structure of a serial/parallel multiplier as the basis for constructing a double acting serial parallel multiplier, we differentiate between the summating part of the multiplier, which is based on carry save accumulation, (as opposed to a carry look ahead adder, or a ripple adder, the first of which is considerably

more complicated and the second very slow), and call it a carry save adder or accumulator, and deal separately with the preloading mechanism and the multiplexer and latches, which allow us to simultaneously multiply A times B and C times D, while continuously summate both results with a previous result, S, e.g., $A \cdot B + C \cdot D + S$, converting this accumulator into a more versatile engine. Additional logic is added to this multiplier in order to provide for an anticipated sense operation necessary for modular reduction and serial summation necessary to provide for modular arithmetic and ordinary integer arithmetic on very large numbers.

Montgomery Modular Multiplication in GF(p)

The following description refers to Montgomery arithmetic in the GF(p) of numbers. The present device may be used for Montgomery arithmetic on polynomial based numbers in GF(2^n), but would be degraded in performance, as computations would be in the P field, where all executable operands are multiplied by a factor of 2^n.

In a classic approach for computing a modular multiplication, $A \cdot B \bmod N$, the remainder of the product $A \cdot B$ is computed by a division process. Implementing a conventional division of large operands is more difficult to perform than serial/parallel multiplications.

Using Montgomery's modular reduction method, division is essentially replaced by multiplications using two precomputed constants. In the procedure demonstrated herein, there is only one precomputed constant, which is a function of the modulus. This constant is, or can be, computed using this ALU device.

A simplified presentation of the Montgomery process, as is used in this device is now provided, followed by a complete preferred description.

If we have an odd number (an LS bit one), e.g., 1010001 ($=81_{10}$) we can always transform this odd number to an even number (a single LS bit of zero) by adding to it another fixing, compensating odd number, e.g., 1111 ($=15_{10}$); as 1111 + 1010001 = 1100000 (96_{10}). In this particular case, we have found a number that produced five LS zeros, because we knew in advance the whole string, 81, and could easily determine a binary number which we could add to 81, and would produce a new binary number that would have as many LS zeros as we might need. This fixing number must have a right hand one, else it has no effect on the progressive LS characters of a result.

If our process is a clocked serial/parallel carry save process, where it is desired to have a continuous number of LS zeros, and wherein at each clock cycle we only have to fix the next bit, at each clock it is sufficient to add the fix, if the next bit were to be a one or not to add the fix if the anticipated bit were to be a zero. However, in order not to cause interbit overflows (double carries), this fix is preferably summated previously with the multiplicand, to be added into the accumulator when the relevant multiplier bit is one, and the Y Sense also anticipates a one.

Now, as in modular arithmetic, we only are interested in the remainder of a value divided by the modulus, we know that we can add the modulus any number of times to a value, and still have a value that would have the same remainder. This means that we can add $Y \cdot N = \sum y_i \cdot r^i \cdot N$ to any integer, and still have the same remainder; Y being the number of times we add in the modulus, N, to produce the required k right hand zeros. As described, the modulus that we add can only be odd. (Methods exist wherein even moduli are defined as r^i times the odd number that results when i is the number of LS zeros in the even number.)

Montgomery interleaved reductions typically reduce storage requirements, and the cost effective size of the multiplication devices. This is especially useful when performing public key cryptographic functions where we multiply one large integer, e.g., $n = 1024$ bit, by another same length large integer; a process that would ordinarily produce a double length integer.

We can add in Ns (the modulus) enough times to $A \cdot B = X$ or $A \cdot B + S = X$ during the process of multiplication (or squaring) so that we will have a number, Z, that has n LS zeros, and, at most, $n + 1$ MS characters.

We can continue using such numbers, disregarding the LS n characters, if we remember that by disregarding these zeros, we have divided the desired result by r^n.

When the LS n characters are disregarded, and we only use the most significant n (or $n + 1$) characters, then we have effectively multiplied the result by r^{-n}, the modular inverse of r^n. If we would subsequently re-multiply this result by $r^n \bmod N$ (or r^n) we would obtain a value congruent to the desired result (having the same remainder) as $A \cdot B + S \bmod N$. As is seen, using MM, the result is preferably multiplied by r^{2n} to overcome the r^{-n} parasitic factor reintroduced by the MM.

Example:

$A \cdot B + S \bmod N = (12 \cdot 11 + 10) \bmod 13 = (1100 \cdot 1011 + 1010)_2 \bmod 1011_2$.

$l = 1$, $r = 2$

We will add in $2^i N$ whenever a fix is necessary on one of the n LS bits.

B	1011	
× A	1100	
add S	1010	
add $A(0) \cdot B$	0000	
	----	sum of LS bit = 0 not add N
add 2^0 $(N \cdot 0)$	0000	
sum and shift	0101	→0 LS bit leaves carry save adder
add $A(1) \cdot B$	0000	
	----	sum of LS bit = 0 - add N
add $2^1(N \cdot 1)$	1101	
sum and shift	1001	→0 LS bit leaves CS adder
add $A(2) \cdot B$	1011	
	----	sum LS bit = 0 don't add N
add $2^2(N \cdot 0)$	0000	
sum and shift	1010	→0 LS bit leaves CS adder
add $A(3) \cdot B$	1011	
	----	sum LS bit = 1 add N
add $2^3(N \cdot 1)$	1101	
sum and shift	10001	→0 LS bit leaves CS adder

And the result is $10001\ 0000_2 \bmod 13 = 17 \cdot 2^4 \bmod 13$.

As 17 is larger than 13 we subtract 13, and the result is:

$$17 \cdot 2^4 \equiv 4 \cdot 2^4 \bmod 13.$$

formally $2^{-n}(A \cdot B + S) \bmod N = 9\ (12 \cdot 11 + 10) \bmod 13 \equiv 4$

In Montgomery arithmetic we utilize only the MS non-zero result, 4, and effectively remember that the real result has been divided by 2^n; n zeros having been forced onto the MM result.

We have added in $(8+2)\cdot 13 = 10\cdot 13$ which effectively multiplied the result by $2^4 \bmod 13 \equiv 3$. In effect, had we used the superfluous zeros, we can say that we have performed, $A\cdot B + Y\cdot N + S$ - $(12\cdot 11+10\cdot 13+10)$ in one process, which will be described possible on a preferred embodiment.

Check- $(12\cdot 11+10) \bmod 13 = 12$; $4 \cdot 3 = 12$.

In summary, the result of a Montgomery Multiplication is the desired result multiplied by 2^{-n}.

To retrieve the previous result back into a desired result using the same multiplication method, we would have to Montgomery Multiply the previous result by 2^{2n}, which we will call H, as each MM leaves us with a parasitic factor of 2^{-n}.

The Montgomery Multiply function $\mathcal{P}(A\cdot B)N$ performs a multiplication modulo N of the $A\cdot B$ product into the P field. (In the above example, where we derived 4). The retrieval from the P field back into the normal modular field is performed by enacting P on the result of $\mathcal{P}(A\cdot B)N$ using the precomputed constant H. Now, if $P \equiv \mathcal{P}(A\cdot B)N$, it follows that $\mathcal{P}(P\cdot H)N \equiv A\cdot B \bmod N$; thereby performing a normal modular multiplication in two P field multiplications.

Montgomery modular reduction averts a series of multiplication and division operations on operands that are n and $2n$ characters long, by performing a series of multiplications, additions, and subtractions on operands that are n or $n+1$ characters long. The entire process yields a result which is smaller than or equal to N. For given A, B and odd N there is always a Q, such that $A\cdot B + Q\cdot N$ will result in a number whose n LS characters are zero, or:

$$P\cdot 2^n = A\cdot B + Q\cdot N$$

This means that we have an expression that is $2n$ characters long (with a possible one bit overflow), whose n LS characters are zero.

Now, for radix $r = 2^j$; let $I\cdot r^n \equiv 1 \bmod N$ (I exists for all odd N). Multiplying both sides of the previous equation by I yields the following congruences:

from the left side of the equation:

$P \cdot I \cdot r^n \equiv P \bmod N$; (Remember that $I \cdot r^n \equiv 1 \bmod N$)

and from the right side:

$A \cdot B \cdot I + Q \cdot N \cdot I \equiv A \cdot B \cdot I \bmod N$; (Remember that $Q \cdot N \cdot I \equiv 0 \bmod N$)

therefore:

$P \equiv A \cdot B \cdot I \bmod N$.

This also means that a parasitic factor $I = r^{-n} \bmod N$ is introduced each time a P field multiplication is performed.

We define the P operator such that:

$P \equiv A \cdot B \cdot I \bmod N \equiv \mathcal{P}(A \cdot B)N$.

and we call this "multiplication of A times B in the $\mathcal{P}$ field", or Montgomery Multiplication.

The retrieval from the P field can be computed by operating $\mathcal{P}$ on $P \cdot H$, making:

$\mathcal{P}(P \cdot H)N \equiv A \cdot B \bmod N$;

We can derive the value of H by substituting P in the previous congruence.

We find:

$\mathcal{P}(P \cdot H)N \equiv (A \cdot B \cdot I)(H)(I) \bmod N$;

(see that $A \cdot B \cdot I \leftarrow P$; $H \leftarrow H$; $I \leftarrow$ and any multiplication operation introduces a parasitic I)

If H is congruent to the multiple inverse of I^2 then the congruence is valid, therefore:

$H = I^{-2} \bmod N \equiv r^{2n} \bmod N$

(H is a function of N and we call it the H parameter)

In conventional Montgomery methods, to enact the P operator on $A \cdot B$, the following process may be employed, using the precomputed constant J:

1) $X = A \cdot B$
2) $Y = (X \cdot J) \bmod r^n$ (only the n LS characters are necessary)
3) $Z = X + Y \cdot N$
4) $S = Z / r^n$ (The requirement on J is that it forces Z to be divisible by r^n)
5) $P = S \bmod N$ (N is to be subtracted from S, if $S \geq N$)

Finally, at step 5):

$P \neq \mathcal{P}(A \cdot B)N$,

[After the subtraction of N, if necessary:

$P = \mathcal{P}(A \cdot B)N$]

Following the above:

$Y = A \cdot B \cdot J \bmod r^n$ (using only the n LS characters);

and:

$Z = A \cdot B + (A \cdot B \cdot J \bmod r^n) \cdot N$.

In order that Z be divisible by r^n (the n LS characters of Z are preferably zero) and the following congruence will exist:

$$[A \cdot B + (A \cdot B \cdot J \bmod r^n) \cdot N] \bmod r^n \equiv 0$$

In order that this congruence will exist, $N \cdot J \bmod r^n$ is congruent to -1 or:

$$J \equiv -N^{-1} \bmod r^n.$$

and we have found the constant J.

J, therefore, is a precomputed constant which is a function of N only. However, in a machine that outputs a MM result, character by character, provision should be made to add in Ns at each instance where the output character in the LS string would otherwise have been a zero, thereby obviating the necessity of precomputing J and subsequently computing $Y = A \cdot B \cdot J \bmod r^n$, as Y can be detected character by character using hardwired logic. We have also described that this method can only work for odd Ns.

Therefore, as is apparent, the process described employs three multiplications, one summation, and a maximum of one subtraction, for the given A, B, N, and a precomputed constant to obtain $\mathcal{P}(A \cdot B)N$. Using this result, the same process and a precomputed constant, H, (a function of the module N) we are able to find $A \cdot B \bmod N$. As A can also be equal to B, this basic operator can be used as a device to square or multiply in the modular arithmetic.

Interleaved Montgomery Modular Multiplication

The previous section describes a method for modular multiplication which involved multiplications of operands which were all n characters long, and results which required $2n + 1$ characters of storage space.

Using Montgomery's interleaved reduction as described in P1, it is possible to perform the multiplication operations with shorter operands, registers, and hardware multipliers; enabling the implementation of an electronic device with relatively few logic gates.

First we will describe how the device can work, if at each iteration of the interleave, we compute the number of times that N is added, using the J_0 constant. Later, we describe how to interleave, using a hardwire derivation of Y_0, which will eliminate the J_0+ phase of each multiplication {(2) *in the following example*}, and enable us to integrate the functions of two separate serial/multipliers into the new single generic multiplier which can perform $A \cdot B + C \cdot N + S$ at better than double speed using similar silicon resources.

Using a k character multiplier, it is convenient to define segments of k character length; there are m segments in n characters; i.e., $m \cdot k = n$.

J_0 will be the LS segment of J.

Therefore:

$J_0 \equiv -N_0^{-1} \bmod r^k$ (J_0 exists as N is odd).

Note, the J and J_0 constants are compensating numbers that when enacted on the unreduced output, tell us how many times to add the modulus, in order to have a predefined number of least significant zeros. We will later describe an additional advantage to the present serial device; since, as the next serial bit of output can be easily determined, we can always add the modulus (always odd) to the next intermediate result. This is the case if, without this addition, the output character, the LS serial bit exiting the CSA, would have been a "1"; thereby adding in the modulus to the previous even intermediate result, and thereby promising another LS zero in the output string. Remember, congruency is maintained, as no matter how many times the modulus is added to the result, the remainder is constant.

In the conventional use of Montgomery's interleaved reduction, $\mathcal{P}(A \cdot B)N$ is enacted in m iterations as described in steps (1) to (5):

Initially $S(0) = 0$ (the ¥ value of S at the outset of the first iteration).

For $i = 1, 2 \quad m$:

1) $X = S(i\text{-}1) + A_{i-1} \cdot B$ (A_{i-1} is the i-1 'th character of A ; $S(i\text{-}1)$ is the value

of S at the outset of the i 'th iteration.)

2) $Y_0 = X_0 \cdot J_0 \bmod r^k$ (The LS k characters of the product of $X_0 \cdot J_0$)

(The process uses and computes the k LS characters only, e.g., the least significant 64 characters) In the preferred implementation, this step is obviated, because in a serial machine Y_0 can be anticipated character by character.

3) $Z = X + Y_0 \cdot N$

4) $S(i) = Z/r^k$ (The k LS characters of Z are always 0, therefore Z is always divisible by r^k. This division is tantamount to a k character right shift as the LS k characters of Z are all zeros; or as will be seen in the circuit, the LS k characters of Z are simply disregarded.

5) $S(i) = S(i) \bmod N$ (N is to be subtracted from those $S(i)$'s which are larger than N). Finally, at the last iteration (after the subtraction of N, when necessary), $C = S(m) = \mathcal{P}(A \cdot B)N$.

To derive $F = A \cdot B \bmod N$, the P field computation, $\mathcal{P}(C \cdot H)N$, is performed.

It is desired to know, in a preferred embodiment, that for all $S(i)$'s, $S(i)$ is smaller than $2N$. This also means, that the last result ($S(m)$) can always be reduced to a quantity less than N with, at most, one subtraction of N.

We observe that for operands which are used in the process:

$S(i\text{-}1) < r^{n+1}$ (the temporary register can be one bit longer than the B or N register),

$B < N < r^n$ and $A_{i-1} < r^k$.

By definition:

$S(i) = Z/r^k$ (The value of S at the end of the process, before a possible subtraction)

For all Z, $Z(i) < r^{n+k+1}$.

$X_{max} = S_{max} + A_i \cdot B < r^{n+1} - 1 + (r^k\text{-}1)(r^n\text{-}1)$

$Q_{max} = Y_0 N < (r^k\text{-}1)(r^n\text{-}1)$

therefore:

$Z_{max} < r^{k+n+1} - r^{k+1} + 1 < r^{k+n+1} - 1$.

and as Z_{max} is divided by r^k:

$$S(m) < r^{n+1} - r^1.$$

Because $N_{min} > r^n - r$, $S(m)_{max}$ is always less than $2 \cdot N_{min}$, and therefore, one subtraction is all that is necessary on a final result.

$$S(m)_{max} - N_{min} = (r^{n+1} - r^1 - 1) - (r^n - 1) = r^n - 4 < N_{min}.$$

Example of a Montgomery interleaved modular multiplication:

The following computations in the hexadecimal format clarify the meaning of the interleaved method:

N = a59, (the modulo), A = 99b, (the multiplier), B = 5c3 (the multiplicand), n = 12, r = 2, (the character length of N), k = 4, (the size in characters of the multiplier and also the size of a segment), and m = 3, as $n = k \cdot m$.

$J_0 = 7$ as $7 \cdot 9 \equiv -1 \bmod 16$ and $H \equiv 2^{2 \cdot 12} \bmod a59 \equiv 44b$.

The expected result is $F \equiv A \cdot B \bmod N \equiv 99b \cdot 5c3 \bmod a59 \equiv 375811 \bmod a59 = 220_{16}$.

Initially: $S(0) = 0$

Step 1 $X = S(0) + A_0 \cdot B = 0 + b \cdot 5c3 = 3f61$

$Y_0 = X_0 \cdot J_0 \bmod r^k = 7$ (Y_0 - hardwire anticipated in SuperMAP)

$Z = X + Y_0 \cdot N = 3f61 + 7 \cdot a59 = 87d0$

$S(1) = Z / r^k = 87d$

Step 2 $X = S(1) + A_1 \cdot B = 87d + 9 \cdot 5c3 = 3c58$

$Y_0 = X_0 \cdot J_0 \bmod r^k = 8 \cdot 7 \bmod 2^4 = 8$ (Hardwire anticipated)

$Z = X + Y_0 \cdot N = 3c58 + 52c8 = 8f20$

$S(2) = Z / r^k = 8f2$

Step 3 $X = S(2) + A_2 \cdot B = 8f2 + 9 \cdot 5c3 = 3ccd$

$Y_0 = d \cdot 7 \bmod 2^4 = b$ (Hardwire anticipated)

$Z = X + Y_0 \cdot N = 3ccd + b \cdot a59 = aea0$

$S(3) = Z / r^k = aea$,

as $S(3) > N$,

$S(m) = S(3) - N = aea - a59 = 91$

Therefore $C = \mathcal{P}(A \cdot B)N = 91_{16}$.

Retrieval from the P field is performed by computing $\mathcal{P}(C \cdot H)N$:

Again initially: $S(0) = 0$

Step 1 $X = S(0) + C_0 \cdot H = 0 + 1 \cdot 44b = 44b$

$Y_0 = d$ (Hardwire anticipated in SuperMAP)

$Z = X + Y_0 \cdot N = 44b + 8685 = 8ad0$

$S(1) = Z / r^k = 8ad$

Step 2 $X = S(1) + C_1 \cdot H = 8ad + 9 \cdot 44b = 2f50$

$Y_0 = 0$ (Hardwire anticipated in SuperMAP)

$Z = X + Y_0 \cdot N = 2f50 + 0 = 2f50$

$S(2) = Z / r^k = 2f5$

Step 3 $X = S(2) + C_2 \cdot H = 2f5 + 0 \cdot 44b = 2f5$

$Y_0 = 3$ (Hardwire anticipated in SuperMAP)

$Z = X + Y_0 \cdot N = 2f5 + 3 \cdot a59 = 2200$

$S(3) = Z / r^k = 220_{16}$

which is the expected value of 99b·5c3 mod a59.

If at each step we disregard k LS zeros, we are in essence multiplying the n MS characters by r^k. Likewise, at each step, the i 'th segment of the multiplier is also a number multiplied by r^{ik}, giving it the same rank as $S(i)$.

It can also be noted that in another preferred embodiment, wherein it is of some potential value to know the J_0 constant, if $A_i \cdot B + S = 1$; then $Y_0 = -N_0^{-1} = J_0$

Exponentiation:

The following derivation of a sequence [D. Knuth, *The art of computer programming*, vol. 2: Seminumerical algorithms, Addison-Wesley, Reading Mass., 1981] hereinafter referred to as "Knuth", explains a sequence of squares and multiplies, which implements a modular exponentiation.

After precomputing the Montgomery constant, $H = 2^{2n}$, as this device can both square and multiply in the P field, we compute:

$C = A^E \bmod N$.

Let $E(j)$ denote the j bit in the binary representation of the exponent E, starting with the MS bit whose index is 1 and concluding with the LS bit whose index is q, we can exponentiate as follows for odd exponents:

$A^* ¥ \mathcal{P}(A \cdot H)N$ A^* is now equal to $A \cdot 2^n$.

$B = A^*$

FOR $j = 2$ TO q-1

$B ¥ \mathcal{P}(B \cdot B)N$

IF $E(j) = 1$ THEN

$B ¥ \mathcal{P}(B \cdot A^*)N$

ENDFOR

$B ¥ \mathcal{P}(B \cdot A)N$ $E(0) = 1$; B is the last desired temporary result multiplied by 2^n,

A is the original A.

$C = B$

$C = C - N$ if $C \geq N$.

After the last iteration, the value B is ¥ to $A^E \bmod N$, and C is the final value.

To clarify, we shall use the following example:

$E = 1011 \longrightarrow E(1) = 1; E(2) = 0; E(3) = 1; E(4) = 1;$

To find $A^{1011} \bmod N$; $q = 4$

$A^* = \mathcal{P}(A \cdot H)N = A \cdot I^{-2}\ I = A \cdot I^{-1} \bmod N$

$B = A^*$

FOR $j = 2$ to q

$B = \mathcal{P}(B \cdot B)N$ which produces: $A^2(I^{-1})^2 \cdot I = A^2 \cdot I^{-1}$

$E(2) = 0$; $B = A^2 \cdot I^{-1}$

$j = 3$ $B = \mathcal{P}(B \cdot B)N = A^2(I^{-1})^2 \cdot I = A^4 \cdot I^{-1}$

$E(3) = 1$ $B = \mathcal{P}(B \cdot A^*)N = (A^4 \cdot I^{-1})\ (A \cdot I^{-1}) \cdot I = A^5 \cdot I^{-1}$

$j = 4$ $B = \mathcal{P}(B \cdot B)N = A^{10} \cdot I^{-2} \cdot I = A^{10} \cdot I^{-1}$

As $E(4)$ was odd, the last multiplication will be by A, to remove the parasitic I^{-1}.

$B = \mathcal{P}(B \cdot A)N = A^{10} \cdot I^{-1} \cdot A \cdot I = A^{11}$

$C = B$

A method for computing the H parameter by a reciprocal process is described in US Patent 5,513,133.

Reference is now made to Fig. 3, which is a simplified block diagram showing how the present invention may be implemented in smart cards and other security devices. An internal bus, 500, links components including a CPU, 502, a RAM, 504, non-volatile memory, 506, controlled access EEPROM, 508, and modular arithmetic coprocessor, 510. As shown herein, the coprocessor, 510, is connected via data, 512, and control, 514, registers to the internal bus, 500. The controlled access ROM, 508, is connected via address and data latch means, 516, and a control and test register, 518. Various other devices may be attached to the bus such as a physical sequence random generator, 520, security logic, 522, smart card and external port interfacing circuitry, 524, and 526, respectively.

When a cryptographic program, such as verifying an RSA signature is executed, it may require modular arithmetic functions such as modular exponentiation. The cryptographic program that calls the cryptographic function is preferably run on the CPU, 502.

Reference is now made to Fig. 4, which is another simplified block diagram of an implementation of the present invention for use in a smart card. Parts that are the same as those shown in Fig. 3 are given the same reference numerals and are not described again, except as necessary for an understanding of the present embodiment. In Fig. 4 the CPU 502 is shown with an external accumulator 7350. Data Disable Switch, 7340, detaches the CPU Accumulator from the Data Bus 500, while unloading data from the arithmetic coprocessor enables direct transfer of data from the SMAP to memory.

Fig. 5 is a simplified block diagram of a preferred embodiment of a data register bank, 6205, within a coprocessor 6075, as depicted in coprocessors of Figs. 2, 6 and 7, with a J_0 generator, wherein the J_0 generator typically compiles an l bit primary zero forcing function.

The coprocessor 6075 is connected to a data bus with a CPU as in previous figures. A register bank, 6205, comprises a B register 6070, an A register 6130, an S register 6180, and an N register 6200. The outputs of each of the registers are connected

to a serial data switch and serial process conditioner 6020, which in turn is connected to an operational unit, 6206, which carries out the modular arithmetic operations. Connected between the N register, 6200, and the operational unit, 6206, is a J_0 generator, 552.

In the embodiment the J_0 generator compiles an l bit primary zero forcing function for use in the modular arithmetic functions described above.

Fig. 6 is a simplified internal block diagram of the operational unit of Fig. 5. The unit, preferably supports accelerated squaring operations, in that the additional Y_0B_0 serial buffer accepts Y_0 in the first phase, and in the second phase a modular reduced B_0 for a subsequent squaring operation, wherein it is found that B is larger than N.

Reference is now made to Fig. 7A, which is a block diagram of the main computational part of the operational unit of Fig. 6. Numbers appearing in circles relate to the sequence diagrams of Figs. 7B and 7D.

Reference is now made to Fig. 7B, which is an event timer pointer diagram showing progressively the process leading to and including the first iteration of a squaring operation.

Reference is now made to Fig. 7C which is a generalized event sequence showing a method of eliminating the Next Montgomery Squaring delays in a first iteration of a squaring sequence. Circled numbers refer to Figs. 7A, 7B and 7D.

Reference is now made to Fig. 7D which is a generalized event timer pointer diagram illustrating the timing of the computational output of the first iteration of a squaring operation.

Reference is now made to Fig 8A, a set of look up tables, which typically show the choices of J_0, which is the negative of the multiplicative inverse over modulus 2^l of the right hand character of N_0. As N_0 is always either monic for GF(2^p) or odd for GF(p), J_0 always exists.

In Figs. 8A and 8B, we refer to this right hand character of the modulus as N_0. We refer to N_{0j} as the j'th bit of the locally defined N_0 character

Fig. 8B is a schematic for designing either a 4 bit or a 2 bit Y_0 zero forcing function character. The variable inputs into the force function are the N_0 bits (constant throughout a multiplication), the l, S_0 bits, and the l right hand bits of the product of the

l multiplier and multiplicand bits, A_{i0} and B_{0j}, and the carry switch, $\mathcal{S}$, which determines whether functions work in GF(2^q) or GF(p). The A and B bits are input into a $\otimes$ multiplier and $\oplus$ added to the S_0. When $\mathcal{S} = 0$, all carries are disabled.

It is appreciated that various features of the invention, which are, for clarity, described in the contexts of separate embodiments, may also be provided in combination in a single embodiment. Conversely, various features of the invention, which are, for brevity, described in the context of a single embodiment, may also be provided separately or in any suitable subcombination.

It will be appreciated by persons skilled in the art, that the present invention is not limited to what has been particularly shown and described hereinabove. Rather, the scope of the present invention includes both combinations and subcombinations of the various features described hereinabove as well as variations and modifications thereof, which would occur to persons skilled in the art upon reading the foregoing description which are not in the prior art.

In the following claims, symbols such as have the meanings given in the preceding description.

CLAIMS

1. A microelectronic apparatus for performing $\otimes$ multiplication and squaring in both polynomial based GF(2^q) and GF(p) field arithmetic, squaring and reduction using a serial fed radix 2^ℓ multiplier, B, with k character multiplicand segments, A_i, and a k character $\oplus$ accumulator wherein reduction to a limited congruence is performed "on the fly", in a systolic manner, with A_i, a multiplicand, times B, a multiplier, over a modulus, N, and a result being at most $2k+1$ characters long, including the k first emitting disregarded zero characters, which are not saved, where k characters have no less bits than the modulus, the apparatus comprising;

a first (B), and second (N) main memory register means, each register operative to hold at least n bit long operands, respectively operative to store a multiplier value designated B, and a modulus, denoted N, wherein the modulus is smaller than 2^n;

a digital logic sensing detector, Y_0, operative to anticipate "on the fly" when a modulus value is to be $\oplus$ added to the value in the $\oplus$ adder accumulator device such that all first k characters emitting from the device are forced to zero;

a modular multiplying device for at least k character input multiplicands, with only one, at least k characters long $\oplus$ adder, $\oplus$ summation device operative to accept k character multiplicands, the $\otimes$ multiplication device operative to switch into the $\oplus$ accumulator device, in turn, multiplicand values, and in turn to receive multiplier values from a B register, and an "on the fly" simultaneously generated anticipated value as a multiplier which is operative to force k first emitting zero output characters in the first phase, wherein at each effective machine cycle at least one designated multiplicand is $\oplus$ added into the $\oplus$ accumulation device;

the multiplicand values to be switched in turn into the $\oplus$ accumulation device consisting of one or two of the following three multiplicands, the first multiplicand being an all-zero string value, a second value, being the multiplicand A_i, and a third value, the N_0 segment of the modulus;

an apparatus to anticipate the ℓ bit k character serial input Y_0 multiplier values;

the multiplier values which are input in turn into the multiplying device in the first phase being first the B operand, and concurrently, the second multiplier value consisting of the Y_0, "on the fly" anticipated k character string, to force first emitting zeroes in the output;

an $\underline{\oplus}$ accumulation device, operative to output values simultaneously as multiplicands are $\underline{\oplus}$ added into the $\underline{\oplus}$ accumulation device;

an output transfer mechanism, in the second phase operative to output a final modular $\otimes$ multiplication result from the $\underline{\oplus}$ accumulation device.

2. An apparatus as in claim 1 wherein $\underline{\oplus}$ summations into the $\underline{\oplus}$ accumulation device are activated by each new serially loaded higher order multiplier characters.

3. An apparatus as in claim 1, wherein the multiplier characters;

are operative to cause no $\underline{\oplus}$ summation into the $\underline{\oplus}$ accumulation device if both the input B character and the corresponding input Y_0 character are zeroes;

are operative to $\underline{\oplus}$ add in only the A_i multiplicand if the input B character is a one and the corresponding Y_0 character is a zero;

are operative to $\underline{\oplus}$ add in only the N, modulus, if the B character is a zero, and the corresponding Y_0 character is a one; and

are operative to $\underline{\oplus}$ add in the $\underline{\oplus}$ summation of the modulus, N, with the multiplicand A_i if both the B input character and the corresponding Y_0 character are ones.

4. An apparatus as in claim 1, operative to preload multiplicand values A_i and N, into two designated preload buffers, and to $\underline{\oplus}$ summate these values into a third multiplicand preload buffer, obviating the necessity of $\underline{\oplus}$ adding in each multiplicand value separately.

5. An apparatus as in claim 1, wherein the multiplier values are serial single character in input and the output of the $\underline{\oplus}$ accumulation device is serial single character output, wherein the Y_0 detect device is operative to anticipate only one character in a clocked turn.

6. An apparatus as in claim 1, wherein the $\underline{\oplus}$ accumulation device performs modulo 2, XOR addition/subtraction, wherein all carry bits in addition and subtraction components are disregarded, thereby precluding provisions for overflow and further limiting convergence in computations.

7. A $\underline{\otimes}$ multiplication apparatus as in claim 1 wherein all carry inputs are disabled to zero, denoted, $\mathcal{S}^{\ell}=0$, typically operative to perform polynomial based multiplication.

8. An apparatus as in claim 1 wherein an $\mathcal{S}^{\ell}$ equal to zero acting on an element in a circuit equation computing in GF(2^{ℓ}), the $\mathcal{S}^{\ell}$ designates omitted circuitry and all adders and subtractors, designated $\underline{\oplus}$ have been reduced to XOR, modulo 2 addition/subtraction elements.

9. An apparatus as in claim 1 wherein k first emitting zeroes will egress from the device controlled by the following four quantities in anticipating the next in turn Y_0 character:

i. the ℓ bit S_{out} bits of the result of the ℓ bit by ℓ bit mod 2^{ℓ} $\underline{\otimes}$ multiplication of the right-hand character of the A_i register times the B_d character of the B Stream, $A_0 \cdot B_d \bmod 2^{\ell}$;

ii. the first emitting carry out character from the $\underline{\oplus}$ accumulation device, $\mathcal{S}^{\ell}(CO_0)$;

iii. the ℓ bit S_{out} character from the second from the right character emitting cell of the $\underline{\oplus}$ accumulation device, SO_1;

iv. the ℓ bit J_0 value, which is the negative multiplicative inverse of the right-hand character in the N_0 modulus multiplicand register.

wherein values, $A_0 \cdot B_d \bmod 2^{\ell}$, $\mathcal{S}^{\ell}(CO_0)$, and SO_1 are $\underline{\oplus}$ added character to character together and "on the fly" multiplied by the J_0 character to output a valid Y_0 zero-forcing anticipatory character to force an ℓ bit egressing string of zeroes.

10. An apparatus as in claim 1, wherein ⊗ multiplication on polynomial based operands is performed in a reverse mode, multiplying from right hand MS characters to left hand LS characters, operative to perform modular reduced ⊗ multiplication without Montgomery type parasitic functions.

11. An apparatus as in claim 1 where the preload buffers are serially fed and where multiplicand values are preloaded into the preload buffers on the fly from a multiplicity of memory devices.

12. An apparatus as in claim 1, wherein a previous value, emitting from an additional n bit register, S, is ⊕ summated into the output value of the ⊕ accumulation device via an ℓ bit ⊕ adder circuit such that first emitting output characters are zeroes when the Y_0 detector is operative to detect the necessity of ⊕ adding moduli to the ⊕ summation in the ⊕ accumulation device, wherein the Y_0 detector is operative to detect utilizing the next in turn ⊕ added characters $A_0 \cdot B_d \bmod 2^{\ell}$, $\mathcal{S}^{\ell}(CO_0)$, SO_1, S_d and $\mathcal{S}^{\ell}(CO_2)$, the composite of ⊕ added characters to be finite field ⊗ multiplied on the fly by the ℓ bit J_0 value, where ⊕ defines the addition and ⊗ defines the multiplication as befits the finite field used in the process.

13. An apparatus as in claim 1, wherein for $\ell = 1$, J_0 is implicitly 1, and the J_0 ⊗ multiplication is implicit, without additional hardware.

14. An apparatus as in claim 1 wherein a comparator is operative to sense a finite field output from the ⊗ modular multiplication device, working in GF(p), where the first right hand emitting k zero characters are disregarded, where the output is larger than the modulus, N, thereby operative to control a modular reduction whence said value is output from the memory register to which the output stream from the multiplier device is destined, and thereby precluding allotting a second memory storage device for the smaller product values.

15. A device as in claim 1 wherein for $\otimes$ modular multiplication in the GF(2^g), the apparatus is operative to multiply without an externally precomputed more than ℓbit zero-forcing factor.

16. A method according to claim 1 operative to compute a J_0 constant by resetting either the A operand value or the B operand value to zero and setting the partial result value, S_0, to 1.

17. A microelectronic apparatus for performing interleaved finite field $\otimes$ modular multiplication of integers A and B operative to generate an output stream of A times B modulus N wherein n the number of characters in the modulus operand register is larger than k, wherein the $\otimes$ multiplication process is performed in iterations, wherein at each interleaved iteration with operands input into a $\otimes$ multiplying device, consisting of N, the modulus, B, a multiplier, a previously computed partial result, S, and a k character string segment of A, a multiplicand, the segments progressing from the A_0 string segment to the A_{m-1} string segment, wherein each iterative result is $\oplus$ summated into a next in turn S, temporary result, in turn, wherein first emitting characters of iterative results are zeroes, the apparatus comprising:

first (B), second (S) and third (N) main memory registers, each register capable of storing and outputting operands, respectively operative to store a multiplier value, a partial result value and a modulus, also denoted N;

a modular multiplying device operative to $\oplus$ summate into the $\oplus$ accumulation device, in turn one or two of a plurality of multiplicand values, in turn, during the phases of the iterative $\otimes$ multiplication process, and in turn to receive as multipliers, in turn, inputs from a first value B register, second, from an "on the fly" anticipating value, Y_0, as a multiplier to force first emitting right-hand zero output characters in each iteration, and third values from the modulus, N, register;

the multiplicand parallel registers operative at least to receive in turn, values from the A, B, and N register sources, and in turn, also a multiplicand zero forcing Y_0, value;

a first emitting zero forcing Y_0 detect device operative to generate a binary string operative to be a multiplier during the first phase and operative to be a multiplicand in the second phase;

multiplicand values to be switched into the accumulation device for the first phase consisting of a first zero value, a second value, A_i, which is a k character string segment of a multiplicand, A, and a third value N_0, being the first emitting k characters of the modulus, N;

a temporary result value, S, resulting from a previous iteration, operative to be summated with the value emanating from the accumulation device, to generate a partial result for the next in turn iteration;

multiplicand values to be input, in turn, into the accumulation device for the second phase being, a first zero value, a second A_i operand, remaining in place from the first phase, and a third Y_0 value having been anticipated in the first phase;

multiplier values input into the multiplying device in the first phase being a first emitting string, B_0, being the first emitting string segment of the B operand, concurrently multiplying with the second multiplier value consisting of the anticipated Y_0 string which is simultaneously loaded character by character as it is generated into a preload multiplicand buffer for the second phase;

the two multiplier values input into the apparatus during the second phase being the left hand $n - k$ character values from the B operand, designated $\underline{B}$, and the left hand $n - k$ characters of the N modulus, designated $\underline{N}$, respectively; and

a multiplying flush out device operative in the last phase to transfer the left hand segment of a result value remaining in the accumulation device into a result register.

18. An apparatus as in claim 17, wherein multiplication on polynomial based operands is performed in a reverse mode, multiplying from MS characters to LS characters, operative to perform modular reduction without Montgomery type parasitic functions.

19. An apparatus operative to anticipate the Y_0 value using first emitting values of the multiplicand, and present inputs of the B multiplier, carry out values from

the accumulation device, summation values from the accumulation device, the present values from the previously computed partial result, and carry out values from the adder which summates the result from the accumulation device with the previous partial result.

20. An apparatus as in claim 19 wherein k first emitting zeroes will egress from the device controlled by the following six quantities in anticipating the next in turn Y_0 character:

i. the ℓ bit S_{out} bits of the result of the ℓ bit by ℓ bit mod 2^{ℓ} multiplication of the right-hand character of the A_i register times the B_d character of the B Stream, $A_0 \cdot B_d \bmod 2^{\ell}$;

ii. the first emitting carry out character from the accumulation device, $\mathcal{S}(CO_0)$;

iii. the ℓ bit S_{out} character from the second from the right-hand character emitting cell of the accumulation device, SO_1;

iv. the next in turn character value from the S stream, S_d;

v. the ℓ bit carry out character from the Z output full adder, $\mathcal{S}(CO_Z)$;

vi. the ℓ bit J_0 value, which is the negative multiplicative inverse of the right-hand character in the N_0 modulus multiplicand register;

wherein values, $A_0 \cdot B_d \bmod 2^{\ell}$, $\mathcal{S}(CO_0)$, SO_1, S_d are added character to character together and "on the fly" multiplied by the J_0 character to output a valid Y_0 zero-forcing anticipatory character to force an ℓ bit egressing character string of zeroes.

21. An apparatus as in claim 17 comprised of at least one sensor operative to compare the output result to N, the modulus, the mechanism operative to actuate a second subtractor on the output of the result register, thereby to output a modular reduced value which is limited congruent to the output result value precluding the necessity to allot a second memory storage for a smaller result.

22. An apparatus as in claim 17 where a value which is a summation of two multiplicands is loaded into a preload character buffer with at least a *k* characters memory means register concurrently whilst one of the values is loaded into a preload buffer.

23. An apparatus with only one accumulation device, and an anticipating zero forcing mechanism operative to perform a series of interleaved modular multiplications and squarings concurrently performing the equivalent of three natural integer multiplication operations, such that a result is an exponentiation.

24. An apparatus as in claim 17 where next in turn used multiplicands are preloaded into preload register buffer means on the fly.

25. An apparatus as in claim 17 where a value which is a summation of two multiplicands is summated into at least a *k* character register concurrently whilst one of the values is loaded into its preload buffer.

26. An apparatus as in claim 17 wherein apparatus buffers and registers are operative to be loaded with values from external memory sources and said buffers and registers are operative to be unloaded into the external memory source during computations, such that the maximum size of the operands is dependent on available memory means.

27. An apparatus as in claim 17 wherein memory register means are typically serial single character in/serial single character out, parallel at least *k* characters in/parallel at least *k* characters out, serial single character in/parallel at least *k* characters out, and parallel *k* characters in/serial single character out.

28. An apparatus as in claim 17 wherein the final phase of a multiplication type iteration, the multiplier inputs are zero characters operative to flush out the left hand segment of the carry save accumulator memory.

29. An apparatus as in claim 17 where next in turn used multiplicands are preloaded into preload memory buffers on the fly.

30. An apparatus as in claim 17 where multiplicand values are preloaded into the preload buffers on the fly from central storage memory means.

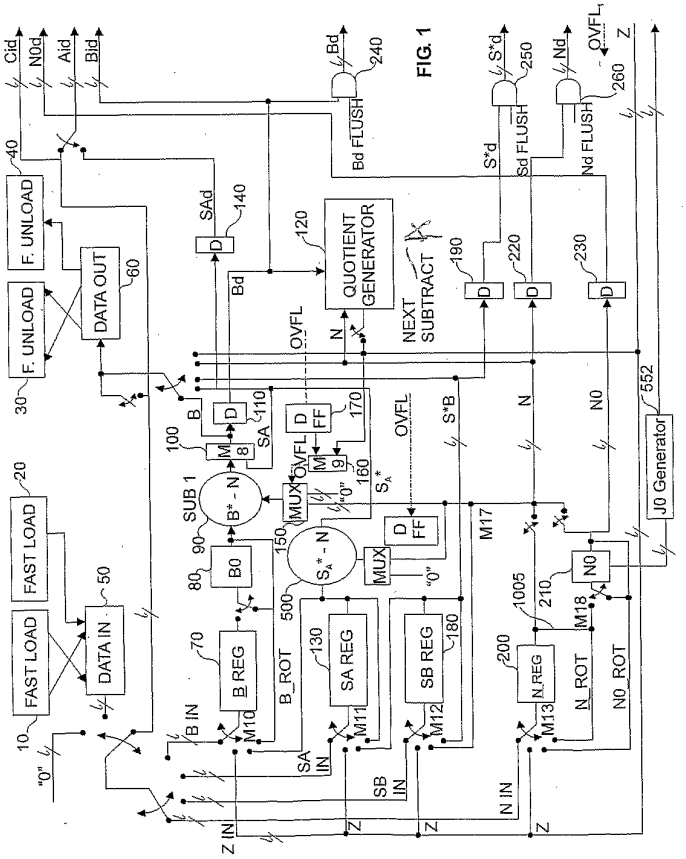
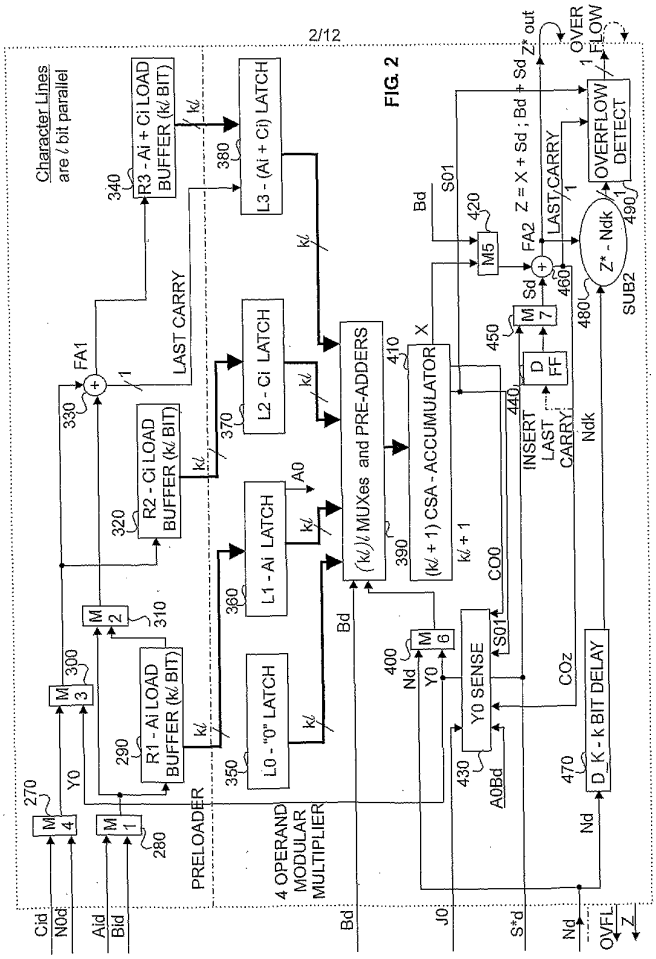

Cid
N0d
Aid
Bid
Bd
240
Bd FLUSH
S*d
250
Sd FLUSH
Nd
260
Nd FLUSH
OVFL1
Z
40
F. UNLOAD
F. UNLOAD
30
DATA OUT
60
SAd
140
120
QUOTIENT GENERATOR
NEXT SUBTRACT
190
220
230
OVFL
N
B
D
SA
110
100
M
8
SUB 1
B* - N
MUX
"0"
9
160
D FF
170
OVFL
S*B
N
N0
552
J0 Generator
20
FAST LOAD
FAST LOAD
DATA IN
50
90
B0
80
150
SA* - N
MUX
"0"
M17
1005
210
N0
M18
500
B_ROT
130
SA REG
SB REG
180
200
N REG
M13
N_ROT
N0_ROT
70
B REG
M10
B IN
M11
M12
10
"0"
SA IN
SB IN
N IN
Z IN
Z

FIG. 1


Character Lines are ℓ bit parallel
FIG. 2
PRELOADER
4 OPERAND MODULAR MULTIPLIER
R1 - Ai LOAD BUFFER (kℓ BIT)
R2 - Ci LOAD BUFFER (kℓ BIT)
R3 - Ai + Ci LOAD BUFFER (kℓ BIT)
L0 - "0" LATCH
L1 - Ai LATCH
L2 - Ci LATCH
L3 - (Ai + Ci) LATCH
(kℓ)/ℓ MUXes and PRE-ADDERS
(k+1) CSA - ACCUMULATOR
Y0 SENSE
D_K - k BIT DELAY
FA1
FA2
LAST CARRY
LAST_CARRY
INSERT LAST CARRY
OVERFLOW DETECT
OVER FLOW
Z* - Ndk
SUB2
Z = X + Sd ; Bd + Sd
Z* out
Cid
N0d
Aid
Bid
Y0
A0
A0Bd
CO0
COz
S01
Nd
Ndk
Bd
J0
S*d
OVFL
Z

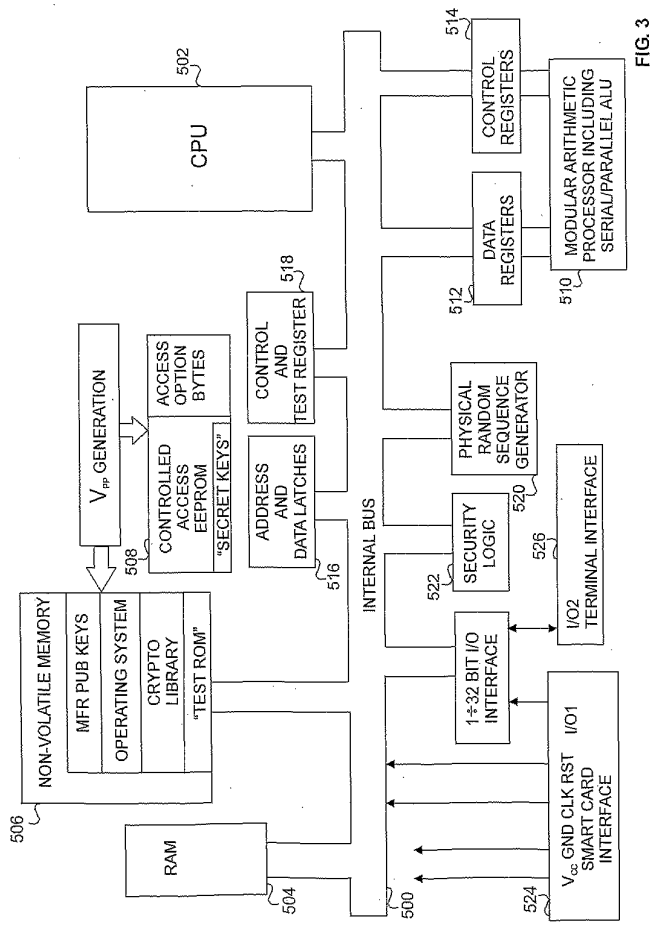

CPU
502
CONTROL REGISTERS
514
DATA REGISTERS
512
MODULAR ARITHMETIC PROCESSOR INCLUDING SERIAL/PARALLEL ALU
510
CONTROL AND TEST REGISTER
518
ACCESS OPTION BYTES
CONTROLLED ACCESS EEPROM
"SECRET KEYS"
508
V_{pp} GENERATION
ADDRESS AND DATA LATCHES
516
PHYSICAL RANDOM SEQUENCE GENERATOR
520
INTERNAL BUS
SECURITY LOGIC
522
I/O2 TERMINAL INTERFACE
526
NON-VOLATILE MEMORY
MFR PUB KEYS
OPERATING SYSTEM
CRYPTO LIBRARY
"TEST ROM"
506
1÷32 BIT I/O INTERFACE
V_{cc} GND CLK RST I/O1 SMART CARD INTERFACE
524
RAM
504
500

FIG. 3

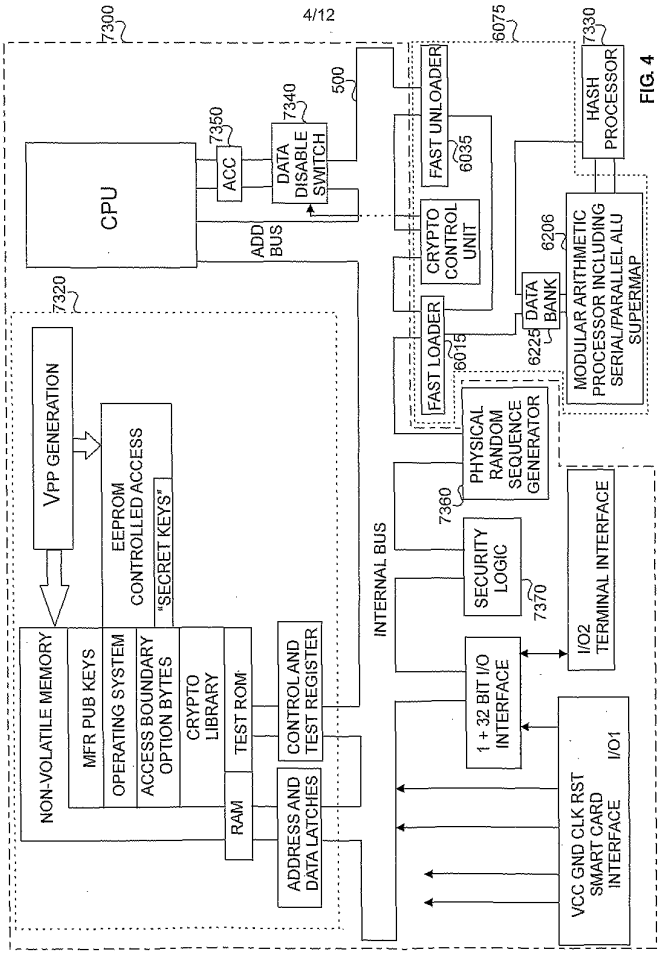

7300
6075
7330
HASH PROCESSOR
500
7340
DATA DISABLE SWITCH
7350
ACC
FAST UNLOADER
6035
CPU
ADD BUS
CRYPTO CONTROL UNIT
6206
DATA BANK
6225
MODULAR ARITHMETIC PROCESSOR INCLUDING SERIAL/PARALLEL ALU SUPERMAP
7320
FAST LOADER
6075
VPP GENERATION
EEPROM CONTROLLED ACCESS
"SECRET KEYS"
PHYSICAL RANDOM SEQUENCE GENERATOR
INTERNAL BUS
7360
SECURITY LOGIC
7370
NON-VOLATILE MEMORY
MFR PUB KEYS
OPERATING SYSTEM
ACCESS BOUNDARY
OPTION BYTES
CRYPTO LIBRARY
TEST ROM
RAM
CONTROL AND TEST REGISTER
ADDRESS AND DATA LATCHES
1 + 32 BIT I/O INTERFACE
I/O2 TERMINAL INTERFACE
VCC GND CLK RST SMART CARD INTERFACE I/O1

FIG. 4

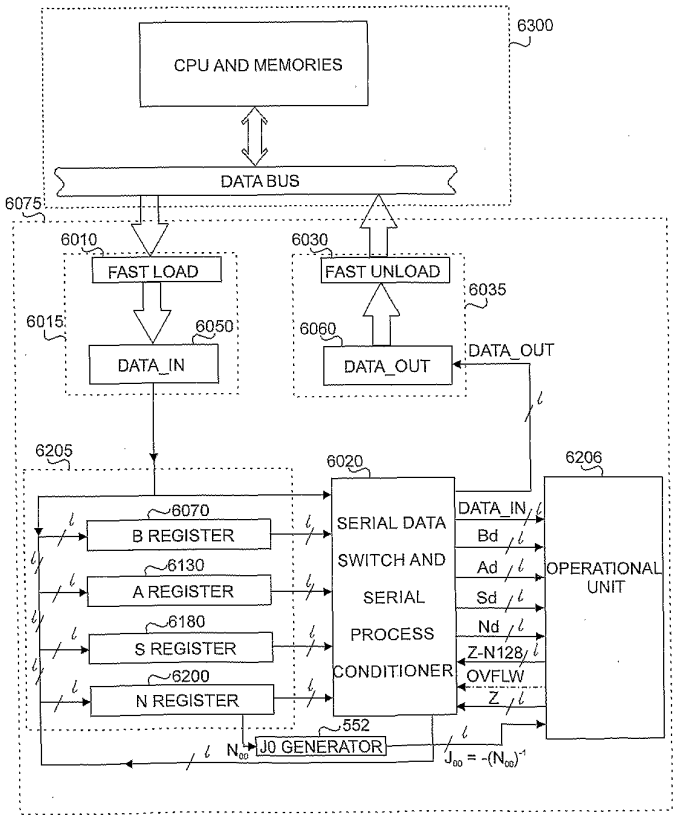

6300
CPU AND MEMORIES
DATA BUS
6075
6010
FAST LOAD
6015
6050
DATA_IN
6030
FAST UNLOAD
6035
6060
DATA_OUT
DATA_OUT
6205
6020
6206
6070
B REGISTER
6130
A REGISTER
6180
S REGISTER
6200
N REGISTER
SERIAL DATA SWITCH AND SERIAL PROCESS CONDITIONER
DATA_IN
Bd
Ad
Sd
Nd
Z-N128
OVFLW
Z
OPERATIONAL UNIT
552
J0 GENERATOR
N_{00}
$J_{00} = -(N_{00})^{-1}$

FIG. 5

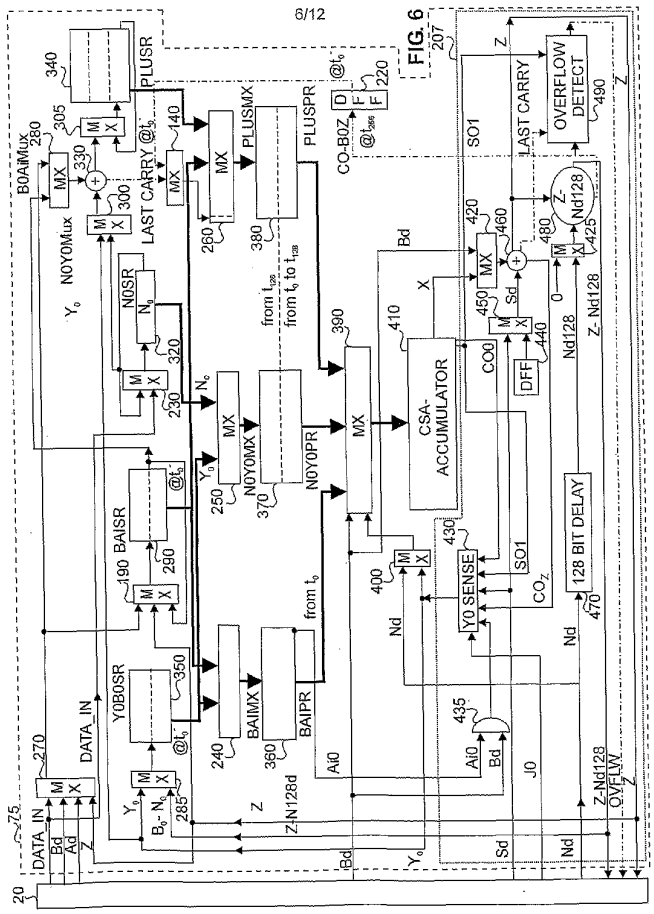


FIG. 6
DATA_IN
DATA_IN
Bd
Ad
Z
Y0B0SR
BAISR
NOSR
NOYOMux
B0AIMux
PLUSR
LAST CARRY
PLUSMX
PLUSPR
CO-B0Z
BAIMX
BAIPR
NOYOMX
NOY0PR
CSA-ACCUMULATOR
DFF
Y0 SENSE
128 BIT DELAY
LAST CARRY
OVERFLOW DETECT
SO1
CO0
CO_Z
Ai0
Bd
J0
Sd
Nd
Z-Nd128
OVFLW
Z-N128d
from t_0
from t_128
from t_0 to t_128
Y_0
N_0
B_0 - N_0
Z
Nd128
Z - Nd128
0
X
M
MX
DFF
20
75
140
190
220
230
240
250
260
270
280
285
290
300
305
320
330
340
350
360
370
380
390
400
410
420
425
430
435
440
450
460
470
480
490
207

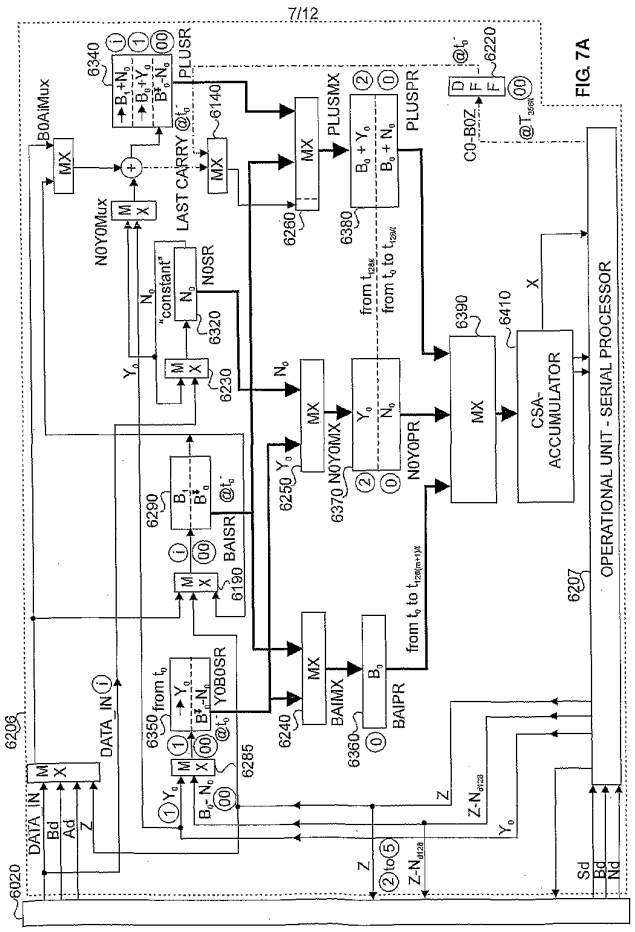

6206
6020
DATA_IN
Bd
Ad
Z
M
X
6350
from t_0
Y_0
B_0-N_0
@t_0
00
6285
Y0B0SR
6240
MX
BAIMX
6360
B_0
BAIPR
from t_0 to $t_{128(n+1)X}$
6250
Y_0
6370
N0Y0MX
Y_0
N_0
N0Y0PR
6380
PLUSMX
$B_0 + Y_0$
$B_0 + N_0$
PLUSPR
from t_{128X}
from t_0 to t_{128X}
6260
6140
LAST CARRY
@t_0
PLUSR
6340
B_0+N_0
B_0+Y_0
B^*_0-N_0
B0AIMux
NOY0Mux
"constant"
N_0
N0SR
6320
6230
6290
B_0
B^*_0
BAISR
6190
6390
6410
CSA-ACCUMULATOR
X
6207
OPERATIONAL UNIT - SERIAL PROCESSOR
6220
D
F
C0-B0Z
@T_{128X}
00
Z
$Z-N_{df128}$
Y_0
Sd
Bd
Nd

FIG. 7A

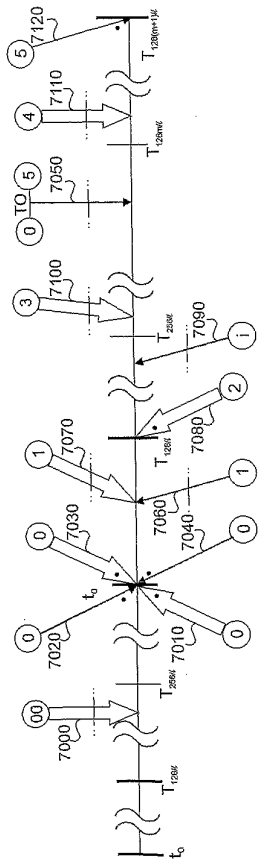

7120
7110
7050
7100
7090
7070
7080
7030
7060
7040
7020
7010
7000
t_0

FIG. 7B

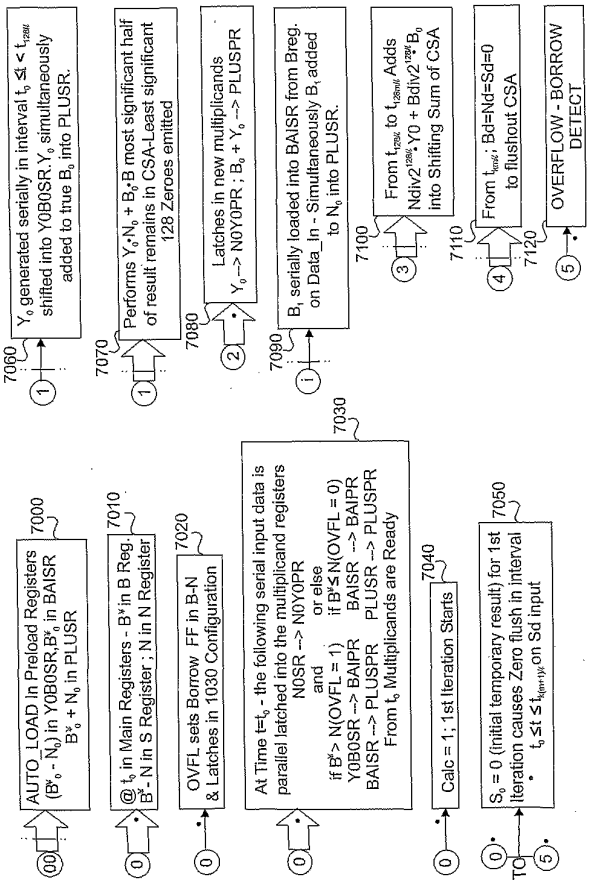

AUTO_LOAD In Preload Registers
(B*o - No) in Y0B0SR,B*o in BAISR
B*o + No in PLUSR
7000
@ to in Main Registers - B* in B Reg.
B* - N in S Register ; N in N Register
7010
OVFL sets Borrow_FF in B-N
& Latches in 1030 Configuration
7020
At Time t=to - the following serial input data is
parallel latched into the multiplicand registers
N0SR --> N0Y0PR
or else
if B*> N(OVFL = 1) Y0B0SR --> BAIPR
if B*≤ N(OVFL = 0) BAISR --> BAIPR
BAISR --> PLUSPR PLUSR --> PLUSPR
From to Multiplicands are Ready
7030
Calc = 1; 1st Iteration Starts
7040
So = 0 (initial temporary result) for 1st
Iteration causes Zero flush in interval
to ≤t ≤t(n+1)p on Sd input
7050
Yo generated serially in interval to ≤t < t128ns
shifted into Y0B0SR. Yo simultaneously
added to true Bo into PLUSR.
7060
Performs Yo*No + Bo*B most significant half
of result remains in CSA-Least significant
128 Zeroes emitted
7070
Latches in new multiplicands
Yo --> N0Y0PR ; Bo + Yo --> PLUSPR
7080
Bi serially loaded into BAISR from Breg.
on Data_In - Simultaneously Bi added
to No into PLUSR.
7090
From t128ns to t128ns Adds
Ndiv2128ns*Y0 + Bdiv2128ns*Bo
into Shifting Sum of CSA
7100
From tconst ; Bd=Nd=Sd=0
to flushout CSA
7110
OVERFLOW - BORROW
DETECT
7120
00
0
0
0
0
0
TC
5
1
1
2
3
4
5

FIG. 7C

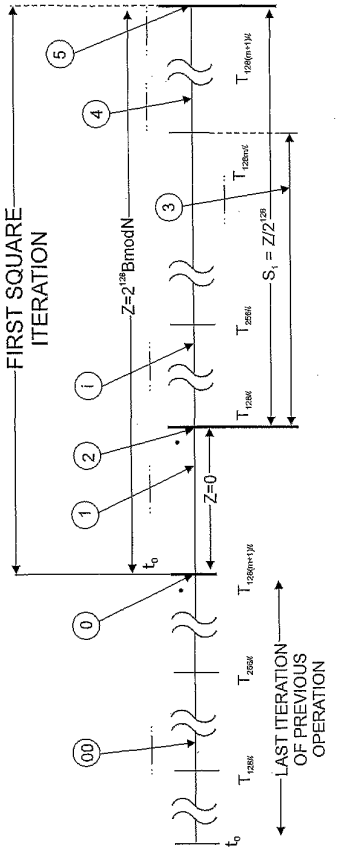

FIRST SQUARE ITERATION
$Z=2^{128}BmodN$
$S_i = Z/2^{128}$
$Z=0$
t_0
$T_{128(m+1)i}$
T_{128mi}
T_{256i}
T_{128i}
LAST ITERATION OF PREVIOUS OPERATION
00
0
1
2
i
3
4
5

FIG. 7D

	N_0	N_0^{-1}	$-(N_0^{-1})$
1	0001	0001	1111
3	0011	1011	0101
5	0101	1101	0011
7	0111	0111	1001
9	1001	1001	0111
11	1011	0011	1101
13	1101	0101	1011
15	1111	1111	0001

$\mathcal{S} = 1$
GF(p)
$\ell = 4$

N_0	N_0^{-1}	$-N_0^{-1}$
01	01	11
11	11	01

$\mathcal{S} = 1$
GF(p)
$\ell = 2$

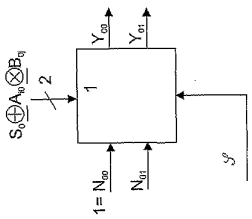
	N_0	$-N_0^{-1} = N_0^{-1}$
1	0001	0001
3	0011	1111
5	0101	0101
7	0111	1011
9	1001	1001
11	1011	0111
13	1101	1101
15	1111	0011

$\mathcal{S} = 0$
No Carry
GF(2^n)
$\ell = 4$

N_0	N_0^{-1}
01	01
11	11

$\mathcal{S} = 0$
No Carry
GF(2^n)
$\ell = 2$

FIG. 8A


Y_{00}
Y_{01}
$S_0 \oplus A_{i0} \otimes B_{j}$
2
1
$1 = N_{00}$
N_{01}
$\mathcal{S}$

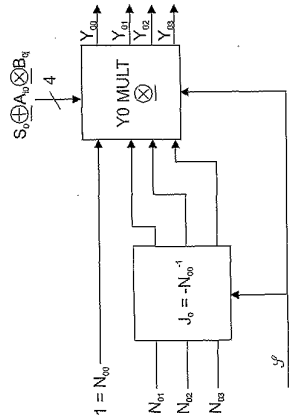

Y_{00}
Y_{01}
Y_{02}
Y_{03}
$S_0 \oplus A_{i0} \otimes B_{j}$
4
Y0 MULT
$J_0 = -N_{00}^{-1}$
$1 = N_{00}$
N_{01}
N_{02}
N_{03}
$\mathcal{S}$

FIG. 8B

【国際公開パンフレット（コレクトバージョン）】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 November 2001 (22.11.2001)

PCT

(10) International Publication Number
WO 01/89129 A3

(51) International Patent Classification[7]: G06F 7/49

(21) International Application Number: PCT/IL01/00425

(22) International Filing Date: 14 May 2001 (14.05.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
136151 15 May 2000 (15.05.2000) IL
139674 14 November 2000 (14.11.2000) IL

(71) Applicant *(for all designated States except US)*: **M-SYSTEMS FLASH DISK PIONEERS LTD.** [IL/IL]; 3B Omer Industrial Park, 84965 Omer (IL).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **DROR, Itai** [IL/IL]; 13 Hartzit Street, 84965 Omer (IL). **GRESSEL, Carmi, David** [IL/IL]; Kibbutz Urim, 85530 Mobile Post Negev (IL). **MOSTOVOY, Michael** [IL/IL]; 3 Michael Hazani Street, 84480 Beer Sheva (IL). **MOLCHANOV, Alexey** [IL/IL]; 12 Jabotinsky Street, 84000 Beer Sheva (IL).

(74) Agents: **COLB, Sanford, T Sanford T. Colb & CO.** et al.; P.O. Box 2273, 76122 Rehovot (IL).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

[Continued on next page]

(54) **Title:** EXTENDING THE RANGE OF COMPUTATIONAL FIELDS OF INTEGERS

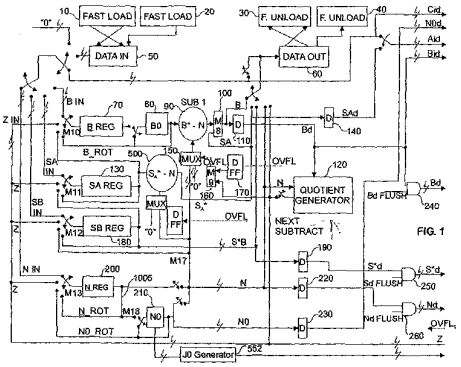


FIG. 1

(57) **Abstract:** An extension of serial (60)/parallel (50) Montgomery multiplication method (Figs. 1-2) with simultaneous reduction as previously implemented by the applicants, adapted innovatively to perform both in the prime number and in the GF(2^n) polynomial based number field, in such a way as to simplify the flow of operands, by performing a multiple anticipatory function (430) to enhance the previous modular multiplication procedure.

WO 01/89129 A3

WO 01/89129 A3

(88) Date of publication of the international search report:
28 March 2002

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IL01/00425

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : G06F 7/49
US CL : 708/492
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 708/252-254, 491-492, 713/200, 380/44

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
WEST; Terms used; "Galios and modular adj multiplication or exponentiation", "finite field and random adj number", "Key adj generation same finite field"

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,073,870 A (Morita) 17 December 1991, column 4, lines 41-68, Fig. 2(A), column 33, lines 15-60.	1-30
Y	US 5,513,133 A (Gressel et al.) 30 April 1996, column 12, lines 36-67, column 13, lines 1-66, column 23, lines 18-61.	1-30
X,P	US 6,185,596 B1 (Hadad et al.) 06 February 2001, column 20, lines 43-67, columns 21-23, lines 1-67.	1-30
Y	US 5,144,574 A (Morita) 01 September 1992, column 10, lines 33-68, column 11, lines 1-52.	1-16

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search
11 December 2001 (11.12.2001)

Date of mailing of the international search report
10 Jan 2002

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231
Facsimile No. (703)305-3230

Authorized officer
Gail O. Hayes
James R. Matthews
Telephone No. (703) 305-4274

Form PCT/ISA/210 (second sheet) (July 1998)

フロントページの続き

(81)指定国 AP(GH,GM,KE,LS,MW,MZ,SD,SL,SZ,TZ,UG,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,CH,CY,DE,DK,ES,FI,FR,GB,GR,IE,IT,LU,MC,NL,PT,SE,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GW,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,DZ,EC,EE,ES,FI,GB,GD,GE,GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KP,KR,KZ,LC,LK,LR,LS,LT,LU,LV,MA,MD,MG,MK,MN,MW,MX,MZ,NO,NZ,PL,PT,RO,RU,SD,SE,SG,SI,SK,SL,TJ,TM,TR,TT,TZ,UA,UG,US,UZ,VN,YU,ZA,ZW

(72)発明者 グレセル、カルミ、デヴット
イスラエル モービル ポスト ネゲブ ８５５３０、 キブトズ ウリム
(72)発明者 モストボイ、ミハエル
イスラエル ベール シェバ ８４４８０、 ミハエル ハザニ ストリート ３
(72)発明者 モルシャノブ、アレクシー
イスラエル ベール シェバ ８４０００、 ジェボチンスキー ストリート １２

Ｆターム(参考) 5J104 AA18 JA25 NA16