RSA 暗号ハードウェアマクロ 仕様書（日本語版）

Version	Update	Description
0.1	2007/10/01	Initial version is created
	2007/10/11	

1. 概要

本 RSA ハードウェアマクロの概要を Table 1 に示す．本マクロは，RSA 暗号[1]のべき乗剰余演算処理を行う．べき乗剰余演算は左バイナリ法に基づいており，また，演算中の乗剰余演算には高基数モンゴメリ乗算アルゴリズムを用いている．本マクロに用いた高基数モンゴメリ乗算アルゴリズムの詳細は, [2]の Coarsely Integrated Operand Scanning(CIOS)を参照されたい．なお，CRT モードには対応していない．

Table 1　RSA ハードウェアマクロ概要

Algorithm	RSA
Data block size	1024 bits
Key size	1024 bits
Description Language	Verilog-HDL
Top module name	RSA
Throughput	1024 bit / about 7 million clocks

2. 入出力信号

RSA 暗号ハードウェアマクロの入出力信号を Table 2 に示す.

Table 2　入出力ポート

Port name	Direction	Width	Description
Kin	In	32	鍵入力. 1,024 ビットの鍵データを最下位ビットから 32 ビット毎，32 サイクルかけてシーケンシャルに入力.
Min	In	32	法入力. 1,024 ビットの法データを最下位ビットから 32 ビット毎，32 サイクルかけてシーケンシャルに入力.
Din	In	32	データ入力．1,024 ビットのデータを最下位ビットから 32 ビット毎，32 サイクルかけてシーケンシャルに入力.
Dout	Out	32	データ出力．Dvld=1 が出力された後，1,024 ビットのデータを最下位ビットから 32 ビット毎，32 サイクルかけてシーケンシャルに出力.
Krdy	In	1	Krdy=1 とした後，内部のレジスタに鍵を取り込む.

			Mrdy と Krdy の両方が 1 のときは，Krdy を優先する．
Mrdy	In		Mrdy=1 とした後，法への入力を内部のメモリに取り込む．
Drdy	In	1	Drdy=1 とした後，データへの入力を内部のメモリに取り込む．その後，続けて暗号化を開始する．
RSTn	In	1	リセット信号．RSTn=0 の時，シーケンサおよび内部レジスタをリセットする．負論理で指定される．
EN	In	1	イネーブル信号．EN=1 の時、マクロがアクティブ状態になる．
CLK	In	1	システムクロック．すべてのレジスタは，CLK の立ち上がりに同期する．
BSY	Out	1	処理中であることを表すフラグ．暗号化，もしくはデータを取り込んでいる時に 1 となる．この間，Drdy, Mrdy, Krdy の変化は無視される．
Kvld	Out	1	内部状態を表すフラグ．鍵が取り込まれた時に 1 クロックだけ立つ．
Mvld	Out	1	内部状態を表すフラグ．法が取り込まれた時に 1 クロックだけ立つ．
Dvld	Out	1	内部状態を表すフラグ．べき乗剰余演算の処理が完了する最後のクロックに，１クロックだけ立つ．

3. ハードウェアアーキテクチャ

3.1 トップモジュール

Fig.1 に RSA 暗号マクロのトップモジュールを示す．本マクロは，主に Sequencer block, Memory，Multiplication block の３つのモジュールで構成される．

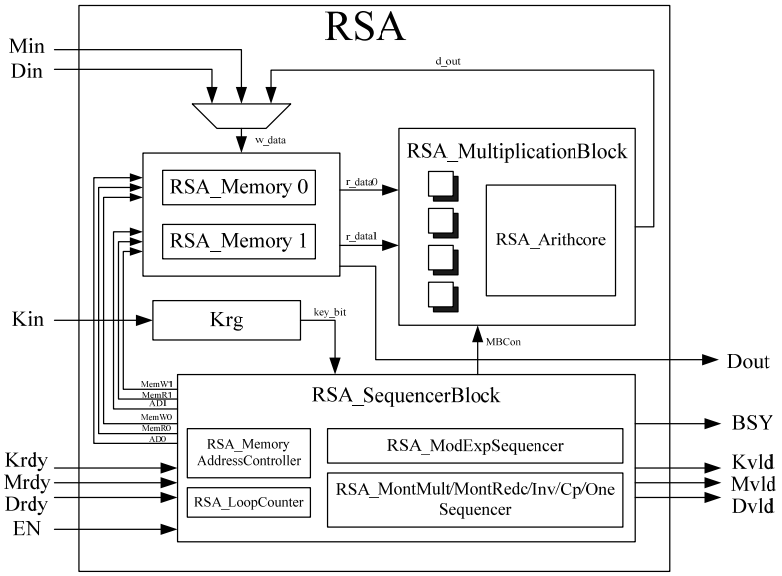

Fig.1　RSA 暗号ハードウェアのトップモジュール

3.2 Multiplication Block

Fig.2 は，左バイナリ法によるべき乗剰余演算アルゴリズム，および ，[2] に示されている CIOS（Coarsely Integrated Operand Scanning）と呼ばれる高基数モンゴメリ乗算アルゴリズムである．本マクロは，CIOS が効率的に実現できるように設計されている．1,024 ビットの多倍長演算をワード長毎（ワード長 r = 32）の演算に分割（分割数 m=32）して，その繰り返しにより実現する．

MODULAR EXPONETIATION (*ModExp*)

Input:	X, N $E = (e_{k-1}, ..., e_1, e_0)_2$,
Output:	$Z = X^E \bmod N$

```
1 :  W := InvN (N);
2 :  Y := MontRedc (X, N);
3 :  Z := X;
4 :  for i = k − 1 downto 0
5 :     Z := MontMult (Z, Z, N, W);   – square
6 :     if (e_i = 1) then
7 :        Z := MontMult (Z, Y, N, W);– multiply
8 :     end if
9 :  end for
10:  Z := MontMult (Z, 1, N, W);
```

MONTGOMERY MULTIPLICATION (*MontMult*)

Input:	$X = (x_{m-1}, ..., x_1, x_0)_{2^r}$, $Y = (y_{m-1}, ..., y_1, y_0)_{2^r}$, $N = (n_{m-1}, ..., n_1, n_0)_{2^r}$, $W = -N^{-1} \bmod 2^r$
Output:	$Z = XY2^{-r \cdot m} \bmod N$

```
1 :  Z := 0;    V := 0;
2 :  for i = 0 to m − 1
3 :     C := 0;
4 :     t_i := (z_0 + x_i y_0) mod 2^r;
5 :     t_i := t_i W mod 2^r;
6 :     for j = 0 to m − 1
7 :        Q := z_j + x_i y_j + C;
8 :        z_j := Q mod 2^r;    C := Q/2^r;
9 :     end for
10:     z_m := C;
11:     C := 0;
12:     for j = 0 to m − 1
13:        Q := z_j + n_j t_i + C;
14:        if (j ≠ 0) then z_{j−1} := Q mod 2^r;
15:        C := Q/2^r;
16:     end for
17:     Q := z_m + V + C;
18:     z_{m−1} := Q mod 2^r;    V := Q/2^r;
19:  end for
20:  C := 1;
21:  for j = 0 to m − 1
22:     Q := z_j+!n_j + C;
23:     z_j := Q mod 2^r;    C := Q/2^r;
24:  end for
25:  if (C == 1 || V == 1) then return
26:  C := 0;
27:  for j = 0 to m − 1
28:     Q := z_j + n_j + C;
29:     z_j := Q mod 2^r;    C := Q/2^r;
30:  end for
```

Fig.2　べき乗剰余演算アルゴリズムおよび高基数モンゴメリ乗算アルゴリズム（CIOS）

Fig. 3 は Multiplication Block のデータパスである．32 ビットの積和演算器 Arithmetic core とレジスタ X, Y, Z, C から主に構成される．Arithmetic core は，レジスタに格納されているワードに対して 3 項積和演算（Q = z + xy + C もしくは Q = z + nt + C）を行う．中間データを格納するメモリは一度に読み出し/書き込みのどちらかだけ行う RAM を想定している．そのため，積和演算の 1 ステップは，メモリからワードをレジスタに読み出すサイクルと演算を行い結果をメモリに書き込むサイクルの 2 サイクルをからなる．

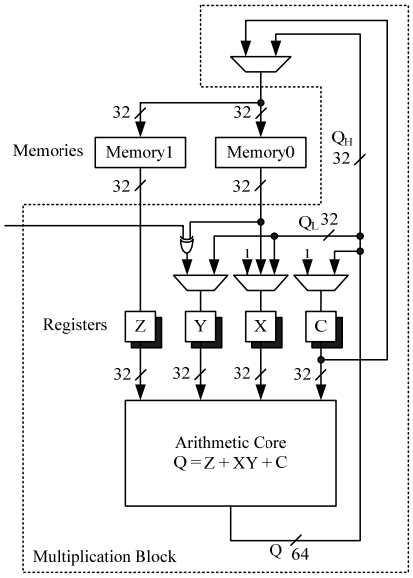

Fig.3　Multiplication block のデータパス

Fig. 4 は CIOS の基本となる各演算におけるアクティブとなるパスを示したものである．図では，演算を行い結果を書き込むサイクルについて示した．

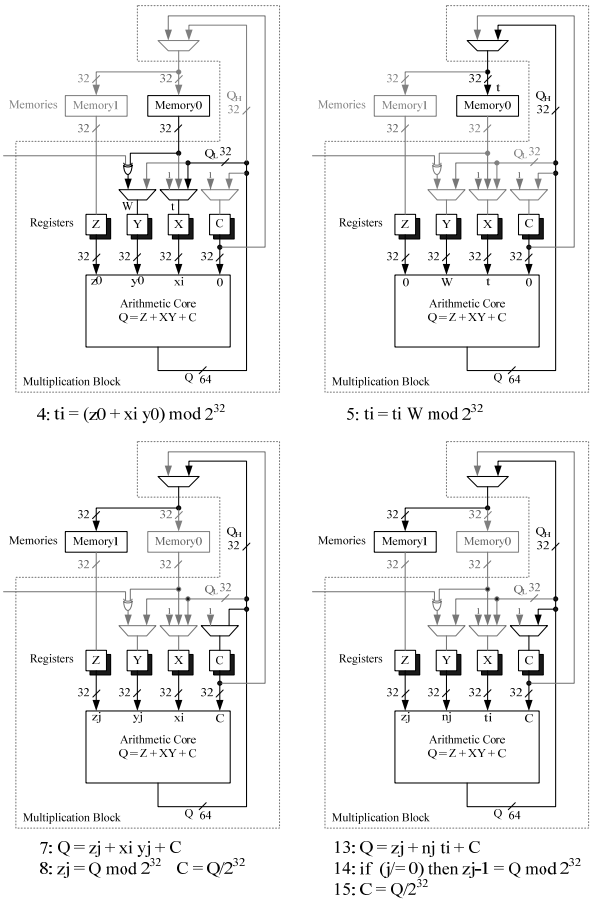

Fig.4　CIOS の各演算におけてアクティブとなるパス

3.5 状態遷移図

Sequencer block は図 2 に示す状態遷移図に従い状態遷移を行う．Fig.5 は，入出力およびべき乗剰余演算（左バイナリ法）のメインシーケンスである．べき乗剰余演算では，各状態において，サブーシーケンスを呼び出す．各サブシーケンスの状態遷移に関しては割愛する．なお，Table 3 に各状態での動作を示す．

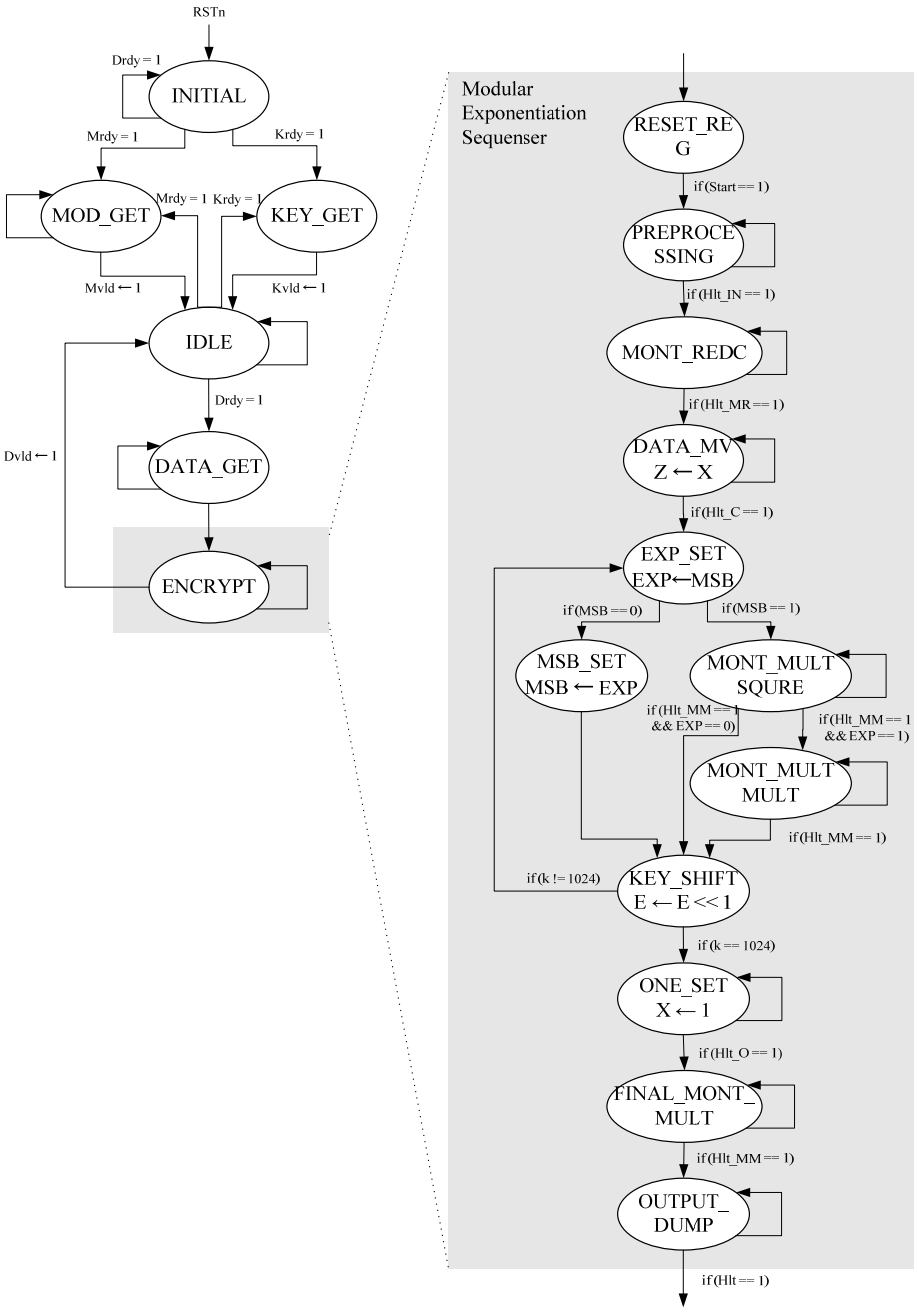

Fig. 5　シーケンサの状態遷移図

Table 3　状態の説明

State	Description
INITIAL	初期状態．鍵の入力・法入力を受け付ける
IDLE	待機状態．暗号化／復号処理を開始できる．また，鍵・法を変更することもできる
KEY_GET	鍵データを鍵レジスタに取り込む．32 サイクル有する．
MOD_GET	法データを内部メモリに取り込む．32 サイクル有する
DATA_GET	平文を内部メモリに取り込む．32 サイクル有する
ENCRYPT	暗号化処理（べき乗剰余演算）を行う．
RESET_REG	レジスタの値をリセットする．
PREPROCESSING	前処理の演算．W = -N mod 2^1024 を計算する． 69 サイクル有する．
MONT_REDC	前処理の演算．Z = X＊2^1024 mod N を計算する．99427 サイクル有する
DATA_MV	メモリ内のデータを移動させる（Z←X）．66 サイクル有する．
EXP_SET	EXP レジスタをセットする．
MSB_SET	MSB レジスタをセットする．
MONT_MULT_SQURE	モンゴメリ乗算を実行する（自乗算）．Z = X * X * 2^1024 mod N を計算する．4578 サイクル有する．
MONT_MULT_MULT	モンゴメリ乗算を実行する（乗算）．Z = Z* X * 2^1024 mod N を計算する．4578 サイクル有する．
KEY_SHIFT	鍵シフト演算を行う．
ONE_SET	メモリに 1 をセットする．34 サイクル有する．
FINAL_MONT_MULT	最後のモンゴメリ乗算処理を行う．4578 サイクル有する．
OUT_PUT_DUMP	メモリから出力レジスタに答えを書き込む．34 サイクル有する．

4. タイミングチャート

Fig. 6 に RSA 暗号ハードウェアマクロのタイミングチャートを示す．このタイミングチャートに従って，ハードウェアマクロの動作を説明する．なお，入力信号は全て最短のタイミングで制御している．

CLK1: RSTn を 0 とすることによりシーケンサおよびレジスタをリセットする．

CLK2~34: Krdy=1 とした後，鍵データ 1,024 ビットを鍵レジスタに格納する．入力ポートが 32 ビットであるため，最下位ビットから 32 ビット毎にシーケンシャルに入力する．この時 BSY=1 となる．32 クロック後，BSY=0 となり，アイドリング状態に移行する．また，**CLK34** に Kvld が 1 クロックの間出力される．

CLK35~47: Mrdy=1 とした後，法データ 1,024 ビットをメモリに格納する．鍵データ同様，最下位ビットから 32 ビット毎にシーケンシャルに入力する．また，BSY=1 となる．32

クロック後，BSY=0 となり，アイドリング状態に移行する．また，**CLK47** に Mvld が 1 クロックの間出力される．

CLK48~80: 鍵および法が格納された状態で Drdy=1 とすると，平文データ 1,024 ビットをメモリに格納する．最下位ビットから 32 ビット毎にシーケンシャルに入力する．また，BSY=1 となる． さらに，取り込み後，暗号化状態に移行する．

CLK81~: およそ 7,000,000 クロックかけてべき乗剰余演算を行う（鍵のビットパターンによって処理時間は変化する）．処理終了後，Dvld=1 を 1 クロックの間出力する．その後，最下位ビットから 32 ビット毎にシーケンシャルに演算結果を出力する．32 クロック後，BSY=0 となり，アイドリング状態に移行する．

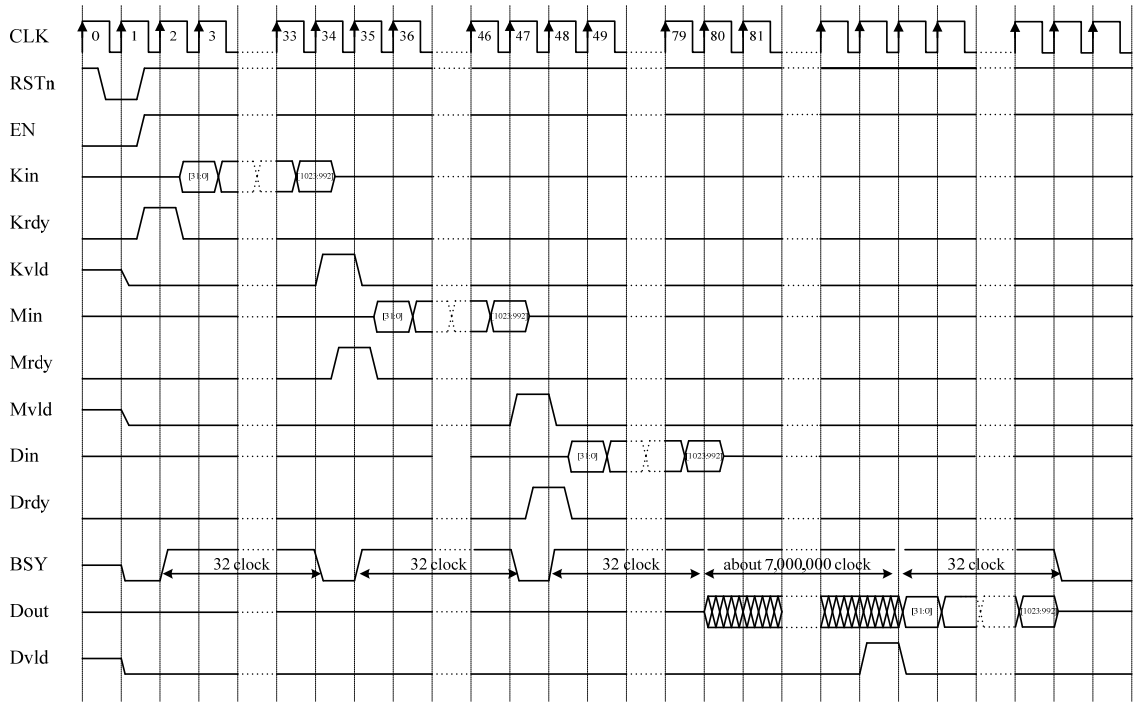

Fig. 7　タイミングチャート

5. Reference

[1] R.L. Rivest, A. Shamir, and L. Adliman, “A method for obtaining digital signatures and public-key cryptosystems,” Communications of the ACM, vol.21, no.2, pp.120--126, 1978.

[2] C.K. Koc, T. Acar, and J. Burton S. Kaliski, “Analyzing and comparing montgomery multiplication algorithms,” IEEE Micro, vol.16, no.3, pp.26--33, 1996.