分割式 Montgomery 模乘运算的线性高基心动阵列新结构

王 冕 周玉洁

(信息工程大学信息工程学院信息研究系 郑州 450002)

摘 要 本文基于提高并行性、加速模乘的思想,利用分割操作数的方法,提出了分割式 Montgomery 模乘算法(PMMM),并且基于 C. D. Walter[3]发明的心动阵列结构,提出了新的线性高基心动阵列模乘结构,较好地实现了 PMMM。对于基 $r(r=2^w)$ 的 n 位模乘运算,Walter[3]使用 $(n+1)(n+2)$ 个 PE 来实现 Montgomery 模乘,我们用 $n+2$ 个 PE 实现 Montgomery 模乘,最大并行性为 Walter[3]的 2 倍。将此结构应用于模幂运算,仅需一次预计算便可使得非平方模乘的输入输出延迟为 Walter[3]中的 $\frac{1}{2}$,且平方模乘延迟与其相当,从而提高了模幂的运算速度。当然,考虑到对速度和硬件资源的不同需求,我们也给出了使用 $\frac{n}{2}+1$ 个 PE 来计算模乘、模幂的实现算法,并做出了相应的数据分析。

关键词 心动阵列,Montgomery 模乘运算,模幂运算

A Novel Systolic Linear Array Architecture for Partitioning Montgomery Modular Multiplication

WANG Mian ZHOU Yu-Jie

(Institute of Information Engineering, Information Engineering University, Zhengzhou 450002)

Abstract A new partitioning Montgomery modular multiplication algorithm(PMMM)is proposed in this paper together with a hardware architecture proper for it to get high simultaneity and performance. And the architecture is a liner high-index systolic array one that is refined from the one proposed by C. D. Walter[3]. While Walter[3] used $(n+1)(n+2)$PEs to accomplish Montgomery modular multiplication, we use $n+2$ PEs, and the simultaneity is two times higher than Walter'sone. Utilizing the new architecture in modular exponentiation operation, we can reduce the latencies of non-square modular multiplication to half of that of[3]in the cost of one precomputation, and the latencies of square modular are the same as in [3], in that the speed is faster. Of course, to keep balance between speed and hardware resource, we also provide methods to realize modular multiplication and exponentiation with $\frac{n}{2}+1$ PEs, for which we have also provided comprehensive analysis.

Keywords Systolic array, Montgomery modular multiplication, Modular exponentiation

1 引言

大数模幂乘运算是很多公钥密码体制(如 RSA[1])的核心运算,它由一系列的模乘运算构成。根据文[2]研究表明,大数的位数至少应超过 664bit,才能保证 RSA 的安全性要求。所以,大数运算所带来的极高的硬件复杂性和低效率性。已成为制约公钥密码体制发展和应用的瓶颈。处理单元(PE)局部互连规则的心动阵列架构特别适合这类运算的 VLSI 实现,在高时钟频率下能获得高的加解密速度。人们提出了几种心动阵列架构模乘运算器[2~10],其中文[2,4,9,10]都是研究 $r=2$ 的快速模乘算法及相应的心动阵列结构,Walter[3]提出的心动阵列结构需用 $(n+1)(n+2)$ 个 PE、总时钟周期为 $3n+2$、输入输出延迟为 $2n+2$;Heo[5]、Iwamura[6]和 Chen[7]提出了用 $n+1$ 个 PE 的心动阵列结构,Lee[8]提出了用 $n+2$ 个 PE、总时钟周期数为 $3n+4$、输入输出延迟为 $2n+3$ 的心动阵列结构。本文针对大数模乘的 Montgomery 算法和 Walter[3]发明的心动阵列结构进行改进。利用分割操作数、提高并行性的思想,提出了分割式 Montgomery 算法和适合于此算法运算的一种新的线性高基心动阵列模乘器结构。用 $n+2$ 个 PE 实现了 Montgomery 模乘运算,最大并行性由原来的 $\frac{n}{2}+1$ 提高到 $n+2$。这种结构应用于 LR 二进制快速模幂运算,可使得非平方模乘的时钟总数为 $2n+2$、输入输出延迟为 $n+1$,平方模乘的时钟总数为 $3n+3$、输入输出延迟为 $2n+2$,从而提高了整体模幂的运算速度。由于该结构使用了操作数高基表示,从而使得在相对较低的时钟频率下也能获得较高的运算速度。

2 Montgomery 算法及心动阵列

2.1 Montgomery 算法

为方便地实现大数模乘运算,P. L. Montgomery 在 1985 年发明了一种不带除法的模乘运算[11],该算法描述如下:

假设奇模数 $N>1$,选择正整数 R,满足 $R>N$,且 $(R,N)=1$。

设 R^{-1} 和 N' 为满足下列条件的两个正整数:

$0<R^{-1}<N, RR^{-1}=1 \pmod N; 0<N'<R, NN'=-1 \pmod R$。

王 冕 硕士研究生,主要研究方向:信息安全;周玉洁 教授,博导,国家 863 专家,主要研究方向:加密芯片 SOC 芯片设计。

另设 A 和 B 是任意两个正整数，则 Montgomery 模乘算法的原始形式见算法 1。

算法 1 $Mont(A,B,N)$

{$T=A\cdot B$;

$M=(T \bmod R)N' \pmod R$;

$S=(T+MN)/R$;

if $S\geqslant N$ then $S=S-N$;

return S。}

在 Montgomery 模乘算法中，最终结果并不是严格意义上的模乘 $AB \pmod N$，而是多了一个因子 R^{-1}，即 $ARB^{-1} \pmod N$。模乘 $AB \pmod N$ 可通过两次 Montgomery 模乘得到，即：$AB \pmod N = Mont(Mont(A,B,N)R^2,N)$。

2.2 心动阵列

算法 1 中的操作数均为大数，难以在硬件上实现。为了采用心动阵列的电路结构，Montgomery 模乘算法可以转化为如下的形式：

以正整数 r 为基表示原算法中的大数，为了方便起见，r 通常取 2^w，w 称为字长。假设所有的操作数均为基 r 的 n 位整数，即 $A=\sum_{i=0}^{n}A[i]r^i$，$B=\sum_{j=0}^{n-1}B[j]r^j$，$N=\sum_{j=0}^{n-1}N[j]r^j$。

算法 2 Walter[3] 对 Montgomery 模乘算法的改进

$P_i:=0$

for $i:=0$ to $n-1$ do

Begin

$Q[i]:=((P[0]+A[i]\times B[0])\times(r-N[0]^{-1}) \bmod r$

$P:=((P+A[i]\times B)\times(r-N[0])^{-1})/r$

End

其中 P 为部分积，P 的每一位可由下式计算：

$P_{i+1}[j-1]+r\times carryout=P_i[j]+A[i]\times B[j]+Q[i]\times N[j]+Carryin$

在硬件实现上，上式可由一个 PE(Processing Element) 单元完成。

根据上述算法，可得心动阵列的结构如图 1 所示。

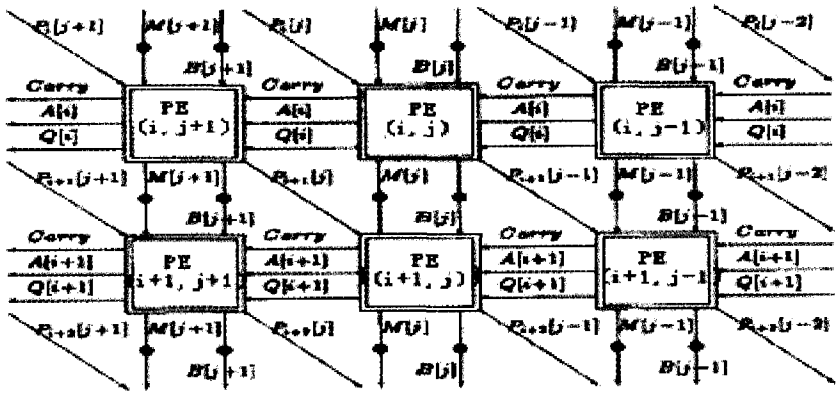

图 1 C. D. Welter 的心动阵列结构

$PE[i,j]$ 在 $2i+j$ 第个时钟周期计算，则在一个时钟周期里最多有 $\frac{n}{2}+1$ 个 PE 单元在并行计算，即这种结构的最大并行性为 $\frac{n}{2}+1$，且总的时钟周期数为 $3n+2$，输入输出延迟为 $2n+2$，需 $(n+1)(n+2)$ 个 PE(详情见参考文[3])。该心动阵列方案虽然结构规则，但是需要的电路面积比较大。根据此情况，我们提出了新的线性心动阵列结构，有效地减少了 PE 个数。

3 新的线性心动阵列结构

首先我们对算法 2 提出了改进。操作数 A 为基 r 的 m 位整数、操作数 B 和 N 为基 r 的 n 位整数，即 $A=\sum_{i=0}^{m-1}A[i]r^i$，$B=\sum_{j=0}^{n-1}B[j]r^j$，$N=\sum_{j=0}^{n-1}N[j]r^j$，并记 $pmmm(A,B,N,m)=ABr^{-m} \bmod N$。

在文[12]中，Eldridge 和 Walter 提出一种通过左移 B 而使得 $B[0]=0$，从而简化计算 $Q[i]$ 的方法。按照其思想，我们提出算法 3，取 $A'=\sum_{i=0}^{m}A'[i]\cdot r^i$，$A'[m]=0$，$A'[i]=A[i]$ $(i=0,1,\cdots,m-1)$，$B'=\sum_{j=0}^{n}B'[j]\cdot r^j$，$B'[0]=0$，$B'[j]=B[j-1]$。

算法 3 $pmmm(A,B,N,m)$

$P_0:=0$

for $i:=0$ to m do

Begin

$Q[i]:=(P_i[0]\times(r-N[0])^{-1}) \bmod r$

$P_{i+1}:=(P_j+A'[i]\times B'+Q[i]\times N)/r$

End

if$(P_{m+1}>N)$do $P=P_{m+1}-N$;

else do $P=P_{m+1}$.

证明：由归纳可得

$r^{i+1}P_{i+1}=\sum_{j=0}^{i}A'[j]r^j\cdot B'+\sum_{j=0}^{i}Q[j]r^j\cdot N$;

当 $i=m$ 时，$r^{m+1}P_{m+1}=\sum_{j=0}^{m}A'[j]r^j\cdot B'+\sum_{j=0}^{m}Q[j]r^j\cdot N$

$r^{m+1}P_{m+1}=A'\cdot B'+Q\cdot N$，$P_{m+1}=A'\cdot B'\cdot r^{-(m+1)} \bmod N$

$A'=A$，$B'=B\cdot r$；$P_{m+1}=A\cdot B\cdot r^{-m} \bmod N$.

注：1)部分积 $P_i<B'+N<r^{n+2}(i=0,\cdots,m)$。

2)P 的每一位可由下式计算：

$$P_{i+1}[j-1]+r\times Carryout=P_i[j]+A'[i]\times B'[j]+Q[i]\times N[j]+Carryin \quad (1)$$

$N[m]=0$，$j=0,1,\cdots,(n+2)$，$P_{i+1}[-1]=P_0[j]=P_i[n+2]=0$。$B'[n+1]$，$B'[n+2]$，$N[n]$，$N[n+1]$，$N[n+2]$ 均为 0。(1)式的计算可由一个 PE 在一个时钟周期里完成。

3)由于 $A'[m]=0$，则 $P_{m+1}<\frac{(B'+N+(r-1)\cdot N}{r}=\frac{B\cdot r+N\cdot r}{r}=B+N$。

当 $m \leqslant \frac{n}{2}$ 时，根据算法 3，仿照图 1 的心动阵列结构，我们画出其数据流通图 2。

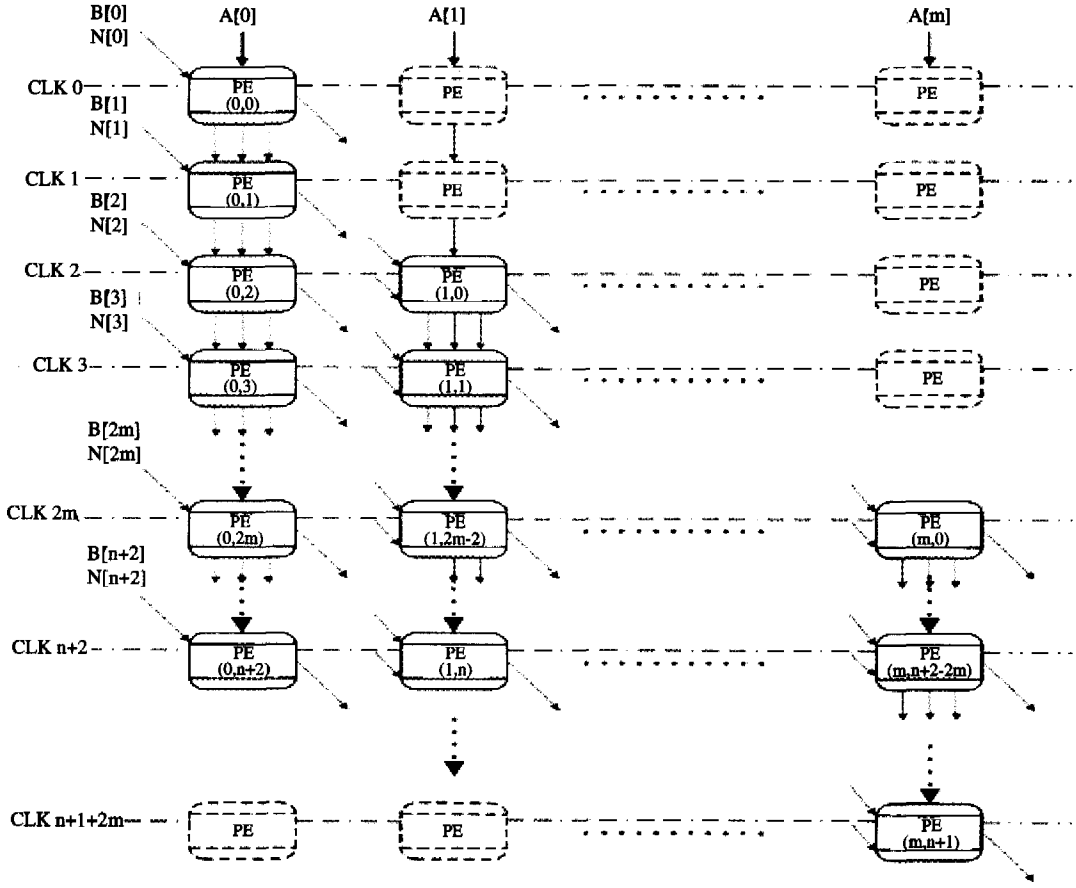

图 2　数据流通图

对于图 2 中的每个 PE，其输入输出数据如图 3 所示。

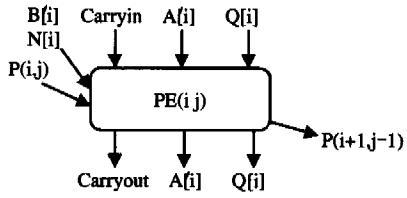

图 3　Carryout $A'[i]Q[i]$

对图 2 数据流通图，分析如下：

最大并行性为 $m+1$；

时钟周期总数为 $2m+n+2$。

具体可分为三个阶段：

阶段一：从第 0 个时钟开始到第 $2m-1$ 个时钟结束。

这段时间内利用 PE 的个数从 1 逐渐增加到 m；

阶段二：从第 $2m$ 个时钟开始到第 $n+2$ 个时钟结束。

这段时间充分利用 $m+1$ 个 PE；

阶段三：从第 $n+3$ 个时钟开始到第 $2m+n+1$ 个时钟结束。

这段时间内利用 PE 的个数从 m 逐渐减少到 1；

根据图 2，我们构造了新的线性心动阵列结构(见图 4)。

从图 2 中我们发现，对于输入的 $B'[j]$，$N[j]$，在偶时钟周期内，j 均为偶数；在奇时钟周期内，j 均为奇数。在硬件实现上，我们利用 $2m+1$ 个寄存器解决了在相邻两个时钟周期里 $B'[j]$，$N[j]$ 不连续的问题。

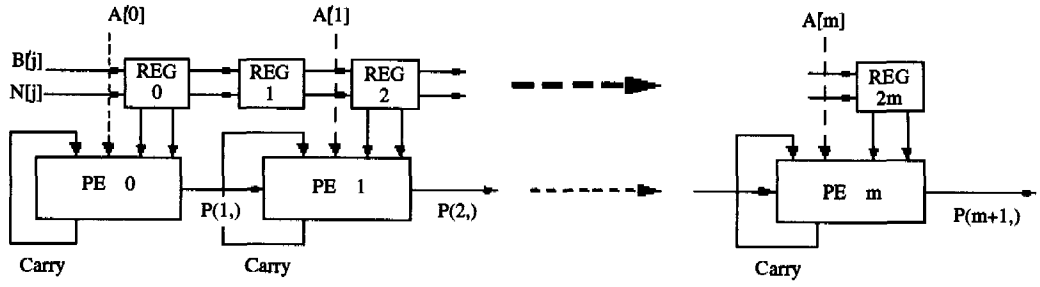

图 4　线性心动阵列结构

在阶段一中，$A'[i]$ 逐渐进入 PEi，并产生 $Q[i]$。在此后的时钟周期里 $A'[i]$，$Q[i]$ 都将存在 PEi 中，不再改变，直到整个模乘运算结束或者有新的 $A'[i]$ 输入。

4　分割式 Montgomery 模乘算法

对于 Montgomery 模乘 $P=ABR^{-1} \bmod N$，其中 $A=\sum_{i=0}^{n-1}$

$A[i]r^i$，$B=\sum_{j=0}^{n-1}B[j]r^j$，$N=\sum_{j=0}^{n-1}N[j]r^j$，$R=r^n$。如果我们把 A 表示为 $A=A_H\cdot r^{\frac{n}{2}}+A_L$，则 $P=(A_H\cdot B\cdot r^{\frac{n}{2}}+A_L\cdot B)\cdot r^{-n} \bmod N=A_H\cdot B\cdot r^{-\frac{n}{2}}+A_L\cdot(B\cdot r^{-\frac{n}{2}})\cdot r^{-\frac{n}{2}} \bmod N$ (*)令 $B^*=B\cdot r^{-\frac{n}{2}} \bmod N$，$P_H=A_H\cdot B\cdot r^{-\frac{n}{2}} \bmod N$，$P_L=A_L\cdot B^*\cdot r^{-\frac{n}{2}} \bmod N$

则 $P=P_H+P_L \bmod N$，(*)式中 A_H，A_L 均为基 r 的 $n/2$ 位整数，B，N 均为基 r 的 n 位整数，因此可用图 3 中的硬件结构(可取 $m=\frac{n}{2}$)来计算 B_1，$P_H(P_L)$：$B_1=pmmm(1,B,N,n/2)$，$P_H=pmmm(A_H,B,N,n/2)$，$P_L=pmmm(A_L,B^*,N,n/2)$。计算 $B_1P_H(P_L)$，需 $n/2+1$ 个 PE，总时钟周期数为 $2n+2$，输入输出延迟为 $n+1$。

因此，计算 $PMMM(A,B,N)=ABr^{-n} \bmod N$，可分为下列四种情况考虑。

1)$PMMM1(A,B,B^*,N)$：B^* 预计算，P_H，P_L 并行计算。

Step1　并行计算 $P_L=pmmm(A_L,B^*,N,n/2)$，$P_H=pmmm(A_H,B,N,n/2)$；

Step2　计算 $P=P_H+P_L \bmod N$。

需 $n+2$ 个 PE，总时钟周期数为 $2n+2$，输入输出延迟为 $n+1$。

2)$PMMM2(A,B,B^*,N)$：B^* 预计算，P_H，P_L 串行计算。

Step1　计算 $P_L=pmmm(A_L,B^*,N,n/2)$；

Step2　计算 $P_H=pmmm(A_H,B,N,n/2)$；

Step3　计算 $P=P_H+P_L \bmod N$。

需 $\frac{n}{2}+1$ 个 PE，总时钟周期数为 $3n+3$，输入输出延迟为 $2n+2$。

3)$PMMM3(A,B,N)$：B^* 与 P_H 并行计算，然后计算 P_L。

Step1　并行计算 $B^*=pmmm(1,B,N,n/2)$，$P_H=pmmm(A_H,B,N,n/2)$；

Step2　计算 $P_L=pmmm(A_L,B^*,N,n/2)$；

Step3　计算 $P=P_H+P_L \bmod N$。

需 $n+2$ 个 PE，总时钟周期数为 $3n+3$，输入输出延迟为 $2n+2$。

4)$PMMM4(A,B,N)$：B^*，P_L，P_H 串行计算。

Step1　计算 $B^*=pmmm(1,B,N,n/2)$；

Step2　计算 $P_L=pmmm(A_L,B^*,N,n/2)$；

Step3　计算 $P_H=pmmm(A_H,B,N,n/2)$；

Step4　计算 $P=P_H+P_L \bmod N$。

需 $\frac{n}{2}+1$ 个 PE，总时钟周期数为 $4n+4$，输入输出延迟为 $3n+3$。

5　模乘单元在 RSA 中的应用

RSA 算法加解密的运算公式如下：$C=M^E \bmod N$，模幂运算可以用多次模乘运算实现。$E=\sum_{i=0}^{k-1}e_i2^i$，$e^i=0$ 或 1，M，N 均为基 r 的 n 位整数。

算法 5　$PMMMExp(M,E,N)$ L－R 分割式 Montgomery 模幂算法

预计算 $R'=r^n \bmod N$，$R''=r^{2n} \bmod N$，$M'=PMMM3(M,R'',N)$，$M^*=pmmm(1,M',N,n/2)$

1)If $e_{k-1}=1$ then $C':=M'$ else $C':=R'$

2)For $i=k-2$ downto 0

2a)$C':=PMMM3(C',C',N)$

2b)if $e_i=1$ then $C':=PMMM1(C',M',M^*,N)$

3)Return $C=PMMM3(C',1,N)$

算法 5 的复杂度为 k 次模平方和平均为 $\frac{k-1}{2}$ 次模乘，最坏的情况需 k 次模平方和 $k-1$ 次模乘。计算一次 $C=M^E \bmod N$，我们分两种方案来讨论：

方案一　使用 $n+2$ 个 PE(算法 5)。

平均所需的时钟数 $(2n+2)\cdot(k+1)+(n+1)\cdot\frac{k-1}{2}$

最坏情况所需时钟数为 $(2n+2)\cdot(k+1)+(n+1)\cdot(k-1)$

方案二　使用 $\frac{n}{2}+1$ 个 PE(算法 5 中将 $PMMM1$、$PMMM3$、分别替换为 $PMMM2$、$PMMM4$)。平均所需的时钟数 $(3n+3)\cdot(k+1)+(2n+2)\cdot\frac{k-1}{2}$；最坏情况所需时钟数为 $(3n+3)\cdot(k+1)+(2n+2)\cdot(k-1)$。

结束语　本文在 Walter[3] 基础上构造了一种新的线性高基心动阵列模乘器结构，较好地实现了分割式 Montgomery 模乘算法。新结构与原结构[3]相比，不仅有效地降低了芯片的面积，而且减少了模幂运算的时钟周期数。根据对速度和硬件资源的不同需求，我们提出了两套方案：第一套方案所需的硬件资源比第二套方案要多一倍，但速度要快得多。因此在实际应用中，可以针对不同的需求采取不同的方案。

参 考 文 献

1　Rivest R, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystem[J]. Communications of the ACM, 1978, 21: 120～126

2　Koc C K, Hung C Y. Bit-level systolic arrays for modular multiplication. J VLSI Signal Processing, 1991, 3: 215～223

3　Walter C D. Systolic modular multiplication[J]. IEEE transactions on computer, 1993, 42(3): 376～378

4　Kornerup P. A systolic, linear-array multiplier for a class of right-shift algorithms[J]. IEEE transactions on computers, 1994, 43(8): 892～898

5　Heo Y J, Lee K J, Yoo K Y. An optimal linear systolic array for computing Montgomery modular multiplication[C]. In: 11th International Conference on Parallel and Distributed Computing Systems(PDCS'98), 1998. 581～585

6　Iwamura K, Matsumoto T, Hidekilmai. Systolic Arrays for Modular Exponentiation using Montgomery Method[C]. In: Proceedings of Euro CRYPT'92, 1992. 477～481

7　Chen Po-Song, Hwang Shih-Arn, Wu Cheng-Wen. A systolic RSA public key cryptosystem[C]. In: IEEE International Symposium On Circuits and Systems, 1996, 4

8　Lee Sung-Woo. Efficient Fixed-Size Systolic Arrays for the Modular Multipication[C]. In: COCOON, 1999

9　Kang Min-Sup, Kurdahi F J. A novel systolic VLSI architecture for fast rsa modular multiplication[C]. In: IEEE Asia-Pacific Conference on ASIC, 2002. 81～84

10　Ors S B, Lejla B, Preneel B. Hardware Implementation of a Montgomery Modular Multiplier in a Systolic Array[C]. In: International Parallel and Distributed Processing Symposium (IPDPS2003), 2003. 184b

11　Montgomery P L. Modular multiplication without trial division [J]. Mathematics of Computation, 1985, 44(170): 519～521

12　Eldridge S E, Walter C D. Hardware Implementation of Montgomery modular Multiplication Algorithm[J]. IEEE transactions on computer, 1993, 42(6): 693～699