[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
H04L 9/28 (2006.01)
G06F 7/72 (2006.01)

[12] 发明专利申请公开说明书

[21] 申请号 200510029516.X

[43] 公开日 2006 年 2 月 22 日

[11] 公开号 CN 1738238A

[22] 申请日 2005.9.8
[21] 申请号 200510029516.X
[71] 申请人 上海微科集成电路有限公司
地址 200433 上海市国定路 335 号 5005 室
共同申请人 复旦大学
[72] 发明人 曾晓洋 范益波 于 宇 章倩苓 郭亚炜

[74] 专利代理机构 上海正旦专利代理有限公司
代理人 陆 飞 盛志范

权利要求书 4 页 说明书 12 页 附图 3 页

[54] 发明名称

高速可配置 RSA 加密算法及协处理器

[57] 摘要

本发明属信息安全技术领域，具体涉及一种能在加密速度和支持密钥长度两个方面都能进行配置的 RSA 加密算法及协处理器。 RSA 加密算法包括改进的流水线模幂算法和流水线模乘算法。 协处理器的流水线数据通路采用改进的模幂和模乘流水线双重结构。 它通过配置模幂流水线和模乘流水线的级数来配置加密系统的加密速度；通过配置模乘流水线级数和模乘运算单元先入先出寄存器大小来配置支持不同的密钥长度，从而可获得支持各种密钥长度和加密速度的 RSA 密码协处理器。

1、一种 RSA 加密算法，包括流水线模幂算法和流水线模乘算法，分别记为算法 3 和算法 4，其特征在于具体如下：

算法 3：

输入: P、N、e、M、r ($r=2^N$)

输出: $C=P^e \bmod M$

1. Set $C_{-1}=1, S_{-1}=P$

========pipeline 1========

2. $S_{-1} = MM(P, r^2) = Pr \bmod M$

3. For i=0 to T-2

3.a $S_i = MM(S_{i-1}, S_{i-1})$

3.b If $e_i=1$ then $C_i = MM(S_{i-1}, C_{i-1})$

End For

======== pipeline 1========

4. For j=1 to (N+1/)T-1

========pipeline j========

5. For k=0 to T-1

5.a $S_{T*j+k-1} = MM(S_{T*j+k-2}, S_{T*j+k-2})$

5.b If $e_{T*j+k-1}=1$ then $C_{T*j+k-1} = MM(S_{T*j+k-2}, C_{T*j+k-2})$

End For

========pipeline j========

End For

6. Return C_{N-1}

该算法第 1 步对(C, S)赋初值；第 2－3 步是流水线的第一级，做 T 次 MM 运算，其中的第 2 步是域转换，第 3 步是 Montgomery 模乘；算法第 4－5 步是一个(N+1/)T-1 级的流水线，其中每级流水线分别计算 T 次的 Montgomery 模乘；最后在算法第 6 步返回模幂结果；

算法 4 ：流水线模乘算法(记为 MM)：

输入：N、X、Y、M、r ($r=2^N$)

输出：$S = \mathcal{MM}(X \times Y) = XYr^{-1} \bmod M$

1. S=0、$x_{-1}=0$

2. For j=0 to N-1 Step k

3. $q_{Yj} = Booth(x_{j+k-1..j-1})$

4. $(C_a,S^{(0)}) = S^{(0)} + (q_{Yj} \times Y)^{(0)}$

5. $q_{Mj} := S^{(0)}_{k-1..0} \times (2^k - M^{(0)-1}_{k-1..0}) \bmod 2^k$

6. $(C_b,S^{(0)}) = S^{(0)} + (q_{Mj} \times M)^{(0)}$

7. For i = 1 to NW – 1

8. $(C_a,S^{(i)}) := C_a + S^{(i)} + (q_{Yj} \times Y)^{(i)}$

9. $(C_b,S^{(i)}) := C_b + S^{(i)} + (q_{Mj} \times M)^{(i)}$

10. $S^{(i-1)} = (S^{(i)}_{k-1..0}, S^{(i-1)}_{BPW-1..k})$

End For

11. $C_a = C_a$ or C_b

12. $S^{(NW-1)} = \text{sign ext}\ (C_a, S^{(NW-1)}_{BPW-1..k})$

End For

13. If S>=M Then S = S – M

该算法采用了按字读取操作数的方法，每个操作数分成为若干个字，每次读入其中的一个；该算法第一步对（S，x_{-1}）赋初值；第 2 步是流水线运算的外层循环，针对每一个输入 x，分别采用流水线按字运算方法来计算乘加结果；第 3－6 步是每级流水线运算的初始值计算，分别计算出 q_{Yj} 和 q_{Mj} 以供后继输入数据运算，同时计算出输入操作数第一个字（$(C_a,S^{(0)})$、$Y^{(0)}$、$M^{(0)}$）的运算结果$(C_b,S^{(0)})$；第 7－12 步分别计算出余下的操作数字的运算结果$(C_b,S^{(i)})$；最后在第 13 步输出最终 Montgomery 模乘结果；

其中，参数含义如下：P 为明文，N 为密钥长度，E 为密钥，M 为模，C 为模幂结果，S 为模乘结果。

2、一种基于如权利要求 1 所述 RSA 加密算法的高速可配置 RSA 加密协处理器，其特征在于其流水线数据通路采用模幂和模乘模块流水线双重结构：其一方面，通过配置模幂流水线和模乘流水线的级数来配置加密系统的加密速度；另一方面，通过配置模乘流水线级数和模乘运算单元先进出寄存器 FIFO 大小来配置支持不同的密钥长度。

3、根据权利要求 2 所述的协处理器，其特征在于所述模幂和模乘模块流水线双重结构中，模幂流水结构由 N+1 个基本运算单元 ME(4)依次连接组成，一个基本运算单元 ME（4）表示流水线运算的一级，其输入信号为：Sq：对应算法 3 中的 S，C：对应算法 3 中的 C，M：对应算法 3 中的 M，e：对应算法 3 中的 e，控制信号：Control；其中，基本运算单元 ME(4)包括：

两个模乘运算单元 MM（6、7），为并行工作模式，它们分别执行算法 3 中的平方运算和乘法运算；

一个寄存器堆(8)，用于存放输入数据 Sq、C、M、e 和控制信号 Control；

一个数据选择器(10-18)；

一个控制逻辑状态机(9)，是整个 ME(4)单元的控制逻辑，它通过数据选择器(10-18)控制数据流向。

4、根据权利要求 3 所述的协处理器，其特征在于每个模乘运算单元 MM（6、7）包括一个模乘流水线数据通路(19)和一个先入先出寄存器 FIFO(30)；其中，模乘流水线数据通路(19)包括：

一组流水线的基本执行单元 MM Cell，共 N+1 个；

一组控制状态机 FSM,共 N+1 个，对应控制 N+1 个基本执行单元 MM Cell；

一组寄存器 REHTEERS 共 N 个，用于寄存执行单元 MM Cell 的运算数据；

2 组数据选择器（28、29），分别用于调度流水线的输入和输出数据；

先入先出寄存器 FIFO（30）包括：

一个由加法逻辑单元(32)和寄存器(31)组成的加法器，寄存器(31)用来暂存进位信息；

一个存贮模块 FIFO(33)；

一个数据选择器(34)，用来旁路 FIFO(33)。

5、根据权利要求 4 所述的协处理器，其特征在于所说的基本执行单元 MM Cell 包括：

2 个寄存器（36、37），将 MM Cell 内部分为两个层次，使得关键路经延迟更低；

2 个 4－to－2 加法器（38、39），采用 carry-save 形式以降低加法器延迟；

2 个译码器（40、41），分别对应算法 4 中的 q_Y 和 q_M；

一组数据选择器（42、43、44、45），根据译码器(41、42)的输出的 qy 和 qm，选择相应的数据进入到 4－to－2 加法器（38、39）。

6、根据权利要求 5 所述的协处理器，其特征在于：

模幂的执行时间 T_{ME} 的计算式为：

$$T_{ME}=\begin{cases}\left(\left\lceil\frac{N}{NS*k}\right\rceil*(2NS+1)+\frac{N}{BPW}+1\right)*T & IF\ \frac{N}{BPW}<=2NS\\ \left(\left\lceil\frac{N}{NS*k}\right\rceil*\left(\frac{N}{BPW}+1\right)+2NS\right)*T & IF\ \frac{N}{BPW}>2NS\end{cases} \quad (4)$$

支持的密钥长度 L 的计算式为：

$$L=2\times NS\times BPW+L_{FIFO} \quad (7)$$

其中 N 为 RSA 加密密钥位数，k 为基的位数，NS 为流水线级数，BPW 为算法 4 中 Y、M、S 的字长，L_{FIFO} 为 FIFO（33）的长度；

于是，配置加密速度：根据计算式(4)，通过调节参数 T、k、NS、BPW 和 L_{FIFO} 来实

现；配置支持密钥长度：根据计算式(7)，通过调节参数 NS、BPW 和 L_{FIFO} 来实现，从而设计出支持各种密钥长度和加密速度的 RSA 密码协处理器。

高速可配置RSA加密算法及协处理器

技术领域

本发明属信息安全技术领域，具体涉及一种能在加密速度和支持密钥长度两个方面都可进行配置的RSA加密算法及其协处理器。

背景技术

随着全球经济信息化的飞速发展，特别是伴随着电子商务、电子政务、金融电子化进程的不断深入，信息的安全问题日益突出，信息安全作为“海、陆、空”之外的“第四国防”，已经被各国政府提高到国家战略发展的层次上，同时也已经得到社会各个方面的广泛关注。信息安全主要是利用密码技术来保证信息的保密性、完整性、可用性和抗抵赖性。密码技术，特别是公钥密码技术RSA算法的芯片级实现，代表着一个国家在信息安全领域的水平。为此，各个国家都投入了大量的人力、物力进行这方面的研究。公钥制密码学中，应用最为广泛的是RSA公钥密码算法，为了保证安全等级，一般采用512位或1024位的密钥长度，在银行等需要非常高的安全等级系统中，会采用更高位数的RSA密钥长度。

目前，世界上RSA密码算法芯片1024位RSA运算速度多数在每秒几千次。国内单位的1024位RSA运算速度更低。

造成RSA运算速度低的原因在于RSA加密运算的复杂性。RSA密码算法的安全性是基于数论中的著名数学难题：将两个大的素数合成一个大数很容易，而相反过程则非常困难。其算法基本的思想如下：

如果有某个明文分组P和密文分组C，加密和解密的过程如下：

$$\text{加密：} \quad C=P^e \bmod M$$

$$\text{解密：} \quad P=C^d \bmod M$$

其中P是明文，C是密文，e为加密密钥，d为解密密钥，M是模（它一个大整数且等于两个素数的乘积）。发送方和接受方都必须知道M的值，发送方知道e的值，接受方知道d的值，所以公钥是E_{key}＝{e、M}，私钥是D_{key}={d、M}。

可见，RSA的基本运算就是模幂运算，它的操作数长度一般在256到2048比特或者更长，一个底数和指数长度都达到几百上千位的模幂运算可想而知是多么费时。对于RSA密码算法来说，密钥长度越长，安全等级越高，同时计算量也越大，速度就越慢。用软件来做1024位的RSA加密，每秒大概只能完成4－8次模幂运算，对于个人用户加来说，这个速度还能接收；但是对于如银行系统、CA中心等，这样的速度就太慢了，人们需要高

速的RSA密码芯片来满足这种需求。所以，硬件实现高速的RSA加密算法将成为未来密码产品的主流，也是当前密码学的研究热点。

RSA 模幂运算是通过一系列的模乘运算得到的。模幂算法根据幂指数扫描顺序不同可以分为两种：从左到右的L-R算法和从右到左的R-L算法。两种算法大同小异，但是从速度和面积角度上来看，L-R 算法体现的是串行运算，运算速度较慢且运算速度不仅取决于密钥的长度，还同密钥的数字特征息息相关；与之相对应的R-L算法则是牺牲面积换取速度，体现了并行运算的思想，速度较快且速度仅仅取决于密钥长度，面积是L-R算法的一倍。R-L算法如下所示：(N：密钥长度；P：明文；C：密文；e：密钥；M：模)

算法 1：*R-L（Right-to-Left）*模幂运算

输入：P、N、e、M

输出：$C=P^e \bmod M$

1 Set $C_{-1}=1, S_{-1}=P$

2 For i=0 to N-1

==========parallel==========

2.a $S_i = S_{i-1} \times S_{i-1} \bmod M$

2.b If $e_i=1$ then $C_i=S_{i-1} \times C_{i-1} \bmod M$

==========parallel==========

3 Return C_{N-1}

首先，算法第1步给C和S赋初值，然后从右到左（也就是从最低位到最高位）扫描密钥e，如果e_i的值是1则执行2.a和2.b，如果e_i的值是0则只执行2.a，2.a和2.b是并行执行的。通过N次的循环，最后返回模幂结果C。

模幂运算中涉及到了大量的模乘运算，模乘运算需要做一次乘法和一次除法，最后取余数，运算量非常大。蒙哥马利于 1985 年提出了著名的蒙哥马利模乘算法，它将复杂的模乘运算简化为做简单的加法和移位运算，从而大大提高了模乘的运算速度，其算法如下：(N：密钥长度；X：乘数；Y：被乘数；M：模；r：常数)

算法 *2 $\mathcal{MM}$*（Montgomery Multiplication）运算：

输入：N、X、Y、M、r ($r=2^N$)

输出：$S = \mathcal{MM}(X, Y) = XYr^{-1} \bmod M$

1. $S_{-1}=0$
2. For i=0 to N-1
3. $S_i=S_{i-1} + x_i \times Y$
4. $S_i=S_i + s_0 \times M$

5. $S_i=S_i/2$

6. End For

7. If $S_{N-1}>=M$ Then $S_{N-1}=S_{N-1}-M$

Return S_{N-1}

算法第 1 步给 S 赋初值 0，第 3 步根据乘数 X 的 i 位 x_i 进行乘加，第 4 步根据 S 的最低位 s_0 判断是否加 M，第 5 步右移一位，如此循环 N 次，第 7 步判断 S 是否大于 M，第 8 步输出模乘结果 S。

发明内容

本发明的目的是提出一种高速可配置 RSA 加密算法及协处理器，使得在 RSA 的加密速度和支持的密钥长度两个方面都能进行配置。

本发明提出的 RSA 加密算法包括流水线模幂算法和流水线模乘算法。分别是对上文中的算法 1 和算法 2 进行改进而提出的流水线算法。具体算法如下：

算法 3：流水线模幂算法（记为 ME）

输入: P、N、e、M、r ($r=2^N$)

输出: $C=P^e \bmod M$

1. Set $C_{-1}=1, S_{-1}=P$

========pipeline 1========

2. $S_{-1} = MM(P, r^2) = Pr \bmod M$

3. For i=0 to T-2

   3.a $S_i = MM(S_{i-1}, S_{i-1})$

   3.b If $e_i=1$ then $C_i= MM(S_{i-1}, C_{i-1})$

   End For

======== pipeline 1========

4. For j=1 to (N+1/)T-1

========pipeline j=========

5. For k=0 to T-1

   5.a $S_{T*j+k-1} = MM(S_{T*j+k-2}, S_{T*j+k-2})$

   5.b If $e_{T*j+k-1}=1$ then $C_{T*j+k-1}= MM(S_{T*j+k-2}, C_{T*j+k-2})$

   End For

========pipeline j=========

   End For

6. Return C_{N-1}

该算法第 1 步对(C, S)赋初值；第 2－3 步是流水线的第一级，做 T 次 MM 运算，其中

的第 2 步是域转换，第 3 步是 Montgomery 模乘；算法第 4－5 步是一个(N+1/)T-1 级的流水线，其中每级流水线分别计算 T 次的 Montgomery 模乘；最后在算法第 6 步返回模幂结果。

相比算法 1，算法 3 增加了一个域转化步骤(2)，它将整数域形式转化成为 Montgomery 域形式。算法 3 将整个模幂运算过程分割成为（N+1）/T 个步骤，每步执行 T 次 MM 算法。相比没有流水线的算法 1，算法 3 能比算法 1 快（N+1）/T 倍。其中，参数的含义同前。

算法 4 ：流水线模乘算法(记为 MM)：

输入：N、X、Y、M、r ($r=2^N$)

输出：$S = \mathcal{MM}(X \times Y) = XYr^{-1} \bmod M$

1. S=0、$x_{-1}=0$
2. For j=0 to N-1 Step k
3. $q_{Yj} = \text{Booth}(x_{j+k-1..j-1})$
4. $(C_a, S^{(0)}) = S^{(0)} + (q_{Yj} \times Y)^{(0)}$
5. $q_{Mj} := S^{(0)}_{k-1..0} \times (2^k - M^{(0)-1}_{k-1..0}) \bmod 2^k$
6. $(C_b, S^{(0)}) = S^{(0)} + (q_{Mj} \times M)^{(0)}$
7. For i = 1 to NW – 1
8. $(C_a, S^{(i)}) := C_a + S^{(i)} + (q_{Yj} \times Y)^{(i)}$
9. $(C_b, S^{(i)}) := C_b + S^{(i)} + (q_{Mj} \times M)^{(i)}$
10. $S^{(i-1)} = (S^{(i)}_{k-1..0}, S^{(i-1)}_{BPW-1..k})$

    End For
11. $C_a = C_a$ or C_b
12. $S^{(NW-1)} = \text{sign ext}\ (C_a, S^{(NW-1)}_{BPW-1..k})$

    End For
13. If S>=M Then S = S – M

该算法采用了按字读取操作数的方法，每个操作数分成为若干个字，每次读入其中的一个；该算法第一步对（S，x_{-1}）赋初值；第 2 步是流水线运算的外层循环，针对每一个输入 x，分别采用流水线按字运算方法来计算乘加结果；第 3－6 步是每级流水线运算的初始值计算，分别计算出 q_{Yj} 和 q_{Mj} 以供后继输入数据运算，同时计算出输入操作数第一个字（$(C_a,S^{(0)})$、$Y^{(0)}$、$M^{(0)}$）的运算结果$(C_b,S^{(0)})$；第 7－12 步分别计算出余下的操作数字的运算结果$(C_b,S^{(i)})$；最后在第 13 步输出最终 Montgomery 模乘结果。其中，参数含义同前。

在算法 4 中，x_i 表示输入操作数 X 的第 i 位，$S^{(i)}$、$Y^{(i)}$、$M^{(i)}$表示操作数 S、Y、M 的第 i 个字，$(C, S^{(i)})$表示 carry-save 格式的第 i 个字。比如，$S^{(0)}_{k-1..0}$ 表示 S 的第 0 个字的第 k-1 到第 0 位。NW（Number of Word）是操作数的字数, BPW（Bit Per Word）是每个操作

数字所包含的位数。

相比算法 2，该算法按字读取操作数，每个操作数分成为若干个字，每次读入其中的一个。算法 2 中第 3、4 步的加法都是 N 位长加法，延迟时间相当大限制了系统的速度；在算法 4 中改进成一个字长的加法（第 8、9 步），并且采用了 Carry Save 的结构，使得加法的路径延迟大大降低。

本发明在上述流水线算法的基础上，设计了 RSA 加密协处理器，其流水线数据通路采用上述模幂和模乘模块流水线双重结构，从而在运算的速度和支持密钥的长度两个方面实现可配置。即一方面，可以通过配置模幂流水线和模乘流水线的级数来配置加密系统的加密速度；另一方面，可以通过配置模乘流水线级数和模乘运算单元先进出寄存器 FIFO 大小来配置支持不同的密钥长度。模幂流水线如图 1 所示，模乘流水线如图 3 所示。下面分别详细阐述模幂流水线和模乘流水线的结构和原理。

模幂流水线结构如图 1 所示，根据算法 3，采用（N+1）/T 级流水线，每级流水线只需要做 T 次 $\mathcal{MM}$ 模乘，两次模幂运算之间的时间间隔仅仅为一级流水线的延迟。采用 n 级流水线结构的模幂运算速度是不采用流水线结构的模幂运算速度的 n 倍。即模幂流水结构由 N+1 个基本运算单元 ME(4)依次连接组成，一个流水线基本运算单元 ME（4）代表流水线运算的一级，其输入信号为：Sq（对应算法 3 中的 S，但 Sq 包含了两份 S 的拷贝，所以实际上 Sq 是两个信号），C（对应算法 3 中的 C），M（对应算法 3 中的 M），e（对应算法 3 中的 e），Control（控制信号）。这些参数的下标 i(i=0,1…N)表示该参数第 i 级运算的输入值。ME 单元结构如图 2 所示，它包括：

两个模乘运算单元 MM（6、7），运算单元 MM（6、7）为并行工作模式，它们分别执行算法 3 中的平方运算（3.a、5.a）和乘法运算（3.b、5.b）。

一个寄存堆(8)，用于存放输入数据 Sq、C、M、e 和控制信号 Control；

一组数据选择器(10-18)；

一个控制逻辑状态机 9，是整个 ME 单元的控制逻辑，它通过数据选择器（10-18）控制数据流向。

具体的数据调度流程如下所示：

1、系统接收输入数据信号 Sq、C、M、e 以及控制信号 Control；

2、根据输入控制信号，在控制逻辑状态机 9 中产生 copy0 和 copy1 信号，将输入数据读入到寄存器堆中；

3、控制逻辑状态机 9 每次读入 E 的最低位，然后根据输入的值判断下一步的运算（算法 3 中的 3.a、3.b）。图 2 中的运算模块 $\mathcal{MM}0$ 和 $\mathcal{MM}1$ 分别对应于算法 3 中的 3.a 和 3.b；

4、当 *MM* 模乘运算结束时，通过 state0、state1 信号进行选择，将正确的结果输出到寄存器堆中（这里，又需要 copy0 和 copy1 信号将结果数据导入寄存器堆中）。

表 1 是一个模幂运算单元在 RSA 加密过程中对数据通路进行调度的一个例子。采用 3 级流水线，每级流水线执行 3 次模乘，密钥 e 为 10110110。第一步 init 步骤相当于算法 3 中的 2，他只出现在流水线的第一级。

表 1

	Before *MM*			*MM* Compute		After *MM*	
E	Sq		C	MM 0	MM 1	Sq	C
Stage 1							
init	X	r^2	1	√	-	Xr	1
0	Xr		1	√	-	X^2r	1
1	X^2r		1	√	√	X^4r	X^2
Stage 2							
1	X^4r		X^2	√	√	X^8r	X^6
0	X^8r		X^6	√	-	$X^{16}r$	X^6
1	$X^{16}r$		X^6	√	√	$X^{32}r$	X^{22}
Stage 3							
1	$X^{32}r$		X^{22}	√	√	$X^{64}r$	X^{54}
0	$X^{64}r$		X^{54}	√	-	$X^{128}r$	X^{54}
1	$X^{128}r$		X^{54}	√	√	$X^{256}r$	X^{182}

本发明中，每个模乘运算单元 MM（6、7）包含两个部分，其一是模乘流水线数据通路 19；其二是 FIFO（先入先出寄存器）30。

模乘流水线数据通路 19 的结构如图 3 所示。根据算法 4，k 表示基，每次循环读入乘数 X 的 k 位，通过 Booth 编码得到 q_Y，被乘数 Y 和模 M 按字读取（$Y^{(i)}$表示 Y 的第 i 个字，如算法 2），故该算法非常容易组织成流水线的结构。如图 3 所示，模乘流水线结构，包括：

一组流水线的基本执行单元 MM Cell，共 N+1 个；

一组控制状态机 FSM，共 N+1 个；对应控制 N+1 个基本执行单元 MM Cell；

一组寄存器 REHTEERS，共 N 个，用于寄存执行单元 MM Cell 的运算数据；

2 组数据选择器（28、29），用于调度流水线的输入和输出数据。因为流出流水线的数据需要重新进入流水线再次运算。如果当输入流水线的数据还未输入完毕，此时的需要再次输入流水线的流水线输出数据不能直接进入流水线，而需要暂时进入 FIFO 中等待上一轮的数据输入完毕。

对于流水线数据通路（19），流水线输入数据为 Y（被乘数）、M（模）、SS、SC（S 的 Carry Save 形式）、Xj（乘数）、Control（控制信号）。为了提高运算速度，在 MM Cell 的加法器设计中采用了 CSA（Carry Save Adder）的结构，在 MM 运算的最后阶段需要将 SS、SC 相加得到 S。流水线的每一级完成一次算法中的 FOR 循环运算，并将结果传递给下一级。流水线输入的控制信号被流水线第 1 级接收，并逐级传递给下一级。Xj 是乘数 X 的字，它被输入到每一级的 MM Cell 中，通过各级的 FSM（有限状态机）的控制信号决定是否读取 Xj。OS、OC 是该流水线输出的结果；注意，这里并非是 MM 运算的最终结果。

通过算法 4 可以知道，数据每过两个时钟周期向前传递一级。若流水线有 NS 级，则需要 2NS 个周期数据才能流出流水线。所以对于有 NS 级的流水线来说，数据经过一次流水线后的结果仅仅是扫描了 NS×k（k 是改进算法中的基）位的 X，如果 X 的位数超过 NS×k，则需要将数据再次注入流水线进行运算。同时，我们发现对于被乘数 Y 和模 M 来说，流水线中最多能容纳的 Y、M 的字数为 NS×2 个字，即如果当 $Y^{(0)}$、$M^{(0)}$ 流到了最后一级（NS 级）时，第 1 级的 Y、M 的字为 $Y^{(NS)}$、$M^{(NS)}$。如果 Y、M 的字数超过了 NS×2，同时 X 的字数也超过了 NS×k，则当数据 OS、OC 需要再次注入流水线时会出现等待。下面举例解释这一过程：

若 k=4，NS=3，NW_x=6，NW_{ym}=7

表 2

CLK	Stage 1	Stage 2	Stage 3
1	$X^{(0)}$、$Y^{(0)}$、$M^{(0)}$、0		
2	$X^{(0)}$、$Y^{(1)}$、$M^{(1)}$、0		
3	$X^{(0)}$、$Y^{(2)}$、$M^{(2)}$、0	$X^{(1)}$、$Y^{(0)}$、$M^{(0)}$、$S^{(0)}_1$	
4	$X^{(0)}$、$Y^{(3)}$、$M^{(3)}$、0	$X^{(1)}$、$Y^{(1)}$、$M^{(1)}$、$S^{(1)}_1$	
5	$X^{(0)}$、$Y^{(4)}$、$M^{(4)}$、0	$X^{(1)}$、$Y^{(2)}$、$M^{(2)}$、$S^{(2)}_1$	$X^{(2)}$、$Y^{(0)}$、$M^{(0)}$、$S^{(0)}_2$

6	$X^{(0)}$、$Y^{(5)}$、$M^{(5)}$、0	$X^{(1)}$、$Y^{(3)}$、$M^{(3)}$、$S^{(3)}_1$	$X^{(2)}$、$Y^{(1)}$、$M^{(1)}$、$S^{(1)}_2$
7	$X^{(0)}$、$Y^{(6)}$、$M^{(6)}$、0	$X^{(1)}$、$Y^{(4)}$、$M^{(4)}$、$S^{(4)}_1$	$X^{(2)}$、$Y^{(2)}$、$M^{(2)}$、$S^{(2)}_2$
8	$X^{(3)}$、$Y^{(0)}$、$M^{(0)}$、$S^{(0)}_3$	$X^{(1)}$、$Y^{(5)}$、$M^{(5)}$、$S^{(5)}_1$	$X^{(2)}$、$Y^{(3)}$、$M^{(3)}$、$S^{(3)}_2$
9	$X^{(3)}$、$Y^{(1)}$、$M^{(1)}$、$S^{(1)}_3$	$X^{(1)}$、$Y^{(6)}$、$M^{(6)}$、$S^{(6)}_1$	$X^{(2)}$、$Y^{(4)}$、$M^{(4)}$、$S^{(4)}_2$
10	$X^{(3)}$、$Y^{(2)}$、$M^{(2)}$、$S^{(2)}_3$	$X^{(4)}$、$Y^{(0)}$、$M^{(0)}$、$S^{(0)}_4$	$X^{(2)}$、$Y^{(5)}$、$M^{(5)}$、$S^{(5)}_2$
11	$X^{(3)}$、$Y^{(3)}$、$M^{(3)}$、$S^{(3)}_3$	$X^{(4)}$、$Y^{(1)}$、$M^{(1)}$、$S^{(1)}_4$	$X^{(2)}$、$Y^{(6)}$、$M^{(6)}$、$S^{(6)}_2$
12	$X^{(3)}$、$Y^{(4)}$、$M^{(4)}$、$S^{(4)}_3$	$X^{(4)}$、$Y^{(2)}$、$M^{(2)}$、$S^{(2)}_4$	$X^{(5)}$、$Y^{(0)}$、$M^{(0)}$、$S^{(0)}_5$
13	$X^{(3)}$、$Y^{(5)}$、$M^{(5)}$、$S^{(5)}_3$	$X^{(4)}$、$Y^{(3)}$、$M^{(3)}$、$S^{(3)}_4$	$X^{(5)}$、$Y^{(1)}$、$M^{(1)}$、$S^{(1)}_5$
14	$X^{(3)}$、$Y^{(6)}$、$M^{(6)}$、$S^{(6)}_3$	$X^{(4)}$、$Y^{(4)}$、$M^{(4)}$、$S^{(4)}_4$	$X^{(5)}$、$Y^{(2)}$、$M^{(2)}$、$S^{(2)}_5$
15		$X^{(4)}$、$Y^{(5)}$、$M^{(5)}$、$S^{(5)}_4$	$X^{(5)}$、$Y^{(3)}$、$M^{(3)}$、$S^{(3)}_5$
16		$X^{(4)}$、$Y^{(6)}$、$M^{(6)}$、$S^{(6)}_4$	$X^{(5)}$、$Y^{(4)}$、$M^{(4)}$、$S^{(4)}_5$
17			$X^{(5)}$、$Y^{(5)}$、$M^{(5)}$、$S^{(5)}_5$
18			$X^{(5)}$、$Y^{(6)}$、$M^{(6)}$、$S^{(6)}_5$

如表 2 所示，因为 $NW_{ym}>2NS$，同时 $NW_x>NS\times k$，导致 $S^{(0)}_2$ 在第 7 周期试图重新进入流水线受阻，被迫等到 Y、M 的字节流全部处理完才再次进入流水线。

对于 FIFO（30），其作用是暂存需要再次输入到流水线中的流水线输出数据。对于 FIFO(30)来说，它的输入数据是 carry-save 格式的，其输出数据也是 carry-save 格式的。在本发明中，为了节省 FIFO(30)，将 carry-save 格式的数据先行进行加法，然后再输入到存贮模块 FIFO(33)中。这样就可以节省一半的 FIFO 开销。如图 4 所示，先入先出寄存器 FIFO（30）包括一个由加法逻辑单元 DEF(32)和寄存器(31)组成的加法器，一个存贮模块

FIFO(33)，一个数据选择器(34)。寄存器 32 用来暂存进位信息。FIFO（33）可以用 register file 或者 ram 实现。数据选择器（34）用来旁路 FIFO，当 NW_{ym} 小于或等于 2NS 时不需要将数据输入到 FIFO 中。

本发明中，基本执行单元 MM Cell 结构框图如图 5 所示。MM Cell 是构成模乘流水线的关键部件。本发明设计了一种高基(基为 16)的 MM Cell。它包括 2 个寄存器 Register(36、37)、2 个 4－to－2 加法器(38、39)、2 个译码器 DEC（40、41）和一组数据选择器(42、43、44、45)；寄存器（36、37）将 MM Cell 内部分为两个层次，使得关键路径延迟更低，能有效提高时钟频率。加法器 4－to－2（38、39）采用 carry-save 形式，能降低加法器延迟。译码器（40、41）分别对应算法 4 中的 q_Y 和 q_M。数据选择器（42－45）根据译码器的输出 q_Y、q_M 选择相应的数据进入到 4－to－2 加法器（38、39）。

对于基为 16 的设计，译码器（40）的译码表如表 3 所示，译码器（41）的译码表分为两部分，一部分是预译码（如表 4 所示），另一部分是后译码，采用和译码器（40）同样的译码表（如表 3 所示）。

对于基为 16 的设计来说，乘数 X 每次扫描 4 位，经 Booth 编码后得到 5 位的输入数据，被乘数 Y 和模 M 每次输入一个字。乘数 X 的 Booth 编码如下式所示：

$$X^{(i)} = (x_{i+3}, x_{i+2}, x_{i+1}, x_i, x_{i-1})$$

x_{i-1} 是前一次扫描的数据最高位，q_Y 是 Booth 解码的结果，等于

$$q_Y = -8x_{i+3} + 4x_{i+2} + 2x_{i+1} + x_i + x_{i-1}$$

可以发现，q_Y 在[-8，8]范围内，为了方便计算 $S^{(0)} + (q_{Yj} \times Y)^{(0)}$（算法 4 中第 4 步），可以将其分解为两个 2 的幂级数来实现，如图 5 中所示。举例来说，假如 q_Y 等于 3，可以将其分解为 2 和 1，或者 4 和-1，这样 3×Y 就可以通过 2Y+Y 或者 4Y-Y 来实现，可以将 2Y、Y 或责 4Y、-Y 直接输入到 4-to-2 加法器，而不需要去真正做一次加法或者减法。这样，可以通过对 q_Y 需要建立一张查找表来简化计算。q_Y 查找表如表 3 所示，q_Y 是 Booth 解码结果，q_{Y1} 是数据选择信号，它决定 $Y^{(i)}_p$ 的值（如图 5），$Y^{(i)}_p$ 可以等于[0, Y, 2Y, 4Y, 8Y, -Y]；$Y^{(i)}_n$ 出 q_{Y2} 决定，它可以等于[0, -Y, -2Y, -4Y, -8Y, Y]。cin 用来处理数据取补码时的最低位，若数据是正数，则 cin 等于 0，输出数据不变；若数据是负数，则 cin 等于 1，同时将数据取反。

对于 q_M 来说，它取决于 S 和 M 的最低 4 位的值，如算法 2 中所示：

$$q_M = S^{(0)}_{k-1..0} \times (2^k - M^{(0)-1}_{k-1..0}) \bmod 2^k$$

在基为 16 的情况下，k 等于 4，q_M 的值在[0, 15]。经分析，[0, 15]中的数据 11 和 13 不能拆分为两个 2 的幂级数的和，这样就不能采用 4-to-2 的加法器来计算了，而需要再另

外加上一级加法器来计算，这样会导致关键路径延迟大大增加，所以这里需要对 q_M 做相应的处理以简化硬件设计。如算法 4 中第 6 步所示，系数 q_M 的目的是确保 S 的低 4 位的值变为 0，使得算法第 10 步能进行移位。在本发明中，通过将 q_M 转换到[-8, +8]范围内，使得 S 的低 4 位的值同样变为 0。系数 q_M（[0, 15]）转化为系数 q_M'（ [-8, 8]）如表 4 所示。q_M'类似于 q_Y，都可以通过表 3 用相应的数据选择器实现，如图 5 中所示。

表 3

q_Y	q_{Y1}	cin_1	q_{Y2}	cin_2	q_Y	q_{Y1}	cin_1	q_{Y2}	cin_2
0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	-1	0	0	-1	1
2	2	0	0	0	-2	0	0	-2	1
3	4	0	-1	1	-3	1	0	-4	1
4	4	0	0	0	-4	0	0	-4	1
5	4	0	1	0	-5	-1	1	-4	1
6	8	0	-2	1	-6	2	0	-8	1
7	8	0	-1	1	-7	1	0	-8	1
8	8	0	0	0	-8	0	0	-8	1

表 4

q_M	q_M'	q_M	q_M'	q_M	q_M'	q_M	q_M'
0	0	4	4	8	8	12	-4
1	1	5	5	9	-7	13	-3
2	2	6	6	10	-6	14	-2
3	3	7	7	11	-5	15	-1

附图说明

图 1 为模幂流水线结构框图。

图 2 为模幂流水线运算单元（ME）结构框图。

图 3 为模乘流水线结构框图。

图 4 为 FIFO 结构框图。

图 5 为模乘流水线运算单元（MM Cell）结构框图。

图中标号：1 为模幂流水线，2 为模幂流水线运算单元（ME）输入数据，3 为模幂流水线运算单元（ME）输出数据，4 为模幂流水线运算单元（ME）接口图，5 为模幂流水线运

算单元（ME）详细结构框图，6-7 为模乘单元（MM）框图，8 为 ME 的寄存器堆，9 为 ME 的控制逻辑单元，10-18 为 ME 的数据选择器，19 为模乘流水线，20-22 为模乘流水线运算单元（MM Cell）接口图，23-24 为模乘流水线的寄存器，25-27 为 MM Cell 的控制状态机，28-19 为模乘流水线的数据选择器，30 是 MM 的 FIFO 单元，31 是 FIFO 单元加法进位寄存器，32 是 FIFO 单元加法器，33 是 FIFO 单元的存贮模块，34 是 FIFO 单元选择器，35 是模乘流水线运算单元（MM Cell）详细框图，36-37 是 MM Cell 数据寄存器，38-39 是 4-to-2 加法器，40 是 q_Y 译码器，41 是 q_M 译码器，42-45 是 MM Cell 的数据选择器。

具体实施方式

下面结合附图和算法 3、4 和 5 进一步描述本发明。

根据模幂流水线算法 3 所示，流水线化的模幂运算将一次模幂分成多个流水线单元的子运算，从而使得每个流水线单元只需要化少量的时间来处理模幂。具体的量化执行时间为：

$$T_{ME}=T\times T_{MM} \qquad (1)$$

T_{ME} 代表模幂执行时间，T_{MM} 代表模乘执行时间，T 代表每个模幂流水线运算单元（5）所需要执行模乘运算的次数。根据上式，可以得到，模幂流水线的级数是

$$S_{ME}=(N+1)/T \qquad (2)$$

S_{ME} 代表模幂流水线的级数，N 代表 RSA 加密密钥的长度。

对于 T_{MM} 来说，T_{MM} 的大小取决于模乘流水线（19）的级数、基的大小、密钥长度和操作数字长。T_{MM} 计算公式如下：

$$T_{MM}=\begin{cases}\left\lceil\frac{N}{NS*k}\right\rceil*(2NS+1)+\frac{N}{BPW}+1 & IF\ \frac{N}{BPW}<=2NS\\ \left\lceil\frac{N}{NS*k}\right\rceil*(\frac{N}{BPW}+1)+2NS & IF\ \frac{N}{BPW}>2NS\end{cases} \qquad (3)$$

其中 N 表示 RSA 加密密钥位数，k 表示基的位数，NS 表示流水线级数，BPW 是算法 4 中 Y、M、S 的字长。

通过上述公式，最终可以得到双流水线结构的高速 RSA 密码协处理器的总执行时间等于：

$$T_{ME}=\begin{cases}(\left\lceil\frac{N}{NS*k}\right\rceil*(2NS+1)+\frac{N}{BPW}+1)*T & IF\ \frac{N}{BPW}<=2NS\\ (\left\lceil\frac{N}{NS*k}\right\rceil*(\frac{N}{BPW}+1)+2NS)*T & IF\ \frac{N}{BPW}>2NS\end{cases} \qquad (4)$$

相应的，模幂流水线级数

$$S_{ME}=\left\lceil\frac{N+1}{T}\right\rceil \quad (5)$$

模乘流水线级数

$$S_{MM}=NS \quad (6)$$

支持的密钥长度

$$L=2\times NS\times BPW+L_{FIFO} \quad (7)$$

其中 L_{FIFO} 等于 FIFO（33）的长度

本发明要实现的在 RSA 的加密速度方面和支持加密密钥长度方面的可配置性，就是通过调整上述参数来实现，具体的来说：

配置加密速度：根据公式（4），可以通过调整参数 T、k、NS、BPW 来实现

配置支持密钥长度：根据公式（7），可以通过调整 NS、BPW 和 L_{FIFO} 来实现

下面举例说明如何通过设定上述参数来配置 RSA 加密处理器的速度和支持密钥长度。

设计 SPEC：

支持密钥长度：1024 位

加密速度：每秒 5000 次

根据公式（7），要使得 L 等于 1024，可以设定 NS=16，BPW=32，$L_{FIFO}=0$

要使得加密速度达到 5000 次每秒的速度，需要采用高基的设计，这里设定 k=4。根据公式（3）可以得到

$$\begin{aligned}T_{MM} &= \left\lceil\frac{N}{k\times NS}\right\rceil\times(2NS+1)+\frac{N}{BPW}+1\\ &= \left\lceil\frac{1024}{4\times 16}\right\rceil\times(2\times 16+1)+\frac{1024}{32}+1\\ &= 561\end{aligned}$$

此时，可以根据已经设计好的 MM 来做综合，得到 MM 的最大时钟频率。根据实验分析，在 0.25μm 的工艺下，该 MM 能达到 150MHz 的时钟频率。在 150MHz 的频率下，做一次模乘的时间为 3.74×10^{-6} 秒，为了达到每秒 5000 次的速度，每个模幂运算单元（5）在每次 RSA 加密运算中只能做 52 次的模乘。

$$52\times5000\times3.74\times10^{-6}\approx1$$

所以，模幂流水线级数

$$S_{ME}=\left\lceil\frac{N+1}{T}\right\rceil=\left\lceil\frac{1023}{52}\right\rceil=20$$

通过采用上述的设计参数，就可以方便地设计出支持各种密钥长度和加密速度的 RSA 密码协处理器。

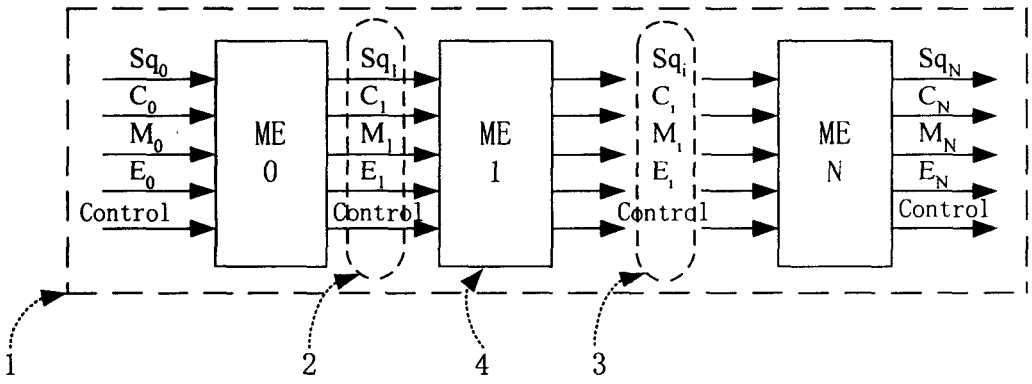
Sq0
C0
M0
E0
Control
ME 0
Sqi
Ci
Mi
Ei
Control
ME 1
Sqi
Ci
Mi
Ei
Control
ME N
SqN
CN
MN
EN
Control
1
2
4
3

图 1

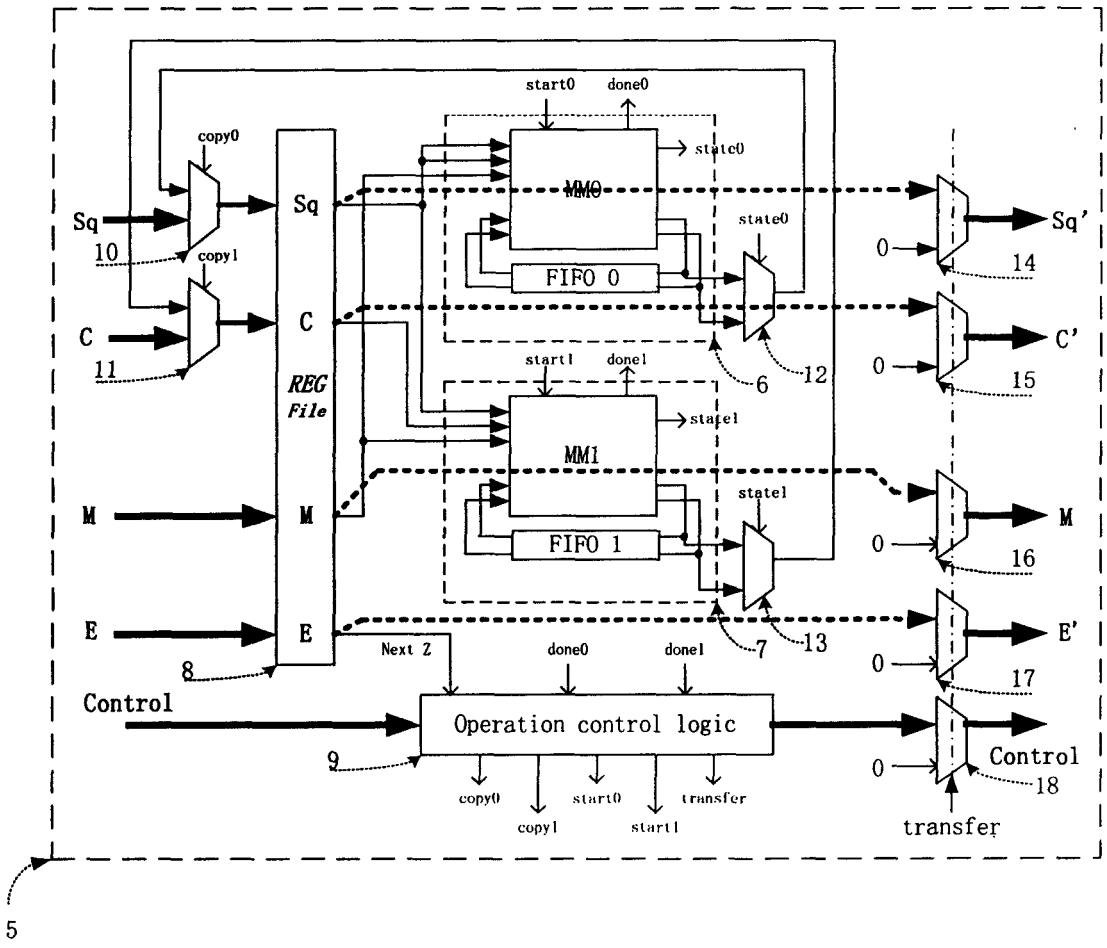
start0
done0
state0
MM0
FIFO 0
copy0
copy1
Sq
C
REG File
M
E
Next Z
done0
done1
Control
Operation control logic
copy0
copy1
start0
start1
transfer
Sq'
C'
M
E'
Control
transfer
10
11
6
12
7
13
8
9
14
15
16
17
18
5

图 2

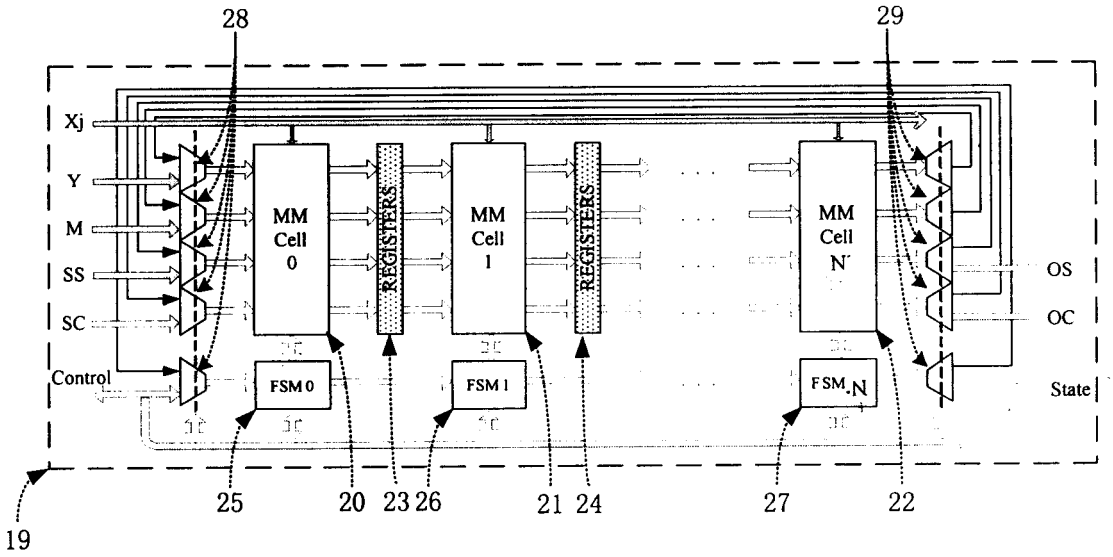
28
29
Xj
Y
M
SS
SC
Control
MM Cell 0
MM Cell 1
MM Cell N
REGISTERS
REGISTERS
FSM 0
FSM 1
FSM N
OS
OC
State
19
25
20
23
26
21
24
27
22

图 3

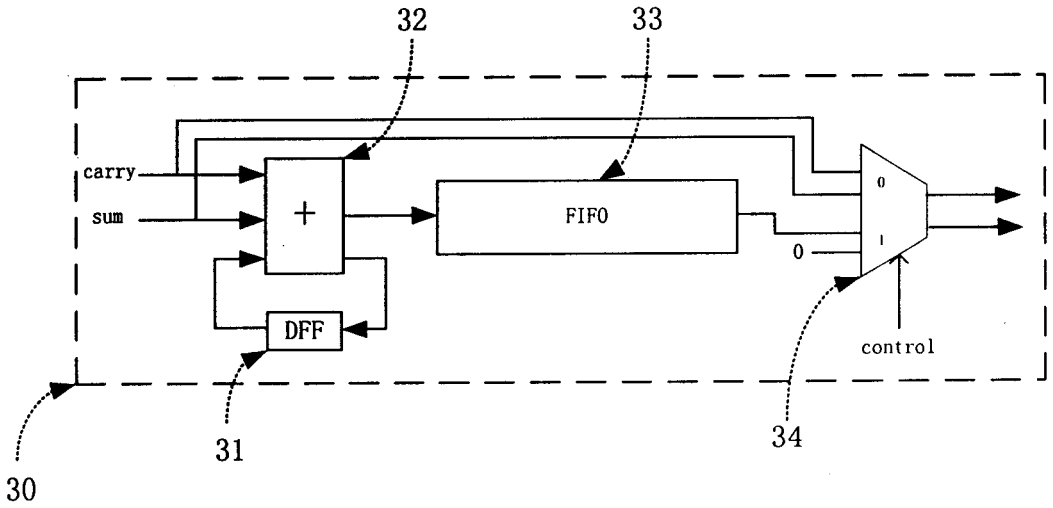
32
33
carry
sum
+
FIFO
DFF
0
1
0
control
30
31
34

图 4

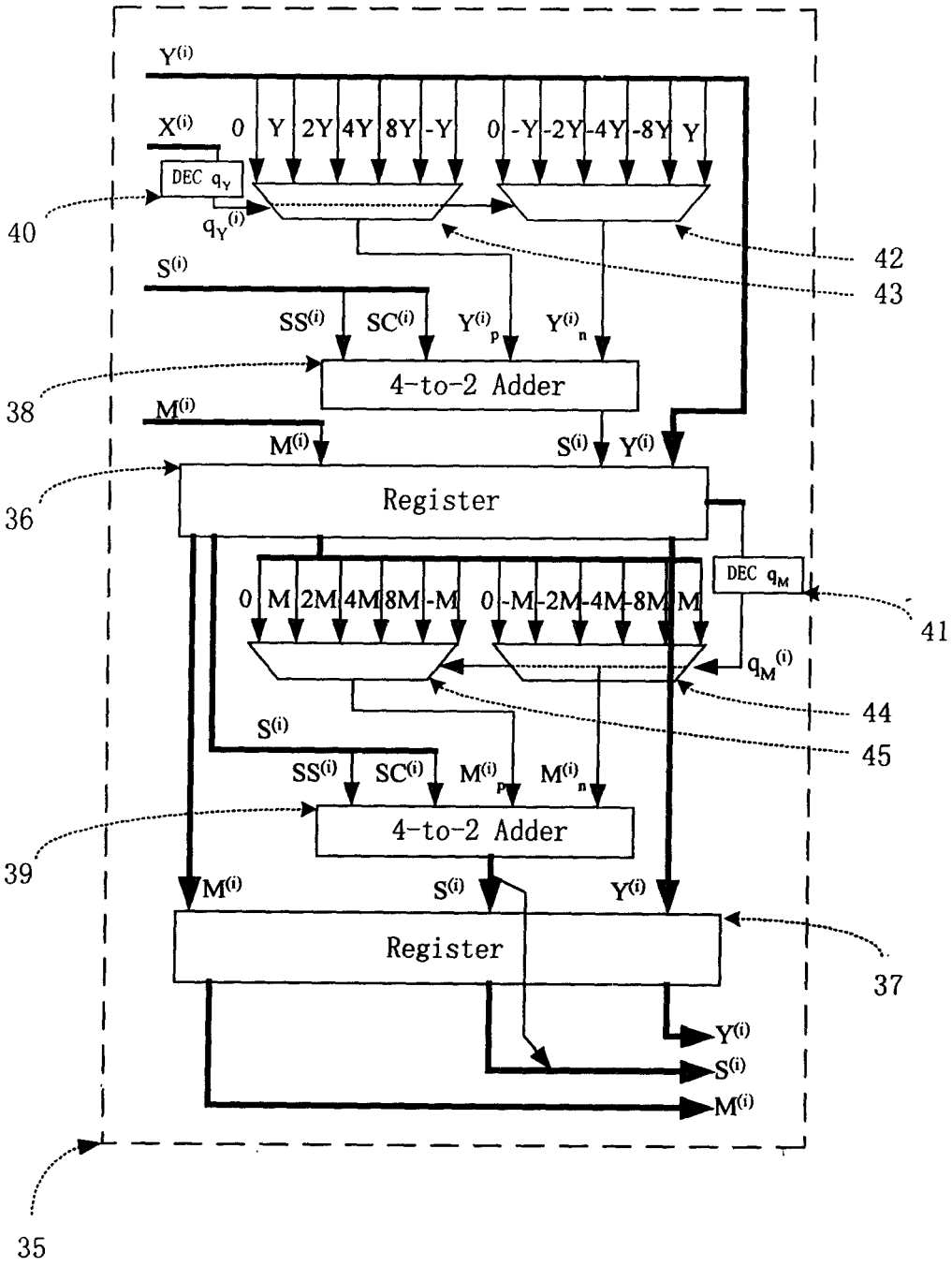
Y(i)
X(i)
0 Y 2Y 4Y 8Y -Y 0 -Y -2Y -4Y -8Y Y
DEC qY
qY(i)
40
42
43
S(i)
SS(i) SC(i) Yp(i) Yn(i)
4-to-2 Adder
38
M(i)
M(i) S(i) Y(i)
Register
36
DEC qM
41
0 M 2M 4M 8M -M 0 -M -2M -4M -8M M
qM(i)
44
45
S(i)
SS(i) SC(i) Mp(i) Mn(i)
4-to-2 Adder
39
M(i) S(i) Y(i)
Register
37
Y(i)
S(i)
M(i)
35

图 5