



(12) 发明专利

(10) 授权公告号 CN 112486457 B

(45) 授权公告日 2022. 12. 20

(21) 申请号 202011319638.3

(56) 对比文件

(22) 申请日 2020.11.23

US 2015277855 A1, 2015.10.01

(65) 同一申请的已公布的文献号
申请公布号 CN 112486457 A

审查员 余祖滢

(43) 申请公布日 2021.03.12

(73) 专利权人 杭州电子科技大学
地址 310018 浙江省杭州市下沙高教园区2
号大街

(72) 发明人 王敏杰 孙浩 孙玲玲

(74) 专利代理机构 杭州君度专利代理事务所
(特殊普通合伙) 33240
专利代理师 杨舟涛

(51) Int. Cl.
G06F 7/72 (2006.01)

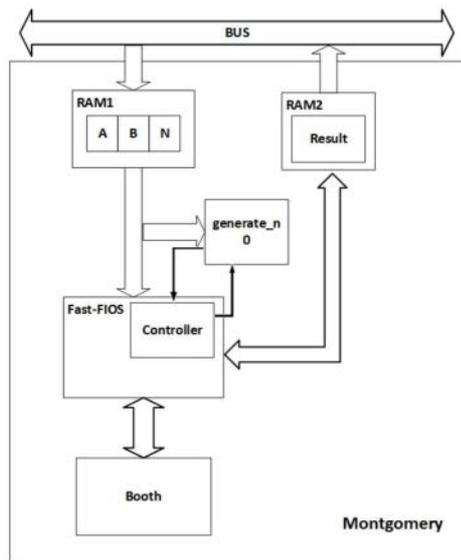
权利要求书1页 说明书4页 附图4页

(54) 发明名称

一种实现改进的FIOS模乘算法的硬件系统

(57) 摘要

本发明公开了一种实现改进的FIOS模乘算法的硬件系统。本发明采用硬件实现模乘电路，通过寄存器复用，减少逻辑资源消耗；重新排布流水线和整个算法时序，将关键路径的加法操作拆解为多级流水的加法树操作，使得运行速率最高达到600MHz；通过并行化处理无依赖的操作，提高单个时钟周期内的操作个数；使用基128的乘法器作为基本计算单元，完成4096位的模乘只需要3463个周期，耗时约5.75us，显著的减少计算过程中的循环次数，减少运算所需时钟数，提高单位时间内的计算吞吐率。本发明对部分积生成电路进行改进，进一步减少了逻辑门的使用。本发明通过改进降低了蒙哥马利模乘算法中的代码长度，提高了模乘过程的运行效率。



1. 一种实现改进的FIOS模乘算法的硬件系统,其特征在于:包括存储模块、模乘算法模块、booth乘法器模块以及模乘参数 n' [0]选取模块;

所述存储模块包括RAM1与RAM2两个存储器,其中RAM1用于存储输入的被乘数A、乘数B和模数N,RAM2用于存储输出结果;

所述模乘算法模块增加了一组寄存器,通过并行化处理预计算环节1与进位计算环节3,压缩了流水线级数,将蒙哥马利模乘的关键路径在一个循环中进行三个数连接的地方,拆解为两步完成连接计算;模乘算法模块用于将4096bit的被乘数A、乘数B和模数N分为等长的若干字段,完成 $A*B \bmod N$ 的模乘计算;

所述booth乘法器模块包括部分积产生模块、部分积压缩模块及向量相加模块;将现有的乘法器部分积产生模块的后续电路替换为多路选择器;部分积压缩模块采用CSA压缩器完成十级压缩,并在最后一级电路前插入了一级寄存器,实现将输入向量压缩至两行;向量相加模块通过超前进位加法器将部分积压缩模块输出的两行向量进行相加;完成模乘计算过程中的乘法操作;

所述模乘参数 n' [0]选取模块包括一个输入口、一个输出口、一个循环移位模块、两个D触发器、一个加法器和一个二选一选择器;所述循环移位模块的输入与模乘参数 n' [0]选取模块的输入口相连,输出端与第一D触发器的D端相连;所述加法器的一个输入端与第一D触发器的Q端相连,另一个输入端输入乘数B,加法器的输出端与第二D触发器的D端相连;第二D触发器的Q端通过二选一选择器与模乘参数 n' [0]选取模块的输出口相连,输出被乘数A;

系统从输入读取4096bit的被乘数A、乘数B和模数N,并将其存储在RAM1中;模乘参数 n' [0]选取模块读取RAM1中存储的模数N,预计算得到模数N的低128位 n' [0];模乘算法模块读取RAM1中存储的被乘数A与乘数B以及模乘参数 n' [0]选取模块计算得到的 n' [0],通过调用booth乘法器完成乘法操作,将最后得到的结果存储到RAM2中,完成整个模乘运算过程。

2. 如权利要求1所述一种实现改进的FIOS模乘算法的硬件系统,其特征在于:所述RAM1的大小为 $96*128\text{bit}$,所述RAM2的大小为 $40*128\text{bit}$ 。

一种实现改进的FIOS模乘算法的硬件系统

技术领域

[0001] 本发明涉及数据加解密领域,具体设计一种实现改进的FIOS模乘算法的硬件系统。

背景技术

[0002] 随着互联网技术的飞速发展,信息物联已经渗透到社会生活的方方面面,而信息安全技术的发展却相对滞后,大数据时代给人们带来便利的同时,也带来了许多的安全问题,如何保障网络信息安全成为了人们研究的热点问题。

[0003] 保障网络信息安全的关键技术之一就是加解密技术,目前,广泛使用的加密形式有两种:传统加密和公钥加密。1978年期间,R.Rivest、A.Shamir及L.Adleman提出了RSA算法,RSA公钥加密算法也是目前理论上最成熟完善和应用最广泛的加解密算法机制。大整数因子分解的困难性决定了RSA加密的安全性,因此模乘运算是RSA算法中最核心的运算。

[0004] RSA加解密算法,主要有软件实现和硬件实现两种方式。与传统的软件实现加密相比,硬件实现的方式可以使加密更加稳定,速度更快且兼容性更好,安全性更高。而提高模乘运算速度是改善RSA性能的关键,目前运用最多的算法是蒙哥马利算法。在蒙哥马利算法中,由于运算数据和中间结果的位数很大,需要消耗很多硬件资源,运行效率也会降低。

发明内容

[0005] 针对现有技术的不足,本发明提出了一种实现改进的FIOS模乘算法的硬件系统,通过复用寄存器和重新排布流水线等方式,减少硬件资源消耗的同时,有效减少计算过程中的循环次数,减少运算所需时钟数,达到提高单位时间内计算吞吐率即提高RSA加密算法速度的目的。

[0006] 一种实现改进的FIOS模乘算法的硬件系统,包括存储模块、模乘算法模块、booth乘法器模块以及模乘参数 n' [0]选取模块。

[0007] 所述存储模块包括RAM1与RAM2两个存储器,其中RAM1用于存储输入的被乘数A、乘数B和模数N,RAM2用于存储输出结果。

[0008] 作为优选,RAM1的大小为 $96*128\text{bit}$,RAM2的大小为 $40*128\text{bit}$ 。

[0009] 所述模乘算法模块在现有的蒙哥马利模乘器的基础上增加了一组寄存器,实现预计算环节与进位计算环节的并行计算,然后再计算迭代环节。模乘算法模块用于将 4096bit 的被乘数A、乘数B和模数N分为等长的若干字段,完成 $A*B\text{mod}N$ 的模乘计算,输出保持在 $0\sim 2N$ 之间。

[0010] 所述booth乘法器模块包括部分积产生模块、部分积压缩模块及向量相加模块。将现有的乘法器部分积产生模块的后续电路替换为多路选择器。部分积压缩模块将输入向量压缩至两行,采用CSA压缩器完成十级压缩,并在最后一级电路前插入了一级寄存器,优化时序。向量相加模块通过超前进位加法器将部分积压缩模块输出的两行向量进行相加。完成模乘计算过程中的乘法操作。

[0011] 所述模乘参数 $n'[0]$ 选取模块包括一个输入口、一个输出口、一个循环移位模块、两个D触发器、一个加法器和一个二选一选择器。所述循环移位模块的输入与模乘参数 $n'[0]$ 选取模块的输入口相连,输出端与第一D触发器的D端相连;所述加法器的一个输入端与第一D触发器的Q端相连,另一个输入端输入乘数B,加法器的输出端与第二D触发器的D端相连。第二D触发器的Q端通过二选一选择器与模乘参数 $n'[0]$ 选取模块的输出口相连,输出被乘数A。

[0012] 基于上述硬件系统的模乘计算过程为:系统从输入读取4096bit的被乘数A、乘数B和模数N,并将其存储在RAM1中。模乘参数 $n'[0]$ 选取模块读取RAM1中存储的模数N,预计算得到模数N的低128位 $n'[0]$ 。模乘算法模块读取RAM1中存储的被乘数A与乘数B以及模乘参数 $n'[0]$ 选取模块计算得到的 $n'[0]$,通过调用booth乘法器完成乘法操作,将最后得到的结果存储到RAM2中,完成整个模乘运算过程。

[0013] 本发明具有以下有益效果:

[0014] 1、通过在原有的蒙哥马利模乘器的基础上增加一组寄存器以及寄存器复用,重新排布流水线与算法时序,实现预计算环节与进位计算环节的并行计算,降低了代码长度,以及硬件资源的消耗,减少了 s^2-s 次的字加、 $2s^2-s$ 次的读和 $2s^2-s$ 次的写,使模乘器的最大运行速率可以达到600MHz。

[0015] 2、改进了booth乘法器,在关键路径插入一级寄存器,减少逻辑门的使用,提高了乘法器的计算位宽与计算速度,同时降低了硬件面积消耗,减少了模乘计算过程中的循环次数,从而减少运算周期。

[0016] 3、模乘算法模块通过调用booth乘法器完成乘法部分的计算,提高了单位时间内的处理量级,实现4096位的模乘只需要3463个周期,耗时约5.75us,提高了单位时间内的处理量级。

附图说明

[0017] 图1为本发明的原理框图;

[0018] 图2为本发明改进后的模乘计算过程;

[0019] 图3为本发明模乘计算过程的数据转移图;

[0020] 图4为本发明部分积编解码方案的电路逻辑;

[0021] 图5为本发明部分积编解码方案的优化电路逻辑;

[0022] 图6为本发明部分积压缩过程;

[0023] 图7为本发明模乘参数 $n'[0]$ 选取模块的电路图。

具体实施方式

[0024] 以下结合附图对本发明作进一步的解释说明;

[0025] 如图1所示,一种实现改进的FIOS模乘算法的硬件系统,包括存储模块、模乘算法模块、booth乘法器模块以及模乘参数 $n'[0]$ 选取模块。

[0026] 所述存储模块包括 $96*128\text{bit}$ 的RAM1与 $40*128\text{bit}$ 的RAM2两个存储器,其中RAM1用于存储输入的被乘数A、乘数B和模数N,RAM2用于存储输出结果。

[0027] 现有技术中的蒙哥马利模乘算法为:

Input: $A(a_s a_{s-1} \dots a_0), B(b_s b_{s-1} \dots b_0), N(n_s n_{s-1} \dots n_0), n'[0] = -n[0]^{-1} \bmod w$

Output: $T(t_s t_{s-1} \dots t_0)$

1: *for* $i = 0$ *to* $s - 1$

2: $(R0, S0) = t[0] + a[0] * b[i];$

3: $m = S * n' [0] \bmod w;$

4: $(R1, S0) = S + m * n[0];$ 1

5: *for* $j = 1$ *to* $s - 1$

[0028] 6: $(R0, S1) = t[j] + a[j] * b[i] + R0;$

7: $(R1, S1) = S + m * n[j] + R1;$

8: $t[j - 1] = S;$ 2

9: $(R0, S1) = R0 + R1;$

10: $(R1, S1) = t[s] + S;$

11: $t[s - 1] = S1;$

12: $t[s] = R0 + R1;$ 3

[0029] 所述模乘算法模块,增加了一组寄存器,通过并行化处理预计算环节1与进位计算环节3,压缩了流水线级数,将原算法的关键路径在一个循环中进行三个数连接的地方,拆解为两步完成连接计算。模乘算法模块的数据转移如图3所示,算法为:

Input: $A(a_s a_{s-1} \dots a_0), B(b_s b_{s-1} \dots b_0), N(n_s n_{s-1} \dots n_0), n'[0] = -n[0]^{-1} \bmod w$

Output: $T(t_s t_{s-1} \dots t_0)$

1: *for* $i = 0$ *to* $s - 1$

2: $(R0, S0) = t[0] + a[0] * b[i];$ $(R0, S1) = R0 + R1;$

3: $m = S * n' [0] \bmod w;$ $(R1, S1) = t[s] + S;$

[0030] 4: $(R1, S0) = S + m * n[0];$ $t[s - 1] = S1;$ $t[s] = R0 + R1;$ 1

5: *for* $j = 1$ *to* $s - 1$

6: $(R0, S1) = t[j] + a[j] * b[i] + R0;$

7: $(R1, S1) = S + m * n[j] + R1;$

8: $t[j - 1] = S;$ 2

[0031] 所述booth乘法器,包括部分积产生模块、部分积压缩模块及向量相加模块。将现有的乘法器部分积产生模块的后续电路替换为多路选择器,根据编码逻辑关系与部分积生成表达式,可以得到图4所示的编解码方案电路逻辑图,部分积产生模块的电路逻辑图如图

5所示。部分积压缩模块将输入部分积产生模块产生的向量通作为Wallace树形结构的输入,经过CSA压缩器进行十级压缩,压缩至两项。在部分积压缩模块的最后一级电路前插入了一级寄存器,保证电路的保持时间,使电路可以达到更高的时钟频率。将压缩得到的两行向量输入到向量相加模块,通过超前进位加法器将部分积压缩模块输出的两行向量进行相加,得到最终的乘法运算结果。

[0032] 所述模乘参数 $n'[0]$ 选取模块包括一个输入口、一个输出口、一个循环移位模块、两个D触发器、一个加法器和一个二选一选择器,电路结构如图7所示,输入模数N的低128位 n , n 的位宽与booth乘法器的位宽相关;通过循环移位模块对 n 进行左循环移位处理,经过第一个D触发器后与乘数B通过加法器相加,通过第二个D触发器暂存输出进行 $B+n[0]$ 的计算,最终通过二选一选择器输出被乘数A,经过八次循环输出后得到一个完整字节,即完成 $n'[0]$ 的计算, $n'[0]$ 的算法为:

Input: $n[0]$

[0033] *Output: $n'[0] = -n[0]^{-1} \bmod w$*

1: $A = 0, B = 1;$

2: *for* $i = 0$ *to* k ($k = \log_2 w$)

3: *if* $B[i] == 1$ *then*

4: $A[i] = 1;$

5: $B = B + n[0];$

[0034]

6: *else*

7: $A[i] = 0;$

8: $n[0] = 2n[0] \ll 1;$

9: *return* A

[0035] 系统从输入读取4096bit的被乘数A、乘数B和模数N,并将其存储在RAM1中。模乘参数 $n'[0]$ 选取模块读取RAM1中存储的模数N,预计算得到模数N的低128位 $n'[0]$ 。模乘算法模块读取RAM1中存储的被乘数A与乘数B以及模乘参数 $n'[0]$ 选取模块计算得到的 $n'[0]$,通过调用booth乘法器完成乘法操作,将最后得到的结果存储到RAM2中,完成整个模乘运算过程。

[0036] 以上所述仅是本发明的优选实施方式,应当指出,对于本技术领域的普通技术人员,在不脱离本发明构思的前提下,还可以做出若干改进和润饰,这些改进和润饰也应视为本发明保护范围内。

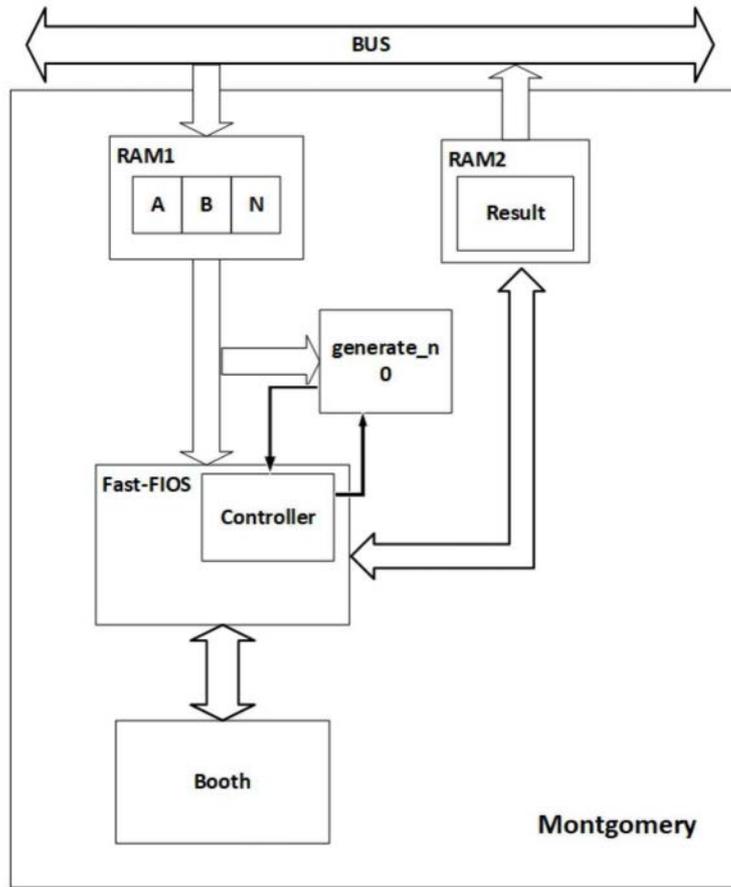


图1

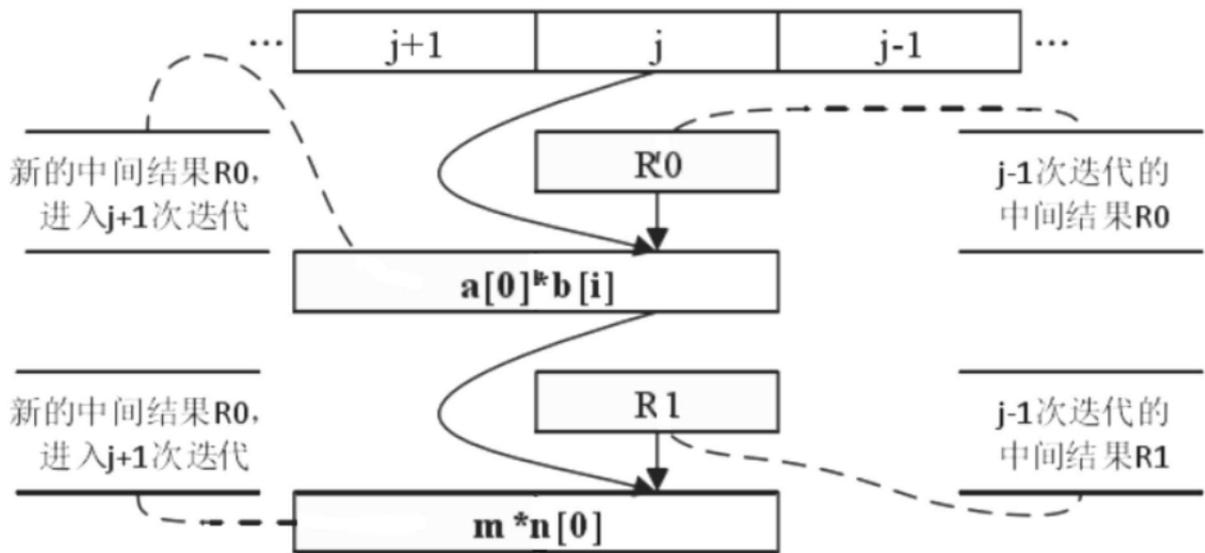


图2

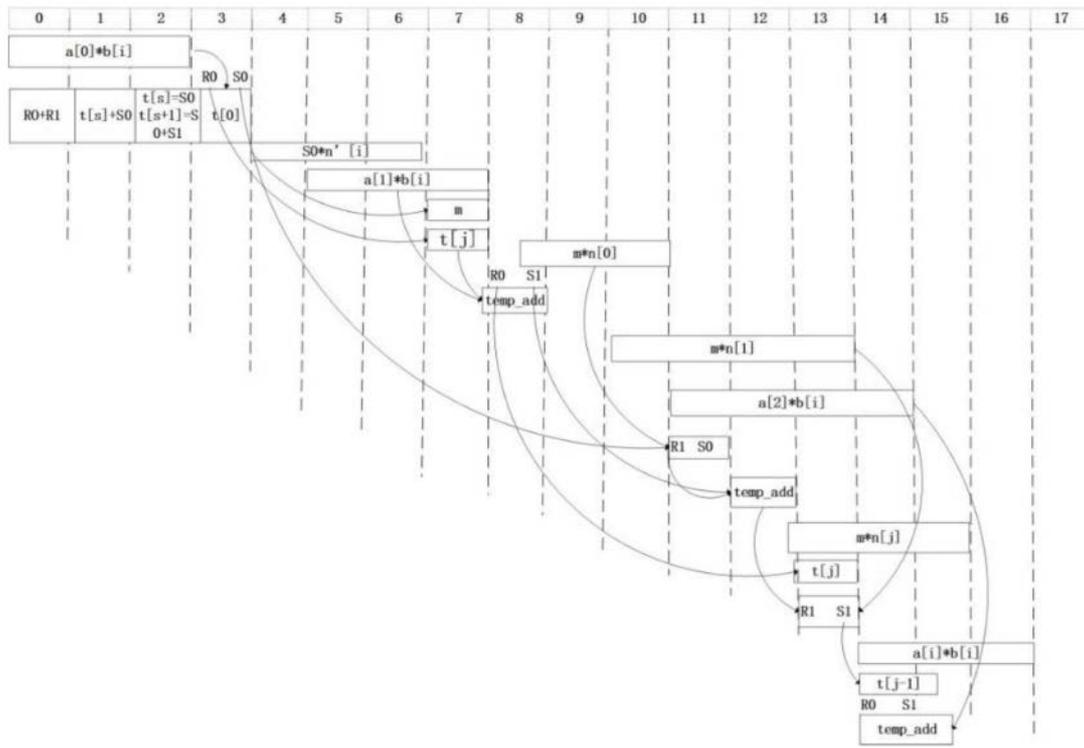


图3

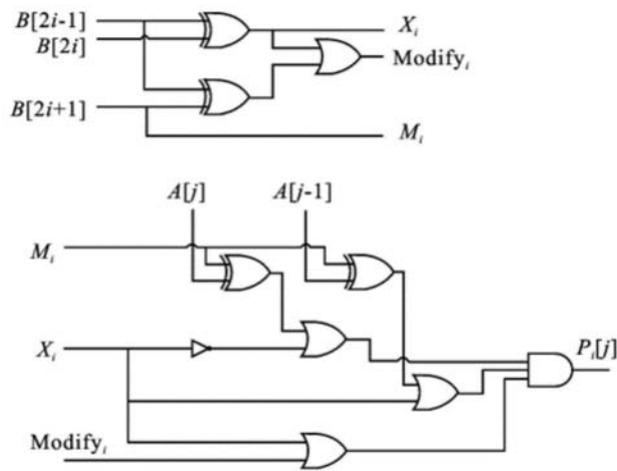


图4

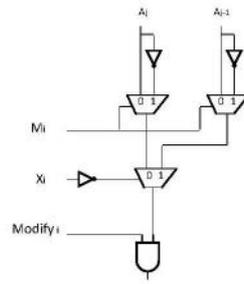


图5

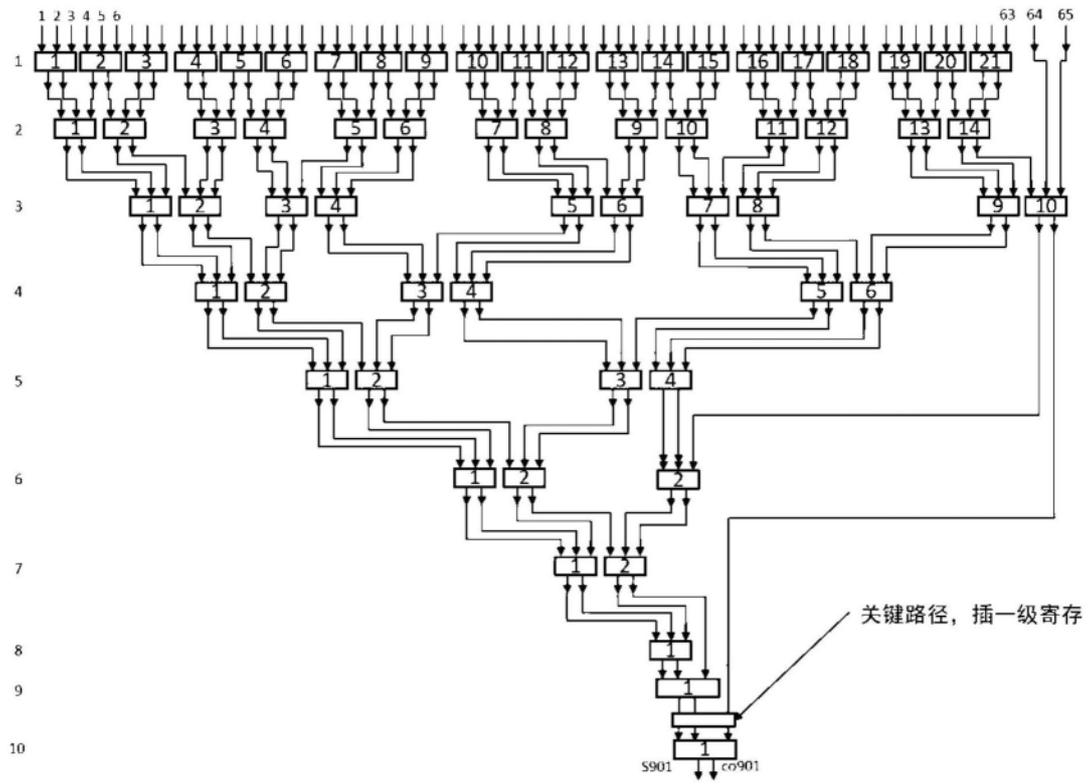


图6

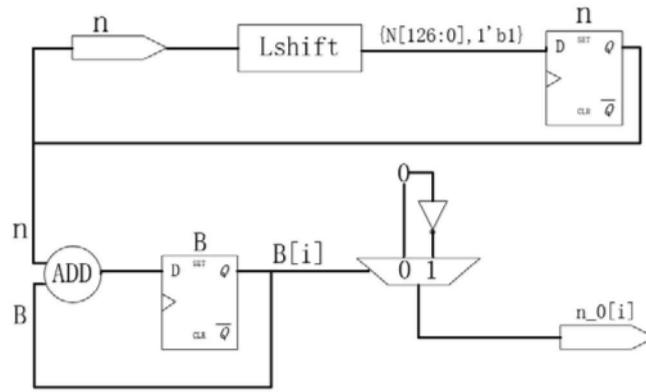


图7