第 20 卷 第 5 期 上海大学学报(自然科学版) Vol. 20 No. 5
2014年10月 JOURNAL OF SHANGHAI UNIVERSITY (NATURAL SCIENCE) Oct. 2014

DOI: 10.3969/j.issn.1007-2861.2014.01.003

三值光学计算机的 40 位乘法例程

胡晓俊, 金 翊, 欧阳山
(上海大学 计算机工程与科学学院, 上海 200444)

摘要: 针对三值光学计算机的特点, 利用其运算器可重构、数据位数众多、MSD 加法器无进位延时等优点, 设计并实现了一种用于三值光学计算机的 40 位乘法例程. 该例程采用三值光学计算机中通用的 MSD 数表示数值, 通过三值逻辑中的 M 变换产生部分积, 再运用两两相加迭代的计算方法对部分积进行了 MSD 加法求和, 得到乘积, 其中 M 变换采用了一种比较特殊的快速变换实现方案, 而部分积的 MSD 加法求和则采用流水技术来实现. 详细给出了这个乘法例程的具体实现步骤和模拟实验细节, 并与电子计算机中类似的乘法器做了运算复杂度对比分析.

关键词: 三值光学计算机; 乘法例程; MSD加法器
中图分类号: TP 332.21 **文献标志码:** A **文章编号:** 1007-2861(2014)05-0645-13

A 40-Bit Multiplication Routine of Ternary Optical Computer

HU Xiao-jun, JIN Yi, OUYANG Shan
(School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China)

Abstract: According to the advantages of ternary optical computer such as reconfigurable arithmetic unit, large number of data bits and no-carry-delay MSD adder, a 40-bit multiplication routine is designed and developed. In the routine, the numerical value is expressed with the MSD number system as in a ternary optical computer. Partial products are generated via a three-valued logic transform (M). The product is obtained by summing all partial products through an MSD adder using an iterative method. A fast-calculation method is applied in the M transform, and a pipeline technology used in the MSD adder to accumulate the partial products. The operation steps and simulation experiments of the routine are given in detail, and the performance comparing with electronic computer analyzed.

Key words: ternary optical computer; multiplication routine; MSD adder

以面向应用研究的实验系统 SD11 (上大 2011) 为标志, 三值光学计算机已经展现出数据位数众多、处理器按位可重构两个特点[1]. 这种计算机利用光的两个相互正交的偏振状态 (如

收稿日期: 2013-10-16
基金项目: 国家自然科学基金资助项目(61073049); 国家自然科学基金青年基金资助项目(61103054); 上海市教委科研创新基金资助项目(13ZZ074, 13YZ005); 上海市自然科学基金资助项目(13ZR1416000); 上海高校青年教师培养基金资助项目(ZZSD13035)
通信作者: 金 翊(1962—), 男, 教授, 博士生导师, 研究方向为三值光学计算机、计算机体系结构和人工智能等. E-mail: yijin@shu.edu.cn

水平方向和垂直方向) 和无光强状态来表示信息, 因此每个数据位有 3 个取值[2]. 另外, 三值光学计算机采用像素数量众多的液晶阵列来构造光状态变换器, 使得其数据位数众多. 由于它依据降值设计理论来构造光学处理器[3], 因此每一个数据位都可以按需重构成为 19 683 种二元三值逻辑变换器中的任何一种.

2009 年, 金翊等[4] 提出了一种三值光学计算机的 MSD 加法器理论和结构, 即将三值光学计算机推进到数值计算领域, 进一步拓展了这种新型计算机的数值计算能力, 其乘法例程也成为一个重要的研究课题.

回顾电子计算机数字乘法器的发展和进程, 从移位相加乘法器、Booth 乘法器、加法器阵列乘法器到 Wallace 树乘法器等[5-8]的发展历程, 它们始终围绕着提高运算效率这个主题. 由于数字乘法的主要运算量集中在部分积求和阶段, 因此提高运算效率的手段大致可以分为两类: ① 从直接位与产生部分积, 发展为通过对操作数的编码 (Booth 编码) 后产生部分积, 以此来减少部分积的数量, 继而减少部分积求和的次数; ② 从保留进位加法器阵列结构对部分积求和, 发展到通过 ZM 树结构、OS 树结构和 Wallace 结构等技术对部分积进行求和[7], 增强了部分积求和的并行程度, 从而提高了求和速度.

电子计算机乘法器的成熟技术可以为研究三值光学计算机乘法例程提供参考, 但是鉴于现阶段三值光学计算机部件的制造能力所限, 过于复杂的结构和方案不适合在当前 SD11 上得以快速的工程实现. 因此必须寻找一种切实可行的工程实现方案来改变当前三值光学计算机中尚无乘法器/例程的现状. 在目前的各种乘法器中, 移位相加乘法器的结构简单且易于实现, 它是通过逻辑与电路产生部分积, 用迭代的方式对部分积进行移位相加, 两两求和的最终结果就是乘积. 这种方案的缺点是由于每轮迭代中都有部分积相加的串行进位延时, 严重地制约了乘法运算的速度. 在电子计算机中, 虽然理论上已通过先行进位加法器改善求和效率, 但是随着操作数位数的增加, 进位电路的复杂程度迅速提高, 以至于超过 5 位的先行进位加法器难以实现. 然而, 由于三值光学计算机采用的 MSD 加法器具有无进位的特点, 恰好弥补了这个方案的不足. 本研究正是利用了三值光学计算机的这个特点完成了一款三值光学计算机中使用的移位相加乘法例程.

可重构光学处理器的每个基本模块有 1 024 位, 每个 SD11 最多可接入 16 个基本模块, 而每个基本模块由 4 组液晶阵列构成, 每个液晶阵列有 512 个像素. 鉴于40位的乘法例程需要液晶像素 489 个, 针对 SD11 光学处理器使用的液晶阵列有 512 个像素之特点, 本研究以 40 位 MSD 乘法例程为例, 详细讨论了构造乘法例程的原理和方法. 这个例程的基本思想是: 在完成乘法任务的过程中, 使用最少的控制语句和数据组织语句来形成规范的步骤, 尽量发挥三值光学计算机的特点. 本研究已经将完成的软件包装成一个基本例程, 该例程放在三值光学计算机的底层软件包中供用户调用. 用户只需在 C 语言环境下调用 SZGCYY-GH2012 软件提供的扩展机制[9], 或在 MPI 环境下调用 SZGMPI-ZQ2012 软件提供的扩展机制[10], 并在 SZG 文件中输入 2 个相乘的因子和乘法运算符, 该例程将自动完成乘法运算并输出计算结果. 因此, 对用户而言, 利用三值光学计算机做乘法运算只是使用一条“乘法指令”.

1 MSD数据的乘法

本研究描述的 MSD 数据的乘法是指: 使用 M 变换和 MSD 加法器来完成 2 个 MSD 数据的相乘. 设被乘数 A 有 p 位, 记 $A = a_p \cdots a_i \cdots a_2 a_1$, 其 MSD 表达式 $A = \sum a_i \times 2^{i-1}$; 乘数 B 有 q 位, 记 $B = b_q \cdots b_j \cdots b_2 b_1$, 其 MSD 表达式 $B = \sum b_j \times 2^{j-1}$. 其中 $i = 1, 2, \cdots, p$; $j = 1, 2, \cdots, q$; a_i 和 b_j 的值域都为 $\{\bar{1}, 0, 1\}$, $\bar{1}$ 表示 -1; 2^{i-1} 表明该 MSD 数据是二进制计数

法. A 和 B 的乘积 C 可根据式 (1) 确定:

$$C = A \times B = A \times \left(\sum b_j \times 2^{j-1} \right) = \sum A \times b_j \times 2^{j-1} = \sum S_j \times 2^{j-1} = \sum P_j, \qquad (1)$$

式中, 部分积 $S_j = A \times b_j$, 和数项 $P_j = S_j \times 2^{j-1}$. 由于 b_j 只有一位, 取值为 $\bar{1}$, 0 和 1, 所以只要 A 逐位与 b_j 进行 M 变换 (见表 1) 就可以得到部分积 S_j, 而 M 变换实质上就是一种三值逻辑变换. 式 (1) 中的因子 2^{j-1} 表明只要将部分积 S_j 左移 $j-1$ 位 (或在 S_j 后面补 $j-1$ 个 0) 就可以得到和数项 P_j, 然后对所有和数项 P_1, P_2, $\cdots$, P_q 用 MSD 数加法进行求和, 就可以得到最终乘积 C.

表 1　M变换真值表

Table 1　M transformation

a	b		
	$\bar{1}$	0	1
$\bar{1}$	0	1	1
0	0	0	0
1	$\bar{1}$	0	1

由此可知, MSD 数据的乘法可以由表 1 给出的 M 变换和 MSD 加法器来完成, 其运算过程可分为以下 4 个步骤.

步骤 1　由 B 的每一位 b_j 生成一个有 p 位 b_j 值的辅助数据 B_j, $B_j = b_j \cdots b_j \cdots b_j b_j$ (共 p 位), 得到 q 个新数据 $B_1, B_2, \cdots, B_q$.

步骤 2　将 A 与每个 B_j 按位进行 M 变换, 变换结果为 q 个部分积 $S_1, S_2, \cdots, S_q$.

步骤 3　将每个部分积 S_j 左移 $j-1$ 位, 即在每个 S_j 后面补 $j-1$ 个 0, 得到 q 个和数项 $P_1, P_2, \cdots, P_q$.

步骤 4　对 q 个和数项 $P_1, P_2, \cdots, P_q$ 用 MSD 加法器进行求和, 累加的结果为乘积 C.

2　M变换的实现方案

MSD 数据乘法运算过程中的步骤 2 需要将 A 分别与 q 个 B_j 按位进行 M 变换, 因此在三值光学计算机中有 2 种方式来完成这步运算.

方式 1　在三值光学计算机上重构出 1 个 p 位的 M 变换器, 依次将 q 对数据进行 M 变换. 这种方式所需光学处理器的位数为 p, 所需光学处理器的操作周期数为 q.

方式 2　在三值光学计算机上重构出 q 个 p 位的 M 变换器, 同时完成 q 对数据的 M 变换. 这种方式所需光学处理器的位数为 $p \times q$, 所需光学处理器的操作周期数为 1.

三值光学处理器的操作周期等于其完成一次逻辑运算所需的时间, 主要由处理器中液晶的响应时间 t 决定. 一般情况下, 一次逻辑运算的过程首先需要经过编码液晶的响应, 产生主光路和控制光路上稳定的三值光输入信号, 然后再由控制光路上的稳定信号控制运算液晶的旋光状态, 等待运算液晶响应完成后才能输出稳定的运算结果信号. 这是一个需要串行完成的同步过程, 因此光学处理器的操作周期 T 约等于 2 次液晶响应的时间, 即 $T = 2t$.

当使用方式 1 进行 M 变换时, 由于 q 对数据中都含有 A, 如果将 A 作为控制光路的输入数据, 那么只需经过一次控制光路编码和运算液晶响应, 之后保持运算液晶的旋光状态不变, 剩下的 $q-1$ 次 M 变换就不再需要进行控制光路的编码和等待运算液晶的响应了. 因此, 从第一次 M 变换产生输出开始, 每经过一次主光路编码液晶响应 (时间为 t) 就能完成一次 M 变

换. 这种改进后的方式称为方式 1′[11], 它所需光学处理器的操作周期数减少为 $(q+1)/2$. 在实现 40 位 MSD 数据的乘法运算时 (p=q=40), 这 3 种方式的使用情况如表 2 所示.

表 2 光学处理器使用情况

Table 2 Usage of optical processor

	方式 1	方式 2	方式 1′
使用位数	40	1 600	40
使用时间	$80t$	$2t$	$41t$

因为目前 SD11 中的光学处理器的液晶阵列像素位数为 512 位, 暂时无法满足方式2的需求, 所以本研究选用改进后的方式 1′ 来实现 M 变换过程, 这样只需将光学处理器的液晶阵列的 40 位像素数据位重构为 M 变换器, 剩下的 472 个数据位可用于重构 MSD 加法器.

3 和数项求和的实现方案

MSD 数据乘法运算过程的步骤 4 是对 q 个和数项 P_j 求和, 可以采用两两相加迭代的计算方法来实现.

(1) 当和数项个数 q 为奇数时, 用一个 0 值补充为第 q+1 个和数项 P_{q+1}, 使整个和数项的个数为偶数;

(2) 将 P_{2i-1} 与 P_{2i} 进行相加 ($i = 1, 2, \cdots, \lceil q/2 \rceil$), 得到 (1) 中迭代的 $\lceil q/2 \rceil$ 个部分和 $H_i^{(1)}$, 其中 $\lceil q/2 \rceil$ 为上取整函数;

(3) 当第 k 轮迭代得到的部分和个数 $\lceil q/2^k \rceil$ 为奇数时, 用一个 0 值补充为第 $\lceil q/2^k \rceil$+1 个部分和 ${H_{\lceil q/2^k \rceil+1}}^{(k)}$, 使整个部分和的个数为偶数;

(4) 将 ${H_{2i-1}}^{(k)}$ 与 $H_{2i}^{(k)}$ 进行相加 ($i = 1, 2, \cdots, \lceil q/2^{k+1} \rceil$), 得到第 k+1 轮迭代的 $\lceil q/2^{k+1} \rceil$ 个部分和 ${H_i}^{(k+1)}$;

(5) 重复 (3), (4) 步, 直到第 $k = \lceil \log_2 q \rceil$ 轮迭代, 则求和的结果为乘积 C.

在三值光学计算机上进行和数项的两两相加迭代求和时, 有 2 种实现方案: 流水方案和并行方案. 由于用并行方案构造 40 位乘法例程时所需三值光学处理器的数据位数已经远远超出 SD11 设计规划的位数, 而且流水方案比并行方案具有更高的资源利用率和可扩展性, 因此本研究采用流水方案.

流水方案具体操作如下: 在三值光学计算机上同时重构出 MSD 加法器的 T, W, T′, W′ 和 T_2 变换器, 当 M 变换采用方式 1′ 实现时, 每个周期 t_c 可以产生两个 M 变换的结果 (部分积). 将这两个部分积补 0 为和数项后, 送入 MSD 加法器进行 T 和 W 变换. 经过 1 个周期 t_c 后, 第 1 对和数项完成 T 和 W 变换, 进入 T′ 和 W′ 变换时, M 变换器恰好产生第 2 对部分积, 补 0 为和数项送入 MSD 加法器进行 T 和 W 变换; 再经过 1 个周期 t_c 后, 第 1 对和数项进入 T_2 变换, 第 2 对和数项进入 T′ 和 W′ 变换时, M 变换器产生第 3 对部分积, 补 0 为和数项送入 MSD 加法器进行 T 和 W 变换. 以此类推, 经过 $(q+7)/2$ 个周期 t_c 后, 得到第 1 轮迭代的 $\lceil q/2 \rceil$ 个部分和 $H_i^{(1)}$, 此时 M 变换器停止工作, 而 MSD 加法器继续以这种流水计算方式对第 1 轮迭代产生的 $\lceil q/2 \rceil$ 个部分和进行两两相加, 得到第 2 轮迭代的 $\lceil q/4 \rceil$ 个部分和 ${H_i}^{(2)}$, 重复这个过程直到第 $\lceil \log_2 q \rceil$ 轮两两相加产生最后的乘积 C, MSD 加法器停止工作. 这种将每对相邻的和数项或部分和依次送入 MSD 加法器按流水方式进行计算的方案称为流水方案, 其组成结构如图 1 所示.

理论上, 在每一轮迭代过程中, 从输出第 1 个部分和开始, 每经过 1 个周期 t_c 后就可以输出 1 个部分和, 直到这一轮迭代结束. 其操作时序如图 2 所示.

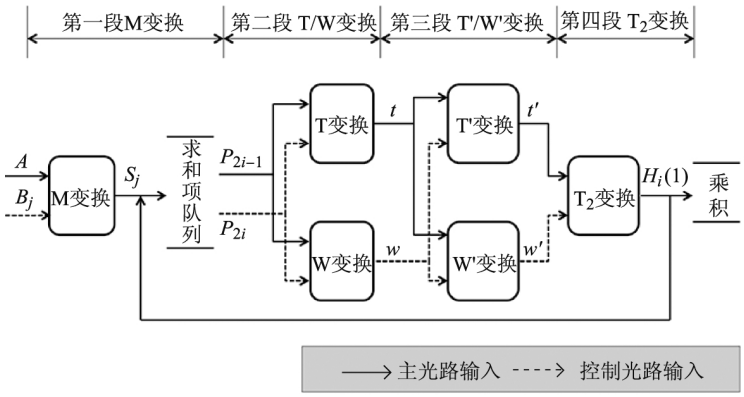

图 1　流水方案结构示意图

Fig. 1　Structure digram of pipeline scheme

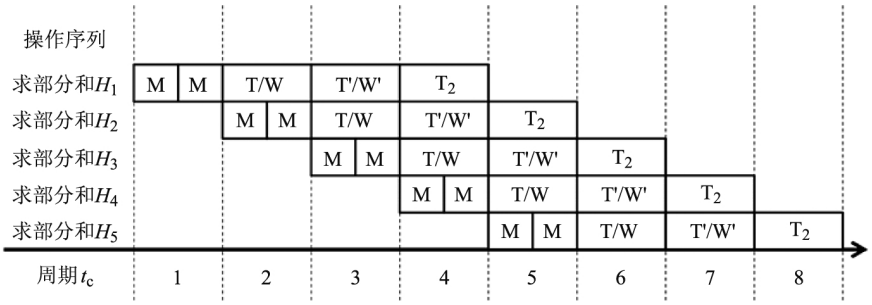

图 2　流水方案操作时序

Fig. 2　Timing digram of pipeline scheme

3.1　流水方案所需光学处理器的位数分析

第 1 轮迭代所需 MSD 加法器的位数 $v^{(1)}$ 等于和数项的位数 $p+q-1$, 即 $v^{(1)}=p+q-1$.

由于 MSD 加法运算的结果位数比操作数位数多 2 位, 因此参与第 2 轮迭代的部分和位数为 $p+q+1$. 第 2 轮迭代所需 MSD 加法器的位数 $v^{(2)}$ 等于部分和的位数 $p+q+1$, 即 $v^{(2)}=p+q+1$.

依次类推, 第 k 轮迭代所需 MSD 加法器的位数 $v^{(k)}$ 等于和数项的位数 $p+q-1$ 加上之前 $k-1$ 轮 MSD 加法增加的位数 $2(k-1)$, 即 $v^{(k)}=p+q+2k-3$.

MSD 数据的乘法运算一共需要进行 $K=\lceil\log_2 q\rceil$ 轮两两相加迭代. 流水方案所需 MSD 加法器的位数 V 取各轮迭代所需位数 $v^{(k)}$ 的最大值:

$$V=\max(v^{(1)},v^{(2)},\cdots,v^{(K)})=v^{(K)}=p+q+2K-3=p+q+2\lceil\log_2 q\rceil-3. \tag{2}$$

因此, 实现 40 位乘法例程时 V=89.

流水方案需要同时重构出 MSD 加法器的 T, W, T′, W′ 和 T_2 变换器, 而一个 V 位的 MSD 加法器分别需要 V 位的 T 和 W 变换器、V+1 位的 T′ 和 W′ 变换器以及 V+2 位的 T_2 变换器来构成. 按流水方案构造 MSD 加法器所需光学处理器的位数 V' 为这 5 个变换器所需

位数之和:

$$V' = 5V + 4 = 5p + 5q + 10\lceil \log_2 q \rceil - 11. \tag{3}$$

因此, 实现 40 位乘法例程时 V'=449.

3.2 流水方案所需光学处理器的操作周期数分析

在流水方案的每一轮迭代过程中, 数据两两相加的过程采用串行流水的方式. MSD 加法器的 5 个独立的逻辑运算部件分为 3 组: {T, W}, {T′, W′} 和 {T_2}, 按顺序构成 3 段式流水线. 同一组的两个逻辑运算部件可以并行操作, 因而每组逻辑运算部件的延时都为 1 个周期 t_c. 每轮迭代开始, 经过 3 个周期 t_c 填满流水线并输出第 1 个相加结果, 此后每个周期 t_c 就可以输出 1 个相加结果, 直至本轮迭代结束. 第 k 轮迭代有 $\lceil q/2^k \rceil$ 对数据需要进行两两相加, 所需光学处理器的操作周期数为 $g^{(k)} = \lceil q/2^k \rceil + 2$.

MSD 数据的乘法运算一共需要进行 $K = \lceil \log_2 q \rceil$ 轮两两相加迭代. 流水方案所需光学处理器的操作周期数 G 为各轮迭代所需操作周期数 $g^{(k)}$ 的总和:

$$G = \sum g^{(k)} = \sum (\lceil q/2^k \rceil + 2), \quad k = 1, 2, \cdots, K. \tag{4}$$

因此, 在实现 40 位乘法例程时 $G = 53$.

由式 (4) 可知, MSD 数据的乘法迭代次数与被乘数 A 的位数 p 没有关系, 它只与乘数 B 的位数 q 成正比. 因此, 在对 2 个位数不同的数据进行乘法运算时, 应该选择位数较少的数据作为乘数 B, 以减少 MSD 数据乘法迭代次数.

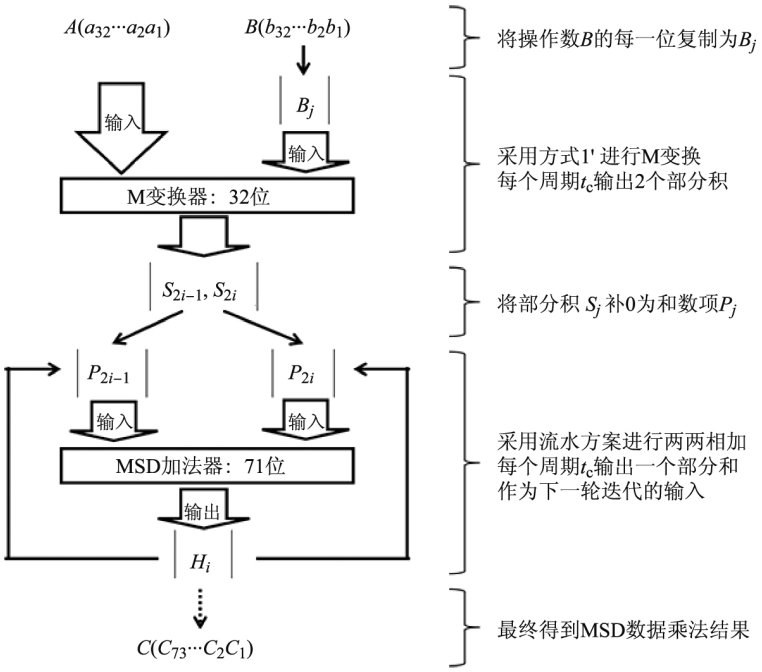

图 3 MSD数据的乘法例程结构

Fig. 3 Program flow diagram of MSD multiplication routine

4 MSD数据的乘法例程

根据前述的 M 方案实现方案讨论的结果, MSD 数据乘法过程的步骤 2 中 M 变换采用方式 1′ 实现; 根据第 3 节讨论的结果, MSD 数据乘法过程的步骤 4 中的两两相加迭代过程采用

流水方案实现; MSD 数据乘法过程的步骤 1 和步骤 3 中数据的复制和补 0 过程是为了生成三值光学计算机的操作数据编码, 由上位机监控系统中的任务处理模块和下位机解码控制模块完成.

三值光学计算机 SD11 中, 光学处理器的液晶阵列像素位数为 512 位, 按流水方案最多可以构造出 42 位的乘法例程, 因此可以满足本研究的 40 位乘法例程所需. 在 40 位乘法例程中被乘数 A 和乘数 B 的长度都为 40 位, 即 $p=40$, $q=40$. 乘法例程的组成模块按“SZG_$\star$”形式命名, 其中“SZG”是“三值光”的汉语拼音首字母, “$\star$”是具体模块名称的通配符. 例程的具体实现步骤如下.

步骤 1　上位机监控系统中任务处理模块 SZG_RWCL (RWCL 是“任务处理”的汉语拼音首字母) 生成并发送重构指令至下位机运算控制模块 SZG_YSKZ (YSKZ 是“运算控制”的汉语拼音首字母), SZG_YSKZ 将重构指令送入三值光学计算机的重构指令锁存器. 将数据位地址 0∼39 的 $U(U=40)$ 个光学处理器数据位重构成 40 位的 M 变换器; 将数据位地址 40∼488 的 $V(V=449)$ 个光学处理器数据位重构成 89 位的 MSD 加法器, 其中地址 40∼128 为 T 变换器, 129∼217 为 W 变换器, 218∼307 为 T′ 变换器, 308∼397 为 W′ 变换器, 398∼488 为 T_2 变换器. U 和 V 分别根据式 (5) 和 (6) 确定:

$$U=p=40, \tag{5}$$

$$V=5p+5q+10\lceil\log_2 q\rceil-11=449. \tag{6}$$

步骤 2　SZG_RWCL 将乘数 B 的每一位都复制成 $p(p=40)$ 位, 存入 QB 数据队列中. 根据被乘数 A 和 QB 数据队列分别生成 M 变换器的控制光路编码信息和主光路编码信息, 发送给 SZG_YSKZ. QB 数据队列如表 3 所示.

表 3　QB 数据队列

Table 3　Data queue-QB

	第 p 位	⋯	第 2 位	第 1 位
数据 1	b_1	⋯	b_1	b_1
数据 2	b_2	⋯	b_2	b_2
⋯	⋯	⋯	⋯	⋯
数据 q	b_q	⋯	b_q	b_q

步骤 3　SZG_YSKZ 将主光路和控制光路的编码信息分别送入 M 变换器的主光路和控制光路编码锁存器, 执行 M 变换产生部分积. 下位机解码控制模块 SZG_JMKZ (JMKZ 是“解码控制”的汉语拼音首字母) 将部分积补 0 为 $p+q-1=79$ 位的和数项, 存入 QS 数据队列中. 补 0 规则为

$$P_j=S_j\times 2^{j-1}. \tag{7}$$

QS 数据队列如表 4 所示.

表 4　QS 数据队列

Table 4　Data queue-QS

	第 $p+q-1$ 位	⋯	第 2 位	第 1 位
和数项 P_1	0	⋯	s_{12}	s_{11}
和数项 P_2	0	⋯	s_{21}	0
⋯	⋯	⋯	⋯	⋯
和数项 P_q	s_{qp}	⋯	0	0

步骤 4　重复执行步骤 2∼3 共 40 次, 结果将产生 $q=40$ 个部分积 $S_1, S_2, \cdots, S_q$. 此时 M 变换器停止工作.

步骤 5　在开始执行第 k=1 轮的两两相加之前, 为了保证 QS 数据队列中奇数项和数项与偶数项和数项的个数相同, 如果 q 为奇数, SZG_JMKZ 在 QS 数据队列后面插入一个值为 0 的偶数项和数项 P_{q+1}. 由于 40 位乘法例程中 q=40 为偶数, 不会执行该步操作.

步骤 6　SZG_JMKZ 读取 QS 数据队列中奇数项和数项, 生成 MSD 加法器的主光路编码信息; 读取偶数项和数项, 生成 MSD 加法器的控制光路编码信息. SZG_JMKZ 将主光路和控制光路编码信息发送给 SZG_YSKZ. SZG_YSKZ 将编码信息分别送入 MSD 加法器的主光路和控制光路编码锁存器, 执行 MSD 加法产生部分和. SZG_JMKZ 将部分和存入 $\mathrm{QH}^{(1)}$ 数据队列中. $\mathrm{QH}^{(k)}$ 数据队列如表 5 所示, 其中 $k=1,2,\cdots,\lceil\log_2 q\rceil$.

表 5　$\mathrm{QH}^{(k)}$数据队列

Table 5　Data queue-QH(k)

	第$j^{(k)}$位	$\cdots$	第2位	第1位
部分和$H_1^{(k)}$	h_{1j}	$\cdots$	h_{12}	h_{11}
部分和$H_2^{(k)}$	h_{2j}	$\cdots$	h_{22}	h_{21}
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
部分和$H_n^{(k)}$	h_{nj}	$\cdots$	h_{n2}	h_{n1}

步骤 7　重复执行步骤 6 共 20 次, 结果将产生第 k=1 轮两两相加的 $n^{(1)}$=20 个部分和 $H_1^{(1)}, H_2^{(1)}, \cdots, H_n^{(1)}$, 每个部分和的长度为 $j^{(1)}$=81 位. $n^{(k)}$ 和 $j^{(k)}$ 分别根据式 (8) 和 (9) 确定:

$$n^{(k)}=\lceil q/2^k\rceil, k=1,2,\cdots,\lceil\log_2 q\rceil, \tag{8}$$

$$j^{(k)}=p+q+2k-1, k=1,2,\cdots,\lceil\log_2 q\rceil. \tag{9}$$

步骤 8　在开始执行第 $k+1$ 轮两两相加之前, 为保证上一轮两两相加得到的 $\mathrm{QH}^{(k)}$ 数据队列中奇数项部分和与偶数项部分和的个数相同, 如果 $n^{(k)}$ 为奇数, SZG_JMKZ 在 $\mathrm{QH}^{(k)}$ 数据队列的后面插入一个值为 0 的偶数项部分和 $H_{n+1}^{(k)}$.

步骤 9　SZG_JMKZ 读取 $\mathrm{QH}^{(k)}$ 数据队列中奇数项部分和, 生成 MSD 加法器的主光路编码信息; 读取偶数项部分和, 生成 MSD 加法器的控制光路编码信息. SZG_JMKZ 将主光路和控制光路编码信息发送给 SZG_YSKZ. SZG_YSKZ 将编码信息分别送入 MSD 加法器的主光路和控制光路编码锁存器, 执行 MSD 加法产生部分和. SZG_JMKZ 将部分和存入 $\mathrm{QH}^{(k+1)}$ 数据队列中.

步骤 10　重复执行步骤 9 步 $n^{(k+1)}$ 次, 将产生第 $k+1$ 轮两两相加的 $n^{(k+1)}$ 个部分和 $H_1^{(k)}, H_2^{(k)}, \cdots, H_n^{(k)}$, 每个部分和的长度为 $j^{(k+1)}$ 位.

步骤 11　重复执行步骤 8∼10, 直到完成第 $K(K=6)$ 轮两两相加, 产生最终乘积 C, MSD 加法器停止工作. 乘积 C 的长度为 $j^{(K)}$=91 位, SZG_JMKZ 将乘积 C 返回给 SZG_RWCL, K 由式 (10) 确定:

$$K=\lceil\log_2 q\rceil=6. \tag{10}$$

5　模拟实验

由于 SD11 还在建设中, 用数台 PC 机搭建了一个三值光学计算机模拟实验平台, 所有研

发出来的软件都首先在这个平台上进行实验, 用以验证相关理论和技术的正确性. 模拟环境分为前台机模拟和后台机模拟两个部分, 分别模拟用户使用环境和后台运算处理环境. 本研究主要通过运行后台运算处理环境的模拟程序来验证乘法例程功能的正确性.

5.1　软件模拟程序

模拟程序中与乘法例程各功能相对应的函数模块和数据类型实现代码如下:

(1) 程序中使用二维映射表 T[a][b], W[a][b], $\mathrm{T_1}$[a][b], $\mathrm{W_1}$[a][b], $\mathrm{T_2}$[a][b] 和 M[a][b], 分别存放乘法例程中的 T, W, T′, W′, $\mathrm{T_2}$ 和 M 变换的真值表, 提供给三值逻辑运算器来模拟函数使用.

```
1   typedef map<char, map<char, char≫TRUTH_TABLE;
2   TRUTH_TABLE T, W, T1, W1, T2, M; //存放真值表的二维映射表
```

(2) 函数 Logical_Operator(a, b, n, &table) 模拟操作数为 a 和 b, 真值表为 table 的 n 位三值逻辑运算器. 将二维映射表作为参数传给该函数后, 分别模拟乘法例程所使用的 T 变换器、W 变换器、T′ 变换器、W′ 变换器、$\mathrm{T_2}$ 变换器以及 M 变换器的功能.

```
4   TERNARY_DATA Logical_Operator(TERNARY_DATA a, TERNARY_DATA b,
5                                 DATA_SIZE n, TRUTH_TABLE & table){ //模拟
                                  三值逻辑运算器
6       TERNARY_DATA c(n, '0');
7       for (int i=0; i!=n; ++i){
8           c[i]=table[a[i]][b[i]];
9       }
10      return c;
11  }
```

(3) 用 Logical_Operator 函数模拟 T, W, T′, W′, $\mathrm{T_2}$ 变换器来构建 V 位的 MSD 加法器模拟函数 MSD_Adder(a, b, V).

```
13  TERNARY_DATA MSD_Adder(TERNARY_DATA a, TERNARY_DATA b,DATA_SIZE
    V){//模拟 MSD 加法器
14      TERNARY_DATA x(V, '0'), y(V, '0');
15      for (int i=0; i!=V; ++i){
16          if ((int)a.length()-1-i>=0)x[V-1-i]=a[a.length()-1-i];
17          if ((int)b.length()-1-i>=0)y[V-1-i]=b[b.length()-1-i];
18      }
19      TERNARY_DATA t=Logical_Operator(x, y, V, T)+'0';
20      TERNARY_DATA w='0'+Logical_Operator(x, y, V, W);
21      TERNARY_DATA t1=Logical_Operator(t, w, V+1, T1)+'0';
22      TERNARY_DATA w1='0'+Logical_Operator(t, w, V+1, W1);
23      TERNARY_DATA c=Logical_Operator(t1, w1, V+2, T2);
24      return c;
25  }
```

(4) 在 Logical_Operator 函数模拟的M变换器和 MSD_Adder 函数模拟的 MSD 加法器的

基础上, 按照乘法例程的实现步骤, 编写模拟函数 Multiplication_Routine(A, B) 来完成 MSD 数据 A 和 B 的乘法运算.

```
TERNARY_DATA Multiplication_Routine(TERNARY_DATA A, TERNARY_DATA B)
{//模拟乘法例程
      TERNARY_DATA QB[q];
      for (int i=0; i!=q; ++i){ //将乘数 B 的每一位都复制成 p 位, 存入 QB 数据队列
            QB[i].insert(0, p, B[q-1-i]);
      }
      TERNARY_DATA QS[q+1];
      for (int i=0; i!=q; ++i){//将 A 和 QB[i] 送入 M 变换器进行运算, 结果补零为和数项,
存入 QS 数据队列
            QS[i]=Logical_Operator(QB[i], A, p, M);
            QS [i].insert(QS[i].length(), i, '0');
      }
      TERNARY_DATA *QH=QS;
      for (int k=1; k<=ceil(log2(q)); ++k){//和数项或部分和两两相加迭代, 结果存入 QH
      数据队列
            if ((int)ceil(q/pow(2, k-1))%2==1)//求和项个数为奇数时, 增加一个值为0的偶数项
                  QH[(int)ceil(q/pow(2, k-1))]="0";
            for (int n=0; n<=ceil(q/pow(2, k))-1; ++n){//第 k 轮两两相加迭代
                  QH[n]=MSD_Adder(QH[2*n], QH[2*n+1], p+q+2*ceil(log2(q)-3);
            }
      }
      return QH[0];
}
```

5.2 模拟实验用例

因为 40 位乘法例程的操作数据为 2 个 40 位的 MSD 数据, 输入的组合有 3^{80}种, 所以无法采用穷举方式来进行乘法例程的功能检验. 这里将特殊数据和随机数据相结合, 作为乘法例程的输入数据, 运算结果与理论值相比较, 以验证乘法例程功能的正确性.

(1) 特殊数据测试用例. 选用数据位全为 $\bar{1}$、全为 0、全为 1、$\bar{1}0$ 交替出现、01 交替出现、$1\bar{1}$ 交替出现和 $\bar{1}01$ 交替出现的 MSD 数据作为操作数, 将这 7 种操作数组合成 49 组输入数据, 作为手工选择的边界测试用例. 例如:

$A=\bar{1}$,

$B=\bar{1}01\bar{1}01\bar{1}01\bar{1}01\ \bar{1}01\bar{1}01\bar{1}01\bar{1}01\bar{1}01\bar{1}01\bar{1}01\bar{1}01\bar{1}$.

(2) 随机数据测试用例. 使用伪随机函数自动生成 10 000 组测试用例, 例如:

$A=\bar{1}\bar{1}110001\bar{1}1\bar{1}\bar{1}\bar{1}0\bar{1}\bar{1}010\bar{1}10\bar{1}\bar{1}01\bar{1}00\bar{1}1111\bar{1}0\bar{1}\bar{1}\bar{1}0$,

$B=1\bar{1}000\bar{1}\bar{1}1\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}010\bar{1}11\bar{1}\bar{1}0\bar{1}0\bar{1}\bar{1}0010\bar{1}111001010$.

5.3 模拟实验过程

本研究严格按照该乘法例程的用户使用方法, 对每组实验用例逐一进行测试.

在用户使用该乘法例程时, 首先需要打开用户运算请求输入界面[9], 分别输入每组乘法运算请求的操作数 A、操作数 B 以及整数乘法运算符 $\star$, 按“确定”键后, 系统将自动生成相应的

运算请求数据文件 (*.SZG 文件); 然后在用户编写的 C 语言程序中使用 SZGCYY-GH2012 扩充指令集[9], 或者在 MPI 程序中使用 SZGMPI-ZQ2012 扩展机制[10], 将运算请求文件发送到三值光学计算机; 三值光学计算机监控系统接收到用户运算请求文件后, 调用本研究实现的乘法例程来完成用户的乘法运算任务, 并将乘法运算结果的 SZG 数据文件返回给用户; 用户可以在 C 语言程序或 MPI 程序中对三值光学计算机返回的乘法运算结果做进一步的处理. 而其中监控系统如何调用该例程以及例程的具体实现细节对于用户来说是不可见的, 即用户无需了解例程本身的实现过程就可以方便地使用该乘法例程来完成其运算需求.

因此, 与用户使用场景相一致的模拟实验过程如下:

(1) 用运算请求输入界面生成实验用例的运算请求文件;

(2) 在 C 语言程序和 MPI 程序中使用扩展机制发送请求文件给乘法例程模拟程序;

(3) 乘法例程模拟程序返回乘积结果文件给 C 语言程序和 MPI 程序;

(4) 在 C 语言程序和 MPI 程序中打印输出例程返回的乘积结果, 并与理论值进行比较, 以判定测试结果的正确性.

5.4　模拟实验结果

经过对 49 组特殊数据测试用例和 10 000 组随机测试用例的模拟计算, 得到的 10 049 个模拟实验结果与理论值基本一致, 这表明本研究提出的 40 位 MSD 数据的乘法例程功能正确. 限于篇幅, 表 6 中只展示该模拟实验的 6 组结果.

6　与电子计算机同类乘法器对比

本研究实现的三值光学计算机乘法例程采用了移位相加乘法器原理, 但是其具体的实现技术和运算性能与电子计算机的同类型乘法器 (如移位相加乘法器) 存在明显的不同.

(1) 在生成部分积时, 电子计算机移位相加乘法器一般使用位与逻辑电路或 Booth 编码电路产生部分积; 而三值光学计算机乘法例程则使用三值逻辑变换中的 M 运算来产生部分积.

(2) 在部分积求和时, 电子计算机移位相加乘法器采用二进制加法, 因此无法避免每次相加时的进位延时问题, 并且随着操作数位数的增加, 每次加法的进位延时会线性增加; 而三值光学计算机乘法例程采用无进位的 MSD 数据加法, 无论操作数位数有多少, 每次相加所需的时间都相同, 为 MSD 加法器的运算时间.

对于加法过程带给乘法器/例程的延时问题, 可以有如下的具体对比. 电子计算机移位相加乘法器进行 n 位数据的乘法时, 需要完成 n 次二进制加法, 而每次加法所需的时间为数据位数 n 与每位进位延时 t_{e} 的函数, 如在串行进位加法器中, 一次加法运算的总延时为 $n \times t_{\mathrm{e}}$, 因此通过 n 次加法来完成 n 位数据乘法所需的总时间为 $T_{\mathrm{e}} \approx n^2 \times t_{\mathrm{e}}$.

而在本研究实现的三值光学计算机乘法例程中, 在进行 n 位数据的乘法时, 其 MSD 加法的次数不会超过 $2n$ 次, 且每次 MSD 加法所需的时间与数据位数 n 无关, 仅与光学器件的延时 t_{o} 有关. 即使不采用流水线, 最多也只需 $6n$ 次的三值逻辑变换, 其所需的总时间也不会超过 $T_{\mathrm{o}} \approx 6n \times t_{\mathrm{o}}$.

对比 $T_{\mathrm{e}} \approx n^2 \times t_{\mathrm{e}}$ 和 $T_{\mathrm{o}} \approx 6n \times t_{\mathrm{o}}$, 可以发现前者是 $O(n^2)$ 时间复杂度的, 而后者是 $O(n)$ 时间复杂度, 因而三值光学计算机乘法例程在处理大数乘法时优势明显. 例如进行 40 位数据的乘法时, $T_{\mathrm{e}} \approx 1\,600 t_{\mathrm{e}}$, 而 $T_{\mathrm{o}} \approx 240 \times t_{\mathrm{o}}$. 假设电子计算机移位加法器每位的延时 t_{e} 为 0.1 ns, 三值逻辑变换的光学器件延时 t_{o} 为 1 ns, 则 T_{e} ≈160 ns, 而 T_{o} ≈240 ns, 二者速度相差不多; 但是如果进行 1 000 位数据的乘法, 通过同样方式推算可得 T_{e} ≈100 μs, T_{o} ≈6 μs, 此时光学计算机乘法例程比电子乘法器约快 2 个数量级.

表 6 模拟实验结果

Table 6 Simulation results

		MSD数字	十进制数	测试结果
第 1 组	输入 A	$\bar{1}\bar{1}110001\bar{1}1\bar{1}\bar{1}\bar{1}0\bar{1}\bar{1}010\bar{1}10\bar{1}\bar{1}01\bar{1}00\bar{1}1111\bar{1}0\bar{1}\bar{1}\bar{1}0$	-616240439406	正确
	输入 B	$1\bar{1}000\bar{1}\bar{1}1\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}010\bar{1}11\bar{1}\bar{1}0\bar{1}0\bar{1}\bar{1}0010\bar{1}111001010$	249272027082	
	输出值	$000000000000001\bar{0}0000\bar{1}000\bar{1}0010\bar{1}0\bar{1}0100011\bar{0}10\bar{1}01\bar{1}1\bar{1}$	-153611503500	
		$\bar{1}1\bar{1}10\bar{1}1\bar{1}\bar{1}1001\bar{1}001\bar{1}0100001\bar{0}\bar{1}\bar{1}\bar{1}\bar{1}11\bar{1}00\bar{1}1\bar{1}\bar{1}\bar{1}\bar{1}00$	636011993292	
	理论值		-153611503500	
			636011993292	
第 2 组	输入 A	$0011\bar{1}000\bar{1}010\bar{1}1\bar{1}\bar{1}0\bar{1}\bar{1}\bar{1}1\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}11\bar{1}\bar{1}\bar{1}\bar{1}011\bar{1}11$	170096384343	正确
	输入 B	$1\bar{1}010\bar{1}\bar{1}\bar{1}00\bar{1}\bar{1}\bar{1}11\bar{1}01110$	312928066990	
	输出值	$000000000000000\bar{1}0\bar{1}\bar{1}11\bar{1}0010\bar{1}\bar{1}100000\bar{1}0000\bar{1}1\bar{1}010\bar{1}00$	532279327544	
		$11\bar{1}010000\bar{1}100\bar{1}1000000\bar{1}11\bar{1}01\bar{1}011\bar{1}010001\bar{1}00010$	43091137570	
	理论值		532279327544	
			43091137570	
第 3 组	输入 A	$10000101010\bar{1}0\bar{1}\bar{1}01\bar{1}01\bar{1}\bar{1}\bar{1}1100\bar{1}1\bar{1}\bar{1}01\bar{1}10010000$	571575486864	正确
	输入 B	$110110\bar{1}1000\bar{1}\bar{1}0\bar{1}1001\bar{1}0\bar{1}\bar{1}01\bar{1}\bar{1}\bar{1}1101\bar{1}10\bar{1}0\bar{1}10\bar{1}$	922999208283	
	输出值	$000000000001\bar{1}0\bar{1}10000\bar{1}100\bar{1}00\bar{1}\bar{1}01\bar{1}01\bar{1}0110010\bar{1}0001$	527563721849	
		$00\bar{1}00\bar{1}\bar{1}1\bar{1}1010\bar{1}\bar{1}1000000\bar{1}01000\bar{1}10\bar{1}10\bar{1}0010\bar{1}0000$	442266494512	
	理论值		527563721849	
			442266494512	
第 4 组	输入 A	$\bar{1}1\bar{1}100001\bar{1}10\bar{1}\bar{1}01\bar{1}1\bar{1}001\bar{1}\bar{1}1\bar{1}0\bar{1}\bar{1}00\bar{1}0\bar{1}\bar{1}11\bar{1}10$	-342177536330	正确
	输入 B	$\bar{1}\bar{1}\bar{1}0\bar{1}00\bar{1}1101\bar{1}\bar{1}1011111010100\bar{1}\bar{1}\bar{1}11\bar{1}1\bar{1}\bar{1}0111\bar{1}$	-997389080939	
	输出值	$000000000000010010001\bar{1}00011\bar{1}00010\bar{1}0100\bar{1}00\bar{1}\bar{1}101\bar{1}0$	341284138478	
		$001\bar{1}00001\bar{1}0000001001000\bar{1}0100000\bar{1}0\bar{1}0000\bar{1}100\bar{1}0$	149983013870	
	理论值		341284138478	
			149983013870	
第 5 组	输入 A	$\bar{1}011\bar{1}0011\bar{1}10111010\bar{1}101\bar{1}00100100\bar{1}0\bar{1}\bar{1}101\bar{1}1$	-371809171789	正确
	输入 B	$11\bar{1}10\bar{1}0100\bar{1}00\bar{1}0110011001110010\bar{1}\bar{1}0\bar{1}10\bar{1}10\bar{1}$	742452217051	
	输出值	$000000000000\bar{1}0010\bar{1}\bar{1}10010\bar{1}\bar{1}11\bar{1}001\bar{1}0100100011\bar{1}\bar{1}010$	-276050543914	
		$\bar{1}00001\bar{1}0\bar{1}010\bar{1}10000100\bar{1}100\bar{1}11\bar{1}001\bar{1}00\bar{1}00100001\bar{1}$	639173974239	
	理论值		-27605054391	
			4639173974239	
第 6 组	输入 A	$10010011011\bar{1}0\bar{1}\bar{1}1\bar{1}\bar{1}110\bar{1}0\bar{1}0010\bar{1}\bar{1}0\bar{1}1101011$	632613603819	正确
	输入 B	$0\bar{1}10\bar{1}\bar{1}110\bar{1}\bar{1}11\bar{1}\bar{1}10100\bar{1}00\bar{1}\bar{1}11\bar{1}\bar{1}\bar{1}1\bar{1}\bar{1}1\bar{1}\bar{1}0011$	-177314805101	
	输出值	$000000000000\bar{1}0100001000\bar{1}101\bar{1}0100011\bar{1}\bar{1}010\bar{1}\bar{1}100\bar{1}$	-11217175786	
		$1\bar{1}010\bar{1}100\bar{1}\bar{1}11\bar{1}0\bar{1}1010\bar{1}0100100\bar{1}0100000000\bar{1}0001$	5407214280719	
	理论值		-11217175786	
			5407214280719	

7 结 束 语

本研究的三值光学计算机 40 位 MSD 数据乘法例程实际应用性强, 其揭示的乘法器原理在计算大数值的乘法时具有速度超过一般电子计算机的优势. 本研究建立的例程利用了三值光学计算机的 M 变换器和 MSD 加法器等的优势, 并针对 SD11 千位实验系统进行了具体实

现. 经过对几种实现方案的对比、分析和权衡, 本例程中采用了单 M 运算器方案, 以及流水计算“和数项”相加迭代方案, 并结合三值光学计算机监控系统的设计规划建立了具体的操作例程, 给出了上层用户使用本例程的方法. 同时, 本例程为下一步实现三值光学计算机向量矩阵乘法例程奠定了基础.

参考文献:

[1] 金翊, 王宏健, 沈云付, 等. 可重构三值光学处理器的原理、基本结构和实现 [J]. 中国科学: 信息科学, 2012, 42(6): 012.

[2] 金翊, 何华灿, 吕养天. 三值光计算机的基本原理 [J]. 中国科学: E 辑, 2003, 33(2): 111-115.

[3] 严军勇, 金翊, 左开中. 无进 (借) 位运算器的降值设计理论及其在三值光计算机中的应用 [J]. 中国科学: E 辑, 2008, 38(12): 2112-2122.

[4] 金翊, 沈云付, 彭俊杰, 等. 三值光学计算机中MSD 加法器的理论和结构 [J]. 中国科学: 信息科学, 2011, 41(5): 541-551.

[5] Waser S. High-speed monolithic multipliers for real-time digital signal processing [J]. Computer, 1978, 11(10): 19-29.

[6] Ma G K, Taylor F J. Multiplier policies for digital signal processing [J]. ASSP Magazine, IEEE, 1990, 7(1): 6-20.

[7] 于敦山, 沈绪榜. 32 位定/浮点乘法器设计 [J]. 半导体学报, 2001, 22(1): 91-95.

[8] 葛亮, 唐志敏. 一种支持无符号数的流水线乘法器 [J]. 微电子学与计算机, 2002, 19(10): 17-19.

[9] 高桓, 金翊, 宋凯. 针对三值光学计算机的 C 语言扩展 [J]. 上海大学学报: 自然科学版, 2013, 19(3): 280-285.

[10] 张茜, 金翊, 宋凯, 等. 在超算集群中使用三值光学计算机的 MPI 编程技术 [J]. 上海大学学报: 自然科学版, 2014, 20(2): 180-189.

[11] 金翊. 三值光计算机高数据宽度的管理策略 [J]. 上海大学学报: 自然科学版, 2007, 13(5): 519-523.