三值光学计算机中 SJ-MSD 加法器的设计与实现

江家宝[1,2],张晓峰[3],沈云付[1],欧阳山[1],周时强[1],彭俊杰[1],刘跃军[1],金 翊[1]
(1. 上海大学计算机科学与工程学院,上海 200444; 2. 巢湖学院信息工程学院,安徽巢湖 238000;
3. 中国航天科学与工业集团第 25 研究所毫米波遥感技术国家重点实验室,北京 100039)

摘 要: 本文介绍三值光学计算机的一种新型并行加法器—SJ-MSD 加法器的设计与实现. 介绍判断一组三值逻辑变换能够实现并行无连续进位二进制 MSD 数加法的充分条件(沈氏充分性定理). 给出了构成 SJ-MSD 加法器的五个三值逻辑变换:S_1、S_2、J_1、J_2和 J_3(简称 SJ 变换),及其操作规则(简称 SJ 规则),并依据沈氏充分性定理推证了 SJ 变换和 SJ 规则构成 MSD 并行加法器的可靠性. 尔后又详细阐述了 SJ-MSD 加法器在三值光学计算机原型系统 SD16 上的设计方案和实现方法,并阐述了流水计算和多数据共享 SJ-MSD 加法器的实现方法. 文中最后详细介绍了对 SJ-MSD 加法器的测试实验. 与先期的 TW-MSD 加法器相比,SJ-MSD 加法器减少占用处理器位数约 25%,有效提高了三值光学处理器的使用率.

关键词: 三值光学计算机; 重构; 像素; 处理器位; 共享流水加法器
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2021)02-0275-11
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.12263/DZXB.20180572

Design and Implementation of SJ-MSD Adder in Ternary Optical Computer

JIANG Jia-bao[1,2], ZHANG Xiao-feng[3], SHEN Yun-fu[1], OUYANG Shan[1], ZHOU Shi-qiang[1], PENG Jun-jie[1], LIU Yue-jun[1], JIN Yi[1]
(1. *School of Computer Engineering and Science, University of Shanghai, Shanghai* 200444, *China*;
2. *College of Information Engineering, University of Chaohu, Chaohu, Anhui* 238000, *China*;
3. *State Key Laboratory of Millimeter Wave Remote Sensing Technology, 25th Institute of China Aerospace Science and Industry Group, Beijing* 100039, *China*)

Abstract: In this paper the design and implementation of a new parallel adder, SJ-MSD adder, in a ternary optical computer (i. e. TOC) are proposed. It is introduced that a sufficient conditions (Shen's sufficient theorem) which judging that a group of ternary logical transformations is able to build a MSD (Modified Signed Digit) addition. Five logical transformations S_1, S_2, J_1, J_2 and J_3 (i. e. SJ transformation) and a parallel carry-free MSD addition rule (i. e. SJ rule) are proposed to construct SJ-MSD adder. Meanwhile according to Shen's sufficient theorem, the reliability of the SJ transformation and SJ rule used to construct MSD parallel adder is proved. Subsequently, in the ternary optical computer prototype system SD16, the design and implement of SJ-MSD adder are discussed in detail. At the same time the implementation methods of pipeline addition and sharing one SJ-MSD adder for multiple data are expatiated. Some experiments of SJ-MSD adder is introduced. Compared with the previous TW-MSD adder, the SJ-MSD adder occupies processor bits less about 25%. So the optical processor' efficiency is effectively improved.

Key words: TOC(ternary optical computer); reconfiguration; pixel; processor bit; shareable pipelined adder

收稿日期:2018-06-20;修回日期:2019-04-26;责任编辑:王天慧
基金项目:安徽省教育厅自然科学基金(No. KJ2017A452);上海市科研计划专项基金(No. 15700500400);国家自然科学基金(No. 61572305, No. 61073049)

1 引言

加法器是计算机中进行数值计算的最基本部件,其速度决定了计算机进行数值计算的耗时.然而,在常规加法运算中,存在着从最低位到最高位的进位传递过程,这个过程造成了加法器的进位延迟,严重制约着计算机的数值计算速度.电子计算机为了克服进位延迟,采用带超前进位部件的二进制加法器,但超前进位部件的复杂结构,使得这种加法器数据位数很少超过5位.对于位数较多的加法器,目前的电子计算机采用将多个小超前进位加法器分成两级的结构,或接受加法器的串行进位延时.本文给出的SJ-MSD加法器与其它MSD并行加法器一样都没有进位延时问题,也没有先行进位部件的复杂结构,因而这种并行加法器是处理器硬件发展的一个重要方向.

如何克服进位延时一直是加法器研究的热点.早在1961年,Avizienis就提出了一种能够消除连续进位延时的加法技术—MSD(Modified Signed Digit)数字加法器.这个技术的核心是采用有符号冗余数字[1],依靠合理选择计算结果中各个位的表达符,将进位值的传播限定在两位内,从而消除进位值的长传播,实现所有数据位并行相加.1986年,Drake等人在采用符号替换技术的光学计算研究中首次将MSD数加法规则改为符号替换规则[2],称为T、W、T′和W′替换,并用这四个替换规则,分三次替换完成MSD数加法运算.2009年,金翊等在三值光学处理器上构造了这四个符号替换规则对应的三值逻辑运算器,并通过三步连续运算完成对所有数据位并行计算的加法器[3],本文称之为:TW-MSD加法器.

MSD加法器的并行特性吸引了该领域学者的长期注意,迄今已经提出了十数种MSD加法器结构,按其运算步骤,它们被分为三步式[2~6]、两步式[7~11]和一步式[12~14]MSD加法器结构.2009年在实验上实现TW-MSD加法器结构后,2011年沈云付教授为三值光学计算机发明了限定输入符号的一步式MSD加法器结构[14],2012年沈云付教授又发明了限制输入符号的MSD连加器结构[15].2017年3月18日,上海大学光学计算机实验室在图1所示的三值光学处理器SD16的模块1上构建出36位并行TW-MSD加法器[16~18],开启了MSD并行加法器进入实际应用的序幕.

在SD16上实现的第一个光学加法器是TW-MSD型,但并不排斥用其它三值逻辑运算器来构造效率更高的MSD加法器.2016年10~12月,作者综合前期研究成果,给出了一种MSD加法器新结构,称为SJ-MSD加法器,它比TW-MSD加法器节约三值光学处理器硬件资源25%以上.2017年6月作者首次在SD16的一个模块上实现了46位的SJ-MSD加法器.目前的SD16可加装64个模块,可构造出超过3071位的SJ-MSD加法器.本文详细介绍这一新型MSD加法器的原理与结构.

2 MSD数简介

MSD并行加法器的基础是对二进制数值用三个符号所做的冗余数表达.在计算机内部使用这种冗余表达的数值,能够消除加法器的进位延时,从而加快数值计算.

2.1 MSD数表示

MSD数是用0、1和u三个符号表达的二进制数,其中u的取值为-1.任意数值的十进制数表达和MSD数表达的转换关系如式(1)所示[18,19].

$$[a.b]_{10}=\sum_{i=0}^{m}x_i\times 2^i+\sum_{j=1}^{n}y_j\times 2^{-j} \tag{1}$$

式(1)中:$x_i,y_j\in\{u,0,1\}$,且$i=0,1,2,\cdots,m$;$j=1,2,\cdots,n$,值域中的u代表数值-1;由此可知,MSD数的三值变换是三值逻辑运算的一个子集—限定自变量x和y的值域为$\{u,0,1\}$.右侧的MSD数表达有$m+1$位整数和n位小数.左侧十进制数表达有a位整数和b位小数.

2.2 MSD数特征

由于MSD数使用了冗余符号,与常规二进制数表示相比,它有如下特点[19]:

(1)除了数据0值之外,任意数值都有无数个MSD数表达,如

例1 $[-24]_{10}=[u1u000]_M=[u11u000]_M=\cdots\cdots$;$[15.5]_{10}=[1111.1]_M=[1u00u1.1]_M=\cdots\cdots$

(2)一个数值的MSD数逐位取反(0不变,1和u互相转换)得到该数值相反数的MSD数.

例2 $[-22.25]_{10}=[uu010.0u]_M=[u1u01u.11]_M$;$[22.25]_{10}=[110u0.01]_M=[1u10u1.uu]_M$

(3)常规二进制数是MSD数的不使用u符号的子集.

2.3 沈氏充分条件

由于每个数值都有很多种MSD数表达,当两个MSD数相加时,就有可能为和值找到一种表达,其特征在于每一位的进位值都不形成进位传播,形成这一和值表达的过程必定是针对各个数据位的逻辑变换过程.这意味着相应的加法器就是对各数据位并行实现相同特定变换的一组逻辑变换器.2017年初,沈云付教授在数学上建立并证明了判断一组三值逻辑变换能够并行实现MSD数加法的充分条件—沈氏充分条件[21],具体如下.

若对任意两个MSD数:$a=a_{n-1}\cdots a_1a_0$,$b=b_{n-1}$

$\cdots b_1 b_0$,相继进行 Y、F、Y′、F′和 S 变换,并满足下列三条件,则 MSD 数 $s = s_{n+1}\, s_n \cdots s_1 s_0$ 是 a 与 b 的和值. 以示区别,本文中补加的 0 用符号 ϕ 表示.

条件 1　$a_i + b_i = y_{i+1} \times 2 + f_i$; 其中,$i = 0,1,\cdots,n-1$; $y_0 = f_n = \phi$.

条件 2　$y_i + f_i = y'_{i+1} \times 2 + f'_i$; 其中,$i = 0,1,\cdots,n$; $y'_0 = f'_{n+1} = \phi$.

条件 3　$y'_i + f'_i = s_i$,且 y'_i与 f'_i不可能同时为 1 或 u; 其中,$i = 0,1,\cdots,n+1$.

其中,y_i、f_i、y'_i、f'_i和 s_i分别来自下列操作:

(1)a_i和 b_i进行 Y 变换得到 y_i,进行 F 变换得到 f_i; 并且 $y = y_n \cdots y_2 y_1 \phi$, $f = \phi f_{n-1} \cdots f_1 f_0$.

(2)对 y_i和 f_i进行 Y'变换得到 y'_i,进行 F'变换得到 f'_i; 并且 $y' = y'_{n+1} \cdots y'_2 y'_1 \phi$, $f' = \phi f'_n \cdots f'_1 f'_0$.

(3)对 y'_i和 f'_i进行 S 变换得到 s_i,并且 $s = s_{n+1} s_n \cdots s_1 s_0$.

3　SJ-MSD 加法器数学原理与结构

SJ-MSD 加法由一组名为 SJ 变换的三值逻辑变换和一组名为 SJ 规则的操作规则组成.

3.1　加法变换和操作步骤

SJ-MSD 加法运算使用表 1 所示的 5 个三值变换,其中 S_1和 S_2是沈云付教授给出的三值变换,J_1、J_2和 J_3是江家宝博士给出的三值变换. 鉴于这 5 个三值变换由两位学者分别给出,故用二人姓氏的汉语拼音首字母命名该加法器为 SJ-MSD 加法器.

表 1　SJ-MSD 加法中的五种变换

S_1变换				S_2变换				J_1变换			J_2变换			J_3变换		
b	a			b	a			s_1	s_2		s_1	s_2		j_2	j_1	
	0	1	u		0	1	u		0	1		0	1		0	1
0	0	0	u	0	0	1	1	0	0	1	0	0	u	0	0	1
1	0	1	0	1	1	0	0	1	0	1	1	1	0	1	1	0
u	u	0	u	u	1	0	0	u	0	0	u	u	0	u	u	0

对两个 n 位 MSD 数 $a = a_{n-1} \cdots a_1 a_0$ 和 $b = b_{n-1} \cdots b_1 b_0$,完成 SJ-MSD 加法运算的操作规则(简称为 SJ 规则)如下:

步骤 1　对 a 和 b 逐位进行 S_1变换,在变换结果后面补 1 位 0,得到 $n+1$ 位的 s_1值. 同时对 a 和 b 逐位进行 S_2变换,在变换结果的前面补 1 位 0,得到 $n+1$ 位的 s_2值.

步骤 2　由于 s_2的第 n 位为 0,变换 $J_1(0, s_1) \equiv 0$ 被丢弃;所以仅对 s_2和 s_1的低 n 位逐位进行 J_1变换,在变换结果的后面补 1 位 0,得到 $n+1$ 位的 j_1值. 同时对 $n+1$ 位的 s_2和 s_1逐位进行 J_2变换,得到 $n+1$ 位的 j_2值.

步骤 3　对 $n+1$ 位的 j_1和 j_2逐位进行 J_3变换,得到 $n+1$ 位的和值 $j_3 = j_{3,n} \cdots j_{3,1} j_{3,0} = a + b$.

显然,SJ-MSD 加法属于三步式.

3.2　构成 SJ-MSD 加法的充分性

对于表 1 中的 S_1和 S_2变换,由上述操作步骤 1 可以得到以下关系式:

$$a_i + b_i = s_{1,i+1} \times 2 + s_{2,i};$$

其中,$i = 0,1,\cdots,n-1$; $s_{1,0} = s_{2,n} = \phi$;　(2)

对于表 1 中的 J_1和 J_2变换,由操作步骤 1 和步骤 2 可以得到以下关系式:

$$s_{1,i} + s_{2,i} = j_{1,i+1} \times 2 + j_{2,i};$$

其中,$i = 0,1,\cdots,n$; $j_{1,0} = \phi$, $j_{1,n+1} \equiv 0$;　(3)

对于表 1 中的 J_3变换,由操作步骤 2 和步骤 3 可以得到以下关系式:

除 $j_{1,i} = j_{2,i} = 1$ 或 u 外, $j_{1,i} + j_{2,i} = s_i$;

其中,$i = 0,1,\cdots,n$;　(4)

由表 1 可知:虽然 J_2变换结果出现 u,但 J_1变换结果没有出现 u,所以不可能出现 $j_{1,i} = j_{2,i} = u$ 的情况. 由表 1 和上述操作规则可以得到如下等式:

$$j_{1,i} = J_1(s_{2,i-1}, s_{1,i-1}) = J_1(S_2(a_{i-1}, b_{i-1}), S_1(a_{i-2}, b_{i-2}));$$　(5)

$$j_{2,i} = J_2(s_{2,i}, s_{1,i}) = J_2(S_2(a_i, b_i), S_1(a_{i-1}, b_{i-1}));$$　(6)

如果 $j_{2,i} = 1$,则有如下递推结论:

$j_{2,i} = 1$、等式(6)和 J_2变换$\Rightarrow S_1(a_{i-1}, b_{i-1}) = 1$;

$S_1(a_{i-1}, b_{i-1}) = 1$ 和 S_1变换$\Rightarrow a_{i-1} = b_{i-1} = 1$;

$a_{i-1} = b_{i-1} = 1$ 和 S_2变换$\Rightarrow S_2(a_{i-1}, b_{i-1}) = S_2(1,1) = 0$;

$S_2(a_{i-1}, b_{i-1}) = 0$、等式(5)和 J_1变换$\Rightarrow j_{1,i} = J_1(0, S_1(a_{i-2}, b_{i-2})) = 0$;

所以不可能出现 $j_{1,i} = j_{2,i} = 1$ 的情况.

由此可知:SJ-MSD 变换满足沈氏条件.

由表 1 和 SJ 规则还可得到如下等式:

$$j_{1,n+1} = J_1(j_{2,n}, j_{1,n}) = J_1(\phi, j_{1,n}) = 0;$$　(7)

$$r_{n+1} = j_{3,n+1} = J_3(j_{1,n+1}, j_{2,n+1}) = J_3(0, \phi) = 0;$$　(8)

等式(7)和(8)表明:SJ 操作步骤 2 没有计算 $j_{1,n+1}$,步骤 3 没有计算 $j_{3,n+1}$不影响计算结果.

综上所述,SJ-MSD 变换与规则可用于任意位并行 MSD 数加法运算.

例 3　表 2 给出了用 SJ-MSD 变换计算 $r = a + b$ 的过程. 其中,$a = (\text{u11u10u1u110})_M = (-658)_{10}$; $b = (\text{uu10u011u011})_M = (-2645)_{10}$;

表 2 最后一行显示:$j_3 = (\text{u010u001u101u})_M = (-4096 + 0 + 1024 + 0 - 256 + 0 + 0 + 32 - 16 + 8 + 0 + 2 - 1)_{10} = (-3303)_{10} = (-658)_{10} + (-2645)_{10}$,计算正确.

表 2 SJ-MSD 加法实例

变换结果 \ 位 / 数据项	12	11	10	9	8	7	6	5	4	3	2	1	0
a =		u	1	1	u	1	0	u	1	u	1	1	0
b =		u	u	1	0	u	0	1	1	u	0	1	1
S_1:		u	0	1	u	0	0	0	1	u	0	1	0
s_1 =	u	0	1	u	0	0	0	1	u	0	1	0	ϕ
s_2 =	ϕ	0	0	0	1	0	0	0	0	0	1	0	1
J_1:		0	0	0	1	0	0	0	0	0	1	0	1
j_1 =	0	0	0	1	0	0	0	0	0	1	0	1	ϕ
j_2 =	u	0	1	u	u	0	0	1	u	0	0	0	u
和值 j_3 =	u	0	1	0	u	0	0	1	u	1	0	1	u

注:(1)s_1 值由 S_1 变换结果的最低位后补 1 位 0 构成;

(2)s_2 值由 S_2 变换结果的最高位前补 1 位 0 构成;

(3)j_1 值由 J_1 变换结果的最低位后补 1 位 0 构成.

3.3 SJ-MSD 加法器的结构

根据降值设计理论构造的三值光学处理器 SD16,能够在一个处理器位上重构出任意三值变换器的一位. 于是用 SD16 的 5 个处理器位就能构造出表 1 给出的 5 个三值变换器各一位. 由 3.1 节给出的 SJ 规则可知:对于一个 n 位数据的 SJ-MSD 加法器,应该构造出 n 位的 S_1 变换、n 位的 S_2 变换、n 位的 J_1 变换、$n+1$ 位的 J_2 变换和 $n+1$ 位的 J_3 变换. 这些变换器共占用 $5n+2$ 个处理器位. 然而,在构造 SJ-MSD 加法器的工程实现上可显著减少占用的处理器位. 为此,我们考察表 1 中的 J_1 和 J_2 变换.

先考察 J_1 变换,J_1 的 s_2 输入只有 0 和 1 两个值,且当 s_2 取 0 时,无论 s_1 取何值,J_1 变换的结果(j_1)都是 0. 于是,可以把 J_1 变换的真值表改写为一列—J_1 变换中 $s_2=1$ 的列. 这意味着在构造 J_1 变换器时,对于 s_2 信号可以用无光态 W 表达信息 0,用一种有光态(垂直偏振光 V 或水平偏振光 H)来表达信息 1,并在变换器的结构上满足:只在 s_2 为选定的有光态(V 或 H),且 s_1 为表达信息 0 或 1 的光状态时,该变换器输出选定的有光态;而当 s_1 和 s_2 取其它组合值时,该变换器都输出无光态 W.

再考察 J_2 变换,J_2 的 s_2 输入也只有 0 和 1 两个值,但当 s_2 取这两个值时,J_2 变换结果都不全为 0. 若用无光态 W 表示信息 0,用垂直偏振光 V 表示 1,用水平偏振光 H 表示 u,则 J_2 变换器的 s_2 输入光信号只使用了光状态 W 和 V,没有使用光状态 H,而输出光信号用到了三种光状态. 若选择 J_1 变换器的 s_2 输入端用光状态 H 表示 1,光状态 W 表示 0,就可以将 J_1 变换器和 J_2 变换器合并在一个处理器位上实现. 与此对应,J_1 和 J_2 真值表可合并为表 3,其中 H′表示此处用水平偏振光 H 表示 J_1 变换的 s_2 输入信息为 1.

表 3 J_1 和 J_2 的合并

s_1	s_2		
	0	1	H′
0	0	u	1
1	1	0	1
u	u	0	0

在工程实现上完成 J_1 和 J_2 变换器的合并后,就在 4 个处理器位上实现了 SJ-MSD 加法器一位的 5 个三值逻辑变换. 于是,构造一个 n 位数据的 SJ-MSD 加法器,就只需构造出 n 位 S_1 变换器、n 位 S_2 变换器、$n+1$ 位 J_{12} 变换器(J_1/J_2 合并变换器)和 $n+1$ 位 J_3 变换器,占用处理器位总数为 $4n+2$.

必须说明的是,支撑这一工程技巧实现 J_1 和 J_2 变换器合并的科学原理在于:每个运算器中各个信号的物理状态所表达的信息都是独立的,一旦指定,在运算过程中不得改变. 应用这一原理,在 J_1 变换器中,指定 s_2 信号用水平偏振光态表达 1,无光态表达 0,而对 s_1 信号的信息表达指定与其它四个变换相同. 若 SJ-MSD 加法器在这个指定下构造好后保持不变,则结果信息不会混乱.

4 相比于前期 TW-MSD 并行加法器的优势

文献[17]论述了在三值光学计算机中实现的 TW-MSD 加法器由 5 个三值变换器构成,也要经过三步变换操作完成并行加法运算. 其第一步操作获得的中间计算结果 t 和 w 比原始数据 a 和 b 多一位,而第二步操作获得的中间计算结果 t'和 w'比 t 和 w 又多出一位. 因此,n 位的 TM-MSD 数加法由一个 n 位 T 变换器、一个 n 位 W 变换器、一个 $n+1$ 位 T′变换器、一个 $n+1$ 位 W′变换器和一个 $n+2$ 位 T_2 变换器构成,占用处理器位总数为 $5n+4$. 并且计算结果为 $n+2$ 位,比原始数据增多了两位.

SJ-MSD 加法器和 TW-MSD 加法器都能对两个 MSD 数作并行加法,都遵守同一数学计算规则——MSD 数加法规则,都满足沈氏充分条件. 二者的差异在于所使用的三值变换器的结构复杂度不同,其工程实现的难度不同. TW-MSD 加法器从 1960 年给出的一个特殊加法运算过程演变而来,其结构没有达到尽善尽美. SJ-MSD 加法器是在沈氏充分性定理奠定了这个方面的数学理论之后,通过对多种 MSD 加法器的三值变换进行筛选和论证,简化了各个变换器的结构、进而又在工程上将两个变换合并在一起,得到的一个高效的 MSD 加法器. 与先前使用的 TW-MSD 加法器相比,SJ-MSD 加法器有三个

明显的优势:

(1)比 TW-MSD 加法器少占用处理器位数 25%

在工程上将 J_1 和 J_2 变换合并在一个处理器位实现后,n 位 SJ-MSD 加法器占用处理器位总数为 $4n+2$ 位. 而 TW-MSD 加法器占用处理器位总数为 $5n+4$ 位. 显然,SJ-MSD 加法器占用处理器位的数量比 TW-MSD 加法器减少了 25% 稍强.

(2)比 TW-MSD 加法器减少一位附加位数

对于两个 n 位的原始数据,在 TW-MSD 加法器中运算,得到的和值是 $n+2$ 位. 而在 SJ-MSD 加法器中运算,得到的和值是 $n+1$ 位. 即 SJ-MSD 加法器的计算结果比 TW-MSD 加法器少一个附加数据位. 虽然两个结果的 MSD 数有不同的形式,但它们所表达的值是相同的. 即:如果将二者转换成非冗余表达形式,比如十进制数,则会得到完全相同的形式. 这意味着在连加运算中,TW-MSD 加法器的位数膨胀速度比 SJ-MSD 加法器快一倍,在对大量数据进行连加或乘法运算中, SJ-MSD 加法器表现出对处理器的高使用率.

(3)拼接两个小数据时比 TW-MSD 少补一位 0

由于 TW-MSD 加法器的和值比原始数据多出两位,因此在将两个小数据拼接成一个大数据时,要在两数据之间空出两个数据位,这两位被赋值 0,因此该技术被称为夹双 0 拼接[17]. 而 SJ-MSD 加法器的计算结果位数只比原始数据位多一位,于是,在将两个小数据拼接成一个大数据时,只需在两数据之间空出一个数据位,因此可称之为“夹单 0 拼接技术”. 其证明过程与文献[17]给出的方法类同.

小数据拼接技术可以实现多个数据共享一个加法器,它等价于把一个大加法器割裂成多个小加法器,每个小加法器计算一对数据. 反之,一个超过加法器位数的大数据也可以被截成多个段,逐段完成相加,再把各段的计算结果连接起来而获得原始大数据的和值,这一技术在 TW-MSD 加法器和 SJ-MSD 加法器上完全相同.

5　SJ-MSD 加法器的设计与实现

SD16 处理器基本模块的像素排列如图 2 所示,它有 192 个处理器位,每个处理器位的主光路有三个像素,其输入光信号的状态分别为无光态、垂直偏振光态和水平偏振光态,故称之为 W、V 和 H 像素. 每个像素都可以输出无光态或水平偏振光态或垂直偏振光态,于是主光路的每个像素可以重构出 3 个最简运算基元,故每个处理器位可以重构出 9 个最简运算基元. 根据降值设计理论[20],任意三值逻辑运算器的一位可以用不超过 6 个最简运算基元构成,于是 SD16 每个数据位都可以构成任意三值逻辑运算器的一位.

(1)重构出 n 位的 SJ-MSD 并行加法器

采用 3.3 节讨论的技术,把 SJ-MSD 加法器的 J_1 和 J_2 变换合并在一个处理器位上. 加法器的每一位可以任意选取处理器位来构建,为了数据整理方便,我们选取位号连续的处理器位构建 SJ-MSD 加法器,于是从第 c_0 号处理器位开始构造 n 位 SJ-MSD 加法器的设计如下:

①将 $c_0 - c_{n-1}$ 号处理器位构造成 n 位的 S_1 变换器,将 W、V、H 像素的输出迭合后形成 S_1 的输出.

②将 $c_n - c_{2n-1}$ 号处理器位构造成 n 位的 S_2 变换器,将 W、V、H 像素的输出迭合后形成 S_2 的输出.

③将 $c_{2n} - c_{3n}$ 号处理器位构造成 $n+1$ 位的 J_{12} 变换器,即:这些处理器位的 W 像素和 V 像素构造成 $n+1$ 位 J_2 变换器,H 像素构造成 $n+1$ 位 J_1 变换器. 相应地给 V 像素和 H 像素输入相同的数据 s_2;将 W 像素和 V 像素的输出进行迭合后形成 J_2 输出,而 H 像素的输出为 J_1 输出.

④将 $c_{3n+1} - c_{4n+1}$ 号处理器位构造成 $n+1$ 位 J_3 变换器,将 W、V、H 像素的输出迭合成 J_3 输出.

(2)流水计算算法

三值光学处理器重构完成后,计算的数据越多,重构耗时相对于有效计算时间就越小,光学处理器的效率就越高. 故三值光学处理器常用于计算批量数据. 批量数据在此 SJ-MSD 加法器会以流水计算方式处理批量数据,计算过程如下:

①将变量 s_1、s_2、j_1、j_2 和 j_3 清零,计数变量 c_n 清零.

②形成处理器控制光路输入数据帧:

n 位原始数据 b 送入控制光路输入数据帧的 $c_0 - c_{n-1}$ 位(对应于 S_1 变换)和 $c_n - c_{2n-1}$ 位(对应于 S_2 变换);$n+1$ 位变量 s_1 送入帧的 $c_{2n} - c_{3n}$ 位(对应于 J_1 和 J_2 变换);$n+1$ 位变量 j_2 送入帧的 $c_{3n+1} - c_{4n+1}$ 位(对应于 J_3 变换).

③形成处理器主光路输入数据帧:

n 位原始数据 a 送入主光路输入数据帧的 $c_0 - c_{n-1}$ 位和 $c_n - c_{2n-1}$ 位;$n+1$ 位变量 s_2 送入帧的 $c_{2n} - c_{3n}$ 位,并使 V 像素与 H 像素有相同的输入;$n+1$ 位变量 j_1 送入帧的 $c_{3n+1} - c_{4n+1}$ 位.

④将处理器控制光路输入数据送入控制光路编码器,产生本次运算处理器控制光路三态光信号.

⑤将处理器主光路输入数据送入主光路编码器,产生本次运算处理器主光路三态光信号.

⑥启动解码器,获取处理器所有位的输出光信号并转换成对应的电信号,形成处理器的输出数据.

⑦将 $c_0 - c_{n-1}$ 号处理器位(S_1 变换)输出的数据后面补 0 后送 s_1 变量,将 $c_n - c_{2n-1}$ 号处理器位(S_2 变换)输出的数据前面补 0 后送 s_2 变量. 将 $c_{2n} - c_{3n}$ 号处理器位的 H 像素的输出数据送 j_1 变量. 将 $c_{2n} - c_{3n}$ 号处理器位的 W 像素和 V 像素的输出合并后送 j_2 变量. 将 $c_{3n+1} -$

c_{4n+1}号处理器位的输出送j_3变量作计算结果.

⑧计数变量c_n增1. 当$c_n>2$时,将j_3变量的值输出,形成一对原始数据的计算结果.

当$c_n<N$(原始数据个数)时,取下一对原始数据送入a和b,返回步骤②.

当$N<=c_n<N+2$时,取0值送入a和b,返回步骤②.

当$c_n=N+2$时,执行步骤⑨.

⑨SJ-MSD加法运算结束.

(3)小数据拼接算法

在一批原始数据中,数据的位数往往参差不齐. 而三值光学处理器的位数众多,为了减少计算的总时间,大多情况下构造一个与该批原始数据中最大数据位数相同的SJ-MSD加法器,这时就要将位数较少的小数据拼接成位数接近SJ-MSD加法器位数的大数据. 因此,数据拼接是实际工作中常常用到的数据预处理技术,实现这个技术的算法如下:

①设有K对原始小数据,第i对小数据的数据位数为N_i,$i=K,K-1,\cdots,1$.

②分别在第$K-1$、第$K-2$、…、第1对小数据的最高位前补一位0.

③然后将这K对原始数据依次相接,形成一对大原始数据. 显然,这对大原始数据的位数$N=N_K+N_{K-1}+\cdots+N_1+K-1$.

④将该对大原始数据送SJ-MSD加法器进行计算,记获得的和值为h. 则h的位数为$N+1$位.

⑤从h的最低位起,将h截成K段. 第i段h_i是第i对原始小数据的和,其数据位数为N_i+1位.

例4 有3对小数据,$c_3=(\mathrm{u01111u110111u11u1u1})_M=-279701$;$c_2=(\mathrm{u01111u110111})_M=-2185$;$c_1=(\mathrm{11u11u1u1})_M=363$;$d_3=(\mathrm{1u01u11101uu11101u1u})_M=324069$;$d_2=(\mathrm{1u01u11101uu1})_M=2531$;$d_1=(\mathrm{u11101u1u})_M=-27$. 为了便于阅读,数据$c_2$和$d_2$加了下划线,拼接后形成的一对大原始数据如下:

$c=(\mathrm{u01111u110111u11u1u1\phi u01111u110111\phi 11u11u1u1})_M$;

$d=(\mathrm{1u01u11101uu11101u1u\phi 1u01u11101uu1\phi u11101u1u})_M$;

大原始数据c和d送入SJ-MSD加法器进行计算,得到和值为:$e=c+d=(\mathrm{0000010110u01011u00000000010110u01001011u0000})_M$.

对e值进行截取,得到三对原始小数据的和值:

$e_3=(\mathrm{0000010110u01011u0000})_M=44368=-279701+324069=c_3+d_3$;

$e_2=(\mathrm{0000010110u010})_M=346=-2185+2531=c_2+d_2$;

$e_1=(\mathrm{01011u0000})_M=336=363+(-27)=c_1+d_1$.

(4)大数据截断算法

若计算数据的位数比三值光学处理器中加法器的数据位数多,为了不再重构新加法器,需要把大数据截断成若干段,分段运算后再把各段的和值连接起来,获得原始数据的和值. 因此,数据截断也是实际工作中常用到的数据预处理技术. 在SJ-MSD加法器中采用的数据截断技术与文献[17]中描述的完全相同,在此不再赘述.

6 SJ-MSD加法器实验

继在SD16的1号机上首次成功实现46位的SJ-MSD加法器之后,为验证SJ-MSD加法器的正确性,并比较它与TW-MSD加法器的特征,2017年8月作者在此光学处理器上同时构造出一个20位TW-MSD加法器和一个19位SJ-MSD加法器. 随后对它们进行了全面测试. 测试实验分三个阶段,首先是实验准备阶段,包括规划实验内容、选择实验用例、准备实验用例的理论结果、调试实验系统等. 其次是实验实施阶段,包括重构两个加法器、逐一把实验用例数据送入两个加法器、逐一拍照加法器的输出结果、将加法器输出结果与理论结果对比并记录对比情况. 最后是实验分析阶段,主要分析每一实验用例的实验结果和理论结果的差异,找出产生差异的原因,排除错误,完成补充实验,得出最终的实验结论等. 具体描述如下.

6.1 实验准备

这部分工作主要是准备实验设备、实验规划、实验用例并推算出各实验用例处理器的理论输出.

(1)实验设备简介——SD16处理器输出画面

主要实验设备是三值光学计算机原型系统SD16的1号机. 其外观如图1所示,中部有亮点的黑色区域是光学处理器,共有576个像素,排列成24×24的阵列. 高亮点是输出垂直偏振光(V状态)的像素,次亮点是输出水平偏振光(H状态)的像素,其余点是输出无光态(W状态)的像素. 每行中相邻的三个像素构成处理器的一位,其192个处理器位的编号如图2中V像素上的数字所示.

图1 SD16的1号机外形

(2)构造加法器

我们在SD16的1号机处理器上同时构造的TW-MSD和SJ-MSD加法器如图2所示,其中0到103号处理器位构造成TW-MSD加法器,104到181号处理器位构造成SJ-MSD加法器. 各三值变换器占用的处理器位如下:

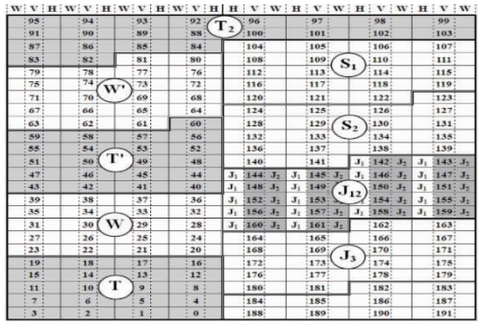


图2　SD16处理器位排列和加法器的分布图

对于 20 位的 TW-MSD 加法器有:0 ~ 19 号处理器位组成 20 位 T 变换器,20 ~ 39 号处理器位组成 20 位 W 变换器,40 ~ 60 号处理器位组成 21 位 T′变换器,61 ~ 81 号处理器位组成 21 位 W′变换器,82 ~ 103 号处理器位组成 22 位 T_2 变换器.

对于 19 位的 SJ-MSD 加法器有:104 ~ 122 号处理器位组成 19 位 S_1 变换器;123 ~ 141 号处理器位组成 19 位 S_2 变换器;142 ~ 161 号处理器位组成 20 位 J_{12} 变换器,其中,H 像素(H 像素上标记 J_1)组成 20 位 J_1 变换器,W 像素和 V 像素(W 像素上标记 J_2)组成 20 位 J_2 变换器;162 ~ 181 号处理器位组成 20 位 J_3 变换器;其余处理器位没有使用.

为便于研究数据截断技术,我们将 20 位 TW-MSD 加法器视为一个 6 位 TW-MSD 加法器 A 和一个 12 位 TW-MSD 加法器 B,其中,加法器 A 的 T、W、T′、W′、T_2 变换器依次为:0 ~ 5 号、20 ~ 25 号、40 ~ 46 号、61 ~ 67 号和 82 ~ 89 号处理器位. 加法器 B 的 T、W、T′、W′、T_2 变换器依次为:8 ~ 19 号、28 ~ 39 号、48 ~ 60 号、69 ~ 81 号和 90 ~ 103 号处理器位. 同理,19 位的 SJ-MSD 加法器可被视为一个 6 位 SJ-MSD 加法器 C 和一个 12 位 SJ-MSD 加法器 D,其中,加法器 C 的 S_1、S_2、J_1、J_2、J_3 变换器依次为:104 ~ 109 号、123 ~ 128 号、142 ~ 148 号、142 ~ 148 号和 162 ~ 168 号处理器位. 加法器 D 的 S_1、S_2、J_1、J_2、J_3 变换器依次为:111 ~ 122 号、130 ~ 141 号、149 ~ 161 号、149 ~ 161 号和 169 ~ 181 号处理器位.

(3)实验规划

①验证 SJ-MSD 加法器和 TW-MSD 加法器的计算结果正确,都能实现两个 MSD 数的并行加法.

②比较 SJ-MSD 和 TW-MSD 加法器处理相同数据需要的处理器位数.

③验证 SJ-MSD 加法器和 TW-MSD 加法器都可以实施数据拼接与截断技术.

④验证 SJ-MSD 加法器和 TW-MSD 加法器都可以采用数据流水方式连续计算多对加法数据.

(4)准备测试用例

为了验证两种加法器都可正确计算任意 MSD 数据,并比较两者在处理相同数据时各自占用的处理器位资源,用例 1、用例 3、用例 5、……、用例 17 中,两种加法器输入数据的低 16 位完全相同,其余的高位均为补加的 0.

为验证基于 TW-MSD 和 SJ-MSD 加法器的数据拼接技术,实验用例 2、用例 4、用例 6、……、用例 18 中 TW-MSD 加法器的 20 位输入数据(TWD_{19} ~ TWD_0)被视为两个小数据的拼接,其中高 12 位(TWD_{19} ~ TWD_8)被视为“小数据 1”,低 6 位(TWD_5 ~ TWD_0)被视为“小数据 2”,TWD_7 ~ TWD_6 为两个小数据拼接时中间补的两个 0. 同理,SJ-MSD 加法器的 19 位输入数据(SJD_{18} ~ SJD_0)也被视为两个小数据的拼接,其中高 12 位(SJD_{18} ~ SJD_7)被视为“小数据 3”,低 6 位(SJD_5 ~ SJD_0)被视为“小数据 4”,SJD_6 为两个小数据拼接时中间补的一个 0. 从用例 2、用例 4、……、用例 18 的运算结果中截取并分析小数据 1、小数据 2、小数据 3 和小数据 4 的结果,可以判断在两种加法器上实施数据拼接技术的效果.

为验证基于 TW-MSD 和 SJ-MSD 加法器的数据截断技术,用例 2 的输入数据来自用例 1. 具体做法为:取用例 1 的 TWD_{15} ~ TWD_0 位,在 TWD_6 位和 TWD_5 位之间截断;其中 TWD_5 ~ TWD_0 构成用例 2 的 6 位小数据 2(TWD_5 ~ TWD_0),用例 2 的 TWD_7 和 TWD_6 用 0 填充,用例 1 的 TWD_{15} ~ TWD_6 和 TWD_5 ~ TWD_4(断口重复 2 位,表 4 中用斜体表示)构成用例 2 的 12 位小数据 1(TWD_{19} ~ TWD_8). 显然,用例 2 的两个小数据就是用例 1 的低 16 位输入数据的一个截断. 同理,用例 2 中 SJ-MSD 加法器的两个输入小数据也来自用例 1 的 16 位输入数据 SJD_{15} ~ SJD_0,与 TW-MSD 加法器输入数据的差别仅在小数据 4 最高位只补一个 0.

用例 4、用例 6、……、用例 18 是仿照用例 2 生成办法,分别由用例 3、用例 5、用例 7、……、用例 17 生成. 比较用例 1 与用例 2、用例 3 与用例 4、……、用例 17 与用例 18 的计算结果,可判断在 SJ-MSD 加法器上实施数据截断的效果.

用例 1、用例 3、……、用例 9 的输入数据也是随机数据,但它们在输入的第 6 位和第 5 位(截断位置)处的数据不同,用例 1 中两个加法器的输入数据 m_1 和 n_1 在这个位置的值分别为 01 和 u0,而用例 3 的输入数据 m_3 和 n_3 在这个位置的值分别为 01 和 10;用例 5 的输入数据 m_5 和 n_5 在这个位置的值分别为 uu 和 11;用例 7 的输入数据 m_7 和 n_7 在这个位置的值分别为 u0 和 u1;用例 9 的输入数据 m_9 和 n_9 在这个位置的值分别为 11 和 11. 这是考察断口数值会给数据截断带来何种影响.

用例 11 的输入数据 m_{11} 和 n_{11} 低 16 位全 1. 用例 13 的输入数据 m_{13} 和 n_{13} 低 16 位全 u;用例 15 的输入数据

m_{15}和n_{15}的低16位全0;用例17的输入数据m_{17}的低16位全为1而n_{17}低16位全为u;这几个实验例用于验证两种加法器对边界值的运算正确性. 用例19是用例7的一个错误截断;用例20是用例3正确截断后的一个错误拼接;这两个实验例是数据剪辑技术的反证例. 用例21和22是两个给光学处理器充入全0值,其作用是促使光学处理器完成对用19和用例20的后两步操作.

准备好实验用例如表4所示.

表4 实验用例表

功能	用例	加数	TWD_{19}~TWD_8(小数据1)	TWD_7~TWD_0(小数据2)	SJD_{18}~SJD_7(小数据3)	SJD_6~SJD_0(小数据4)
拼接与截断	1	m_1	ϕ ϕ ϕ ϕ 1 u 0 u 1 u 1 1	u 0 0 1 u 1 0 0	ϕ ϕ ϕ 1 u 0 u 1 u 1 1 u	0 0 1 u 1 0 0
		n_1	ϕ ϕ ϕ ϕ 0 u 1 1 0 u 1 0	1 1 u 0 1 u 0 u	ϕ ϕ ϕ 0 u 1 1 0 u 1 0 1	1 u 0 1 u 0 u
	2	m_2	1 u 0 u 1 u 1 1 u 0 0 *1*	ϕ ϕ *0* *1* u 1 0 0	1 u 0 u 1 u 1 1 u 0 0 *1*	ϕ *0* *1* u 1 0 0
		n_2	0 u 1 1 0 u 1 0 1 1 *u* *0*	ϕ ϕ *u* *0* 1 u 0 u	0 u 1 1 0 u 1 0 1 1 *u* *0*	ϕ *u* *0* 1 u 0 u
	3	m_3	ϕ ϕ ϕ ϕ u 1 0 u 1 1 1 0	u 1 0 1 u 1 0 1	ϕ ϕ ϕ u 1 0 u 1 1 1 0 u	1 0 1 u 1 0 1
		n_3	ϕ ϕ ϕ ϕ u 0 1 1 1 u 1 0	1 1 1 0 1 u 1 u	ϕ ϕ ϕ u 0 1 1 1 u 1 0 1	1 1 0 1 u 1 u
	4	m_4	u 1 0 u 1 1 1 0 u 1 *0* *1*	ϕ ϕ *0* *1* u 1 0 1	u 1 0 u 1 1 1 0 u 1 *0* *1*	ϕ *0* *1* u 1 0 1
		n_4	u 0 1 1 1 u 1 0 1 1 *1* *0*	ϕ ϕ *1* *0* 1 u 1 u	u 0 1 1 1 u 1 0 1 1 *1* *0*	ϕ *1* *0* 1 u 1 u
	5	m_5	ϕ ϕ ϕ ϕ 1 u 0 1 u 1 1 0	0 1 u u u 1 0 u	ϕ ϕ ϕ 1 u 0 1 u 1 1 0 0	1 u u u 1 0 u
		n_5	ϕ ϕ ϕ ϕ 1 1 u 1 0 1 1 u	u 0 1 1 1 u u u	ϕ ϕ ϕ 1 1 u 1 0 1 1 u u	0 1 1 1 u u u
	6	m_6	1 u 0 1 u 1 1 0 0 1 *u* *u*	ϕ ϕ *u* *u* u 1 0 u	1 u 0 1 u 1 1 0 0 1 *u* *u*	ϕ *u* *u* u 1 0 u
		n_6	1 1 u 1 0 1 1 u u 0 *1* *1*	ϕ ϕ *1* *1* 1 u u u	1 1 u 1 0 1 1 u u 0 *1* *1*	ϕ *1* *1* 1 u u u
	7	m_7	ϕ ϕ ϕ ϕ 1 u u 1 u 1 u 0	1 1 u 0 1 u u 0	ϕ ϕ ϕ 1 u u 1 u 1 u 0 1	1 u 0 1 u u 0
		n_7	ϕ ϕ ϕ ϕ u u u 1 0 u u u	1 0 u 1 1 u 0 1	ϕ ϕ ϕ u u u 1 0 u u u 1	0 u 1 1 u 0 1
	8	m_8	1 u u 1 u 1 u 0 1 1 *u* *0*	ϕ ϕ *u* *0* 1 u u 0	1 u u 1 u 1 u 0 1 1 *u* *0*	ϕ *u* *0* 1 u u 0
		n_8	u u u 1 0 u u u 1 0 *u* *1*	ϕ ϕ *u* *1* 1 u 0 1	u u u 1 0 u u u 1 0 *u* *1*	ϕ *u* *1* 1 u 0 1
	9	m_9	ϕ ϕ ϕ ϕ 1 u 0 1 u 1 u u	1 u 1 1 u 0 u u	ϕ ϕ ϕ 1 u 0 1 u 1 u u 1	u 1 1 u 0 u u
		n_9	ϕ ϕ ϕ ϕ 1 1 u 1 0 u 1 u	0 u 1 1 1 u 1 u	ϕ ϕ ϕ 1 1 u 1 0 u 1 u 0	u 1 1 1 u 1 u
	10	m_{10}	1 u 0 1 u 1 u u 1 u *1* *1*	ϕ ϕ *1* *1* u 0 u u	1 u 0 1 u 1 u u 1 u *1* *1*	ϕ *1* *1* u 0 u u
		n_{10}	1 1 u 1 0 u 1 u 0 u *1* *1*	ϕ ϕ *1* *1* 1 u 1 u	1 1 u 1 0 u 1 u 0 u *1* *1*	ϕ *1* *1* 1 u 1 u
	11	m_{11}	ϕ ϕ ϕ ϕ 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	ϕ ϕ ϕ 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1
		n_{11}	ϕ ϕ ϕ ϕ 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	ϕ ϕ ϕ 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1
	12	m_{12}	1 1 1 1 1 1 1 1 1 1 1 *1* *1*	ϕ ϕ *1* *1* 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 *1* *1*	ϕ *1* *1* 1 1 1 1
		n_{12}	1 1 1 1 1 1 1 1 1 1 1 *1* *1*	ϕ ϕ *1* *1* 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 *1* *1*	ϕ *1* *1* 1 1 1 1
	13	m_{13}	ϕ ϕ ϕ ϕ u u u u u u u u	u u u u u u u u	ϕ ϕ ϕ u u u u u u u u u	u u u u u u u u
		n_{13}	ϕ ϕ ϕ ϕ u u u u u u u u	u u u u u u u u	ϕ ϕ ϕ u u u u u u u u u	u u u u u u u u
	14	m_{14}	u u u u u u u u u u *u* *u*	ϕ ϕ *u* *u* u u u u	u u u u u u u u u u *u* *u*	ϕ *u* *u* u u u u
		n_{14}	u u u u u u u u u u *u* *u*	ϕ ϕ *u* *u* u u u u	u u u u u u u u u u *u* *u*	ϕ *u* *u* u u u u
	15	m_{15}	ϕ ϕ ϕ ϕ 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	ϕ ϕ ϕ 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0
		n_{15}	ϕ ϕ ϕ ϕ 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	ϕ ϕ ϕ 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0
	16	m_{16}	0 0 0 0 0 0 0 0 0 0 0 *0* *0*	ϕ ϕ *0* *0* 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 *0* *0*	ϕ *0* *0* 0 0 0 0
		n_{16}	0 0 0 0 0 0 0 0 0 0 0 *0* *0*	ϕ ϕ *0* *0* 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 *0* *0*	ϕ *0* *0* 0 0 0 0
	17	m_{17}	ϕ ϕ ϕ ϕ 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	ϕ ϕ ϕ 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1
		n_{17}	ϕ ϕ ϕ ϕ u u u u u u u u	u u u u u u u u	ϕ ϕ ϕ u u u u u u u u u	u u u u u u u
	18	m_{18}	1 1 1 1 1 1 1 1 1 1 1 *1* *1*	ϕ ϕ *1* *1* 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 *1* *1*	ϕ *1* *1* 1 1 1 1
		n_{18}	u u u u u u u u u u *u* *u*	ϕ ϕ *u* *u* u u u u	u u u u u u u u u u *u* *u*	ϕ *u* *u* u u u u
错误截断	19	m_{19}	ϕ ϕ 1 u u 1 u 1 u 0 1 u	ϕ ϕ 0 1 1 u 1 0	ϕ ϕ 1 u u 1 u 1 u 0 1 1	ϕ u 0 1 u u 0
		n_{19}	ϕ ϕ u u u 1 0 u u u 1 u	ϕ ϕ 1 u 1 u 0 1	ϕ ϕ u u u 1 0 u u u 1 0	ϕ u 1 1 u 0 1
错误拼接	20	m_{20}	ϕ ϕ u 1 0 u 1 1 1 0 u 0	*1* *0* *1* *0* u u 0 1	ϕ u 1 0 u 1 1 1 0 u 1 *0*	*1* *0* *1* u 1 0 1
		n_{20}	ϕ ϕ u 0 1 1 1 u 1 0 1 1	*0* *u* *0* *u* 1 0 1 u	ϕ u 0 1 1 1 u 1 0 1 1 *1*	*0* *1* *0* 1 u 1 u
充入全0	21	m_{21}	0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0
		n_{21}	0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0
	22	m_{22}	0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0
		n_{22}	0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0

(5)准备实验用例的理论输出图像

根据两个加法器的结构,可以从理论上推测出对每一个实验例的每一步操作后光学处理器的输出状态,实验前先将这些从理论上预测到的光学处理器输出状态绘制成图像. 针对表4给出的实验例,共绘制处理器的理论输出图像22幅. 第1幅是实验用例1完成

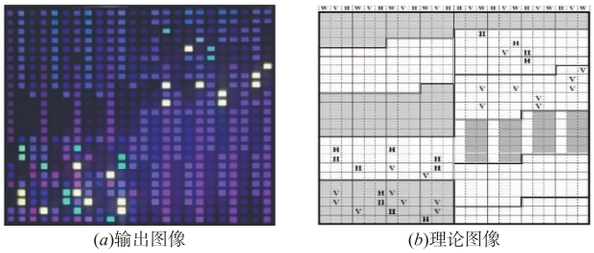
第 1 步操作后光学处理器应该输出的图像,如图 3(*b*)所示,其中 H 表示该像素应该输出水平偏振光,V 表示应该输出垂直偏振光,无符号表示应该输出无光态;第 2 幅是用例 1 完成第 2 步操作,同时用例 2 完成第 1 步操作后光学处理器应该输出的图像,如图 4(*b*)所示;第 3 幅是用例 1 完成第 3 步操作获得计算结果,同时用例 2 完成第 2 步操作,用例 3 完成第 1 步操作后光学处理器应该输出的图像,如图 5(*b*)所示. 以此类推,直至 20 个用例以流水方式计算完毕.

(6)按照第 5 节给出的操作步骤和算法准备好操控软件

该软件将加法器的三步操作分离,每一步操作都等待上位机发出步进指令后方进行,这样做的目的是在光学处理器的每一个操作步之后留有拍摄处理器输出图像的时间,为分析光学处理器的运行状况提供依据.

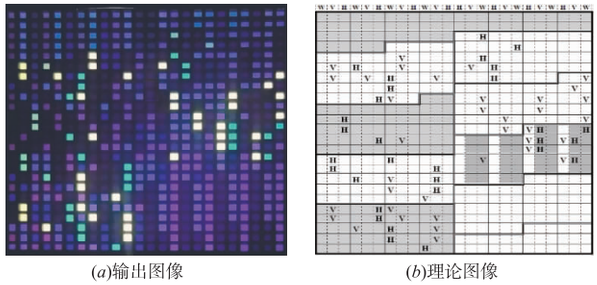
表 5　实验结果记录表

实验用例号	1	2	3	4	5	6	7	8	9	10	11
与理论图像比较	√	√	√	√	√	√	√	√	√	√	√
实验用例号	12	13	14	15	16	17	18	19	20	21	22
与理论图像比较	√	√	√	√	√	√	√	√	√	√	√



(*a*)输出图像　　(*b*)理论图像

图3　用例1第1步操作



(*a*)输出图像　　(*b*)理论图像

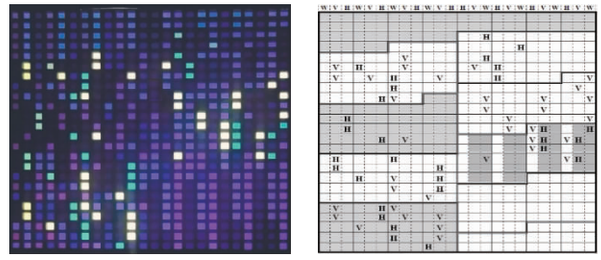
图4　用例1第2步操作

6.2 实验实施过程

实验的实施过程严格按下列步骤进行:

(1)使用处理器重构指令,在 SD16 的 1 号机上,按图 2 的规划构造出 TW-MSD 和 SJ-MSD 加法器.

(2)将表 4 给出的实验用例 1 的输入数据送入光学处理器的数据输入端,然后发出操控光学处理器运行的"步进指令".



(*a*)输出图像　　(*b*)理论图像

图5　用例1第3步操作

(3)拍摄光学处理器输出的图像,如图 3(*a*).

(4)将拍摄到的图像与实现准备好的理论图像对比. 若二者相同,在实验记录表 5 中对应位置打√号,进行步骤(5);若两个图像不相同,则对照理论输入数据检查实际输入的数据是否有错误,若输入数据有误,则更正输入后重新拍照;若输入数据无误,则在实验记录表中对应位置打 ×,并停止实验,进入实验结果分析阶段.

(5)对表 4 给出的下一个实验例重复执行步骤(2)、(3)和(4),直到 22 个实验例都送入光学处理器.

(6)结束实验.

在实验过程中出现过几次输错数据的情况,没有出现其它错误. 排除输入数据错误之后,得到的实验结果记录表为表 5,所有实验实施过程中拍摄的光学处理器输出图像都与对应的理论图像相同. 实验结果表明:在计算实验用例的过程中,两个加法器的工作状态与理论预测完全一致.

作为示例,图 3、图 4 和图 5 分别给出了实验用例 1 进行第一步、第二步和第三步操作后的光学处理器输出图像和对应的理论图像.

6.3　实验结果分析

从光学处理器的输出图像或理论图像都可以判读出两个加法器所拥有的各个三值变换器输出数据. 为分析数据剪辑技术在两个加法器上的实施情况,判读出实验用例 1 至用例 20 的两个加法器输出的数值,并将这些和值列出在表 6 中.

对于 TW-MSD 加法器,将 r_2 的小数据 1 的最低两位剪去,同时将小数据 2 的最高两位剪去,剪去的数据在表 6 中用斜体表示,然后将两个小数据剩下的部分连接起来就得到 r_1 中的 TW-MSD 加法器结果. 对于 SJ-MSD 加法器,将 r_2 的小数据 3 的最低两位剪去,同时将小数据 4 的最高位剪去,剪去的数据在表 6 中也用斜体表示,然后将两个小数据剩下的部分连接起来就得到 r_1 中的 SJ-MSD 加法器结果. 所以对用例 1 给出的随机大数据截断成两个小数据分别进行计算的数据截断技术成立.

表 6 实验用例的计算结果值

用例	和值	TWR_{21} ~ TWR_8(小数据 1)	TWR_7 ~ TWR_0(小数据 2)	SJR_{19} ~ SJR_7(小数据 3)	SJR_6 ~ SJR_0(小数据 4)
1	r_1	0 0 0 0 0 0 0 1 u 0 0 1 1 u	0 1 0 u 0 0 u 1	0 0 0 0 0 1 u 0 0 1 1 u 0	1 0 u 0 0 0 u
2	r_2	0 0 0 1 u 0 0 1 1 u 0 1 *0 u*	*0 0* 0 u 0 0 u 1	0 0 1 u 0 0 1 1 u 0 1 *0 u*	*0* 0 u 0 0 0 u
3	r_3	0 0 0 0 0 u 1 0 u 1 0 1 0 0	1 1 0 u 0 1 u 0	0 0 0 u 1 0 u 1 0 1 0 0 1	1 0 u 0 1 u 0
4	r_4	0 u 1 0 u 1 0 1 0 0 1 1 *0 u*	*0 1* 0 u 0 1 u 0	u 1 0 u 1 0 1 0 0 1 1 *0 u*	*1* 0 u 0 1 u 0
5	r_5	0 0 0 0 0 1 0 0 0 0 0 1 u 1	0 u 0 0 0 u 0 0	0 0 0 1 0 0 0 0 0 1 u 1 0	u 0 0 0 u 0 0
6	r_6	0 1 0 0 0 0 0 1 u 1 0 u *0 0*	*0 0* 0 0 0 u 0 0	1 0 0 0 0 0 1 u 1 0 u *0 0*	*0* 0 0 0 u 0 0
7	r_7	0 0 0 0 0 0 u u 1 u 1 u 0 0	0 0 1 0 u 0 0 u	0 0 0 0 u u 1 0 u u 0 0 0	0 1 0 u 0 0 u
8	r_8	0 0 u u 1 u 1 u 0 0 0 0 *1 u*	*0 u* 1 0 u 0 0 u	0 u u 1 0 u u 0 0 0 0 *1 u*	*u* 1 0 u 0 0 u
9	r_9	0 0 0 0 0 1 0 0 0 u 1 0 u 0	0 1 1 0 u 1 u 0	0 0 0 1 0 0 0 0 u 0 u 0 0	1 1 0 0 u u 0
10	r_{10}	0 1 0 0 0 u 1 0 u 0 0 1 *1 0*	*0 1* 1 0 u 1 u 0	1 0 0 0 0 u 0 u 0 0 1 *1 0*	*1* 1 0 0 u u 0
11	r_{11}	0 0 0 0 0 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 0	0 0 0 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 0
12	r_{12}	0 1 1 1 1 1 1 1 1 1 1 1 1 *1 0*	*0 1* 1 1 1 1 1 0	1 1 1 1 1 1 1 1 1 1 1 1 1 *1 0*	*1* 1 1 1 1 1 0
13	r_{13}	0 0 0 0 0 u u u u u u u u u u	u u u u u u u 0	0 0 0 u u u u u u u u u u	u u u u u u 0
14	r_{14}	0 u u u u u u u u u u u u *0*	*0 u* u u u u u 0	u u u u u u u u u u u u *0*	*u* u u u u u 0
15	r_{15}	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0
16	r_{16}	0 0 0 0 0 0 0 0 0 0 0 0 0 *0 0*	*0 0* 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 *0 0*	*0* 0 0 0 0 0 0
17	r_{17}	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0
18	r_{18}	0 0 0 0 0 0 0 0 0 0 0 0 0 *0 0*	*0 0* 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 *0 0*	*0* 0 0 0 0 0 0
19	r_{19}	0 0 0 0 u u 1 u 1 u 0 0 u 0	0 1 u 1 u 1 0 u	0 0 0 u u 1 0 u u 0 0 1 u	u 1 0 u 0 0 u
20	r_{20}	0 0 0 u 1 0 u 1 0 1 0 0 *1 u*	*1 u* 0 1 0 0 u 0	0 u 1 0 u 1 0 1 0 0 1 1 *0*	*0* 0 u 0 1 u 0

同理,比较 r_3 和 r_4、比较 r_5 和 r_6、……、比较 r_{17} 和 r_{18},可以看到两种加法器的数据截断技术都成立,而且断口值对这项技术没有影响.

不难分析,对于结果 r_2、r_4、……、r_{18},TWR_{21} ~ TWR_8 正好是小数据 1 的和值,TWR_7 ~ TWR_0 正好是小数据 2 的和值,SJR_{19} ~ SJR_7 正好是小数据 3 的和值,SJR_6 ~ SJR_0 正好是小数据 4 的和值. 由此可见,两种加法器的数据拼接技术也成立,即 SJ-MSD 加法器的"夹单 0 剪辑技术"成立;还可以看到:r_{19} 显示的截断和 r_{20} 显示的拼接都是错误的.

表 6 中的结果显示:TW-MSD 加法器与 SJ-MSD 加法器均可用于加法计算,且运算相同数据,SJ-MSD 加法器占据的处理器位明显较少.

7 结论

三值光学计算机采用 MSD 加法器,不仅实现了并行加/减法运算,而且由于加法器位与位之间的无关联性决定了处理器的位与位之间也没有关联性. 于是处理器位可以被任意分组使用,使得处理器位数众多的优势得以充分发挥.

与 TW-MSD 加法器相比,本文介绍的 SJ-MSD 加法器占用的处理器硬件资源节约 25%. 此外,在将多个小数据拼接成一个大数据时,SJ-MSD 加法器的夹单 0 拼接技术也能节约处理器资源. 这些优势大大提高了三值光学处理器的使用效率,有效地提高了三值光学计算机系统的整体性能. SJ-MSD 并行加法器的设计与实现为不断优化三值光学计算机的硬件结构提供了很好的范例.

参考文献

[1] Avizienis A, Signed digit number representation for fast parallel arithmetic[J]. IRE Transactions on Electronic Computers EC, 1961, 10(3): 389 – 400.

[2] Bocker R P, Drake B L, Lasher M E, Henderson T B, Modified signed-digit addition and subtraction using optical symbolic substitution[J]. Applied Optics, 1986, 25(15): 2456 – 2457.

[3] PENG Jun-jie, SHEN Rong, JIN Yi, SHEN Yun-fu, LUO Sheng. Design and implementation of modified signed-digit adder[J]. IEEE Transactions on Computers, 2014, 63(5): 1134 – 1143.

[4] Ha B, Li Y. Parallel modified signed-digit arithmetic using an optoelectronic shared content-addressable-memory processor[J]. Applied Optics, 1994, 33(17): 3647 – 3662.

[5] Casasent D, Woodford P. Symbolic substitution modified signed-digit optical adder[J]. Applied Optics, 1994, 33(8): 1498 – 1506.

[6] Cherri A K, Karim M A. Modified signed-digit arithmetic using an efficient symbolic substitution[J]. Applied Optics, 1988, 27(18): 3824 – 3827.

[7] Yao Li, George Eichmann. Conditional symbolic modified signed-digit arithmetic using optical content-addressable memory logic elements[J]. Applied Optics, 1987, 26(12): 2328 – 2333.

[8] Cherri A K, Symmetrically recoded modified signed-digit

optical addition and subtraction[J]. Applied Optics,1994,33(20):4378 – 4882.

[9] LI Guo-qiang,QIAN Feng,RUAN Hao, LIU Li-ren,Compact parallel optical modified-signed-digit arithmetic-logic array processor with electron-trapping device[J]. Applied Optics,1999,38(23):5039 – 5045.

[10] QIAN Feng,LI Guo-qiang,RUAN Hao,JING Hong-mei, LIU Li-ren. Two-step digit-set-restricted modified signed-digit addition-subtraction algorithmetic and its optoelectornic implementation[J]. Applied Optics,1999,38(26):5621 – 5630.

[11] Fyath R S, Alsaffar A A, Alam M S. Optical two-step modified signed-digit addition based on binary logic gates [J]. Optics Communications,2002,208:263 – 273.

[12] HUANG Hong-xin,Masahide Itoh,Toyohiko Yatagai. Classified one-step modified signed-digit arithmetic and its optical implementation[J]. Optical Engineering, 1996, 35: 1134 – 1140.

[13] Cherri, Abdallah K. Efficient optical negabinary modified signed-digit arithmetic:one-step addition and subtraction algorithms[J]. Optical Engineering,2004,43(2):420 – 425.

[14] 沈云付,潘磊,金翊等. 三值光学计算机一种限制输入一步式 MSD 加法器[J]. 中国科学. 信息科学,2012,42(7):869 – 881.
SHEN Yun-fu, PAN Lei, JIN Yi, et al. One-step binary MSD adder for ternary optical computer[J]. Sci China Inf Sci,2012,42(7):869 – 881. (in Chinese)

[15] Shen Yun-fu,Jiang Ben-peng,Jin Yi,et al. Principle and design of ternary optical accumulator implementing M-k-B addition[J]. Optical Engineering,2014,53(9):095108-1 ~8.

[16] 王宏健,金翊,欧阳山. 一位可重构三值光学处理器的设计和实现[J]. 计算机学报,2014,37(7):1500 – 1507.
WANG Hong-jian, JIN Yi, OUYANG Shan. Design and implementation of a 1-bit reconfigurable ternary optical processor[J]. Chinese Journal of computers, 2014, 37(7):1500 – 1507. (in Chinese)

[17] 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.
JIN Yi,SHEN Yun-fu,PENG Jun-jie,et al. Principles and construction of MSD adder in ternary optical computer[J]. Sci China Inf Sci,2011,41(5):541 – 551. (in Chinese)

[18] 金翊,王宏健,欧阳山,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):778 – 788.
JIN Yi,WANG Hong-jian,OUYANG Shan,et al. Principles,structures,and implementation of reconfigurable ternary optical processors[J]. Sci China Inf Sci,2012,42(6):778 – 788. (in Chinese)

[19] 江家宝,陈迅雷,欧阳山. 三值光计算机 MSD 转标准二进制数的硬件实现[J]. 南京理工大学学报(自然科学版),2016,40(3):278 – 284.
JIANG Jia-bao, CHEN Xun-lei, OUYANG Shan. Hardware implementation of converting ternary optical computer MSD into standard binary data[J]. Journal of Nanjing University of Science and Technology,2016,40(3):278 – 284. (in Chinese)

[20] YAN Jun-yong, JIN Yi, ZUO Kai-zhong. Decrease-radix design principle for carrying/borrowing free multi-valued and application in ternary optical computer[J]. Sci China Ser F-Inf Sci,2008,51(10):1415 – 1426. (in Chinese)

[21] JIN Yi, SHEN Yun-fu, OUYANG Shan, PENG Jun-jie, ZHANG Jun-jie, WANG Hong-jian. MSD parallel adder and its construction method based on ternary logic operator[P]. PCT International Patent: SHZPCT029-1, 2020-07-28.

作者简介

江家宝　男,1968 年生于安徽无为,副教授,现在上海大学计算机工程与科学学院攻读博士学位. 主要研究方向为三值光学计算机和嵌入式系
E-mail:jjb15820109@163.com

张晓峰　男,1979 年生于湖北襄阳,教授,863 专家组成员,2005 年在哈工大获博士学位. 现在中国航天科学与工业集团第 25 研究所工作. 主要研究方向为精确制导和毫米工程.

沈云付　男,1960 年生于浙江平湖,博士,教授,现在上海大学工作. 主要研究方向为软件形式化、模型检验、计算技术、三值光学计算机及其可靠性.

金　翊(通信作者)　男,1957 年生于陕西西安,教授,博导. 2003 年在西北理工大学获计算机科学博士学位. 现在上海大学工作. 主要研究方向为三值光学计算机和计算机体系结构.