·体系结构与软件技术·　　文章编号：1000-3428(2023)01-0201-09　　文献标志码：A　　中图分类号：TP331.2

基于互补电阻开关的忆阻乘法器设计

李志刚，陈　辉，刘　鹏，武继刚
（广东工业大学 计算机学院，广州 510006）

摘　要：现有的忆阻算术逻辑多采用单个忆阻器作为存储单元，在忆阻交叉阵列中易受到漏电流以及设计逻辑电路时逻辑综合复杂度高的影响，导致当前乘法器设计中串行化加法操作的延时和面积开销增加。互补电阻开关具有可重构逻辑电路的运算速度和抑制忆阻交叉阵列中漏电流的性能，是实现忆阻算术逻辑的关键器件。提出一种弱进位依赖的忆阻乘法器。为提升忆阻器的逻辑性能，基于互补电阻开关电路结构，设计两种加法器的优化方案，简化操作步骤。在此基础上，通过改进传统的乘法实现方式，并对进位数据进行拆解，降低运算过程中进位数据之间的依赖性，实现并行化的加法运算。将设计的乘法器映射到混合CMOS/crossbar结构中，乘法计算性能得到大幅提高。在Spice仿真环境下验证所提乘法器的可行性。仿真实验结果表明，与现有的乘法器相比，所提乘法器的延时开销从$O(n^2)$降低为线性级别，同时面积开销降低约70%。

关键词：忆阻器；互补电阻开关；混合CMOS/crossbar结构；加法器；乘法器

开放科学（资源服务）标志码（OSID）：

中文引用格式：李志刚，陈辉，刘鹏，等.基于互补电阻开关的忆阻乘法器设计[J].计算机工程，2023，49(1)：201-209.
英文引用格式：LI Z G，CHEN H，LIU P，et al.Design of memristive multiplier based on complementary resistive switch[J].Computer Engineering，2023，49(1)：201-209.

Design of Memristive Multiplier Based on Complementary Resistive Switch

LI Zhigang，CHEN Hui，LIU Peng，WU Jigang
（School of Computer and Technology，Guangdong University of Technology，Guangzhou 510006，China）

【Abstract】The existing memristive arithmetic logic primarily uses a single memristor as the storage unit，which is vulnerable to leakage currents in the memristor cross array and the high complexity of logic synthesis in a logic circuit design；consequently，increased delays and area overheads will be indicated in the serial addition operation of the current multiplier design.A Complementary Resistive Switch（CRS）is a key device for realizing the memristive arithmetic logic as it can reconfigure the logic circuit operation speed and restrains leakage currents in a memristor cross array. A memristive multiplier with weak carry dependency is proposed herein. To improve the logic performance of the memristor，based on the circuit structure of the CRS，two types of optimization schemes for the adder are designed to simplify the operation steps. Subsequently，by improving the classical multiplication method and disassembling the carry data，the dependency between carry data during the operation is reduced，and a parallel addition operation is realized.By mapping the designed multiplier to a hybrid CMOS/crossbar structure，the performance of multiplication computation is improved significantly. The feasibility of the proposed multiplier is verified in a Spice simulation environment. The simulation results show that compared with the existing multiplier，the proposed multiplier reduces the delay cost from $O(n^2)$ to the linear level and reduces the area cost by approximately 70%.

【Key words】memristor；Complementary Resistive Switch（CRS）；hybrid CMOS/crossbar structure；adder；multiplier
DOI：10. 19678/j. issn. 1000-3428. 0064367

0　概述

传统的冯诺依曼体系结构分别在存储器和中央处理单元进行数据存储和计算。因此，物理上分离的存储器与中央处理单元之间需要进行频繁的数据通信，导致性能开销和能耗增加。为提高计算机性能，研究人员探索利用新型非易失性纳米器件构建“非冯诺依曼”架构的新领域[1-2]。忆阻器具有存算

基金项目：国家自然科学基金“忆阻交叉阵列高质量低费用测试方法研究”（62174038）；广东省基础与应用基础研究基金“利用多种读写机制和忆阻器逻辑设计的阻性存储器高速测试方法研究”（2019A1515110284）。

作者简介：李志刚（1998—），男，硕士研究生，主研方向为高性能体系结构、数字电路设计；陈　辉，副教授；刘　鹏，讲师、博士；武继刚，教授、博士。

收稿日期：2022-04-02　　**修回日期：**2022-07-05　　**E-mail：**2929478121@qq.com

一体、尺寸小、低功耗等多种特性[3]，成为“非冯诺依曼”架构的关键器件[4]。近年来，基于忆阻器的数字逻辑电路设计被广泛应用于人工神经网络以及通信电路设计等领域。但关于忆阻乘法器的研究多数通过传统的部分乘积算法来实现，存在串行化进位比特问题，导致延时和面积开销增大。

为解决该问题，本文提出一种弱进位依赖的忆阻乘法器，通过对乘法运算中的进位比特进行拆解，减弱计算过程中的进位依赖，使得加法操作并行化，最终实现在线性时间内的忆阻乘法器。利用互补电阻开关（Complementary Resistive Switch，CRS）的读出操作破坏所存储逻辑状态的特性，设计一种基于忆阻器的部分乘积运算方式，为实现忆阻乘法器奠定基础。通过对传统的TC加法器和PC加法器进行优化，减少加法运算的延时和面积开销。提出一种基于互补电阻开关的可并行实现的乘法运算方案，并将其映射到混合CMOS/crossbar阵列结构之中，优化忆阻乘法器的延时和面积开销。

1 相关工作

忆阻器被广泛应用在数字逻辑设计领域中。赵毅等基于互补电阻开关提出一种可重构的忆阻器逻辑设计方法，实现了与、或、非蕴含以及异或四种基本逻辑门，并设计2-1和4-1多路复用器电路[5]。2-1多路复用器需要3个忆阻器通过2个步骤实现。4-1多路复用器需要6个忆阻器通过5个步骤实现。李志刚等[6]在择多-非-图的基础上，利用忆阻器设计延时优化和面积优化加法器。延时优化的一位加法器需要4个忆阻器通过4个步骤实现。面积优化的一位加法器需要3个忆阻器通过5个步骤实现。SIEMON等[7]从不同的优化角度，利用互补电阻开关设计TC加法器和PC加法器。针对面积开销性能，N位TC加法器利用N+2个存储单元通过4N+5个步骤实现。针对延时开销性能，N位PC加法器利用2(N+1)个存储单元并通过2(N+1)+2个步骤实现。

在传统的二进制乘法器中，乘法运算可分为部分乘积计算、部分乘积位移以及部分乘积累加共3个步骤[8]。HAJ-ALI等根据不同的功能将忆阻交叉阵列中的一行存储单元划分为多个区域，并利用TALATI等[9]提出的加法器实现乘法运算[8]。该乘法器可以在一行或一列中进行并行运算。而乘法运算的方式仍然采用部分乘积算法，导致单次乘法运算的延时增长。IMANI等[10]提出一种快速加法器和近似计算存内架构，该架构利用所提的快速加法器，以降低一定的精度为前提来加快乘法器的运算速度。TAHERINEJAD等[11]提出的半串行加法器结构。RADAKOVITS等[12]利用文献[11]所提的半串行加法器结构进行半串行乘法的运算。对于N位乘法运算，该乘法器将2N−1比特的半串行加法器电路构建$\lceil n/2 \rceil$次。在半串行加法器之间并行计算两个部分乘积，并进行$\lceil \mathrm{lb}n \rceil$次并行加法以实现乘法运算。ALAM等设计的乘法器依赖于确定性随机计算算法。该算法将乘数和被乘数转换成特定的二进制比特流，并利用时钟分频技术对二进制比特流进行逻辑与运算来完成乘法操作[13]。该算法利用3×2^{2n}个存储单元通过3个步骤来实现。WANG等在择多-非-图的基础上设计一种利用3个忆阻器通过4个步骤实现的全加器，在此基础上，进一步利用Wallace树实现一种4 bit乘法器。该乘法器需要51个忆阻器并通过64个步骤完成运算[14]。GUCKERT等在实质蕴涵操作[15]和MAD（Memristors-as-Drivers）逻辑门[16]基础上，采用忆阻器和传统CMOS管相结合的方式分别实现了两种忆阻乘法器[17]，并与传统的CMOS乘法器相比，两种忆阻乘法器在延时和面积开销方面均有所优化。HAI等[18]将数字量与模拟量相混合，设计一种用于支持矩阵乘法操作的浮点乘法器，以加快卷积神经网络的训练速度。

上述所实现的乘法操作大多是在加法器的基础上采用传统的部分乘积算法实现。而在传统的部分乘积算法中，乘法运算过程中的多个进位比特处于不同的时钟周期，并且在同一时钟周期仅利用一位进位比特。因此，现有的忆阻乘法器存在串行化加法操作，导致延时和面积开销增大。

2 忆阻器与互补电阻开关

忆阻器是具有可变电阻的双端无源器件。忆阻器的电阻值取决于两端施加的电压。忆阻器与互补电阻开关的电路结构及伏安特性曲线如图1所示。忆阻器的器件符号和伏安特性曲线如图1(a)所示。在忆阻器两端施加正电压V_{reset}时，忆阻器的阻值状态会切换为高电阻状态（High Resistance State，HRS），此时对应的电阻值为R_{off}，表示的逻辑状态为逻辑0。在忆阻器两端施加负电压V_{set}时，忆阻器的阻值状态会切换为低电阻状态（Low Resistance State，LRS），对应电阻值为R_{on}，表示的逻辑状态为逻辑1[19]。忆阻器具有尺寸小、运算速度快、非易失性等特点，一般采用交叉阵列的形式构建。但无源忆阻交叉阵列存在漏电流问题，导致数据读取错误[9]。

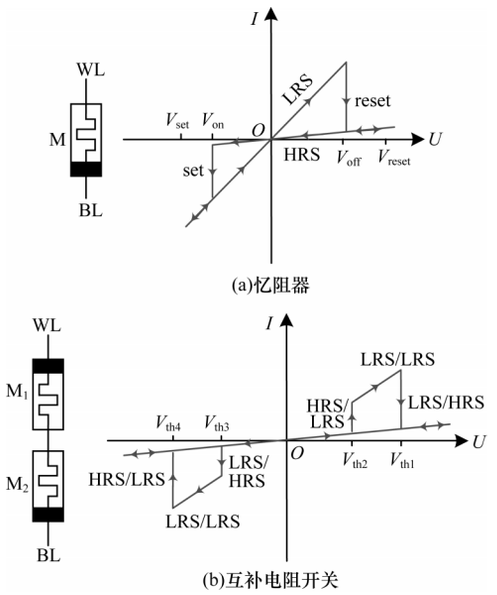


图1　忆阻器与互补电阻开关的电路结构及伏安特性曲线

Fig.1　Circuit structure and volt ampere characteristic curves of memristor and complementary resistive switch

为解决无源忆阻交叉阵列中漏电流的问题,LINN等提出CRS结构[20]。CRS的电路结构和伏安特性曲线如图1(b)所示。忆阻器M_1和忆阻器M_2的阻值状态组合表示CRS的逻辑状态。LRS/HRS阻值状态和施加在CRS两端的高电压V_{th1}=max{2|V_{set}|,2V_{reset}}表示逻辑状态1,HRS/LRS或接地表示逻辑状态0。在CRS两端施加V_{th1}=max{2|V_{set}|,2V_{reset}}的正电压时,CRS的阻值状态将切换为LRS/HRS。在CRS两端施加V_{th4}=min{−2|V_{set}|,−2V_{reset}}的负电压时,CRS的阻值状态将切换为HRS/LRS。在CRS两端施加的电压在V_{th2}~V_{th3}之间时,CRS的阻值状态不会改变。

当CRS处于稳定的逻辑状态时,电阻值总为高阻,施加幅值较小的忆阻器读出电压,无法区分所存储的逻辑值。因此,CRS的读出操作是在字线(Word Line,WL)端施加大于V_{th1}的电压信号,同时以位线(Bit Line,BL)端接地的方式实现。从图1(b)可以看出,在CRS两端施加大于V_{th1}的电压信号时,若在输出中检测到电流脉冲,则表示该CRS的阻值状态为HRS/LRS,即逻辑0。若在电流输出中无法检测到电流脉冲,则表明CRS的阻值状态为LRS/HRS,即逻辑1。在此读出过程中,CRS的阻值状态会被置为逻辑1,所存储的逻辑值有可能被破坏[7]。

CRS的逻辑转换机制由有限状态机表示。CRS有限状态机示意图如图2所示。

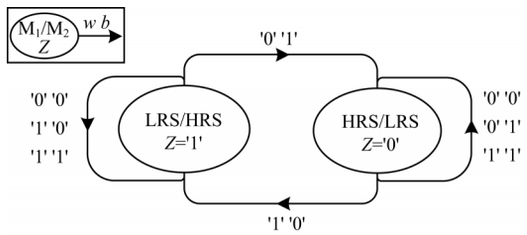


图2　互补电阻开关有限状态机示意图

Fig.2　Schematic diagram of finite state machine of complementary resistive switch

CRS的逻辑表达如式(1)所示:

$$Z=(w\mathrm{RIMP}b)Z'+(w\mathrm{NIMP}b)\overline{Z'} \tag{1}$$

其中:w和b分别为连接CRS字线和位线的电压信号所对应的逻辑变量;Z'和Z分别为施加电压信号前和施加电压信号后的逻辑状态;RIMP和NIMP分别为实质蕴含逻辑的求逆和取反。RIMP和NIMP的逻辑表达[21]如式(2)和式(3)所示:

$$p\mathrm{RIMP}q=p+\overline{q} \tag{2}$$

$$p\mathrm{NIMP}q=p\cdot\overline{q} \tag{3}$$

LINN等进一步利用CRS实现了14种基本逻辑[21]。由CRS的逻辑表达式可知,当w为逻辑1,b为逻辑0时,CRS的逻辑状态将被置为逻辑1,实现了TRUE逻辑操作:

$$Z=(1\mathrm{RIMP}0)Z'+(1\mathrm{NIMP}0)\overline{Z'}=Z'+\overline{Z'}=1 \tag{4}$$

将CRS置为逻辑状态1后,令w为逻辑q,b为逻辑1,其逻辑状态将被置为逻辑q:

$$Z=(q\mathrm{RIMP}1)\cdot 1+(q\mathrm{NIMP}1)\cdot\overline{1}=q+0=q \tag{5}$$

将CRS置为逻辑状态0后,通过将w设置为逻辑1,b设置为逻辑$\overline{q}$,也可实现置q逻辑操作:

$$Z=(1\mathrm{RIMP}\overline{q})\cdot 0+(1\mathrm{NIMP}\overline{q})\cdot\overline{0}=0+q=q \tag{6}$$

BREUER等[22]通过简化加法器公式实现基于互补电阻开关的全加器。全加器表示如式(7)~式(9)所示:

$$C_{i+1}=(a\mathrm{RIMP}\,\overline{b})C_i+(a\mathrm{NIMP}\,\overline{b})\,\overline{C_i} \tag{7}$$

$$S_i'=(a\mathrm{RIMP}b)C_i+(a\mathrm{NIMP}b)\,\overline{C_i} \tag{8}$$

$$S_i=(b\mathrm{RIMP}C_{i+1})S_i'+(b\mathrm{NIMP}C_{i+1})\,\overline{S_i'} \tag{9}$$

其中:a、b和C_i为一位全加器的输入;S_i'和S_i分别为中间和最终求和输出值;C_{i+1}为进位输出值。

3　乘法器设计

本文针对延时开销和面积开销,在TC加法器和PC加法器优化方案基础上,利用CRS构建乘法器电路,并将其映射到混合CMOS/crossbar结构中。混合CMOS/crossbar结构如图3所示。该结构由多个阵列和一个控制单元组成。控制单元用于协调每个阵列并将信号寻址到特定的字线和位线。

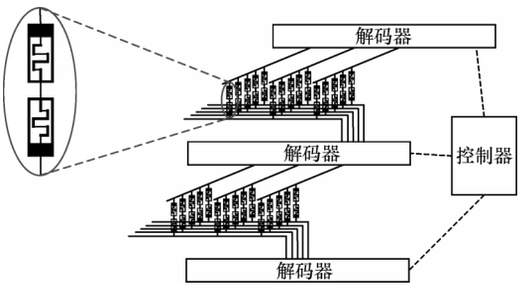


图3 混合CMOS/crossbar结构

Fig.3 Structure of hybrid CMOS/crossbar

3.1 加法器优化

加法器是乘法器运算过程中必不可少的一部分。加法器的优化有利于降低乘法器的延时开销和面积开销。在传统的TC加法器和PC加法器中,包括最高输出位在内的所有结果均在C_{i+1}和s_i'的基础上运算得到。然而,加法器的最高输出位运算结果本质上是进位比特,其运算过程并不需要计算s_i',只计算C_{i+1}。因此,本文通过简化最高输出位的运算步骤来优化TC加法器和PC加法器。

优化后的一位TC加法器的电路结构如图4所示,其中$A_0(1)$表示在阵列0中的第1个存储单元。优化后的一位TC加法器的实现步骤如表1所示。

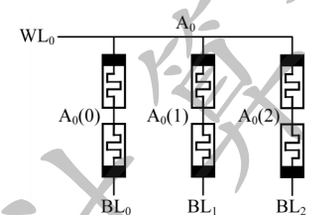


图4 优化后一位TC加法器电路结构

Fig.4 The circuit structure of optimized 1-bit TC adder

表1 优化的一位TC加法器实现步骤

Table 1 Implementation steps of optimized 1-bit TC adder

步骤	目标	逻辑操作	CRS逻辑状态
1	初始化	WL_0=1,$BL_{0\text{-}2}$=0	$A_0(0),A_0(1),A_0(2)$=1
2	计算C_0	$WL_0=C_0$,$BL_{0\text{-}2}$=1	$A_0(0),A_0(1),A_0(2)=C_0$
3	计算S_0'和C_1	$WL_0=a$,$BL_1=b$ $BL_0,BL_2=\bar{b}$	$A_0(0),A_0(2)=C_1$ $A_0(1)=S_0'$
4	读出	WL_0=1,BL_2=0	—
5	计算S_0	$WL_0=b$,$BL_1=C_1$	$A_0(1)=S_0$

从表1可以看出,优化后的TC加法器的实现步骤主要有初始化操作、C_0的计算、C_1和S_0'的计算、读出操作以及最终S_0的运算。由式(4)可知,在初始化操作中,将代表逻辑1的高电压信号V_{th1}施加在WL_0上,同时将$BL_{0\text{-}2}$三个端口接地,$A_0(0)$、$A_0(1)$、$A_0(2)$三个CRS被置为逻辑1。为了将C_0写入到$A_0(0)$、$A_0(1)$、$A_0(2)$中,根据式(5)可知,表示C_0和逻辑1的电压被分别施加在WL_0和$BL_{0\text{-}2}$上。根据式(1)和式(8)可知,在WL_0和BL_1上分别施加与逻辑a和逻辑b对应的电压值,运算结果S_0'将以阻值的形式存储在$A_0(1)$中。同时,根据式(1)和式(7)可知,在WL_0上施加与逻辑a对应的电压信号基础上,BL_0和BL_2上施加与逻辑$\bar{b}$对应的电压值,得到两个逻辑运算结果C_1,并分别存储于$A_0(0)$和$A_0(2)$中。其中存储于$A_0(0)$中的C_1作为最终运算结果保存,而存储于$A_0(2)$中的C_1则作为步骤5的输入。为了利用$A_0(2)$中的C_1,读出操作将读出C_1的阻值。在S_0的运算过程中,根据式(9),在WL_0和BL_1上分别施加与逻辑b和逻辑C_1对应的电压信号,得到逻辑运算结果S_0并存储于$A_0(1)$中。至此,优化后的一位TC加法器完成加法操作,C_1和S_0的逻辑值存储在$A_0(0)$和$A_0(1)$中。

优化后的一位PC加法器的电路结构和实现步骤分别如图5和表2所示。PC加法器的前三个步骤与TC加法器相似,唯一不同之处在于PC加法器采用两个阵列,需要在两个WL端施加对应的电压值。PC加法器最关键之处在于其步骤4利用两个阵列的并行性来同时完成C_1的读出操作和S_0的运算,减少了一个执行步骤。

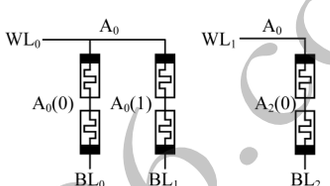


图5 优化后一位PC加法器电路结构

Fig.5 The circuit structure of optimized 1-bit PC adder

表2 优化的一位PC加法器实现步骤

Table 2 Implementation steps of optimized 1-bit PC adder

步骤	目标	逻辑操作	CRS逻辑状态
1	初始化	$WL_{0\text{-}1}$=1,$BL_{0\text{-}1}$=0 BL_2=0	$A_0(0),A_0(1),A_1(0)$=1
2	计算C_0	$WL_{0\text{-}1}=C_0$,$BL_{0\text{-}2}$=1	$A_0(0),A_0(1),A_1(0)=C_0$
3	计算S_0'和C_1	$WL_{0\text{-}1}=a$,$BL_1=b$, $BL_0,BL_2=\bar{b}$	$A_0(0),A_1(0)=C_1$ $A_0(1)=S_0'$
4	计算S_0	$WL_0=b$,WL_1=1 $BL_1=C_1$,BL_2=0	$A_0(1)=S_0$

3.2 乘法器

N位乘法运算产生$2N$位的运算结果,其中包含了大量的加法运算。如何优化乘法运算的加法过程尤为重要。在传统的乘法计算过程中,只有完成低位的加法运算后,才能产生相应的进位比特并开始高位的加法运算过程,极大地限制了计算速度。为了提高乘法运算中加法的并行性,本文基于优化的TC加法器和PC加法器,提出一种弱进位依赖性乘法器。该乘法器基于混合CMOS/crossbar结构的并行能力实现N位乘法运算。

在乘法运算中首要步骤是计算部分乘积。部分乘积由两比特的“与”逻辑运算得到。“与”逻辑是利用一个CRS通过3个步骤实现。部分乘积计算的实现步骤如表3所示。

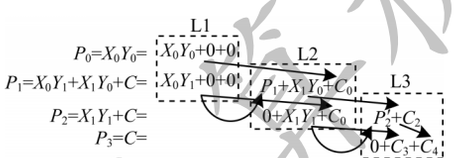
表3　部分乘积计算的实现步骤

Table 3　Implementation steps of partial product calculation

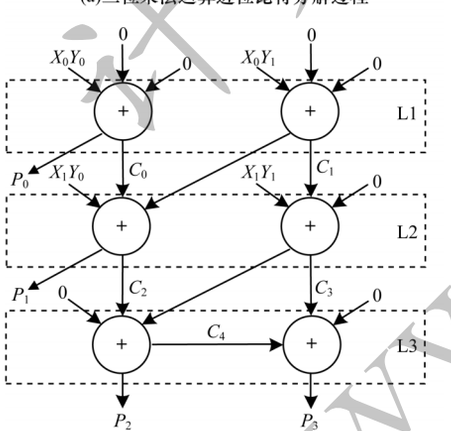
步骤	目标	逻辑操作	CRS逻辑状态
1	初始化	WL=1,BL=0	1
2	计算p	WL=p,BL=1	p
3	计算pq	WL=q,BL=1	pq

首先,在WL和BL上分别施加与逻辑1和逻辑0对应的电压信号,CRS被初始化为逻辑1;其次,在WL和BL上分别施加与逻辑p和逻辑1对应的电压信号,CRS被置为逻辑p;最后,在WL和BL上分别施加与逻辑q和逻辑1对应的电压信号,得到pq运算结果并存储于CRS中。

乘法运算的后续步骤是对部分乘积进行加法运算。与传统乘法运算不同,本文采用分解一位进位比特为多位进位比特求和的方式来减弱乘法运算过程中的数据依赖性,使乘法运算中的加法运算过程得以并行化,加快乘法运算速度。图6所示为二位乘法运算的进位比特分解过程以及对应的数据依赖图。圆圈表示一个三输入和二输出的加法器。指向加法器的箭头表示加法器所需的输入值,由加法器指出的箭头表示加法器的输出值。



(a)二位乘法运算进位比特分解过程



(b)二位乘法运算进位比特依赖图

图6　二位乘法器数据依赖图

Fig.6　The data dependency graph of 2-bit multiplier

从图6(b)可以看出,在完成部分乘积运算后,二位乘法器的弱化数据依赖图共有三层运算过程,包含了两层并行加法运算(L1和L2)以及一层串行加法运算(L3)。在L3中的两个加法器之间由于存在C_4的进位依赖,因此必须串行实现。此外,每一层并行计算得到的中间计算结果用于下一层的计算。经过L1层和L2层的并行加法后将得到二位乘法运算结果的低2位P_0和P_1。L3层的串行加法运算将得到乘法运算结果的高2位P_2和P_3。

本文采用混合CMOS/crossbar结构实现所提的弱进位依赖性乘法器。二位乘法器的电路结构如图7所示。

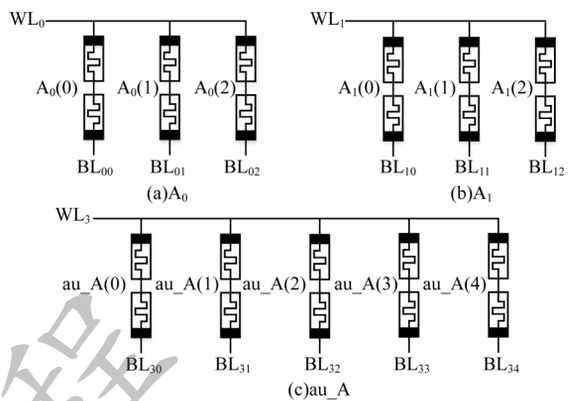


图7　二位乘法器电路结构

Fig.7　The circuit structure of 2-bit multiplier

从图7可以看出,A_0(0-2)和A_1(0-2)表示用于运算加法的计算阵列,au_A(0-4)表示用于存储运算结果的辅助阵列。对于二位乘法器,每个计算阵列利用3个CRS来执行一位加法器,辅助阵列采用5个CRS来存储临时和最终的运算结果。

二位乘法器的流程图和具体执行步骤分别如图8和表4所示。

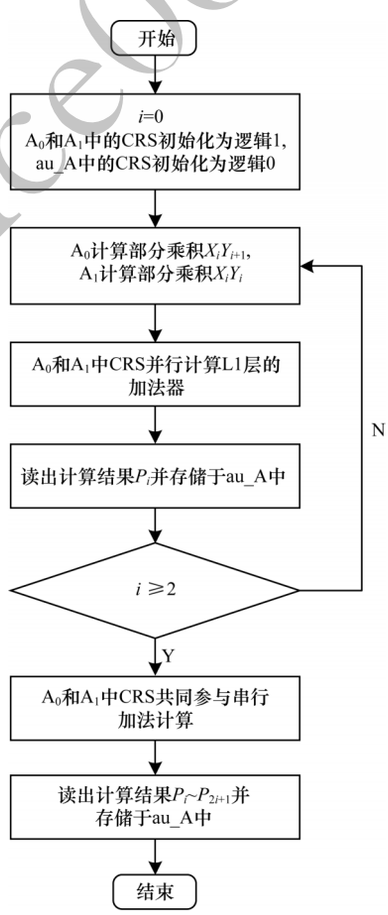


图8　二位乘法器实现流程

Fig.8　Implementation procedure of 2-bit multiplier

表4 二位乘法器实现步骤

Table 4 Implementation steps of 2-bit multiplier

步骤	CRS逻辑状态
1	$A_0(0\text{-}2)=1, A_1(0\text{-}2)=1$ au_A(0-4)=0
2	$A_0(0\text{-}2)=X_0, A_1(0\text{-}2)=X_0$
3	$A_0(0\text{-}2)=X_0Y_1$ $A_1(0\text{-}2)=X_0Y_0$
4	$A_0(0)=C_1, A_1(0)=C_0$
5	$A_0(1)=P_1', A_1(1)=P_0$
6	$A_0(2)=1, A_1(2)=1$
7	au_A(4)=P_0
8	$A_0(0\text{-}2)=X_1, A_1(0\text{-}2)=X_1$
9	$A_0(0\text{-}2)=X_1Y_1$, $A_1(0\text{-}2)=X_1Y_0$
10	$A_0(0)=C_3, A_1(0)=C_2$
11	$A_0(1)=P_2', A_1(1)=P_1$,
12	$A_0(2)=1, A_1(2)=1$
13	au_A(3)=P_1
14	$A_0(0\text{-}2)=0, A_1(0\text{-}2)=0$
15~18	$A_0(1)=P_2, A_0(2)=P_3, A_0(2)=C_4$
19	au_A(2)=P_2, au_A(1)=P_3

步骤1是初始化操作,将表示逻辑1的电压信号施加在$WL_{0\text{-}1}$上,$BL_{00\text{-}02}$以及$BL_{10\text{-}12}$接地,实现了$A_0(0\text{-}2)$和$A_1(0\text{-}2)$的置1操作。同时,为了完成au_A(0-4)的置0操作,外围电路将与逻辑1对应的电压信号施加在$BL_{20\text{-}24}$上,WL_2接地。步骤2和步骤3是在$BL_{00\text{-}02}$与$BL_{10\text{-}12}$端口均施加代表逻辑1的高电压,但在步骤2中$WL_{0\text{-}1}$施加代表X_0的电压值,而在步骤3中,WL_0和WL_1上分别施加代表Y_1和Y_0的电压值。当步骤3完成后,得到两个部分乘积X_0Y_1和X_0Y_0并分别存储于$A_0(0\text{-}2)$以及$A_1(0\text{-}2)$中。

由于本文提出的优化TC加法器和乘法运算过程中的加法操作均可以并行实现,因此步骤4~步骤6利用优化TC加法器进行L1层的并行加法。为进行L2层的并行加法运算,步骤7在$WL_{0\text{-}2}$上施加与逻辑1对应的电压信号,同时将$BL_{00\text{-}01}$以及$BL_{10\text{-}11}$接地,读出上一个步骤中的运算结果,其中包含结果P_0,并进一步利用反相器在BL_{24}上施加逻辑$\overline{P_0}$对应的电压信号,将P_0以阻值形式重新写入au_A(4)中。

步骤8~步骤13利用优化TC加法器完成L2层的并行加法运算,以得到逻辑运算结果P_1并存储于au_A(3)中。为了复用存储单元,需要$A_0(0\text{-}2)$和$A_1(0\text{-}2)$初始化为逻辑0,因此,在步骤14中将表示逻辑1的电压信号施加在$BL_{00\text{-}02}$以及$BL_{10\text{-}12}$上,$WL_{0\text{-}1}$接地。步骤15~步骤18则利用优化的PC加法器进行串行加法运算,最终得到逻辑运算结果P_2和P_3。步骤19是在$WL_{0\text{-}2}$上施加与逻辑1对应的电压信号,同时将$BL_{00\text{-}01}$以及$BL_{10\text{-}11}$接地,读出P_2和P_3并将其以电压形式施加到$BL_{21\text{-}22}$上,最终以阻值形式分别存储于au_A(1-2)中。

N位乘法运算的数据依赖图如图9所示,其执行步骤可由二位乘法运算的步骤扩展得到。在N位乘法运算中,每层并行加法产生$2N-1$位中间计算结果和一位最终乘法运算结果。为了在不同层之间重用存储单元,本文分别采用优化的TC加法器和优化的PC加法器进行乘法器中的并行加法运算和串行加法运算。因此,每一层的并行加法需要$3N$个存储单元,串行加法需要$2N+1$个存储单元。

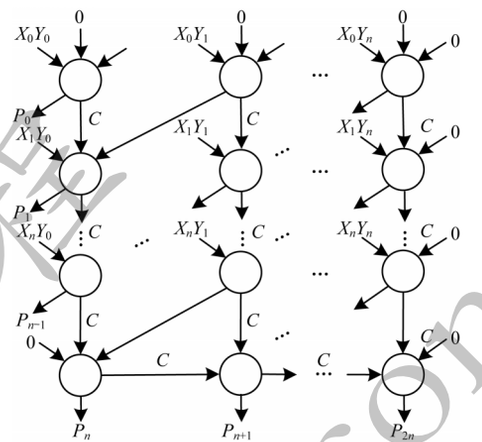


图9 N位乘法运算的数据依赖图

Fig.9 The data dependency graph of N-bit multiplier

第一层的并行操作需要7个步骤,其中包含步骤1~步骤3的部分乘积计算,步骤4~步骤6的TC加法器以及步骤7的读写操作。由于CRS的读出操作会将其逻辑状态重置为逻辑1,因此省略了在后续部分乘积运算中的第一步初始化步骤。N位乘法器的并行加法运算需要$6N+1$个步骤。在N位乘法器中采用优化的PC加法器实现的串行加法器需要$2N+2$个步骤。因此,本文所提的N位弱进位依赖性乘法器需要$8N+3$个步骤。此外,所需存储单元的数量将由计算阵列中的存储单元和辅助单元的总和决定。

本文所提的乘法器能够有效提升延时和面积开销性能,其原因为图6中乘法运算方案的并行性。本文对乘法运算过程中的进位比特进行分解,减弱了进位依赖性,通过混合CMOS/crossbar阵列结构的分阵列形式并行实现乘法运算过程中的串行加法操作,执行N个一位加法操作仅需要一位TC加法器,极大地优化了乘法器的延时开销。此外,本文分别采用一位TC加法器和N位PC加法器作为乘法运算过程中的并行和串行加法器,减少额外的存储空间,在每层加法运算过程中的CRS均可以复用,进一步减少面积开销。

4 仿真

研究人员提出多种忆阻器模型,包括Simmons隧道势垒模型[23]、非线性离子漂移模型[24]、阈值自适应忆阻器模型(TEAM)[25]、电压阈值自适应忆阻器模型(VTEAM)[26]等。VTEAM具有简单、通用、灵活的特点。因此,本文选择VTEAM模型进行仿真。

VTEAM模型的相关参数参考文献[26],具体如下:V_{off}=0.5 V,V_{on}=−0.5 V,R_{off}=2.5 kΩ,R_{on}=100 Ω,k_{off}=4.03×10^{-8}m/s,k_{on}=−80m/s,a_{off}=1,a_{on}=3,其中V_{off}和V_{on}为忆阻器的阈值电压,R_{off}和R_{on}为忆阻器的高阻和低阻,k_{off}、k_{on}、a_{off}以及a_{on}为VTEAM模型拟合Pt-Hf-Ti忆阻器实体器件后的常量参数。在此参数下VTEAM模型与Pt-Hf-Ti实体器件相拟合后的伏安特性曲线可参考文献[26]。

本文利用CRS的与逻辑实现了部分乘积。图10所示为所有一位输入组合下部分乘积计算的仿真结果。纵坐标表示忆阻器阻值。10 ns为一个时钟周期。从图10可以看出:当q=1、p=1时,M_1与M_2的阻值状态分别为LRS和HRS,代表输出逻辑1;对于其他输入情况,M_1与M_2的最终阻值状态分别为HRS和LRS,代表输出逻辑0,与理论分析相符。

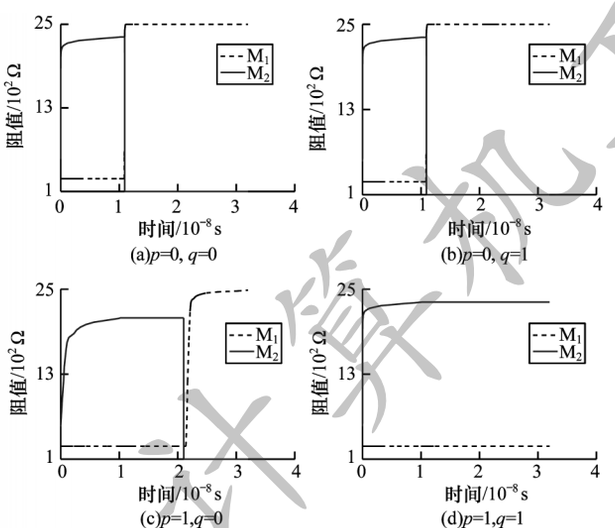


图10　部分乘积运算的仿真结果

Fig.10　Simulation results of partial product operation

图11所示为在a=0、b=1、C_i=1情况下优化的TC加法器的仿真结果。从图11可以看出,TC加法器仿真结果中A_0(0)和A_0(1)最终的阻值状态分别为LRS/HRS和HRS/LRS,代表运算结果为10。

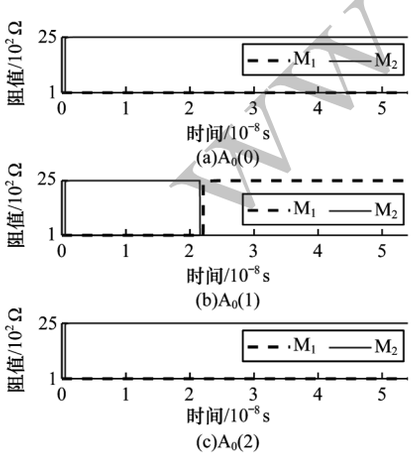


图11　一位TC加法器的仿真结果

Fig.11　Simulation results of 1-bit TC adder

图12所示为在a=00、b=11、C_i=01输入情况下优化的PC加法器的仿真结果。PC加法器仿真结果中A_0(0)、A_0(1)、A_0(2)最终的阻值状态分别为LRS/HRS、HRS/LRS、HRS/LRS,表示最终运算结果为100。两种加法器的仿真结果均与理论预期相符。

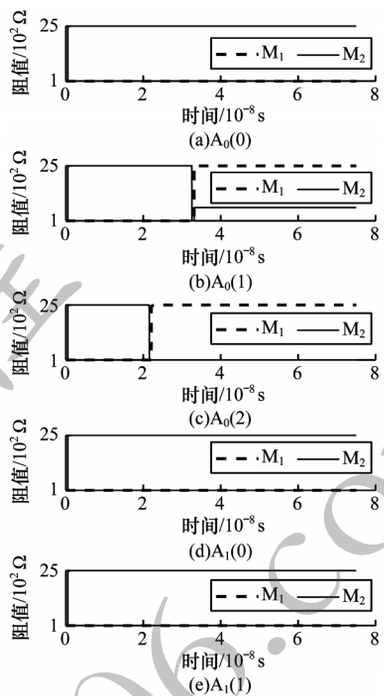


图12　二位PC加法器的仿真结果

Fig.12　Simulation results of 2-bit PC adder

本文验证输入为01与11情况下所提出的弱进位依赖乘法器的可行性。图13和图14分别所示为计算阵列A_0以及A_1的仿真结果。图15所示为辅助阵列au_A的仿真结果。从图15可以看出,二位乘法运算共需要19个步骤和11个CRS。辅助阵列中au_A(1-4)的仿真结果为0011,与理论相符。

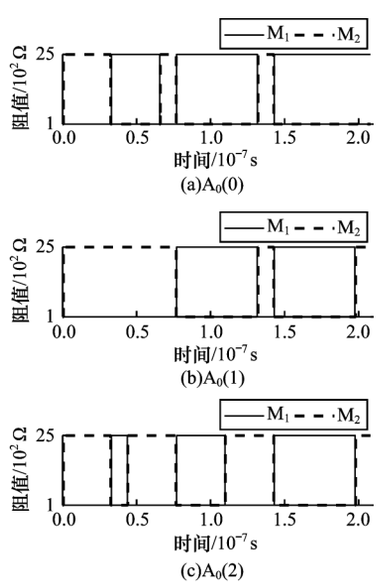


图13　在乘法器运算中A_0的仿真结果

Fig.13　Simulation results of A_0 in multiplier operation

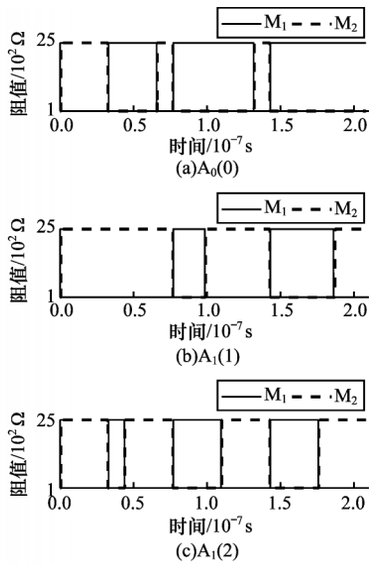


图14 在乘法器运算中A_1仿真结果

Fig.14 Simulation results of A_1 in multiplier operation

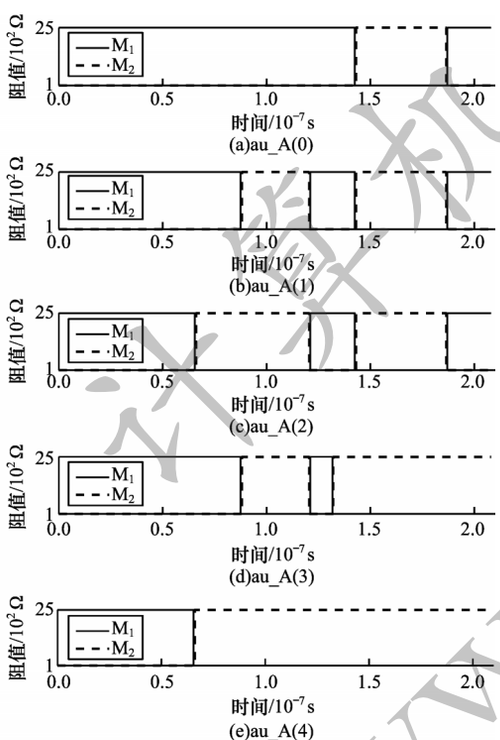


图15 在乘法器运算中au_A仿真结果

Fig.15 Simulation results of au_A in multiplier operation

表5所示为本文所设计的方法与其他方法的结果对比，延时表示N位乘法器的执行步骤个数，面积表示N位乘法器执行所需要的忆阻器个数。与文献[8]相比，本文的延时开销降低了一个数量级，并且面积开销降低了约70%。与文献[10]和文献[12]相比，本文所提的方法在面积和延时开销性能上均降低了一个数量级。文献[13]在延时开销方面为常数级别，但其所提的方法需要在阵列的外围中增加额外的CMOS晶体管电路，用于连接阵列的字线或位线，以构建MAGIC电路，增加了面积开销，并且该方法并未计算输入的乘法运算数据与特定的二进制比特流之间转换所需的延时开销和面积开销。同时，本文所提的方法在面积开销方面相比文献[13]方法降低到了线性级别。

表5 N位乘法器性能对比

Table 5 Performance comparison of N-bit multiplier

方法	延时开销	面积开销
文献[8]方法	$13n^2-14n+6$	$20n-5$
文献[10]方法	$15n^2-11n-1$	$15n^2-9n-1$
文献[12]方法	$\lceil \mathrm{lb}\ n \rceil(10n+2)+4n+2$	$2n^2+n+2$
文献[13]方法	3	3×2^{2n}
本文方法	$8n+3$	$6n-1$

5 结束语

本文针对现有忆阻乘法器设计的局限性，提出两种不同的加法器优化方案，并在此基础上设计一种乘法器。通过设计基于忆阻器的部分乘积运算，并对TC加法器和PC加法器进行优化，同时，针对乘法器的串行加法部分，通过减弱进位比特的数据依赖性，设计一种基于互补电阻开关的可并行实现的乘法运算方案。仿真实验结果表明，该乘法器减弱进位比特的数据依赖性，实现了并行乘法运算，与大部分现有乘法器相比，在延时和面积开销性能方面均有所提升。下一步将在本文工作的基础上，将忆阻加法器与忆阻乘法器相结合，研究忆阻浮点乘法器的实现方案，进一步完善忆阻器的复杂逻辑设计。

参考文献

[1] KVATINSKY S. Real processing-in-memory with memristive memory processing unit (mMPU)[C]// Proceedings of International Conference on Application-Specific Systems, Architectures and Processors. Washington D. C., USA: IEEE Press, 2019: 142-148.

[2] SEBASTIAN A, LE GALLO M, KHADDAM-ALJAMEH R, et al. Memory devices and applications for in-memory computing[J]. Nature Nanotechnology, 2020, 15(7): 529-544.

[3] 段飞腾，崔宝同. 基于忆阻器时滞神经网络的全局稳定性新判据[J]. 计算机工程，2015，41(7)：210-214.
DUAN F T, CUI B T. New criterion for global stability based on memristor time delay neural network[J]. Computer Engineering, 2015, 41(7): 210-214. (in Chinese)

[4] YANG J J, STRUKOV D B, STEWART D R. Memristive devices for computing[J]. Nature Nanotechnology, 2013, 8(1): 13-24.

[5] 赵毅，陈辉，刘鹏，等. 基于互补式忆阻器的多路复用器设计与实现[J]. 微电子学与计算机，2022，39(5)：96-103.
ZHAO Y, CHEN H, LIU P, et al. Design and

implementation of multiplexer based on complementary resistive switches[J]. Microelectronics & Computer,2022,39(5):96-103. (in Chinese)

[6] 李志刚,陈辉,刘鹏,等. 基于择多-非-图的忆阻加法器设计[J]. 微电子学与计算机,2022,39(5):104-110.
LI Z G,CHEN H,LIU P,et al. Memristive adder design based on majority-inverter-graph[J]. Microelectronics & Computer,2022,39(5):104-110. (in Chinese)

[7] SIEMON A,MENZEL S,WASER R,et al. A complementary resistive switch-based crossbar array adder[J]. IEEE Journal on Emerging and Selected Topics in Circuits and Systems,2015,5(1):64-74.

[8] HAJ-ALI A,BEN-HUR R,WALD N,et al. Efficient algorithms for in-memory fixed point multiplication using MAGIC[C]//Proceedings of International Symposium on Circuits and Systems. Washington D. C. ,USA:IEEE Press,2018:1-5.

[9] TALATI N,GUPTA S,MANE P,et al. Logic design within memristive memories using memristor-aided loGIC[J]. IEEE Transactions on Nanotechnology,2016,15(4):635-650.

[10] IMANI M,GUPTA S,ROSING T. Ultra-efficient processing in-memory for data intensive applications[C]// Proceedings of the 54th Annual Design Automation Conference. New York,USA:ACM Press,2017:1-6.

[11] TAHERINEJAD N,DELAROCHE T,RADAKOVITS D,et al. A semi-serial topology for compact and fast IMPLY-based memristive full adders[C]//Proceedings of the 17th International New Circuits and Systems Conference. New York,USA:ACM Press,2019:1-4.

[12] RADAKOVITS D,TAHERINEJAD N,CAI M Y,et al. A memristive multiplier using semi-serial IMPLY-based adder[J]. IEEE Transactions on Circuits and Systems I:Regular Papers,2020,67(5):1495-1506.

[13] ALAM M R,NAJAFI M H,TAHERINEJAD N. Exact in-memory multiplication based on deterministic stochastic computing[C]//Proceedings of International Symposium on Circuits and Systems. Washington D. C. ,USA:IEEE Press,2020:1-5.

[14] WANG Y M,LI Y,SHEN H H,et al. A few-step and low-cost memristor logic based on MIG logic for frequent-off instant-on circuits in IoT applications [J]. IEEE Transactions on Circuits and Systems II:Express Briefs,2019,66(4):662-666.

[15] KVATINSKY S,SATAT G,WALD N,et al. Memristor-based material implication (IMPLY) logic: design principles and methodologies[J]. IEEE Transactions on Very Large Scale Integration Systems,2014,22(10):2054-2066.

[16] GUCKERT L,SWARTZLANDER E E. MAD gates—memristor logic design using driver circuitry[J]. IEEE Transactions on Circuits and Systems II:Express Briefs,2017,64(2):171-175.

[17] GUCKERT L,SWARTZLANDER E E. Optimized memristor-based multipliers[J]. IEEE Transactions on Circuits and Systems I:Regular Papers,2017,64(2):373-385.

[18] JIN H,LIU C,LIU H K,et al. ReHy:a ReRAM-based digital/analog hybrid PIM architecture for accelerating CNN training[J]. IEEE Transactions on Parallel and Distributed Systems,2022,33(11):2872-2884.

[19] STRUKOV D B,SNIDER G S,STEWART D R,et al. The missing memristor found[J]. Nature,2008,453(7191):80-83.

[20] LINN E,ROSEZIN R,KÜGELER C,et al. Complementary resistive switches for passive nanocrossbar memories[J]. Nature Materials,2010,9(5):403-406.

[21] LINN E,ROSEZIN R,TAPPERTZHOFEN S,et al. Beyond von Neumann—logic operations in passive crossbar arrays alongside memory operations[J]. Nanotechnology,2012,23(30):305205.

[22] BREUER T,SIEMON A,LINN E,et al. A HfO2-based complementary switching crossbar adder[J]. Advanced Electronic Materials,2015,1(10):1-10.

[23] PICKETT M D,STRUKOV D B,BORGHETTI J L,et al. Switching dynamics in titanium dioxide memristive devices[J]. Journal of Applied Physics,2009,106(7):1-6.

[24] BIOLEK D,BIOLKOVÁ V,BIOLEK Z. SPICE model of memristor with nonlinear dopant drift[J]. Radioengineering,2009,18(2):210-214.

[25] KVATINSKY S,FRIEDMAN E G,KOLODNY A,et al. TEAM:ThrEshold adaptive memristor model[J]. IEEE Transactions on Circuits and Systems I:Regular Papers,2013,60(1):211-221.

[26] KVATINSKY S,RAMADAN M,FRIEDMAN E G,et al. VTEAM: a general model for voltage-controlled memristors [J]. IEEE Transactions on Circuits and Systems II:Express Briefs,2015,62(8):786-790.

编辑　薛晋栋