https://crad.ict.ac.cn
62(3): 620−632, 2025

面向低磨损存内计算的多状态逻辑门综合

赵安宁[1] 许 诺[1,2] 刘 康[1] 罗 莉[1,2] 潘炳征[1] 薄子怡[1] 谭承浩[1]

[1](国防科技大学计算机学院 长沙 410073)

[2](先进微处理器芯片与系统重点实验室(国防科技大学) 长沙 410073)

(zhaoanning@nudt.edu.cn)

The Synthesis of Multiple Stateful Logic Gates for In-memory Computing with Low Wear

Zhao Anning[1], Xu Nuo[1,2], Liu Kang[1], Luo Li[1,2], Pan Bingzheng[1], Bo Ziyi[1], and Tan Chenghao[1]

[1](*School of Computer Science and Technology*, *National University of Defense Technology*, *Changsha* 410073)

[2](*Key Laboratory of Advanced Microprocessor Chips and Systems* (*National University of Defense Technology*), *Changsha* 410073)

Abstract By merging the functions of Boolean logic and non-volatile memory, memristive stateful logic can achieve the real sense of in-memory computing through eliminating data movement during computation, which breaks the "memory wall" and "energy wall" of traditional von Neumann computing system. In recent years, a series of the memristor-based in-memory stateful logic gates have been proposed by linking the conditional switching process and mathematical logic function, whose functions cover multiple logic functions such as IMP, NAND, NOR, and NIMP etc. However, the automated synthesis and mapping method for implementing the in-memory complex stateful logic computation by cascading the stateful logic gates is still embryonic, especially lacking the investigations on the device wear, which limits the application of in-memory stateful logic in edge computing scenarios. To reduce the device wear (toggle rate) in a complex in-memory stateful logic computation process, we propose a stateful logic synthesis and mapping process based on multiple stateful logic gates for low-wear in-memory computing. Compared with the two state-of-art stateful logic synthesis and mapping tools of SIMPLER MAGIC and LOSSS, the proposed low-wear logic synthesis and mapping process achieves an average improvement of over 35.55% and 8.48% in the toggle rates respectively under the EPFL combinational benchmark circuits. Moreover, the proposed tool achieves an average improvement of over 47.26% and 6.72% in the toggle rates respectively under the LGSynth91 benchmark circuits.

Key words memristor; stateful logic; toggle rate; logic synthesis and mapping; low wear

摘 要 通过融合布尔逻辑和非易失存储的功能,忆阻状态逻辑电路可以消除计算过程中的数据移动,实现在存储器中计算,打破传统冯·诺依曼计算系统的"存储墙"和"能耗墙".近年来,通过构建条件转变到数学逻辑关系之间的映射,已经有一系列存内状态逻辑门被提出,功能覆盖 IMP, NAND, NOR, NIMP 等多个逻辑运算.然而,复杂计算过程到存内状态逻辑实现的自动化综合映射方法仍处于萌芽阶段,特别是缺少针对器件磨损的探讨,限制了设备维修不便的边缘计算场景应用.为降低复杂存内状态逻辑计算过

收稿日期: 2024-07-20; **修回日期**: 2025-01-17

基金项目: 国家自然科学基金项目(62202483); 湖南省自然科学基金项目(2022JJ40563); 武汉光电国家研究中心开放基金项目(2021WNLOKF019); PDL 开放基金项目(WDZC20235250112)

This work was supported by the National Natural Science Foundation of China (62202483), the Hunan Provincial Natural Science Foundation of China (2022JJ40563), the Open Project Program of Wuhan National Laboratory for Optoelectronics (2021WNLOKF019), and the Open Fund of PDL (WDZC20235250112).

通信作者: 许诺(xunuo@nudt.edu.cn)

程的磨损（翻转率），实现了一种面向低磨损存内计算的多状态逻辑门综合映射过程. 与领域内熟知的SIMPLER MAGIC 状态逻辑综合流程相比，该综合映射流程在复杂计算过程的翻转率上实现了对 EPFL，LGSynth91 的典型基准测试电路分别平均 35.55%，47.26% 以上的改进；与最新提出的 LOSSS 状态逻辑综合流程相比，在复杂计算过程的翻转率上实现了对 EPFL，LGSynth91 的典型基准测试电路分别平均 8.48%，6.72% 以上的改进.

关键词 忆阻器；状态逻辑；翻转率；逻辑综合与映射；低磨损

中图法分类号 TP333

DOI: 10.7544/issn1000-1239.202440627 **CSTR:** 32373.14.issn1000-1239.202440627

存储模块与计算模块相分离的冯·诺伊曼体系结构存在“存储墙”问题[1]，严重制约了处理器的性能提升，并伴随着较大的能量消耗. 为彻底突破该“瓶颈”，需要在体系结构层次上进行突破，研究新型存内计算架构[2-4]. 忆阻状态逻辑为存内计算提供了电路基础. 通过融合布尔逻辑和非易失性存储的功能，忆阻状态逻辑可以消除计算过程中的数据移动（消除访存延时和能耗），实现存储与计算的细粒度融合. 目前，通过理论推导（仿真）和实测实验已经有诸如 IMP, FALSE, NOR 等多个忆阻状态逻辑门得到验证，功能覆盖布尔逻辑完备集，为实现复杂逻辑计算提供了可行基础. 然而，从复杂计算功能到忆阻存储阵列内状态逻辑门级联序列转换的自动化设计研究仍处于萌芽阶段，一些挑战亟待解决.

一是“门单元类型单一”. 目前大多数针对复杂计算功能状态逻辑实现的研究都集中在使用功能完备的单个状态逻辑门级联，如 IMP, NOR, NAND 等，缺少对多个兼容状态逻辑门的使用，极大地限制了复杂状态逻辑计算过程的优化空间. 在综合策略中加入多种状态逻辑门，能够为复杂逻辑功能的实现提供更多的基本功能选择，有效减少最终的映射规模、操作数量以及执行延迟. 因此，有必要探索面向存内计算的多状态逻辑门综合映射方法.

二是“综合映射目标单一”. 当前大多数面向复杂计算功能的状态逻辑实现的研究皆以计算延时为优化目标，鲜有针对其他设计目标的探讨. 忆阻状态逻辑门实际是在外加电压控制下的“条件写”过程，根据相应的输入数据，门的成功执行必然伴随一次擦写过程（状态翻转）. 在当前的工艺成熟度下，实际忆阻器产品的可擦写次数相较于传统的动态随机存取存储器（DRAM）和静态随机存取存储器（SRAM）仍有不足. 在状态逻辑计算过程中，多次擦写可能导致器件磨损而失效，在设备维修维护不便的边缘计算场景下值得重点关注. 因此，有必要研究减少状态逻辑计算过程中器件磨损的方法来提高边缘计算设备的使用寿命，从而降低维护和更换成本.

针对以上 2 个问题，本文研究面向低磨损存内计算的多状态逻辑门综合，探索采用多种状态逻辑门的综合映射过程来降低复杂存内状态逻辑计算过程的翻转率.

1 相关工作

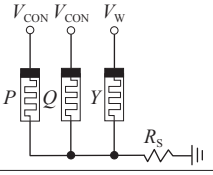
1.1 状态逻辑门

沿用 CMOS 逻辑中门的定义，状态逻辑门是用来表示基于忆阻器电路和相应的逻辑功能的概念[5]. 状态逻辑门操作过程中信息的载体为忆阻器的电阻状态，逻辑操作的过程即为各个忆阻器在电压激励下的条件转变过程. 这种由电阻代表逻辑信息，通过电阻变化过程来映射逻辑函数的逻辑实现体系被称为状态逻辑[6].

通常，用于构建状态逻辑门的忆阻器具有 2 个稳定的电阻状态，分别为高电阻状态（HRS，通常定义为逻辑 0）和低电阻状态（LRS，通常定义为逻辑 1）. 对于双极型忆阻器，高电阻状态和低电阻状态之间的转换是由极性相反的外部工作电压触发的，它们被称为 SET 和 RESET 电压[7]. 当让高电阻状态和低电阻状态分别表示数字信号 0 和 1 时，对忆阻器构成的电路施加一组特定的电压信号序列，忆阻器的状态信息会相应地发生改变，这样就可以在忆阻器初始状态和最终状态之间映射一个逻辑函数. 状态逻辑门通过外围电路施加的控制电压信号触发“条件写”过程，在操作后的输出器件与输入器件之间映射逻辑关系，完成布尔逻辑功能.

目前的研究工作已经通过改变电路结构或者针对同种电路结构来改变控制电压信号大小的方法实现了多种不同状态逻辑门，其逻辑功能覆盖 IMP, NOR, NAND, NOT 等完备集合. 本文沿用 Xu 等人[5] 使用的

状态逻辑门命名规则, 使用符号“结构-*N*-*n* 功能”表示一种状态逻辑门. 其中“结构”表示电路连接的类型, “*N*”表示逻辑输入的数量, “*n*”表示所包含的忆阻器件的数量, “功能”表示实现的逻辑功能. 图 1 中给出了 2 种典型的实现 NOR 功能的状态逻辑门结构, 分别为“PMR-two-3NOR”和“PMASM-two-3NOR”.

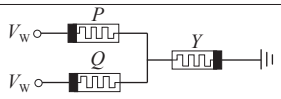


PMR-three-3ONOR

PMR-two-3NOR

输入			输出	输入			输出
P	*Q*	*Y*	*Y*′	*P*	*Q*	*Y*	*Y*′
0	0	0	1	0	0	1	1
0	1	0	0	0	1	1	1
1	0	0	0	1	0	1	1
1	1	0	0	1	1	1	1

(a) PMR结构



PMASM-three-3ANOR

PMASM-two-3NOR

输入			输出	输入			输出
P	*Q*	*Y*	*Y*′	*P*	*Q*	*Y*	*Y*′
0	0	1	1	0	0	0	0
0	1	1	0	0	1	0	0
1	0	1	0	1	0	0	0
1	1	1	0	1	1	0	0

(b) PMASM结构

Fig. 1 Circuit structure diagram of the stateful logic gates and their truth tables

图 1 状态逻辑门电路结构图及其真值表

为方便描述, 本文使用简单门(simple gate)和复合门(composite gate)来指代状态逻辑门[8]. 其中, 简单门是指逻辑输入和输出映射到不同忆阻器件的逻辑门, 如图 1 中的“PMR-two-3NOR”和“PMASM-two-3NOR”. 在这类逻辑门中, 逻辑输出器件的初始状态一般设置为常数 0 或者 1, 状态逻辑门的执行过程就是对输出器件的“条件写”过程, 逻辑操作完成后, 将产生逻辑输出并保存在逻辑输出器件中. 复合门是指逻辑输出与其中一个逻辑输入共用同一个器件的状态逻辑门, 其可由简单门扩展而来. 若将简单门的输出器件的初始值不设为常数 0 或 1, 而是作为第 3 个逻辑输入, 那么简单门的逻辑功能将会扩展为由原始简单门功能与第 3 个逻辑输入的“或”(原始简单门基于输出器件的“条件置位”构建)或者“与”(原始简单门基于输出器件的“条件复位”构建). 例如, 对于图 1(a)中的“PMR-two-3NOR”门, 若将其输出器件的初始值作为第 3 个逻辑输入, 那么其实现的逻辑功能为 ONOR($\overline{P+Q}+Y$), 我们将此复合门命名为“PMR-three-3ONOR”; 类似地, 对于图 1(b)中的“PMASM-two-3NOR”门, 其实现的复合门为“PMASM-three-3ANOR”($\overline{P+Q\cdot Y}$). 由此, 我们看到, 每种简单状态逻辑门都对应一个由其扩展而来的复合门. 复合门的功能为对应的简单门的功能与“或逻辑”或者“与逻辑”的级联. 因此, 复合门的功能可以拆分成“简单门的功能 + 或逻辑(与逻辑)”; 反过来, “简单门的功能 + 或逻辑(与逻辑)”可以构成复合门的功能. 简单门和复合门的相互转化为本文后续逻辑网表后处理的理论基础.

1.2 复杂逻辑的级联实现

仅仅采用状态逻辑门完成单步逻辑无法满足实现复杂计算, 复杂计算过程的执行需要对状态逻辑门进行灵活级联, 即将前一级状态逻辑门的输出连接到下一级状态逻辑门的输入. 与 CMOS 逻辑门的级联不同, 状态逻辑门中的逻辑信息由电阻状态表示, 逻辑信息不能通过物理金属线进行传输. 然而, 归功于忆阻交叉阵列的灵活性, 忆阻状态逻辑门可以通过在忆阻交叉阵列中对器件的交叠使用来实现门的级联.

依托忆阻交叉阵列完成状态逻辑门级联的前提条件是状态逻辑门要能在忆阻交叉阵列中灵活配置. 通过前期调研发现, 可级联状态逻辑门主要为 3 种电路结构: PMR[9], PMASM[10], APMR[11]. 其中, 前 2 种电路结构适合在 2 维忆阻交叉阵列中配置, 如图 2 所示. 第 3 种电路结构适合在 3 维忆阻交叉阵列的层间器件之间配置.

对于可在忆阻阵列中灵活配置的状态逻辑门, 可以通过时空级联的方式实现复杂的逻辑计算[12]. 图 3 展示了在 2 维忆阻交叉阵列中级联 2 个“PMR-two-3NOR”状态逻辑门的步骤. 可以看到, 状态逻辑门的级联不仅需要协调忆阻阵列中的忆阻器单元(空间维度), 还需要按顺序触发这些门(时间维度). 因此, 依托忆阻交叉阵列完成复杂状态逻辑计算的过程就是通过施加操作电压序列将一个个状态逻辑门配置到忆阻交叉阵列中完成逻辑门功能的级联过程.

尽管存内状态逻辑计算系统消除了数据加载和存储的过程, 但其时空级联特性使得状态逻辑在计算过程本身上难以超越 CMOS 组合逻辑电路[5,11]. 然

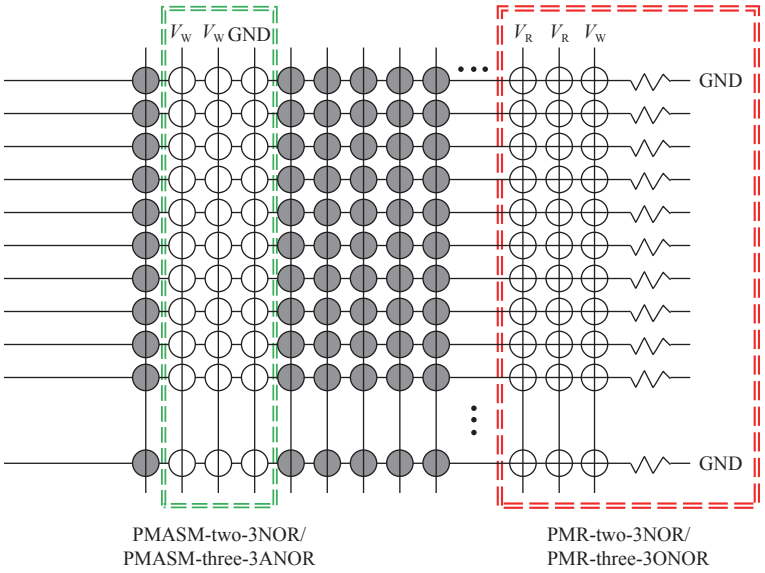


Fig. 2 Configuration of stateful logic gates in a 2D memristor-based crossbar array

图 2 状态逻辑门在 2 维忆阻交叉阵列中的配置

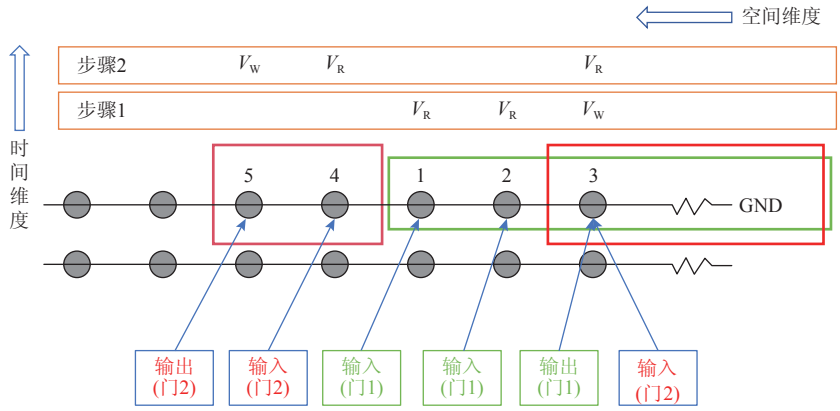


Fig. 3 Cascade of two “PMR-two-3NOR” stateful logic gates in a 2D memristor-based crossbar array

图 3 2 维忆阻交叉阵列中 2 个“PMR-two-3NOR”状态逻辑门的级联

而，根据研究表明，在忆阻交叉阵列中通过并行执行多个计算实例可以弥补这一弱点[13-15]. 但是，在较少步骤内实现复杂计算功能的状态逻辑操作仍然值得深入研究，因为它直接关系到存内状态逻辑计算系统的效率. 而针对复杂计算过程，如何准确找到最优状态逻辑门级联序列就是状态逻辑综合映射问题.

在状态逻辑研究的早期阶段，大多研究工作都是通过手动设计状态逻辑门的级联序列来实现复杂计算实例. 例如，Talati 等人[10]使用“PMASM-two-3NOR”逻辑门，通过 12 步级联实现了 1 位全加器操作；Adam 等人[16]使用“PMR-two-2IMP”“APMR-two-2IMP”逻辑门级联实现了 3 维忆阻阵列中的 1 位全加器，其中由于涉及许多读取和写入操作，级联需要 35 个步骤；Huang 等人[9]使用“PMR-two-3NAND”状态逻辑门，通过 10 步级联实现了 1 位全加法器；Xu 等人[11]级联 6 个 APMR 状态逻辑门，通过 14 步级联完成 1 位全加器的操作；Sun 等人[17]采用多输入复合状态逻辑门将 1 位全加器的实现步数减小为 2 步. 这种手动设计的策略能够应用于小规模电路功能的固定设计，但对于复杂的大规模电路功能，手动设计是耗时并且容易出错的. 因此，需要研发自动化的逻辑综合工具，以实现复杂计算.

目前，研究者们已经开发了多种状态逻辑综合工具，能够在忆阻交叉阵列中以较少的时间成本（或其他目标）找到实现复杂逻辑功能的状态逻辑门执行序列[8,15,18-26]. 对状态逻辑自动化综合工具的研究主要分为 2 个阶段：

第 1 个阶段考虑复杂计算功能到状态逻辑门功能的分解，并未过多地考虑阵列映射的约束[27-30]. 如

Chakraborti 等人[27]提出了一种采用忆阻器有效实现2-1多路复用器的方法,并提出了一种综合方法,该方法将给定的布尔逻辑表示为简化有序二进制决策图(ROBDD);Chattopadhyay 等人[28]对传统的综合算法进行了扩展,提出了新的启发式算法.此外,Bürger 等人[18]提出了一种使用面向 CMOS 的综合工具(如 ABC 工具)将复杂逻辑功能分解为状态逻辑门的基本功能的方法.

第 2 个阶段是利用 CMOS 综合工具自动化地完成逻辑综合,然后考虑阵列约束,完成状态逻辑门到阵列的位置映射.该阶段的研究大体又可以分为 2 类:一类是面向全阵列范围映射的状态逻辑综合映射工作,此类工作主要以状态逻辑门数最少和最大限度门级并行为综合映射的优化目标,少数工作探讨了面积(器件个数)的约束.例如,Hur 等人[19]提出了一种通用的综合映射流程(SIMPLE MAGIC),该流程使用了“PMASM-two-3NOR”“PMASM-one-2NOT”逻辑门,面向全阵列范围优化状态逻辑门的执行,并考虑所涉及的阵列约束.Bhattacharjee 等人[21]提出的综合映射流程(CONTRA)使用基于查找表(LUT)的输入函数映射到忆阻交叉阵列上,最大限度地实现并行操作,并使用一种新的搜索技术在忆阻交叉阵列内以最佳方式移动数据.然而,面向全阵列范围的综合映射方法多数需要依赖求解器遍历求解空间,是一个耗时的过程,从降低求解时间的角度考虑,出现了另一类面向单行/列映射的综合映射方法.这类方法的代表性工作为 Hur 等人[15]提出的改进的自动综合和映射方法,称为 SIMPLER MAGIC.该综合映射流程的优化目标是从以前的最小延迟(操作步骤数)转换为最小面积(使用的器件数量),在需要时重用单元以节省面积[5].

可以看到,当前的多数状态逻辑综合映射工作的研究皆以计算延时为优化目标,少数工作针对阵列面积(器件数目)的约束进行了讨论,鲜有针对其他目标,特别是器件磨损的研究和探讨.因此,本文以降低状态逻辑计算过程的器件磨损为目标,探索新的状态逻辑综合映射方法,提高边缘计算设备的使用寿命,从而降低维护和更换成本.

2 状态逻辑门的兼容性验证

如 1.2 节所述,复杂状态逻辑计算过程的自动化设计实现需要 2 个方面的内容:一是有功能完备且可在同一忆阻交叉阵列中配置的状态逻辑门;二是有自动化综合映射方法的支持.因此,在进行状态逻辑综合映射研究之前,首先需要对综合映射中所要使用的多种状态逻辑门进行功能和兼容性验证,保证逻辑功能的正确性和阵列可配置性.本文使用 SPICE 电路仿真工具对 6 种 PMR 结构的状态逻辑门进行功能验证,包括 COPY, NOT, NOR, OR 共 4 种简单门和 IMP, ONOR 共 2 种复合门.验证过程中使用 Stanford 大学的开源 ReRAM 器件模型(metal oxide resistive random access memory Verilog-A models, Version 1.0.0)[31],仿真使用的器件参数如表 1 所示.

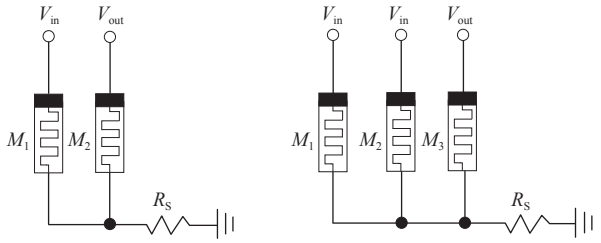
Table 1 Device Parameters

表 1 器件参数

参数	解释	默认值
T_ini /K	温度	298
F_min /(V/m)	促进隧穿间隙形成的最小场强	1.4E9
Tox /nm	氧化层厚度	12
gap_ini /nm	初始隧穿间隙	1.8
gap_min /nm	最小隧穿间隙	0.2
gap_max /nm	最大隧穿间隙	1.8

在所采用的忆阻器模型中,离子和空穴迁移的复杂过程被简化为 1 维导电细丝的生长/溶解,并保留了基本的转变物理特性.隧穿间隙(gap distance)的大小,即导电细丝尖端与顶部电极之间的距离,是决定器件电阻的主要变量[31].因此,在实际功能验证过程中,通过设置间隙距离将器件的初始状态设置为高阻态(HRS)或低阻态(LRS),选择 1.7 nm 的间隙距离所对应的电阻状态作为 HRS, 0.3 nm 的间隙距离所对应的电阻状态作为 LRS.通过尝试,我们取置位电压为 1.4 V, 复位电压为 −1.0 V, 作为状态逻辑门操作条件的求解参数.

根据 1.1 节所使用的状态逻辑门命名规则,本文所使用的 6 个状态逻辑门可分为 4 类:第 1 类是“PMR-one-2*x*”,包括“PMR-one-2NOT”“PMR-one-2COPY”;第 2 类是“PMR-two-2*x*”,包括“PMR-two-2IMP”;第 3 类是“PMR-two-3*x*”,包括“PMR-two-3NOR”“PMR-two-3OR”;第 4 类是“PMR-three-3*x*”,包括“PMR-three-3ONOR”.在上述分类中,第 1, 2 类状态逻辑门的电路结构相同,如图 4(a)所示.其中,“PMR-two-2IMP”是由“PMR-one-2NOT”扩展而来的复合门;第 3, 4 类状态逻辑门的电路结构相同,如图 4(b)所示,且“PMR-three-3ONOR”是由“PMR-two-3NOR”扩展而来的复合门.以下分 2 个小节对上述 4 类状态逻辑门的仿真验证进行阐述.



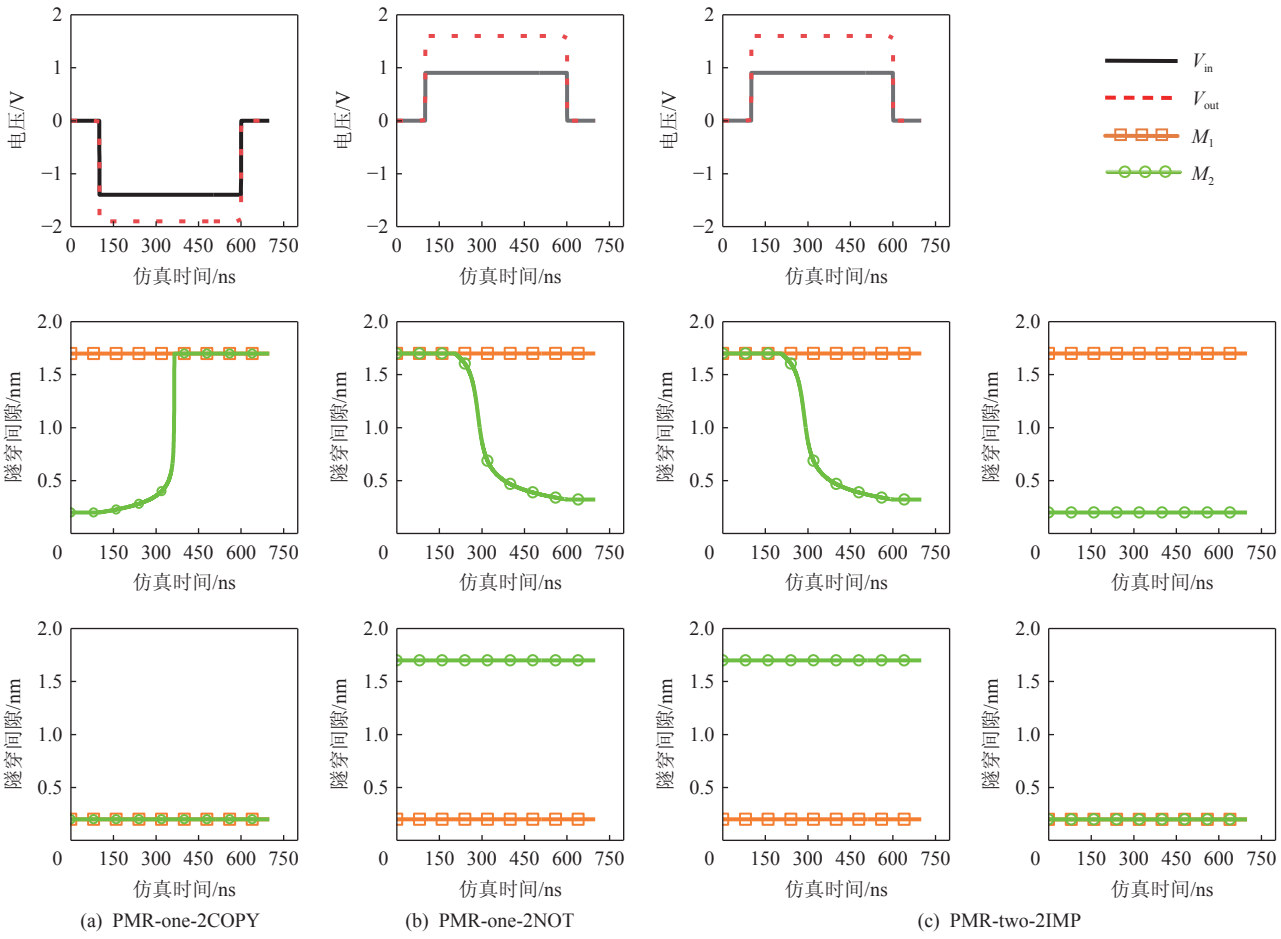
(a) PMR-one/two-2x (b) PMR-two/three-3x

Fig. 4 The circuit structure diagram of the stateful logic gates

图 4 状态逻辑门电路结构图

2.1 “PMR-one/two-2x”状态逻辑门验证

“PMR-one-2NOT”“PMR-one-2COPY”“PMR-two-2IMP”的电路结构由 2 个并联的忆阻器 M_1、M_2 和 1 个串联的电阻 R_S（50 Ω）构成. 在仿真验证时，根据逻辑门的状态转换，通过在 V_{in}，V_{out} 端口施加特定的操作电压，使忆阻器获得相应分压，由此实现不同的逻辑功能. 仿真结果如图 5 所示，对于每一个分图，最上方第 1 幅图展示了施加的电压激励，其他的图展示了各种逻辑状态变化情况下间隙距离的变化曲线.



(a) PMR-one-2COPY (b) PMR-one-2NOT (c) PMR-two-2IMP

Fig. 5 Simulation verification results of COPY, NOT, and IMP gates

图 5 COPY，NOT，IMP 门的仿真验证结果

2.2 “PMR-two/three-3x”状态逻辑门验证

“PMR-two-3NOR”“PMR-two-3OR”“PMR-three-3ONOR”的电路结构由 3 个并联的忆阻器 M_1、M_2、M_3 和 1 个串联的电阻 R_S（50 Ω）构成，仿真结果如图 6 所示.

由于所有状态逻辑门的仿真皆基于相同参数的忆阻器模型，且它们的结构皆为兼容于忆阻交叉阵列的电路结构. 因此，可以认为这 6 种状态逻辑门可在由该参数忆阻器构成的交叉阵列中成功执行. 接下来，将介绍采用这 6 种状态逻辑门，依托忆阻交叉阵列完成复杂状态逻辑计算的低磨损综合映射方法.

3 低磨损综合映射

本节介绍面向低磨损存内计算的多状态逻辑门综合映射方法. 该方法采用包含多种状态逻辑门的综合映射过程来降低复杂存内状态逻辑计算过程的翻转率（toggle rate），综合映射流程如图 7 所示.

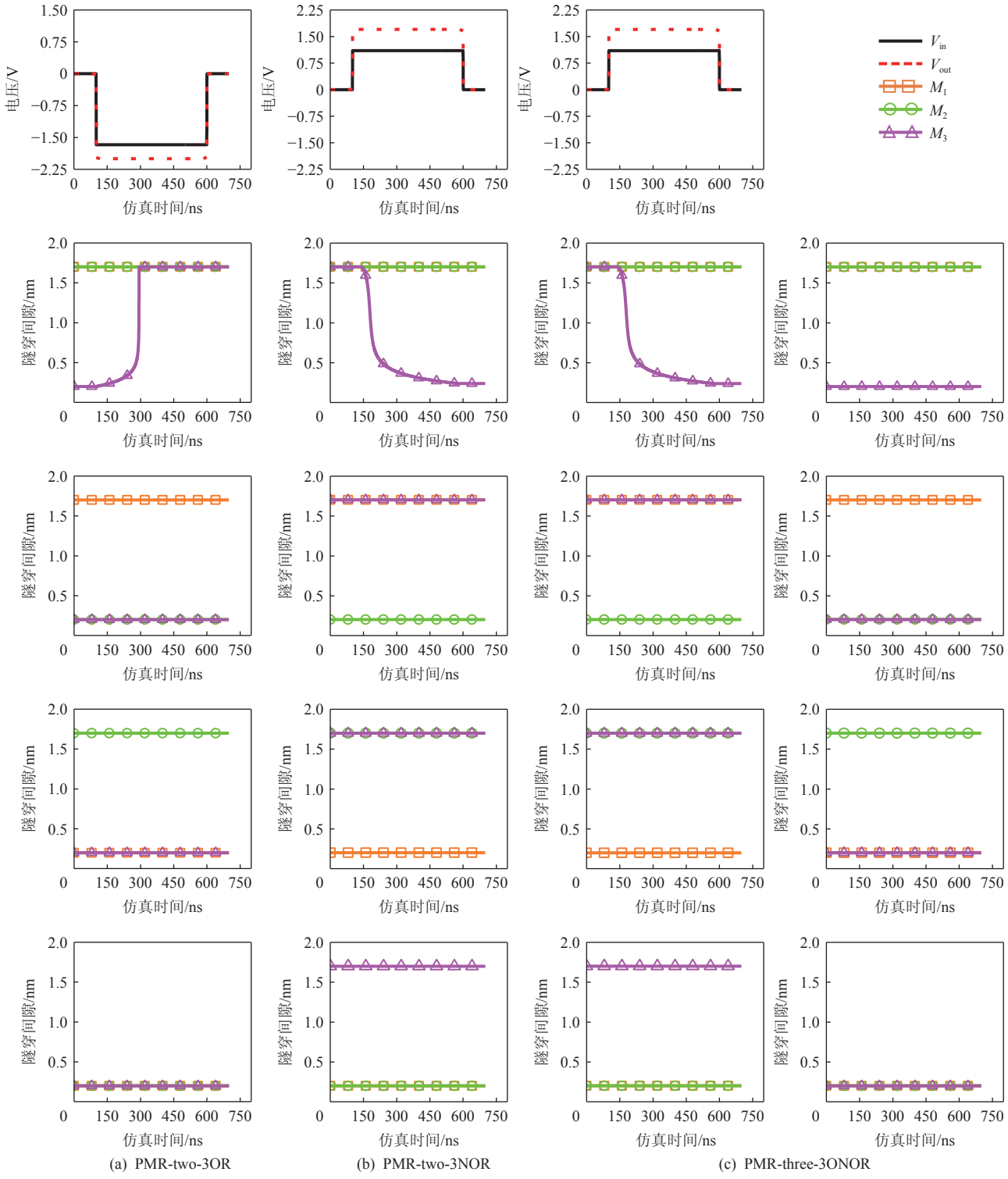


Fig. 6 Simulation verification results of OR, NOR, and ONOR gates

图 6 OR，NOR，ONOR 门的仿真验证结果

首先，我们使用商用逻辑综合工具将复杂逻辑功能综合为由“PMR-one-2NOT”“PMR-two-3NOR”“PMR-two-3OR”逻辑功能构成的网表，在此过程中以门的总翻转率最小为优化目标. 然后，对该网表进行后处理，按照合并规则将可合并的简单门功能合并为复合门功能，从而进一步引入“PMR-two-2IMP”“PMR-three-3ONOR”“PMR-one-2COPY”(解决循环依赖[8])功能，合并过程同样以降低翻转率为判断条件. 最后，将经过后处理的网表功能与对应的状态逻辑门一一映射并将状态逻辑门按执行顺序配置到单

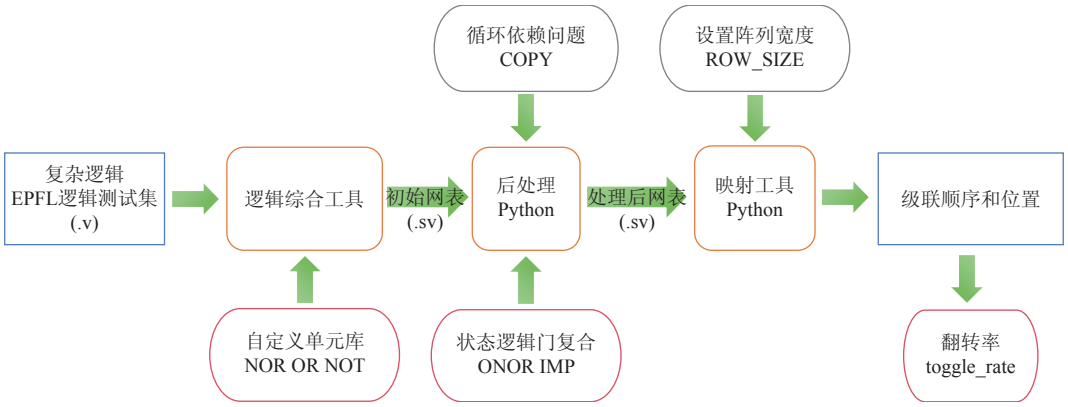


Fig. 7 Stateful logic synthesis and mapping flow for low wear

图 7 低磨损状态逻辑综合与映射流程

行忆阻交叉阵列上，得到相应的状态逻辑门级联顺序和位置，并计算得到其翻转率.

3.1 状态逻辑门翻转率的计算

状态逻辑门的翻转率是其逻辑状态转变的平均概率. 以“PMR-two-3NOR”为例，输出忆阻器 M_3 的初始状态为逻辑 0(HRS)，在经过逻辑操作后，4 种情况中仅有 1 种情况的状态会发生改变，如图 1 中真值表所示. 因此，“PMR-two-3NOR”在进行逻辑操作时状态发生转变的平均概率为 0.25.同理，“PMR-three-3ONOR”的翻转率为 0.125. 表 2 列出了各个状态逻辑门的翻转率，该翻转率可以衡量门的磨损程度.

Table 2 Toggle Rates of the Six Stateful Logic Gates Used in This Paper

表 2 本文使用到的 6 种状态逻辑门的翻转率

状态逻辑门	翻转率
COPY	0.5
NOT	0.5
NOR	0.25
OR	0.25
IMP	0.25
ONOR	0.125

3.2 逻辑综合

复杂逻辑综合过程使用商用 CMOS 逻辑电路的综合工具完成复杂逻辑功能到状态逻辑门功能的分解. 具体综合过程如下：

首先，根据所使用的状态逻辑门功能定义单元库，即.lib 文件. 从标准单元库中定义 NOT, NOR, OR 门作为一个新的自定义单元库. 然后，修改所定义门的面积(area)参数为对应状态逻辑门的翻转率. 最后，设置面积最小为综合目标，完成综合过程，得到由 3 种简单门功能构成的低翻转率的网表.

3.3 后处理

上一步得到的简单门功能网表中，可能存在{NOT, OR}, {NOR, OR}, {NOT, NOR}, {NOR, NOR}, {NOT, NOT}这些功能团组. 根据 1.1 节中描述的简单门和复合门对应关系以及逻辑等价性变换关系，可对网表进行后处理变换.

值得注意的是，为避免输入覆盖造成错误，在对简单门进行合并时，要遵循 2 个规则[8]：

1) 若合并后的复合门覆盖的输入同时也是其他状态逻辑门的输入时，则 2 个简单门不能合并；

2) 若第 2 个简单门的输入是其他复合门的被覆盖输入，则 2 个简单门不能合并.

在满足上述规则的情况下，可以进行的合并如表 3 所示.

Table 3 Merges of Stateful Logic Gates

表 3 状态逻辑门的合并

情况	功能团组	合并后
1	{NOR，OR}	ONOR
2	{NOT，OR}	IMP
3	{NOT，NOR}	IMP，NOT
4	{NOR，NOR}	ONOR，NOT
5	{NOT，NOT}	NOT

由合并前后的翻转率计算可知，进行情况 3 合并后翻转率保持不变(0.5+0.25 = 0.25 + 0.5)，进行情况 4 合并后翻转率会上升(0.25+0.25 < 0.125 + 0.5). 在以低磨损为目标的综合映射方法中，还需分别对情况 3 和情况 4 进行处理.

为使得翻转率进一步降低，应取消情况 3 的合并，保留下来的 NOR 门和 NOT 门可以进行其他使得翻转率降低的合并.

对于情况 4 的处理，单纯地像情况 3 那样取消合

并, 并不能得到预期的优化效果. 这是由于情况 4 中第 2 个 NOR 分解为 OR 和 NOT 门后, 会出现 2 个 NOT 门相连的情况, 满足情况 5. 可以同时考虑情况 4 和情况 5 合并使得翻转率进一步降低, 新的合并过程为:

NOR+ NOR+ NOT=>

NOR+ OR+ NOT+ NOT=>

ONOR+ NOT.

综上所述, 后处理阶段状态逻辑门的合并规则为:

1) NOR(0.25)+ OR(0.25)=> ONOR(0.125);

2) NOT(0.5)+ OR(0.25)=> IMP(0.25);

3) NOR(0.25)+ NOR(0.25)+ NOT(0.5)=>

NOR(0.25)+ OR(0.25)+ NOT(0.5)+ NOT(0.5)=>

ONOR(0.125)+ NOT(0.5);

4) NOT(0.5)+ NOT(0.5)=> NOT(0.5).

其中, 前 3 种变换, 式子左右逻辑功能等价, 前一个简单门的输入可直接指向复合门. 而第 4 种变换, 前一个 NOT 门的输入即为正确的输出, 直接连向其他状态逻辑门即可.

3.4 映 射

完成后处理过程后, 得到新的包含复合门功能的网表, 进一步需要基于该网表的级联关系, 完成状态逻辑门到忆阻交叉阵列的映射. 本文遵循 LOSSS 中的映射方法[8], 以单指令多数据(single instruction multiple data, SIMD)计算场景为背景, 采用了面向行/列的映射模式, 允许同时执行复杂逻辑的多个实例, 每个实例都压缩到交叉阵列的一行中. 通过修改现有的 SIMPLER MAGIC 映射工具[15], 以满足对多状态逻辑门映射的需求.

首先, 读入经过后处理的网表文件, 识别所使用的逻辑功能并根据逻辑功能匹配到相应的状态逻辑门, 提取相应逻辑结构以及节点信息.

其次, 确定状态逻辑门的执行顺序. 状态逻辑门的执行顺序与该门所代表的节点的单元使用值(cell usage, CU)有关. 在 SIMPLER MAIGC 的算法中, 该值为执行一个门所需要的内存单元(作为输入的节点)的估计值[15]. 单元使用值较大的门应该先执行, 由此作为该门的输入节点所占用的忆阻器单元可以尽早地被释放, 重新分配新的节点. 此外, 为保证逻辑功能的正确性, 复合门所代表的节点应在其所有兄弟节点中最后被映射和执行.

最后, 根据设定的阵列宽度(row size)为每个节点分配内存单元, 得到整个逻辑执行的延迟和重用单元数. 其中, 每一个节点包含 3 个状态: 1)可使用(available)状态; 2)已使用(used)状态; 3)尚未初始化(uninitialized)状态. 处于状态 3)的节点经过初始化转变为状态 1); 处于状态 1)的节点可以被分配使用, 并转为状态 2); 当处于状态 2)的节点不再参与后续执行时, 可以释放并重用该节点. 此外, 在对映射后结果统计时, 重用单元的平均翻转率记为 0.5, 需要计入总翻转率.

4 结果与讨论

为评价优化的效果, 我们采用提出的低磨损综合映射方法对 EPFL[32], LGSynth91[33] 基准电路测试集进行实验测试.LGSynth91 是一个在集成电路(IC)设计和测试领域广泛使用的基准电路集合, 包含了多种用于评估和设计优化算法的标准电路. 相较于 LGSynth91, EPFL 测试集的电路规模更大, 对逻辑优化工具提出了更高的要求. 本文分别选取 EPFL, LGSynth91 中的 10 个测试电路, 经过综合映射后, 统计最终状态逻辑门映射序列的延迟和翻转率, 并与采用当前 2 种典型的状态逻辑综合映射工具 SIMPLER MAGIC[15], LOSSS[8] 得到的结果进行对比.

4.1 实验设置

为了公平比较, 3 个综合映射流程的 CMOS 逻辑综合阶段均使用相同的商用 CMOS 逻辑电路综合工具. 本文的方法和 LOSSS 的自定义单元库中包含 OR, NOR, NOT 门, 而 SIMPLER MAGIC 中仅包含 NOR, NOT 门. 本文所提出的低磨损综合映射方法的自定义单元库中各个门的面积参数设置为其对应的翻转率, 而 LOSSS, SIMPLER MAGIC 的自定义单元库中各个门的面积参数设置为相同的值. 除自定义单元库不同之外, 综合环境、综合约束等均与原流程保持一致.

在进行 3 种综合映射方法的比较时, 每个测试样例映射的阵列宽度设置为 3 个综合映射流程能够进行综合映射的最小宽度的最大值. 表 4 中罗列了 3 个综合映射流程下各个测试集的最小阵列宽度, 再对每个测试集取阵列宽度的最大值, 即为最终的阵列宽度.

4.2 实验结果

针对 3 种综合映射流程, 分别得到各个测试样例的执行延迟和翻转率. 为进行更为直观的比较, 选取 SIMPLER MAGIC 综合映射流程所得的结果为基准值, 分别计算本文和 LOSSS 相较于 SIMPLER MAGIC

Table 4 Selection of Row Size

表 4 阵列宽度选取

EPFL				
测试电路	本文	LOSSS	SIMPLER MAGIC	阵列宽度选取
adder	510	463	390	510
arbiter	2 189	2 147	1 719	2 189
bar	636	636	399	636
cavlc	168	169	124	169
ctrl	54	56	45	56
dec	371	371	267	371
int2float	50	59	41	59
max	870	854	783	870
priority	250	191	194	250
voter	1 235	1 110	1 354	1 354
LGSynth91				
测试电路	本文	LOSSS	SIMPLER MAGIC	阵列宽度选取
alu2	74	80	78	80
cm138a	30	30	17	30
cm42a	22	25	16	25
cmb	38	36	27	38
cht	92	94	88	94
term1	88	70	70	88
f51m	37	42	32	42
mux	31	31	31	31
ttt2	64	67	57	67
z4ml	18	16	20	20

在执行延迟和翻转率 2 个指标上优化的比例(即数值下降的百分比)，如图 8~11 所示.

由图 8~11 中数据可知：在执行延迟指标上，与 SIMPLER MAGIC 相比，本文所述综合映射流程在 EPFL 测试集上有最高 45.35% 和最低 15.94% 的降低，整体上平均有 24.18% 的降低；在 LGSynth91 测试集上有最高 51.35% 和最低 21.74% 的降低，整体上平均有 34.67% 的降低. 本文与 LOSSS 所得结果相差不大，在 2 个测试集下平均仅有不到 1.20% 的差距. 可以看到，虽然本文报道的综合映射方法是基于降低整个计算过程器件的总翻转率进行优化的，但是在计算延迟上相较于先前报道的 LOSSS 工具亦有改善. 这可能是由于目前所采用的商用 CMOS 综合映射工具基于启发式算法来进行优化，基于 CMOS 关键路径的延时优化是工具默认的优先级最高的综合优化属性. 而本文在基于 CMOS 的综合流程中，将门的面积设置为状态逻辑门翻转率的方法，或许产生了更好的起始网表，故而使得后处理和映射之后的状态逻

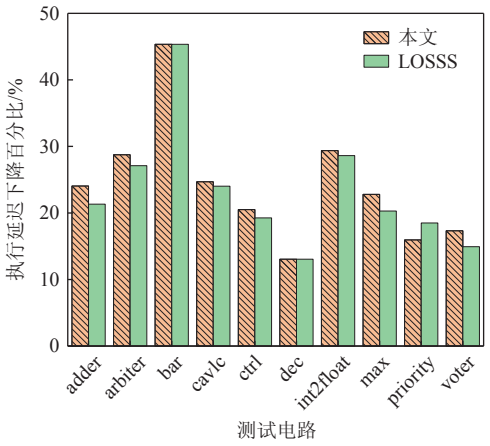


Fig. 8 Percentage of reduced execution latency compared with SIMPLER MAGIC (EPFL)

图 8 在执行延迟上相对于 SIMPLER MAGIC 减少的百分比（EPFL）

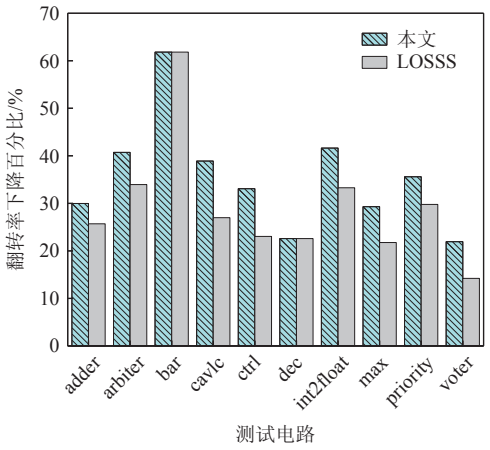


Fig. 9 Percentage of reduced toggle rate compared with SIMPLER MAGIC (EPFL)

图 9 在翻转率上相对于 SIMPLER MAGIC 减少的百分比（EPFL）

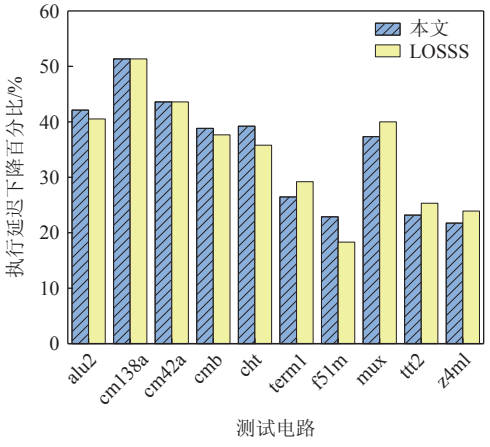


Fig. 10 Percentage of reduced execution latency compared with SIMPLER MAGIC (LGSynth91)

图 10 在执行延迟上相对于 SIMPLER MAGIC 减少的百分比（LGSynth91）

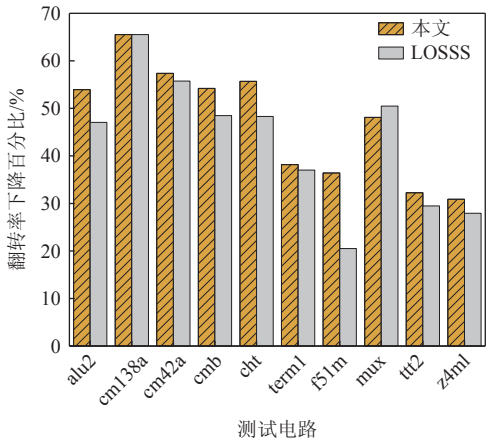


Fig. 11 Percentage of reduced toggle rate compared with SIMPLER MAGIC (LGSynth91)

图 11 在翻转率上相对于 SIMPLER MAGIC 减少的百分比(LGSynth91)

辑门序列的总延时也有所降低.

在翻转率指标上, 与 SIMPLER MAGIC 相比, 本文所述综合映射流程在 EPFL 测试集上有最高 61.82% 和最低 21.94% 的降低, 整体上平均有 35.55% 的降低; 在 LGSynth91 测试集上有最高 65.52% 和最低 30.88% 的降低, 整体上平均有 47.26% 的降低. 同时, 本文在 2 个测试集上相较于 LOSSS 综合映射流程平均分别有 8.48% 和 6.72% 的降低, 与本文低磨损工具研发的初衷一致. 特别地, 本文所述综合映射流程在 mux 测试电路上的总翻转率高于 LOSSS.这是由于总翻转率的计算包含了单元重用部分, 若仅考虑后处理结束后的结果, 本文所述综合映射流程下的翻转率仍然低于 LOSSS.

综上所述, 本文提出的综合流程与 SIMPLER MAGIC 和 LOSSS 相比, 在翻转率和执行延迟上均得到了一定优化.

5 总 结

在本文工作中, 首先验证了多种状态逻辑门对同一忆阻存储阵列的兼容性. 然后以翻转率最优为约束, 研究面向忆阻存储阵列内低磨损计算的状态逻辑综合映射方法, 建立了包含多种状态逻辑门的复杂逻辑计算综合映射流程, 可以针对任意给定计算功能, 给出低磨损的状态逻辑门级联序列和位置, 具有重要的理论意义. 后续的工作中可以考虑加入更多的状态逻辑门或是选择更优的状态逻辑门组合. 同时, 可以综合考虑多个优化目标, 在阵列规模、处理时效以及器件寿命上, 取得更优的折中.

作者贡献声明: 赵安宁与许诺为共同第一作者, 许诺提出了论文的总体框架和算法思路, 赵安宁完善了想法和算法细节并完成了实验和结果分析; 许诺和赵安宁撰写论文的主体部分; 刘康和罗莉参与了想法和方案讨论; 所有作者都参与了论文讨论和修改.

参 考 文 献

[1] Wulf W A, McKee S A. Hitting the memory wall: Implications of the obvious[J]. ACM SIGARCH Computer Architecture News, 1995, 23(1): 20−24

[2] Ding Wenlong, Wang Chengning, Tong Wei. Energy-efficient floating-point memristive in-memory processing system based on self-selective mantissa compaction[J]. Journal of Computer Research and Development, 2022, 59(3): 533−552 (in Chinese)
(丁文隆, 汪承宁, 童薇. 基于自选尾数压缩的高能效浮点忆阻存内处理系统[J]. 计算机研究与发展, 2022, 59(3): 533−552)

[3] Wei Zheng, Zhang Xingjun, Zhuo Zhimin, et al. PPO-based automated quantization for ReRAM-based hardware accelerator[J]. Journal of Computer Research and Development, 2022, 59(3): 518−532 (in Chinese)
(魏正, 张兴军, 卓志敏, 等. 基于近端策略优化的阻变存储硬件加速器自动量化[J]. 计算机研究与发展, 2022, 59(3): 518−532)

[4] Zhang Zhang, Shi Gang, Wang Qifan, et al. Survey of in-memory computing technology based on SRAM and non-volatile memory[J]. Journal of Computer Research and Development, 2024, 61(12): 2937−2951(in Chinese)
(张章, 施刚, 王启帆, 等. 基于 SRAM 和 NVM 的存内计算技术综述[J]. 计算机研究与发展, 2024, 61(12): 2937−2951)

[5] Xu Nuo, Park T, Yoon K J, et al. In-memory stateful logic computing using memristors: Gate, calculation, and application[J]. Physica Status Solidi (RRL)–Rapid Research Letters, 2021, 15(9): 2100208

[6] Borghetti J, Snider G S, Kuekes P J, et al. 'Memristive'switches enable 'stateful'logic operations via material implication[J]. Nature, 2010, 464(7290): 873−876

[7] Gaillardon P E, Amarú L, Siemon A, et al. The programmable logic-in-memory (PLiM) computer[C]//Proc of the 19th Design, Automation & Test in Europe Conf & Exhibition (DATE). Piscataway, NJ: IEEE, 2016: 427−432

[8] Hu Yihong, Xu Nuo, Feng Chaochao, et al. LOSSS-logic synthesis based on several stateful logic gates for high time-efficient computing[C]//Proc of the 29th Asia and South Pacific Design Automation Conf (ASP-DAC). Piscataway, NJ: IEEE, 2024: 805−811

[9] Huang Peng, Kang Jinfeng, Zhao Yudi, et al. Reconfigurable nonvolatile logic operations in resistance switching crossbar array for large-scale circuits[J]. Advanced Materials, 2016, 28(44): 9758−9764

[10] Talati N, Gupta S, Mane P, et al. Logic design within memristive memories using memristor-aided logic (MAGIC)[J]. IEEE Transactions on Nanotechnology, 2016, 15(4): 635−650

[11] Xu Nuo, Park T G, Kim H J, et al. A stateful logic family based on a new logic primitive circuit composed of two antiparallel bipolar memristors[J]. Advanced Intelligent Systems, 2020, 2(1): 1900082

[12] Seok J Y, Song S J, Yoon J H, et al. A review of three - dimensional resistive switching cross - bar array memories from the integration and materials property points of view[J]. Advanced Functional Materials, 2014, 24(34): 5316−5339

[13] Ronen R, Eliahu A, Leitersdorf O, et al. The bitlet model: A parameterized analytical model to compare PIM and CPU systems[J]. ACM Journal on Emerging Technologies in Computing Systems, 2022, 18(2): 1−29

[14] Leitersdorf O, Leitersdorf D, Gal J, et al. AritPIM: High-throughput in-memory arithmetic[J]. IEEE Transactions on Emerging Topics in Computing, 2023, 11(3): 720−735

[15] Hur R B, Ronen R, Haj-Ali A, et al. SIMPLER MAGIC: Synthesis and mapping of in-memory logic executed in a single row to improve throughput[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019, 39(10): 2434−2447

[16] Adam G C, Hoskins B D, Prezioso M, et al. Optimized stateful material implication logic for three-dimensional data manipulation[J]. Nano Research, 2016, 9(12): 3914−3923

[17] Sun Zhong, Ambrosi E, Bricalli A, et al. Logic computing with stateful neural networks of resistive switches[J]. Advanced Materials, 2018, 30(38): 1802554

[18] Bürger J, Teuscher C, Perkowski M. Digital logic synthesis for memristors[C]// Proc of the 11th Reed-Muller Workshop. Los Alamitos, CA: IEEE Computer Society, 2013: 31−40

[19] Hur R B, Wald N, Talati N, et al. SIMPLE MAGIC: Synthesis and in-memory mapping of logic execution for memristor-aided logic[C]// Proc of the 36th IEEE/ACM Int Conf on Computer-Aided Design (ICCAD). Piscataway, NJ: IEEE, 2017: 225−232

[20] Tenace V, Rizzo R G, Bhattacharjee D, et al. SAID: A supergate-aided logic synthesis flow for memristive crossbars[C]//Proc of the 22nd Design, Automation & Test in Europe Conf & Exhibition (DATE). Piscataway, NJ: IEEE, 2019: 372−377

[21] Bhattacharjee D, Chattopadhyay A, Dutta S, et al. CONTRA: Area-constrained technology mapping framework for memristive memory processing unit[C/OL]//Proc of the 39th Int Conf on Computer-Aided Design. Los Alamitos, CA: IEEE Computer Society, 2020[2024-01-22]. https://dl.acm.org/doi/10.1145/3400302.3415681

[22] Peled N, Hur R B, Ronen R, et al. X-MAGIC: Enhancing PIM using input overwriting capabilities[C]//Proc of the 28th IFIP/IEEE Int Conf on Very Large Scale Integration (VLSI-SOC). Piscataway, NJ: IEEE, 2020: 64−69

[23] Zou Minhui, Zhou Junlong, Sun Jin, et al. Improving efficiency and lifetime of logic-in-memory by combining IMPLY and MAGIC families[J]. Journal of Systems Architecture, 2021, 119(C): 102232

[24] Gupta S, Imani M, Rosing T. Felix: Fast and energy-efficient logic in memory[C/OL]//Proc of the 37th IEEE/ACM Int Conf on Computer-Aided Design (ICCAD). Piscataway, NJ: IEEE, 2018[2024-01-28]. https://doi.org/10.1145/3240765.3240811

[25] Gharpinde R, Thangkhiew P L, Datta K, et al. A scalable in-memory logic synthesis approach using memristor crossbar[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2017, 26(2): 355−366

[26] Zhu Zhenhua, Ma Mingyu, Liu Jialong, et al. A general logic synthesis framework for memristor-based logic design[C/OL]//Proc of the 38th IEEE/ACM Int Conf on Computer-Aided Design (ICCAD). Piscataway, NJ: IEEE, 2019[2024-02-15]. https://ieeexplore.ieee.org/document/8942111

[27] Chakraborti S, Chowdhary P V, Datta K, et al. BDD based synthesis of Boolean functions using memristors[C]//Proc of the 9th Int Design and Test Symp (IDT). Piscataway, NJ: IEEE, 2014: 136−141

[28] Chattopadhyay A, Rakosi Z. Combinational logic synthesis for material implication[C]//Proc of the 19th IEEE/IFIP Int Conf on VLSI and System-on-Chip. Piscataway, NJ: IEEE, 2011: 200−203

[29] Raghuvanshi A, Perkowski M. Logic synthesis and a generalized notation for memristor-realized material implication gates[C]//Proc of the 33rd IEEE/ACM Int Conf on Computer-Aided Design (ICCAD). Piscataway, NJ: IEEE, 2014: 470−477

[30] Marranghello F S, Callegaro V, Reis A I, et al. SOP based logic synthesis for memristive IMPLY stateful logic[C]//Proc of the 33rd IEEE Int Conf on Computer Design (ICCD). Piscataway, NJ: IEEE, 2015: 228−235

[31] Jiang Zizhen, Yu Shimeng, Wu Yi, et al. Verilog-A compact model for oxide-based resistive random access memory (RRAM)[C]//Proc of the 23rd Int Conf on Simulation of Semiconductor Processes and Devices (SISPAD). Piscataway, NJ: IEEE, 2014: 41−44

[32] Amarú L, Gaillardon P E, De Micheli G. The EPFL combinational benchmark suite[C/OL]//Proc of the 24th Int Workshop on Logic & Synthesis (IWLS). Los Alamitos, CA: IEEE Computer Society, 2015[2024-02-18]. https://infoscience.epfl.ch/entities/publication/309aea67-b5a1-4532-8a6f-0a141d8f1ab3

[33] Yang S. Logic synthesis and optimization benchmarks user guide version 3.0[R]. Durham, NC: Microelectronics Center of North Carolina (MCNC), 1991



Zhao Anning, born in 2001. Master candidate. Student member of CCF. His main research interest includes automation of in-memory stateful logic.

赵安宁, 2001 年生. 硕士研究生. CCF 学生会员. 主要研究方向为存内状态逻辑自动化.



Xu Nuo, born in 1989. PhD, assistant researcher. Senior member of CCF. His main research interests include emerging non-volatile memory technologies (e.g. ReRAM /memristor) and their employment in in-memory computation.

许　诺, 1989 年生. 博士，助理研究员. CCF 高级会员. 主要研究方向为新兴的非易失性存储技术(如 ReRAM/忆阻器)及其在存内计算中的应用.



Liu Kang, born in 2001. Master candidate. Student member of CCF. His main research interests include memristor stateful logic, in-memory computing.

刘　康, 2001 年生. 硕士研究生. CCF 学生会员. 主要研究方向为忆阻状态逻辑和存内计算.



Luo Li, born in 1971. PhD, research fellow. Member of CCF. Her main research interest includes high-performance general-purpose micro-processor design and verification technology.

罗　莉, 1971 年生. 博士，研究员. CCF 会员. 主要研究方向为高性能通用微处理器设计与验证技术.



Pan Bingzheng, born in 1999. PhD candidate. Student member of CCF. His main research interests include the applications of memristor technology and memristor-based convolutional neural network computing.

潘炳征, 1999 年生. 博士研究生. CCF 学生会员. 主要研究方向为忆阻器技术应用和忆阻卷积神经网络计算.



Bo Ziyi, born in 2001. Master candidate. Student member of CCF. Her main research interest includes in-memory computing.

薄子怡, 2001 年生. 硕士研究生. CCF 学生会员. 主要研究方向为存内计算.



Tan Chenghao, born in 2002. Master candidate. Student member of CCF. His main research interests include the applications of memristor technology and memristor-based convolutional neural network computing.

谭承浩, 2002 年生. 硕士研究生. CCF 学生会员. 主要研究方向为忆阻器技术应用和忆阻卷积神经网络计算.