doi:10.13682/j.issn.2095-6533.2025.06.008

基于二元决策图重排序优化的忆阻逻辑综合框架

刘 鹏,朱亚军,姚 廉,武继刚

(广东工业大学 计算机学院,广东 广州 510006)

摘 要:为了改善在基于路径的忆阻逻辑计算框架中,将二元决策图(Binary Decision Diagrams, BDD)映射至忆阻交叉阵列时硬件开销过大的问题,提出一种基于 BDD 重排序优化的忆阻逻辑综合框架。该框架首创性地将自适应重启遗传算法(Adaptive Restart Genetic Algorithm, ARGA)用于 BDD 变量顺序优化,通过 ARGA 生成更适配忆阻交叉阵列的 BDD 结构,而 ARGA 中内置的自适应重启机制可保障 BDD 变量顺序优化的高效性,进而优化映射后阵列的行列数,有效减少硬件面积。对 17 个基准电路进行评估,实验结果表明,与改进前的忆阻逻辑框架相比,所提方法实现 15% 的阵列面积减少,并降低 26% 的运行能耗和 12% 的时延。且与 COMPACT、CONTRA 类型忆阻逻辑框架相比,运行能耗降低 3~4 个数量级,时延分别降低 80%和 97%。通过 BDD 结构与忆阻阵列映射约束的协同优化,为提升忆阻逻辑电路的综合效率提供了有效途径。

关键词:内存计算;忆阻器;二元决策图;遗传算法;逻辑综合

中图分类号:TN402 **文献标志码**:A **文章编号**:2095-6533(2025)06-0068-09

A memristor synthesis framework optimized by BDD reordering

LIU Peng, ZHU Yajun, YAO Lian, WU Jigang

(School of Computer Science and Technology, Guangdong University of Technology, Guangzhou 510006, China)

Abstract: To address the issue of excessive hardware overhead that arises when mapping binary decision diagrams (BDD) to memristor crossbar arrays within path-based memristive logic computing frameworks, a memristive logic synthesis framework based on BDD reordering optimization is proposed. The framework pioneers the application of the adaptive restart genetic algorithm (ARGA) to BDD variable order optimization, which generates BDD structures more suitable for mapping to memristor crossbar arrays, while its built-in adaptive restart mechanism ensures the efficiency of this optimization process which further optimizes the number of rows and columns in the mapped crossbar array, thereby effectively reducing hardware area. Evaluations were conducted on 17 benchmark circuits, and the experimental results show that compared with the original memristive logic framework, the proposed method reduces crossbar area by 15%, operation energy consumption by 26%, and latency by 12%. Moreover, in comparison with other memristive logic frameworks such as COMPACT and CONTRA, the proposed method reduces operation energy consumption by 3 and 4 orders of magnitude respectively, and decreases latency by 80% and 97%, respectively. Through the collaborative optimization of BDD structures and memristor array mapping constraints, this research provides an effective approach to enhance the synthesis ef-

收稿日期:2025-08-17

基金项目:国家自然科学基金项目(62374047, 62174038);计算机体系结构国家重点实验室开放课题(CLQ 202407)

引文格式:刘鹏,朱亚军,姚廉,等.基于二元决策图重排序优化的忆阻逻辑综合框架[J].西安邮电大学学报,2025,30(6):68-76.

LIU P, ZHU Y J, YAO L, et al. A memristor synthesis framework optimized by BDD reordering[J]. Journal of Xi'an University of Posts and Telecommunications, 2025, 30(6): 68-76.

ficiency of memristive logic circuits.

Keywords: in-memory computing; memristor; binary decision diagrams; genetic algorithm; logic synthesis

随着数字数据的指数级增长，传统计算架构面临日益严峻的性能瓶颈[1]。当前数据密集型应用普遍受限于冯·诺依曼体系架构中存储与计算单元分离的设计，该设计导致内存与处理器间频繁的数据迁移，形成制约计算性能提升的存储墙问题。在此背景下，存算一体技术通过融合存储与计算功能，为突破这一瓶颈提供了新思路[2]。其中，基于非易失性存储器的实现方案因其物理特性优势备受关注[3]。忆阻器作为典型的非易失性存储器，不仅具备高密度集成特性[4-5]，并且能与标准互补金属氧化物半导体(Complementary Metal Oxide Semiconductor, CMOS)工艺兼容，这使得基于忆阻器的存算一体化架构成为当前研究热点[6-7]。

忆阻器由夹在上下金属电极之间的金属氧化物薄膜构成，其电阻值通过电场调控的离子迁移实现可逆改变[8]。通过在忆阻器上施加合适的电压，能够使得忆阻器的电阻在低阻态和高阻态之间实现可逆的转变[9]。忆阻器存储阵列通常会采用 1 个晶体管-1 个忆阻器(1Transistor-1Resistor, 1T1R)结构，通过将晶体管与忆阻器连接使得每个单元能够被单独选通，并且还能限制忆阻器漏电流的问题[10-11]。

在存算一体架构中，对逻辑“0”和“1”明确区分的数字逻辑计算可满足高精度计算需求[12]。当前主流的布尔逻辑实现方案包括：基于忆阻阵列的蕴含逻辑(IMPLY)[13-14]、忆阻器辅助逻辑(MAGIC)[15-16]、多数表决逻辑(MAJORITY)[17-18]、以及新近提出的基于流计算[19]和基于路径的新型忆阻逻辑[20-21]。通常，各布尔逻辑实现方案均包含编译与执行两个阶段：编译阶段均需忆阻阵列写操作，执行阶段中前 4 类方案需读写操作，而基于路径的忆阻逻辑仅需读操作。由于写操作涉及阻值改变，其不仅加速器件老化，还导致更高能耗[22-23]，因此，基于路径的忆阻逻辑方案在执行阶段同时具备低功耗与高可靠性的双重优势。

尽管上述的忆阻逻辑范式能够实现任意布尔逻辑函数，但是开发适配存算架构的自动化映射工具对实现高效布尔逻辑综合至关重要。现有研究已提出多种忆阻逻辑综合框架，如基于 MAGIC 的 CONTRA[24]、采用 MAJORITY 的 Arc[25]，以及基于 IMPLY 的优化方法[13]等经典框架；此外，还有新兴的基于流计算的 COMPACT[19]和基于路径计算的 PATH 框架[20]。这两个框架都通过将布尔函数转换为二元决策图(Binary Decision Diagrams, BDD)，并优化 BDD 到忆阻阵列的映射策略，进一步提升了硬件资源利用率。

然而，上述依赖于 BDD 的忆阻逻辑综合框架忽略了一个关键问题，其在映射前的 BDD 优化环节，仍沿用传统逻辑综合的思路，将减少 BDD 的总节点数和总边数作为核心目标，却未充分关注该逻辑计算框架中忆阻交叉阵列独特的映射机制与 BDD 结构之间的关联性[19-20]。在传统逻辑设计中，总节点数、总边数的精简确实能简化逻辑运算流程，但忆阻交叉阵列的硬件实现逻辑与传统电路存在本质差异，其硬件面积并非由 BDD 的总节点或边数量直接决定，而是与 BDD 结构中真正参与逻辑计算的关键路径特性相关。这种仅关注总规模、忽视架构适配性的优化方式，导致现有框架的 BDD 优化结果难以匹配忆阻阵列的映射需求，最终无法有效发挥忆阻器件在硬件面积优化上的潜力，也为后续忆阻逻辑电路的高效实现埋下了瓶颈。

在前人工作基础上，设计了一种基于自适应重启遗传算法(Adaptive Restart Genetic Algorithm, ARGA)的 BDD 重排序优化方案，并实现了集成该方案的逻辑综合框架。主要贡献包括：开发集成 ARGA 的逻辑综合框架，有效减小基于路径逻辑计算架构下的忆阻器阵列面积；针对 BDD 重排序过程中面临的局部最优陷阱问题，在算法中加入自适应重启机制，实现高效最优解搜索。

1　相关背景

1.1　二元决策图

二元决策图是一种基于有向无环图的布尔函数表示方法，其结构如图 1 所示[26]。

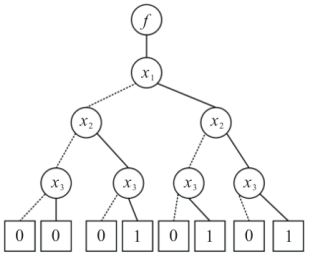


图 1　函数 $f=(x_1+x_2)\cdot x_3$ 的 BDD 结构

BDD 由两类节点构成：内部节点表示布尔变量，每个内部节点延伸出两条有向边（0 边和 1 边）指向子节点；终端节点表征布尔常量 0 和 1，作为逻辑路径的终止节点且无出边。通过香农展开定理

$$f(x_1,x_2,\cdots,x_i,\cdots,x_n)=\bar{x}_i f(x_1,x_2,\cdots,x_{i-1},0,x_{i+1},\cdots,x_n)+x_i f(x_1,x_2,\cdots,x_{i-1},0,x_{i+1},\cdots,x_n)\quad i=1,2,\cdots,n \tag{1}$$

式中：f 和 x_i 分别代表函数和第 i 个变量，任意布尔函数均可分解为 $f=x_i fx_i+\bar{x}_i f\bar{x}_i$，其中 fx_i 和 $f\bar{x}_i$ 分别对应变量 x_i 取 1 和 0 时函数 f 的值。以函数 $f=(x_1+x_2)\cdot x_3$ 为例，当变量顺序固定为 $x_1\rightarrow x_2\rightarrow x_3$ 时，如图 1 所示的 BDD 结构，内部结点 x_1 通过 0 边（对应 $\bar{x}$），连接子函数 $fx_1=(1+x_2)\cdot x_3$，通过 1 边（对应），连接子函数，这种分层决策机制通过对变量顺序的逐级展开，完整表征了布尔函数的逻辑行为。值得注意的是，BDD 还有一些其他的扩展形式：通过合并和消除冗余结构生成的降序二元决策图（Reduced Ordered Binary Decision Diagrams，ROBDD）以及通过共享子图来支持多输入多输出的共享二元决策图（Shared Binary Decision Diagrams，SBDD）[26]。

1.2 忆阻交叉阵列

典型的 1T1R 交叉阵列由字线、位线、选择线和字线与位线连接处的晶体管－忆阻器（1T1R）单元构成[28]，如图 2 所示。每个 1T1R 单元由串联的忆阻器和晶体管构成，其中晶体管栅极垂直连接至公共选择线。该交叉阵列具有双重开关特性：忆阻器通过高低阻实现开关功能，当对目标位线施加特定电压时，忆阻器可在低阻（“开”）与高阻（“关”）之间可逆切换。晶体管则作为另一个开关，通断状态由选择线电压控制。

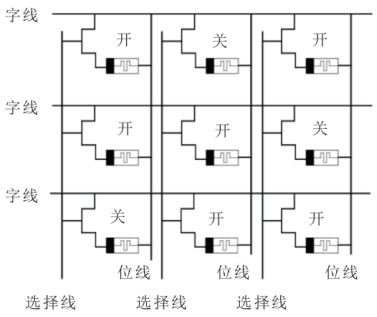


图 2 1T1R 交叉阵列

1.3 基于路径的忆阻逻辑计算

基于路径的忆阻逻辑计算通过检测忆阻阵列中的导电路径实现布尔函数运算[20]，其逻辑计算如图 3 所示，包含编译阶段和执行阶段。编译阶段将硬件描述语言定义的布尔函数（如图 3(a)）转化为抽象交叉阵列设计（如图 3(b)），该抽象交叉阵列设计明确各单元忆阻器的目标状态（低阻“1”/高阻“0”）、选择线（Selector lines）绑定的布尔变量（如 $SL_1\rightarrow x_1$，$SL_2\rightarrow x_2$，$SL_3\rightarrow x_3$）以及输入/输出端口映射的字线位置。

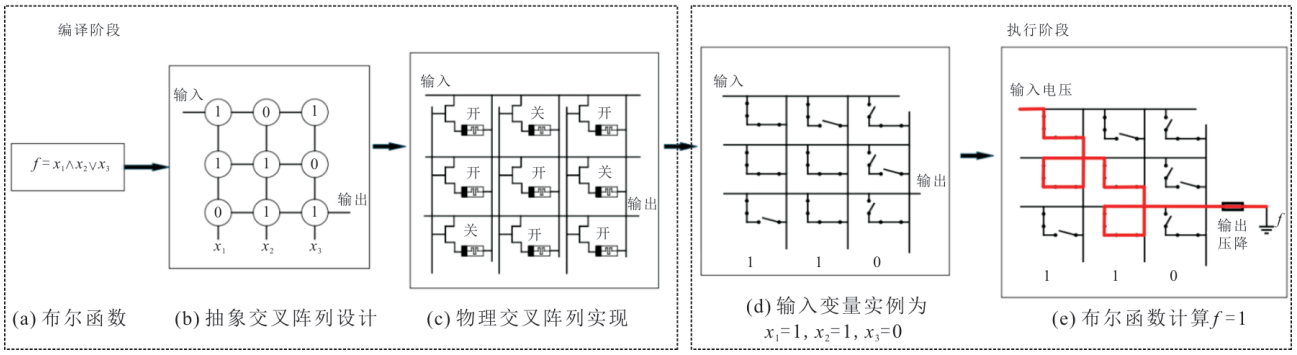


图 3 基于路径的逻辑计算

通过位线施加定向电压脉冲，将对应忆阻器编程为低阻（“开”）或高阻（“关”）状态（如图 3(c)）。在执行阶段，将输入变量加载至选择线，从而控制晶体管的开关状态（如图 3(d)）。输入字线施加电压值：若为高电平则函数为真，反之为假。例如输入向量 $(x_1,x_2,x_3)=(1,1,0)$ 时（如图 3(e)所示），存在一条输入到输出的导通路径，函数评估为 1。

现有的 PATH 框架[20]已建立了完整的布尔逻辑综合方法。该方法通过将布尔函数的 BDD 映射至 1T1R 忆阻阵列，其流程包括 BDD 预处理、二分图转换、节点合并、图分区及阵列实现。本部分重点阐述 BDD 到阵列的映射机制。图 4 为布尔函数映射至交叉阵列步骤图，展示了编译阶段从布尔函数生成抽象阵列的全过程。具体流程如下：首先将布尔函数 $f=(c\vee(b\wedge a)\vee d)$ 转化为 BDD 结构（如图 4(a)所示）；随后进行预处理，即移除常量 0 节点及所有其关联边，形成精简 BDD（图 4(b)）；

最后在映射过程中，每个 BDD 节点对应一条字线，每条边对应一条选择线，节点与边的连接点标记为 1，非连接点标记为 0，最终生成图 4(c)所示的抽象阵列拓扑。

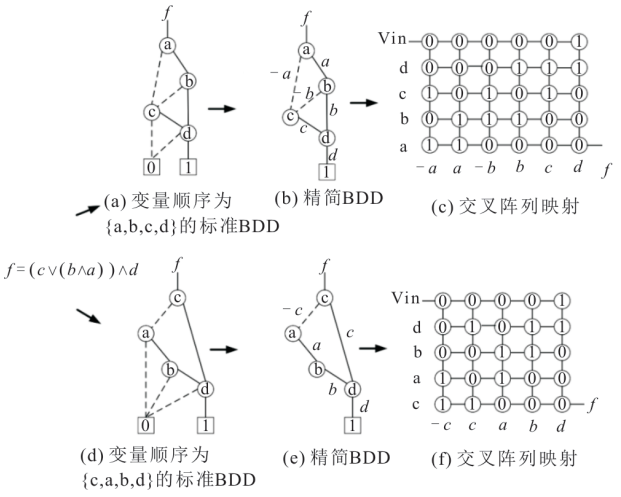


图 4　布尔函数映射至交叉阵列步骤图

需要注意的是，变量顺序的差异会导致生成的 BDD 结构不同，从而直接影响映射后交叉阵列的规模。例如，如图 4(a) 所示，由变量顺序(a,b,c,d)生成的 BDD 结构，在经过预处理后生成 5 个结点和 6 条非 0 连接边的精简 BDD(如图 4(b)所示)，其最终生成的忆阻交叉阵列大小为 5×6。而对于同一布尔函数，当变量顺序为(c,a,b,d)时，生成的 BDD 结构如图 4(d)所示，在经过预处理后生成 5 个结点 5 条非 0 连接边的精简 BDD(图 4(e) 所示)，其最终生成的忆阻交叉阵列大小为 5×5。然而，现有框架尚未考虑到该特性对综合性能的影响。所提方法则通过优化 BDD 变量顺序来提升映射策略的资源利用效率。

2　基于 ARGA 的 BDD 重排序算法

本节首先介绍所提框架的整体流程，随后对 BDD 重排序优化问题进行表述并设计优化目标函数，最后再详细描述优化算法实现细节。

2.1　逻辑综合框架

图 5 为整体框架，详细展示了集成 ARGA 重排序算法的逻辑综合框架全流程。流程以 PLA 等格式的布尔函数描述文件为输入，首先进入 BDD 构建环节，借助 ABC 工具将输入文件转换为标准 BDD 结构，这一步为后续的优化与映射提供了基础的逻辑表示形式。随后，进入 BDD 优化阶段，利用所提出的 ARGA 重排序算法，以目标函数 $\min(N\cdot E\cdot(1+\log(n_p)))$ 为约束，对标准 BDD 进行变量顺序优化，生成更利于后续硬件映射的 BDD 结构，该结构能有效降低行/列映射代价。完成优化后，进入交叉阵列映射环节，先对优化后的 BDD 进行预处理，保留非 0 路径以精简逻辑表示，接着为 BDD 节点分配字线、为边分配位线，完成从逻辑结构到忆阻交叉阵列硬件资源的映射。最终，输出可执行布尔逻辑计算的忆阻交叉阵列，实现从布尔函数到硬件阵列的完整转化流程。

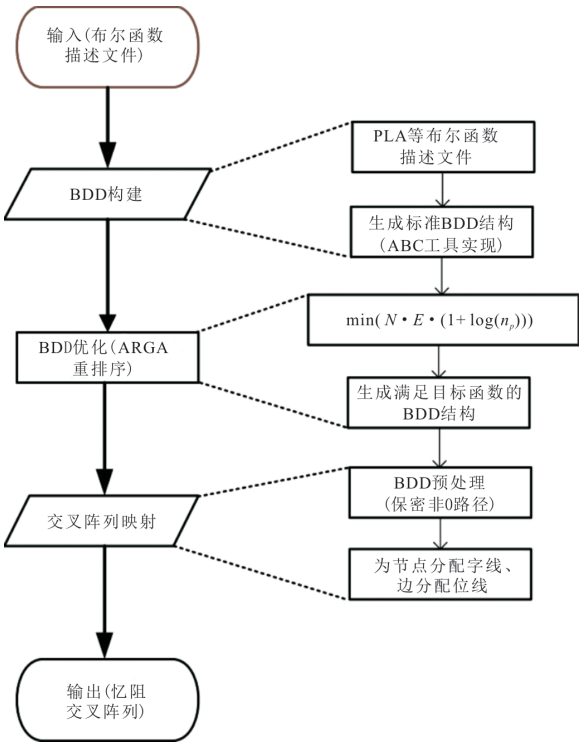


图 5　整体框架

2.2　逻辑综合优化的问题表述

基于自适应重启遗传算法(ARGA)的 BDD 重排序算法，其目标是为布尔函数寻找最优变量顺序的 BDD 结构。该结构需满足：拥有最少的节点数、非 0 连接边数以及最少非 0 路径数。具体优化目标函数定义为

$$\min(N\cdot E\cdot(1+\log(n_p))) \tag{2}$$

式中：N、E 和 n_p 分别代表当前变量顺序下 BDD 的结点数、非 0 路径边数以及非 0 路径的数量。该目标函数的意义在于：结点数 N 与非 0 路径边数 E 决定了映射后忆阻交叉阵列的行数与列数，而非 0 路径数 n_p 反映了函数执行时存在的路径数量。通过协同优化这 3 个参数，算法能够在有限迭代次数内生成最优变量顺序，从而提升阵列映射效率。

2.3 算法实现

基于 ARGA 的 BDD 重排序算法描述了变量顺序优化流程。该算法旨在预设的最大迭代次数 G_{Max} 内，对 BDD 的初始变量顺序 O 进行优化，最终得到适应度值 S 最小的最优变量顺序 O_{best}。在该重排序算法中，种群 P 内的每个个体 p 均包含两个核心属性：一是当前对应的 BDD 变量顺序 O，二是该变量顺序下计算得到的适应度值 S。同时，为实现自适应重启机制，算法额外设置了重启阈值 T。算法伪代码如下。

算法：基于 ARGA 的 BDD 重排序算法

输入：BDD $p=\{O,S\}$
输出：BDD $p^*=\{O_{\text{best}},S_{\text{best}}\}$
1：$P\leftarrow\{p_0,p_1,\cdots,p_{n-1}\}$
2：$S^*\leftarrow$EvaluateFitness(P,p^*)
3：FOR $g=0$ to $G_{\max}$ DO
4：　IF $u>T$ THEN
5：　　$u\leftarrow0$,restart(P)
6：　END IF
7：　FOR $i=0$ to $n-1$ DO
8：　　$p_{t1}\leftarrow$tournamentSelection(P)
9：　　$p_{t2}\leftarrow$tournamentSelection(P)
10：　　$p\leftarrow$PMX(p_{t1},p_{t2})MU(p_{t1})
11：　　$P'\leftarrow P'\cup p'$
12：　END FOR
13：　$P\leftarrow P'$
14：　$S'\leftarrow$EvaluateFitness(P',p')
15：　IF $S^*<S'$ THEN
16：　　$S^*\leftarrow S',p^*\leftarrow p'$
17：　ELSE
18：　　$u\leftarrow u+1$
19：　END IF
20：END FOR
21：REUTRN p^*

输入为包含初始变量顺序 O 及其适应度 S 的 BDD 结构，输出为具有最优顺序 O_{best} 和最小适应度 S_{best} 的 BDD。具体执行流程为：首先构建包含 n 个个体的初始种群 P，其中 p_0 保留 BDD 原始变量顺序，其余个体通过 Fisher-Yates 算法随机生成，兼顾解的稳定性与搜索多样性。随后计算初始种群各成员的适应度后，记录当前最优解 p 及其适应度 S。接下来在预设的 G_{Max} 次迭代中，算法持续监测最优解的更新状态：若最优解连续 T 次迭代未改进，则触发重启机制，即保留当前最优个体及 10% 种群成员，其余个体重新随机生成以突破局部最优。未触发重启时，每次迭代通过锦标赛选择法从种群中选取父代个体 p_{t1} 与 p_{t2}，根据预设交叉率/变异率执行部分映射交叉（PMX）或单点变异（MU），生成新个体 p' 并构建新种群 P'。更新种群后重新评估适应度，若发现更优解则更新 p 与 S，否则累计停滞计数器 u。最终当迭代次数达到 G_{Max} 后返回最优变量顺序 p^* 生成的 BDD。该算法核心操作机制包括：1）锦标赛选择确保优质基因传递；2）PMX 交叉保留有效变量顺序片段；3）置换变异增强局部搜索能力；4）停滞计数器驱动的自适应重启机制，有效缓解组合优化中的早熟收敛问题。

3 实验数据及分析

为评估所提出的逻辑综合优化方法，实验选取 Revlib 与 EPFL 基准库中的 17 个多输出布尔函数作为测试用例。实验平台通过集成 ABC 逻辑综合工具与 PATH 框架[20]实现完整流程。

通过多次实验验证，为保证收敛精度与计算效率的最优平衡，其中 ARGA 的关键参数设置如下：最大迭代次数 $G_{\text{Max}}=1\,000$，种群规模 $n=20$，重启阈值 $T=30$。在功耗评估环节，为确保横向对比的公平性，实验的参数设置与计算方式均参照文献[20]，设定单忆阻阵列规模为 128 * 128，总线与忆阻阵列功耗分别设定为 13 mW 与 0.3 mW，时延分别设定为 15 ns 与 100 ns。

表 1 对比了所提方法与文献[20]的预处理 BDD 规模差异。实验数据表明，优化后的 BDD 结构在节点数量上平均降低 9%，非 0 连接边数量减少 10%。当映射至未分区的交叉阵列时，阵列行数（由节点数决定）与列数（由边数决定）分别缩减 9% 与 10%，总面积优化幅度达到 15%。该结果验证了变量顺序优化对硬件资源占用的显著改善效果。

表 1 所提方法与 PATH[20] 中生成的 BDD 大小比较，以及单阵列映射面积（行×列）比较

基准测试	PATH[20]		所提方法		单阵列映射（PATH[20]）			单阵列映射（所提方法）		
	节点数	边数	节点数	边数	行数	列数	面积	行数	列数	面积
in0	384	680	314	499	384	565	216 960	314	419	131 566
apex2	566	1042	331	600	566	879	497 514	331	462	152 922

续表 1　所提方法与 PATH[20] 中生成的 BDD 大小比较，以及单阵列映射面积（行×列）比较

基准测试	PATH[20]		所提方法		单阵列映射（PATH[20]）			单阵列映射（所提方法）		
	节点数	边数	节点数	边数	行数	列数	面积	行数	列数	面积
spla	593	864	597	846	593	767	454 831	597	750	447 750
pdc	620	887	610	881	620	750	465 000	610	740	451 400
misex3	673	1 094	554	895	673	849	573 177	554	675	373 950
tial	896	1 717	854	1 562	896	1 143	1 024 128	854	1 104	942 816
apex4	989	1 874	972	1 830	989	1 157	1 144 273	972	1 151	1 118 772
cps	1 079	1 633	994	1 561	1 079	1 248	1 346 592	994	1 165	1 158 010
apex5	1 258	2 387	1 061	1 913	1 258	2 132	2 682 056	1 061	1 790	1 899 190
seq	1 301	2 041	1 275	2 035	1 301	1 560	2 029 560	1 275	1 586	20 022 150
cavlc	435	776	398	701	435	530	230 550	398	476	189 448
ctrl	88	128	86	124	88	100	8800	86	98	8 428
dec	511	510	511	510	511	510	260 610	511	510	260 610
i2c	1 203	1 936	1 109	1 820	1 203	1 837	22 09 911	1 109	1 721	1 908 589
int2float	158	301	129	218	158	265	41 870	129	170	21 930
priority	771	1 539	771	1 539	771	1 539	1 186 569	771	1 539	1 186 569
router	218	379	180	311	218	351	76 518	180	299	53 820
归一化平均值	1.00	1.00	0.91	0.90	1.00	1.00	1.00	0.91	0.90	0.85

实验进一步对比了不同忆阻逻辑框架在执行阶段的功耗与时延，结果分别如图 6、图 7 所示。从图 6 和图 7 可以看出，所提方法框架与 PATH 框架[20]相比，能耗与时延分别降低 26% 和 12%。所提方法的能耗较 COMPACT[19] 与 CONTRA[24] 框架分别降低 3 个与 4 个数量级。而所提方法的时延较 COMPACT 与 CONTRA 框架分别降低 80% 和 97%。此外，CONTRA 采用多级逻辑综合优化方法对函数对应的 AIG 进行优化，然后采用查找表映射（Look-Up Table mapping，LUT mapping）方式进行映射，在该过程中充分利用 MAGIC-或非门在忆阻阵列的并行性来加速运算。

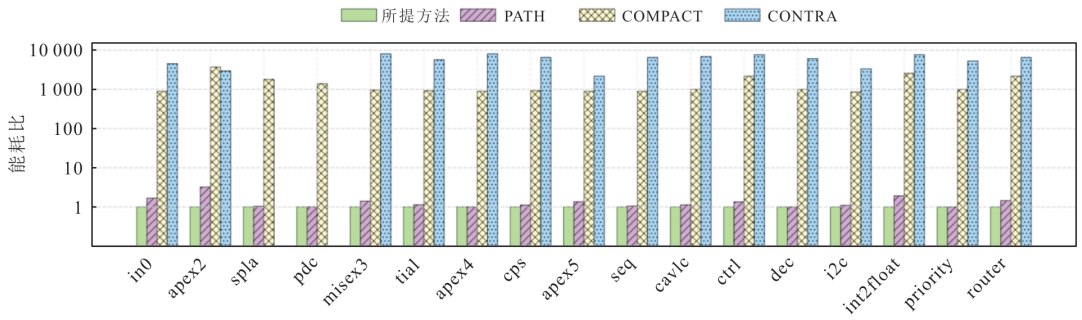


图 6　所提方法与其他忆阻逻辑综合方法能耗对比

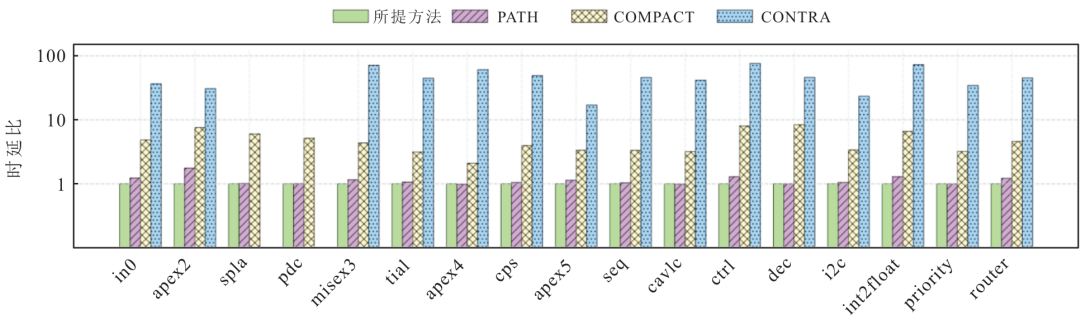


图 7　所提方法与其他忆阻逻辑综合方法时延对比

然而，并行执行 MAGIC-或非运算对输入的实际映射位置要求较高，即，并行运算的输入数据需要位于忆阻阵列同行不同列。因此，CONTRA 需要大量的数据对齐操作，进而造成额外的延时开销。与其相比，所提方法通过硬件原生的路径连续性，从根本上规避了传统方法的数据对齐问题，实验证明可显著降低延时与能耗。

在 BDD 变量顺序优化问题中，该问题本质属于 NP-hard 问题。当待优化的变量数量达到一定规模时，算法在搜索过程中极易陷入局部最优解，此时搜索得到的变量顺序难以进一步优化，导致整体优化性能停滞。为解决这一问题，该重排序算法引入自适应重启机制。当算法连续 30 次迭代未更新最优解时，会自动保留当前最优个体并重置 90% 的种群成员，通过引入新的搜索个体打破局部最优约束，进而增强算法的全局搜索能力。

为验证该机制的有效性，图 8 给出了适应度值随迭代次数的演化过程。结果显示，当迭代次数达到 500 次时，启用该重启机制的算法，其适应度值较基准方法提升 20.7%，充分证明了该机制对突破局部最优、提升优化性能的作用。

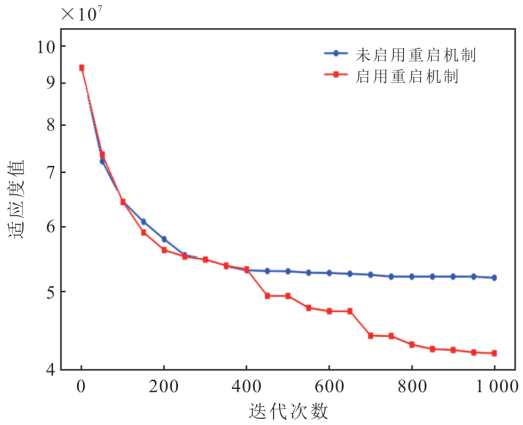


图 8 适应度值随迭代次数的变化

在传统逻辑综合中，BDD 优化聚焦于抽象层面的规模压缩（如节点数、边数的最小化）；而在忆阻逻辑综合中，所提方法更注重 BDD 结构与忆阻阵列映射约束的协同优化，特别是变量顺序对映射效果的影响。需特别指出，基于路径计算的逻辑综合框架在实现面积上存在固有缺陷。其根本原因在于结构性约束：为实现 BDD 中所有可能的逻辑路径，必须将每个节点完整映射为阵列中的独立行，每条边映射为独立列。这种刚性映射机制导致硬件资源利用率存在理论瓶颈——阵列行数等于节点总数，列数等于边总数，且需为所有潜在路径预留物理连接。这本质上限制了面积压缩的空间，成为制约路径计算框架发展的核心问题。

4 结语

针对基于路径计算的框架在实现面积上存在的缺陷，所提方法设计了基于 BDD 重排序优化的忆阻逻辑综合框架。通过基于 ARGA 的 BDD 重排序算法，生成更适配忆阻交叉阵列的 BDD 结构，尽可能减少映射过程中需实现的 BDD 结构中非 0 路径下的节点与边数，从而以减少硬件面积，实现了忆阻交叉阵列硬件资源的高效利用。并构建了从布尔函数→BDD 优化→物理映射的完整逻辑综合流程。实验结果表明，该框架使阵列面积平均减少 15%，执行功耗和时延分别降低 26% 和 12%，总体性能优于传统路径计算方法。实验结果验证了在忆阻阵列映射约束下对 BDD 结构的优化，能够有效提升忆阻逻辑综合效率，为存算一体技术中忆阻逻辑电路的高效设计提供了可行路径。

然而，在映射过程中，受必须为节点与边都分配字线和位线的机制影响，即使经过现有预处理分区操作，映射后的忆阻交叉阵列实际使用率仍不高，通常体现在交叉阵列对角区域的忆阻器多赋值为 0（高阻状态）。这也是基于路径的忆阻逻辑计算面积开销较大的原因，后续工作应考虑开发更高效的分区算法，在确保尽可能少引入额外分区开销的同时，提升忆阻阵列使用率，从而进一步减少映射面积。

参 考 文 献

[1] DUARTE F. Amount of data created daily (2025) [EB/OL]. [2025-08-10]. https://explodingtopics.com/blog/data-generated-per-day.

[2] HAQ RASHED M R, THIJSSEN S, JHA S K, et al. Automated synthesis for in-memory computing [C]//2023 IEEE/ACM International Conference on Computer Aided Design. San Francisco: IEEE, 2023: 1-9.

[3] 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.

[4] PHILIP WONG H S, LEE H Y, YU S M, et al. Metal-oxide rram[J]. Proceedings of the IEEE, 2012, 100(6): 1951-1970.

[5] 孙广辉，徐兆青. 基于有源电感的二极管桥忆阻模拟器[J]. 电子设计工程，2019(24)：66-69.
SUN G H, XU Z Q. Generalized memristor emulator based on diode bridge and active inductor[J]. Electronic Design Engineering，2019(24)：66-69.

[6] KUMAR M, WU M H, HOU T H, et al. CMOS-RRAM based non-volatile ternary content addressable memory (nvTCAM)[J]. IEEE Transactions on Nanotechnology, 2024, 23: 203-207.

[7] ALI HUSSAIN S, PRASAD V P N S B S V, BEVARA V, et al. A high-speed low-power CMOS-memristor based hybrid comparator using m_GDI technique for IoT applications[C]//2022 IEEE International Symposium on Smart Electronic Systems (iSES). Warangal: IEEE, 2022: 631-634.

[8] WANG Z Y, NALLA P S, KRISHNAN G, et al. Digital-assisted analog in-memory computing with RRAM devices[C]//2023 International VLSI Symposium on Technology, Systems and Applications. Hsin-Chu：IEEE, 2023: 1-4.

[9] 刘东青，程海峰，朱玄，等. 忆阻器及其阻变机理研究进展[J]. 物理学报，2014，63(18)：20-28.
LIU D Q, CHENG H F, ZHU X, et al. Research progress of memristors and memristive mechanism [J]. Acta Physica Sinica, 2014, 63(18): 20-28. (in Chinese)

[10] 陈传兵，许晓欣，李晓燕，等. 1T1R 结构 RRAM 的故障可测性设计[J]. 半导体技术，2018，43(5)：388-393.
CHEN C B, XU X X, LI X Y, et al. Fault measurability design of the RRAM based on 1T1R structure [J]. Semiconductor Technology, 2018, 43(5): 388-393. (in Chinese)

[11] LIU X H, BENGEL C, CüPPERS F, et al. Effect of transistor transfer characteristics on the programming process in 1T1R configuration[J]. IEEE Transactions on Electron Devices, 2024, 71(4): 2423-2430.

[12] GLINT T, PAUL G, BENDE A, et al. Resilience of digital and analog RRAM-based ML models to device variability: A comparative study[C]//2024 31st IEEE International Conference on Electronics, Circuits and Systems. Nancy：IEEE, 2024: 1-4.

[13] LEHTONEN E, POIKONEN J, LAIHO M. Implication logic synthesis methods for memristors[C]//2012 IEEE International Symposium on Circuits and Systems. Seoul：IEEE, 2012: 2441-2444.

[14] MALAGAR S, P D, S R. Implementation of a 16-bit multiplier leveraging the kogge-stone adder with memristor IMPLY logic[C]//2024 International Conference on Distributed Systems, Computer Networks and Cybersecurity. Bengaluru：IEEE, 2024: 1-5.

[15] KVATINSKY S, BELOUSOV D, LIMAN S, et al. MAGIC: Memristor-aided logic[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2014, 61(11): 895-899.

[16] NABIPOUR S, DATTA K, WEINGARTEN L, et al. Multi-input MAGIC synthesis and verification for in-memory computing design[C]//2025 IEEE 55th International Symposium on Multiple-Valued Logic. Montreal：IEEE, 2025: 178-183.

[17] GAILLARDON P E, AMARú L, SIEMON A, et al. The programmable logic-in-memory (PLiM) computer [C]//2016 Design, Automation & Test in Europe Conference & Exhibition. Dresden: IEEE, 2016: 427-432.

[18] LAKSHMI V, REUBEN J, PUDI V. Majority logic based in-memory comparator[C]//2022 IEEE International Conference on Semiconductor Electronics. Kuala Lumpur：IEEE, 2022: 105-108.

[19] THIJSSEN S, JHA S K, EWETZ R. COMPACT: Flow-based computing on nanoscale crossbars with minimal semiperimeter and maximum dimension[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022, 41(11): 4600-4611.

[20] THIJSSEN S, RASHED M R H, JHA S K, et al. PATH: Evaluation of Boolean logic using path-based in-memory computing systems[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2024, 43(5): 1387-1400.

[21] SINHA P, CHAVAN A, RAJ S. Designing energy-efficient PATH-based decision tree memristor crossbar circuits[C]//2024 IEEE 24th International Conference on Nanotechnology. Gijon: IEEE, 2024: 209-213.

[22] LIAO Z H, FU J Y, WANG J H. Ameliorate performance of memristor-based ANNs in edge computing[J]. IEEE Transactions on Computers, 2021, 70(8): 1299-1310.

[23] 高耀梁，甘朝晖，尹力. 忆阻器的建模及高精度忆阻值读写电路的设计[J]. 电子器件，2014，37(6)：

1145-1150.

GAO Y L, GAN Z H, YIN L. Modeling of the memristor and high precision resistance read and write circuit design[J]. Chinese Journal of Electron Devices, 2014, 37(6): 1145-1150. (in Chinese)

[24] BHATTACHARJEE D, CHATTOPADHYAY A, DUTTA S, et al. CONTRA: Area-constrained technology mapping framework for memristive memory processing unit[C]//2020 IEEE/ACM International Conference on Computer Aided Design. San Diego: IEEE, 2020: 1-9.

[25] BHATTACHARJEE D, DEVADOSS R, CHATTOPADHYAY A. ReVAMP: ReRAM based VLIW architecture for in-memory computing [C]//Design, Automation & Test in Europe Conference & Exhibition (DATE). Lausanne: IEEE, 2017: 782-787.

[26] TOWHIDI F, HABIBI LASHKARI A, HOSSEINI R S. Binary decision diagram (BDD)[C]//2009 International Conference on Future Computer and Communication. Kuala Lumpar: IEEE, 2009: 496-499.

[27] MINATO S I, ISHIURA N, YAJIMA S. Shared binary decision diagram with attributed edges for efficient Boolean function manipulation[C]//27th ACM/IEEE Design Automation Conference. Florida: ACM, 1991: 52-57.

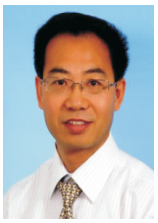
[28] WANG M, LIAN X J, PAN Y M, et al. A selector device based on graphene-oxide heterostructures for memristor crossbar applications[J]. Applied Physics A, 2015, 120(2): 403-407.

[作者简介]

刘鹏(1984—),男,湖南郴州人,博士,广东工业大学副教授,主要研究方向为集成电路设计与测试和神经网络架构设计。E-mail:liupeng@gdut.edu.cn

朱亚军(2001—),男,湖南娄底人,广东工业大学硕士研究生,主要研究方向为忆阻逻辑综合。E-mail:2586319842@qq.com

姚廉(1996—),男,湖北荆门人,广东工业大学博士研究生,主要研究方向为忆阻逻辑设计和逻辑综合。E-mail:1668870703@qq.com



武继刚(1963—),男,江苏沛县人,博士,广东工业大学教授,主要研究方向为高性能计算。E-mail:asjgwucn@outlook.com

[责任编辑:蔡秀梅]