结合因式分解与布尔表达式图的可逆电路综合方法

卜登立[1,2)]

[1)] (广西科技大学电气电子与计算机科学学院 柳州 545006)
[2)] (井冈山大学电子与信息工程学院 吉安 343009)
(bodengli@163.com)

摘 要: 为降低由布尔表达式图(BED)综合所得可逆电路的成本, 提出一种将因式分解与 BED 表示模型相结合的可逆电路综合方法. 给定布尔函数的积之异或和(ESOP)覆盖, 首先由 ESOP 立方体的共享零抑制多输出决策图表示借助代数除法对立方体实施因式分解, 并在此基础上构建 BED; 然后将 BED 结点映射为可逆门级联. 对基准函数的可逆电路综合结果表明, 该方法具有较高的时间效率. 与现有将有向无环图作为函数表示模型的综合方法相比, 该方法在许多情况下能降低综合所得可逆电路的量子成本和量子位数. 从平均角度看, 与结合变量分组和 BED 表示模型的综合方法相比, 该方法可将量子成本和量子位数分别降低 5.01%和 5.47%.

关键词: 可逆电路; 积之异或和展开; 因式分解; 共享零抑制多输出决策图; 布尔表达式图
中图法分类号: TP331.2; TP391.72 **DOI:** 10.3724/SP.J.1089.2021.18738

Reversible Circuit Synthesis Method by Combining Factoring and Boolean Expression Diagram

Bu Dengli[1,2)]

[1)] (*School of Electrical, Electronic and Computer Science, Guangxi University of Science and Technology, Liuzhou* 545006)
[2)] (*School of Electronics and Information Engineering, Jinggangshan University, Ji'an* 343009)

Abstract: In order to reduce the cost of the resulting circuits, a reversible circuit synthesis method by combining factoring and Boolean expression diagram (BED) is proposed. For a given cover representing the exclusive-sum-of-products (ESOP) expansion of a Boolean function, cubes in the ESOP expansion represented by shared zero-suppressed multiple-output decision diagrams are first factored by leveraging algebraic division, then a BED is built from the factored ESOP cover and a reversible circuit is synthesized from the BED by mapping a BED node to a cascade of reversible gates. The proposed synthesis method is evaluated using benchmark functions. Experimental results show that the proposed synthesis method is time efficient. Compared to the existing reversible circuit synthesis methods using a directed acyclic graph as the representation model for Boolean functions, the proposed synthesis method can reduce the quantum cost and the number of qubits of the resulting reversible circuits in many cases. On average, compared to the synthesis method by combining variables grouping and BED, the proposed synthesis method can reduce the quantum cost and the number of qubits by 5.01% and 5.47%, respectively.

Key words: reversible circuits; exclusive-sum-of-products expansion; factoring; shared zero-suppressed multiple-output decision diagrams; Boolean expression diagram

收稿日期: 2020-10-27; 修回日期: 2021-06-13. **基金项目**: 国家自然科学基金(61961023, 61640412); 江西省自然科学基金(20202BABL202007); 广西科技大学博士基金(21Z04). **卜登立**(1975—), 男, 博士, 副教授, 硕士生导师, CCF 会员, 主要研究方向为电路逻辑综合、量子电路综合、启发式优化算法.

因能实现信息无损的计算, 可逆逻辑在许多新兴技术领域, 特别是在量子计算领域有着广阔应用前景[1]. 量子计算机中执行经典计算的组件需要建模为可逆逻辑, 因此可逆电路综合被视为量子电路综合与设计中一个非常重要的步骤[2].

根据采用的函数表示模型, 现有可逆电路综合方法可分为功能综合方法和结构综合方法[3]. 功能综合方法因其采用的真值表、轮换或正极性 Reed-Muller 展开等函数表示模型具有指数级复杂度, 以及在处理不可逆函数时需预先将其嵌入为可逆函数, 故仅适用于规模相当小的函数[3]. 为改善功能综合方法的可扩展性, 文献[3]将不可逆函数的嵌入结合到可逆电路综合过程, 文献[4]将正极性 Reed-Muller 展开与决策图相结合, 但以上方法仍不能很好地应用于大规模函数.

结构综合方法使用积之异或和(exclusive-sum-of-products, ESOP)展开或使用有向无环图(directed acyclic diagram, DAG)表示函数, 通过将乘积项或 DAG 结点映射为可逆门级联, 不仅将不可逆函数的嵌入结合到电路综合过程, 而且能很好地处理大规模函数. 与采用 ESOP 表示模型相比, 采用 DAG 表示模型综合可逆电路能获得更低的量子成本, 有助于降低可逆逻辑量子电路实现的容错成本, 因此引起了人们较多的兴趣. 简约有序决策图(decision diagram, DD)作为布尔函数的标准形 DAG 表示形式, 在可逆电路综合中已经得到了广泛的应用. 文献[5]基于二元决策图(binary DD, BDD)综合可逆电路, 文献[6]研究了 BDD 的优化对综合所得可逆电路成本的影响, 文献[7]通过设计电路模板并采用子图同构技术降低可逆电路的量子位数和量子成本. BDD 的构建仅应用香农分解, Davio 分解或双条件分解有助于降低函数决策图表示的复杂度, 从而降低由决策图综合所得电路的成本. 因此, 有基于正 Davio 决策图(positive Davio DD, PDD)[8]、功能决策图(functional DD, FDD)[9], Kronecker 功能决策图(Kronecker FDD, KFDD)[10]或双条件 BDD(biconditional BDD, BBDD)[11]表示模型的可逆电路综合方法. 为降低量子成本和量子位数, 有在 KFDD 中使用补边并结合负控制线 Toffoli 门的方法[12], 以及对 FDD 或 KFDD 中的结点进行排序的方法[9,13]. 在描述布尔函数时, 与标准形 DAG 表示相比, 非标准形 DAG 表示形式常具有更低的复杂度, 因此, 近年来出现了一些采用非标准形 DAG 作为函数表示模型的可逆电路综合方法. 如采用与-非图(and-inverter graph, AIG)[14]、表决器-非图(majority-inverter graph, MIG)[14]或布尔表达式图(Boolean expression diagram, BED)[15]作为函数表示模型.

与 AIG 或 MIG 仅应用 inverter, and 或 majority 逻辑原语不同, BED 对应用的逻辑运算符没有限制, 在描述布尔函数时可能具有更低的复杂度. 然而, 由函数的 2 级逻辑描述构建 BED 时, BED 的复杂度依赖逻辑表达式中乘积项间公因子的识别[15]. ESOP 展开是函数的 2 级逻辑描述[16], 常使用立方体表示. 为降低由 ESOP 立方体集合构建所得 BED 的复杂度, 以及由 BED 综合所得可逆电路的成本, 本文借助共享零抑制多输出决策图(shared zero-suppressed multiple-output decision diagrams, SZMODD)对 ESOP 立方体实施因式分解, 并在此基础上使用 BED 表示模型综合可逆电路. 本文重点介绍了 SZMODD 表示模型以及借助其对 ESOP 立方体实施因式分解的过程, 并给出了结合因式分解与 BED 表示模型的可逆电路综合方法的算法描述. 最后使用 LGSynth[17]和 RevLib 函数[18]对所提出的综合方法进行了验证, 并与现有基于 DAG 表示模型的可逆电路综合方法的结果进行了比较.

1 可逆及量子电路

可逆电路由可逆门级联而成, 且电路网络中不允许直接使用扇出或反馈[1].

量子位是量子信息的单位, 量子门在量子位上执行本质上可逆的酉运算[4]. 量子电路由若干量子位以及作用于其中 1 个或 2 个量子位的量子门构成. 可逆电路可以采用不同类型的量子门实现, 最常用的量子门是 NCV(NOT, CNOT, controlled-V, controlled-V^+)门[2]. 因量子计算中容错的重要性, 近年来由 Clifford+T 门实现可逆电路也得到了较多关注[2-3].

可逆电路的成本常使用量子成本度量, 而量子成本与实现可逆电路采用的量子门有关. 当采用 NCV 门时, 量子成本称为 NCV 成本, 指的是实现可逆电路所需的 NCV 门的总数量[4]. 当采用 Clifford+T 门时, 常使用 T 深度评价可逆电路的量子成本, T 深度指的是可逆电路的 Clifford+T 门实现中需要被串行处理的 T 门数量[3]. 采用 NCV 门实现的电路称为 NCV 电路, 采用 Clifford+T 门实现的电路称为 Clifford+T 电路[2].

量子成本定义了可逆电路的计算复杂度[4], 直接影响量子电路实现的容错成本[2,19]. 可逆电路或

量子电路的硬件成本使用量子位数度量[4]. 对于可逆电路, 量子位数指的是电路中的电路线数.

本文使用 NOT, CNOT 和混合极性 Peres 门[15]综合可逆电路. 混合极性 Peres 门是 3 量子位可逆逻辑门, 其 NCV 门实现和 Clifford+T 门实现的量子成本请参阅文献[2,15].

2　基于 SZMODD 的因式分解

本文使用ESOP展开作为布尔函数的2级逻辑描述, 并借助 SZMODD 实施 ESOP 立方体的因式分解, 即提取 ESOP 乘积项间的公因子.

2.1　ESOP 展开的立方体表示

ESOP 展开由异或运算符连接乘积项而成[16]. 对于有 n 个输入变量 x_j ($1\leqslant j\leqslant n$)和 m 个输出变量 f_j ($1\leqslant j\leqslant m$)的布尔函数 $\boldsymbol{F}$, 常使用称为 ESOP 覆盖的立方体集合表示其 ESOP 展开. 采用位置标记法的多输出立方体表示为

$$\boldsymbol{c}=[\boldsymbol{i_c},\boldsymbol{o_c}]=[i_{\boldsymbol{c},1},i_{\boldsymbol{c},2},\cdots,i_{\boldsymbol{c},n},o_{\boldsymbol{c},1},o_{\boldsymbol{c},2},\cdots,o_{\boldsymbol{c},m}].$$

其中, $\boldsymbol{i_c}=[i_{\boldsymbol{c},1},i_{\boldsymbol{c},2},\cdots,i_{\boldsymbol{c},n}]$ 为立方体的输入部分, 表示乘积项 $\pi_{\boldsymbol{c}}=\prod_{j=1}^{n}\tilde{x}_j$, $i_{\boldsymbol{c},j}\in\{0,1,-\}$ 对应 $\tilde{x}_j\in\{\overline{x}_j,x_j,-\}$ ($1\leqslant j\leqslant n$); $\boldsymbol{o_c}=[o_{\boldsymbol{c},1},o_{\boldsymbol{c},2},\cdots,o_{\boldsymbol{c},m}]$ 为立方体的输出部分, $o_{\boldsymbol{c},j}\in\{0,1\}$ ($1\leqslant j\leqslant m$), $o_{\boldsymbol{c},j}=1$ 时表示函数输出变量 f_j 包含乘积项 $\pi_{\boldsymbol{c}}$.

2.2　SZMODD 表示模型

零抑制多输出决策图(zero-suppressed multiple-output DD, ZMODD)[20]是零抑制决策图(zero-suppressed DD, ZDD)[21]的扩展. 与 ZDD 相比, ZMODD 表示多输出立方体可以更好地体现乘积项在多个函数输出之间的共享[20].

ZMODD 有 2 个分别表示常量 0 和 1 的终端结点, 称为常量 0 结点和常量 1 结点. ZMODD 在表示多输出立方体 $\boldsymbol{c}$ 时, 若 $i_{\boldsymbol{c},j}\neq-$, 则将其映射为一个 ZMODD 输入变量结点 v; $i_{\boldsymbol{c},j}=1$ 和 $i_{\boldsymbol{c},j}=0$ 时, 结点 v 分别使用 ZMODD 输入变量 x_j 和 $\overline{x}_j$ 标记. 当 $o_{\boldsymbol{c},j}\neq0$ 时, 将其映射为使用 ZMODD 输出变量 f_j 标记的输出变量结点. 用以标记结点的变量 x_j, $\overline{x}_j$ 和 f_j 统称为 ZMODD 变量. ZMODD 仅有一个根结点.

本文提出 SZMODD 表示模型并借助其对立方体实施因式分解.

定义 1. 对于有 M 个立方体的集合 $\boldsymbol{C}=\{\boldsymbol{c}_l|1\leqslant l\leqslant M\}$, 采用同样的变量顺序, 为每个立方体 $\boldsymbol{c}_l$ 构建一个 ZMODD, 记为 $G_{\boldsymbol{c}_l}$. 这 M 个 ZMODD 通过共享子图构成一个 SZMODD, 记为 $\boldsymbol{G}=\{G_{\boldsymbol{c}_l}|1\leqslant l\leqslant M\}$. $\boldsymbol{G}$ 有 M 个根结点.

在构建 SZMODD 时, 由 ZMODD 输出变量和输入变量标记的结点分别统一位于 SZMODD 的上部和下部, 并且输入变量和输出变量标记的结点在 SZMODD 中的位置不会出现交叠.

在 SZMODD 中, 由一个根结点至常量 1 结点的路径形成一个多输出立方体 $\boldsymbol{c}$. 除终端结点外, 每个结点 v 均有 2 条出边, 分别为实线边和虚线边. 如果 v 为输入变量结点, 那么实线边或虚线边分别表示该 ZMODD 输入变量出现或不出现在立方体 $\boldsymbol{c}$ 对应的乘积项 $\pi_{\boldsymbol{c}}$ 中. 如果 v 为输出变量结点, 那么实线边或虚线边分别表示该 ZMODD 输出变量包含或不包含立方体 $\boldsymbol{c}$ 对应的乘积项 $\pi_{\boldsymbol{c}}$.

2.3　迭代因式分解

假设已知布尔函数 $\boldsymbol{F}$ 的 ESOP 覆盖 $\boldsymbol{C}$, 图 $\boldsymbol{G}$ 的结点数使用 $|\boldsymbol{G}|$ 表示, 下面给出立方体因式分解的算法描述.

算法 1. ESOP 立方体因式分解算法.

Step1. 由 ESOP 覆盖 $\boldsymbol{C}$ 构建 SZMODD, 得到 $\boldsymbol{G}$.

Step2. 对 $\boldsymbol{G}$ 实施 ZMODD 变量排序.

Step3. 由 $G_{\boldsymbol{c}_1}$ 至 $G_{\boldsymbol{c}_M}$, 采用深度优先后序方式遍历 $\boldsymbol{G}$, 如果图 $\boldsymbol{G}$ 中没有入度大于 1 的结点, 转至 Step6; 否则, 执行下一步.

Step4. 假设图 $\boldsymbol{G}$ 中入度大于 1 的结点为 v_j ($1\leqslant j\leqslant|\boldsymbol{G}|$), v_j 至常量 1 结点的路径为 p. 如果 p 中不包含 ZMODD 输入变量结点, 转至 Step6; 否则, 执行下一步.

Step5. 假设路径 p 中距离 v_j 最近的 ZMODD 输入变量结点为 v_k ($1\leqslant k\leqslant|\boldsymbol{G}|$), 若 v_j 为输入变量结点, 则令 $v_k=v_j$. 由以 v_k 为根结点的子图 G_{s} 提取公因子 $\boldsymbol{c}_{\mathrm{s}}$, 记录包含此公因子的 $G_{\boldsymbol{c}_l}$ ($G_{\boldsymbol{c}_l}\in\boldsymbol{G}$). 令 $\boldsymbol{G}=\boldsymbol{G}\setminus G_{\mathrm{s}}$, 并转至 Step2.

Step6. 遍历 $\boldsymbol{G}$ 得到因式分解后每个 $G_{\boldsymbol{c}_l}$ 对应的立方体 $\boldsymbol{c}_l$, 并用链表建立各个立方体 $\boldsymbol{c}_l$ 与所提取的公因子 $\boldsymbol{c}_{\mathrm{s}}$ 之间的联系, 得到立方体集合 $\boldsymbol{C}_{\mathrm{s}}$.

算法 1 中, Step2 采用成熟且具有较高时间效率的 Sifting 技术[21]对图 $\boldsymbol{G}$ 实施变量排序. Step4 和 Step5 中, 如果结点 v_j 被多个 ZMODD 共享, 说明 ZMODD 输入变量结点 v_k 被多个 ZMODD 共享, 由 v_k 至常量 1 结点的路径形成的乘积项(即公因子)

被多个立方体共享. Step5 中的$\boldsymbol{G}\backslash G_{\mathrm{s}}$采用 ZDD 代数除法[21]实现. 公因子$\boldsymbol{c}_{\mathrm{s}}$使用立方体表示.

例 1. 某函数的 ESOP 立方体表示如图 1 所示, 该函数包含 4 个输入和 3 个输出, 共有 5 个立方体, 分别如图 1 中的$\boldsymbol{c}_1 \sim \boldsymbol{c}_5$所示.

采用相同的变量顺序, 将每个立方体均表示为一个 ZMODD, 得到 SZMODD 图$\boldsymbol{G}$. 对图$\boldsymbol{G}$实施变量排序后得到如图 2a 所示的 SZMODD. 为简单起见, 图 2 的 SZMODD 中仅给出结点间的实线边以及常量 1 结点. 为描述方便, 本文中的公因子也使用乘积项表示.

由图 2a 可以看出, 输入变量结点v_1的入度大于 1. 由v_1至常量 1 结点的路径形成乘积项x_3x_4. 可知$\boldsymbol{c}_1$与$\boldsymbol{c}_2$包含公因子x_3x_4. 提取公因子x_3x_4后, 对 SZMODD 实施变量排序, 结果如图 2b 所示.

	x_1	x_2	x_3	x_4	f_1	f_2	f_3
$\boldsymbol{c}_1$	1	1	1	1	1	1	1
$\boldsymbol{c}_2$	–	–	1	1	1	1	0
$\boldsymbol{c}_3$	1	1	–	–	1	0	1
$\boldsymbol{c}_4$	–	0	1	–	1	1	1
$\boldsymbol{c}_5$	0	–	–	1	1	1	1

图 1 ESOP 立方体表示

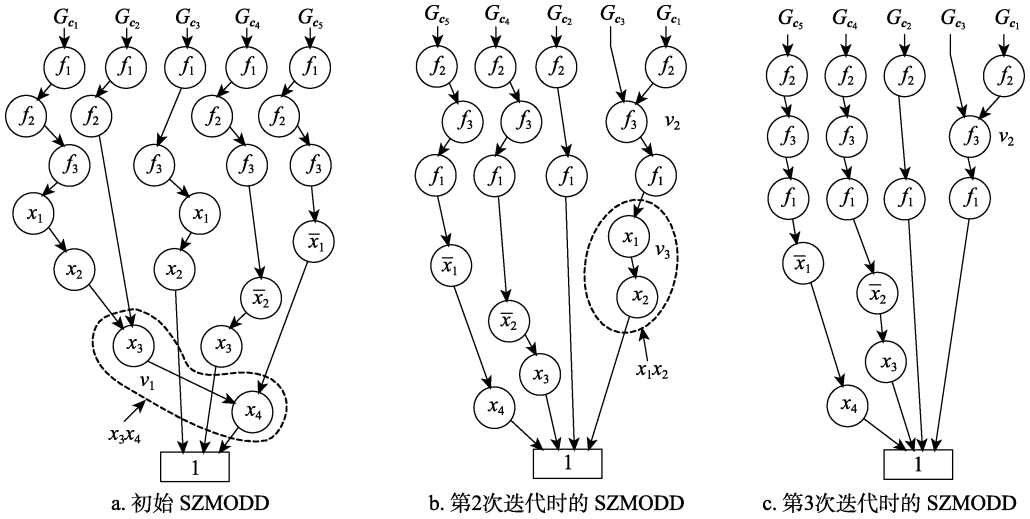

图 2 迭代因式分解示意图

由图 2b 可以看出, 输出变量结点v_2的入度大于 1. v_2至常量 1 结点的路径中距离v_2最近的输入变量结点为v_3, 由v_3至常量 1 结点的路径形成乘积项x_1x_2. 可知$\boldsymbol{c}_1$与$\boldsymbol{c}_3$包含公因子x_1x_2. 提取公因子x_1x_2后, 对 SZMODD 实施变量排序, 结果如图 2c 所示.

由图 2c 可知, SZMODD 不再共享 ZMODD 输入变量结点, 因式分解迭代过程结束. 得到如图 3 所示的因式分解后的立方体集合$\boldsymbol{C}_{\mathrm{s}}$. 立方体$\boldsymbol{c}_{\mathrm{s}_1}$与$\boldsymbol{c}_{\mathrm{s}_2}$的输出部分中的"–"表示输出$o_{c_{\mathrm{s}},j}$没有意义.

	x_1	x_2	x_3	x_4	f_1	f_2	f_3
$\boldsymbol{c}_1$	–	–	–	–	1	1	1
$\boldsymbol{c}_2$	–	–	–	–	1	1	0
$\boldsymbol{c}_3$	–	–	–	–	1	0	1
$\boldsymbol{c}_4$	–	0	1	–	1	1	1
$\boldsymbol{c}_5$	0	–	–	1	1	1	1
$\boldsymbol{c}_{\mathrm{s}_1}$	–	–	1	1	–	–	–
$\boldsymbol{c}_{\mathrm{s}_2}$	1	1	–	–	–	–	–

图 3 因式分解后的立方体集合

立方体间可能存在单文字公因子x_j或$\overline{x}_j$. 单文字公因子对降低 BED 的复杂度没有太大帮助. 因此, 对于单文字公因子a, 如果存在相邻公因子b, 则提取将a与b合并后的公因子; 否则忽略单文字公因子a, 即不提取该公因子.

图 4 给出了单文字公因子合并的示例, 假设 SZMODD 的根结点至v_1的路径中没有 ZMODD 输入变量结点被共享.

对于图 4a 所示的情形 1, 仅存在一个单文字公因子x_k, 因此不提取该公因子. 对于图 4b 所示

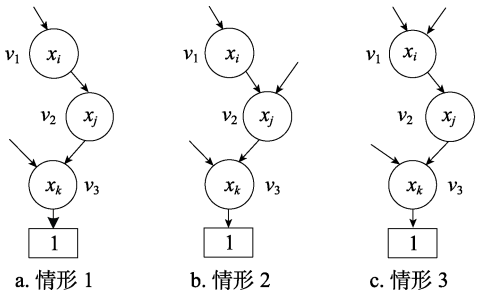

图 4 单文字公因子合并示例

的情形 2, 因式分解的第 1 次迭代中, 结点 v_3 的入度大于 1, 说明存在公因子 x_k, 由于 x_k 是单文字公因子, 故不提取该公因子, 直接进入第 2 次迭代; 第 2 次迭代中, 结点 v_2 的入度大于 1, 说明存在公因子 x_jx_k, 故提取公因子 x_jx_k.

类似地, 对于图 4c 所示的情形 3, 由于 x_k 是单文字公因子, 故将其与 x_ix_j 合并, 得到公因子 $x_ix_jx_k$.

3　可逆电路的综合及验证

3.1　可逆电路综合

假设已知布尔函数 $\boldsymbol{F}$ 的 ESOP 覆盖 $\boldsymbol{C}$, 下面给出本文可逆电路综合的算法描述.

算法 2. 可逆电路综合算法.

Step1. 使用算法 1 对 $\boldsymbol{C}$ 中的立方体实施因式分解, 得到立方体集合 $\boldsymbol{C}_{\mathrm{s}}$.

Step2. 由 $\boldsymbol{C}_{\mathrm{s}}$ 构建 BED.

Step3. 深度优先遍历 BED, 将结点映射为可逆门级联, 并构建实现函数 $\boldsymbol{F}$ 功能的电路.

算法 2 的 Step2 中, 由 $\boldsymbol{C}_{\mathrm{s}}$ 构建 BED 时, 采用递归方式解析立方体, 即对于立方体 $\boldsymbol{c}$, 先解析其包含的公因子 $\boldsymbol{c}_{\mathrm{s}}$, 再解析立方体 $\boldsymbol{c}$. Step3 采用 BED 结点映射方法[15].

算法 2 所得电路的电路线数即为量子位数. 电路的 NCV 成本为电路中所有可逆门的 NCV 成本之和, T 深度则估算为电路中所有可逆门的 T 深度之和.

假设电路线数为 q, 那么电路有 q 个输入, 其中 n 个输入分别使用 x_k ($1\leqslant k\leqslant n$)标记, 表示函数 $\boldsymbol{F}$ 的输入 x_k, 其他 $q-n$ 个输入为辅助输入, 其输入值为常量. 同时电路有 q 个输出, 其中 m 个输出分别使用 f_j ($1\leqslant j\leqslant m$)标记, 输出 $\boldsymbol{F}$ 的输出变量 f_j 的计算结果, 其他 $q-m$ 个输出为垃圾输出[2].

例 2. 以例 1 中的函数为例, 得到如图 3 所示的立方体集合 $\boldsymbol{C}_{\mathrm{s}}$ 后, 由 $\boldsymbol{C}_{\mathrm{s}}$ 构建如图 5 所示的 BED.

图 6 所示为由图 5 所示 BED 综合例 1 中的函数所得电路, 其中, v_j ($5\leqslant j\leqslant 19$)指示的可逆门级联由图 5 中的结点 v_j 映射所得; 虚线框中的可逆门为混合极性 Peres 门[15]. 该电路由 7 个混合极性 Peres 门和 10 个 CNOT 门组成, 其量子位数为 11, 其 NCV 门实现的量子成本, 即 NCV 成本为 7×4+10×1=38, Clifford+T 门实现的 T 深度为 7×3+10×0=21.

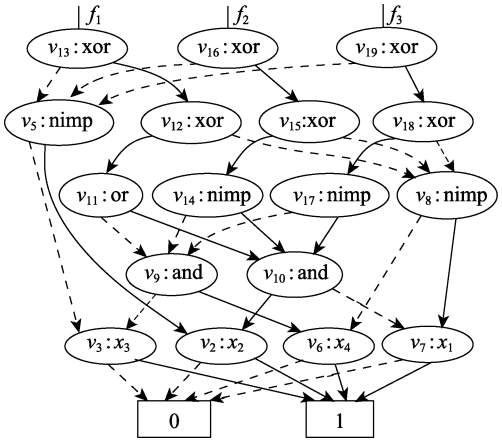

图 5　因式分解后的 BED

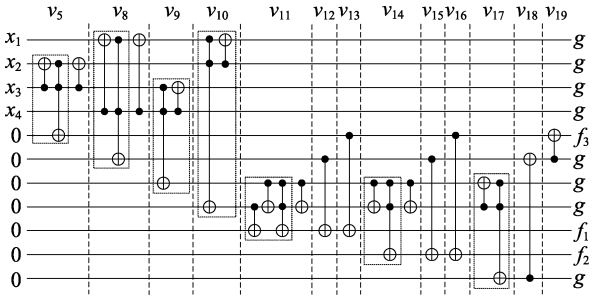

图 6　综合所得电路

此外, 图 6 中电路输入侧的 0 表示该电路线为辅助线, 其输入为常量 0; 输出侧的 g 表示该电路线的输出为垃圾输出. 由于 $q=11$, $m=3$, 可知图 6 所示电路的垃圾输出数为 8.

3.2　电路功能等价性验证

使用 BDD 完成算法 2 综合所得电路与给定布尔函数 $\boldsymbol{F}$ 的功能等价性验证.

先由函数 $\boldsymbol{F}$ 的 ESOP 覆盖 $\boldsymbol{C}$ 构建 BDD, 得到图 $\boldsymbol{G}=\{G_j\,|1\leqslant j\leqslant m\}$; 其中, G_j 为函数输出变量 f_j 的 BDD. 再由综合所得电路根据电路输入和可逆门实现的函数构建 BDD, 不考虑垃圾输出, 得到图 $\boldsymbol{H}=\{H_j\,|1\leqslant j\leqslant m\}$; 其中, H_j 为电路输出 f_j 的 BDD. 若 $\forall j$ ($1\leqslant j\leqslant m$), 有 $G_j\oplus H_j\equiv 0$, 说明综合所得电路实现了函数 $\boldsymbol{F}$ 的功能.

4　实验及结果分析

用于因式分解的 SZMODD 基于 EXTRA 库[21]实现, 算法 2 采用 C 语言实现. 使用 LGSynth 和 RevLib 函数集中不同规模的函数, 在配置为 Intel Core i7-7700 CPU 32 GB RAM 和运行 Ubuntu 16.04 64 位操作系统的计算机上对本文方法进行验证.

与文献[15]类似, 本文先使用 EXORCISM-4 工具[22]利用 q0 参数对函数实施 ESOP 最小化, 得到算法 2 需要的 ESOP 覆盖 $\boldsymbol{C}$.

4.1 与结合变量分组和 BED 的可逆电路综合方法比较

文献[15]先根据 ESOP 覆盖对函数的输入变量分组, 再根据变量分组由 ESOP 覆盖构建布尔函数的 BED 表示, 然后由 BED 综合可逆电路. 分别使用文献[15]方法和本文方法对 32 个不同规模的 LGSynth 和 RevLib 函数进行可逆电路综合, 表 1 给出了结果. 其中, n/m 表示函数的输入数/输出数; 时间为可逆电路综合算法运行所需时间; 改善表示相对于文献[15]方法的结果, 本文方法结果的 NCV 成本和量子位数减少的百分比.

表 1 2 种方法结果比较

函数名	n/m	文献[15]			本文			改善/%	
		NCV 成本	量子位数	时间/s	NCV 成本	量子位数	时间/s	NCV 成本	量子位数
4mod5	4/1	17	7	<0.01	17	7	0.01	0.00	0.00
decod24	2/4	19	6	<0.01	19	6	0.01	0.00	0.00
ex1	5/1	4	5	<0.01	4	5	0.01	0.00	0.00
graycode6	6/6	5	6	<0.01	5	6	0.01	0.00	0.00
peres	3/3	8	4	<0.01	8	4	0.01	0.00	0.00
xor5	5/1	4	5	<0.01	4	5	0.01	0.00	0.00
5mod5	6/6	60	17	<0.01	52	15	0.01	13.33	11.76
aj-e11	4/4	67	16	<0.01	66	15	0.01	1.49	6.25
apex5	117/88	3208	732	0.32	3189	725	0.58	0.59	0.96
bw	5/28	391	65	0.01	385	59	0.01	1.53	9.23
cps	24/109	3068	410	0.22	2856	373	0.12	6.91	9.02
ex5p	8/63	1344	215	0.06	1193	188	0.04	11.24	12.56
ham15	15/15	198	45	0.01	164	35	0.01	17.17	22.22
ham3	3/3	13	6	<0.01	11	5	0.01	15.38	16.67
ham7	7/7	64	18	<0.01	48	13	0.01	25.00	27.78
i6	138/67	738	279	0.22	694	277	0.32	5.96	0.72
i7	199/67	1051	404	0.42	1022	404	0.64	2.76	0.00
i8	133/81	7319	1351	0.46	7125	1311	0.91	2.65	2.96
inc	7/9	313	63	<0.01	287	58	0.01	8.31	7.94
mod10	4/4	51	14	<0.01	44	13	0.01	13.73	7.14
mod5adder	6/6	166	37	<0.01	160	36	0.01	3.61	2.70
sao2	10/4	459	103	0.01	377	84	0.01	17.86	18.45
seq	41/35	4640	687	0.06	4251	592	0.12	8.38	13.83
urf5	9/9	1291	246	<0.01	1255	233	0.01	2.79	5.28
wim	4/7	90	22	<0.01	81	19	0.01	10.00	13.64
dc1	4/7	99	19	<0.01	93	20	0.01	6.06	−5.26
one-two-three	3/3	28	8	<0.01	23	9	0.01	17.86	−12.50
sqr6	6/12	280	58	<0.01	278	59	0.01	0.71	−1.72
ex-1	3/3	18	6	<0.01	21	7	0.01	−16.67	−16.67
majority	5/1	29	11	<0.01	38	13	0.01	−31.03	−18.18
mod5d2	5/5	27	10	<0.01	33	10	0.01	−22.22	0.00
rd32	3/2	14	6	<0.01	23	8	0.01	−64.29	−33.33

由表 1 数据可以看出, 在 32 个函数中, 有 6 个函数, 本文方法可以得到与文献[15]方法具有相同 NCV 成本和量子位数的电路; 有 19 个函数, 从 NCV 成本和量子位数来看, 本文方法可以得到比文献[15]方法更优的电路; 有 3 个函数, 尽管本文方法所得电路的 NCV 成本低于文献[15]方法所得电路, 但量子位数却有一定程度的增加; 对于其余 4 个函数, 本文方法所得 NCV 电路要劣于文献[15]

方法所得NCV电路. 与文献[15]方法相比, 本文方法将可逆电路的 NCV 成本和量子位数分别最大降低了 25.00%和 27.78%(ham7 函数). 从平均角度看, 与文献[15]方法相比, 本文方法将电路的 NCV 成本和量子位数分别降低了 5.01%和 5.47%.

这说明在由 ESOP 覆盖构建 BED 并基于 BED 表示模型综合可逆电路时, 对 ESOP 覆盖实施因式分解后构建 BED, 比对函数输入变量进行分组后构建 BED 更有助于降低由 BED 综合所得电路的成本.

本文方法能适用于大规模函数, 且具有较高的时间效率. 虽然借助 SZMODD 实施 ESOP 立方体的因式分解导致电路综合算法的运行时间有所增加, 但对于表 1 中的大部分函数, 本文方法可在不大于 0.01 s 的时间内完成电路综合. 对于输入数大于 100 的函数, 本文方法也能够在 1 s 内完成电路综合.

4.2　与基于 FDD 的可逆电路综合方法比较

BDD, PDD, FDD, KFDD 和 BBDD 是函数的标准形 DAG 表示. 分析基于此类决策图表示模型的可逆电路综合方法相关文献给出的结果发现, 就结果电路的量子成本和量子位数而言, 这些方法中, 没有哪一种方法具有绝对优势. 由于文献[9]的结果是此类方法中的最新研究结果, 因此选择与文献[9]的基于 FDD 表示模型的可逆电路综合方法(FDD 方法)比较.

表 2 所示为使用本文方法(算法 2)对文献[9]中的 26 个 RevLib 函数进行电路综合的结果. 其中, 零极性 FDD 结果来自文献[9]中的表 4 和表 5; 最优极性 FDD 结果来自文献[9]中的表 7 和表 8. 零极性 FDD 结果和最优极性 FDD 结果是文献[9]根据 FDD 结点相关矩阵分别由零极性 FDD 和最优固定极性 FDD 而得. 在最优极性 FDD 结果中, 有 6 个

表 2　本文与 FDD 方法的结果比较

函数名	*n*/*m*	函数类型	零极性 FDD[9]		最优极性 FDD[9]		本文	
			量子位数	NCV 成本	量子位数	NCV 成本	量子位数	NCV 成本
4mod5	4/1	算术	5	19	5	18	7	17
alu	5/1	ALU	8	28	8	28	11	29
apex2	39/3	一般逻辑	5983	49133			3665	18099
bw	5/28	一般逻辑	78	709	74	619	59	385
cordic	23/2	一般逻辑	44	427			1275	6936
cycle10_2	12/12	一般逻辑	97	552	97	552	52	177
decod24	2/4	编码	6	23	6	23	6	19
e64	65/65	一般逻辑	1498	8495			243	773
ex5p	8/63	一般逻辑	229	1846	225	1803	188	1193
ham15	15/15	编码	42	188			35	164
ham7	7/7	编码	18	74	15	85	13	48
hwb5	5/5	最坏情况	26	233	18	196	41	218
hwb6	6/6	最坏情况	49	470	34	378	70	398
hwb7	7/7	最坏情况	97	965	57	678	178	1071
hwb8	8/8	最坏情况	145	1622	85	1087	401	2469
mini-alu	4/2	ALU	8	43	8	43	14	45
mod5adder	6/6	算术	21	170	20	150	36	160
plus127mod8192	13/13	算术	24	98	24	73	34	125
plus63mod4096	12/12	算术	22	86	22	66	34	124
rd53	5/3	算术	8	44	8	44	28	117
rd73	7/3	算术	10	76	10	76	77	357
rd84	8/4	算术	15	112	15	112	124	588
seq	41/35	算术	1303	12238			592	4251
spla	16/16	一般逻辑	707	5946			742	3942
sym6	6/1	对称	10	69	10	69	29	114
sym9	9/1	对称	12	106	12	106	198	880

函数, 文献[9]没有给出其结果.

由表 2 数据可以看出, 就结果电路的量子成本和量子位数而言, 本文方法和文献[9]中的 FDD 方法都不具有绝对优势. 对于文献[9]给出的 2 种结果, 由最优固定极性 FDD 得到的结果(最优极性 FDD 结果)不一定优于由零极性 FDD 得到的结果(零极性 FDD 结果). 例如, 对于 alu, cycle10_2, rd53 和 sym6 等函数, 最优极性 FDD 结果与零极性 FDD 结果相同; 对于 ham7 函数, 零极性 FDD 结果的 NCV 成本更低, 而最优极性 FDD 结果的量子位数更少. 与文献[9]的 FDD 方法相比, 在这 26 个函数中, 尽管仅有 9 个函数, 本文方法能够得到量子位数和 NCV 成本更低的电路, 另有 2 个函数, 本文方法仅能够得到 NCV 成本更低的电路. 但从平均角度看, 与零极性 FDD 结果和最优极性 FDD 结果中的 NCV 成本最优结果相比, 本文方法将可逆电路的量子位数和 NCV 成本分别降低了 21.11%和 48.32%.

从表 2 第 3 列的函数类型[18]来看, 对于算术逻辑单元(arithmetic logical unit, ALU)、算术、对称和最坏情况函数, 文献[9]的 FDD 方法能够得到更好的结果. 但 seq 函数是例外. 对于 seq 函数, 相对于 FDD 方法, 本文方法能够得到量子位数和 NCV 成本均更低的结果. 对于编码和一般逻辑函数, 本文方法能够得到更好的结果. 但 cordic 和 spla 函数是例外. 对于 cordic 函数, FDD 方法能够得到比本文方法更好的结果. 对于 spla 函数, 与 FDD 方法相比, 本文方法将电路的 NCV 成本降低了 33.70%, 但使量子位数增加了 4.95%.

由 FDD 或 BED 综合所得电路的量子位数和量子成本与 FDD 或 BED 的复杂度有关. FDD 是函数的标准形 DAG 表示, 由对函数实施 Davio 分解而得. BED 是函数的非标准形 DAG 表示, 尽管本文采用了对 ESOP 覆盖实施因式分解后再构建 BED 的方法, 但从基本原理来看, BED 是利用二分解(bi-decomposition)理论对函数实施分解. 对于对称函数、最坏情况函数、ALU 和绝大多数算术函数, 由 FDD 能够得到量子位数和 NCV 成本更低的电路, 这说明宜采用 Davio 分解理论对这些函数实施分解. 而对于编码函数和绝大多数的一般逻辑函数, 由 BED 能够得到量子位数和 NCV 成本更低的电路, 这说明宜利用二分解理论对这些函数实施分解.

4.3 与基于 AIG 和 MIG 的可逆电路综合方法比较

AIG, MIG 与 BED 同属于函数的非标准形 DAG 表示. 文献[14]分别基于 AIG 和 MIG 表示模型综合可逆电路. 使用本文方法(算法 2)对文献[14]表 4 和表 5 中的 22 个 LGSynth 和 RevLib 函数进行可逆电路综合, 表 3 给出了结果. 其中, AIG 方法和 MIG 方法分别表示基于 AIG 和基于 MIG 表示模型的可逆电路综合方法, 其结果来自文献[14]中的表 4 和表 5. 本文方法需要事先获得函数的 ESOP 展开, 但对于 i4 函数, EXORCISM-4 工具[22]无法在合理的时间内获得其 ESOP 最小化的结果, 因此表 3 中本文方法没有给出该函数的结果.

由于文献[14]没有给出表 3 中函数的电路的量子位数, 因此本文仅针对电路的量子成本进行比较. 由表 3 数据可以看出, 与 AIG 方法相比, 除 5 个函数(apex2, cordic, ex4p, i4 和 i8)外, 本文方法均能够获得 NCV 成本和 T 深度更低的电路, 换言之, 本文方法既能够获得量子成本更优的 NCV 电路, 也能够获得量子成本更优的 Clifford+T 电路. 对于 i8 函数, 尽管本文方法增加了 NCV 成本, 却将 T 深度降低了 80.54%, 即本文方法能够获得量子成本更优的 Clifford+T 电路. 对于除 i4 外的 21 个函数, 从平均角度看, 本文方法将 NCV 成本降低了 25.29%, 将 T 深度降低了 45.14%.

由表 3 数据可以看出, 与 MIG 方法相比, 除 7 个函数(apex2, clip, cordic, ex4p, i4, i5 和 majority)外, 本文方法既能获得量子成本更优的 NCV 电路, 也能获得量子成本更优的 Clifford+T 电路. 对于 majority 函数, 尽管本文方法无法改善可逆电路的 Clifford+T 门实现的量子成本, 却将 NCV 门实现的量子成本降低了 5.00%. 对于除 i4 外的 21 个函数, 从平均角度看, 本文方法将 T 深度降低了 25.97%, 但却使 NCV 成本增加了 1.16%.

从表 3 第 3 列的函数类型[17-18]来看, 与文献[14]的 AIG 方法和 MIG 方法相比, 本文方法对算术函数更有优势. 作为函数的 2 级逻辑描述, 在表示算术函数时, ESOP 展开比传统的 2 级“与-或”逻辑更为精简. 由于本文方法是由因式分解后的 ESOP 立方体集合构建 BED, 因此对于算术函数, 相对于 AIG 方法和 MIG 方法, 本文方法能够获得量子成本更低的 NCV 电路和 Clifford+T 电路.

同时, 由于本文方法需要事先获得函数的 ESOP 展开, 因此从方法的可扩展性来看, 本文方法要劣于文献[14]的 AIG 方法和 MIG 方法.

此外, 对于表 1~表 3 中的函数, 本文方法综合所得电路均使用第 3.2 节所述方法进行了功能等价性验证. 电路功能等价性验证方法采用 C++语言实现.

表 3　3 种方法结果比较

函数名	*n*/*m*	函数类型	AIG[14]		MIG[14]		本文		
			NCV 成本	T 深度	NCV 成本	T 深度	量子位数	NCV 成本	T 深度
9sym	9/1	算术	1380	714	1049	678	198	880	567
apex2	39/3	一般逻辑	3750	2250	2540	1329	3665	18099	10878
apex5	117/88	一般逻辑	11972	6453	7888	6072	725	3189	1788
clip	9/5	一般逻辑	1029	525	520	351	144	722	402
cordic	23/2	一般逻辑	485	249	272	174	1275	6936	3756
ex4p	128/28	一般逻辑	4603	2421	2873	1809	1121	4648	2979
frg2	143/139	算术	12012	6342	8137	6243	1500	7840	4017
i4	192/6	一般逻辑	4132	2436	1240	996			
i5	133/66	一般逻辑	6401	3813	772	564	540	2175	1221
i8	133/81	一般逻辑	3310	18129	7340	6852	1311	7125	3528
majority	5/1	算术	50	30	40	24	13	38	24
misex2	25/18	一般逻辑	631	357	476	300	89	303	192
parity	16/1	算术	253	135	360	216	16	15	0
rd53	5/3	算术	327	165	286	189	28	117	66
rd73	7/3	算术	900	456	898	618	77	357	201
rd84	8/4	算术	1356	690	1265	852	124	588	345
seq	41/35	算术	18307	9750	13724	9750	592	4251	1587
spla	16/46	一般逻辑	13685	8211	10996	8790	742	3942	2124
sqn	7/3	算术	695	417	584	408	76	342	201
sqr6	6/12	算术	825	495	593	417	59	278	150
squar5	5/8	算术	320	192	339	225	29	125	66
Z5xp1	7/10	算术	1010	606	568	384	57	266	141

5　结　语

为对 ESOP 立方体进行因式分解, 降低由 ESOP 覆盖构建的 BED 综合所得可逆电路的成本, 本文提出了立方体集合的 SZMODD 表示模型. 先借助 SZMODD 对 ESOP 立方体实施因式分解, 再在此基础上构建 BED. 与在对变量进行分组的基础上由 ESOP 覆盖构建 BED 的方法相比, 从平均角度看, 本文方法将由 BED 综合所得电路的量子成本和量子位数分别降低了 5.01%和 5.47%. 借助代数除法实现 ESOP 立方体的因式分解可很好地处理大规模函数, 且具有较高的时间效率, 这使得本文的可逆电路综合方法能应用于大规模函数.

与基于 FDD 表示模型的可逆电路综合方法相比, 对于编码函数和绝大多数的一般逻辑函数, 本文方法能降低可逆电路的量子成本与量子位数. 与基于 AIG 和基于 MIG 表示模型的可逆电路综合方法相比, 对于算术函数, 本文方法能降低可逆电路的量子成本.

参考文献(References):

[1] Wille R, Drechsler R. Towards a design flow for reversible logic[M]. Heidelberg: Springer, 2010

[2] Abdessaied N, Drechsler R. Reversible and quantum circuits: optimization and complexity analysis[M]. Heidelberg: Springer, 2016

[3] Zulehner A, Wille R. One-pass design of reversible circuits: combining embedding and synthesis for reversible logic[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 37(5): 996-1008

[4] Lin C C, Jha N K. RMDDS: Reed-Muller decision diagram synthesis of reversible logic circuits[J]. ACM Journal on Emerging Technologies in Computing Systems, 2014, 10(2): Article No.14

[5] Wille R, Drechsler R. BDD-based synthesis of reversible logic for large functions[C] //Proceedings of the 46th ACM/IEEE Design Automation Conference. Los Alamitos: IEEE Computer Society Press, 2009: 270-275

[6] Wille R, Drechsler R. Effect of BDD optimization on synthesis of reversible and quantum logic[J]. Electronic Notes in Theoretical Computer Science, 2010, 253(6): 57-70

[7] Krishna M, Chattopadhyay A. Efficient reversible logic synthesis via isomorphic subgraph matching[C] //Proceedings of the

44th IEEE International Symposium on Multiple-Valued Logic. Los Alamitos: IEEE Computer Society Press, 2014: 103-108

[8] Pang Y, Yan Y F, Lin J Z, *et al*. An efficient method to synthesize reversible logic by using positive Davio decision diagrams[J]. Circuits, Systems, and Signal Processing, 2014, 33(10): 3107-3121

[9] Stojković S, Stanković R, Moraga C, *et al*. Reversible circuits synthesis from functional decision diagrams by using node dependency matrices[J]. Journal of Circuits, Systems and Computers, 2020, 29(5): 2050079

[10] Wang Youren, Shen Xiankun, Zhou Yinghui. Synthesis design method of reversible logic circuit based on Kronecker functional decision diagram[J]. Acta Electronic Sinica, 2014, 42(5): 1025-1029(in Chinese)
(王友仁, 沈先坤, 周影辉. 基于KFDD的可逆逻辑电路综合设计方法[J]. 电子学报, 2014, 42(5): 1025-1029)

[11] Chattopadhyay A, Littarru A, Amarú L, *et al*. Reversible logic synthesis via biconditional binary decision diagrams[C] //Proceedings of the IEEE International Symposium on Multiple-Valued Logic. Los Alamitos: IEEE Computer Society Press, 2015: 2-7

[12] Schönborn E, Datta K, Wille R, *et al*. Optimizing DD-based synthesis of reversible circuits using negative control lines[C] //Proceedings of the 17th IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems. Los Alamitos: IEEE Computer Society Press, 2014: 129-134

[13] Law J J, Rice J E. Line reduction in reversible circuits using KFDDs[C] //Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing. Los Alamitos: IEEE Computer Society Press, 2015: 113-118

[14] Bandyopadhyay C, Das R, Chattopadhyay A, *et al*. Design and synthesis of improved reversible circuits using AIG and MIG based graph data-structures[J]. IET Computers & Digital Techniques, 2019, 13(1): 38-48

[15] Bu Dengli, Guo Ming. Reversible circuit synthesis method based on Boolean expression diagram[J]. Acta Electronic Sinica, 2020, 48(3): 494-502(in Chinese)
(卜登立, 郭鸣. 基于布尔表达式图的可逆电路综合方法[J]. 电子学报, 2020, 48(3): 494-502)

[16] Zhang Qiaowen, Wang Pengjun, Hu Jiang. Exact minimization of ESOP expressions based on hierarchical hypercube[J]. Journal of Computer-Aided Design & Computer Graphics, 2016, 28(1): 172-179(in Chinese)
(张巧文, 汪鹏君, 胡江. 基于分层超立方体的精确 ESOP 最小化[J]. 计算机辅助设计与图形学学报, 2016, 28(1): 172-179)

[17] Yang S. Logic synthesis and optimization benchmark user guide: version 3.0[R]. Durham: Microelectronic Center of North Carolina, 1991

[18] Wille R, Große D, Teuber L, *et al*. RevLib: an online resource for reversible functions and reversible circuits[C] //Proceedings of the 38th International Symposium on Multiple-Valued Logic. Los Alamitos: IEEE Computer Society Press, 2008: 220-225

[19] Meter R V, Oskin M. Architectural implications of quantum computing technologies[J]. ACM Journal on Emerging Technologies in Computing Systems, 2006, 2(1): 31-63

[20] Bu Dengli. Reversible circuit synthesis method based on maximum weighted output-compatibility class of ESOP[J]. Acta Electronic Sinica, 2018, 46(8): 1866-1875(in Chinese)
(卜登立. 基于 ESOP 最大加权输出相容类的可逆电路综合方法[J]. 电子学报, 2018, 46(8): 1866-1875)

[21] Sasao T, Butler J T. Applications of zero-suppressed decision diagrams[M]. San Rafael: Morgan & Claypool Publishers, 2014: 1-34

[22] Mishchenko A, Perkowski M. Fast heuristic minimization of exclusive-sums-of-products[OL]. [2020-10-27]. http://people.eecs.berkeley.edu/~alanmi/publications/2001/rm01_heu.pdf