# 基于 XMG 的乘法器电路等价性验证算法

朱柏成, 储著飞*, 潘鸿洋, 王伦耀, 夏银水

(宁波大学信息科学与工程学院 宁波 315211)
(chuzhufei@nbu.edu.cn)

**摘 要:** 组合电路等价性验证是数字集成电路设计自动化(EDA)中的重要部分, 随着算术电路在现代计算机系统中的占比逐渐增大, 传统的等价性验证算法在验证多比特算术电路, 尤其是乘法器电路时面临挑战. 对此, 提出一种基于 XOR-Majority Graph(XMG)逻辑表示的组合电路等价性验证算法. 首先将 2 个待验证电路构建成的联接(Miter)电路进行 XMG 逻辑重写; 然后在等价性一致的前提下对 XMG 的节点个数和逻辑深度进行逻辑重写优化; 最后调用布尔可满足性(SAT)求解器和仿真器进行验证, 得到最终等价性验证结果. 实验结果表明, 与 ABC, Lingeling 等工具相比, 所提算法在验证时间上实现了平均 489 倍、最高 1 472 倍的加速.

**关键词:** 逻辑综合; 等价性验证; 乘法器电路; 异或-多数逻辑图
**中图法分类号:** TP391.41 **DOI:** 10.3724/SP.J.1089.2024.19949

# An XMG Based Equivalence Checking Algorithm for Large Bit-Width Multiplier Circuits

Zhu Baicheng, Chu Zhufei*, Pan Hongyang, Wang Lunyao, and Xia Yinshui

(*Faculty of Electrical Engineering and Computer Science(EECS), Ningbo University, Ningbo* 315211)

**Abstract:** Combinational equivalence checking is an essential part of electronic design automation (EDA). Modern computers have an increasing proportion of arithmetic logic circuits, which poses challenges for traditional equivalence checking algorithms, especially when checking multi-bit logic circuits. An equivalence checking method based on XOR-majority graph (XMG) is proposed to address this issue. Firstly, a miter circuit is constructed using an XMG representation of the referenced design and the design to be verified. Secondly, a functional rewrite is performed on the XMG of the miter, in which the logic level and number of nodes are optimized. Finally, the Boolean satisfiability (SAT) solver and simulator are used to verify the final equivalence verification result. As compared to tools such as ABC and Lingeling, the algorithm performs 489 times CPU time speedup on average, and a maximum acceleration of 1 472 times in validation time.

**Key words:** logic synthesis; equivalence checking; multiplier circuits; XOR-majority graph

电路设计自动化(electronic design automation, EDA)已经高度模块化, 自顶向下的设计流程包括系统级、行为级、逻辑级和物理级等, 验证伴随着 EDA 流程全周期, 包括设计是否满足规范, 不同级别的功能是否等价等. 验证能确保设计产品功能的正确性, 降低流片失败风险, 加速产品面世周

收稿日期: 2022-09-06; 修回日期: 2022-11-07. **基金项目**: 国家自然科学基金(61871242); 专用集成电路与系统国家重点实验室开放课题(2021KF008). **朱柏成**(1997—), 男, 硕士研究生, CCF 学生会员, 主要研究方向为逻辑综合; **储著飞**(1986—), 男, 博士, 副教授, 博士生导师, CCF 高级会员, 论文通信作者, 主要研究方向为集成电路设计自动化; **潘鸿洋**(1997—), 男, 硕士研究生, CCF 学生会员, 主要研究方向为逻辑综合; **王伦耀**(1972—), 男, 博士, 教授, 博士生导师, 主要研究方向为集成电路逻辑综合与优化; **夏银水**(1963—), 男, 博士, 教授, 博士生导师, 主要研究方向为集成电路设计及设计自动化.

期. 考虑集成电路的设计规模日益增大, 高效的验证方法学至关重要.

由于逻辑门级电路描述准确, 常被用于不同层次的等价性验证[1], 因此本文着重考虑组合电路中乘法器电路的等价性验证算法. 给定 2 个待检测的门级电路, 分别为参考设计和优化设计, 门级电路的等价性验证就是判定对于任何可能的输入, 参考设计和优化设计均具有相同的输出.

门级电路的等价性验证算法包括功能性算法[2]和结构性算法[3]. 功能性算法是将待检测的门级电路通过仿真或者构建的方法得到待检测电路的正则性表示, 如仿真出真值表, 构建出电路精简有序二元决策图(reduced ordered binary decision diagram, ROBDD)等. 正则性是指每个布尔函数具有唯一的对应表示, 因真值表和 ROBDD 表示均具有正则性, 只需判定 2 个待检测电路的正则性表示是否相同, 即可判定电路是否等价. 但功能性算法仅适用于小规模电路, 无论是真值表还是 ROBDD, 当电路规模增大后都会出现非线性的内存和时间增长. 结构性算法的原理是找到待检测电路中内部可能的等价节点, 将大规模电路的等价性验证简化为多个小规模电路的验证问题. 常用的方法是基于布尔可满足性(Boolean satisfiability, SAT)求解器的形式化方法来判定节点的等价性.

组合电路进一步可分为控制逻辑电路和算术逻辑电路. 针对控制逻辑电路的验证研究较多, 难度也较低; 算术逻辑电路常涉及多位宽的字级算术运算, 验证时需要通过将字级运算展平成比特级运算, 加大了验证难度. 文献[4]基于代数系统进行算术电路验证, 分别将算术电路设计规范和设计实现建模为多项式代数方程.

算术电路的形式化验证在电路设计中占据重要地位. 由于验证乘法器电路的等价性问题是一个 NP-Complete 电路 SAT 问题, 因此乘法器的等价性验证面临着巨大挑战. 流行的开源逻辑综合工具 ABC(简称 ABC 工具)[5]等基于功能和结构算法的混合算法对 2 个电路进行逻辑等价性检查, 面对大部分组合及时序电路, ABC 等验证工具表现优异, 即使在乘法器电路上, 只要实现电路和参考电路有较大的结构相似性就能快速求解; 然而, 具有不同架构的乘法器电路不能提供足够的结构相似性来进行验证, 这些验证工具对于结构相似性很小的大型复杂算术电路(包括乘法器电路的等价性)无法快速求解[6].

目前, 验证乘法器电路比较成功的算法是用计算机代数技术[7-11]进行多项式推理. 代数方法的难点在于乘法器的末级加法器, 因为一些加法器的设计依赖 OR-gate 序列, 导致中间结果的多项式表示的爆发. Mahzoon 等[7-8]采用收敛锥中消除单项式的局部消去法, 使得验证各种乘法器的架构比已有的工作更有效. 文献[9-10]通过识别乘法器中的全加法器和半加法器来消除冗余多项式.

通常, 组合电路逻辑等价性验证首先构建联接电路(Miter), 即将待检测电路的相应的输出用“异或(exclusive-OR, XOR)”逻辑门分别连接, 检测该 XOR 门的输出是否恒为 0. 若是, 则相应输出端口逻辑等价; 否则, 不等价. 将所有的 XOR 门用“或(OR)”逻辑门连接, 若该 OR 门输出恒为 0, 则整个待检测的电路等价; 否则, 不等价. 可以看出, 一种简单的验证分割算法是依据输出构成的 XOR 门生成的子电路进行分割. 基于 Miter 电路的验证可充分利用逻辑电路综合优化方法, 如通过逻辑化简生成更少节点的等价电路加速验证, 而基于多项式代数方程的验证算法则不能充分利用这个优点.

为了解决乘法器验证难的问题, 本文基于异或-多数逻辑图(XOR-majority graph, XMG)构建 Miter 电路. XMG 是逻辑函数的有向无环图(directed acyclic graph, DAG)表示, 包含 3 输入 XOR 和 majority(MAJ)逻辑门, 因为 3 输入逻辑门的表达能力更强, 所以 XMG 比 2 输入逻辑门的 DAG 更加紧凑. 由于 XOR 门在逻辑重写后会大量存在于 Miter 电路内部, 因此可通过检测 XOR 节点是否恒为常数 0 对 Miter 电路化简. 若 Miter 电路恒为 0, 则整体电路等价; 否则, 可进一步通过 SAT 或者仿真验证. 对 $n(n\leqslant 8)$ 位宽的乘法器电路进行实验的结果表明, 与 ABC 工具中的等价性验证算法相比, 本文提出的等价性验证算法具有运行时间优势; 在相同验证方法中, 本文算法在运行时间和内存上具有优势.

## 1 相关工作

### 1.1 基于 DAG 的逻辑表示

组合电路可以看作多输入/多输出的逻辑函数, 其表示方式有多种, 包括从真值表、卡诺图到积之和(sum-of-products, SOP)表达式等[12]. 从逻辑表示能处理的电路规模来看, 目前主流的大规模组合电路表示方式是基于 DAG 的表示方式, 常见的有与-反相器图(and-inverter graph, AIG), BDD, 多数-

反相器图(majority-inverter graph, MIG)和 XMG 等. 本文重点考虑图形化的逻辑表示方式.

1.1.1　AIG

AIG 是一种同构 DAG, 包括 4 种元素集合: (1) 原始输入(primary input, PI)集合; (2) 原始输出(primary output, PO)集合; (3) 内部计算节点, 2 输入“与(AND)”逻辑门集合; (4) 连接边集合, 包含常规边和补边, 补边表示连接有反相器. 图 1 所示为一个 AIG 示例, PI 集合为 $\{a,b,c,d\}$, PO 集合为 $\{f\}$, 内部计算节点集合为 $\{n_1,n_2,n_3\}$, 连接边集合为 $\{(a,n_1),\cdots,(n_3,f)\}$. 通过自 PI~PO 逐级计算, 可得到内部节点和 PI 的逻辑表达式, 例如, $n_2=cd$, $f=n_3=a\bar{b}\bar{c}\bar{d}$.

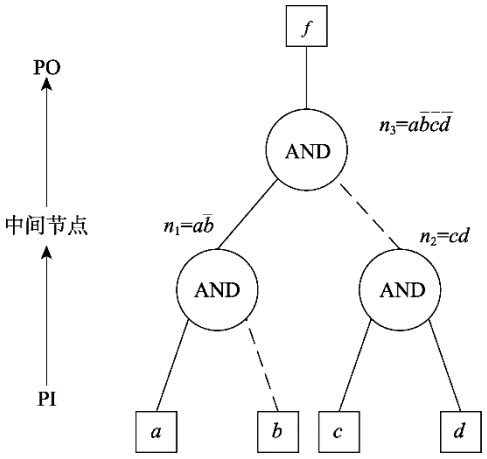


图 1　AIG 示例

由于 AIG 包含 AND 逻辑门和反相器, 因此它是逻辑完备的. ABC 工具包含完整的基于 AIG 的逻辑综合优化算法. AIGER[13]是一种 AIG 的文件格式, 能够高效地表示 AIG. 但是, AIG 不是正则性表达方式, 同一个电路有很多种不同的 AIG 表示.

1.1.2　BDD

BDD 可根据香农分解得到, 给定 $n$ 输入布尔函数 $f(x_1,x_2,\cdots,x_n)$, 对其中一个输入 $i$, $1\leqslant i\leqslant n$, 定义

$$f_{x_i}=f(x_1,x_2,\cdots,x_{i-1},x_i=1,x_{i+1},x_{i+2},\cdots,x_n),$$

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

则香农展开式为

$$f(x_1,x_2,\cdots,x_n)=x_i f_{x_i}+\bar{x}_i f_{\bar{x}_i}.$$

例如, 函数

$$f(x_1,x_2,x_3)=x_1+\bar{x}_2x_3,$$

$$f_{x_1}=f(x_1=1,x_2,x_3)=1,$$

$$f_{\bar{x}_1}=f(x_1=0,x_2,x_3)=\bar{x}_2x_3,$$

依次对所有变量进行香农分解, 即可得到 BDD.

图 2a 所示为 $f(x_1,x_2,x_3)=x_1+\bar{x}_2x_3$ 的基本 BDD, 图中虚线代表 $x_1=0$, 实线代表 $x_1=1$, 自顶点向下依次经过变量 $x_1$, $x_2$ 和 $x_3$, 即可得到不同输入组合下的输出. 可以看出, 图 2a 的规模随着变量数增多呈现指数级增长, 本质上和真值表相同. 通过对同构节点进行合并、消去等操作, 得到的 ROBDD 如图 2b 所示, 显然比图 2a 简洁. 1986 年, 文献[2]证明了每个布尔函数均有唯一的 ROBDD 与之对应, 即具有正则性. 因此, 判定 2 个逻辑电路是否等价, 只需要构建出其 ROBDD, 若 ROBDD 相同, 则等价.

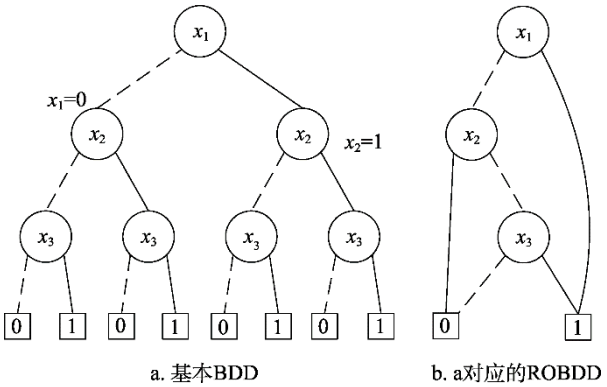


图 2　BDD 示例

1.1.3　XMG

近年来, 3 输入多数逻辑门 MAJ 受到广泛关注, 原因是很多纳米电路能以较低的成本实现[14], 因此基于 MAJ 的逻辑综合得到进一步发展. 给定 3 个布尔变量 $x,y,z$, 3 输入的 MAJ 逻辑门记为 $M_3$, $M_3(x,y,z)=xy+xz+yz$, 设置任意一个输入变量为常量 0 或 1, 分别得到 AND 和 OR 逻辑门, 即可以得到公式 $M_3(x,y,z=0)=xy$, $M_3(x,y,z=1)=x+y$. 因此, AIG 通过“一对一”映射为 MIG. AIG 和 MIG 均是同构 DAG, 其中, AIG 和 MIG 在表示 XOR 逻辑时均需要较高代价, 如至少需要 3 个 $M_3$ 逻辑门才能表示 2 输入 XOR 逻辑门运算. XMG 则是 MIG 的拓展, 内部计算节点既可以是 3 输入 MAJ, 也可以是 3 输入 XOR.

从图 3 中可以看出, XMG 比 MIG 具有更少的节点数和具有更少的层级数. XMG 已被用于现场可编程门阵列(field programmable gate array, FPGA)电路、量子电路综合、精确综合和 QCA 电

路设计等[15]. 值得指出的是, XMG 兼容常见的二元 AND/OR/“非(inverter, INV)”/XOR 逻辑运算, 如上所述, MAJ 逻辑门的 1 个输入为常数时, 可得到 AND/OR 逻辑门, 而 3 输入 XOR 其中 1 个输入为常数时, 亦可表示 3 输入 XOR 和“同或(XNOR)”操作.

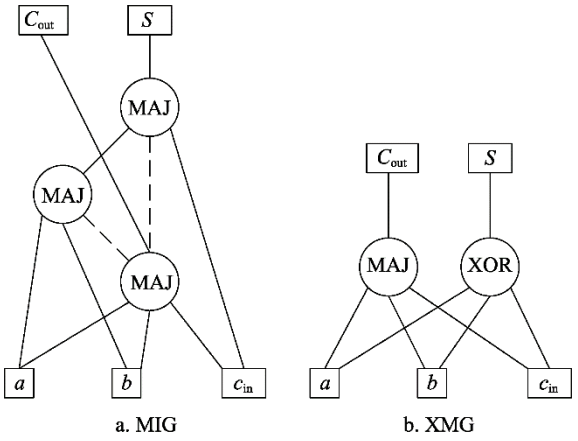


图 3 2 种全加器表示

## 1.2 等价性验证算法

组合电路的等价性验证常构建 Miter 电路, 如图 4 所示, 给定 2 个待检测的组合电路 $C_1$ 和 $C_2$, 首先要确保这 2 个电路具有相同的原始输入 $\{x_1, x_2, \cdots, x_n\}$ 和原始输出 $\{y^1, y^2, \cdots, y^m\}$, 否则直接判定 $C_1$ 和 $C_2$ 不等价. 对于 $C_1$ 和 $C_2$ 对应的输出, 用一个 XOR 门连接, 即 $y_1^i \oplus y_2^i$, $i \in [1, m]$. 若待检测的电路等价, 则每个 XOR 门的输出必然为 0, 将所有的 XOR 作为 $m$ 输入的 OR 门输入, 只需检测该 OR 门的输出为 0.

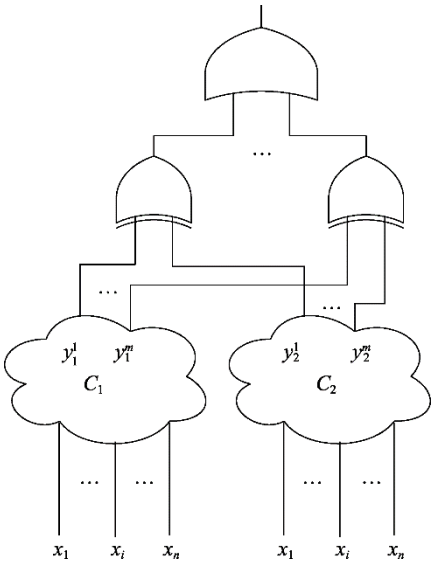


图 4 Miter 电路示例

### 1.2.1 基于 BDD 的等价性验证

若能证明 Miter 电路的输出恒为 0, 则待检测的电路等价. 因此, 对于中小规模电路, 通过仿真即可得出结果; 或者基于正则性表达, 即求解 Miter 电路的 BDD, 若该 BDD 恒为 0 即可. 但是, BDD 的规模取决于变量的排序, 为了在构建 BDD 时不引起内存爆炸, 在构建 BDD 之前就需要找到一个合适的排序, 在构建 BDD 的过程中还需要使用启发式算法对变量排序进行调整. 此外, 一些电路(如乘法器)无论如何进行变量排序, BDD 的规模始终较大. 因此, BDD 只适用于中小规模电路的验证, 且对算术电路的验证存在一定缺陷.

### 1.2.2 基于 SAT 的等价性验证

SAT 是一个有广泛实际应用的经典问题. 通过推理, 判定是否存在一组或多组变量的取值组合, 能够使给定问题的布尔公式取值为真. 如果存在使布尔公式取值为真的变量取值组合, 那么称该布尔公式是可满足的(SAT); 否则, 该布尔公式是不可满足的(UNSAT).

通常以合取范式(conjunctive normal form, CNF)表示 SAT 求解的布尔公式. 因此, 为了解决逻辑电路的 SAT 问题, 需要通过 Tesitin 编码将电路的门级网表转换为相应的 CNF. 如一个 AND 门

$$C = A \cdot B \tag{1}$$

通过 Tesitin 编码生成 CNF

$$(\overline{A} \vee \overline{B} \vee C) \wedge (A \vee \overline{C}) \wedge (B \vee \overline{C}) \tag{2}$$

式(1)与式(2)是等价的, 若式(2)解得结果为真, 则 $A$, $B$, $C$ 的取值一定符合式(1)的规则.

SAT 被广泛应用于等价性检验工具中. 为了通过 SAT 求解器检验函数 $f$ 和函数 $g$ 之间的等价性, 需要生成一个新的函数 $h = f \otimes g$, 并将其转换为一组 CNF 子句. $f$ 和 $g$ 的等价性检验问题被简化为 $h$ 的一个 NP-Complete 电路 SAT 问题. 如果 $h$ 是可满足的, 那么得到 $f$ 和 $g$ 不等价. 实际上, 可满足的情况是一种更简单的情况, 因为 SAT 求解器可以在找到可满足的输入赋值时返回(通常非常快). 然而, 具有挑战性的情况是当 $f$ 和 $g$ 是等价的, $h$ 是不能满足的, 这可能是一个最坏情况, 任何 SAT 求解器需要检查所有可能的输入赋值, 这在本质上是指数型的. 其原因如下: 虽然在已知的 SAT 求解器中决策和学习过程已被证明是相当有效的, 但是 SAT 求解器高度依赖于比较逻辑的内部等价点的成功定位. 即使对于内部等效点很少的非常小的电路, 在最坏的情况下, 求解时间可

能会呈指数增长．例如，比较不同结构设计的乘法器，如 non-Booth Wallace 和 Booth 就是最糟糕的例子之一[16]．表 1 所示为用 ABC 工具和开源 SAT 求解器 Lingeling[17]以及本文来验证 $a(bc)$ 和 $(ab)c$ 这 2 个乘法器的等价性的实验数据，这 2 种求解器都需要指数级的时间来比较这些乘法器．类似地，大多数已知的逻辑综合工具在验证高位宽的乘法器时可能会终止，如 ABC 工具中的&cec．当前常用的 SAT 技术有 Look-Ahead[18]、冲突驱动子句学习[19]、增量式 SAT 求解[20-21]等．

**表 1　不同算法对 *a*(*bc*)和(*ab*)*c* 电路的等价性验证**

| 位宽 | PI/PO | ABC 工具的 cec[22] | | ABC 工具的&cec[23] | | Lingeling[17] | | 本文 | |
|---|---|---|---|---|---|---|---|---|---|
| | | 时间/s | 内存/MB | 时间/s | 内存/MB | 时间/s | 内存/MB | 时间/s | 内存/MB |
| 4 | 12/16 | 1.95 | 11.8 | 0.77 | 7.4 | 0.03 | 2.2 | 0.11 | 18.3 |
| 5 | 15/20 | 31.28 | 15.5 | 18.49 | 10.8 | 8.30 | 4.7 | 0.14 | 19.3 |
| 6 | 18/24 | 168.60 | 18.7 | 797.40 | 23.4 | 87.90 | 14.0 | 0.31 | 28.8 |
| 7 | 21/28 | 5924.86 | 48.0 | 6107.76 | 47.4 | 1580.20 | 42.3 | 1.42 | 331.5 |
| 8 | 24/32 | >1 d | | | | >1 d | | 18.64 | 3263.1 |

## 2　本文算法

### 2.1　基本思路

在构建 Miter 电路后，传统算法将其看成一个整体，用 SAT，BDD 或者仿真求解，常因问题规模增大而失效．通过结构相似性分析或者随机模拟能找到内部可能等价的节点，证明内部节点等价性对 Miter 电路进行简化，能够加速等价性验证．本文提出基于 XMG 的组合电路等价性验证算法，通过对 Miter 电路进行逻辑重写使 Miter 电路的内部含有 XOR 节点，若能证明 XOR 节点及其输入的逻辑子电路等价，则直接用常量 0 替代，或者通过 XMG 的一般代数和布尔方法进行优化，从而大规模简化 Miter 电路．若能将 Miter 电路优化为 0 个节点，则表明 Miter 电路恒等于 0，待检测电路等价；或者将 Miter 电路简化后再结合 SAT 求解和仿真，亦能降低问题复杂度．

图 5 所示为基本思路的示意图．给定任意 DAG 描述的组合电路 $C_1$ 和 $C_2$，以其中 1 个输出的 Miter 电路为例，通过逻辑重写得到 XMG 电路中 XOR 节点遍布内部，此时只需遍历所有 XOR 节点，并对其输入的 2 个子电路进行等价性检查．若等价，则将该 XOR 节点置为 0，并更新 Miter 电路；如此循环，直至电路被优化为 0 个节点；或者 Miter 电路优化后的规模可以被 SAT 或者仿真直接验证．

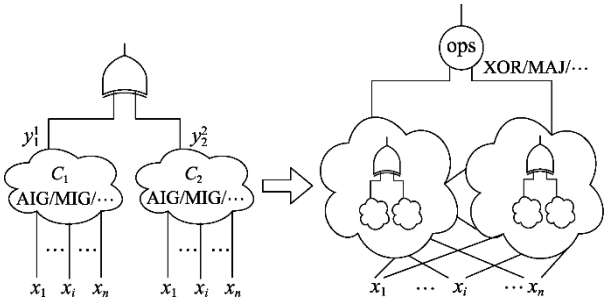


图 5　将构建的 Miter 电路逻辑重写成 XMG

### 2.2　算法步骤

本文算法步骤如下：

输入．任意 2 个 DAG 表示的组合电路 $C_1$ 和 $C_2$．

输出．XMG 逻辑网络．

Step1．将 $C_1$ 和 $C_2$ 构建成基于 XMG 的 Miter 电路．

Step2．对该 Miter 电路进行 XMG 逻辑重写．

Step3．在当前 XMG 网络依据拓扑排序找出所有的 XOR 节点集合 $N$．

Step4．遍历 $N$ 中所有的节点 $n$，检查其输入的 2 个子电路是否等价．若等价，则纪录该节点，且用常数 0 替代．

Step5．若存在可用常用 0 替代的 XOR 节点，则逐个替代并更新 Miter 电路．

Step6．若 Miter 电路包含 0 个节点，则证明整体电路是等价的，退出程序；否则，转 Step2，继续 XMG 逻辑重写寻求进一步优化机会．当遍历所有 XOR 节点后，退出程序，返回当前 XMG.

### 2.3　XMG 逻辑重写

逻辑重写包括函数式重写和代数式重写．前者通过布尔代数或者已知的最优结构对小规模电路，如 4 输入逻辑函数，进行等价替换；后者则利用一般代数，如分配律、结合律，进行变换．

例如，给定一个十六进制表示的真值表 0x17E8，已知其采用 MAJ 和 XOR 逻辑表示的最优表达式为 $M_3(a,b,c)\oplus d$，则任意电路中只要其真值表一致，均可用该表达式进行等价替换，即函数式重写．而代数等式 $ab+ac=a(b+c)$ 执行等价变换自左向右或自右向左均不改变逻辑功能，基于一般代数等式的变换被称为代数式重写．

函数式重写适用于大规模电路的优化以及不同 DAG 之间的变换. 例如, 通过函数式重写可将 AIG 重写为 XMG, 一旦获得初始 XMG, 则继续利用代数式重写和启发式算法进一步优化 XMG 的节点个数和逻辑深度.

给定任意 DAG 表示组合电路, 均可以将其逻辑重写成 XMG. 重写基于逻辑函数的等价性替换, 一个组合电路 $C$ 可看作若干 $k$ 输入单输出逻辑函数的组合. 对于每个 $k$ 输入函数, 现有的方法是通过精确综合的方法计算出 4 输入逻辑函数的 XMG 最优表达式(如节点数最少), 这些最优表达式构成库, 重写时直接查询库中的函数替换即可.

图 6 所示为 XMG 逻辑重写示例. 给定任意的组合电路, 通过映射(如 FPGA 中的查找表映射)将其表示成 $k=3$ 输入单输出函数组合, 通过查询 XMG 的逻辑表达库, 即可等价替换得到整体 XMG.

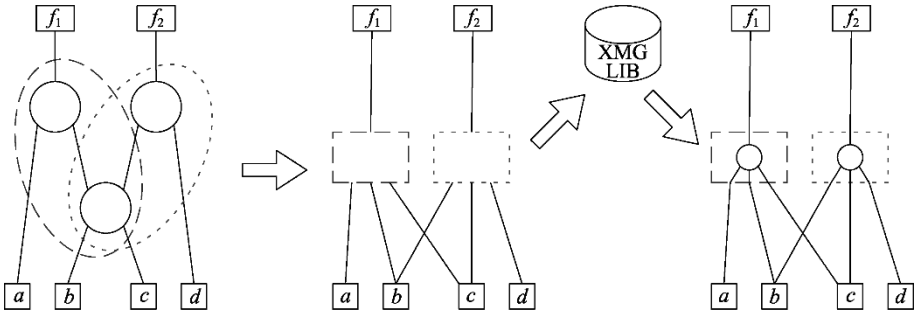


图 6 XMG 逻辑重写

得到初始 XMG 后, 可进一步基于 XOR 和 MAJ 的代数恒等式进行结构重写[24], 降低 XMG 的节点个数、逻辑层级以及反相器个数等. 通过逻辑重写可为等价性验证带来 2 个方面的优化机会: (1) 更多的 XOR 内部节点可看作一个子 Miter 电路, 通过对这些小规模 Miter 电路判定, 可以加速整体验证速度, 且一旦这些子 Miter 电路等价, 原 Miter 电路中的 XOR 节点可用常数 0 替代, 达到优化 Miter 电路的目的; (2) XMG 优化后具有更加紧凑的逻辑表达形式, 有助于基于 SAT 和仿真进一步处理.

### 2.4 XMG 常量代入

通过对 XOR 节点的 2 个输入电路进行等价性验证, 根据返回结果是等价还是不等价的来决定是否对内部 XOR 节点进行快速迭代和替换.

首先获取 XOR 节点两端的子节点, 将 2 个子节点构建 Miter 电路的输出转换成布尔表达式, 将布尔表达式通过 Tesitin 变换转换成 CNF 公式. 然后将 CNF 交给 SAT 求解器求解, 若求解结果是 UNSAT, 则 2 个子节点是等价的, 用常数 0 将 XOR 节点替换, 并清除下方的悬空节点; 若求解结果是 SAT, 则 2 个子节点是不等价的, 根据电路的拓扑结构跳转到下一个 XOR 节点.

以图 7a 中的 XMG 为例, 3 输入 XOR 节点有常数输入 0, 实际上表示的是 2 输入 XOR, 通过判定得到

$$f_1 = M_3\left(M_3(a,b,0),c,0\right) = (ab)c,$$

$$f_2 = M_3\left(M_3(b,c,0),a,0\right) = (bc)a.$$

因此 $f_1 = f_2$, 则 XOR 节点用常数 0 替代, 得到如图 7b 所示简化的 XMG. 需要指出的是, 如果 2 个待检测电路构建的 Miter 电路可以优化为 0 个节点的 XMG, 那么待检测电路必然是等价的. 因此, 若判定出足够的常量代入后 XMG 的节点数为 0, 则表明 2 个待检测电路等价.

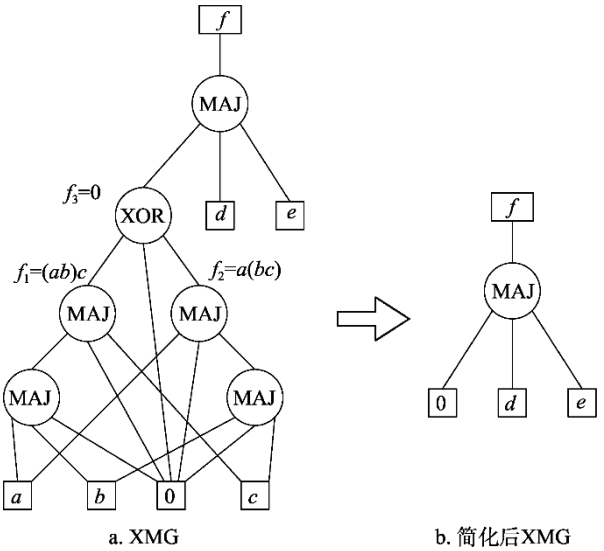


图 7 XMG 常量判定与代入示例

## 3 实验及结果分析

本文实验在 Intel(R) Xeon(R) Silver 4210R @ 2.4GHz CPU 的服务器上进行, 所用到的算法命令

已在开源逻辑综合工具 ALSO 上用 C++实现, 其源代码是公开的. 其中, cremiter 命令用于构建 Miter 电路, convert --aig_to_xmg 命令用于将已生成的 Miter 电路转换成 XMG, xmgrw -u 命令实现了 XOR 节点的常数替换, sim 命令实现对逻辑网络的仿真.

实验在具有任意位宽的简单乘法器电路的测试用例上进行. (1) 验证第 1 节提出的 XMG 逻辑重写等价性验证算法的高效性. 分别对 3 输入乘法器和 3 输入乘加算术电路进行对比实验, 比较其用时和内存消耗, 结果如表 1 和表 2 所示; (2) 为了分析电路中 XOR 节点的替换对电路等价性验证效率的影响, 分析了多种结构的待验证 Miter 电路进行 XOR 节点替换效果, 以用时和节点替换数量作为比较.

**表 2　不同算法对$(a+b)c$ 和 $ac+bc$ 电路的等价性验证**

| 位宽 | PI/PO | ABC 工具的 cec[22] | | ABC 工具的&cec[23] | | Lingeling[17] | | 本文 | |
|---|---|---|---|---|---|---|---|---|---|
| | | 时间/s | 内存/MB | 时间/s | 内存/MB | 时间/s | 内存/MB | 时间/s | 内存/MB |
| 4 | 12/16 | 0.62 | 9.5 | 0.41 | 7.4 | 0.02 | 1.7 | 0.01 | 17.7 |
| 5 | 15/20 | 9.53 | 13.4 | 5.65 | 8.8 | 3.60 | 3.8 | 0.13 | 19.1 |
| 6 | 18/24 | 65.71 | 16.4 | 167.31 | 15.5 | 40.40 | 9.9 | 0.21 | 33.9 |
| 7 | 21/28 | 931.27 | 31.9 | 1197.97 | 28.3 | 473.50 | 25.6 | 0.92 | 211.9 |
| 8 | 24/32 | 37578.20 | 100.0 | | | 8025.40 | 82.9 | 8.48 | 2091.3 |

本文实验对比对象是工业级 ABC 工具和知名的 SAT 求解器 Lingeling. 其中, ABC 工具的 cec[22] 命令是对由 2 个 AIG 电路构建的 Miter 电路进行优化, 再用 ABC 工具内置 SAT 求解器求解 CNF 进行等价性验证; ABC 工具的&cec[23]命令则是在 cec 命令的基础上, 增加 BDD sweeping 和 SAT sweeping 方法, 以验证 2 个待验证电路的等价性; SAT 求解器 Lingeling 以由 2 个 AIG 电路构建的 Miter 电路生成的 CNF 作为输入, 通过 SAT 求解得出等价与否的结论. 本文实验通过 ABC 工具分别读取 4 个功能正确的 Verilog 文件, 再使用 write_aiger 命令得到输入 AIG, 其中, SAT 求解器的输入文件 CNF 由 ABC 工具的 Miter 命令得到.

### 3.1　与 ABC 工具等价性验证算法对比

分别对比 4~8 位宽算术电路的等价性验证效率, 以内存消耗和算法运行时间作为对比标准. 其中, cec 命令求解时间>1 d 就停止求解. 本文算法的运行时间包括读取 2 个电路、生成 Miter 电路和进行等价性验证的时间. ABC 工具和本文算法内存包含读入 AIG 文件、构建 Miter 结构、XMG 逻辑重写、SAT 求解等内存消耗, Lingeling 的内存占用只包含对 Miter 电路 SAT 求解的消耗.

(1) 验证乘法器 $a(bc)$ 和 $(ab)c$ 的等价性. 在不同位宽下, 算法运行时间和内存的结果如表 1 所示. 可以看出, 在乘法器的为 4~7 位宽情况下, 对比 ABC 工具的 cec 命令, 本文算法可以实现 3 个数量级运行时间加速比提升, 对比 Lingeling 求解器, 本文算法可以实现平均 846 倍运行时间加速比; 在 8 位宽情况下, 由于电路规模过大, ABC 工具和 Lingeling 无法求解, 而本文算法仍能在相对较短的时间内计算出结果; 在内存消耗部分, 本文算法消耗一定的内存空间来加速验证的速度, 内存的增长在可接受范围之内.

(2) 验证加法乘法 $(a+b)c$ 和 $ac+bc$ 电路的等价性. 从表 2 可以看出, 在 4~7 位宽的情况下, 本文算法可以实现平均 792 倍、最高 1011 倍的运行时间加速比; 在 8 位宽情况下, 本文算法与表 1 的实验结果类似, 可以在相对较短的时间内计算出 ABC 工具和 Lingeling 无法计算出的结果; 本文算法的内存消耗情况与表 1 相同, 内存的增长在可接受范围之内.

从表 1 和表 2 可以得出结论: ABC, Lingeling 等验证工具对于乘法器电路的验证存在明显不足, 无法在规定时间内验证 8 位宽的 3 输入乘法器或加法乘法器; 而本文算法可以快速验证具有高位宽电路的等价性; 在内存消耗方面, 本文算法虽然消耗了更多的内存空间, 但是可以快速验证出结果, 内存消耗增长在可接受范围之内.

### 3.2　不同 DAG 结构结果对比

(1) 不同 DAG 结构的仿真验证结果. 2 个待检测组合电路 $C_1$ 和 $C_2$ 的结构对 Miter 电路中 XOR 节点替换效果会产生影响. 表 3 中, 优化前仿真指 Miter 两端的待检测电路都是 AIG, 将生成的 Miter 电路(AIG 形式)进行仿真的结果; 优化后仿真指 Miter 两端的待检测电路分别是 XMG 和 XMG, 将生成的 Miter 电路(XMG 形式)经过常量节点替换算法后进行仿真的结果.

(2) 不同 DAG 结构的 SAT 验证结果. 表 4 中,

优化前 SAT 求解指 SAT 求解将生成的 Miter 电路(AIG)直接输出为 CNF 格式电路交给 SAT 求解器求解的结果; 优化后 SAT 求解指将待检测 AIG 通过 ALSO 的 convert --aig_to_xmg 命令转换成 XMG, 再经过 ALSO 的 xmgrw -u 命令, 使用常量节点替换算法优化后的 Miter 电路输出 CNF 交给 SAT 求解器求解的结果.

**表 3 对于$(a+b)c$ 和 $ac+bc$ 电路的仿真验证时间及内存对比**

| 位宽 | PI/PO | 优化前仿真 | | 优化后仿真 | |
|---|---|---|---|---|---|
| | | 时间/s | 内存/MB | 时间/s | 内存/MB |
| 4 | 12/10 | 0.05 | 0.1 | 0.01 | 17.7 |
| 5 | 15/12 | 0.06 | 0.1 | 0.13 | 19.1 |
| 6 | 18/14 | 0.17 | 29.2 | 0.21 | 33.9 |
| 7 | 21/16 | 1.44 | 381.6 | 0.92 | 211.9 |
| 8 | 24/18 | 18.39 | 2930.7 | 8.48 | 2091.3 |
| 合计 | | 20.11 | 3341.7 | 9.75 | 2373.9 |

**表 4 对于$(a+b)c$ 和 $ac+bc$ 电路的 SAT 验证时间及内存对比**

| 位宽 | PI/PO | 优化前 SAT 求解 | | 优化后 SAT 求解 | |
|---|---|---|---|---|---|
| | | 时间/s | 内存/MB | 时间/s | 内存/MB |
| 4 | 12/16 | 0.09 | 3 | 0.04 | 3 |
| 5 | 15/20 | 0.71 | 4 | 0.66 | 4 |
| 6 | 18/24 | 9.83 | 7 | 9.79 | 6 |
| 7 | 21/28 | 126.85 | 12 | 112.47 | 11 |
| 8 | 24/32 | 4585.41 | 27 | 2261.89 | 27 |
| 合计 | | 4722.89 | 53 | 2384.85 | 51 |

从表 3 和表 4 可以看出, 经过常量节点替换算法优化后的电路, 仿真和 SAT 求解的速度和内存都有较大的提升, 证明本文算法实现了在电路结构内部 XOR 节点到常数 0 的替换, 提供了更多子节点等价性验证的机会, 实现了加速验证的效果; 不同的 DAG 结构对于求解速度和内存消耗都有较大的影响, 而经过节点常量替换算法优化后的 Miter 电路具有更紧凑更高效的结构, 能够加速等价性验证的速度, 降低等价性验证消耗的空间.

## 4 结 语

本文提出乘法器电路的等价性验证算法, 该算法基于 XMG 表示 Miter 电路, 并通过逻辑重写、常量替换对 XMG 电路优化, 减少了内存消耗, 达到加速电路验证的速度的目的. 实验结果表明, 本文算法能够快速地验证多比特乘法器的等价性; 与工业级 ABC 工具和 SAT 求解器 Lingeling 相比, 该算法能够验证高达 8 位宽的乘法器算术电路, 并且提高了验证效率. 本文算法不仅适用于乘法器算术电路, 而且适用于组合电路, 未来可能用到更大型的组合电路或算术电路中.

## 参考文献(References):


[1] Yang Jun. A study of logic equivalence verification of an integrated circuit[D]. Zhejiang: Zhejiang University, 2007(in Chinese)
(杨军. 集成电路的逻辑等价性验证研究[D]. 杭州: 浙江大学, 2007)

[2] Bryant R E. Graph-based algorithms for Boolean function manipulation[J]. IEEE Transactions on Computers, 1986, C-35(8): 677-691

[3] Berman C L, Trevillyan L H. Functional comparison of logic designs for VLSI circuits[C] //Proceedings of the IEEE International Conference on Computer-Aided Design. Los Alamitos: IEEE Computer Society Press, 1989: 456-459

[4] Lv J P, Kalla P, Enescu F. Efficient Gröbner basis reductions for formal verification of Galois field arithmetic circuits[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2013, 32(9): 1409-1420

[5] Brayton R, Mishchenko A. ABC: an academic industrial-strength verification tool[C] //Proceedings of the 22nd International Conference on Computer Aided Verification. New York: ACM Press, 2010: 24-40

[6] Chen J C, Chen Y A. Equivalence checking of integer multipliers[C] //Proceedings of the Asia and South Pacific Design Automation Conference. New York: ACM Press, 2001: 169- 174

[7] Mahzoon A, Große D, Drechsler R. RevSCA: using reverse engineering to bring light into backward rewriting for big and dirty multipliers[C] //Proceedings of the 56th ACM/IEEE Design Automation Conference. Los Alamitos: IEEE Computer Society Press, 2019: 1-6

[8] Mahzoon A, Große D, Drechsler R. PolyCleaner: clean your polynomials before backward rewriting to verify million-gate multipliers[C] //Proceedings of the IEEE/ACM International Conference on Computer-Aided Design. Los Alamitos: IEEE Computer Society Press, 2018: 1-8

[9] Ciesielski M, Su T K, Yasin A, *et al*. Understanding algebraic rewriting for arithmetic circuit verification: a bit-flow model[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39(6): 1346-1357

[10] Yu C X, Ciesielski M, Mishchenko A, *et al*. Fast algebraic rewriting based on and-inverter graphs[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 37(9): 1907-1911

[11] Kaufmann D, Biere A, Kauers M. Incremental column-wise verification of arithmetic circuits using computer algebra[J]. Formal Methods in System Design, 2020, 56(1): 22-54

[12] Chu Zhufei, Wang Lunyao, Xia Yinshui. Research progress of logic synthesis based on multi-logic domain[J]. Micro/Nano

Electronics and Intelligent Manufacturing, 2021, 3(2): 64-73(in Chinese)
(储著飞，王伦耀，夏银水. 基于多逻辑域的逻辑综合研究进展[J]. 微纳电子与智能制造, 2021, 3(2): 64-73)

[13] Brayton R, Mishchenko A. ABC: a system for sequential synthesis and verification[OL]. [2022-09-06]. http://people.eecs.berkeley.edu/~alanmi/abc/abc.htm

[14] Amarú L, Gaillardon P E, Mitra S, *et al*. New logic synthesis as nanotechnology enabler[J]. Proceedings of the IEEE, 2015, 103(11): 2168-2195

[15] Chu Z F, Soeken M, Xia Y S, *et al*. Advanced functional decomposition using majority and its applications[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39(8): 1621-1634

[16] Diao Y, Wei X, Lam T K, *et al*. Coupling reverse engineering and SAT to tackle NP-complete arithmetic circuitry verification in∼ O (# of gates)[C] //Proceedings of the 21st Asia and South Pacific Design Automation Conference. Los Alamitos: IEEE Computer Society Press, 2016: 139-146

[17] Biere A. Lingeling essentials, A tutorial on design and implementation aspects of the the SAT solver Lingeling[C] //Proceedings of the 5th Pragmatics of SAT Workshop. Vienna: Easychair, 2014: 88

[18] Heule M, van Maaren H. Look-ahead based SAT solvers[M] //Frontiers in Artificial Intelligence and Applications. Holland: IOS Press, 2009: 155-184

[19] Marques-Silva J, Lynce I, Malik S. Conflict-driven clause learning SAT solvers[M] //Frontiers in Artificial Intelligence and Applications. Holland: IOS Press, 2021: 133-182

[20] Eén N, Sörensson N. An extensible SAT-solver[C] //Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing. Heidelberg: Springer, 2004: 502-518

[21] Eén N, Sörensson N. Temporal induction by incremental SAT solving[J]. Electronic Notes in Theoretical Computer Science, 2003, 89(4): 543-560

[22] Mishchenko A, Chatterjee S, Brayton R, *et al*. Improvements to combinational equivalence checking[C] //Proceedings of the IEEE/ACM International Conference on Computer Aided Design. Los Alamitos: IEEE Computer Society Press, 2006: 836-843

[23] Possani V N, Mishchenko A, Ribas R P, *et al*. Parallel combinational equivalence checking[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39(10): 3081-3092

[24] Chu Z F, Soeken M, Xia Y S, *et al*. Structural rewriting in XOR-majority graphs[C] //Proceedings of the 24th Asia and South Pacific Design Automation Conference. New York: ACM Press, 2019: 663-668