Advances in Applied Mathematics 应用数学进展, 2022, 11(11), 8265-8273
Published Online November 2022 in Hans. http://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2022.1111874

Hans汉斯

基于散列表的加法器重写优化算法

黄怡桐，刘　帅，魏峰玉，江建国*

辽宁师范大学数学学院，辽宁 大连

收稿日期：2022年10月23日；录用日期：2022年11月18日；发布日期：2022年11月29日

摘　要

在大规模算术集成电路设计领域中，乘法器电路的验证是一重大难题。当前主流的方法是Gröbner基法。在此基础上，研究者们提出了加法器重写等优化方法，但重写过程依赖进位变量且需遍历全部变量，导致验证效率降低。为此，本文利用散列表对该方法进行优化，使用散列表存储所有和位输出变量，遍历表中元素进行扩充，再结合加法器的结构特性去识别加法器，从而不再依赖进位变量并且减少了需要遍历的变量数目。由实验结果可知，优化算法提高了Gröbner基的生成效率，也提高了乘法器的验证效率。

关键词

乘法器电路验证，Gröbner基，加法器重写方法，散列表

Hash Table Based Adder Rewriting Optimization Algorithm

Yitong Huang, Shuai Liu, Fengyu Wei, Jianguo Jiang*

School of Mathematics, Liaoning Normal University, Dalian Liaoning

Received: Oct. 23rd, 2022; accepted: Nov. 18th, 2022; published: Nov. 29th, 2022

Abstract

In the field of very large scale integration design, the verification of multiplier circuits is a major problem. The current mainstream method is the Gröbner basic method. On this basis, researchers have proposed optimization methods such as adder rewriting, but the rewriting process relies on carrying variables and needs to traverse all variables, resulting in reduced verification efficiency. For this reason, this paper optimizes the method by using a hash table, uses a hash table to store all the sum bit output variables, expands the elements in the traversal table, and then identifies the adder by combining the structural characteristics of the adder, so that it no longer depends on

*通讯作者。

文章引用：黄怡桐，刘帅，魏峰玉，江建国．基于散列表的加法器重写优化算法[J]．应用数学进展，2022，11(11): 8265-8273. DOI: 10.12677/aam.2022.1111874

carrying variables and reduces the number of variables that need to be traversed. The experimental results show that the optimization algorithm improves the generation efficiency of Gröbner basis and the verification efficiency of multipliers.

Keywords

Multiplier Circuit Verification, Gröbner Base, Adder Rewriting Method, Hash Table

1. 引言

在大规模集成电路设计领域中，电路正确性的验证是极其重要的一环。若不加以验证，可能会出现产品的研发失败、缺陷等等问题。例如最著名的“奔腾浮点运算错误[1]”，直接导致 inter 公司损失上亿美元的资产。为了防止此类错误的发生以及减少损失，需要对电路的正确性进行验证。

传统的模拟验证方法费时低效，后经研究发现，形式化验证方法[2]不仅可以发现类似“奔腾浮点运算错误”，而且还可以对错误进行改正。正是因其优越性，形式化验证方法在国内外不断地向更高的理论水平和实践领域蓬勃发展。

形式化验证是通过数学的方法去证明一个系统或电路满足所给的设计规范[3]。形式化验证的方法分为三大类：模型验证、定理证明以及等价性验证。在乘法器电路验证问题上，常采用等价性验证方法。等价性验证分为图表示法和代数表示法[4]。其中，图表示法是利用二叉判定图(BDD) [5]，将电路建模，然后基于有限状态机遍历算法去计算可达状态[6]。此方法虽经长久发展，但状态集呈指数型增长，会出现状态空间爆炸问题；受到图形的限制，有些问题仍无法通过图表示法去解决。因此在乘法器电路验证中，代数表示方法更加适合。

基于 Gröbner 基的代数表示方法是当前主流的验证方法。在此方法中，会对规范进行归约，归约过程中的单项式数目会爆炸式增长，导致验证的速度和效率都会降低；当乘法器位数较高时，Gröbner 基中的多项式数目会非常庞大。针对这类问题，研究者们提出了加法器重写[7]、逐列验证[8]、末级加法器替换[9]等方法，这些方法都有效的解决了规范归约过程中单项式数目爆炸增长的问题。

在加法器重写方法中，随着乘法器位数增加，需要遍历的变量数目会爆炸式增加，并且在识别加法器的过程中依赖进位变量，这些情况都会导致验证效率的降低。

针对以上存在的问题，文本提出了基于散列表的加法器重写优化算法。使用散列表储存电路中全部和位输出变量，结合电路逻辑门中输出和输入的关系，遍历散列表进行扩充，然后根据乘法器的结构特性，再次遍历去识别加法器。此算法有如下的几个优点：第一，优化算法针对全部和位输出变量，而非对全部变量进行遍历，减少了遍历的变量数目；第二，不依赖进位变量识别加法器，而是利用加法器的结构特性来进行识别，识别的速度和效率会提升；第三，加法器重写会出现重复遍历变量的问题，结构也比较混乱，而优化算法把搜寻加法器分为了两个遍历，算法的结构更加清晰明了。由实验结果可知，优化算法加快了 Gröbner 基的生成效率，提高了乘法器电路的验证效率。

2. Gröbner 基理论

本节介绍 Gröbner 基方法的理论基础和理想成员的判定方法，更详尽的的定义及定理可参考[10] [11] [12]。

设 k 是一任意数域，$x_1,\cdots,x_n$ 是 n 个变量(或称文字)，称 $k[x_1,\cdots,x_n]$ 是关于变量 $x_1,\cdots,x_n$ 的 n 元多项式环。

定义 2.1. 设非空子集 $I\subseteq k[x_1,\cdots,x_n]$，若 I 满足下列条件：

1) $0\in I$；

2) 对任意的多项式 $f,g\in I$，都有 $f+g\in I$；

3) 对任意的多项式 $f\in I$，$p\in k[x_1,\cdots,x_n]$ 都有 $pf\in I$。

则称 I 是一个多项式理想。

定义 2.2. 对于任意的 $f_1,\cdots,f_s\in k[x_1,\cdots,x_n]$，若有

$$I=\langle f_1,\cdots,f_s\rangle=\left\{\sum_{i=1}^{s}h_if_i\,\middle|\,h_i\in k[x_1,\cdots,x_n]\right\} \tag{1}$$

成立，则 I 是一个多项式理想，称 $\{f_1,\cdots,f_s\}$ 是 I 的一组基。

定义 2.3. 给定多项式环 $k[x_1,\cdots,x_n]$ 特定的序关系<，I 为任意一个多项式理想，对于非空有限子集 $G=\{g_1,\cdots,g_s\}\subseteq I$，若满足

$$\langle LT(g_1),\cdots,LT(g_s)\rangle=\langle LT(I)\rangle \tag{2}$$

则称 G 是 I 关于序关系<的一组 Gröbner 基，其中 $LT(g_i)$ 表示 g_i 的首项。

对于上述的 G 并不唯一，它与序关系<的选择有关，同时，Gröbner 基理论为理想成员的判定提供了方法：

定理 2.4. 设 I 为任意一个多项式理想，G 是 I 的一组 Gröbner 基，$q\in k[x_1,\cdots,x_n]$，q 除以 G 所得的余式 r 满足 $q-r\in I$，当且仅当 $r=0$ 时，$q\in I$。

3. 代数模型

由上文可知验证乘法器电路的正确性需要得到理想的一组 Gröbner 基。利用 Buchberger 算法虽然可以计算出某一序下的 Gröbner 基，但该算法的时间和空间复杂度都是呈指数级的，所以本文将电路直接建模为一组 Gröbner 基。

设乘法器电路 C 有 $2n$ 个输入变量 $a_0,\cdots,a_{n-1},b_0,\cdots,b_{n-1}$ 和 $2n$ 个输出变量 $s_0,\cdots,s_{2n-1}$，电路中每个内部门变量由 $g_0,\cdots,g_{k-1}$ 表示，给定字典序 $<_{lex}$：

$$X=a_0,\cdots,a_{n-1},b_0,\cdots,b_{n-1},g_0,\cdots,g_{k-1},s_0,\cdots,s_{2n-1} \tag{3}$$

把电路中的逻辑门建模为多项式，其中常见逻辑门的多项式如下 u 所示：

$$\begin{aligned}&u=\neg v\Leftrightarrow -u+1-v=0\\&u=v\wedge w\Leftrightarrow -u+vw=0\\&u=v\vee w\Leftrightarrow -u+v+w-vw=0\\&u=v\oplus w\Leftrightarrow -u+v+w-2vw=0\end{aligned} \tag{4}$$

由于(3)中的变量均是布尔变量，所以变量取值是 0 或 1。对于(3)中任意变量设为 u，称 $-u^2+u$ 为域多项式，它们所构成的集合称为域多项式集。

定义 2.5. 设 C 是一个电路，G 是所有逻辑门的多项式(4)和域多项式集构成的集合，G 生成的理想 $\langle G\rangle$ 记为 $J(C)$。

由定义 2.3 和定义 2.5 可知，在字典序 $<_{lex}$ 下，G 是 $J(C)$ 的一组 Gröbner 基。

定义 2.6. 设 C 是一个电路，多项式 $p\in k[x_1,\cdots,x_n]$。任意 $(a_0,\cdots,a_{n-1},b_0,\cdots,b_{n-1})\in\{0,1\}^{2n}$ 及

$g_0,\cdots,g_{k-1},s_0,\cdots,s_{2n-1}$的结果值代入 p 中使得 p 为 0，则称 p 为电路 C 的规范多项式，记为 PCC，记 $I(C)$ 为所有 PCC 的集合。

定理 2.7. 对于任意无回路电路 C，有 $I(C)=J(C)$。

定义 2.8. 设 C 是一个电路，若多项式

$$\sum_{i=0}^{2n-1}2^i s_i-\left(\sum_{i=0}^{n-1}2^i a_i\right)\left(\sum_{i=0}^{n-1}2^i b_i\right) \tag{5}$$

是一个 PCC，则称电路 C 为乘法器电路，称上述多项式为规范多项式 sp。

根据本节的定理、定义及推论可知，验证乘法器电路的正确性问题转化为了理想成员的判定问题，即验证规范多项式 sp 是否属于 $J(C)$。再由定理 2.4，等价于验证 sp 除以 G 所得余式 r 是否为 0。若为 0，则电路 C 是乘法器电路；若不为 0，则电路 C 不是乘法器电路。

全加器是乘法器电路的重要构成部分，接下来以全加器电路为例，验证其正确性。

例 1：如图 1 所示，验证全加器电路 C 的正确性。

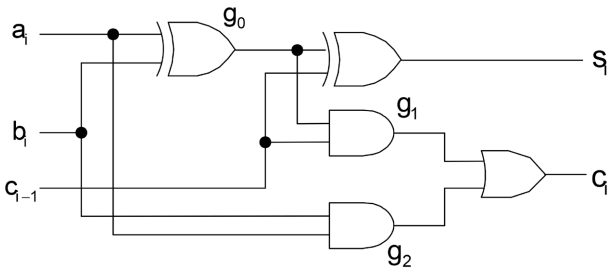

Figure 1. Full-adder circuit
图 1. 全加器电路

电路 C 的规范多项式 $sp=-2c_i-s_i+a_i+b_i+c_{i-1}$，将电路中的变量按逆拓扑序排序：

$$a_i<b_i<c_{i-1}<g_0<g_1<g_2<s_i<c_i \tag{6}$$

则电路 C 在该序下的 Gröbner 基为：

$$\begin{aligned}G_C=\{&-c_i-g_2g_1+g_2+g_1,-s_i-2g_0c_{i-1}+g_0+c_{i-1},-g_2+b_ia_i,\\&-g_1+g_0c_{i-1},-g_0-2b_ia_i+b_i+a_i,-c_i^2+c_i,-s_i^2+s_i,\\&-g_2^2+g_2,-g_1^2+g_1,-g_0^2+g_0,-c_{i-1}^2+c_{i-1},-b_i^2+b_i,-a_i^2+a_i\}\end{aligned}$$

接下来用规范多项式 sp 除以 G_C：

$$\begin{aligned}sp&\xrightarrow{-c_i-g_2g_1+g_2+g_1}-s_i+2g_2g_1-2g_2-2g_1+c_{i-1}+b_i+a_i\\&\xrightarrow{-s_i-2g_0c_{i-1}+g_0+c_{i-1}}2g_2g_1-2g_2-2g_1+2g_0c_{i-1}-g_0+b_i+a_i\\&\xrightarrow{-g_2+b_ia_i}2g_1b_ia_i-2g_1+2g_0c_{i-1}-g_0-2b_ia_i+b_i+a_i\\&\xrightarrow{-g_1+g_0c_{i-1}}2g_0c_{i-1}b_ia_i-g_0-2b_ia_i+b_i+a_i\\&\xrightarrow{-g_0-2b_ia_i+b_i+a_i}-4c_{i-1}b_i^2a_i^2+2c_{i-1}b_i^2a_i+2c_{i-1}b_ia_i^2\\&\xrightarrow{-b_i^2+b_i}-4c_{i-1}b_ia_i^2+2c_{i-1}b_ia_i+2c_{i-1}b_ia_i^2\\&\xrightarrow{-a_i^2+a_i}0\end{aligned}$$

由上述计算得到余式 $r=0$，所以全加器电路 C 是正确的。

4. 加法器重写及优化

4.1. 加法器重写

随着乘法器位数的增加，乘法器电路的 Gröbner 基中多项式数目会急剧增多，导致乘法器验证效率大大降低。为了解决此问题，研究者们提出了加法器重写算法[7]。算法的基本思想是遍历所有变量去识别进位变量，然后利用进位变量去搜索加法器，再进行变量消去，即使用加法器的规范替换 Gröbner 基中对应的所有内部门多项式。该方法简化了 Gröbner 基，提高了乘法器电路的验证效率。

加法器分为全加器(Full-adder)和半加器(Half-adder)，定义如下：

定义 4.1. 设 C 是一个电路，对于输入 a,b,i 及输出 c,s，若 $-2c-s+a+b+i$ 是 PCC，则称电路 C 是全加器；对于输入 a,b 及输出 c,s，若 $-2c-s+a+b$ 是 PCC，则称电路 C 是半加器。

对于例 1 中全加器电路 C 的 Gröbner 基 G_C 经过简化，可得：

$$H_C=\{-c_i-g_2g_1+g_2+g_1,-s_i-2g_0c_{i-1}+g_0+c_{i-1},-g_2+b_ia_i,$$
$$-g_1+g_0c_{i-1},-g_0-2b_ia_i+b_i+a_i\}$$

其中 $\langle H_C\rangle=J(C)$，H_C 是电路 C 的 Gröbner 基。

接下来给出相关定理及定义，消去 H_C 中的内部门变量 g_0,g_1,g_2，用加法器关于输入 a,b,i 的多项式表示输出 s,c。

设 R 为环 $\mathbb{Q}[a_i,b_i,c_{i-1},g_0,g_1,g_2,s_i,c_i]$，$R_{\text{elim}}$ 为环 $\mathbb{Q}[a_i,b_i,c_{i-1},s_i,c_i]$。

定义 4.2. 对于 $J\subset R$，$J_{\text{elim}}=J\cap R_{\text{elim}}$，则称 J_{elim} 是 R_{elim} 的消去理想。

定理 4.3. 设 $J\subset R$ 是理想，如果 H 是 J 关于给定序

$$a_i<b_i<c_{i-1}<s_i<c_i<g_0<g_1<g_2 \tag{7}$$

的 Gröbner 基，则 $H_{\text{elim}}=H\cap R_{\text{elim}}$ 是 J_{elim} 的 Gröbner 基[10]。

根据定理 4.3 中的序关系(7)，计算全加器电路 C 在该序下的 Gröbner 基 H'_C，然后把 H'_C 中关于 g_0,g_1,g_2 的多项式删去，即可得出如下的多项式集合：

$$H'_{\text{elim}}=\{-2c_i-s_i+a_i+b_i+c_{i-1},-s_i+4c_{i-1}b_ia_i-2c_{i-1}a_i+c_{i-1}-2b_ia_i+b_i+a_i\}$$

由定理 4.3 可知 H'_{elim} 是 J_{elim} 的 Gröbner 基。

同样地，对于半加器电路 A 也可以利用加法器重写简化 Gröbner 基，如图 2。

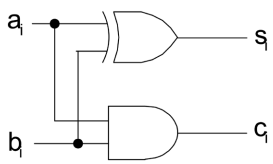

Figure 2. Half-adder circuit
图 2. 半加器电路

半加器电路 A 在逆拓扑序 $a_i<b_i<s_i<c_i$ 下的 Gröbner 基为：

$$H_A=\{c_i-b_ia_i,s_i+2b_ia_i-b_i-a_i,c_i^2-c_i,s_i^2-s_i,b_i^2-b_i,a_i^2-a_i\}$$

经过加法器重写之后的 Gröbner 基为：

$$H_{\text{elim}}=\{2c_i+s_i-b_i-a_i,s_i+2b_ia_i-b_i-a_i\}$$

比较加法器重写前后 Gröbner 基中的多项式数量可以发现，在全加器电路 C 中，Gröbner 基的多项式数量由 13 变为 2；在半加器电路 A 中，Gröbner 基的多项式数量由 6 变为 2。由此得出结论，加法器重写简化了 Gröbner 基，加快了验证效率。但加法器重写方法需要遍历全部变量，随着乘法器位数的增加，遍历的变量数目会急剧增多，验证效率会随之降低；另外，该方法需要借助进位变量去识别加法器，没有对乘法器的结构特点进行利用，也导致了乘法器验证的效率降低。

4.2. 散列表

针对上文提出的加法器重写方法出现的问题，本文将致力于对该方法进行优化，选择了散列表[13]数据结构来储存变量。

散列表又称哈希表或字典，在动态编程中常用于存储键/值(key/value)数据。散列表针对键执行一个散列计算，产生一个整数，使用这一整数找到一个桶(bucket)，然后进行取值或者设值[14]。散列表包含多个桶，即包含多个动态数组，动态数组的数量可以根据需求进行改变，并且易于操作大量数据。当访问散列表某一结点的值时，只需取其 key 值，便可对应到固定的一个结点值。如果使用链表来访问某一元素，就需要遍历它前面所有的元素，消耗相对较长的时间。散列表速度很快，而且它针对任何数据都能工作，因此本文选择散列表对加法器重写方法进行优化。

4.3. 优化算法

基于散列表的优化算法是使用散列表存储乘法器电路中所有的和位输出变量 $s_0, s_1, \cdots, s_{2n-1}$，结合电路逻辑门中输出和输入的关系，利用遍历函数增加表中的变量，根据加法器的结构特性再次遍历表中变量，搜索全加器和半加器，直到遍历结束。此算法相比之前的加法器重写方法，理论上速度更快且效率更高。

如下所示的算法 1，给出了优化算法的实现过程。散列表 Hashmap 中不同的桶存放着不同的结点，结点 HashmapNode 中存放着和位输出变量 vars。算法 1 包含如下几个步骤：1) 设置默认桶数量为 1009，此值可根据和位输出变量数目灵活变动，并且声明变量 hashmap；2) 为散列表 hashmap 分配内存，给定默认比较函数和散列函数 my_hash，为了便于存储，my_hash 为本文自行设计的散列函数；3) 将所有和位输出变量 $s_0, s_1, \cdots, s_{2n-1}$ 存入 hashmap 中；4) 利用遍历函数 Hashmap traverse 以及参数里的函数指针 traverse_good_cb 在遍历 vars 时把识别为加法器的输入变量添加到 hashmap 中；5) 对于 hashmap 再次遍历，利用函数指针 traverse_good_cc 去搜索全加器和半加器，把识别为加法器对应的规范多项式加入 Gröbner 基中，将不属于加法器内部门变量的多项式也加入 Gröbner 基中；6) 由于变量 vars 数量有限，遍历过程得以终止，返回优化后的 Gröbner 基 G，再提供给计算机代数系统进行之后的验证。

算法 1：

输入：乘法器电路 C 的 AIG 文件

输出：电路 C 优化后的 Gröbner 基 G

1. DEFAULT_EXPAND_RATE ← 1009
2. hashmap: Hashmap*
3. hashmap←Hashmap_create(NULL, my_hash)
4. 把所有和位输出变量 $s_0, s_1, \cdots, s_{2n-1}$ 存入 hashmap 中
5. Hashmap traverse (hashmap, traverse_good_cb)
6. Hashmap traverse (hashmap, traverse_good_cc)
7. return G

图 3 为 2 位 btor 乘法器电路结构图，其中 $p_{i,j}=a_ib_j$，接下来对该电路应用优化算法。首先利用散列函数 my_hash 计算 s_0,s_1,s_2,s_3 的 key 值，同时把 s_0,s_1,s_2,s_3 存入散列表 hashmap 中；接着对表中元素遍历，只有 s_1,s_2 识别出加法器，s_1,s_2 对应加法器的输入变量分别是：p_{10} 和 p_{01}、p_{11} 和 c_1，所以将这四个变量添加到 hashmap 中；再次遍历 hashmap 中的变量 $s_0,s_1,s_2,s_3,p_{10},p_{01},p_{11},c_1$。例如，当遍历到 s_0 时，未识别出加法器，把该变量的多项式加入 Gröbner 基；当遍历到 s_1 时，识别出了半加器，把该半加器规范加入 Gröbner 基，并且将 s_1 标记为加法器的内部变量。其余变量均以此方式进行遍历，直至遍历结束，生成优化的 Gröbner 基 G。

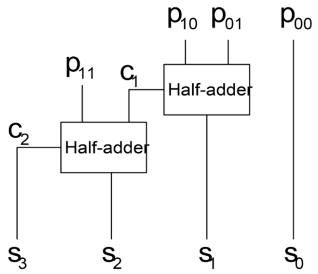

Figure 3. Circuit structure diagram of 2-bit btor multiplier
图 3. 2 位 btor 乘法器电路结构图

基于散列表的优化算法针对所有和位输出变量，而非对全部变量进行遍历，并且利用加法器的结构特性去完成识别，不再利用进位变量完成识别，减少了生成 Gröbner 基的时间，占用的内存也同时减少，算法结构也更加清晰明了。

5. 实验结果

加法器重写方法是 Gröbner 基法中的一个基本策略，目前它没有可替换的方法。本文提出的算法 1 是对加法器重写方法的优化，需要去验证其可行性和有效性，所以将优化算法与加法器重写方法在其他条件都相同的情况下进行实验，并对比分析实验结果。实验是在具有 Ubuntu 18.04.6 虚拟机的笔记本电脑上完成，配置为 Intel i5-6300HQ 2.30 GHz CPU 和 4 GB 内存。实验使用的工具为 AIGMULTOPOLY [7]，计算机代数系统为 Mathematica 和 Singular。选择了两类乘法器电路进行验证，分别是：btor 乘法器、sp-ar-rc 乘法器。这两类乘法器输入位都是部分积，但它们的电路结构有所不同，前者是全加器和半加器以网状结构累积，后者是全加器和半加器以对角结构累积[7]。如图 4 所示，给出 5 位 btor 乘法器和 4 位 sp-ar-rc 乘法器电路结构图。

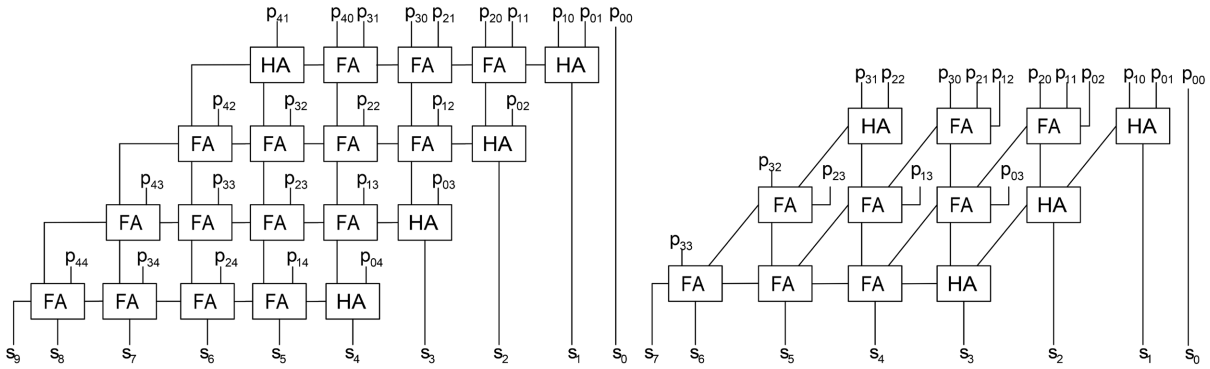

Figure 4. Circuit structure diagram of 5-bit btor multiplier (left) and 4-bit sp-ar-rc multiplier (right)
图 4. 5 位 btor 乘法器(左)与 4 位 sp-ar-rc 乘法器(右)电路结构图

实验的时间表示从开始运行代码到生成 Gröbner 基所花费的时长，单位为秒，时间限制为 1200 秒。内存的单位是 MB，内存限制为 3072 MB。代码及相关文件包含在文件 experiments 中。实验结果如表 1 所示。

表 1 给出了加法器重写方法和优化算法下的实验结果。通过对比分析可以发现，在乘法器类型且位数相同的前提下，也就是同一个乘法器下，相比于加法器重写方法，优化算法耗费了更少的时间，占用了更少的内存。进一步可以发现，随着乘法器的位数增加，时间和内存差距会更加明显。由此可以得出结论，基于散列表的优化算法加快了 Gröbner 基生成速度，减少了内存的使用，提升了 Gröbner 基的生成效率。

Table 1. Comparison of adder-rewriting method and optimization algorithm based on hash table
表 1. 加法器重写方法与基于散列表的优化算法对比

乘法器类型	位数	加法器重写方法		基于散列表的优化算法	
		时间/s	内存/MB	时间/s	内存/MB
btor	16	0.03	2.39	0.01	2.32
btor	32	0.09	3.11	0.05	3.04
btor	64	0.56	8.37	0.20	6.97
btor	128	3.16	29.20	0.78	22.42
sp-ar-rc	16	0.02	2.39	0.01	2.32
sp-ar-rc	32	0.11	3.89	0.06	3.56
sp-ar-rc	64	0.66	11.21	0.25	9.03
sp-ar-rc	128	4.73	40.78	0.96	30.69

6. 结束语

本文旨在基于散列表产生的算法对加法器重写方法进行优化。优化算法使用散列表存储电路中所有的和位输出变量，结合电路逻辑门中输出和输入的关系，遍历表中元素来扩充散列表，结合加法器的结构性知识，再次遍历去识别加法器。结合实验结果可知，优化算法提高了 Gröbner 基的生成效率，减少了内存的使用，不再依赖进位变量并且减少了遍历的变量数目，也提高了乘法器电路的验证速度和验证效率。

在未来该领域的发展中，希望本文的算法可以应用于更加复杂的乘法器架构或除法器。

参考文献

[1] 静思. 让科学之声以更强的力度震撼神州大地——从“奔腾芯片”的错误引发的联想[J]. 电子展望与决策, 1995(3): 23.

[2] 张杰, 王少超, 关永. 基于形式化方法的有限域乘法器的建模与验证[J]. 电子技术应用, 2018, 44(1): 109-113.

[3] 王彦本. 集成电路形式化验证方法研究[J]. 电子科技, 2008, 21(8): 4-6.

[4] Fan, Q.R., Pan, F. and Duan, X.D. (2011) Using Logic Synthesis and Circuit Reasoning for Equivalence Checking. *Advanced Materials Research*, **201-203**, 836-840. https://doi.org/10.4028/www.scientific.net/AMR.201-203.836

[5] Bryant, R.E. (1991) On the Complexity of VLSI Implementations and Graph Representations of Boolean Functions with Application to Integer Multiplication. *IEEE Transactions on Computers*, **40**, 205-213. https://doi.org/10.1109/12.73590

[6] Vedhus, H. (2020) Deep Bayesian Neural Networks for Damage Quantification in Miter Gates of Navigation Locks. *Structural Health Monitoring*, **19**, 1391-1420. https://doi.org/10.1177/1475921719882086

[7] Ritirc, D., Biere, A. and Kauers, M. (2018) Improving and Extending the Algebraic Approach for Verifying Gate-Level Multipliers. 2018 *Design*, *Automation & Test in Europe Conference & Exhibition* (*DATE*), Dresden, 19-23 March 2018, 1556-1561. https://doi.org/10.23919/DATE.2018.8342263

[8] Ritirc, D., Biere, A. and Kauers, M. (2017) Column-Wise Verification of Multipliers Using Compter Algebra. 2017 *Formal Methods in Computer Aided Design* (*FMCAD*), Vienna, 2-6 October 2017, 23-30. https://doi.org/10.23919/FMCAD.2017.8102237

[9] Kaufmann, D., Biere, A. and Kauers, M. (2019) Verifying Large Multipliers by Combining SAT and Computer Algebra. 2019 *Formal Methods in Computer Aided Design* (*FMCAD*), San Jose, 22-25 October 2019, 28-36. https://doi.org/10.23919/FMCAD.2019.8894250

[10] Cox, D., Little, J. and O'shea, D. (2015) Ideals, Varieties, and Algorithms. 4th Edition. Springer, Berlin, 76-104. https://doi.org/10.1007/978-3-319-16721-3

[11] 刘木兰. Gröbner 基理论及其应用[M]. 北京: 科学出版社, 2000: 86-112.

[12] 陈玉福, 张智勇. 计算机代数[M]. 北京: 科学出版社, 2020: 152-167.

[13] 吴洲. 散列表构造与查找的动态实现[J]. 电脑知识与计算, 2014(14): 19-20.

[14] Zed A. Shaw. 笨方法学 C 语言[M]. 王巍巍, 译. 北京: 人民邮电出版社, 2019: 201-202.