Advances in Applied Mathematics 应用数学进展, 2022, 11(11), 8191-8199
Published Online November 2022 in Hans. http://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2022.1111867

Hans汉斯

基于对偶变量的计算机代数证明的机器检验

魏峰玉，黄怡桐，刘　帅，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28日

摘　要

代数推理是目前验证门级整数乘法器最有效的方法之一。实用代数演算是一种涵盖了代数推理并能进行有效证明检验的证明格式。尤其是在代数编码中添加对偶变量生成统一的PAC证明。因为PAC证明文件非常大，且验证过程可能包含错误，本文介绍了一种验证检验器PACHECK2，通过对PAC证明格式进行修改，导出由一系列线性组合组成的证明。进一步识别异或门，分块进行线性组合，减少证明规则的使用，生成简洁的证明。实验表明新的检验器PACHECK2能有效检验证明，验证效率更高，同时提供详细的错误信息。为乘法器的验证提供了一个有效的检验器。

关键词

代数推理，对偶变量，乘法器验证，实用代数演算

Machine Checking for Computer Algebraic Proofs Based on Dual Variables

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

School of Mathematics, Liaoning Normal University, Dalian Liaoning

Received: Oct. 23rd, 2022; accepted: Nov. 18th, 2022; published: Nov. 28th, 2022

Abstract

Algebraic reasoning is one of the most effective methods to verify gate level integer multiplier at present. Practical algebraic calculus is a proof scheme that covers algebraic reasoning and can perform effective proof checking. In particular, adding dual variables to algebraic coding generates uniform PAC proofs. Because the PAC proof file is very large and the verification process may contain errors, this paper introduces a proof checker PACHECK2. By modifying the PAC proof for-

*通讯作者。

文章引用：魏峰玉，黄怡桐，刘帅，江建国．基于对偶变量的计算机代数证明的机器检验[J]．应用数学进展，2022, 11(11): 8191-8199. DOI: 10.12677/aam.2022.1111867

mat, a proof composed of a series of linear combinations is derived. To further identify XOR gates, the linear combination of blocks can reduce the use of proof rules and generate simple proof. The experimental results show that the new checker PACHECK2 can effectively check the proof, the verification efficiency is higher, and at the same time, provide detailed error information. It provides an effective checker for the verification of the multiplier.

Keywords

Algebraic Reasoning, Dual Variables, Multiplier Verification, Practical Algebraic Calculus

1. 引言

形式验证的目的是证明或否定给定系统相对于特定规范的正确性。但是，在计算机代数系统中验证过程也可能包含错误。为了增加验证结果的可信度，必须检验验证工具，一种常见的方法是生成证明证书，该证书包含验证过程中的所有步骤，且可以由一个独立的证明检验器进行检验。

许多形式验证的应用程序使用可满足性(SAT)求解，将验证问题建模为一个 SAT 问题，通过生成消解证明和子句证明[1]如 DRUP [2] [3]、DRAT [4]和 LRAT [5]，来检验验证结果。甚至一年一度的 SAT 竞赛中，从 2003 年开始要求提供不可满足性证书，并提供不同的消解和子句证明格式。但是，该方法不可扩展，一些应用无法使用 SAT 求解，例如算术电路的形式验证，特别是乘法器电路的形式验证。

另一种方法是用计算机代数[6]验证门级乘法器，总体思路是将电路建模成一组多项式，然后进行 Gröbner 基[7]计算，通过对多项式进行约化，检验这些多项式是否隐含电路规范多项式。然而，当乘法器的末级加法器是生成和传播(GP)加法器时，很难用计算机代数来验证。在[8]中用简单行波进位(RC)加法器替换 GP 加法器，将 SAT 和计算机代数结合起来用于验证复杂门级乘法器。最近扩展了[8]的方法，在门级乘法器的多项式编码中表示负文字的对偶变量，在代数推理中考虑了对偶变量，和可以简化复杂末级加法器的新进位重写方法，还引入了尾部替换的概念。从而消除乘法器验证中对 SAT 求解器的调用。

为了验证代数推理结果的正确性，需要调用代数证明系统多项式演算(PC) [9]，PC 作用于多项式，包含每个证明步骤的结论多项式，但缺少验证步骤的来源信息，无法有效检验验证结果。通过将从计算机代数系统中提取的证明转换为 PC 中的多项式反驳，从而将抽象的 PC 规则转化成具体的证明格式，称为实用代数演算(Practical Algebraic Calculus，简称 PAC) [10]。PAC 中存储代数推理验证过程中完整的证明步骤，检验过程中，要确保每一个证明步骤的正确性，并进一步检验至少一个结论多项式与目标多项式匹配，来检验验证结果的有效性。因此，也可以定位证明中错误的证明步骤。

在门级电路的多项式编码中添加了对偶变量，消除乘法器验证中的 SAT 求解器调用，并且能够提供统一的 PAC 证明。避免了两种不同的推理方法生成两种不同证明格式的证明证书 DPUP 和 PAC [4]，导致在证明参数中存在漏洞。同时有些复杂乘法器，尤其是 bp-wt-cl 架构的乘法器，验证过程中使用大量的规则降低了检验的效率。

本文通过对工具 teluma 中获得的单个的、统一的 PAC 证明格式进行修改，导出一系列线性组合组成的证明。该证明格式更加简洁。本文中主要是展示新的检验器 Pacheck2，可以有效检验各种复杂门级乘法器，并且可以找到证明中的错误，检验效率更高。

2. 基本概念

本节介绍算术电路验证，多项式方程的实用代数演算和对偶变量。

2.1. 算数电路验证

在本文中考虑门级无符号整数乘法器电路 C，以与非图(AIG)的形式给出，具有 $2n$ 个输入位 $a_0,\cdots,a_{n-1},b_0,\cdots,b_{n-1}$ 和 $2n$ 个输出位 $s_0,\cdots,s_{2n-1}$，其余内部节点由 $l_1,\cdots,l_k$ 表示。设 $X=\{a_0,\cdots a_{n-1},b_0,\cdots,b_{n-1},l_1,\cdots,l_k,s_0,\cdots,s_{2n-1}\}$。对于所有可能的输入 a_i，b_i，乘法器 C 是正确的当且仅当规范 $L=0$ 成立，其中

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

定义 1. 项 $\tau=x_1^{d_1}\cdots x_r^{d_r}$ 是变量幂的乘积，其中 $d_1,\cdots,d_r\in\mathrm{N}$，$[X]$ 表示项的集合。单项式 $c\tau$ 是项的乘积，其中 $c\in R\setminus\{0\}$，多项式 p 是单项式的有限和。

定义 2. (字典项序)将序 $\le$ 固定在一组项上对所有项 τ，σ_1，σ_2 保持 $1\le\tau$ 和 $\sigma_1\le\sigma_2\Rightarrow\tau\sigma_1\le\tau\sigma_2$，如果对于所有的项 $\sigma_1=x_1^{d_1}\cdots x_r^{d_r}$，$\sigma_2=x_1^{e_1}\cdots x_r^{e_r}$，有 $\sigma_1<\sigma_2$，则令所有 $j<i$ 和 $d_i<e_i$，存在 $d_i=e_i$ 中的 i 索引，则这种顺序称为字典式项顺序。多项式 $p=c\tau+\cdots$ 中 τ (关于≤)称为前导项，记作 $\mathrm{lt}(p)=\tau$。与前导系数 $\mathrm{lc}(p)=c$ 构成前导单项式 $\mathrm{lm}(p)=c\tau$。$p-c\tau$ 叫作 p 的尾部。

在项集上，固定了一个字典项顺序，称为逆拓扑项顺序(RTTO)，这样门的输出变量总是大于输入变量。

定义 3. 一组多项式 $I\subseteq Z[X]$ 称为理想，若 $\forall p,q\in I:p+q\in I$ 和 $\forall p\in Z[X]\ \ \forall q\in I:pq\in I$。如果 $I=\{p_1q_1+\cdots+p_sq_s\mid q_1,\cdots,q_s\in Z[X]\}$，则 $P=\{p_1,\cdots,p_s\}\subseteq Z[X]$ 为 I 的基。则说 I 由 P 生成，写作 $I=\langle P\rangle$。

在[5]中，证明了 L 是否由 C 的门多项式和布尔值约束隐含的问题可以转换为理想成员问题来回答：“给定一个多项式 $q\in Z[X]$ 与多项式的(有限)集 $P\in Z[X]$，决定是否 $q\in\langle P\rangle$。”

定义 4. C 是一个电路，如果 $J(C)=\langle G(C)\cup B(X)\rangle\subseteq Z[X]$ 是由 $G(C)\cup B(X)$ 生成的理想，则电路 C 满足其规范当且仅当 $L\in J(C)$ [5]。

更一般的可以使用 D-Gröbner 基[11]理论。然后应用多元多项式约化并检验唯一最终结果是否为零来确定 $J(C)$。多项式集 $G(C)\cup B(X)$ 对于 $J(C)\subseteq Z[X]$ 的 RTTO，自动产生 D-Gröbner 基[5]，因此可以通过用这些多项式减少 L 来验证电路的正确性。$B(X)$ 的约化通常通过立即约化大于 1 的指数来隐式处理，以加快计算速度。

2.2. 实用代数演算

下面我们介绍 PAC 中用到的一些代数概念[12]。设 X 为变量集 $\{x_1,\cdots,x_n\}$，$G\subseteq Z[X]$ 和 $f\in Z[X]$。

定义 5. 若多项式集 G 的任意一个公共零点都是多项式 f 的零点，则称 G 蕴含 f，记作：$G\models f$。也就是说：$G\models f\Leftrightarrow\forall u\in Z^n:\forall g\in G:g(u)=0\Rightarrow f(u)=0$

代数证明系统通常对多项式方程进行推理。给定 $G\subseteq Z[X]$ 和 $f\in Z[X]$，目的是证明每个 $g\in G(C)\cup B(X)$ 的约束 $g=0$ 隐含了方程 $f=0$。这意味着多项式 $g\in G$ 的每个公共布尔根也是 f 的根。在代数项中，f 属于 $G\cup B(X)$ 生成的理想。

定义 6. 让 $G\subseteq Z[X]$ 是一组有限的多项式。多项式 $f\in Z[X]$ 可以从 G 中推导出来如果 $f\in\langle G\rangle$，记作 $G\vdash f$。

验证证书，如 PAC [9]中的证书，允许监控验证过程。这些证明可以作为推理方法的副产品生成。PAC

证明由三部分组成：1) 多项式的约束集，2) 核心证明，即模拟理想属性的证明步骤序列 P，以及 3) 目标多项式 f。

PAC 的核心使用两个证明规则：ADD 和 MULT。

$$\left[\mathrm{ADD}(i,j,k,p)\right] \quad (X,P)\Rightarrow\left(X,P\left(i\mapsto p\right)\right)$$
$$\text{provided that } P(j)\neq\perp,\ P(k)\neq\perp,\ P(i)=\perp,$$
$$p\in Z[X]/\langle B(X)\rangle,\ \text{and } p=P(j)+P(k) \bmod \langle B(X)\rangle.$$

$$\left[\mathrm{MULT}(i,j,q,p)\right] \quad (X,P)\Rightarrow\left(X,P\left(i\mapsto p\right)\right)$$
$$\text{provided that } P(j)\neq\perp,\ P(i)=\perp,\ p,q\in Z(X)/\langle B(X)\rangle,$$
$$\text{and } p=q\cdot P(j) \bmod \langle B(X)\rangle.$$

这些规则对理想的加法和乘法性质进行建模，例如，在加法规则中，提供了三个多项式 p、q、r，使得 $p+q=r$，以及 p 和 q 要么包含在 S 中，要么在早期的证明步骤中推导。因为 $p,q\in\langle S\rangle$，所以 $r\in\langle S\rangle$。$Z[X]$ 上的 PAC 证明可以由证明检验器 PACHECK 和 PASTEQUE [13]进行检验。

通过添加删除和扩展规则来扩展我们最初的证明规则。在删除规则中，我们从 P 中删除那些后续步骤中不再需要的多项式，以减少工具的内存使用。引入一个扩展规则，它允许在知识库中添加更多的多项式和新变量，同时在原始变量集 X 上保留原始模型。

$$\left[\mathrm{DELETE}(i)\right] \quad (X,P)\Rightarrow\left(X,P\left(i\mapsto\perp\right)\right)$$
$$\left[\mathrm{EXT}(i,v,p)\right] \quad (X,P)\Rightarrow\left(X\cup\{v\},P\left(i\mapsto -v+p\right)\right)$$
$$\text{provided that } P(i)=\perp \quad \text{and} \quad v\notin X \quad \text{and}$$
$$p\in Z[X]/\langle B(X)\rangle,\ \text{and } p^2-p\equiv 0 \bmod \langle B(X)\rangle.$$

考虑下面的 PAC 证明。<input>文件包含给定的多项式集合，PAC 规则包含在文件<proof>中。下面是 PAC 证明格式。

⟨input⟩	⟨proof⟩
1 $x-y$;	3 $*$ 1, z, $xz-yz$;
2 $xz+yz+z$;	4 $+$ 3, 2, $xz+z$;

2.3. 对偶变量

多项式中单项的个数对多项式运算(如加法或乘法)的性能有很大影响。对偶变量是用一个变量来表示另一个变量的求反，为逆变器提供简写符号。将对偶变量集成到电路多项式编码中，引入单独的变量进行门变量的求反，这样，一个多项式的非就可以用另一个单项式来表示，减少多项式的使用。

如下所示，对于一个内部门变量 l_i，$1\le i\le k$，引入对偶变量 $dual(l_i)=f_i$。对偶变量用于 AIG 节点的多项式表示，以编码否定，例如，用多项式 $-l_3+f_1f_2$ 编码图 1 中的门 $l_3=\neg l_1\wedge\neg l_2$。

定义 7. 设 $D(C)=\{-f_i-l_i+1\mid l_i\text{是}C\text{的一个AIG内部节点},f_i=dual(l_i)\}\subseteq Z[X]$，设 $G_D(C)\subseteq Z[X]$ 是用对偶变量生成的门约束集。$J_D(C)=\langle G_D(C)\cup B(X)\cup D(C)\rangle\subseteq Z[X]$，$G_D(C)\cup B(X)\cup D(C)\subseteq Z[X]$ 中的多项式根据 RTTO 排序，对于每个门变量 l_i，有 $f_i>l_i$，即表示逆变器门的对偶变量。

命题 1. 令 $J_D(C)\subseteq Z[X]$ 如定义 7 所示。$G_D(C)\cup B(X)\cup D(C)$ 是 $J_D(C)$ 的 D-Gröbner 基。

命题 2. 对于所有布尔变量 l_i 及其对偶表示 $dual(l_i)=f_i$，$f_i=1-l_i$，有 $l_if_i=0$。

定义 8. 设 m_1，m_2 是 $m_1>m_2$ 的两个单项式，称 m_1 和 m_2 为对偶可合并当且仅当 $m_1=cf_i\tau$，$m_2=cl_i\tau$，

c 为常数， τ 为项，一些指数 i。称单项式 $dmerge(m_1, m_2) = c\tau$ 为其对偶合并。

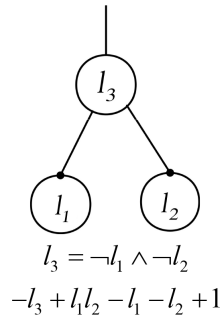

Figure 1. All polynomial encodings covered by AIG nodes
图 1. AIG 节点覆盖的所有多项式编码

Amulet 2.0 [14]在 PAC 中生成证书。在下文中，将描述如何为新的进位重写技术生成证明步骤。首先，通过将对偶约束集 $D(C)$ 添加到证明的约束集，在 PAC 中包含对偶变量。也就是说，多项式的约束集是 $S = G_D(C) \cup D(C)$。 $B(X)$ 在 PAC 中被隐式处理。

算法 1：合并单项式(p)

输入：多项式 p

输出：约化后的多项式 r

```
1  q ← sort-degree-lex(p); r ← 0;
2  While q ≠ 0 do
3     q_l ← lm(q); t ← tial(q); simplify ←⊥
4     While t ≠ 0 and deg(q_l) = deg(lt(t)) and ¬ simplify do
5        q_t ← lt(t);
6        if q_t and q_t are dual mergeable then
7              q ← q − q_l − q_t + dmerge(q_l, q_t);
8              simplify ←⊤;
9        else
10             t ← t − q_t;
11    if ¬ simplify then
12       r ← r + q_l;
13       q ← q − q_l;
14 return sore-lex (r);
```

在核心证明中，必须包括建模命题 2 的步骤。也就是说，每当多项式相乘时，都会生成一个乘法步骤，其中生成的多项式不会减少，并且可能包含 $c \in Z$， $f_i, l_i \in X$ 且 $\tau \in [X]$ 的 $cf_i\tau$ 形式的单项式。生成一个乘法步骤来计算 $(cf_i\tau)(-f_i - l_i + 1) = -cf_i l_i \tau$。使用加法步骤可以取消 $-cf_i\tau$。以类似的方式对算法 1 执

行的约化进行了处理。用 q_l 乘以变量 v 的对偶约束，并将得到的多项式添加到 p，以得到所需的约化。

例 1 令 $p = l_1 f_2 f_3 + l_1 f_2 l_3 + l_1 l_2 f_3 + f_1 f_2 + l_2 \in Z[l_1, l_2, l_3, f_1, f_2, f_3]$，用 q_i 表示迭代 i 后的多项式 q，并表示对偶合并。

$$
\begin{aligned}
q_0 &= l_1 f_2 f_3 + l_1 f_2 l_3 + l_1 l_2 f_3 + f_1 f_2 + l_2 && r = 0\\
q_1 &= l_1 l_2 f_3 + f_1 f_2 + \boxed{l_1 f_2} + l_2 && r = 0\\
q_2 &= f_1 f_2 + l_1 f_2 + l_2 && r = l_1 l_2 f_3\\
q_3 &= \boxed{f_2} + l_2 && r = l_1 l_2 f_3\\
q_4 &= \boxed{1} && r = l_1 l_2 f_3\\
q_5 &= 0 && r = l_1 l_2 f_3 + 1
\end{aligned}
$$

3. PAC 检验器

3.1. PACHECK

Pacheck [13]是 PAC 证明格式检验器 Pactrim [9]的扩展，由 C 语言实现的，可以有效地检验验证工具生成的 PAC 证明。Pacheck 的默认模式支持 PAC 的扩展规则和索引。Pacheck 还支持使用指数进行推理，但扩展规则仅支持布尔模型。

使用工具 Amulet2.0 验证乘法器，同时产生三个文件<constraints>，<proof>和<target>。其中<proof>就是 PAC 证明证书，作为 Amulet2.0 的副产品获得。检验器 Pacheck 将这三个文件作为输入文件，具体检验过程就是理想成员问题：用<proof>中包含的验证计算过程中的全部证明步骤验证<target>中的规范多项式是否包含在由<constraints>中的初始多项式生成的理想中。即用理想证明乘法器，若规范在理想中，则 C 为乘法器电路。

在 Pacheck 中，对变量进行排列支持顺序 strcmp，−1 * strcmp，level，和−1 * level，在默认情况下，使用 strcmp 按字典顺序排列变量。另外一种排序方式是使用与给定证明文件中相同的变量顺序。多项式中的项使用与变量相同的顺序进行排序。为了减少证明文件的大小，引入索引来命名多项式，从而得到了一个更简洁的证明格式。

例 2. 令 $\bar{x} \vee \bar{y}$ 和 $y \vee z$ 为两个子句，从中导出子项 $\bar{x} \vee z$。展示了如何在 PAC 中涵盖这种推导。

使用 DeMorgan 定律和可以用乘法表示逻辑“与”的事实，将这些子句转换为多项式方程。例如，$\bar{x} \vee \bar{y} = \top \Leftrightarrow x \wedge y = \bot$ 得出多项式方程 $xy = 0$。翻译给定的子句，从而构建输入<constraints>和目标<target>。对于<proof>中的 PAC 证明，引入了一个扩展变量 f_z，该变量对 z 的取反进行建模，即 $-f_z + 1 - z = 0$。减少结论多项式的大小。并在 PAC 证明中显示删除规则。例如：

⟨constraints⟩
1 $x*y$;
2 $y*z-y-z+1$;

⟨target⟩
$-x*z+x$;

⟨proof⟩
3 = fz, $-z+1$;
4 * 3, $y-1$, $-fz*y+fz-y*z+y+z-1$;
5 + 2, 4, $-fz*y+fz$;
2 d;
4 d;
6 * 1, fz, $fz*x*y$;
1 d;
7 * 5, x, $-fz*x*y+fz*x$;
8 + 6, 7, $fz*x$;
9 * 3, x, $-fz*x-x*z+x$;
10 + 8, 9, $-x*z+x$;

3.2. PACHECK2

检验器 Pacheck2 是对 Pacheck 进行扩展得到的。Pacheck2 多项式的内部表示与 Pacheck 几乎相同，都是输入由验证工具生成的三个文件<polys>，<proof>和<spec>，同时结合了 Pacheck 的优点，支持扩展规则并能动态的检验中间证明步骤以定位错误。但是，Pacheck2 只支持布尔模型，不再支持使用指数。

初始阶段中，对<polys>的每个多项式进行排序并存储为推理。推理由给定的索引和多项式组成，并存储在哈希表中。检验过程中，动态应用证明检验分析<proof>的每个步骤，用指针 Token 来识别证明中的组件，例如 Token MINUS_TOKEN = “minus operator”，识别证明中的减法操作符；Token PERCENT_TOKEN = “linear combination operator”，识别证明中的线性组合运算。如果证明步骤是 Add 或 Mult，则需要计算该步骤的结论多项式是否等于对先行多项式执行的算术运算。多项式的加法是通过以交错方式合并它们的单项式来执行的，多项式乘法将第一个多项式的每个单项式与第二个多项式的每个单项式相乘。并且对大于 1 的指数进行归一化。然后再解析并检验证明，检验加法或多步的结论多项式是否与<target>中的多项式匹配，以确定是否导出了规范的目标多项式。也就是说，解析一个证明步骤，并计算已知多项式的线性组合等于证明步骤的给定结论多项式。在 Pacheck2 中，支持删除推论，以减少内存使用。

4. 实验

本文中实验使用了一台带有 Ubuntu18.04 虚拟机的电脑，配备 Intel(R) Pentium(R) CPU G4560 3.50 GHz 和 4 GB 主内存。主内存限制为 4 GB。时间以秒为单位，时间限制设置为 300 秒。本文使用工具 Teluma 生成证明证书，以验证输入位宽度为 n 的乘法器的正确性。因为 Teluma 工具去除了验证过程中对 SAT 求解器的需要，获得了单个的、统一的 PAC 证明格式。所以实验选用 aoki 基准，保证乘法器的 FS 加法器都是 GP 加法器。其中乘法器架构包括 PPG：simple (sp)，Booth (bp)；PPA：array (ar)，Balanced delay tree (bd)，compressor tree (ct)，Wallace tree (wt)；FSA：carry look-ahead (cl)，Kogge-Stone (ks)，Ladner-Fisch-er (lf)，Brent-Kung (bk)。

Table 1. Proof checker contrast experiment
表 1. 证明检验器对比实验

architecture	n	[15] PACHECK			our PACHECK2		
		gen (s)	check (s)	total (s)	gen (s)	check (s)	total (s)
sp-ar-cl	32	0.35	0.51	0.86	0.17	0.26	0.43
sp-bd-ks	32	0.40	0.47	0.87	0.21	0.26	0.47
sp-dt-lf	32	0.34	0.36	0.70	0.17	0.19	0.36
bp-ct-bk	32	0.28	0.23	0.51	0.15	0.18	0.33
bp-wt-cl	32	0.95	1.69	2.64	0.49	0.94	1.43
sp-ar-cl	64	2.05	3.49	5.54	1.14	1.99	3.13
sp-bd-ks	64	2.20	2.12	4.32	1.79	1.22	3.01
sp-dt-lf	64	1.79	1.59	3.38	1.04	0.88	1.92
bp-ct-bk	64	1.40	1.43	2.83	0.96	0.90	1.86
bp-wt-cl	64	10.76	25.29	36.05	10.10	18.32	28.42

在实验中，选用了相同的实例。第一块显示了由 Teluma 直接生成的 PAC 证明[15]的证明生成(“gen”)和检验时间(“chk”)，以秒为单位。第二块显示了由 Teluma 生成的新 PAC 证明格式的证明和新的检验器 Pacheck2。从表 1 中可以看出，检验过程中，Pacheck2 比 Pacheck 效率要高，工具生成证明和 Pacheck2 检验证明所需的时间也更少。

5. 结论

本文对在代数编码中包含对偶变量并生成统一的 PAC 证明进行优化，使得该证明格式更加简洁。主要展示了检验器 Pacheck2，能够有效的检验 PAC 校对。实验表明，新的检验器 Pacheck2 可以更高效的检验新扩展的各种乘法器，对复杂的 bp-wt-cl 架构的乘法器，在检验过程中效率也有很大提升。同时能定位证明步骤中的错误位置，提供详细的错误信息。在未来的工作中，希望在 PAC 中捕获更多通用的扩展规则，可以放宽条件 $p^2 = p$。这个条件对于 $v^2 = v$ 是必要的，但是可以取消，即使这意味着 v^n 不能再简化为 v，需要操纵指数。在未来工作中，我们希望能将对偶变量和尾部替换的一般技术应用于更一般的电路验证。

参考文献

[1] Yu, C., Brown, W., Liu, D., Rossi, A. and Ciesielski, M. (2016) Formal Verification of Arithmetic Circuits by Function Extraction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **35**, 2131-2142. https://doi.org/10.1109/TCAD.2016.2547898

[2] Sayed-Ahmed, A., Große, D., Kühne, U., Soeken, M. and Drechsler, R. (2016) Formal Verification of Integer Multipliersby Combining Gröbner Basis with Logic Reduction. *Proceedings of the* 2016 *Conference on Design, Automation & Testin Europe*, Dresden, 14-18 March 2016, 1048-1053. https://doi.org/10.3850/9783981537079_0248

[3] Shekhar, N., Kalla, P. and Enescu, F. (2007) Equivalence Verification of Polynomial Datapaths Using Ideal Membership Testing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **26**, 1320-1330. https://doi.org/10.1109/TCAD.2006.888277

[4] Kaufmann, D., Biere, A. and Kauers, M. (2020) From DRUP to PAC and Back. 2020 *Design, Automation & Test in Europe Conference & Exhibition* (*DATE*), Grenoble, 9-13 March 2020, 654-657. https://doi.org/10.23919/DATE48585.2020.9116276

[5] 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

[6] Kapur, D. (1986) Using Gröbner Bases to Reason about Geometry Problems. *Journal of Symbolic Computation*, **2**, 399-408. https://doi.org/10.1016/S0747-7171(86)80007-4

[7] Buchberger, B. (1965) Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nacheinem nulldimensionalen Polynomideal. Ph.D. Thesis, University of Innsbruck, Innsbruck.

[8] Ritirc, D., Biere, A. and Kauers, M. (2017) Column-Wise Verification of Multipliers Using Computer Algebra. *Proceedings of the* 17*th Conference on Formal Methods in Computer-Aided Design*, Vienna, 2-6 October 2017, 23-30. https://doi.org/10.23919/FMCAD.2017.8102237

[9] Clegg, M., Edmonds, J. and Impagliazzo, R. (1996) Using the Gröbner Basis Algorithm to Find Proofs of Unsatisfiability. *Proceedings of the* 28*th annual ACM Symposium on Theory of Computing*, Philadelphia, July 1996, 174-183. https://doi.org/10.1145/237814.237860

[10] Ritirc, D., Biere, A. and Kauers, M. (2018) A Practical Polynomial Calculus for Arithmetic Circuit Verification. *Workshop on Satisfiability Checking and Symbolic Computation*, Oxford, 11 July 2018, 61-76.

[11] Becker, T., Weispfenning, V. and Kredel, H. (1993) Gröbner Bases. Springer, Berlin. https://doi.org/10.1007/978-1-4612-0913-3

[12] Cox, D., Little, J. and O’Shea, D. (2015) Ideals, Varieties, and Algorithms. 4th Edition, Springer Verlag, New York.

[13] Kaufmann, D., Fleury, M. and Biere, A. (2020) Pacheck and Pasteque, Checking Practical Algebraic Calculus Proofs. *FMCAD* 2020, Volume 1, 264-269.

[14] Kaufmann, D. and Biere, A. (2021) Amulet 2.0 for Verifying Multiplier Circuits. 27*th International Conference*, *TACAS* 2021, Luxembourg City, 27 March-1 April 2021, 357-364. https://doi.org/10.1007/978-3-030-72013-1_19

[15] Kaufmann, D., Beame, P., Biere, A. and Nordström, J. (2022) Adding Dual Variables to Algebraic Reasoning for Gate-Level Multiplier Verification. 2022 *Design*, *Automation & Test in Europe Conference & Exhibition* (*DATE*), Antwerp, 14-23 March 2022, 1431-1436. https://doi.org/10.23919/DATE54114.2022.9774587