Computer Science and Application 计算机科学与应用, 2023, 13(7), 1485-1491
Published Online July 2023 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2023.137147

Hans汉斯

基于指针数组的Gröbner基方法的优化

齐　爽，冯天烁，史美琦，江建国*

辽宁师范大学数学学院，辽宁　大连

收稿日期：2023年6月20日；录用日期：2023年7月20日；发布日期：2023年7月28日

摘　要

门级整数乘法器电路的验证是形式化验证领域内的一个难题，目前最有效的方法是Gröbner基方法。在基于此方法的验证过程中，多项式的表示对内存的使用情况有很大的影响。在验证工具Teluma中，多项式表示为单项式的链表。由于链表结点需要同时存储数据元素本身的信息和一个指示其直接后继的信息，这会占用较大的内存空间。针对这一问题，本文对多项式的数据结构进行了优化，采用了动态数组存储单项式，指针数组表示多项式的方法。实验结果表明，该优化方法减少了验证过程中内存的使用。

关键词

形式化验证，乘法器，对偶变量，Gröbner基，指针数组

Optimization of Gröbner Basis Method Based on Pointer Array

Shuang Qi, Tianshuo Feng, Meiqi Shi, Jianguo Jiang*

School of Mathematics, Liaoning Normal University, Dalian Liaoning

Received: Jun. 20th, 2023; accepted: Jul. 20th, 2023; published: Jul. 28th, 2023

Abstract

The verification of gate-level integer multiplier circuit is a difficult problem in the field of formal verification, and the most effective method is Gröbner basis method. In the verification process based on this method, the representation of the polynomial has a great influence on the amount of memory used. In the verification tool Teluma, polynomials are represented as linked list of monomials. Because linked list nodes need to store both information about the data element itself and a message indicating its immediate successor, this takes up a large amount of memory space. To solve this problem, this paper optimizes the data structure of polynomials. Dynamic array is

*通讯作者。

文章引用: 齐爽, 冯天烁, 史美琦, 江建国. 基于指针数组的 Gröbner 基方法的优化[J]. 计算机科学与应用, 2023, 13(7): 1485-1491. DOI: 10.12677/csa.2023.137147

used to store monomials and pointer array is to represent polynomials. The experimental results show that the optimization method can reduce memory usage in the verification process.

Keywords

Formal Verification, Multiplier, Dual Variable, Gröbner Basis, Pointer Array

1. 引言

形式化验证可用于推导给定电路相对于预定义规范的正确性[1]。为防止与奔腾浮点除错误类似的问题再次发生，算术电路的形式化验证是极其重要的。但到目前为止，算术电路，尤其是整数门级乘法器电路的形式化验证仍然是一个重大难题。

研究者们已经提出了几种用于乘法器的验证方法，包括可满足性(SAT)问题的方法[2]、判定图法[3]、定理证明器法[4]和代数法[5]。目前最有效的方法是代数法中的 Gröbner 基方法。将电路的所有逻辑门和规范用多项式表示，再使用多项式归约算法来检查这些多项式是否隐含了电路规范。但在验证过程中，多项式和冗余单项式的数量过多，严重影响了验证的速度[6] [7]。

为提高验证的速度，研究者们不断优化和改进Gröbner基方法。提出了 XOR 重写和Common 重写[8]、逐位验证[9]和加法器重写[10]等方法，减少了冗余单项式的数量，简化了 Gröbner 基，缩短了验证的时间。但是，当乘法器的末级(FS)加法器为复杂的加法器时，仅应用 Gröbner 基方法很难验证。

在文献[7]中，复杂的 FS 加法器被简单的加法器所替换。结合 SAT 求解器能够验证等价性的特点，用 SAT 求解器验证了替换步骤的正确性，用 Gröbner 基方法验证了简化的乘法器。为节省加法器替换的时间，在文献[11]中，提出了在多项式编码中添加表示否定节点的对偶变量，并通过尾部替换和进位重写算法，简化了复杂的 FS 加法器，实现了验证工具 Teluma。

工具 Teluma 中将多项式表示为单项式的链表。多项式的表示直接影响着多项式的生成效率和多项式所占用的内存空间，进而影响着 Gröbner 基方法的效率和验证过程中内存的使用。由于链表结点在存储数据元素本身的同时，还需要存储直接后继的存储位置，这使得占用的内存空间较大。

为减小多项式占用的内存空间，本文优化了 Teluma 中多项式的数据结构。使用动态数组存储单项式，将多项式表示为一个由指向 Monomial 数据类型的指针组成的指针数组。相比较于将多项式表示为单项式的链表，该方法节省了用于存储直接后继存储位置的内存。实验结果表明，该方法减少了内存的使用。

2. 理论基础

2.1. Gröbner 基理论

本节简单介绍了 Gröbner 基理论的相关定义，其他的定理及其证明参考文献[12]。

定义 2.1. 设 I 为 $R[X]$ 的非空子集，若 $\forall p,q\in I$： $p+q\in I$ 和 $\forall q\in R[X]$， $\forall p\in I$： $pq\in I$。则称 I 是多项式理想。

定义 2.2. 设 $g_1,\cdots,g_s\in R[X]$，若 $I=\{g_1h_1+g_2h_2+\cdots+g_sh_s\,|\,h_i\in R[X]\}$，则集合 $G=\{g_1,\cdots,g_s\}\subseteq R[X]$ 被称为理想 I 的基。即 I 是由 G 生成的，记作 $I=\langle G\rangle$。

定义 2.3. 若项集上有序关系$\le$，则对于所有项τ，σ_1，σ_2，当$1\le\tau$时，有$\sigma_1\le\sigma_2\Rightarrow\tau\sigma_1\le\tau\sigma_2$。

定义 2.4. 设$G=\{g_1,\cdots,g_s\}\subseteq R[X]$为$I\subseteq R[X]$的基，若$\forall h\in I$，$\exists g_i\in G$：$lm(g_i)|lm(h)$。则集合$G$称为理想$I$关于序关系$\le$的一组 Gröbner 基。

Gröbner 基理论为理想成员问题提供了决策过程。对于给定的$h\in R[X]$和基$G=\{g_1,\cdots,g_s\}\subseteq R[X]$，判断$h$是否属于$G$生成的理想。如果$\{g_1,\cdots,g_s\}$是一组 Gröbner 基，那么这个问题可以用一个多元带余除法来解决。当且仅当h除以G的余数为零时，多项式$h\in\langle G\rangle$。

对于上面应用的多项式环$R[X]$，考虑本文的研究内容，我们使用 D-Gröbner 基的更一般的理论，其中系数环是一个主理想域(PID)。设D为 PID，设P为理想$I\subseteq D[X]$的基，若$\forall q\in I$，$\exists p\in P$：$lm(p)|lm(q)$。则称集合P是理想I关于序关系$\le$的一组 D-Gröbner 基[7]。本文中令$D=Z$。

2.2. 代数模型

在应用 Gröbner 基理论的代数推理中，电路需要使用多元多项式建模。在本文中，我们考虑门级无符号整数乘法器电路C，它具有$2n$个输入位$a_0,\cdots,a_{n-1},b_0,\cdots,b_{n-1}$，$2n$个输出位$s_0,\cdots,s_{2n-1}$，以及内部 AIG 节点$l_0,\cdots,l_{k-1}$，所有变量均为布尔变量。

定义 2.5. (字典项序)对于所有项$\sigma_1=x_1^{d_1}\cdots x_r^{d_r}$，$\sigma_2=x_1^{e_1}\cdots x_r^{e_r}$。若$\exists i:d_i<e_i$，对于所有$j<i:d_j=e_j$，则有$\sigma_1<\sigma_2$。

定义在字典项序下的变量为

$$X=a_0,\cdots,a_{n-1},b_0,\cdots,b_{n-1},l_0,\cdots,l_{k-1},s_0,\cdots,s_{2n-1} \tag{1}$$

将每个逻辑门的输出u与输入v，w之间的关系表示为布尔多项式，常见的多项式表示如下：

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

我们称这些多项式为门多项式或门约束。设$G(C)\subseteq Z[X]$为每个 AIG 节点对应的门多项式的集合，$G(C)$中的多项式都按字典项序排列，即每个节点的输出变量都大于其输入变量，这种排序也称为逆拓扑序。为了确保所有变量是布尔值 0 和 1，我们为每个变量$x\in X$加上关系式$x(x-1)=0$，并称这些多项式为布尔值约束，它们的集合表示为$B(X)$，$B(X)\subseteq Z[X]$。

定义 2.6. 设C是一个电路，称$G(C)\cup B(X)$为电路C的多项式集。由多项式集生成的理想记为$J(C)$，$J(C)=\langle G(C)\cup B(X)\rangle\subseteq Z[X]$。

定理 2.1. 设C是一个电路，根据定义 2.4，在字典项序下，$G(C)\cup B(X)$是$J(C)=\langle G(C)\cup B(X)\rangle$的一组 D-Gröbner 基。

定义 2.7. 设C是一个电路，

$$L=-\sum_{i=0}^{2n-1}2^i s_i+\left(\sum_{i=0}^{n-1}2^i a_i\right)\left(\sum_{i=0}^{n-1}2^i b_i\right) \tag{3}$$

称L为规范多项式。若多项式$L\in J(C)$，则C是一个乘法器电路。

电路的规范是输入和输出之间所期望的关系。判断电路正确性的方法是验证电路是否满足其规范。已知在逆拓扑序下，多项式集$G(C)\cup B(X)$是理想$J(C)$的一组 D-Gröbner 基。根据理想成员问题的方法，应用规范L除以多项式集$G(C)\cup B(X)$，并检查结果是否为零。当结果为零时，电路是正确的。

2.3. 对偶变量

在以往的门多项式编码中，分别使用 l_i 和 $1-l_i$ 对 AIG 节点和其否定进行编码。图 1 显示了输入为 l_1 和 l_2 时 AIG 节点 l_3、l_4 和 l_5 的门多项式。可以看到，在子节点的符号为否定时，门多项式中单项式的个数会相应的变多。在多项式的运算过程中，运算时间是根据单项式的个数发生变化的。为了减少运算的时间，考虑引进对偶变量。对于每个内部门变量 l_i，$1\le i\le k$，引进变量 f_i，对 l_i 的否定进行编码。其中 $f_i=1-l_i\in\{0,1\}$，我们称 f_i 为 l_i 的对偶变量，记作 $\mathrm{dual}(l_i)=f_i$。

如图 1 所示，在引进对偶变量后，对于 AIG 节点 $l_5=\neg l_1\wedge\neg l_2$，可以用 $-l_5+f_1f_2$ 进行编码，这使得门多项式中单项式的个数从原来的 5 减小到了 2。

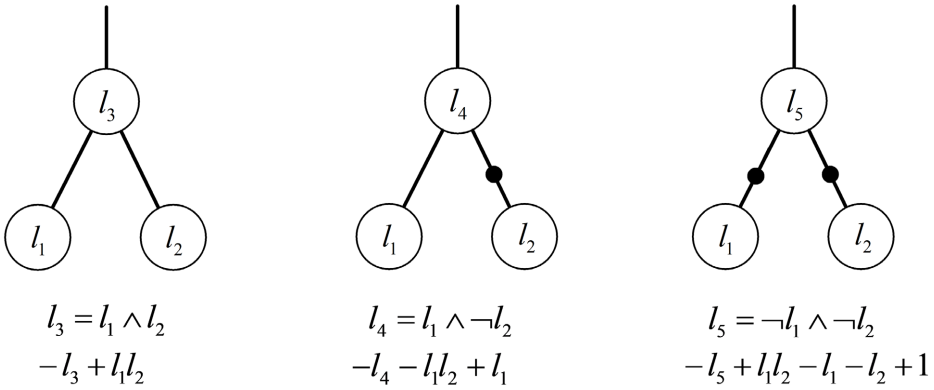

Figure 1. All polynomial encodings covered by AIG nodes
图 1. AIG 节点覆盖的多项式编码

定义 2.8. 设 $D(C)=\{-f_i-l_i+1\mid l_i$ 是 C 的一个内部 AIG 节点，$f_i=\mathrm{dual}(l_i)\}\subseteq Z[X]$，$G_D(C)\subseteq Z[X]$ 为添加对偶变量后的门约束集。电路 C 的多项式集生成的理想为 $J_D(C)=\langle G_D(C)\cup B(X)\cup D(C)\rangle\subseteq Z[X]$，$G_D(C)\cup B(X)\cup D(C)\subseteq Z[X]$ 中的多项式按逆拓扑序排列。对于每个门变量 l_i，有 $f_i>l_i$。

命题 2.1. 设 $J_D(C)\subseteq Z[X]$ 如定义 2.8 所示，则 $G_D(C)\cup B(X)\cup D(C)$ 是 $J_D(C)$ 的 D-Gröbner 基。

在多项式编码中添加对偶变量后，通过进位重写技术简化复杂的 FS 加法器，再应用 Gröbner 基方法进行乘法器的验证。为实现含有对偶变量的验证工具，结合 Amulet 2.0 能够检测复杂的 FS 加法器及其组件等特点[13] [14]，添加基于尾部替换的进位重写算法，生成了 Teluma 工具[11]。

3. Gröbner 基方法优化

对于 Gröbner 基方法的优化，包括简化 Gröbner 基、加快 Gröbner 基的生成速度和减少内存的使用等方面。在应用工具 Teluma 的验证过程中涉及到了众多的数据结构，特别是单项式和多项式的表示。它们的表示影响着 Gröbner 基方法的应用效果，下面分别讨论它们的表示方法。

3.1. 单项式表示

单项式是由系数和项的乘积组成的代数式。在 Monomial 类的代码中，单项式表示为 Monomial (mpz_t _c, Term * _t)，其参数分别表示系数和项。项是变量的幂次方的乘积，由于所有变量均表示布尔值，我们总是默认所有变量的指数均为 1，即假设 $x\cdot x=x$。项可以表示为变量的有序链表，各变量的逻辑顺序通过链表结点的指针链接次序实现。项在归约过程中会被多次使用，因此把它们组织在一个动态放大的哈希表中。每次定义一个新项时，都需要计算它的哈希值并将其插入哈希表。同时使用参考计数器对项进行计数，计数器根据项在多项式中出现的频率递增和递减。

系数的值通常超过 2^{64}，而 C 和 C++语言既没有内建大数运算机制也没有对应的标准库实现。为了解决系数运算的问题，考虑使用 GMP 库进行数字表示，使用 GMP 库中的整数函数进行系数的运算。其中函数 mpz_init_set(coeff, 1)，可将系数 coeff 赋值为 1。例 1 以一个单项式为例，展示了它的表示形式。

例 1. 对于单项式 $2l_1^2l_2l_3$，布尔变量 l_1 的指数可自动减少到 1。若变量 l_1、l_2 和 l_3 的逻辑结构为 (l_1,l_2,l_3)，则它的链式存储结构如图 2 所示。

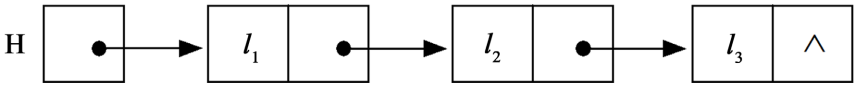

Figure 2. Representation of aterm
图 2. 项的表示

其中 NULL = ∧ = 0，它表示最后一个结点的指针为“空”。设 Monomial*m 是一个指向单项式 $2l_1^2l_2l_3$ 的指针，应用 m−>coeff 即可知道该单项式的系数为 2。

3.2. 多项式表示

在 Teluma 中，使用了 std::deque 将多项式存储为单项式的链表。但是根据容器 deque 的底层结构及链表结点的组成，在存储数据元素本身信息的同时还需要考虑直接后继存储位置的存储，这会占用较大的内存空间。针对这一问题，本文优化了多项式的数据结构。

多项式是有限个单项式的和。为构造多项式，单项式的存储是至关重要的。动态数组具有可动态分配内存的特点。考虑到事先并不清楚多项式中单项式的个数，选择使用动态数组存储用于构造多项式所需的单项式。随着不断将单项式添加到动态数组中，可逐步确定单项式的个数。数组元素全为指针变量的数组称为指针数组。相比较于链表，它仅需要考虑数据元素本身的存储。为减小多项式所占用的内存空间，在单项式的个数确定后，创建一个数组元素为指向 Monomial 数据类型的指针，长度与动态数组相同的指针数组，即应用该指针数组来表示多项式。在 Polynomial 类的代码中，多项式表示为 Polynomial(Monomial**m, size_t len)，其参数分别表示单项式和单项式个数。对于该指针数组，内存大小为动态数组的长度和 sizeof(Monomial*)的乘积。

下面以图 1 中 AIG 节点 l_4 的门约束 $-l_4-l_1l_2+l_1$ 为例，讨论它的构造过程。首先分别构造输出和输入变量的多项式 $-l_4$、l_1 和 $1-l_2$，再根据与门的多项式表示进行多项式的乘法和加法运算。多项式的运算是对内部的单项式进行运算，再根据结果重新构造多项式。在最后的加法运算中，依次将三个单项式 $-l_4$、$-l_1l_2$ 和 l_1 添加到动态数组中，再应用相关函数即可构造出多项式 $-l_4-l_1l_2+l_1$。由于动态数组以 2 倍的方式扩容，所以表示该多项式的指针数组的长度应为 4。例 2 展示了它的表示形式。

例 2. 对于多项式 $-l_4-l_1l_2+l_1$，它是由 3 个单项式组成。根据上述的构造过程，该多项式可表示为一个长度为 4 的指针数组。具体的表示形式如图 3 所示。

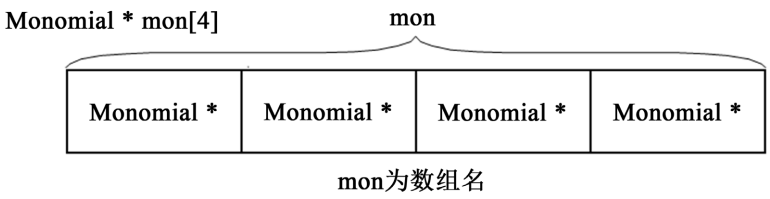

Figure 3. Representation of polynomial
图 3. 多项式的表示

其中 $*\,mon[0]=-l_4$，$*\,mon[1]=-l_1l_2$，$*\,mon[2]=l_1$，$*\,mon[3]=0$。

4. 实验

本次实验使用了一台带有 Ubuntu20.04 虚拟机的电脑，配备 Intel(R) Core(TM) i5-11300H 3.10GHz CPU 和 16 GB 主内存。实验所用时间以秒为单位，时间限制设置为 300 秒。实验代码及相关数据在文件 experiments 中。

首先，我们选择了 AOKI 基准集中[15]的 192 个 64 位无符号乘法器进行实验。其次，为分析优化前后该工具对于大型乘法器的验证情况，我们选择了由 AristKojevnikov 的脚本生成的简单乘法器进行实验，输入位宽分别为 128 位、256 位和 512 位。实验结果如图 4 所示。

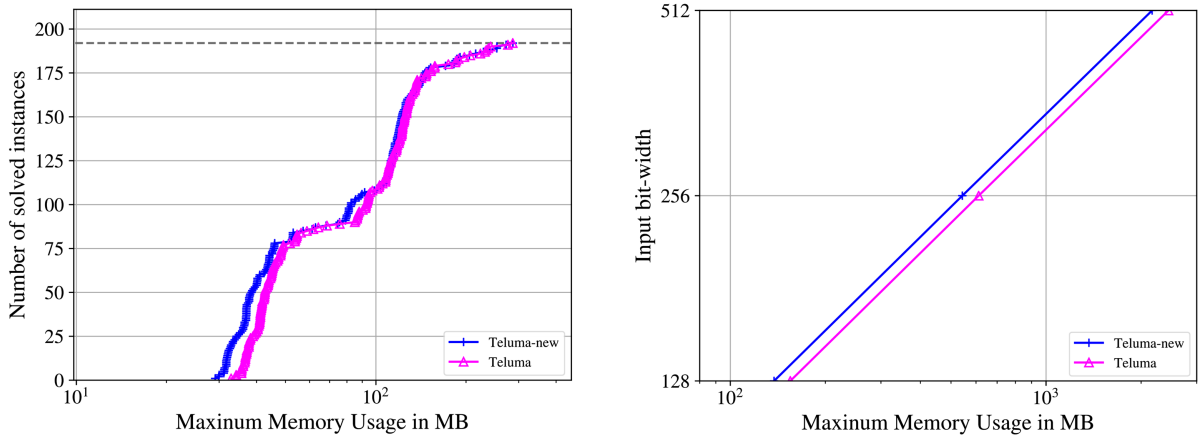

Figure 4. Maximum memory usage of AOKI benchmark set (left) and large multipliers (right)
图 4. AOKI 基准集(左)和大型乘法器(右)的最大内存使用情况

乘法器的验证过程并不一定是完全正确的，工具 Teluma 可以生成证明证书，用于检查验证过程的正确性。如果生成证明证书，会占用更大的内存空间。所以我们选取了 AOKI 基准集中的部分乘法器，用于分析在生成证明证书的情况下，优化多项式的数据结构对内存使用情况的影响。实验结果如表 1 所示。

Table 1. Comparative experimental data before and after optimization
表 1. 优化前后对比实验数据

乘法器	位数	Teluma		优化后	
		时间/s	内存/MB	时间/s	内存/MB
sp-ar-cl	32	0.13	17.19	0.12	15.92
sp-bd-ks	32	0.15	29.68	0.14	27.38
sp-dt-lf	32	0.14	26.55	0.11	25.41
bp-ct-bk	32	0.12	16.69	0.10	16.26
bp-wt-cl	32	0.37	48.86	0.29	47.23
sp-ar-cl	64	0.75	67.87	0.64	63.33
sp-bd-ks	64	0.71	139.48	0.67	133.26
sp-dt-lf	64	0.61	139.50	0.58	137.43
bp-ct-bk	64	0.48	58.46	0.47	57.66
bp-wt-cl	64	3.73	327.27	3.33	323.20

表 1 中显示的是在验证模式下，各类型乘法器的验证时间和占用的最大内存空间，其中默认生成的是统一的 PAC 证明格式的证书。通过查看图 4 的图像和表 1 的数据可以看出，该工具能够在给定时限内验证完成 AOKI 基准集和大型乘法器，并且在优化多项式的数据结构后，所占用的内存空间减小了。

5. 结论

本文优化了 Teluma 中多项式的数据结构，使用动态数组存储单项式，将多项式表示为由指向 Monomial 数据类型的指针组成的指针数组。实验结果表明，使用指针数组表示多项式的优化方法，能够减少验证过程中内存的使用。在未来的工作中，我们希望能够进一步确定添加了对偶变量的布尔多项式的最小表示。

参考文献

[1] 闫硕. 基于多项式符号代数的电路形式验证[D]: [硕士学位论文]. 北京: 北京交通大学, 2011.

[2] Biere, A. (2016) Collection of Combinational Arithmetic Miters Submitted to the SAT Competition 2016. *Proceedings of SAT Competition* 2016—*Solver and Benchmark Descriptions*, Vol. B-2016-1, 65-66.

[3] Bryant, R.E. and Chen, Y. (2001) Verification of Arithmetic Circuits Using Binary Moment Diagrams. *International Journal on Software Tools for Technology Transfer*, **3**, 137-155. https://doi.org/10.1007/s100090100037

[4] Temel, M., Slobodova, A. and Hunt, W.A. (2020) Automated and Scalable Verification of Integer Multipliers. *Computer Aided Verification*, Los Angeles, 21-24 July 2020, 485-507. https://doi.org/10.1007/978-3-030-53288-8_23

[5] Ciesielski, M.J., Su, T., Yasin, A. and Yu, C. (2020) Understanding Algebraic Rewriting for Arithmetic Circuit Verification: A Bit-Flow Model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **39**, 1346-1357. https://doi.org/10.1109/TCAD.2019.2912944

[6] Kaufmann, D. (2022) Formal Verification of Multiplier Circuits Using Computer Algebra. *IT—Information Technology*, **64**, 285-291. https://doi.org/10.1515/itit-2022-0039

[7] 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

[8] Sayed-Ahmed, A., Große, D., Kühne, U., Soeken, M. and Drechsler, R. (2016) Formal Verification of Integer Multipliers by Combining Gröbner Basis with Logic Reduction. 2016 *Design, Automation & Test in Europe Conference & Exhibition* (*DATE*), Dresden, 14-18 March 2016, 1048-1053. https://doi.org/10.3850/9783981537079_0248

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

[10] 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

[11] 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

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

[13] Kaufmann, D. and Biere, A. (2021) AMulet 2.0 for Verifying Multiplier Circuits. 27*th International Conference on Tools and Algorithms for the Construction and Analysis of Systems* (*TACAS* 2021), *Held as Part of the European Joint Conferences on Theory and Practice of Software* (*ETAPS* 2021), Luxembourg City, 27 March-1 April 2021, 357-364. https://doi.org/10.1007/978-3-030-72013-1_19

[14] Kaufmann, D. and Biere, A. (2023) Improving AMulet2 for Verifying Multiplier Circuits Using SAT Solving and Computer Algebra. *International Journal on Software Tools for Technology Transfer*, **25**, 133-144. https://doi.org/10.1007/s10009-022-00688-6

[15] Homma, N., Watanabe, Y., Aoki, T. and Higuchi, T. (2006) Formal Design of Arithmetic Circuits Based on Arithmetic Description Language. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, **E89-A**, 3500-3509. https://doi.org/10.1093/ietfec/e89-a.12.3500