文章编号:1674-2974(2020)12-0061-08 DOI:10.16339/j.cnki.hdxbzkb.2020.12.008

基于最小集合覆盖求解方法的测试向量集约简

欧阳丹彤[1,2],郭江姗[1],张立明[1,2†]
(1. 吉林大学 计算机科学与技术学院,吉林 长春 130012;
2. 符号计算与知识工程教育部重点实验室(吉林大学),吉林 长春 130012)

摘 要:TetraMAX ATPG作为业界性能较优的自动测试向量生成工具,能够使用较短时间产生高故障覆盖率的测试向量集.本文通过对TetraMAX ATPG产生的初始测试向量集进行建模,提出了基于最小集合覆盖求解方法的最小完备测试集生成方法,利用这一算法可以在保证测试向量集故障覆盖率不变的基础上有效地缩减测试集规模,从而降低电路测试成本.实验结果表明该方法对于固定故障类型和静态电路故障类型均具有良好的约简效果.

关键词:自动测试向量生成;测试向量集约简;最小集合覆盖;局部搜索;故障类型
中图分类号:TP301.6 **文献标志码**:A

Test Pattern Set Reduction Based on Minimal Set Covering Solution Method

OUYANG Dantong[1,2], GUO Jiangshan[1], ZHANG Liming[1,2†]
(1. College of Computer Science and Technology, Jilin University, Changchun 130012, China;
2. Key Laboratory of Symbol Computation and Knowledge Engineering (Jilin University), Ministry of Education, Changchun 130012, China)

Abstract: As an excellent ATPG toll, TetraMAX ATPG enable to generate a test pattern set with high fault coverage in a short time. In this paper, by re-modeling the basic test pattern set generated by TetraMAX ATPG, a method for computing an optimal test pattern set based on minimum set covering problem is proposed. This method models the test pattern set reduction problem as an instance of set coverage problem. It can effectively reduce the scale of test set on the basis of guaranteeing the invariable fault coverage of test set, thereby reducing the test cost of the circuit testing. According to the stuck-at fault and the IDDQ Fault, the experimentd results show that the method has good reduction effect on both of them.

Key words: Automatic test pattern generation; test pattern set reduction; minimum set covering; local search; fault type

---

* **收稿日期**:2019-12-18
基金项目:国家自然科学基金资助项目(61872159,61672261,61502199),National Natural Science Foundation of China (61872159, 61672261, 61502199)
作者简介:欧阳丹彤(1968—),女,吉林长春人,吉林大学教授,博士生导师
† 通讯联系人,E-mail:limingzhang@jlu.edu.cn

随着集成电路设计(Integrated Circuit,IC)及工艺技术的发展,人们对设计精度的要求不断提高,IC设计复杂度近乎呈指数增长,芯片故障的测试愈发困难[1].在数字电路测试中,自动测试向量生成(Automatic Test Pattern Generation,ATPG)是对被测电路生成测试向量的过程.首先向被测电路中插入故障列表,将ATPG产生的测试向量作为数字电路的输入序列,根据实际输出信号与无故障电路预期值的差异,区分正常的电路行为和由故障引起的故障电路行为,以此来检测相应的故障[2].由ATPG产生的高质量的测试向量集将进一步被应用到全扫描IC设计或规整的部分扫描设计.因此,ATPG作为IC设计流程的重要组成部分之一[3],工业界和学术界许多学者对其进行了研究.

工业上,ATPG方法逐渐被集成在电子设计自动化(Electronic Design Automation,EDA)工具中以实现工业IC的设计,其中较为著名的ATPG工具有Synopsys研发的Tetra MAX ATPG和Cadence研发的Encounter® Test等.其中Tetra MAX ATPG以其简洁的图形用户界面和强大的命令行被广泛应用,本文中将其简称为TMAX.TMAX支持多种设计风格和测试方法,可以在较短的时间内生成具有高故障覆盖率的测试向量集,以满足后续IC设计要求.

在学术界,国内外许多研究人员都对ATPG方法进行了研究和改进.1966年Roth提出了第一个关于非冗余组合逻辑电路的ATPG方法D方法[4],D方法基于多维敏化思路,对可测故障生成测试向量,理论上解决了非冗余组合电路的测试问题,但不能对冗余电路和有重聚扇出电路进行测试向量生成.Geol提出了D方法的改进方法PODEM方法[5],主旨是对激活的故障回溯其原始输入,搜索所有可能的原始输入值,只要选取一个满足要求的原始输入即可作为测试向量,循环搜索,直到完成所有故障回溯.PODEM方法减少了D方法中回溯与判决的测试,有效提高了计算效率,但仍存在回溯的问题.1983年,Fujiwara等提出了PODEM的改进方法FAN方法[6],引入了唯一蕴涵、唯一敏化、多路回退和头线等概念,采用头线和扇出的回溯方式,有效降低了回溯次数和执行时间,并在扇出点方面做了改进.此后,许多学者相继地提出了ATPG的改进方法,如Schulz等人提出的基于FAN算法的改进算法SOCRATES方法[7],Bryant提出的简化排序二元决策图ROBDD方法[8],Larrabee和Stephan等人提出的基于SAT的ATPG方法[9-10],等等.

随着ATPG方法的发展,由ATPG生成的测试向量集也被应用于更多的研究测试中,但由于生成的测试向量集规模庞大,其中可能包含大量的冗余测试向量,对测试电路成本有较大影响[11-12].因此,对测试向量集进行优化,剔除冗余测试向量,对降低测试成本和测试时间是十分必要的.故障测试向量集的优化常见有动态优化[13]和静态优化[14].动态优化是在测试向量集生成过程中,通过将ATPG方法与测试集压缩方法同时进行,在生成测试向量的同时压缩测试向量集,完成测试向量集优化;静态优化则是在ATPG得到测试向量集后,在保证故障覆盖率不变的基础上缩减测试集规模.

静态测试集优化问题已被证明NP完全问题[15],王小港[16]提出基于遗传算法的测试集约简方法,该方法将测试向量作为染色体,通过编码方式和评估函数的设计,实现测试集约简并取得较好的效果,但由于遗传算法参数设置较多,如种群大小、变异概率、交叉概率等的变化,使得优化效果可能不同;康波等学者[17]提出了基于混沌遗传算法的测试集约简方法,利用混沌序列的随机性、遍历性及规律性等特点来控制遗传算法中的交叉与变异操作,并通过实验验证该方法明显优于标准遗传算法,但是由于混沌序列的随机性,该算法运行时间变动较大;乔家庆等学者[18]采用遗传算法对测试向量排列顺序进行优化,同时采用行列消去法作为适应度评估方法,能有效减少测试向量的数目且优于传统遗传算法,但遗传算法收敛速度慢,容易产生局部最优的情况;侯艳丽等学者[19]将粒子群算法应用于测试集约简,通过构造粒子的表达方式和编码规则,建立粒子群速度-位置模型,利用混沌优化算法来初始化粒子群,使得初始个体本身为一个完备测试集,具有较好的约简效果,但是使用的参数往往依据经验设置,不便于操作;姜伟等学者[20]提出的基于粒子群的多目标测试集优化方法(DPSO),以最大故障检测率、最少测试数目和最小测试代价为优化目标,对测试集进行约简,取得了较好的约简效果,在实际应用中具备一定的局限性.因此,本文提出基于最小集合覆盖求解方法的完备测试向量集求解方法(Complete Test Pattern

Set Solution,CTPSS),该方法将故障集与测试向量集中每个测试向量建立对应关系,通过重新建模,将测试集约简问题转化为最小集合覆盖问题求解,将复杂的实际问题通过简单的模型加以解决.

最小集合覆盖问题作为经典 NP 完全问题,已被应用于各个领域[21],相较精确求解方法,启发式求解方法更适用于大规模问题的求解.在文献[22]中提出一种行加权局部搜索方法 (Row Weighting Local Search,RWLS),该方法提出预处理步骤,在局部搜索前有效减少搜索空间, 且采用多个禁忌策略有效避免搜索的重复和循环, 实验证明该方法针对大规模问题仍可在有限时间内获得更佳解.因此,本文提出的 CTPSS 方法结合 RWLS 方法实现测试向量集约简,其优点为:1)模型适配,最小集合覆盖问题可以有效求解测试向量集约简问题, 且不会降低测试向量集的覆盖率;2)模型简洁,参数设置仅需终止条件,即最大的迭代次数;3)预处理阶段可有效缩减部分数据规模,提高计算效率.

1 预备知识

本节将给出集合覆盖的相关概念,并示例说明.

定义 1 最小集合覆盖 [21]. 给定非空元素集合 R 和一个 R 上的子集族 S, 最小集合覆盖问题 (the Minimum Set Covering Problem,MSCP) 是求解一个 S 的一个子集族 $C\subseteq S$,满足 C 可以覆盖 R 中所有元素且 C 的规模最小. 一个最小集合覆盖实例表示为 $MSCP(R,S)$,其中称 R 为元素集合,S 为初始子集族.

下面通过给出的实例对上述概念进行说明.

例 1 对于 MSCP(R,S),其中 $R=\{a,b,c,d,e\}$, $S=\{\{a,e\},\{b,c\},\{b,c,d\},\{b,d\},\{a,d\}\}$,可知 $S_a=\{\{a,e\},\{b,c,d\}\}$,$S_b=\{\{a,e\},\{b,c\},\{b,c,d\}\}$,$S_c=\{\{a,e\},\{b,c\},\{b,d\}\}$,$S_d=\{\{a,e\},\{b,c\},\{a,d\}\}$ 均可覆盖 R 中所有元素,但 S_a 规模最小,故 $MSCP(R,S)$的解 $C=S_a=\{\{a,e\},\{b,c,d\}\}$.

根据整数规划的定义形式, 一个集合覆盖实例可以用 0-1 矩阵表示.RWLS 方法即在 MSCP(R,S)的 0-1 矩阵基础上,通过行加权的方式,获取最小集合覆盖最优解,下节将给出具体介绍.

2 RWLS 方法

本节将给出 RWLS 方法的伪代码描述及其分值计算方式相关定义.

2.1 RWLS 方法

本小节将给出 RWLS 方法具体介绍, 其伪代码如算法 1 所示,可分为三个部分:前期准备、预处理和局部搜索求解.

算法 1　Algorithm RWLS

输入:MSCP(R,S)
输出:MSCP(R,S)的一个最优解
1. Read problem instance
2. Set stop criteria
3. Preprocessing if necessary
4. bestSol← $\emptyset$
5. C=Initialization ( )
6. LocalSearch (C)
7. Return bestSol

在前期准备阶段,完成 $MSCP(R,S)$问题读入,并设置算法停止条件,以获取限定条件内的最优解,此为 RWLS 方法的唯一参数.

预处理阶段作为 RWLS 方法的一个重要部分,其目的是保证禁忌策略的有效性.在此阶段,对 R 中每一个元素,检查能够覆盖该元素的集合的个数,若其值为 1,即代表该集合一定会被放入候选解集中,则可以将该集合及此集合可覆盖的所有元素从原问题中移除.

局部搜索求解阶段作为 RWLS 方法的关键部分,它实现了一个扰动搜索的框架,假设初始候选解集 C 的大小为 k,若存在更优的解,则其规模一定小于 k. 故首先破坏 C 的可行性,后通过 Add 和Remove 操作[22]不断扰动 C 并试图使 C 重新变成可行,以获取更优解. 同时,RWLS 方法采用了多个禁忌策略[23]避免搜索的重复和循环,包括有禁忌列表 tabu_list, 时间戳 timestamp 以及布尔状态检查 canAddToSolution,并采用一个权重增长策略以帮助算法跳出局部最优. 其算法流程图如图 1 所示,其中,L 为 R 中未被候选解集 C 覆盖的元素的集合.

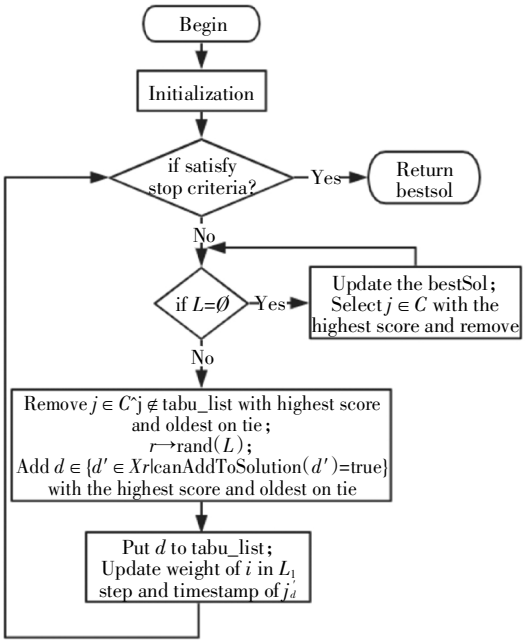

图 1 LocalSearch 方法流程图

Fig.1 The flowchart of LocalSearch

由此可知,每个集合的分值为 LocalSearch 方法集合选取的一个重要因素，下小节将给出 RWLS 方法的分值计算方法.

2.2 RWLS 方法分值计算

本小节将给出 RWLS 方法的分值的计算方式.

定义 2 分值 score(S_j). 若 weight(x_i)表示元素 x_i 的权重,则对于集合族 S 中任意元素 S_j,score(S_j)的值为:

$$\text{score}(S_j)=\begin{cases}\sum\limits_{\substack{x_i\in S_j\\ \sigma(C,x_i)=0}}\text{weight}(x_i) & \text{if } S_j\notin C\\ -\sum\limits_{\substack{x_i\in S_j\\ \sigma(C,x_i)=1}}\text{weight}(x_i) & \text{if } S_j\in C\end{cases}$$

其中,$\sigma(C,x_i)=|C\cap X_i|$,$X_i$ 为集合族 S 中所有可以覆盖元素 x_i 的集合构成的子集族,即 $\sigma(C,x_i)$表示当前候选解集 C 中能够覆盖元素 x_i 的集合的个数.RWLS 方法仅考虑 $\sigma(C,x_i)=0$ 和 $\sigma(C,x_i)=1$ 两种情况:当集合 $S_j\notin C$ 时,score(S_j)为当前所有能被集合 S_j 覆盖且未被 C 覆盖的元素 x_i 的权重之和;当集合 $S_j\in C$ 时,score(S_j)为一个负数,即当前所有能且仅能被集合 S_j 覆盖的元素 x_i 的权重之和的相反数.由此可以看出,当从 C 中添加或删除一个集合 S_j 时，与其具有相同可覆盖元素的集合的分值均有可能发生变化,需重新计算.

3 CTPSS 方法

本节将给出测试集约简问题的建模过程，并给出 CTPSS 方法的相关定义以及其伪代码描述.

3.1 问题建模

本小节将给出测试集约简建模的相关概念及定义,并给出建模过程.

定义 3 故障列表.由电路中所有可能发生的故障组成,记为 $F=\{f_0,f_1,\cdots,f_m\}$,其中 m 为 F 中元素的个数.

定义 4 故障族 F_S. 若对于测试向量集 $T=\{t_0,t_1,t_2,\cdots,t_n\}$中任意一测试向量 $t_j(0\leqslant j\leqslant n)$,$F_j=\{f_a,f_b,f_c,\cdots\}$为 t_j 能够覆盖的故障列表 F 的故障子集,其中 $0\leqslant a,b,c\leqslant m$,则称$\{F_0,F_1,F_2,\cdots,F_n\}$为这个测试向量集 T 对应的故障族,记作 F_S.

定义 5 候选测试向量集 T_C. 称 $T_C=\{t_a,t_b,t_c,\cdots\}$,其中 $0\leqslant a,b,c\leqslant n$ 为故障列表 F 的一个候选测试向量集,当且仅当 $T_C\subseteq T$ 且 $F=\bigcup\limits_{t_k\in T_c}F_k$.

候选测试向量集也可称为完备测试向量集,称候选测试向量集 T_C 是最小完备测试集,当且仅当 T_C 的任意一个真子集都不是候选测试向量集.

定义 6 故障–测试矩阵 FT. 对于故障列表 $F=\{f_0,f_1,\cdots,f_m\}$和测试向量集 $T=\{t_0,t_1,t_2,\cdots,t_n\}$,故障–测试矩阵可以记作:

$$FT_{m\times n}=\begin{bmatrix}ft_{00} & ft_{01} & \cdots & ft_{0n}\\ ft_{10} & ft_{11} & \cdots & ft_{1n}\\ \vdots & \vdots & & \vdots\\ ft_{m0} & ft_{m1} & \cdots & ft_{mn}\end{bmatrix}$$

其中,对于矩阵中任意元素 $ft_{ij}(0\leqslant i\leqslant m,0\leqslant j\leqslant n)$,$ft_{ij}=1$ 表示测试向量 t_j 可以观测到故障 f_i，反之,若测试向量 t_j 不能观测到故障 f_i,则 $ft_{ij}=0$.

测试向量集约简可以描述为，从测试向量 $T=\{t_0,t_1,t_2,\cdots,t_n\}$中选取子集 $T_C\subseteq T$,使之能够覆盖故障列表 $F=\{f_0,f_1,K,f_m\}$中所有元素.根据上述相关定义,测试向量集约简问题可进一步描述为,从测试向量集 T 对应的故障族 $F_S=\{F_0,F_1,F_2,\cdots,F_n\}$中选取一个集合族 $F_C=\{F_a,F_b,F_c,\cdots\}$,$0\leqslant a,b,c\leqslant n$,使之能够覆盖故障列表 $F=\{f_0,f_1,\cdots,f_m\}$中所有元素.根据上述描述，测试向量集约简可以被转化为可由 RWLS 方法求解的最小集合覆盖实例 MSCP(F,F_S),其中，故障–测试矩阵 FT 等同于集合覆盖实例所转

化 0–1 矩阵.此外,本文给出必然测试向量定义如下.

定义 7　必然测试向量. 若存在故障 $f_k \in F$,能且仅能被一个测试向量 $t_j \in T$ 覆盖,则称测试向量 t_j 为必然测试向量.

在预处理阶段，必然测试向量 t_j 及其可覆盖的故障集 F_j 将暂时从原问题中移除，从而缩小局部搜索求解方法的矩阵规模,提高运行效率.下小节将给出 CTPSS 方法的具体过程.

3.2　CTPSS 算法描述

本小节给出了 CTPSS 算法的伪代码，并给予相关说明.

算法 2　Algorithm CTPSS

```
输入:电路网表文件
输出:一个最优测试向量集
1. Begin
2.  readNetlist ( )
3.  Fs←Ø, C←Ø, beatSol←Ø;
4.  F, T = run_tmax( );
5.  For j, pattern in pat_get(T):
6.    Fj = run_tmax_single (pattern);
7.    add (Fj);
8.  End For
9.  Read problem instance ←MSCP (F, Fs);
10.  Set stop criteria;
11.  For fi in F:
12.    If check(fi) ==1:
13.      C ← GetMust(fi);
14.      L ← L–Fj;
15.    End If
16.  End For
17.  C=Initialization (C);
18.  bestSol = LocalSearch(C)
19.  Tc = change(bestSol);
20.  Get_fc(Tc);
21.  End
```

第 2~8 行获取 TMAX 生成的测试向量及其对应故障列表:readNetlist( )用于读取电路文件,若当前电路为时序逻辑电路，则需要进一步读入与其相关的时钟信息文件;初始化故障族 F_S,候选解集 C 和最优解 bestSol；通过运行 TMAX 得到初始测试向量集 T 以及故障列表 F;第 5~8 行为获取集合簇 F_S,通过 pat_get( )函数将测试向量集 T 分割为多个独立可执行测试向量 t_j,使之可以被 TMAX 识别,从而得到 t_j 可覆盖的故障集合 F_j,最终获得故障族 F_S.

第 9~17 行调用 RWLS 方法:读入最小集合覆盖实例 MSCP(F,F_S);设置终止条件,本文中将其设定为一个预期最大迭代次数，同时若当前候选解集规模小于等于必然测试向量数目加 1,则算法可提前终止,当前候选解集即为最优解;在预处理阶段添加函数 check( )来检测每个故障的被覆盖次数,若故障的被覆盖次数为 1,则存在必然测试向量 f_i,通过函数 GetMust( )获取 t_j 并予以标记后加入到候选解集 C 中，将其可覆盖故障集合 F_j 中所有元素从未覆盖列表 L 中移除,以保证后续禁忌策略成功进行;局部搜索求解阶段，以预处理得到的部分候选解集 C 为输入,贪婪算法初始化得到完整的初始候选解集 C,最后调用 LocalSearch( )方法得到最优解.

第 18~19 行为验证过程,使用 change( )函数获取 bestSol 中的测试向量,组合后得到新的测试向量集 T_C,并通过 Get_fc ( )调用 TMAX 对新测试向量集 T_C 与基础测试向量 T 的故障覆盖率进行对比,若两者相同则证明此次约简成功.

4　实验与结果

本节给出了 CTPSS 方法的实验分析，实验环境如下:Ubuntu 14.04，CPU Intel Core i5 –8250U 1.80GHz，8GB RAM,python 和 C++，使用 TetraMAX 生成初始测试向量集并完成后续覆盖率检验. 本文使用的测试电路为基准电路 ISCAS85[24],全扫描版本的 ISCAS89[25]电路和 ITC99[26]电路.

同时在实验对比中,本文选取了文献[20]中的 DPSO 方法,DPSO 方法以最大故障覆盖率、最少测试向量数量和最小代价为优化目标，取得了较好的测试向量集约简效果.因此,在实际所需代价相同的情况下，本文以 TMAX 提供的初始测试向量集故障覆盖率为最大故障覆盖率，即保证测试向量集覆盖率不变,以约简后的测试向量集规模为统一衡量指标,对 CTPSS 方法与 DPSO 方法进行了实验结果对比. 同时,DPSO 方法终止条件与 CTPSS 方法终止条件均为预期最大迭代次数，但 DPSO 方法无预处理步骤,因此,可通过对比实验验证 CTPSS 方法的预处理步骤在测试向量集约简问题中的重要性.

4.1　故障模型

在使用 TMAX 生成故障列表时，使用故障模型为固定型故障（Stuck–at Fault,SAF). SAF 故障模型是 ATPG 工具中常用的故障类型，很多故障模型也可以用不同的 SAF 组合表示而成. 一个能够对 SAF 故障具有较高覆盖率的测试向量集对其它故障类型

也会具有较高的故障覆盖率[27],该测试向量集的实用性较高.

同时，为验证 CTPSS 算法对于多种故障类型同样适用,选取 TMAX 支持的静态电路故障(Integrated Circuit Quiescent Current Fault,IDDQ) 中的 the pseudo-stuck-at model 故障模型[28]进行了实验,此类故障类型也较容易获得较高的故障覆盖率.

4.2 CTPSS 实验结果

表 1 中给出了 CTPSS 方法与 DPSO 方法实验结果对比,其中,约简后测试向量集故障覆盖率均与初始测试向量集故障覆盖率相同,故障类型为 SAF.

其中,第 1 列(Cir Name)为电路名称,以 S 开头的电路为 ISCAS89 电路，以 B 开头的电路为 ITC99 电路;第 2 列(Flt Num)为该电路中插入的 SAF 故障类型的故障列表 F 的规模;第 3 列(TMAX)为 TMAX 生成的具有高故障覆盖率的初始测试向量集 T 的规模；第 4~7 列为 CTPSS 方法与 DPSO 方法得到的故障覆盖率不变的缩减后的测试向量集规模(Pat Num)及其对应的缩减率(Re Rate),其中缩减率为减少的测试向量在初始测试向量集中所占的比例.实验过程中,将 CTPSS 方法和 DPSO 方法的终止条件最高迭代次数均设置为 5×10^3 次. 由于部分电路规模较大,DPSO 算法在可接受范围内不能产生解,表示为“—”.

从表 1 可以看出,CTPSS 方法对测试集的缩减效果明显优于 DPSO 方法，且对于大规模电路,CTPSS 方法同样可以得到良好的缩减效果. 对于表 1 中所有电路,85%的电路测试集在 CTPSS 方法中缩减率高达 20%,其中缩减率高于 30%的占 23%左右.因此，可以看出 CTPSS 方法对测试向量集的缩减有着良好的效果,可以有效缩减测试向量集,从而降低电路测试成本.

表 2 给出了预处理阶段必然测试向量数目(Num *of* t_j)与约简后的测试向量集数目(Num *of* T_C)的对比,实验数据表明,在测试向量集约简这一实际问题中，预处理阶段在缩减算法数据规模上有着明显效果.

同时，表 2 给出了 CTPSS 方法的实际平均迭代次数(iterations),而 DPSO 方法因无预处理步骤不能提前终止算法，其实际平均迭代次数均为 5×10^3. 迭代次数作为 DPSO 方法与 CTPSS 方法的算法终止条件,同时也是影响算法时间复杂度的重要因素. 假设算法最大迭代次数为 m，测试向量集规模为 N,则 DPSO 方法的时间复杂度可表示为 $O(N\times m)$[20-29]. 同

表 1 CTPSS 与 DPSO 结果对比

Tab.1 Comparison of CTPSS and DPSO results

Cir Name	Flt Num	TMAX	DPSO		CTPSS	
			Pat Num	Re Rate/%	Pat Num	Re Rate/%
s27	64	11	8	27.3	7	36.4
s208_1	221	33	27	18.2	27	18.2
s298	396	43	34	20.9	29	32.6
s344	464	38	29	23.7	25	34.2
s349	448	35	31	11.4	22	37.1
s382	538	50	41	18.0	36	28.0
s386	352	65	57	12.3	52	20.0
s400	535	45	38	15.6	33	26.7
s420_1	500	82	74	9.8	72	12.2
s444	517	42	37	11.9	33	21.4
s510	605	85	60	29.4	60	29.4
s526	587	54	47	13.0	40	25.9
s526n	581	50	45	10.0	39	22.0
s641	558	55	47	14.5	43	21.8
s713	565	56	47	16.1	44	21.4
s820	723	116	88	24.1	85	26.7
s832	742	116	86	25.9	88	24.1
s838_1	992	192	164	14.6	155	19.3
s1196	1333	221	177	19.9	166	24.9
s1238	1368	221	189	14.5	171	22.6
s1423	2083	124	96	22.6	81	34.7
s1488	1423	147	118	19.7	114	22.4
s1494	1421	147	119	19.0	110	25.2
s5378	4282	289	249	13.8	229	20.8
s9234_1	3628	192	164	14.6	147	23.4
s13207	5740	211	187	11.4	169	19.9
s15850	2530	124	107	13.7	96	22.6
s35932	34592	60	—	—	45	25.0
s38417	37514	1081	—	—	811	25.0
s38584	31603	602	—	—	427	29.1
b01	179	21	16	23.8	16	23.8
b02	121	17	12	29.4	11	35.3
b03	730	42	34	19.0	29	31.0
b04	2274	113	90	20.4	73	35.4
b05	1794	131	103	21.4	85	35.1
b06	270	21	17	19.0	17	19.0
b07	1615	98	84	14.3	73	25.5
b08	602	63	51	19.0	49	22.2
b09	620	40	35	12.5	32	20.0
b10	650	61	48	21.3	42	31.1
b11	1869	189	165	12.7	145	23.3
b12	4213	236	195	17.4	159	32.6
b13	1213	61	50	18.0	43	29.5
b14_1	15985	620	—	—	451	27.3
b14	17065	655	—	—	493	24.7
b15_1	25920	1210	—	—	870	28.1
b15	25834	1248	—	—	899	28.0

表 2　t_j 数目与 T_C 数目对比

Tab.2　Comparison of the number of t_j and T_C

Cir Name	Num of t_j	Num of T_C	iterations	Cir Name	Num of t_j	Num of T_C	iterations
s27	7	7	1	s9234_1	138	147	5 000
s208_1	26	27	1	s13207	160	169	5 000
s298	25	29	5 000	s15850	92	96	5 000
s344	24	25	1	s35932	35	45	5 000
s349	19	22	3	s38417	784	811	5 000
s382	33	36	3	s38584	393	427	5 000
s386	51	52	1	b01	15	16	1
s400	30	33	3	b02	9	11	2
s420_1	72	72	1	b03	28	29	1
s444	32	33	1	b04	63	73	5 000
s510	57	60	3	b05	80	85	5 000
s526	38	40	2	b06	17	17	1
s526n	37	39	2	b07	67	73	5 000
s641	41	43	2	b08	48	49	1
s713	43	44	1	b09	28	32	5 000
s820	80	85	5 000	b10	37	42	5 000
s832	86	88	2	b11	132	145	5 000
s838_1	153	155	2	b12	141	159	5 000
s1196	158	166	5 000	b13	38	43	5 000
s1238	166	171	5 000	b14_1	418	451	5 000
s1423	76	81	5 000	b14	469	493	5 000
s1488	112	114	2	b15_1	829	870	5 000
s1494	107	110	3	b15	863	899	5 000
s5378	222	229	5 000				

时,对于 CTPSS 方法,其最好时间复杂度为 O(N),即在预处理产生的必然测试向量基础上，仅需少数迭代就可以满足算法终止条件，如表 2 中电路 s27、s208_1 等;其最坏时间复杂度为 $O(d\times m)$,在已标记必然测试向量的大规模电路中,d 的值往往小于 N.因此,由以上分析可知,CTPSS 方法的时间复杂度一般优于 DPSO 方法.

为验证 CTPSS 方法对于其他故障类型同样适用,表 3 给出了 CTPSS 方法在 ISCAS85 电路上分别采用 SAF 故障类型与 IDDQ 故障类型下的实验结果.左侧为故障类型为 SAF 时的实验数据,右侧为故障类型为 IDDQ 时的实验数据. 实验证明对于不同故障类型,CTPSS 方法都能获得良好的测试集约简效果.

表 3　ISCAS85 在 SAF 和 IDDQ 下的实验结果比较

Tab.3　Comparison of ISCAS85 under SAF and IDDQ fault model

Cir Name	SAF				IDDQ			
	Flt Num	TMAX	CTPSS	Re Rate	Flt Num	TMAX	CTPSS	Re Rate/%
c17	14	7	6	14.3	14	3	3	0
c432	86	26	15	42.3	86	6	5	16.7
c499	145	20	10	50.0	146	6	5	16.7
c880a	165	24	14	41.7	172	12	8	33.3
c1355	146	15	10	33.3	146	6	5	16.7
c1908	116	18	9	50.0	116	7	5	28.6
c1908a	116	14	9	35.7	116	7	5	28.6
c2670	588	46	32	30.4	745	12	7	41.7
c2670a	436	48	33	31.3	441	13	8	38.5
c3540	144	30	14	53.3	144	9	6	33.3
c3540a	144	36	20	44.4	144	10	6	40.0
c5315	596	39	17	56.4	602	17	10	41.2
c5315a	596	38	18	52.6	602	15	10	33.3
c6288	128	7	5	28.6	128	8	6	25.0
c7552	612	58	32	44.8	630	14	7	50.0

5　结束语

本文通过对测试向量集约简问题的合理建模，通过建立测试向量集与故障列表之间的关系，将其转化为最小集合覆盖问题，并结合最小集合覆盖问题的求解方法行加权局部搜索方法 RWLS 提出了 CTPSS 方法，实现对 TMAX 产生的初始测试向量集的约简.CTPSS 方法简洁容易操作，提供高效的预处理过程,确保不降低测试向量覆盖率的同时,实现测试向量集的约简. 实验证明其对于约简不同故障类型生成的测试向量集均具有良好效果，对降低后续电路测试成本具有重要的现实意义.但是,当问题规模过大时,CTPSS 方法存在建模占用空间较大的不足,有待后续解决.

参考文献

[1] LIU M,OUYANG D T,CAI S W,*et al*. Efficient zonal diagnosis with maximum satisfiability [J]. Science China Information Sciences, 2018,61(11):17—30.

[2] 刘梦,欧阳丹彤,刘伯文,等. 结合问题特征的分组式诊断方法[J]. 电子学报,2018,46(3):589—594.
LIU M,OUYANG D T,LIU B W,*et al*. Grouped diagnosis approach using the feature of problem [J]. Acta Electronica Sinica,2018,46(3):589—594. (In Chinese)

[3] 曾健平,王振宇,袁甲,等. 一种改进的 SRAM 故障内建自检测算法[J]. 湖南大学学报(自然科学版):2019,46(4):97—101.
ZENG J P,WANG Z Y,YUAN J,*et al*. An improved SRAM fault built-in-self-test algorithm [J]. Journal of Hunan University(Natural Sciences),2019,46(4):97—101.

[4] ROTH J P. Diagnosis of automata failures:a calculus and a method [J]. IBM Journal of Research and Development,1966,10 (4):278—291.

[5] GOEL P. An implicit enumeration algorithm to generate tests for combinational logic circuits [J]. IEEE Transactions on Computers,1981,C-30(3):15—222.

[6] FUJIWARA H,SHIMONO T. On the acceleration of test generation algorithms [J]. IEEE Transactions on Computers,1983,C—32(12):1137—1144.

[7] SCHULZ M H,TRISCHLER E. Socrates:a highly efficient automatic test pattern generation system [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,1988,7(1):126—137.

[8] BRYANT R E. Graph—based algorithms for boolean function manipulation [J]. IEEE Transactions on Compilers,1986,35(8):677—691.

[9] LARRABEE T. Test pattern generation using boolean satisfiability [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,1992,11(1):4—15.

[10] STEPHAN P R,BRAYTON R K,SANGIOVANNI V A L. Combina-tional test generation using satisfiability [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,1996,15(9):1167—1176.

[11] HIGAMI Y,KAJIHARA S,ICHIHARA H,*et al*. Test cost reduction for logic circuits:reduction of test data volume and test application time [J]. Systems and Computers in Japan,2005,36(6):69—83.

[12] XU J J,CHEN R,DU Z J. Probabilistic reasoning in diagnosing causes of program failures [J]. Software Testing,Verification and Reliability,2016,26(3):176—210.

[13] RUDNICK E M,PATEL J H. Efficient techniques for dynamic test sequence compaction [J]. IEEE Transactions on Computers,1999,48(3):323—330.

[14] HAMZAOGLU I,PATEL J H. Test set compaction algorithms for combinational circuits [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems:A Publication of the IEEE Circuits and Systems Society,2000,19(8):957—963.

[15] FUJIWARA H,TOIDA S. The complexity of fault detection problems for combinational logic circuits [J]. IEEE Transactions on Computers,1982,C—31(6):555—560.

[16] 王小港. 遗传算法在 VLSI 设计自动化中的应用研究 [D]. 北京:中国科学院,2001:94—98.
WANG X G. Application and research of genetic algorithms in VLSI design automation [D]. Beijing:Chinese Academy of Sciences,2001:94—98. (In Chinese)

[17] 康波,陈光禹,吕炳朝. 基于混沌遗传算法的故障测试集最小化方法[J].仪器仪表学报,2005,26(1):100—103.
KANG B,CHEN G J,LÜ B C. A minimization approach for fault test set based on chaos genetic algorithm [J]. Chinese Journal of Scientific Instrument,2005,26(1):100—103. (In Chinese)

[18] 乔家庆,付平,尹洪涛. 基于遗传排序的测试集优化[J]. 电子学报,2007,35(12):2335—2338.
QIAO J Q,FU P,YIN H T. Test set optimization based on genetic reordering [J]. Acta Electronica Sinica,2007,35(12):2335—2338. (In Chinese)

[19] 侯艳丽,赵春晖,胡佳伟.基于粒子群算法的故障测试集优化[J]. 电子测量与仪器学报,2008,22(4):21—25.
HOU Y L,ZHAO C H,HU J W. Fault test set optimization based on particle swarm optimization algorithm [J]. Chinese Journal of Electronic Measurement and Instrument,2008,22(4):21—25. (In Chinese)

[20] 姜伟,王宏力,张忠泉,等. DPSO 算法在故障诊断测试集优化中的应用[J]. 自动化仪表,2013,34(4):14—17.
JIANG W,WANG H L,ZHANG Z Q,*et al*. Application of DPSO algorithm in optimizing the test set for fault diagnosis [J]. Process Automation Instrumentation,2013,34(4):14—17. (In Chinese)

[21] 方琼,邵瑾. 检入管理 CIMS 系统中的集合覆盖问题 MSCP 研究[J]. 集成电路应用,2018,35(7):18—21.
FANG Q,SHAO J. Research on set covering problem in check in management system [J]. Applications of IC,2018,35(7):18—21. (In Chinese)

[22] CHAO G,XIN Y,THOMAS W,*et al*. An efficient local search heuristic with row weighting for the unicost set covering problem [J]. European Journal of Operational Research,2015,246(3):750—761.

[23] GLOVER F W,LAGUNA M. Tabu search [M]// Metaheuristic Procedures for Training Neutral Networks. Berlin:Springer,2006:125—151.

[24] BRGLEZ F,FUJIWARA H. A neutral netlist of 10 combinational benchmark circuits and a target translator in Fortran [C]//Proceedings of IEEE International Symposium on Circuits and Systems. Kyoto:IEEE,1985:695—698.

[25] BRGLEZ F,BRYAN D,KOZMINSKI K. Combinational profiles of sequential benchmark circuits [C]//Proceedings of IEEE International Symposium on Circuits and Systems. Portland:IEEE,1989:1929—1934.

[26] CORNO F,REORDA M,SQUILLERO G. RT-level ITC'99 benchmarks and first ATPG results [J]. IEEE Design and Test of Computers,2000,17(3):44—53.

[27] BUSHNELL M,AGRAWAL V D. Essentials of electronic testing for digital,memory and mixed-signal VLSI circuits [M]. Berlin:Springer,2013:155—210.

[28] SYNOPSYS:TetraMAX ATPG user guide,v2017.09 [EB/OL]. [2019-12-18] http://www.synopsys.com.

[29] 王沁,李磊,陆成勇,等. 平均计算时间复杂度优化的动态粒子群优化算法[J]. 计算机科学,2010,37(3):197-200.
WANG Q,LI L,LU C Y,*et al*. Average computational time complexity optimized dynamic particle swarm optimization algorithm [J]. Computer Science,2010,37(3):197-200. (In Chinese)