计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2019.20180754
56(11):2448-2457,2019

基于极小碰集求解算法的测试向量集约简

欧阳丹彤[1,3] 陈晓艳[1] 叶 靖[2,4] 邓召勇[1] 张立明[1,3]
[1](吉林大学计算机科学与技术学院 长春 130012)
[2](计算机体系结构国家重点实验室(中国科学院计算技术研究所) 北京 100190)
[3](符号计算与知识工程教育部重点实验室(吉林大学) 长春 130012)
[4](中国科学院计算技术研究所 北京 100190)
(ouyangdantong@163.com)

Test Pattern Set Reduction Based on the Method of Computing Minimal Hitting Set

Ouyang Dantong[1,3], Chen Xiaoyan[1], Ye Jing[2,4], Deng Zhaoyong[1], and Zhang Liming[1,3]
[1](*College of Computer Science and Technology, Jilin University, Changchun* 130012)
[2](*State Key Laboratory of Computer Architecture (Institute of Computing Technology, Chinese Academy of Sciences), Beijing* 100190)
[3](*Key Laboratory of Symbol Computation and Knowledge Engineering (Jilin University), Ministry of Education, Changchun* 130012)
[4](*Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190)

Abstract The purpose of automatic test pattern generation (ATPG) is to determine a high-quality set of test patterns for a particular fault model. Automatic test pattern generation is a very important part in chip testing. Through using the test set, generated by the automatic test pattern generation process, we can detect most of the faults in the circuit so that the fault coverage of the chip (design) can reach the desired value. Nowadays, there are many commercial tools available to generate the set of test patterns. Among these tools, TetraMAX ATPG 2018 is the most powerful and easy-to-use automatic test pattern generation tool. It can generate the highest quality test pattern set with the highest fault coverage in the shortest amount of time. In this paper, a method for computing minimal complete test pattern set based on the minimal hitting set method is proposed. By re-modeling the test pattern set reduction problem, the test set generated by TetraMAX ATPG 2018 is reduced with the method of computing minimal hitting set. This method can effectively reduce the scale of the test pattern set and ensure that the fault coverage of the test set does not change. It has important practical significance to reduce the test cost of the chip. In the experimental part of the paper, we use stuck-at fault as the fault model. The experimental results show that the proposed method can effectively reduce the size of the test set. At the same time, the method we proposed can guarantee that the obtained test pattern set does not contain redundant test pattern.

收稿日期:2018-11-08;**修回日期**:2019-05-08
基金项目:国家自然科学基金项目(61672261,61502199,61402196,61872159,61704174)
This work was supported by the National Natural Science Foundation of China (61672261, 61502199, 61402196, 61872159, 61704174).
通信作者:张立明(limingzhang@jlu.edu.cn)

Key words circuit test; automatic test pattern generation (ATPG); test pattern set; reduction; fault coverage; minimal hitting set; stuck-at fault

摘 要 自动测试向量生成的目的是对特定的故障模型确定1个高质量测试向量集使得芯片(设计)的故障覆盖率达到期望值,在芯片测试中是非常重要的环节.TetraMAX ATPG 2018是众多ATPG工具中功能最强、最易于使用的自动测试向量生成工具,可以在很短的时间内生成具有高故障覆盖率的高质量测试向量集.提出基于极小碰集求解算法的极小完全测试向量集求解算法,通过对测试向量集约简问题重新建模,利用极小碰集求解算法对TetraMAX ATPG 2018产生的测试向量集进行约简.利用这一算法可以有效地缩减测试向量集规模,且保证其故障覆盖率不变,对降低芯片的测试成本有着重要的现实意义.实验针对固定型故障,结果表明:该算法具有良好的约简效果,而且可以保证所得测试向量集中不包含冗余的测试向量.

关键词 电路测试;自动测试向量生成;测试向量集;约简;故障覆盖率;极小碰集;固定型故障

中图法分类号 TN407; TP306

随着信息产品需求的增长,集成电路市场也在其带领下迅速发展,电路的复杂度不断增加,使得电路的测试成本不断提高.为了解决这一问题,人们在设计阶段就开始考虑可测试性问题,以保证自动测试向量生成(automatic test pattern generation, ATPG)过程得以有效进行.ATPG的任务是针对特定的故障模型确定1个高质量测试向量集(test pattern set, TPS),使得设计的故障覆盖率达到期望值[1].在工业界,以Synopsys,Mentor Graphic,Cadence这3家公司为代表的芯片设计公司都提供了商用的ATPG工具.由Synopsys公司开发的TetraMAX ATPG 2018是现阶段工业界功能最强和最容易使用的ATPG工具,针对不同的电路设计,TMAX 2018可以在很短的时间内生成具有高故障覆盖率的TPS.本文中将TetraMAX ATPG 2018简记为TMAX 2018.

TPS的规模对电路的测试成本,如测试时间、测试功耗以及被测电路的寿命等问题有较大影响[2].为了缩短测试时间、降低测试成本,国内外学者在TPS优化、TPS压缩等领域进行了大量研究.TPS优化问题根据不同的优化目标可以大致分为2类:1)以故障覆盖率与故障隔离率为可测试性指标,以测试代价最小为目标对TPS进行优化,在优化的过程中,可测试性指标要求故障覆盖率与故障隔离率均不低于某个值,是兼顾测试成本、检测与诊断效果的优化方式[3-4].2)TPS优化问题的目的是对TPS进一步约简,求出其没有冗余的最小TPS,使其向量个数最少,同时尽可能保证故障覆盖率保持不变[5-9].在约简的过程中,并没有考虑TPS的诊断效果与测试所需的代价.优化之后的TPS将提供给自动测试仪(automatic test equipment, ATE)的测试程序使用.TPS压缩问题是指,在ATE测试的过程中,测试数据被存储在ATE的存储器中,由于ATE有限的存储资源和测试带宽往往难以满足片上系统高速测试的实际需求,通常要对测试数据进行压缩,减少测试过程中需要存储的数据空间,把数据转换成比原格式更紧凑的形式.测试数据压缩作用于待测电路对应的TPS,并对其进行压缩存储[10-11].

在TPS优化问题研究领域,国内外许多学者对保证TPS覆盖率不变的同时减少TPS数目的TPS约简问题进行了研究.俞龙江等人[5]提出基于蚁群算法的TPS约简算法,对最初的蚁群算法模型进行修正,设置参数较少,多次运行可以获得相同解,具有较强的鲁棒性,但是不能保证最终解中没有冗余的测试向量;乔家庆等人[6]提出利用遗传排序算法对TPS中的测试向量顺序进行优化,并采用行列消去法作为适应度评估算法,有效地减少了测试向量的数目,优于传统的遗传算法,但是遗传算法本身收敛速度慢、寻优效率低,容易陷入局部最优的算法仍然不可避免;侯艳丽等人[7]将粒子群算法应用于TPS约简问题,该算法重新定义粒子的位置和速度,利用具有全局优化能力的混沌算法将初始化粒子群,并且粒子针对故障编码,使得初始个体本身就是1个完备TPS,取得了较好的效果,但使用的参数往往依据经验设置,以最优解的适应性没有变化作为结束条件,在实际的应用中有一定的局限性;曹义亲等人[8]提出基于粗糙集的TPS优化算法中,利用粗糙集的知识约简理论构造电路对应知识系统,

生成区分函数,对该函数进行求解得到最小 TPS,该算法可以有效平衡约简效果与约简所用时间,但是不能保证约简后 TPS 的覆盖率与原 TPS 相同;王宏力等人[9]提出的基于粒子群的多目标 TPS 优化算法(以下简记为 PSO-Td),以最大故障覆盖率和最少测试向量数目为优化目标,对 TPS 进行约简,取得了较好的约简效果.

以上 TPS 约简算法在尽可能保证约简后的 TPS 故障覆盖率保持不变的情况下,最大程度减少 TPS 中的测试向量数目,随机算法得到的 TPS 约简结果具有一定的随机性.在对以上工作的深入研究基础上,本文提出基于极小碰集求解算法[12-13]的极小完全测试集求解算法(minimal complete test pattern set reduction, MCTPSR).

该算法只需要针对 TPS 建立测试向量与需要检测的故障列表之间的对应关系,生成对应的向量覆盖集合簇,将 TPS 的约简问题转化为极小碰集的计算问题,即可使复杂的约简问题通过简单的模型得到解决.该算法的主要优点有 5 方面:

1) 可以得到所有的极小解,是完备性算法,同时保证所有解中都没有冗余的测试向量;

2) 可以很好地适应 TPS 约简问题,且其计算效率比较高,约简产生的开销远远小于因冗余测试所产生的开销;

3) 模型简单,仅需要结合测试向量与故障之间的对应关系重新建模即可求解,且求解过程不需要其他参数和多次迭代;

4) 具有选择的多样性,对于 1 个 TPS,往往可以得到多个极小解,可以根据实际的测试开销选择合适的解;

5) TPS 的约简不会降低故障覆盖率,同时由 MCTPSR 算法产生的所有解都可以保证有效性.

1 预备知识

本节介绍 TPS 约简问题、基于模型诊断和碰集的相关概念,以及 MHS-DMECV 算法中使用的相关概念与定义,并通过实例进行说明.

1.1 TPS 约简

针对特定的电路,对于其中可能出现的所有故障的集合 $F=\{f_0,f_1,\cdots,f_n\}$,TMAX 2018 可以产生 1 个 TPS $T=\{\boldsymbol{t}_0,\boldsymbol{t}_1,\cdots,\boldsymbol{t}_m\}$,用于检测 F 中所有故障.其中,$\boldsymbol{t}_i(0\leqslant i\leqslant m)$表示 1 个测试向量,$\boldsymbol{t}_i$ 可以检测到 F 中 1 个或多个故障,$\boldsymbol{t}_i$ 对应的故障集表示为 $F_i(F_i\subseteq F)$.TPS 约简是指通过删除冗余的测试向量,保留尽可能少的 $\boldsymbol{t}_i$,构成新的 TPS $T'=\{\boldsymbol{t}'_0,\boldsymbol{t}'_1,\cdots,\boldsymbol{t}'_n\}$且 $T'\subseteq T$,使 T'仍然可以检测到 F 中所有的故障,即 $\bigcup_{i'=0}^{n'}F_{i'}=F$.

例 1. 设某电路中的故障集 $F=\{f_0,f_1,f_2,f_3\}$,TPS $T=\{\boldsymbol{t}_0,\boldsymbol{t}_1,\boldsymbol{t}_2\}$,其中 $\boldsymbol{t}_0$ 对应的故障集为 $F_0=\{f_0,f_1\}$,$\boldsymbol{t}_1$ 对应的故障集为 $F_1=\{f_2\}$,$\boldsymbol{t}_2$ 对应的故障集为 $F_2=\{f_2,f_3\}$.显然在 T 中,$\boldsymbol{t}_1$ 是 1 个冗余的测试向量,因此会被删除,最终保留测试向量 $\boldsymbol{t}_1$ 和 $\boldsymbol{t}_3$,$T'=\{\boldsymbol{t}_1,\boldsymbol{t}_3\}$,使 $F_1\cup F_3=F$,TPS T 被约简为 T'.

1.2 碰 集

定义 1[14]. 1 个系统可定义为 1 个三元组(SD, $COMPS$,OBS),其中:

1) SD 为系统描述,是一阶谓词公式集;

2) $COMPS$ 是系统组成部件的集合,是 1 个有限常量集;

3) OBS 为观测集,是一阶谓词公式的有限集.

定义 2[14]. 冲突集 CS 是 1 个部件集 $\{c_1,c_2,\cdots,c_n\}\subseteq COMPS$,当且仅当 $SD\cup OBS\cup\{AB(c_1),AB(c_2),\cdots,AB(c_n)\}$是不一致的.其中 AB 为一元谓词,表示“abnormal”.$AB(c)$为真,当且仅当 c 异常,且 $c\in COMPS$.

定义 3[14]. 碰集.设 $CS=\{S_i\mid i=1,2,\cdots,n\}$是集合簇,称 HS 为 CS 的 1 个碰集,如果 HS 满足:

1) $HS\subseteq\bigcup S$;

2) 对于任意 1 个 $S\in CS$ 都有 $HS\cap S\neq\varnothing$.

集合簇 CS 的 1 个碰集是极小碰集(minimal hitting set, MHS)当且仅当它的任意真子集都不是 CS 的碰集.

例 2. 对于集合簇 $CS=\{\{1,2\},\{2,3\},\{2,4\}\}$,根据碰集的定义,通过计算得到 CS 的碰集 HS 有{2},{1,2},{2,3},{2,4},{1,2,3},{1,2,4},{2,3,4},{1,2,3,4}.其中,集合簇 CS 的 MHS 为{2}和{1,3,4}.

1.3 MHS-DMECV 算法

本节介绍极小碰集算法 MHS-DMECV 中的相关定义与概念,并给出实例进行说明.

定义 4[12]. 容量.若集合簇 CS 中包含元素的个数为 n,则称 n 为集合簇 CS 的容量,记为 $CAP(CS)=n$.

设 U 是集合簇 CS 中所有元素的并集构成的集合,即 $U=\bigcup S_i=\{e_1,e_2,\cdots,e_m\}$,用 $Car(U)$表示集合 U 的势,即集合 U 中元素的个数.

定义 5[12]. 频数.若 $e\in S$,则称集合 S 含有元素 e,记 $Freq(CS,e)$表示集合簇 CS 中含有元素 e 的元素的个数,称为元素 e 在集合簇 CS 中的频数.

定义 6[12]. 元素覆盖值.若元素 e 在集合簇 CS 的某个元素 $S_i(i=1,2,\cdots,n)$中出现,且在当前状态下 S_i 中没有其他元素出现,则称元素 e 覆盖 S_i;若对于集合簇 CS 中的所有元素,元素 e 能覆盖的元素个数为 C,则称 C 为元素 e 的元素覆盖值,记为 $Cover(e)$.显然 $0\leqslant Cover(e)\leqslant Freq(CS,e)$.

例 3. 对于集合簇 $CS=\{\{1,3\},\{2,4\},\{1,2,3\}\}$,$CS$ 的容量 $Cap(CS)=3$,CS 中所有元素的并集 $U=\{1,2,3,4\}$,集合 U 的势为 $Car(U)=4$,元素 1 在 CS 中的频数 $Freq(CS,1)=2$,因为元素 1 在 CS 的元素中出现了 2 次.对于集合簇 CS,若首先选择元素 1,由于它能覆盖集合{1,2}和{1,2,3},所以元素 1 的元素覆盖值 $Cover(1)=2$,若接着选择元素 2,此时集合{1,2}和{1,2,3}已被覆盖,所以它能覆盖的集合只有{2,4},因此 $Cover(2)=1$,接下来若选择元素 3 或 4,此时所有集合都被覆盖,所以 $Cover(3)$和 $Cover(4)$的值在该时刻都为 0,因此不需要选择元素 3 或 4,只需要元素 1 和 2 就可以完成对集合簇 CS 中所有集合的覆盖.不难看出 CS 的 1 个碰集是{1,2},由于其非空真子集{1},{2}都不能覆盖 CS 中的所有元素,因此{1,2}为 CS 的 1 个极小碰集.

2 MHS-DMECV 算法

本节通过伪代码描述 MHS-DMECV 算法,并通过实例介绍其流程.

算法 1. MHS-DMECV 算法[12].

输入:待处理序列 Seq、(与 Seq 对应的)$Head$ 邻接链表数组和元素覆盖值 $Cover$ 数组、碰集保存容器 $HittingSet$、集合簇 CS 的元素覆盖标志数组 $SetFlag$、已覆盖数 $AlreadyCover$;

输出:集合簇 CS 对应的所有 MHS.

① 初始化:

$AlreadyCover=0$,$HittingSet=\varnothing$,

$SetFlag=[0\times CAP(CS)]$,

根据 CS 初始化 Seq 序列、$Head$ 数组、$Cover$ 数组;

② Begin

③ While ($Seq\neq[]$)

④ $e\leftarrow Seq$ 第 1 项;

⑤ If e 是最后 1 项 and $Cover(e)+AlreadyCover=Cap(CS)$

⑥ $container\leftarrow\{e\}\cup HittingSet$;

⑦ EndIf

⑧ If $container$ 是极小碰集

⑨ $Set_{MHS}=Set_{MHS}\cup container$;

⑩ Return;

⑪ Else

⑫ $HittingSet=HittingSet\cup\{e\}$;

⑬ $update()$;

⑭ $create(Seq)$;

⑮ 计算 $CanCover$;

⑯ If $CanCover+AlreadyCover<CAP(CS)$

⑰ $HittingSet=HittingSet-\{e\}$;

⑱ $un\_update()$;

⑲ Else

⑳ $update(Seq')$;

㉑ $MHS\text{-}DMECV(params)$;

㉒ $HittingSet=HittingSet-\{e\}$;

㉓ $update()$;

㉔ EndIf

㉕ EndIf

㉖ EndWhile

㉗ End

Seq 表示对 U 中元素的 $Cover$ 值从大到小排列得到的序列;$Head$ 是用来存储 U 中每个元素对应于 CS 中的集合索引的邻接链表;$update()$,$update(Seq')$表示更新相关变量信息;$un_update()$表示撤销对相关参数的更新;$create(Seq)$表示构建新的 Seq'和与其对应的 $Head'$,$Cover'$.

例 4. 对于集合簇 $CS=\{\{1,3\},\{2,3\}\}$,初始时,$Cover(1)=1$,$Cover(2)=1$,$Cover(3)=2$.根据元素的 $Cover$ 值,得到的 Seq 序列为[3,1,2],其对应的 $Head$ 如图 1 所示,此时还没有元素被选中,因此此时没有集合被覆盖,$AlreadyCover=0$,$SetFlag=[0,0]$.首先处理 Seq 序列中的第 1 项"3",更新相关参数值,$AlreadyCover=2$,$SetFlag=[1,1]$,$Seq'=[]$,$CanCover=0$,$AlreadyCover+CanCover=CAP(CS)$,因此得到 CS 的 1 个碰集{3},将其加入 $HittingSet$,根据极小碰集的定义判断出{3}是集合簇 CS 的 1 个极小碰集,将其加入 Set_{MHS};接着处理元素 1,此时新的待处理序列 $Seq''=[2]$,对应的 $Head$ 如图 2 所示,此时 $AlreadyCover=1$,因为集合{1,3}已经被 Seq 序列中的第 2 项"1"覆盖,而集合

{2,3}没有被 Seq 序列中的第 2 项"1"覆盖,因此元素覆盖标志数组 $SetFlag=[1,0]$,可以覆盖的元素数 $CanCover=1$,由于 $AlreadyCover+CanCover=CAP(CS)$,可以得到 CS 的 1 个碰集{1,2},通过判断得出它是 1 个极小碰集,将其加入 Set_{MHS};然后处理 Seq 序列中的第 3 项"2",此时 Seq 中没有其他元素,$AlreadyCover=1$,且 $AlreadyCover+Cover(2)<CAP(CS)$,返回;算法结束.容易看出,最后得到的 $Set_{MHS}=\{\{1,2\},\{3\}\}$ 是 MHS-DMECV 算法所求得的 CS 所有的极小碰集.

算法的完备性已在文献[12]中进行证明.

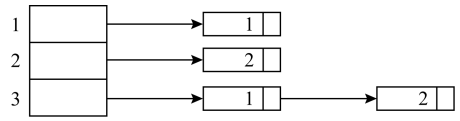

Fig. 1 The adjacency list *Head* corresponding to pending sequence *Seq*′

图 1 待处理序列 *Seq*′对应的邻接链表 *Head*

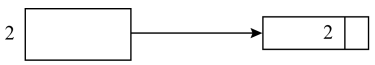

Fig. 2 The adjacency list *Head* corresponding to pending sequence *Seq*″

图 2 待处理序列 *Seq*″对应的邻接链表 *Head*

3 MCTPSR 算法

本节对 TPS 约简问题进行建模,介绍其与极小碰集求解问题中相关联的概念,并通过伪代码描述基于 MHS-DMECV 的 TPS 约简算法 MCTPSR.

3.1 问题建模

本节介绍 TPS 约简问题的相关概念与定义,并通过实例进行说明,同时对 TPS 约简问题进行建模,将其转化为适合极小碰集求解算法处理的模型.

定义 7. 故障序列.电路中可能发生的所有故障组成的拥有确定顺序的集合,记为 $F=\{f_0,f_1,\cdots,f_n\}$,F 的容量为 n.

在文献[6]中描述了完全测试集的概念,下面我们给出标准定义.

定义 8. 完全测试集(complete test pattern set, CTPS).设 $T=\{\boldsymbol{t}_0,\boldsymbol{t}_1,\cdots,\boldsymbol{t}_m\}$ 是电路的 1 个 TPS,$F=\{f_0,f_1,\cdots,f_n\}$ 是电路的故障序列,F_i 是 $\boldsymbol{t}_i$ 可以测试到的故障,称 T 是该电路的 1 个完全测试集,当且仅当 $\bigcup_{i=0}^{m}F_i=F$.

称 CTPS T 是极小完全测试集(minimal complete test pattern set, MCTPS).当且仅当 T 的任意 1 个真子集都不是 CTPS.

定义 9. 覆盖值.如果测试向量 $\boldsymbol{t}_i(\boldsymbol{t}_i\in T)$ 可以检测到电路中的故障 $f_j(f_j\in F)$,则称该测试向量 $\boldsymbol{t}_i$ 覆盖了故障 f_j,$(\boldsymbol{t}_i,f_j)$ 的覆盖值为 1,记为 $C_{ij}=1$,否则 $C_{ij}=0$.

定义 10. 故障覆盖集.若 $\boldsymbol{t}\in T$ 可以检测到电路中的故障 $\{f_i,f_j,\cdots,f_k\}(0\leqslant i,j,k\leqslant n)$,则 $\boldsymbol{t}$ 对应的故障覆盖集为 $\{f_i,f_j,\cdots,f_k\}(0\leqslant i,j,k\leqslant n)$,记为 $F_t=\{f_i,f_j,\cdots,f_k\}(0\leqslant i,j,k\leqslant n)$.

定义 11. 向量-故障矩阵.设 $T=\{\boldsymbol{t}_0,\boldsymbol{t}_1\cdots,\boldsymbol{t}_m\}$ 是电路的 1 个 TPS,$F=\{f_0,f_1,\cdots,f_n\}$ 是电路中的故障序列,由 T,F 中的元素 $\boldsymbol{t}_i$ 与 f_j 的覆盖关系 C_{ij} 组成的矩阵,称为该电路的向量-故障矩阵,记为 ***PFM***.

例 5. 设 TPS $T=\{\boldsymbol{t}_0,\boldsymbol{t}_1,\boldsymbol{t}_2,\boldsymbol{t}_3\}$,故障集合 $F=\{f_0,f_1,f_2,f_3,f_4\}$,其中,$t_0$ 的故障覆盖集为 $F_0=\{f_0,f_3,f_4\}$,因此可以得到覆盖值 $C_{00}=1$,$C_{01}=0$,$C_{02}=0$,$C_{03}=1$,$C_{04}=1$,同理可以得到 T 中其他测试向量与故障 $f_j\in F$ 的覆盖值,这些覆盖值构成 ***PFM***,如表 1 所示.其中,$PFM(0,0)=0$ 表示测试向量 $\boldsymbol{t}_0$ 不能覆盖到故障 f_0,$PFM(1,3)=1$ 表示测试向量 $\boldsymbol{t}_1$ 可以覆盖到故障 f_3.从表 1 可以看出,$T=\{\boldsymbol{t}_0,\boldsymbol{t}_1,\boldsymbol{t}_2,\boldsymbol{t}_3\}$ 是 1 个 TPS,但由于其子集 $\{\boldsymbol{t}_0,\boldsymbol{t}_1,\boldsymbol{t}_2\}$ 依然可以覆盖 F 中的全部故障,它显然不是 1 个 MCTPS.实际上,由 $F_0=\{f_0,f_3,f_4\}$,$F_2=\{f_1,f_2,f_3\}$,可以得出 $F_0\cup F_2=F$,因此 $\{\boldsymbol{t}_0,\boldsymbol{t}_2\}$ 构成 1 个 MCTPS,同理,由于 $F_0\cup F_1\cup F_3=F$,且 $\{\boldsymbol{t}_0,\boldsymbol{t}_1,\boldsymbol{t}_3\}$ 的任意 1 个真子集都不是 TPS,因此 $\{\boldsymbol{t}_0,\boldsymbol{t}_1,\boldsymbol{t}_3\}$ 也是 1 个 MCTPS.

Table 1 An Example of *PFM*

表 1 *PFM* 实例

Pattern	Faults				
	f_0	f_1	f_2	f_3	f_4
$\boldsymbol{t}_0$	1	0	0	1	1
$\boldsymbol{t}_1$	0	1	0	0	1
$\boldsymbol{t}_2$	0	1	1	1	0
$\boldsymbol{t}_3$	0	0	1	1	0

定义 12. 向量覆盖集.若 $f_j\in F$,则称集合 T_j 是故障 f_j 的向量覆盖集,如果对于所有的 $\boldsymbol{t}_i\in T_j$,都有 $C_{ij}=1$.记为 $PCS(f_j)=\{\boldsymbol{t}_i|C_{ij}=1\}$.

例 6. 如表 1 所示,在故障 f_0 对应的 1 列数据

中,$C_{0j}=1$ 的测试向量为 $\boldsymbol{t}_0$,$0\leqslant j\leqslant 3$,因此 $PCS(f_0)=\{0\}$.同理可得,故障 f_1 对应的 $PCS(f_1)=\{1,2\}$,故障 f_2 对应的 $PCS(f_2)=\{2,3\}$,故障 f_3 对应的 $PCS(f_3)=\{0,2,3\}$,故障 f_3 对应的 $PCS(f_4)=\{1,2\}$.

3.2 算法描述

本节给出基于极小碰集算法 MHS-DMECV 的 MCTPS 求解算法 MCTPSR 的伪代码并对其进行说明.

算法 2. MCTPSR 算法.

输入:电路网表文件;

输出:电路对应的 MCTPS.

① *mkdir*();
② *source_gen*();
③ If 需要添加扫描链
④ *run_dc*();
⑤ EndIf
⑥ $T,F=$*run_tmax*();
⑦ *pats*=*pat_split*(T);
⑧ For i, ***pat***$\in$*pats*
⑨ $F'=$*run_tmax_single*(*pats*);
⑩ $\boldsymbol{v}=$*vector_gen*(F',F,T);
⑪ add $\boldsymbol{v}$ to ***PFM***;
⑫ EndFor
⑬ *PCSs*=*get_pcs*(***PFM***);
⑭ *results*=*get_hitting_set*(*PCSs*);
⑮ For *result*$\in$*results*
⑯ *get_coverage*(*result*);
⑰ EndFor

行①*mkdir*()用于生成电路目录及子目录,行②中 *source_gen*()用于生成电路所需的脚本文件;如果当前电路是时序逻辑电路文件,则需要添加扫描链以生成综合网表文件与协议文件(其中规定了扫描链与时钟等约束信息)(行③~⑤);行⑥~⑦首先通过运行 TMAX 2018 生成 TPS T 以及故障序列 F,并且将 T 中的测试向量分离为单独的测试向量 ***pat***;之后,每一个测试向量 ***pat*** 作为外部测试向量读入 TMAX 2018 可以得到单个测试向量 ***pat*** 对应的故障列表 F' 并与 F,T 一起作为 *vector_gen*() 的输入,从而得到 ***PFM*** 中的 1 行向量 $\boldsymbol{v}$,同时更新 ***PFM***(行⑧~⑫);行⑬表示根据 ***PFM*** 生成向量覆盖集 *PCS* 组成的集合 *PCSs*,作为极小碰集求解算法的输入;行⑭调用 MHS-DMECV 算法,产生对应的极小碰集,即电路的 MCTPS *results*;行⑮~⑰中 *get_coverage*() 实际上是结果检验的过程,对 *results* 中的所有结果,形成新的 TPS,载入 TMAX 2018 检测 TPS 的故障覆盖率是否保持不变.

4 实 验

本节对 MCTPSR 算法的性能进行实验分析,在实验对比部分,选取 TMAX 2018 产生的 TPS 作为比较对象.同时在初始潜在解的覆盖率最高的情况下对 PSO-Td 算法进行了重现,并与其实验结果进行对比.实验环境为:Ubuntu 14.04,CPU Intel Core i7-4700HQ 2.40 GHz,8 GB RAM,python.使用 Design Compiler 完成添加扫描链的工作,TMAX 2018 生成 TPS 以及新的 MCTPS 的覆盖率检验.

4.1 故障模型

利用 TMAX 2018 产生测试集时,所使用的故障模型是固定型故障(stuck-at fault, SAF),SAF 的优点有:

1) 易于生成故障列表,且故障总数随着电路中的逻辑门的个数线性增长,故障列表的规模可以控制;

2) 可以覆盖所有 SAF 的测试集,对于芯片中的其他类型的故障也有很高的覆盖率;

3) 很多故障模型都可以用不同的 SAF 的组合表示.

SAF 针对芯片中的单个故障,而实际芯片一般包含多个故障,已有实验表明,可以覆盖所有单故障的测试集在检测多故障时同样拥有很高的覆盖率[15].因此,本文实验选择 SAF 作为 ATPG 过程的故障模型.

4.2 实验结果

本文使用了国际测试界 ATPG 研发经常使用的基准电路 ISCAS85[16]、全扫描版本的 ISCAS89[17] 电路和 ITC99[18] 电路,其电路规模如表 2 所示.

表 3~5 分别描述了对 ISCAS85,ISCAS89,ITC99 中的部分电路对应的 TPS 的约简效果.表格中的列 1(Circuit)是各电路的名字;列 2(TMAX)表示电路对应的 TMAX 2018 生成的 TPS 中的测试向量数量;列 3(MCTPSR)表示通过 MCTPSR 算法得到的 MCTPS 中极小 TPS 中包含的测试向量的数量;列 4(Reduce)表示 MCTPSR 算法约简掉的测试向量的个数;列 5(Reduced Ratio)表示约简的部分在 TPS 中所占的比例.

Table 2 Basic Information of Test Circuits

表 2 测试电路基本信息

Circuit	Fault Number	Circuit	Fault Number
c17	14	s820	723
c432	86	s832	742
c499	146	s838	994
c880	165	s1196	1 327
c1355	146	s1238	1 374
c1908	116	s1423	2 081
c2670	589	s1488	1 423
c3540	144	s1494	1 429
c5315	596	s5378	4 283
c6288	128	s9234	3 650
c7552	613	s13207	5 930
s27	64	s15850	2 643
s208	221	s35932	34 600
s298	396	b01	177
s344	464	b02	129
s349	448	b03	778
s382	538	b04	2 346
s386	352	b05	1 848
s400	535	b06	275
s420	502	b07	1 617
s444	517	b08	628
s510	604	b09	664
s526	587	b10	672
s526n	581	b11	1 991
s641	565	b12	4 324
s713	573	b13	1 299

Table 3 Comparison in TPS of ISCAS85

表 3 ISCAS85 电路 TPS 对比

Circuit	TMAX	MCTPSR	Reduce	Reduced Ratio
c17	7	6	1	0.14
c432	26	15	11	0.42
c499	20	10	10	0.50
c880	24	14	10	0.42
c1355	15	10	5	0.33
c1908	18	9	9	0.50
c2670	46	32	14	0.30
c3540	30	14	16	0.53
c5315	39	17	22	0.56
c6288	7	5	2	0.29
c7552	58	32	26	0.41

Table 4 Comparison in TPS of ISCAS89

表 4 ISCAS89 电路 TPS 对比

Circuit	TMAX	MCTPSR	Reduce	Reduced Ratio
s27	11	7	4	0.36
s208	32	27	5	0.16
s298	42	30	12	0.29
s344	37	25	12	0.32
s349	33	25	8	0.24
s382	47	33	14	0.30
s386	64	52	12	0.19
s400	44	34	10	0.23
s420	86	69	17	0.20
s444	39	31	8	0.21
s510	83	72	11	0.13
s526	53	42	11	0.21
s526n	52	37	15	0.29
s641	54	28	26	0.48
s713	56	43	13	0.23
s820	103	79	24	0.23
s832	115	89	26	0.23
s838	187	157	30	0.16
s1196	221	163	58	0.26
s1238	224	180	44	0.20
s1423	123	82	41	0.33
s1488	138	108	30	0.22
s1494	140	112	28	0.20
s5378	285	222	63	0.22
s9234	193	135	58	0.30
s13207	218	165	53	0.24
s15850	118	93	25	0.21
s35932	58	47	11	0.19

Table 5 Comparison in TPS of ITC99

表 5 ITC99 电路 TPS 对比

Circuit	TMAX	MCTPSR	Reduce	Reduced Ratio
b01	21	14	7	0.33
b02	16	12	4	0.25
b03	46	31	15	0.33
b04	99	77	22	0.22
b05	120	92	28	0.23
b06	20	17	3	0.15
b07	93	70	23	0.25
b08	67	50	17	0.25
b09	44	32	12	0.27
b10	63	46	17	0.27
b11	181	143	38	0.21
b12	218	170	48	0.22
b13	67	44	23	0.34

由表3～5可以看出,针对规模不同、类型不同(ISCAS85是组合逻辑电路,ISCAS89和ITC99是时序逻辑电路)的电路对应的TPS,MCTPSR算法都起到了很好的约简效果.对所有电路的TPS约简比例都在10%以上,针对电路c3540和c5315甚至产生了高于50%的约简效果.

表6中对本文提出的MCTPSR算法与PSO-Td算法结果进行对比,表6中Circuit列表示电路名称,MCTPSR列表示本文所提算法的约简结果中的最小解,PSO-Td列表示PSO-Td中提出算法的约简结果(由于PSO-Td算法的实验结果具有随机性,实验中取近5次实验结果中的最小值进行比较).PSO-Td算法应用于规模较大的电路时,不能在可接受的范围内求出解,表6中用空白表示.可以看出MCTPSR算法明显优于PSO-Td算法.

Table 6 Compare Reduced Results Produced by MCTPSR with Results of PSO-Td

表6 MCTPSR与PSO-Td结果约简对比

Circuit	MCTPSR	PSO-Td	Circuit	MCTPSR	PSO-Td
c17	6	6	c432	15	17
c499	10	12	c880	14	15
c1355	10	10	c1908	9	11
c2670	32	32	c3540	14	15
c5315	17	22	c6288	5	5
c7552	32	37			
s27	7	8	s208	27	28
s298	30	33	s344	25	28
s349	25	30	s382	33	38
s386	52	56	s400	34	38
s420	69	71	s444	31	35
s510	72	64	s526	42	46
s526n	37	42	s641	28	44
s713	43	48	s820	79	83
s832	89	89	s838	157	163
s1196	163	173	s1238	180	194
s1423	82	99	s1488	108	114
s1494	112	121	s5378	222	242
s9234	135	160	s13207	165	182
s15850	93	100	s35932	47	

表7表示约简的比例在10%以上的电路个数,其中列1(Reduced Ratio)的0.10－0.20,0.20－0.30,0.30－0.40,0.40－0.50,＞0.50分别表示约简比10%～20%,20%～30%,30%～40%,40%～50%,50%以上.可以看出,对于纯组合逻辑ISCAS85中的电路来说,约简比例大于40%的电路数占该组11个电路的一半以上,其中约简比例在50%以上的电路个数达到4个;对于时序逻辑ISCAS89中的28个电路,约简比例在20%～30%的比例最大,有16个,其次是约简比例在10%～20%的电路,有7个;对于时序逻辑电路ITC99中的13个电路,约简比例集中在20%～40%.可以看出,MCTPSR算法对不同规模、不同类型的电路都有很好的约简效果.

Table 7 Circuit Number with Different Reduced Ratio Range

表7 不同约简范围的电路数量统计

Reduced Ratio	Number		
	ISCAS85	ISCAS89	ITC99
0.10－0.20	1	7	1
0.20－0.30	1	16	9
0.30－0.40	2	4	3
0.40－0.50	3	1	
＞0.50	4		

Table 8 Number of MCTPS Generated by MCTPSR

表8 MCTPSR算法的MCTPS个数

Circuit	Pattern Number	Circuit	Pattern Number
b01	2	b13	32
b02	4	c17	2
b03	54	c432	3
b04	12	c499	2
b05	48	c880	4
b06	4	c1355	6
b07	96	c1908	17
b08	48	c2670	3
b09	9	c3540	128
b10	4	c5315	1312
b11	1728	c6288	1
b12	512	c7552	24
s27	1	s713	8
s208	2	s820	4
s298	12	s832	4
s344	6	s838	6
s349	2	s1196	128
s382	4	s1238	24
s386	1	s1423	24
s400	7	s1488	2
s420	1	s1494	32
s444	2	s5378	960
s510	2	s9234	3456
s526	8	s13207	8
s526n	12	s15850	2
s641	2	s35932	36

表8中给出由MCTPSR算法得出的MCTPS的个数.可以看出,对于不同的电路,往往有多个MCTPS,MCTPSR算法可以得到所有MCTPS,这为实际的芯片测试过程提供了很强的可选择性,可以通过计算不同测试集的测试开销,选择成本合适的测试集,可以很大程度地降低芯片的测试成本.

实验结果表明:MCTPSR算法对TMAX 2018产生的TPS有很好的约简效果,而且可以得到不止1个MCTPS,为芯片测试提供了多种选择.在实际应用中,可以极大地降低芯片测试成本,加快芯片的生产过程.

5 总 结

本文通过对MCTPS约简问题进行重新建模,将复杂的约简问题转化为极小碰集的求解问题,结合基于动态极大元素覆盖值求取所有极小碰集的MHS-DMECV算法提出MCTPSR算法,对商用工具TMAX 2018产生的TPS进行约简,缩减了工具生成的TPS规模,计算效率较高,约简产生的开销比较小;其次,该算法的模型简单,只需要测试向量与故障的对应关系矩阵就可以进行求解,且过程中不需要其他参数,不需要进行多次迭代就可以得到解;该算法对于1个TPS,往往可以得到不止1个极小解,可以根据实际的测试开销选择合适的解;同时,TPS的约简不会降低故障覆盖率.MCTPSR可以在保证故障覆盖率的前提下提供更小的TPS,对降低芯片生产过程中的测试开销有着重要的现实意义.

参 考 文 献

[1] Wang Laung-Temg, Wu Chengwen, Wen Xiaoqing. VLSI Test Principles and Architectures: Design for Testability [M]. San Francisco: Morgan Kaufmann, 2006: 58-192

[2] Higami Y, Kaijihara S, Ichihara H, et al. Test cost reduction for logic circuits: Reduction of test data volume and test application time [J]. Systems and Computers, 2005, 36(6): 69-83

[3] Lü Xiaofeng, Zhou Deyun, Tang Yongchuan, et al. An improved test selection optimization model based on fault ambiguity group isolation and chaotic discrete PSO [J]. Complexity, 2018, 2018(4): 1-10

[4] Deng Sen, Jing Bo, Zhou Hongliang. Heuristic particle swarm optimization approach for test point selection with imperfect test [J]. Intelligent Manufacturing, 2017, 28(1): 37-50

[5] Yu Longjiang, Peng Xiyuan, Peng Yu. A test set optimization method based on ant algorithm [J]. Acta Electronica Sinica, 2003, 31(8): 1178-1181 (in Chinese)
(俞龙江, 彭喜源, 彭宇. 基于蚁群算法的测试集优化[J]. 电子学报, 2003, 31(8): 1178-1181)

[6] Qiao Jiaqing, Fu Ping, Yin Hongtao. Test set optimization based on genetic reordering [J]. Acta Electronica Sinica, 2007, 35(12): 2335-2338 (in Chinese)
(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)

[7] Hou Yanli, Zhao Chunhui, Hu Jiawei. 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)
(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)

[8] Cao Yiqin, Wei Jiaolong, Chen Liang. A test set optimization approach based on rough set [J]. Experimental Technology and Management, 2009, 26(6): 45-48 (in Chinese)
(曹义亲, 魏蛟龙, 陈亮. 基于粗糙集的测试集优化[J]. 实验技术与管理, 2009, 26(6): 45-48)

[9] Jiang Wei, Wang Hongli, Zhang Zhongquan, 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)
(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)

[10] Wu Haifeng, Zhan Wenfa, Cheng Yifei. Test data compression scheme for fast search best rational approximate fraction [J]. Journal of System Simulation, 2018, 30(6): 2384-2389 (in Chinese)
(吴海峰, 詹文法, 程一飞. 快速查找最佳有理渐近分数的测试数据压缩方法[J]. 系统仿真学报, 2018, 30(6): 2384-2389)

[11] Yuan Haiying, Guo Kun, Sun Xun, et al. A power efficient test data compression method for SoC using alternating statistical run-length coding [J]. Electronic Testing, 2016, 32(1): 59-68

[12] Deng Zhaoyong, Ouyang Dantong, Geng Xuena, et al. Algorithm of computing minimal hitting set based on the structural feature of SE-Tree [J]. Journal of Computer Research and Development, 2018, 55(4): 791-801 (in Chinese)
(邓召勇, 欧阳丹彤, 耿雪娜, 等. 基于动态极大元素覆盖值的极小碰集求解算法[J]. 计算机研究与发展, 2018, 55(4): 791-801)

[13] Liu Siguang, Ouyang Dantong, Wang Yiyuan, et al. Algorithm of computing minimal hitting set based on the structural feature of SE-Tree [J]. Journal of Computer Research and Development, 2016, 53(11): 2556-2566 (in Chinese)

(刘思光,欧阳丹彤,王艺源,等.结合 SE-Tree 结构特征的极小碰集求解算法[J].计算机研究与发展,2016,53(11):2556-2566)

[14] Reiter R. A theory of diagnosis from first principles [J]. Artificial Intelligence, 1987, 32(1): 57-96

[15] Bushnell M, Agrawal V. Essentials of Electronic Testing for Digital, Memory and Mixed-signal VLSI Circuits [M]. Berlin: Springer, 2013

[16] Brglez F, Fujiwara H. A neutral netlist of 10 combinational benchmark circuits and a target translator in Fortran [C] // Proc of IEEE Int Symp on Circuits and Systems. Piscataway, NJ: IEEE, 1985: 695-698

[17] Brglez F, Bryan D, Kozminski K. Combinational profiles of sequential benchmark circuits [C] //Proc of IEEE Int Symp on Circuits and Systems. Piscataway, NJ: IEEE, 1989: 1929-1934

[18] 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

Ouyang Dantong, born in 1968. Professor and PhD supervisor of Jilin University. Senior member of CCF. Her main research interests include model-based diagnosis, model checking and automated reasoning.

Chen Xiaoyan, born in 1995. Master candidate at Jilin University. Her main research interest is model-based diagnosis.

Ye Jing, born in 1988. PhD. Associate professor at University of Chinese Academy of Sciences. His main research interests include hardware security, AI security, VLSI test and diagnosis.

Deng Zhaoyong, born in 1994. Master candidate at Jilin University, His main research interest is model-based diagnosis.

Zhang Liming, born in 1980. PhD. Senior engineer of Jilin University. His main research interests include model-based diagnosis, model checking and Boolean satisfiability.