PCBCover:面向电路图自动化生成的覆盖率引导强化学习框架

赵 旭[1] 邹沛煜[2] 李晓晨[3] 刘璐恺[3] 刘 辉[1]
施重阳[1] 江 贺[3]

[1](北京理工大学计算机学院 北京 100081)
[2](辽宁师范大学计算机与人工智能学院 辽宁 大连 116000)
[3](大连理工大学软件学院 辽宁 大连 116024)

摘 要 印刷电路板(printed circuit board,PCB)设计工具链的缺陷可能导致PCB功能错误。若有缺陷的PCB应用于航空、医疗等安全攸关领域,将引发严重后果。因此,保障PCB设计工具链的可靠性至关重要。现有测试方法主要依赖手工设计测试用例或基于固定配置的随机生成策略,缺乏对高质量测试样本生成的有效引导,导致缺陷触发能力有限。解决现有方法在触发缺陷能力上的不足,主要面临两大挑战,即如何评估参数配置有效性和在庞大参数空间中高效搜索最优配置。为此,本文提出一种自动化测试方法PCBCover。该方法结合覆盖率反馈与基于A2C的强化学习算法,实现对电路图的自动生成,有效提升电路图(测试用例)的缺陷触发能力。PCBCover包含三个核心组件:覆盖率提取模块、基于A2C的动态配置搜索模块和基于差分测试的缺陷检测模块。为量化配置参数的有效性,PCBCover设计了静态(网表)与动态(仿真)两类覆盖率指标,并将其作为奖励信号引导生成器搜索高触发缺陷的电路配置。实验结果表明,PCBCover在多个PCB设计工具(如Ngspice与KiCad)上的缺陷检测能力显著优于现有最新方法,并成功发现13个被开发者确认的真实缺陷,验证了其检测能力和实际应用价值。

关键词 PCB设计工具链;自动化测试;缺陷检测;覆盖率;强化学习
中图法分类号 TP311 **DOI号** 10.11897/SP.J.1016.2025.02893

PCBCover: A Coverage-Guided Reinforcement Learning Framework for Automated Schematic Generation

ZHAO Xu[1] ZOU Pei-Yu[2] LI Xiao-Chen[3] LIU Lu-Kai[3] LIU Hui[1]
SHI Chong-Yang[1] JIANG He[3]

[1](School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081)
[2](School of Computer Science and Artificial Intelligence, Liaoning Normal University, Dalian, Liaoning 116000)
[3](School of Software, Dalian University of Technology, Dalian, Liaoning 116024)

Abstract Defects in printed circuit board (PCB) design tool chain can cause functional errors in the resulting PCBs. If these faulty PCBs are used in safety-critical domains such as aerospace or healthcare, the consequences can be severe. Therefore, it is essential to ensure the reliability of PCB design tool chain. However, current testing methods often rely on manually created test

收稿日期:2025-05-17;在线发布日期:2025-10-17。本课题得到国家自然科学基金(62032004、62572090、62202079)资助。**赵 旭**,博士研究生,主要研究领域为智能软件工程、软件测试和数据挖掘等。E-mail:zhaoxubit@163.com。**邹沛煜**,博士,讲师,主要研究领域为智能软件工程和EDA(电子设计自动化)工业软件。E-mail:zoupeiyu985@163.com。**李晓晨**,博士,副教授,中国计算机学会(CCF)会员,主要研究领域为智能软件工程和软件测试。**刘璐恺**,本科生,主要研究领域为软件测试。**刘 辉**,博士,教授,中国计算机学会(CCF)会员,主要研究领域为基于深度学习的软件工程、软件重构和软件质量。**施重阳**,博士,教授,主要研究领域为信息检索、知识工程和情感分析。**江 贺**(通信作者),博士,教授,中国计算机学会(CCF)会员,主要研究领域为智能软件工程、软件测试和基于搜索的软件工程。E-mail:jianghe@dlut.edu.cn。

cases (e. g. , schematics) or randomly generated schematics from schematic generators with fixed configurations. These methods lack effective guidance for generating high-quality test inputs and have limited capability in triggering hidden defects. To improve the effectiveness of schematic generation, two key challenges must be addressed: designing suitable metrics to evaluate parameter configurations, and efficiently searching the large configuration space to find combinations that can trigger defects. To address these challenges, this paper proposes PCBCover, an automated testing approach designed to enhance defect detection in PCB design tool chain. PCBCover combines coverage feedback with an A2C-based reinforcement learning algorithm to guide the generation of schematics that are more likely to trigger defects. PCBCover includes three main components: a coverage extraction module, an A2C-based dynamic configuration search module, and a differential testing module for detecting defects. It defines both static (i. e. , netlist level) and dynamic (i. e. , simulation level) coverage metrics to evaluate the effectiveness of configurations and uses them as reward signals during the search process. Experiments on real-world tools such as Ngspice and KiCad show that PCBCover significantly outperforms existing methods in detecting defects. It successfully detects 13 real bugs confirmed by tool developers, demonstrating strong detection capability and practical value.

Keywords PCB design tool chain; automated testing; defect detection; coverage; reinforcement learning

1 引 言

印刷电路板(PCB)是一种将电子元件通过导电图形连接并固定于其上的电路承载平台,既提供机械支撑,又实现电气连接功能[1]。PCB设计工具链通过提供电路图绘制、智能布线优化、设计规则检查(DRC)、电路仿真等功能,减少了设计者的手动操作,降低了出错率[2]。与此同时,它还支持设计复用、版本管理和协同工作,提升了设计效率和产品可靠性。因此,PCB设计工具链的出现减轻了PCB设计者的负担,提高了PCB设计效率[3]。由于电子产品的快速发展,PCB在设计中的应用变得更加广泛,所以必须确保PCB设计工具链的健壮性。

尽管PCB设计工具功能日趋完善,但其内部仍可能存在潜在的软件缺陷[4]。某些工具中的缺陷表现为仿真错误、不一致行为或信号处理异常,往往不会以程序崩溃的形式直接暴露,而是潜藏在电路运行逻辑之中。例如,图展示了在Ngspice仿真器中曾发现的一项关键缺陷(#649):在使用仿真工具对晶体管(PMOS)进行AC小信号仿真时,发现仿真得到的栅输入阻抗(Zin)对应的等效电容与工作点分析中提取的总微分电容(cgg)相差极大,偏差最高达97%。图1展示了不同偏置条件下的仿真电路图。图中每个电路图下面的数值是其仿真结果。理论上这些仿真结果应该一致,但是实际仿真结果是不一致的。这将会误导PCB设计者对晶体管的使用,从而导致最终设计的PCB不正确。此类缺陷具有高度隐蔽性和误导性,PCB设计者往往难以在开发阶段察觉,并将故障误认为自身设计不当。更严重的是,若此类缺陷未被及时发现并最终部署到真实系统中,如汽车电子、医疗仪器、航空控制等领域,可能导致功能失效、系统瘫痪甚至严重的安全事故。因此,识别与预防PCB设计工具中的潜在缺陷,是保障系统安全性的关键环节,尤其对于关键任务电子系统,其风险容忍度极低,更加凸显PCB设计工具可靠性的重要性。

自动化测试作为确保软件质量的常用方法,通常通过运行大量测试用例来验证软件的正确性[5-6];因此,利用生成器自动生成测试用例不仅是自动化测试流程的关键步骤,其性能更直接关系到整个测试过程的效率和效果。在PCB设计工具测试技术研究领域,已开发出诸如PCBSmith[7]等用于生成电路图(测试用例)的工具。该工具通过一组配置选项控制所生成电路图的特征,已随机生成电路图。例如,PCBSmith的配置选项包括元器件种类、数量、模型以及电气参数等,从而确保生成的测试用例能够覆盖不同类型的电路特性。

电路图生成器的一个重要目标是生成能触发缺陷的电路图,而缺陷检测的有效性极大依赖于生

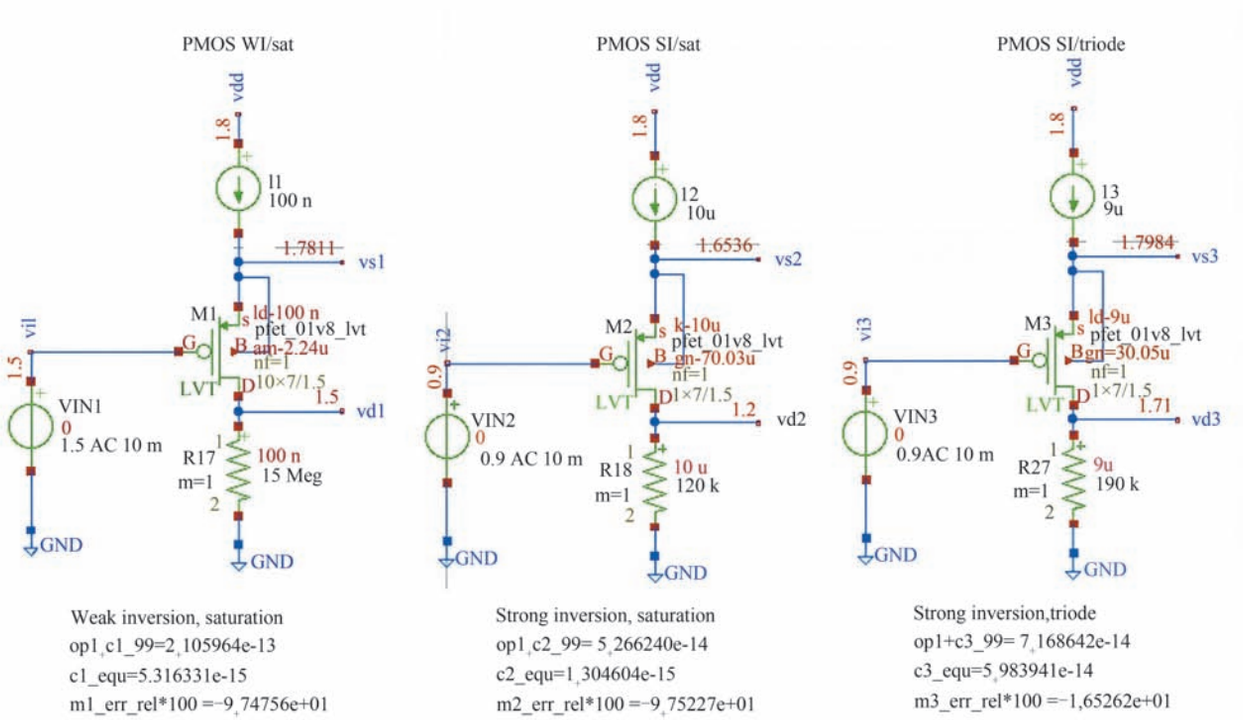


图1　缺陷#649不同参数下的仿真电路图

成器配置参数的合理性[8-10]。目前尚不清楚哪些配置参数组合最能有效触发缺陷。现有电路图生成器多依赖默认配置，难以生成高缺陷触发率的电路图，限制了检测效果。因此，亟需一种快速且有效的方法，以自动化地发现能够显著提升缺陷触发能力的参数配置。为此，现有方法面临两个挑战：第一，如何设计评估指标来量化配置参数的有效性？需要定义既能反映电路图覆盖率，又能直接体现缺陷检测效果的指标，从而判断一个配置组合是否具有较高的缺陷触发能力；第二，如何高效探索庞大且复杂的配置参数空间以识别最优参数组合？由于配置参数较多，且每个参数的取值范围较大，从而形成了一个庞大且复杂的搜索空间。为了解决这一问题，亟需设计一种动态自适应的搜索机制。该机制能够利用实时反馈信息，有效缩小搜索范围，从而快速识别出那些能显著提升缺陷触发概率的最优参数组合。这些挑战的解决将大大增强生成器的故障检测能力。

针对上述挑战，本文提出了一种名为PCBCover的自动化测试方法，旨在提升对PCB设计工具链中潜在缺陷的检测能力。该方法主要由三部分组成：覆盖率提取模块、配置动态搜索模块和缺陷检测模块。覆盖率提取模块提出了一套适用于电路网表的静态（网表）和动态（仿真）覆盖率指标，用于衡量电路结构和仿真行为的覆盖情况。结构覆盖可以反映电路元件的完整性和连接特征，而仿真覆盖则揭示了实际运行过程中信号是否覆盖了关键节点与路径，从而直接关联到缺陷被触发的可能性。因此，它解决了如何量化配置参数有效性的挑战；配置动态搜索模块基于A2C（Advantage Actor-Critic）强化学习框架，动态探索最优电路图配置，以生成高质量的测试用例。A2C结合了策略优化与价值估计，具备在高维、连续、复杂的动作空间中实现快速搜索的能力，能够充分利用覆盖率和仿真反馈，有效引导PCBCover在庞大配置参数空间中进行高效搜索。因此，它解决了如何高效探索庞大且复杂的配置空间以识别最优参数组合的挑战；缺陷检测模块基于差分测试方法，通过对比不同仿真模式或仿真工具的输出结果，有效识别PCB设计工具链中的潜在异常行为和缺陷。PCBCover融合覆盖率反馈与智能搜索策略，实现了高效、自动化的PCB工具链测试流程。

通过实验验证，PCBCover能够有效发现PCB设计工具链中的潜在缺陷。本文在Ngspice与KiCAD上对PCBCover进行了系统评估。本文重点分析了A2C算法中奖励函数的关键参数对缺陷检测性能的影响。该参数用于平衡覆盖率与仿真结果（如失败或超时）在奖励中的权重，从而引导生成

的测试用例在结构多样性与故障触发能力之间取得平衡。实验表明，当该系数设置为0.6时，PCBCover检测到最多的非重复不一致(17个)，表现优于其他取值。这说明在覆盖率与仿真结果之间实现适度平衡有助于提升测试效果。此外，PCBCover提出网表覆盖率与仿真覆盖率两类指标，作为奖励函数以引导测试用例生成。实验进一步分析了这两类覆盖率在分别作为奖励信号时，对PCBCover检测仿真不一致行为的影响。实验发现仿真覆盖率略优于网表覆盖率(15 vs. 13)，而两者联合使用时PCBCover表现最优(17)，表明静态结构信息与动态行为反馈的融合能更全面地提升缺陷触发能力。

与现有配置多样性方法(如Default(基于PCBSmith的默认策略)、RECORD(基于单智能体强化学习)、HIS(基于历史缺陷信息引导)和Swarm(基于粒子群随机搜索))相比，PCBCover在相同测试周期内生成网表最少(126,336个)，却检测到最多的非重复不一致(17个)，显著优于RECORD(12)、HIS(10)、Swarm(5)和Default(3)，体现了更高的单位测试效率。在真实缺陷检测方面，PCBCover共发现13个被开发者确认的缺陷，其中包括11个性能问题与2个仿真不一致问题。综上，PCBCover在检测能力、测试效率及实际适用性方面均优于现有方法，为PCB设计工具链的自动化测试提供了有效支持。

本文做出了如下贡献：

(1) 本文提出PCBCover，一种用于检测PCB设计工具链缺陷的自动化测试方法。该方法基于覆盖率驱动的强化学习框架，能够自动生成具有较高缺陷触发潜力的电路图，从而提升测试用例的缺陷检测能力。

(2) 本文通过在多个PCB设计工具(包括Ngspice和KiCAD)上的系统实验证明，PCBCover在缺陷检测能力上优于现有最先进的方法。在测试效率和准确性方面均表现出更强性能。

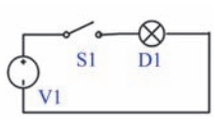
(3) 为促进后续研究的开展，本文已将PCBCover以开源形式发布，作为面向PCB设计工具链的测试研究平台，供社区使用和扩展。

本文第2节阐述了研究背景与动机。第3节详细介绍了PCBCover的设计与实现。第4节给出了实验设计与结果分析。第5节回顾了相关研究工作。最后在第6节对全文进行了总结。

2 研究背景与动机

2.1 PCB测试用例

在PCB设计过程中，电路图(Schematic)是PCB设计者通过图形化界面构建的原理图，用于表达电子元器件之间的连接关系[11]。如图2(a)所示，展示了一个简单的开关电路，包含电源(V1)、开关(S1)以及发光二极管(D1)。为了支持后续的仿真验证与制造流程，该图形化描述会被转换为对应的网表(Netlist)。网表是一种面向机器处理的文本格式，能够精确记录电路中所有元器件的类型、参数及其电气连接关系[12-13]。如图1(b)所示，展示的是上述电路图所生成的网表文件。通常，网表作为仿真分析和自动化测试的输入，是PCB设计工具测试中最基本的测试用例形式[7]。



(a) 电路图(schematic)图例

```
1 .title KiCad schematic
2 .include "D:\spice_lib\Models\Diode\led.lib"
3 V1 Net-_S1-Pad1_ Net-_D1-Pad2_ 2
4 S1 Net-_S1-Pad1_ Net-_S1-Pad2_ SW_Push
5 D1 Net-_S1-Pad2_ Net-_D1-Pad2_ LED_GENERAL
6 . end
```

(b) 网表(netlist)文件

图2 PCB测试用例

网表由多个元器件实例及其连接信息构成[14]。每个元器件包括唯一标识、器件类型(如电阻、电容、二极管)、参数值(如电阻值、电压等)以及其连接的节点。如图2(b)所示，第3行表示一个名称为V1的电压源，其正极连接到节点Net-_S1-Pad1_，负极连接到Net-_D1-Pad2_，电压值为20 V。这条记录完整地定义了电压源的行为和在电路中的连接位置。网表的结构化表示使其易于处理、分析和用于自动化测试，是设计验证流程中的关键中间产物。

网表可被送入仿真器(如Ngspice)进行仿真，以评估电路在各种输入激励下的行为表现[15]。仿真器根据元器件模型与电气定律求解电路响应，例如直流偏置、瞬态变化、电压波形等[16]。仿真结果通常包括各个节点的电压、电流随时间的变化，或某些特定

元件的输出响应[17]。电路仿真类似于对程序的“运行”，体现了电路设计在实际运行条件下的动态特性，是发现潜在错误和不一致行为的重要手段。仿真结果将会存储在仿真结果文件和日志文件中。

在PCB设计工具链的测试中，网表作为测试用例的核心载体，其质量直接决定了缺陷触发能力。通过构造结构多样且电气行为丰富的网表，可以更有效地触发潜在缺陷。

2.2　动　机

在各类软件系统中，测试用例生成器是关键的测试工具。大量研究致力于构建具备高度配置能力的生成器，以满足不同测试需求。例如，Csmith[18]是广泛用于编译器测试的C程序生成器，可通过71个配置选项控制程序特性；SLforge[19]可依据Simulink规范生成有效的信息物理系统（CPS）模型，避免代数环等常见问题；Verismith[20]用于生成确定性Verilog[21]程序，通过配置参数控制语法结构与嵌套深度，从而测试FPGA综合工具。这些工具均通过一系列配置决策，生成结构合法、特性多样的测试用例。

为了测试PCB设计工具链，测试用例生成器同样是必不可少的。然而，PCB设计工具链在自动生成电路图方面面临独特的挑战。首先，需要确保自动生成的电路图能真实反映实际电路的特征；其次，要保证这些电路图符合设计规则和约束条件。目前，诸如PCBSmith[7]这类先进的电路图生成工具已在解决这两个挑战上取得了一定进展。在读取配置文件后，电路图生成器会模仿PCB设计者创建电路图的步骤。首先，它从一个全面的元件库中选取合适的电子元件；接着，根据符号约束和连接约束将这些元件连接起来；随后，电路图生成器为每个元件设定电气参数，并自动确定每个电路图的仿真模型和仿真参数；最后，它生成一个可供仿真的电路图，并将其转换为网表以便进行仿真分析。

尽管现有电路图生成器在测试用例生成方面取得了一定进展，但由于其依赖于随机或默认的参数配置，生成的电路图在触发潜在缺陷方面的能力仍存在局限。本文旨在通过引入策略引导机制，对生成器的配置参数进行动态优化，从而生成更具缺陷触发潜力的电路图，提升对PCB设计工具链的测试效果。因此，当前的关键问题在于如何高效地搜索并识别具备较高缺陷检测能力的参数配置组合。为此，亟需解决两个核心挑战。

挑战1：配置参数评估指标的设计。如何设计评估指标以量化电路图生成配置的有效性，是电路图生成器优化中的关键问题。理想的指标应既能反映电路图的覆盖率，又能体现其缺陷检测效果，从而判断某一配置组合是否具有较高的缺陷触发概率。通过合理的指标设计，有助于提升生成配置的判别效率与测试效能。然而，通过对现有覆盖率相关研究的分析可知，电路图相较于传统测试对象（如代码）具有独特性：其不仅包含元器件的结构连接，还需考虑电气特性与信号传输行为。因此，有必要针对电路图的特性，设计适用的覆盖率指标体系。

挑战2：配置参数的搜索空间庞大。挑战旨在构建一个机制，使得在不断生成电路图的过程中，能够自适应地搜索和选择那些能有效触发缺陷的配置参数组合。理想的方案需要利用实时反馈，对当前测试环境中生成的电路图覆盖情况和缺陷检测效果进行评估，并根据这些评估结果动态调整配置参数。机制应能快速响应测试数据的变化，并在有限的资源下高效探索配置空间，以便发现更具缺陷触发潜力的参数组合。其他配置搜索的工作，它们各有不足。这些工作的方法要么依赖静态规则，难以适应动态测试环境，要么搜索策略过于随机，导致高效缺陷触发配置的探索不足。因此，面对PCB设计工具链测试环境的复杂性，为了检测出更多的缺陷，需要一种更为有效的方法。

3　方法介绍

3.1　方法概述

本章将详细介绍PCBCover的实现细节。如图3所示，PCBCover由三个核心模块构成：配置动态搜索模块、覆盖率提取模块和缺陷检测模块。首先，如图3所示，系统随机初始化配置参数，形成配置文件。配置动态搜索模块利用A2C（Advantage Actor-Critic）算法动态设定配置参数，生成网表。该模块并对网表进行仿真，生成日志文件和仿真结果文件。A2C基于覆盖率提取模块和覆盖率提取模块提供的反馈信息计算奖励函数，用于指导下一轮的生成过程。随后，A2C模型不断迭代，动态调整配置策略，最终搜索到能够有效触发缺陷的最优参数组合。覆盖率提取模块负责提取静态与动态覆盖率指标，作为奖励函数的重要组成部分；缺陷检测模块通过统计网表仿真过程中的输出不一致与仿真超时信息，进一步补充奖励函数信号。

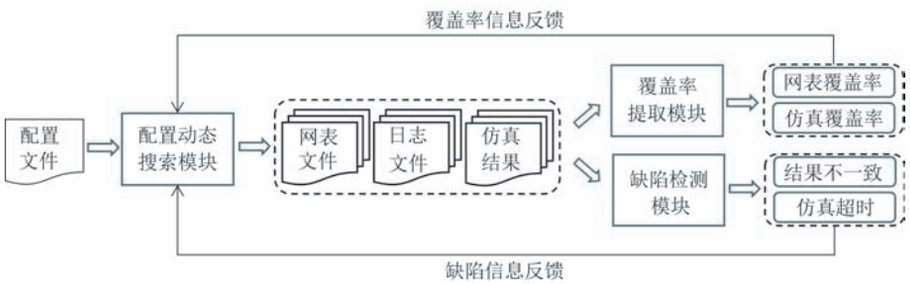


图3 PCBCover方法框架

算法1. PCBCover算法伪代码

输入：*NetlistConfigSpace*（电路图的初始配置参数空间）

输出：*config*（能有效触发缺陷的电路图配置）

```
1:  Function PCBCover_ConfigSearch(NetlistConfigSpace)
2:      Initialize ActorNetwork πθ and CriticNetwork Vω
3:      Initialize config ← RandomInit(NetlistConfigSpace)
4:      for episode = 1 to MaxEpisodes do
5:          for t = 1 to MaxStepsPerEpisode do
6:              Netlist ← GenerateNetlist(config)
7:              StaticCov ← ComputeStaticCoverage
                (Netlist)
8:              SimResult ← RunSimulation(Netlist)
9:              DynamicCov ← ComputeDynamic-
                Coverage(SimResult)
10:             Reward ←θ* (StaticCov + Dynam-
                icCov) + (1 -θ) * SimError(SimRe-
                sult)
11:             NextConfig ← SampleAction(πθ, con-
                fig)
12:             Advantage ← Reward + γ * Vω
                (NextConfig) - Vω(config)
13:             Update CriticNetwork Vω using MSE
                loss
14:             Update ActorNetwork πθ using policy
                gradient
15:             config ← NextConfig
16:         end for
17:     end for
```

通过算法1，本文介绍PCBCover的算法实现。首先（第2行），系统初始化策略网络（Actor）与价值网络（Critic），为后续的策略决策与评估提供模型基础。在每一轮迭代中（第6行），系统根据当前策略生成配置参数，并基于该配置构造出对应的电路图网表。随后（第7至第9行），从网表结构和仿真过程结果（SimResult）中提取静态覆盖率（StaticCov）与动态覆盖率（DynamicCov），用于衡量测试用例的结构性。基于这些覆盖率指标，以及仿真是否失败或超时的结果（第10行），PCBCover计算本轮生成结果对应的奖励值（Reward），用于衡量配置组合的有效性。在强化学习主循环中（第11至第15行），智能体根据当前策略采样动作，更新网络参数，从而逐步优化策略模型，使其更倾向于选择能带来高奖励的配置组合。接下来将分别介绍三个模块的设计思路与具体实现方式。

3.2 配置动态搜索模块

3.2.1 A2C网络结构介绍

在PCB设计工具链的测试过程中，自动生成具有缺陷触发潜力的电路图对于提升缺陷检测效率具有重要意义。电路图的结构通常由元器件的种类、数量、分支拓扑以及子电路模块等多个维度构成，因此电路图的生成过程本质上是一个在高维配置参数空间中进行搜索的问题。由于缺乏关于何种参数组合更易触发缺陷的先验知识，导致该搜索问题具有高度不确定性和复杂性。因此，在有限计算资源约束下，如何高效识别潜在的高效缺陷触发配置组合，成为提升测试有效性的关键挑战之一。

为了解决上述挑战，本文设计了配置动态搜索模块。如图4所示，该模块中引入了A2C多智能体算法。A2C是一种强化学习（Reinforcement Learning）方法，用于解决复杂决策问题[22]。A2C结合了策略梯度（Policy Gradient）和价值函数（Value Function），能够在高维度、动态变化的环境中进行高效决策[23]。简单来说，A2C由两个网络组成：策略网络（Actor）负责选择动作，即根据当前状态决定如何调整电路配置；价值网络（Critic）评估当前状态的好坏，提供一个参考值，以帮助Actor更稳定地更新策略。A2C的核心思想是：（1）Actor负责试探不同的电路配置调整方式，例如，选择调整某个元器件的数量或类型；（2）Critic评估该调整的优劣，告诉Actor这个选择是否值得继续尝试；（3）Actor结合Critic的反馈，更新策略，使得智能体逐步学会更有效的电路图配置参数。

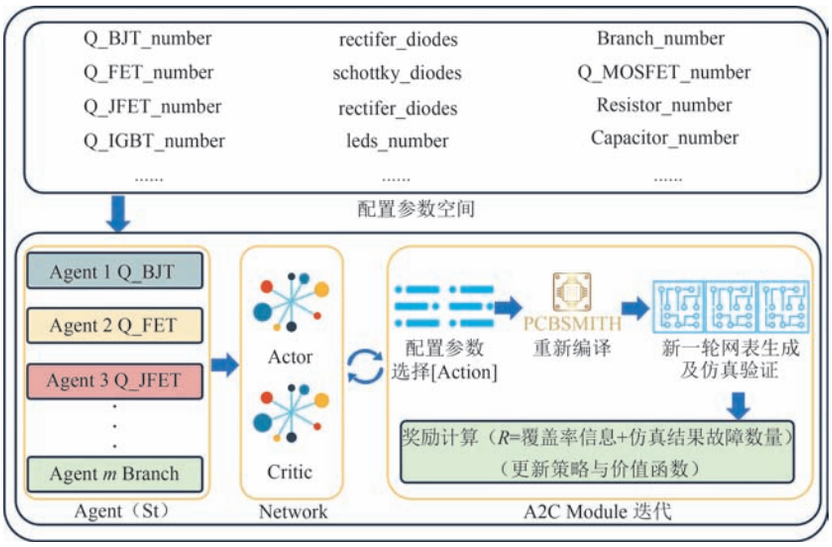


图4　配置动态搜索模块

3.2.2　实现细节

在A2C框架中，状态空间与动作空间是定义智能体行为的核心要素。状态空间定义了系统在某一时刻的所有信息，而动作空间则表示在当前状态下可执行的操作。通过不断地从状态空间中提取信息并执行动作，智能体能够逐步学习如何在复杂的环境中做出最优决策。定义状态空间S表示当前生成的电路图配置情况，其包括各类元器件的种类及其数量等信息；而动作空间A则定义了在当前状态下，智能体所能采取的改变配置参数的操作。

具体来说，每个状态s_t可以表示为一组配置参数，例如：

$$s_t=\{(component_1,num_1),(component_2,num_2),\cdots,(component_n,num_n)\}。$$

智能体的动作a_t包括调整某一元器件数量或更改元器件种类的操作。那么，一个动作可以具体表示为对某个元器件的数量或类型进行改变，例如：

$$a_t=(component_i,new\_num_i)。$$

为引导智能体在复杂的配置空间中快速找到能触发缺陷的有效组合，本模块设计了一个奖励函数R。该奖励函数综合了网表覆盖率和仿真覆盖率，并结合仿真结果（失败或超时）给予正向或负向奖励，公式如下：

$$R=\theta\cdot(NetlistCoverage+SimulationCoverage)-(1-\theta)\cdot(Faults+Timeouts),$$

其中，θ为权重参数，用以平衡各项指标的贡献。网表覆盖率和仿真覆盖率分别反映了生成电路图在静态和动态测试中的覆盖效果，而电路图仿真结果（故障和超时）则直接指示了该电路图缺陷触发的情况。

在A2C框架下，本模块交替更新策略网络（Actor）和价值网络（Critic）在每个时间步t中，智能体根据当前状态s_t由策略网络选择一个动作a_t，执行后获得即时奖励R_t并进入新状态s_{t+1}。价值网络则对状态的期望累计奖励$V(s_t)$进行估计。本模块的优化目标是最大化累计折扣奖励：

$$\max_{\omega}E\left[\sum_{t=0}^{T}\gamma^t R_t\right],$$

其中，ω是策略网络的参数，$\gamma\in[0,1]$为折扣因子，通常设置为0.99[10]。从之前工作得出[22-23]，当策略网络学习率设置为0.001时，能够有效地引导智能体在复杂的配置空间中进行策略更新，从而实现更高效的决策过程。

通过反复迭代训练，智能体能够逐步调整其配置参数选择策略，从而生成更多能够触发PCB设计工具链缺陷的电路图，该方法不仅提高测试用例的生成效率，同时降低了手动调试和验证的工作量，有助于整体提升PCB设计工具链的可靠性和鲁棒性。

3.3　覆盖率提取模块

覆盖率通常用于衡量测试过程中实际执行到的代码或程序结构的比例，以评估测试用例的充分性[24-25]。例如，代码覆盖率可以细分为语句覆盖率、分支覆盖率、函数覆盖率等，这些指标有助于分析测试是否涵盖了大部分可能的执行路径[26]。然而，网表不同于传统编程语言，例如，它描述的是电路结构和信号传输，而非控制流和数据流。PCBCover无法直接采用这些常见的覆盖率度量方式。如图5所示，本模块提出了网表覆盖率（也称为静态覆盖率）和仿真覆盖率（也称为动态覆盖率）。网表覆盖率用

于衡量网表的结构特征。仿真覆盖率用于评估电路在仿真过程中的实际执行情况。覆盖率信息将用于动态搜索模块的奖励函数中。

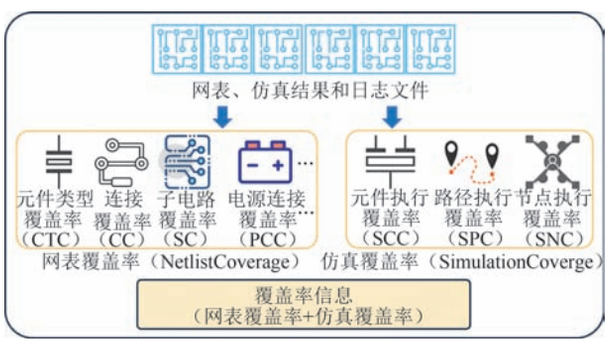


图5 覆盖率提取模块

如表1所示，PCBCover一共获取了6个静态覆盖率指标值，包括元器件类型覆盖率(CTC)、连接覆盖率(CC)、节点覆盖率(NC)、子电路覆盖率(SC)、接地连接覆盖率(GCC)和电源连接覆盖率(PCC)。例如，CTC代表电路中不同元器件类型的覆盖情况，而CC反映了网表中电气连接的完整性。通过这些静态覆盖率指标，PCBCover能够评估电路各个元素的存在情况及其连接关系。如表1所示，网表覆盖率从多个维度对网表进行评估。

接下来介绍各个覆盖率的计算方式：

(1) CTC统元器件类型数量，并计算其相对占比；

(2) CC统计网表中所有元器件的总引脚数P_{all}和已连接的引脚数量P_{CC}，并计算连接率：$CC=P_{CC}/P_{all}$该覆盖率接近1说明网表高度连通，接近0说明网表连接稀疏；

(3) NC统计网表中的总节点数N。然后，计算平均每个节点连接的引脚数量：$P_{avg}=P/N$。进一步分析单引脚节点($P=1$)、双引脚节点($P=2$)和多引脚节点($P>2$)的比例，以评估网表的信号复杂度；

(4) SC统计子电路的定义总数S，统计子电路的实例化次数I。计算平均每个子电路被调用的次数：$SC=I/C$。如果SC较低，说明网表模块化程度较低；如果SC较高，说明网表具有较多的重复模块；

(5) GCC统计网表中所有节点数N和连接到地的节点数N_G。然后，计算接地连接覆盖率：$GCC=N_G/N$。该指标可以描述网表中GND的分布情况，低值可能意味着地连接较少，高值可能意味着地连接较为广泛；

(6) PCC统计网表中所有节点数N和连接到电源的节点数N_P。然后，计算电源连接覆盖率：$PCC=N_P/N$。该指标可以描述网表的供电分布情况，低值可能意味着电源连接较少，高值可能意味着电源连接较为广泛。

然而，电路仿真不仅依赖于网表的静态结构，还受到实际信号传输和电路动态行为的影响。因此，为了更全面地评估网表在仿真执行过程中的表现，本模块进一步引入了仿真覆盖率(也称为动态覆盖率)。如表2所示，它包括仿真节点执行覆盖率(SNC)、路径覆盖率(SPC)和元件执行覆盖率(SCC)。这些指标用于衡量电路在仿真过程中实际被激活和执行的程度，从而更准确地反映电路的运行特性。

如表2所示，仿真覆盖率从多维度衡量网表仿真状态。该组件通过解析仿真结果文件和仿真日志文件获取仿真覆盖率。接下来介绍各个覆盖的计算方式：

(1) SNC统计仿真过程中实际被激活的节点数量N_{active}和网表中定义的所有节点数N_{total}。计算：$SNC=N_{active}/N_{total}$；

表1 网表覆盖率(静态覆盖率)

覆盖率名称	定 义
元器件类型覆盖率(CTC)	统计网表中不同类型的元器件(如电阻、电容、电感、晶体管等)占比，反映网表的元器件构成特征。
连接覆盖率(CC)	统计网表内引脚总数与实际连接数量的比例，反映网表的连通性特征。
节点覆盖率(NC)	统计网表中节点(信号点)的数量，以及节点与元器件的关联程度，反映网表的信号分布特征。
子电路覆盖率(SC)	统计子电路种类与实例化情况，衡量网表中子电路(模块)的使用情况，反映网表的层次化特征。
接地连接覆盖率(GCC)	统计网表中连接到地(GND)的节点比例，反映网表的接地特征。
电源连接覆盖率(PCC)	统计网表中连接到电源(VDD/VCC/VSS等)的节点比例，反映网表的供电特征。

(2) SPC统计仿真过程中被激活的信号路径数量P_{active}(即起点和终点节点对)和网表中预定义的所有可识别传输路径数量P_{total}。计算：$SPC=P_{active}/P_{total}$；

(3) SCC统计仿真过程中参与执行的元器件数量$C_{exectued}$和网表中定义的所有元器件总数C_{total}。计

表2　仿真覆盖率（动态覆盖率）

覆盖率名称	定义
仿真节点执行覆盖率(SNC)	统计在仿真过程中实际激活并执行的电路节点数量与网表中所有定义节点总数之比。该指标用于衡量仿真过程中各节点的测试有效性和触达率。
路径覆盖率(SPC)	统计在仿真过程中实际激活的信号传输路径(即节点对)数与网表中所有预定义路径数之比。该指标反映了仿真过程中测试用例对电路各个传输路径的覆盖程度。
元件执行覆盖率(SCC)	统计在仿真过程中参与执行的电路元器件数量与网表中所有元器件总数之比。该指标评估仿真测试是否充分触发了各个元器件的工作状态，从而验证其功能正确性。

算：$SCC = C_{exectued}/C_{total}$。

通过这些动态覆盖率指标，PCBCover可以全面评价仿真测试的有效性，为缺陷检测提供更精确的依据。

3.4　缺陷检测模块

如图6所示，本模块用于识别PCB设计工具链在仿真阶段可能出现的缺陷。为系统性地覆盖异常类型，我们将仿真过程中的异常划分为两大类：仿真运行异常与仿真行为异常。仿真运行异常是指仿真器无法顺利完成仿真流程，典型表现包括仿真超时（如长时间无响应）、输入网表解析失败（语法错误或不支持结构）、仿真器崩溃（进程异常终止）等。这类异常无需生成仿真输出，即可直接据此判断该测试用例可能触发了工具链在边界条件下的系统性缺陷。仿真行为异常则是指仿真器顺利完成仿真并生成输出结果，但在不同仿真配置下产生的结果存在不一致，说明仿真行为存在非确定性或工具逻辑不一致的风险。此类异常需借助差分测试（Differential Testing）[27]方法识别。

图6　缺陷检测模块

仿真行为异常采用如下两种差分策略识别仿真行为异常：第一种方法是在PCB设计工具链中，使用同一仿真工具的不同仿真模式对网表进行仿真。现有仿真工具通常支持多种仿真模式，以适配不同类型的电路描述文件。在理想情况下，同一网表在同一仿真工具的不同模式下进行仿真，若仿真配置（如元器件模型、初始条件、仿真步长等）一致，则输出结果应相同。然而，如果在某些模式下仿真结果出现不一致，则可能意味着仿真工具在不同模式之间的计算逻辑、模型解析或数值求解方面存在缺陷，从而影响PCB设计工具链的稳定性和可靠性。

形式化地，设N为网表集合，S为仿真工具，M为S的仿真模型，定义$O(n)$表示仿真工具S对网表n的仿真输出。对于任意网表$n \in N$，若存在某些仿真模式$a, b \in M$，使得：

$$O_a(n) \neq O_b(n),$$

则说明在不同仿真模式下，仿真结果不一致，从而检测到PCB设计工具链存在潜在缺陷：

$$\exists n \in N, \exists (a, b) \in M,\ O_a(n) \neq O_b(n) \Rightarrow bug。$$

第二种方法是在不同的仿真工具中对同一网表进行仿真，并对比仿真结果。尽管不同仿真工具的实现方式可能有所不同，但只要仿真配置保持一致，那么仿真结果在理论上应该是相同的。如果在相同输入条件下，不同仿真工具的仿真结果存在显著差异，则可能意味着某个仿真工具的计算逻辑存在问题，或者不同工具对电路模型的解析方式不一致，这可能导致设计错误或仿真不稳定性。需要说明的是，此处的“同一网表”是指每一个生成的测试样本。在实际测试过程中，PCBCover会持续生成大量结构各异的网表，并分别执行该差分测试方法。

形式化地，设N为网表集合，S_1, S_2为不同的仿真工具，定义$O_S(n)$表示仿真工具S对网表n的仿真输出。对于任意网表$n \in N$，若：

$$O_{S_1}(n) \neq O_{S_2}(n),$$

则说明存在不一致，从而检测到PCB设计工具链存在潜在缺陷：

$$\exists n \in N, O_{S_1}(n) \neq O_{S_2}(n) \Rightarrow bug。$$

综上，缺陷检测模块通过结合仿真运行异常捕获机制与差分测试策略，可有效检测PCB设计工具链中的潜在问题。这些方法能在不同仿真场景下验证工具的稳定性和正确性，从而为工具链的优化和缺陷修复提供有力支持。此外，本模块不仅检测仿真行错误，也记录仿真运行异常（如超时、崩溃、解析失败）作为策略反馈信号，辅助强化学习探索潜在缺

陷边界。因此，PCBCover在仿真器本身存在缺陷的情况下，依然能够挖掘有效异常场。

4 实验评估

4.1 实验设置

研究问题(RQs)：本文设定以下四个研究问题(RQs)，以探讨PCBCover在检测PCB设计工具链缺陷方面的能力。

RQ1：奖励函数中的权重参数对PCBCover的影响如何？

RQ2：PCBCover提出的两种覆盖率对发现缺陷的能力影响如何？

RQ3：结构多样性对缺陷触发能力的影响如何？

RQ4：与对比方法相比，PCBCover发现缺陷的能力表现如何？

RQ5：PCBCover可以发现真实缺陷吗？

目标PCB设计工具：借鉴现有的研究，本文选择KiCad和Ngspice作为评估PCBCover的实验平台。两者均为开源的PCB设计工具链组件，并拥有庞大的用户社区支持。KiCad是一款功能全面的PCB设计工具，提供电路图设计、网表生成及电路仿真等功能。而Ngspice作为SPICE系列仿真工具的一部分，支持多种电路描述语言，包括SPICE和HSPICE语法。

硬件与实验环境：实验在运行Windows 10(22H2) 64位的计算机上进行，硬件配置包括Intel Core i9处理器(3.0 GHz)、128 GB内存、2 TB SSD及两块8 TB HDD。实验的源代码及相关数据已在GitHub上公开[28]。

4.2 对比方法

本文选择PCBSmith作为电路图生成器，作为当前PCB设计工具链测试任务中较为先进的结构生成方案。据我们所知，目前没有算法针对PCB设计工具链测试进行特定优化。在此基础上，为评估不同配置策略对测试效果的影响，本文引入四种具有代表性的自动测试方法：Default、Swarm、HIS和RECORD。这些方法最初分别应用于FPGA编译器、传统编译器与信息物理系统(CPS)等领域，其核心思想聚焦于“配置参数空间探索”，与本文针对PCB工具链的测试目标高度契合。为确保对比实验的公平性与可复现性，本文已将上述方法迁移并适配至PCB设计工具链场景，并在统一的仿真工具环境(Ngspice 43 + KiCad 9.0)中实现运行，使其与PCBCover在相同测试周期、生成器平台与配置空间下执行。

Default[7]：此方法采用PCBSmith的默认配置进行测试用例生成。具体而言，测试生成器按照其预设的参数运行，不涉及任何额外的配置调整或优化，以评估默认策略下的测试效果。

Swarm[29]：该方法基于群体测试(Swarm Testing)，在每次测试迭代中对PCBSmith的配置参数进行随机化调整。其核心假设是，通过随机化配置，可生成更广泛的测试用例，从而提升测试覆盖率，发现更多潜在缺陷。该方法的关键在于利用配置组合的变化，探索可能导致错误的更多场景。

HIS[30]：该方法采用历史信息引导测试(History Information-Guided Testing)，结合粒子群优化(PSO)算法，根据先前检测到的缺陷信息调整PCBSmith的配置参数。具体而言，每轮迭代后，系统会基于已发现的缺陷模式优化配置，以提高后续测试的有效性。

RECORD[8]：该方法利用双重深度Q网络(Double DQN)，针对编译器测试任务进行配置策略优化。通过训练神经网络，RECORD能够学习并优化配置决策，以提高测试用例的多样性和有效性。

以上对比方法涵盖了从默认参数测试到基于优化算法和深度强化学习的智能测试策略，为评估PCBCover的性能提供了全面的对比基准。

4.3 RQ1:参数的影响

(1) 动机：本研究深入探讨了PCBCover算法中的一个关键参数：奖励计算公式中的权重系数θ。该参数在奖励函数中平衡覆盖率与仿真结果(失败或超时)的影响。当θ较高时，奖励函数更加关注覆盖率，倾向于引导PCBCover生成结构多样性更高的电路图；而当θ较低时，算法更加关注仿真结果，PCBCover优先生成更可能触发仿真运行异常与仿真行为异常的电路配置。因此，本RQ通过研究分析不同θ取值对PCBCover缺陷检测性能的影响，旨在确定最优设置，从而提升PCBCover的整体测试效率与缺陷触发能力。

(2) 实验方案：本RQ将权重系数θ的取值范围设定为0到1，并以0.1为步长进行调整。基于此，本RQ共设计并执行11组实验，分别测试不同θ取值下PCBCover在KiCad和Ngspice中的缺陷检测能力。在实验阶段，基于最新版的Ngspice(版43)和KiCad(版本9.0)，对不同参数设置下的PCBCover进行为期两周的系统性测试，并记录各

权重系数对应的不一致数量，以评估该系数对算法缺陷检测性能的影响。

本研究将PCBCover检测到的输出不一致数量作为缺陷检测能力的间接度量，具体包括两类：仿真运行异常（如仿真超时、崩溃、语法解析失败）与仿真行为异常。其中，输出不一致指的是相同网表在不同仿真工具或仿真模式下出现仿真结果差异[31-34]。为进一步提升评估的准确性与工程实用性，本文引入了缺陷去重策略[10]，用于从检测结果中识别出具有独立根因的非重复缺陷。该策略包括三步：（1）最小化测试用例结构，通过逐步删减器件与连接，仅保留能触发异常的最小子图；（2）根因归因分析，借助仿真器日志、堆栈信息及触发模式，判断异常是否由相同元器件类型、组合结构或特定输入条件引起；（3）异常聚类比对，将上述最小化用例与历史记录进行比较，识别语义等价或症状相似的异常。若多个不一致归因于相同的触发机制，则视为同源缺陷，仅计为一次有效触发。

（3）实验结果：如图7所示，PCBCover在不同θ值下的不一致检测数量呈现出明显的先上升后下降的趋势，整体曲线近似正态分布，说明该参数对算法的缺陷触发能力具有显著影响。当θ=0（即完全依赖仿真结果作为奖励）时，PCBCover检测到的不一致数量为3；而当θ=1（即完全依赖覆盖率指标）时，不一致数量为5，说明这两种极端设定都无法充分引导测试生成器挖掘潜在缺陷。随着θ值逐步增大，非重复不一致数量稳定上升，并在θ=0.6附近达到峰值（17个），此后开始缓慢下降。这一趋势表明：在覆盖率引导与仿真反馈之间实现适度平衡，能够提升PCBCover生成高质量测试用例的能力，从而更有效地发现缺陷。

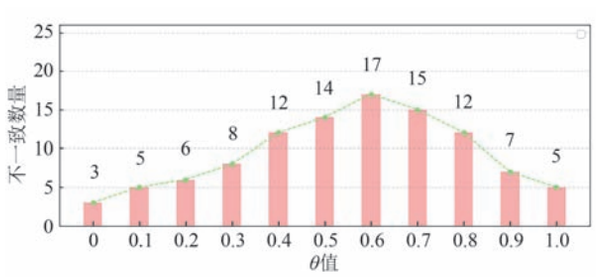


图7　PCBCover在参数θ不同取值下检测不一致的数量

值得注意的是，当θ=0（完全依赖仿真结果）时，检测到的非重复不一致数量（3个）反而低于θ=1（仅依赖覆盖率）时的数量（5个）。这可能是由于当θ过低时，奖励函数过度依赖仿真失败作为反馈信号，而忽略了结构多样性，使得搜索空间受限，导致生成的电路图缺乏结构变化，从而降低了整体的测试广度。相比之下，θ=1时虽然不考虑仿真结果，但通过覆盖率指标驱动的结构变化反而可能触发更多意料之外的边界行为，从而检测到更多的不一致。

（4）实验总结：本研究评估了奖励函数中系数θ对PCBCover缺陷检测能力的影响。实验结果表明，当θ=0.6时检测到最多的不一致（17个），相比两端极值提升明显。说明在覆盖率与仿真结果之间实现平衡，有助于生成更有效的电路图（测试用例）。

4.4　RQ2：覆盖率的影响

（1）动机：本研究问题（RQ）评估网表覆盖率和仿真覆盖率在PCB设计工具链缺陷检测中的有效性。PCBCover将这两类覆盖率作为奖励函数中的关键指标，用于引导测试用例的生成。通过系统地分析不同覆盖率指标对PCBCover缺陷检测性能的影响，可以探索它们在测试效率、缺陷触发能力等方面的优势与局限性，从而优化覆盖率的综合应用策略。

（2）实验方案：实验旨在评估不同覆盖率策略对PCB设计工具链缺陷检测效果的影响。本实验创建了PCBCover的两个变体，分别命名为$PCB_{NetCover}$和$PCB_{SimCover}$。$PCB_{NetCover}$仅使用网表覆盖率作为奖励指标，最大化生成网表的结构多样性，而不考虑电路在仿真过程中的节点激活情况与信号传播路径。相比之下，$PCB_{SimCover}$仅使用仿真覆盖率作为奖励指标，关注仿真过程中哪些器件被实际激活、哪些信号路径被执行，而不直接关注网表结构的复杂性与多样性。

本实验在最新版本的Ngspice和KiCad（即Ngspice 43和KiCad 9.0）上运行了PCBCover、$PCB_{NetCover}$、$PCB_{SimCover}$和Default四种方法，持续进行为期两周的测试实验。在测试过程中，每种方法不断生成网表并进行仿真。本实验对所有仿真结果进行了收集与比对，统计每种方法在仿真过程中触发的输出不一致量（仿真行为异常或仿真运行异常），此作为衡量其缺陷检测能力的指标。其它配置与RQ1相同。同时，PCBCover中奖励函数中权重参数θ取值0.6。

（3）实验结果：如表3所示，从整体结果来看，PCBCover检测出最多的不一致（17个），在四种方法中表现最优。这表明其融合网表覆盖率与仿真覆盖率的联合策略，能够在保持生成样本质量的前提

表3 使用不同覆盖率时检测的不一致数量

	Default	$PCB_{NetCover}$	$PCB_{SimCover}$	PCBCover
网表数	243264	129024	126336	124320
不一致	7	13	15	17

下，引导生成既具结构多样性又具仿真活跃度的测试用例，从而提高潜在缺陷的触发概率。

值得注意的是，PCBCover、$PCB_{NetCover}$ 和 $PCB_{SimCover}$ 所生成的网表数量相对较少（分别为124320、129024 和 126336），远低于 Default 的243264个。这是因为这些方法在生成过程中引入了强化学习控制策略，并以覆盖率反馈为优化目标，从而更注重质量优先而非数量堆叠。结果也印证了该策略的有效性：在生成数量减少的情况下，反而检测到了更多的不一致，说明测试效率显著提升。

在两个仅使用单一覆盖率的变体中，$PCB_{SimCover}$（仿真覆盖率）触发了15个非重复不一致，略高于 $PCB_{NetCover}$（网表覆盖率）的13个。这可能是因为仿真覆盖率更侧重于电路动态行为的反馈，能够更直接反映哪些器件被激活、哪些路径被执行，从而帮助生成器更容易定位可能引发仿真异常的场景。因此，$PCB_{SimCover}$ 所生成的网表在触发仿真不一致方面略占优势。

相比之下，Default方法虽然生成了最多的网表（243，264个），但仅触发了7个非重复不一致，说明在缺乏覆盖率反馈引导的情况下，生成器在大规模生成测试用例时缺乏有效性控制，难以高效探索潜在缺陷所在区域，导致测试质量不高。

综上所述，覆盖率引导策略在PCB设计工具链的缺陷检测中展现出明显的实用价值。特别是将网表结构和仿真行为的覆盖信息进行融合，能够更全面地引导测试数据生成，有效提升缺陷触发能力与测试效率。PCBCover的表现进一步验证了这一策略的可行性和实际应用潜力。

（4）实验总结：PCBCover在生成124，320个网表的情况下检测出17个仿真不一致，优于 $PCB_{SimCover}$（15）、$PCB_{NetCover}$（13）和 Default（7），体现出其融合覆盖率策略的有效性。单独使用仿真覆盖率的 $PCB_{SimCover}$ 表现略优于单独使用网表覆盖率的 $PCB_{NetCover}$，说明动态行为反馈在提升缺陷触发能力方面具有一定优势。整体结果表明，静态与动态覆盖率的联合引导策略有助于生成更高质量的测试用例，显著提升PCB工具链缺陷检测效率。

4.5 RQ3:结构多样性对缺陷触发能力的影响

（1）动机：PCBCover采用覆盖率引导机制自动生成电路图，以提升缺陷检测能力。本研究问题（RQ）旨在实证评估结构多样性对缺陷触发能力的影响，验证元器件类型数量（CTC）是否能够显著提升潜在缺陷的暴露概率。

（2）实验方案：为分析结构多样性与缺陷触发能力之间的因果关系，本实验选取元器件类型覆盖率（CTC）作为代表性结构覆盖指标。CTC定义为单个网表中实际使用的不同元器件类型数量与当前电路图生成器所支持的元器件类型总数之间的比值，归一化后范围为0%～100%。为确保该指标具有现实意义与可操作性，我们基于当前较为先进的电路图生成器PCBSmith进行建模。经系统分析，PCBSmith共支持22种主流元器件类型（如电阻、电容、电感、MOSFET、BJT、运算放大器、子电路模块、二极管、电源、光电器件等），因此定义100%覆盖水平即对应22种器件全部被使用的情形。需要注意的是，CTC本质上是一种静态结构属性，在普通生成流程中难以直接控制其数值变化。为保证实验的可重复性与因果推断的科学性，我们对PCBSmith的生成逻辑进行了定向约束：具体做法是修改其ConfigFile.txt配置文件，为每组实验设定限定的元器件类型集合，从而在结构生成阶段强制满足指定的CTC值。同时，为减少干扰变量，我们保持其他结构参数（如器件连接方式、拓扑形态等）不加干预，采用自然生成策略，以确保实验专注于评估CTC这一单一因素对缺陷触发能力的影响。本实验设置了四组不同覆盖水平的测试组别：25%（6种）、50%（11种）、75%（17种）和100%（22种），分别对应低、中、高及完整结构多样性的配置场景。每组实验均在Ngspice 43与KiCad 9.0环境下运行PCBCover，持续进行为期两周的测试。在测试过程中，PCBCover持续生成满足目标CTC的网表并进行仿真。我们收集所有仿真结果并进行比对，统计各组在仿真过程中触发的输出不一致数量，作为缺陷触发能力的衡量指标。其他配置与RQ1保持一致。由于不同组别在测试周期内生成的网表总数基本一致（控制在±3%以内），生成规模不再单独列出。

（3）实验结果：如表4所示，随着结构多样性（CTC）的提升，PCBCover在相同测试资源下触发的不一致数量显著增加。从CTC=25%（6种）到CTC=100%（22种），不一致触发数量增长了

表4　不同结构多样性(CTC)下触发不一致数量

CTC值(元器件类型占比)	25%(6)	50%(11)	75%(17)	100%(22)
不一致数量	5	12	14	17

240%，呈现出显著的非线性增强趋势。这表明结构多样性是缺陷激活能力的重要影响因子：更多类型元器件的组合使得工具链需处理更复杂的语义交互，进而提升异常边界的暴露概率。

具体而言，不同元器件在仿真器中的处理路径、模型调用逻辑、初始条件设置与数值稳定性均存在差异。例如，MOS管与BJT的建模机制不同，开关器件与子电路的初始化逻辑更为复杂；当这些器件在同一电路中混合出现时，极可能引发仿真器内部的推理冲突、收敛失败或路径不一致，从而激活潜在缺陷。

值得注意的是，尽管实验结果表明缺陷数量随结构多样性(CTC)的提升总体呈上升趋势，但在CTC提升至75%以上后，缺陷数量的增长趋势趋缓，呈现出非线性增强的边际递减特征。受限于当前PCBSmith所支持的元器件类型数量，结构多样性对缺陷激活能力的边界效应尚难全面验证，未来可通过扩展生成器能力进一步探究其收敛性趋势。

此外，尽管更高的CTC通常带来更强的缺陷触发能力，但该指标在实际测试中的作用并非“越高越好”的静态目标。PCBCover采用强化学习机制动态调整，以CTC其作为反馈信号，引导生成具有测试价值的多样化电路结构，而非盲目追求元器件类型覆盖率最大化。过高的结构复杂性可能引入冗余器件、无效路径或仿真负担，反而影响整体测试效率与可控性。

(4) 实验总结：CTC=100%的配置检测到17个非重复不一致，显著高于CTC=75%(14个)、CTC=50%(12个)和CTC=25%(5个)。该结果验证了在生成规模相近的条件下，结构多样性越高，缺陷激活能力越强。需要指出的是，尽管高CTC值在实验中表现出更强的缺陷触发能力，但其效果仍受到具体元器件间组合逻辑及工具链内部处理策略的影响。因此，CTC应作为反馈指标用于策略引导，而非固定阈值式的静态约束。

4.6　RQ4:缺陷检测的有效性

(1) 动机：PCBCover是一种基于覆盖率驱动的A2C强化学习方法，用于动态搜索配置参数并自动生成电路图，以创建网表测试用例，从而检测PCB设计工具链中的潜在缺陷。在本研究问题(RQ)中，实验的目标是系统评估PCBCover的有效性，并分析它是否能够比现有最先进的方法更精准、更高效地发现PCB设计工具链中的潜在问题。

(2) 实验方案：本实验在最新版本的Ngspice 43和KiCad 9.0上运行PCBCover以及对比算法，以评估其在最新工具链环境下的缺陷检测能力和测试效率。实验过程中，确保PCBCover和对比算法在相同的测试条件下运行，以保证实验结果的公平性和可比性。此外，除测试方法外，所有实验设置均与RQ1保持一致，包括测试网表的生成策略、仿真环境的配置以及数据收集方式。

(3) 实验结果：如表5所示，本实验比较了PCBCover与现有四种主流方法在生成网表数量与检测不一致个数两个维度的性能表现。尽管PCBCover所生成的网表数量最少(126,336个)，但它检测到最多的非重复不一致(17个)，在缺陷触发能力上显著优于对比方法，展现出更强的测试有效性。

表5　PCBCover与对比方法检测出不一致的数量

	Default	Swarm	HIS	RECORD	PCBCover
网表数	243264	231840	161952	131040	126336
不一致	194	212	235	255	274
非重复	3	5	10	12	17

Default方法生成网表最多(243,264个)，主要由于其不涉及配置空间搜索，直接采用固定策略，生成效率最高。然而，其检测到的非重复不一致数量仅为3，说明缺乏覆盖率或行为反馈引导的测试用例往往难以触达潜在缺陷区域，导致测试效果受限；Swarm方法通过随机扰动配置实现一定的多样性，生成速度较快，但由于未进行策略优化，其检测到的非重复不一致数量仅为5，提升有限；HIS方法借助历史缺陷信息和粒子群优化进行配置引导，因为缺乏历史故障信息，所以其效果和随机扰动方法类似(10个非重复不一致)；RECORD基于双重DQN进行状态建模和策略学习，与PCBCover在技术路线接近，但采用单智能体结构，缺乏多智能体之间的协同探索，其在生成数量上略高(131,040)，但触发不一致数量略低(12)，说明策略学习虽有效，但在配置空间的覆盖度上略逊一筹；PCBCover采用多智能体A2C强化学习框架，结合覆盖率反馈引导测试生成，在保证覆盖广度的同时优化了测试质量。尽管生成速度受限于策略学习过程，但通过更精准的

状态建模和动态策略更新，能够有效探索高潜力配置区域，实现更优的缺陷触发效果。

综上所述，PCBCover在缺陷检测有效性方面表现最优，不仅显著提升了单位样本的测试价值，还展示出在真实工具链环境中发现潜在问题的能力与实用性。

（4）实验总结：PCBCover检测到17个非重复不一致，显著高于RECORD（12）、HIS（10）、Swarm（5）和Default（3）。尽管生成网表数量最少，PCBCover仍展现出更高的单位测试效果，说明其强化学习与覆盖率引导策略能更有效触发缺陷。该结果验证了PCBCover在提升缺陷检测能力方面的优越性与实用性。

4.7 RQ5:检测真实缺陷的有效性

（1）动机：PCB设计工具链的可靠性对确保电路设计的准确性和产品制造的成功至关重要。若工具链存在缺陷，可能导致电路设计错误、生产故障，甚至在组装和测试阶段产生高昂的修复成本。因此，如何有效检测和修复PCB设计工具链中的潜在缺陷，成为提升工具链稳定性和可靠性的重要研究方向。本研究问题（RQ）旨在系统评估PCBCover在检测PCB设计工具链真实缺陷方面的能力，验证PCBCover是否能够发现真实的缺陷。

（2）实验方案：本实验在最新版本的PCB设计工具链（即Ngspice 43和KiCad 9.0）上运行PCBCover，以评估其在缺陷检测方面的有效性。选择最新版本的原因在于，新版本通常包含功能优化和代码更新，但同时也可能引入新的缺陷，这些缺陷往往较为关键，且尚未经过充分测试。因此，针对最新版本进行测试，有助于尽早发现潜在问题，为开发者提供有价值的反馈。在实验过程中，本实验严格遵循第3.4节缺陷检测模块中定义的缺陷检测策略，以确保评估过程的科学性和可复现性。所有由PCBCover发现的潜在缺陷均被提交至Ngspice和KiCad的开发者社区，由官方维护团队或社区成员进行审核和验证。

为了深入分析PCBCover检测到的缺陷，本实验从三个维度对提交的缺陷进行手动分类，包括开发者社区反馈（FB）、缺陷症状（SP）以及所属软件（S）。在开发者社区反馈（FB）方面，本实验将缺陷划分为三类：新发现（N）表示该缺陷在社区中首次被报告，开发者确认其为之前未知的问题；已知缺陷（K）表示该缺陷已被社区或开发者识别，可能已有修复计划或补丁；待确认（Pe）表示该缺陷尚未得到开发者的明确反馈，仍处于审核状态。

在缺陷症状（SP）方面，本实验将缺陷分为两类：性能问题（P）指仿真过程中出现的计算效率下降、资源消耗异常或运行时间过长等异常表现，这类问题可能影响仿真工具的稳定性和用户体验；仿真不一致（I）指相同的网表在不同的仿真环境下产生不一致的仿真结果，例如相同的电路在Ngspice和KiCad之间的仿真结果不同，或在同一工具的不同版本之间结果存在偏差。这类问题通常反映出工具在电路解析、数值计算或模型兼容性方面的潜在缺陷。

在软件归属（S）方面，本实验进一步标注每个缺陷的来源，以明确其归属于Ngspice（Ng）还是KiCad（Ki）。

通过这种细粒度的缺陷分类，本实验能够更系统地评估PCBCover的检测能力，同时也为PCB设计工具的改进和优化提供有价值的数据支持。

（3）实验结果：本实验利用PCBCover对最新版本的Ngspice和KiCad进行了为期四周的缺陷检测测试。如表6所示，PCBCover共发现13个得到开发者确认的缺陷。这些缺陷覆盖了不同器件变换场景、不同仿真工具，以及多种仿真异常类型，反映了PCBCover在真实环境中触发缺陷的能力和多样性。

从缺陷所属工具来看，Ngspice共检测出7个缺陷，KiCad检测出6个缺陷，说明PCBCover能在多个工具中发挥作用，并具备跨平台的适应性。尤其是在Ngspice中，PCBCover触发的缺陷主要集中于仿真性能相关场景，如仿真不收敛、仿真延迟或仿真中断等问题，说明PCBCover在应对复杂电路结构和高动态行为时，能够高效识别潜在的稳定性风险。

从缺陷症状维度分析，性能问题（P）占比最大，共11个，包括仿真收敛失败、仿真超时、资源异常等问题。这表明PCBCover能有效暴露出工具在仿真效率与稳定性方面的缺陷。此外，仿真不一致（I）类缺陷共2个，即同一电路在不同仿真模式或工具中产生不同结果，揭示了工具在模型解释或求解器兼容性方面的潜在问题。

在开发者反馈方面，有6个缺陷被标记为新发现（N），这些缺陷此前未被社区报告，经过提交后获得开发者的确认，说明PCBCover具有实际的缺陷发现能力。3个缺陷为已知问题（K），表明PCBCover能稳定复现已有问题，验证了其可靠性；其余4个缺陷为待确认（Pe），目前正在开发者社区

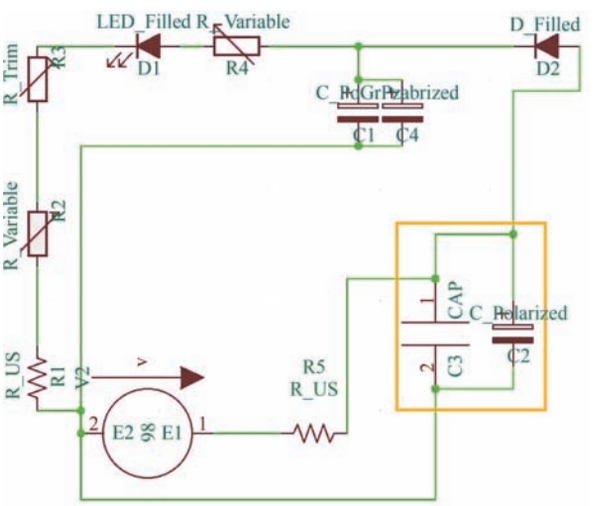
表6　缺陷详情

ID	总结	反馈	症状	软件
#754	应用两个串联电感时仿真崩溃。	K	P	Ng
#755	覆盖引导调整成两个串联电感后，修改后的网表无法收敛。	K	P	Ng
#756	经覆盖引导调整后，原先失败电感网表实现正常收敛。	K	P	Ng
#761	在开关控制电路中使用下拉电阻时出现仿真不一致问题。	Pe	P	Ng
#762	某些边界连接关系在仿真中表现出非预期收敛行为。	Pe	P	Ng
#763	电阻变化会影响仿真的收敛性。	Pe	P	Ng
#764	配置生成串联电感网表，Ngspice仿真超时。	Pe	P	Ng
#20402	在测试并联电容配置时，KiCad仿真失败。	N	I	Ki
#20403	针对多组件条件组合场景，测试暴露KiCad仿真稳定性不足。	N	P	Ki
#20404	在晶体管电路中重新定位电源符号后，KiCad出现仿真不一致问题。	N	I	Ki
#20409	在进行等效并联电容转换后，KiCad仿真无法收敛。	N	P	Ki
#20414	特定串联连接模式导致仿真求解失败，揭示收敛性问题。	N	P	Ki
#20418	一个二极管场景中，并联电容导致仿真终止。	N	P	Ki

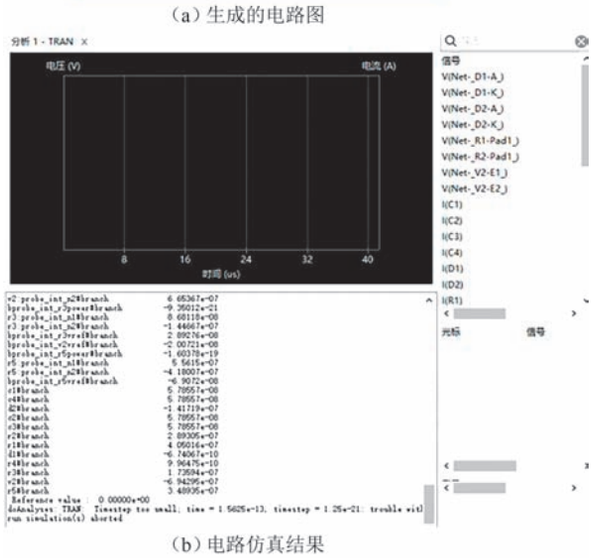
中进一步验证，尚无定论。需要说明的是，相较于其他类型的缺陷，仿真相关问题通常涉及复杂的电路行为与数值计算过程，因此其确认过程往往更加耗时且具有一定的分析难度。

性能问题。Bug(#20409)属于仿真性能类问题。图8(a)所示为由PCBCover自动生成的测试电路图。可以看到，在黄虚线框标注区域，电路中使用了两个并联电容(C2和C3)。当该电路在KiCad中进行仿真时，出现了仿真不收敛的问题。图8(b)展示了仿真结果，可以观察到仿真在初期阶段发生中断，并给出“timestep too small”的错误提示。该错误通常意味着仿真求解器在进行时间积分时出现了数值稳定性问题，即为了维持求解精度，仿真步长被迫不断缩小，最终低于允许的最小时间步长，导致仿真不收敛。这可能由以下几个因素引起：(1)并联电容结构在初始时刻引入了较大的瞬态电流变化，增加了电路的刚性(stiffness)；(2)节点间的电压变化过快，导致数值解出现震荡，仿真器无法快速稳定求解；(3)电容值设置较小或初始条件未初始化，进一步放大了求解难度。缺陷说明，结构上的细微变化也可能引发仿真器在数值处理上的不稳定行为。PCBCover能有效触发此类边界行为，为仿真工具的稳定性验证提供了具有代表性的测试场景。

不一致。Bug(#20402)属于仿真不一致类问题。该案例中，PCBCover生成了一个电路图，使用KiCad的内置仿真功能对其进行仿真时，仿真失败，无法获得有效结果。然而，在不修改电路图的前提下，将该电路通过KiCad导出为网表文件，并在Ngspice中加载并仿真，仿真过程正常完成，结果也



(a) 生成的电路图



(b) 电路仿真结果

图8　Bug(#20409)的电路图和仿真结果

符合预期。图9中展示了在Ngspice中成功仿真的输出。该现象表明，在相同电路结构与仿真配置一

致的前提下，不同仿真环境（KiCad内嵌仿真 vs Ngspice命令行仿真）在执行过程中出现了行为差异。根据差分测试的定义，这种差异性即被视为潜在缺陷，可能源于以下几种原因：（1）KiCad的仿真前处理器（如netlist解析或仿真参数绑定）存在差异或实现不一致；（2）仿真模型或初始化条件在KiCad与Ngspice之间存在兼容性问题；（3）KiCad内嵌的仿真接口对某些器件结构或拓扑不具备稳定支持，导致求解器失败。该缺陷说明，尽管KiCad与Ngspice使用相同的仿真引擎，但实际封装与集成过程中的实现差异仍可能导致仿真行为不一致。PCBCover能有效揭示这类由于工具链集成差异引发的问题，对提升工具链整体一致性与可靠性具有实际意义。

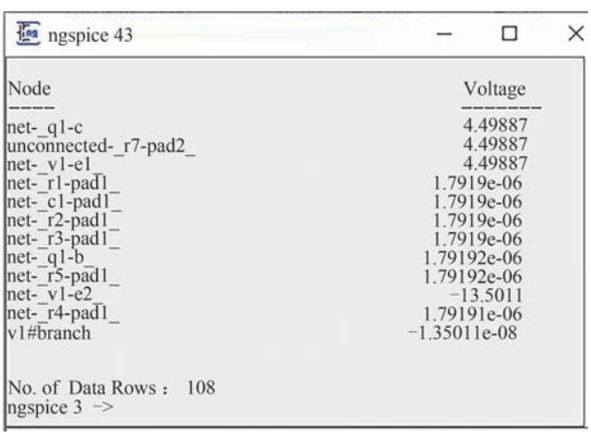


图9 Ngspice仿真结果

（4）实验总结：PCBCover在Ngspice 43与KiCad 9.0上共检测出13个真实缺陷，其中6个被确认为新发现，涵盖11个性能问题与2个仿真不一致问题。此外，PCBCover能够在多个工具中稳定发现关键问题，验证其跨平台适应性与实际工程价值。这些结果表明，PCBCover在真实环境中具备高效识别工具链缺陷的能力，有助于提升工具链的稳定性与可靠性。

5 相关工作

在PCB设计领域，已有研究尝试分析或检测PCB的缺陷，但这些方法多集中在实际制造板子（如焊接、布线、连接）中的物理缺陷检测，例如：Lakshmi等人[35]对用于PCB缺陷检测的多种学习方法进行了综述研究。研究指出这些方法利用图像识别方法检测断裂、短路、焊点虚焊等问题。Sankar等人[36]则从通孔元件（Through-Hole）和表面贴装元件（Surface-Mounted Devsice）两个类别出发，系统分析了PCB中常见的缺陷类型，并探讨了这些缺陷的发生频率及成因。

与之相比，PCB设计工具链的缺陷检测聚焦于设计工具在生成网表、仿真配置等过程中的逻辑或行为错误，其测试目标和方法均与检测PCB产品自身缺陷存在本质区别。值得注意的是，PCBSmith[7]是首个面向PCB设计工具链测试的电路图生成方法。它通过配置参数自动生成结构多样的电路图（即测试用例），并结合差分测试方法，能够有效触发设计工具链中的部分异常行为。但该方法仍依赖固定配置或随机策略，缺乏对“缺陷触发潜力”配置组合的智能识别与引导。

目前，在通用编译器测试（如GCC、LLVM）和CPS编译器（如Simulink）领域，已经提出多种通过配置多样化来生成更多样化程序的方法。例如，Groce等人[29]提出了基于群体测试的方法，通过随机化生成器的配置来生成测试用例。Chen[30]等人提出基于历史信息的引导测试，通过离线挖掘历史缺陷推断一组测试配置。Jiang等人[9]提出了基于单智能体强化学习的在线训练方法，通过实时反馈动态调整生成器配置。但这些方法在生成器配置多样化方面存在一定局限性，例如群体测试方法的高度随机性导致覆盖不足、基于历史信息的引导测试受限于过去的缺陷模式，以及单智能体强化学习方法缺乏协作性且在复杂环境下学习效率较低。因此，针对PCB设计工具链的特性，设计一种融合结构特征建模与动态行为反馈的智能配置搜索机制，显得尤为必要。

6 总结与展望

PCB设计工具链的可靠性至关重要，因其缺陷可能导致电路功能错误，进而引发严重的制造或应用风险。然而，现有测试方法在缺陷触发能力仍存在明显不足。为此，本文提出了PCBCover，一种融合覆盖率反馈与强化学习的自动化测试方法，旨在提升测试用例的质量与缺陷检测效果。PCBCover设计了静态（网表）和动态（仿真）两类覆盖率指标，用作强化学习奖励信号，引导A2C智能体在庞大的参数配置空间中高效搜索。实验在KiCad和Ngspice的最新版本上评估，结果表明PCBCover在测试效率与缺陷触发能力方面均优于四种先进对比

方法。最终，PCBCover共检测出13个真实缺陷，包括11个性能问题与2个仿真不一致问题。实验验证了PCBCover的有效性和实用性。未来工作将探索多目标奖励机制，进一步提升测试能力，并拓展对更多PCB工具的支持。

未来工作还将围绕工具集成与部署场景展开进一步研究，特别是在持续集成（CI）环境下的自动化测试能力构建。我们计划将PCBCover封装为可配置的测试模块，适配GitLab CI、Jenkins等主流持续集成平台的任务调度机制。具体实现上，PCBCover将支持与版本控制系统的事件联动（如代码提交、分支合并），在工具链更新时自动执行回归测试任务，记录并反馈异常信息。此外，我们将规范其输入输出接口，提升与现有EDA工具（如KiCad、Ngspice）的兼容性与稳定性，为在工业CI环境中构建自动化测试流程提供基础支撑。

作者贡献声明　赵旭与邹沛煜为共同一作者。

致　谢　本研究得到国家自然科学基金（项目编号：62032004、62572090、62202079）的资助。

参　考　文　献

[1] Vasilyev F, Isaev V, Korobkov M. The influence of the PCB design and the process of their manufacturing on the possibility of a defect-free production. Przeglad Elektrotechniczny, 2021, 97(3): 91-96

[2] Abdullah S M A, Toulan N M F, Amen A A H. Effective schematic design phase in design process. International Journal of Technology and Design Education, 2024, 34(5): 2005-2039

[3] Wang C N, Nhieu N L, Viet T A P. Enhancing efficiency in PCB assembly for the leading global electronics manufacturing services firm: a TRIZ and Ant Colony Optimization approach. The International Journal of Advanced Manufacturing Technology, 2024, 133(11): 5529-5552

[4] Zhao X, Jiang H, Guo S, et al. A comprehensive study of open-source printed circuit board (PCB) design software bugs. IEEE Transactions on Instrumentation and Measurement, 2024, 73: 2005816

[5] V. Le, M. Afshari, Z. Su. Compiler validation via equivalence modulo inputs//Proceedings of the ACM-SIGPLAN Symposium on Programming Language Design and Implementation (PLDI). Gothenburg, Sweden, 2014:216-226

[6] V.Le, C.Sun, Z.Su. Finding deep compiler bugs via guided stochastic program mutation// Proceedings of the ACM SIGPLAN International Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA). New York, USA, 2015: 386-399

[7] Zhao X, Jiang H, Li X, et al. PCBSmith: an effective schematic generator for testing PCB design tool chain. IEEE Transactions on Reliability, 2025: 1-15

[8] Li X, Guo S, Cheng H, et al. Simulink compiler testing via configuration diversification with reinforcement learning. IEEE Transactions on Reliability, 2023, 73(2): 1060-1074

[9] Jiang H, Zou P, Li X, et al. DeLoSo: detecting logic synthesis optimization faults based on configuration diversity. ACM Transactions on Design Automation of Electronic Systems, 2024, 30(1): 1-26

[10] Zou P, Jiang H, Li X, et al. Logic synthesis tools testing via configuration diversification with combinatorial multi-armed bandit. IEEE Transactions on Instrumentation and Measurement, 2025, 74: 3509417

[11] Bogatin E. Bogatin's practical guide to prototype breadboard and PCB design. Artech House, Norwood , USA, 2021

[12] Lin R C. Human-centered circuit board design with flexible levels of abstraction and ambiguity. University of California, Berkeley, USA, 2021

[13] Wehrli G. Generating platform configuration from netlists. ETH Zurich, Ramistrasse, Zurich, 2024

[14] Hemker D, Maalouly J, Mathis H, et al. From schematics to netlists-electrical circuit analysis using deep-learning methods. Advances in Radio Science, 2024, 22: 61-75

[15] Yang S, Ren J, Liu B, et al. JSICsim—An analog simulator for superconductor integrated circuit. IEEE Transactions on Circuits and Systems II: Express Briefs, 2022, 70(3): 1129-1133

[16] Wei Y, Liu J, Sun D, et al. From netlist to manufacturable layout: An auto-layout algorithm optimized for radio frequency integrated circuits. Symmetry, 2023, 15(6): 1272

[17] Wan G, Dong Q, Sun X, et al. Highly efficient and accurate algorithm for multiscale equivalent modeling and mechanical performance simulation of printed circuit boards. Microelectronics Reliability, 2023, 147: 115134

[18] Yang X, Chen Y, Eide E, et al. Finding and understanding bugs in C compilers//Proceedings of the 32nd ACM SIGPLAN conference on Programming language design and implementation. Berlin, Germany, 2011: 283-294

[19] Chowdhury S A, Mohian S, Mehra S, et al. Automatically finding bugs in a commercial cyber-physical system development tool chain with SLforge// Proceedings of the 40th International Conference on Software Engineering. Gothenburg, Sweden, 2018: 981-992

[20] Herklotz Y, Wickerson J. Finding and understanding bugs in FPGA synthesis tools//Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Seaside, USA, 2020: 277-287

[21] Thomas D, Moorby P. The Verilog Hardware Description Language. Berlin, Gemany: Springer Science & Business Media, 2008

[22] Lu J, Yang J, Li S, et al. A2C-DRL: dynamic scheduling for stochastic edge-cloud environments using A2C and deep reinforcement learning. IEEE Internet of Things Journal, 2024, 11(9): 16915-16927

[23] Wu Q, Wu J, Shen J, et al. Distributed agent-based deep

reinforcement learning for large scale traffic signal control. Knowledge-Based Systems, 2022, 241: 108304

[24] Zhou Z, Zhou Y, Fang C, et al. Coverage goal selector for combining multiple criteria in search-based unit test generation. IEEE Transactions on Software Engineering, 2024, 50(4): 854-883

[25] Sun BC, Gong DW, Yao XJ. Test case generation for path coverage of MPI program guided by surrogate-assisted multi-task evolutionary optimization. Journal of Software, 2021, 31(2): 1-17 (in Chinese)
(孙百才, 巩敦卫, 姚香娟. 代理辅助多任务进化优化引导的MPI程序路径覆盖测试用例生成. 软件学报, 2021, 31(2): 1-17)

[26] Cui Zhan-Qi1, Zhang Jia-Ming, Zheng Li-Wei, Chen Xiang. A survey of researchon coverage-guided greybox fuzzing. Chinese Journal of Computers, 2024, 47(7):1666-1696 (in Chinese)
(崔展齐, 张家铭, 郑丽伟, 陈翔. 覆盖率制导的灰盒模糊测试研究综述. 计算机学报, 2024, 47(7):1666-1696)

[27] Herbold S, Tunkel S. Differential testing for machine learning: an analysis for classification algorithms beyond deep learning. Empirical Software Engineering, 2023, 28(2): 34-35

[28] Sourcecode. 2025, https://github.com/StudyOfPCBDesignSoftware/PCBCover

[29] Groce A, Zhang C, Eide E, et al. Swarm testing// Proceedings of the International Symposium on Software Testing and Analysis. Minneapolis, USA, 2012: 78-88

[30] Chen J, Wang G, Hao D, et al. History-guided configuration diversification for compiler test-program generation//Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering. San Diego, USA, 2019: 305-316

[31] Jiang H, Cheng H, Guo S, et al. Partition based differential testing for finding embedded code generation bugs in Simulink// Proceedings of the 60th ACM/IEEE Design Automation Conference (DAC). San Francisco, USA, 2023: 1-6

[32] Xu Z, Guo S, Li X, et al. SIMTAM: generation diversity test programs for FPGA simulation tools testing via timing area mutation. ACM Transactions on Design Automation of Electronic Systems, 2025, 30(2): 1-25

[33] Li P, Meng W, Lu K. Sediff: scope-aware differential fuzzing to test internal function models in symbolic execution// Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Singapore, 2022: 57-69

[34] Guo S, Jiang H, Xu Z, et al. Detecting Simulink compiler bugs via controllable zombie blocks mutation// Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Singapore, 2022: 1061-1072

[35] Lakshmi G, Sankar V U, Sankar Y S. A survey of PCB defect detection algorithms. Journal of Electronic Testing, 2023, 39(5): 541-554

[36] Sankar V U, Lakshmi G, Sankar Y S. A review of various defects in PCB. Journal of Electronic Testing, 2022, 38(5): 481-491

ZHAO Xu, Ph. D. candidate. His research interests include intelligent software engineering, software testing and data mining

ZOU Pei-Yu, Ph. D., lecturer. His research interests include intelligent software engineering and EDA industrial software.

LI Xiao-Chen, Ph. D., associate professor. His research interests include intelligent software engineering and software testing.

LIU Lu-Kai, undergraduate student. His research interests include software testing.

LIU Hui, Ph. D., professor. His research interests include deep learning-based software engineering, software refactoring and software quality.

SHI Chong-Yang, Ph. D., professor. His research interests include information retrieval, knowledge engineering and sentiment analysis.

JIANG He, Ph. D., professor. His research interests include intelligent software engineering, software testing and search-based software engineering.

Background

BackgroundThe research presented in this paper belongs to the domain of software testing for Electronic Design Automation (EDA), with a specific focus on the automated testing of Printed Circuit Board (PCB) design tool chains. PCB design tools are critical components in EDA workflows, facilitating the graphical construction of schematics, simulation configuration, and netlist generation. However, like all software systems, these tools are susceptible to bugs. Given that PCBs are widely used in safety-critical domains such as aerospace, healthcare, and automotive electronics, even a minor defect in a PCB design tool can lead to severe functional failures and safety risks in real-world applications. Hence, ensuring the correctness and robustness of PCB design tool chains is vital.

While significant progress has been made internationally in

the development of automated test generators in areas such as compiler testing (e. g., Csmith), CPS systems (e. g., SLforge), and hardware design (e. g., Verismith), the testing of PCB design tools remains relatively underexplored. Existing schematic generators like PCBSmith rely on random or default configurations, which limit their effectiveness in generating test cases that can uncover hidden defects. Additionally, there has been a lack of intelligent guidance mechanisms to adaptively explore the large and complex configuration space of schematic generation.

This paper addresses the above gap by proposing PCBCover, a novel automated testing framework that combines coverage feedback with reinforcement learning to improve the effectiveness of schematic-based testing. By designing static (netlist-level) and dynamic (simulation-level) coverage metrics and integrating them as reward signals in an A2C reinforcement learning algorithm, PCBCover can efficiently search for configuration parameters that are more likely to trigger toolchain defects. Compared to existing methods such as Default, RECORD, HIS, and Swarm, PCBCover demonstrates superior effectiveness in defect detection and testing efficiency, and it has successfully discovered 13 real-world bugs in tools like KiCad and Ngspice.

This work is part of a broader research effort by the authors' group, which has previously developed tools such as PCBSmith (a rule-based schematic generator) and PCBFen (a mutation-based variant generator). PCBCover builds on these foundations by introducing intelligence and adaptivity into the testing process, marking a significant advancement in automated testing for PCB design environments.