基于网表控制流分析的硬件木马检测方法

张 宁[1,2], 吕志强[1,2], 张焱琳[1,2], 黄伟庆[1,2]

[1]中国科学院信息工程研究所 第四研究室 北京 中国 100084
[2]中国科学院大学 网络空间安全学院 北京 中国 100049

摘要 在现代集成电路设计中, 随着芯片规模的不断扩大, 越来越多的功能集成其中, 不可避免地引入很多第三方设计的 IP 核。第三方 IP 核的引入极大地缩短了芯片设计周期, 但也导致在芯片设计阶段可能引入具有恶意功能的电路, 即硬件木马。植入的硬件木马一旦触发, 可以改变芯片的原有功能、泄漏芯片内处理的信息, 甚至物理上损坏芯片。因此, 针对 IP 核进行硬件木马检测可以降低芯片设计中引入的风险。随着现代芯片制造工艺发展到纳米尺度及 3D 结构, 从流片后的芯片中检测微小的硬件木马电路变得困难。设计阶段的硬件木马检测变得越来越重要。为了增强隐蔽性, 硬件木马电路通常采用低概率触发的电路设计, 同时保证电路规模较小。现有的硬件木马检测方法通过硬件木马的低概率触发电路的某一特征进行检测, 比如可测性值、扇入扇出电路结构等。近期, 一些学者提出了抗检测的硬件木马设计, 可以针对性擦除硬件木马网表中的检测结构特征。在本文中, 提出了一种新型的建模方法将电路转为节点控制流图 NCFG, 并提出了基于控制流分析的硬件木马检测方法。该方法可以同时分析组合逻辑电路和时序逻辑电路, 还可以量化分析硬件木马的隐蔽性。实验结果表明, 对于常见的硬件木马(TrustHub 硬件木马检测样本库)和新型抗检测硬件木马(如抗 UCI 木马、DeTrust 木马、DeTest 木马等), 该检测方法都可以达到很高的准确性。更重要的是, 本文提出的节点控制流图 NCFG 模型具有很好的扩展性。对于未来出现的硬件木马, 可以基于此模型添加新的特征分析进行扩展。

关键词 芯片安全; 硬件木马检测; 网表; 控制流; 特征分析
中图法分类号 TP309.2 DOI 号 10.19363/J.cnki.cn10-1380/tn.2026.01.20

A Novel Hardware Trojan Detection Method based on the Controllability Flow Analysis of the Netlist

ZHANG Ning[1,2], LV Zhiqiang[1,2], ZHANG Yanlin[1,2], HUANG Weiqing[1,2]

[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100084, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

Abstract With the expansion of modern integrated circuit (IC) scale, it's inevitable to introduce many intellectual property (IP) cores designed by the third party. The insertion of IP cores extremely shortens the IC design process, while it also leads to the potential insertion of malicious circuits called Hardware Trojans (HT). When inserted HT is triggered, it can change the desired functions of IC, leak the secret information in the IC, even destroy the IC physically. Thus, HT detections are essential to mitigate the thread of HT which may be inserted during the IC design flow. With the IC manufacture technology development to nanometer and 3D structure, it's challenging to detect small HT circuits in a whole IC after fabricated. The HT detection in the design flow becomes more and more important. To improve invisibility, HT circuits are usually rarely triggered and consist of small circuits scale. Most of existing HT detection methods are rely on the particular character of rarely triggered HT netlist, like measurability value, the fan-in and fan-out structure, etc. Recently, many researchers present some novel implicitly anti-detection HT designs which can resist netlist character based detections. In this paper, we propose a general modeling method of designs, which transform the netlist to a Node Controllability Flow Graph (NCFG). Meanwhile, we propose a novel HT detection method based on the controllability flow analysis of NCFG considering both combinational and sequential logics. It synthesize the HT detection analysis of combinational andsequential logics. We can use NCFG analysis to quantize the implicity of HT circuits. The experiment result shows that it has high accuracy to detect common HT from TrustHub testbench and novel implicitly HT, like Defeating UCI, DeTrust and DeTest. Moreover, the proposed NCFG model has good expansibility, it has advantages to be extended for future stealthier HT features analysis based on NCFG.

Key words chip security; hardware trojan detection; netlist; controllability flow; feature analysis

通讯作者: 张宁, 硕士, 工程师, Email: zhangning@iie.ac.cn。
本课题得到国家重点研发计划课题(No. 2022YFC3320600)资助。
收稿日期: 2020-12-14; 修改日期: 2021-02-22; 定稿日期: 2023-08-10

1 引言

近年来, 随着集成电路产业的蓬勃发展, 芯片产业全球化程度越来越高, 作为实现信息设备安全的基石, 芯片的安全问题也成为信息安全中的研究重点。由于芯片行业的产业链非常复杂, 从设计电路到芯片生产再到封装测试, 每一个环节都有第三方工具、人员和厂家参与, 存在很大的硬件安全风险[1]。同时, 由于功能需求的复杂化, 芯片的整体规模越来越大, 在设计阶段不可避免地会集成第三方的知识产权核(Intellectual Property core, IP 核)来降低设计复杂度。但第三方的 IP 核及代码极易被植入恶意逻辑电路, 即硬件木马[2]。植入硬件木马电路后, 芯片在硬件木马触发条件下可篡改原始电路功能、泄漏电路内部信息甚至对芯片进行物理性破坏等行为[3], 对信息安全产生极大的威胁。由于硬件木马隐蔽性强、破坏性高且不可逆等特点, 硬件木马的检测方法研究成为实现底层信息安全的重要任务。

针对不同阶段引入实现恶意功能的硬件木马, 研究人员提出了多种可用于识别硬件木马的检测方法[4]。但与此同时, 攻击人员也会针对检测方法中存在的漏洞升级硬件木马设计方案[5], 来躲避相关检测。在硬件木马检测方法研究的初期阶段, 研究人员通过芯片流片前的形式验证[6-7]、流片后的功能测试[8-9]来检查电路功能以识别硬件木马。由于芯片本身的功能非常复杂, 输入测试向量无法覆盖全部电路功能。同时, 由于硬件木马设计巧妙, 隐蔽性强, 有限的测试向量并不能使其触发改变电路功能, 因此传统测试方法很难检测硬件木马。

针对上述问题, 研究人员提出了新的硬件木马检测方法。主要包括基于侧信道分析的检测方法[10-13]和针对电路设计文件分析的检测方法[28-45]。基于侧信道分析的检测方法, 通过分析待测芯片在工作时产生的侧信号(如电磁辐射、温度、功耗、延迟等信息)来区分电路中是否植入硬件木马。该检测方法首先需要对芯片工作时产生的侧信号进行采集, 因此受环境噪声和硬件设备影响较大。此外, 还需要一个确定无硬件木马植入的“黄金芯片”用于侧信号的对比分析。考虑到实际中难以获取“黄金芯片”, 研究人员提出一种逆向重构的检测方法[14-16], 对待测芯片逐层逆向重构, 将重构的版图与设计阶段电子设计自动化(Electronic Design Automation, EDA)工具生成的原始版图进行对比分析, 确认是否植入硬件木马。这种方法会对芯片造成不可逆转的破坏, 且第三方工具的安全性无法保证[17], 原始版图的正确性需要从电子设计文件中得到确认, 因此如果能在电子设计文件上对硬件木马进行正确识别, 能有效降低木马威胁。

针对电路设计文件分析的检测方法针对的是设计阶段引入的第三方 IP 核及电路设计文件进行硬件木马检测。为了增加隐蔽性, 硬件木马通常具有低触发概率, 在电路结构特征上表现出不同于正常电路的特征。研究人员通过对硬件木马网表特征进行量化或建立硬件木马特征库, 对待测网表文件进行特征匹配, 将硬件木马的检测转化为特征识别。然而, 由于硬件木马低概率触发的实现方式多种多样, 建立具有统一性的结构特征很难, 如何构建一个普适的硬件木马特征库成为针对电路设计文件进行硬件木马检测的难点。

针对上述问题, 本文提出了基于网表控制流分析的硬件木马检测方法。本文主要创新点和贡献:

(1) 本文提出了一种将网表抽象为节点控制流图进行统一检测的方法, 针对被测电路寄存器传输级设计(Register Transfer Level, RTL 级)或门级网表文件, 均可根据其逻辑门类型及连接关系建立统一的节点控制流图。

(2) 本文提出了基于节点控制流图的时序逻辑触发硬件木马和组合逻辑触发硬件木马的分析方法, 可有效检测低概率触发的硬件木马。

(3) 针对新型抗检测硬件木马, 经实验验证, 基于网表控制流分析的检测方法有效。

本文的安排如下所述: 第 2 节介绍了相关工作及背景; 第 3 节介绍了基于网表控制流分析的硬件木马检测方法; 第 4 节是实验及结果评估; 第 5 节是对本文硬件木马检测方法的总结和展望。

2 相关工作

为了检测网表文件中是否包含硬件木马, 研究人员提出了多种方法, 从检测方式上可以分为两类, 分别为动态仿真分析和静态逻辑分析。动态仿真分析通过生成测试向量, 对网表文件进行仿真测试, 分析仿真过程中目标电路的内部状态及输出信号, 判断其中是否包含硬件木马疑似电路。静态逻辑分析对网表文件中的逻辑门及其连接关系进行分析, 提取低概率触发电路特征并进行检测, 如可控性可观性特征、触发器自环结构特征等。

2.1 硬件木马的结构及特点

硬件木马结构上由触发电路和有效载荷两部分组成, 如图 1 所示。为了增强隐蔽性, 硬件木马通常处于静默状态, 当触发电路通过输入信号或电路内

部信号触发时激活有效载荷实施攻击, 改变电路正常功能。因此触发电路的设计会直接影响硬件木马的隐蔽性能。

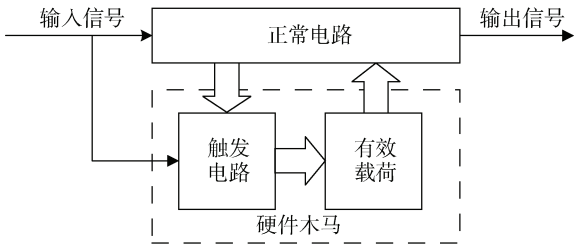


图 1 硬件木马结构示意图

Figure 1 The structure of hardware trojan

隐蔽性是硬件木马电路区别于正常电路的最重要特征, 实现方法包括组合逻辑触发、时序逻辑触发等。因此, 研究人员将硬件木马检测抽象为低概率触发特征的电路检测。

2.2 动态仿真分析

硬件木马的动态仿真分析方法, 依靠集成电路测试工具进行检测, 由于功能仿真过程中, 测试向量无法覆盖全部电路, 硬件木马在此过程中很难被激活, 而正常电路信号在测试过程中通常会发生翻转或状态转换, 因此研究人员将硬件木马检测抽象为动态仿真分析过程中的低测试覆盖率电路检测、冗余电路检测等。

2.2.1 基于测试覆盖率的动态检测方法

在动态仿真分析的过程中, 测试覆盖率是用来衡量测试质量的重要指标, 包括功能覆盖率和代码覆盖率。通过施加合适的测试激励, 可有效提高测试覆盖率, 从而激活可能存在的硬件木马。硬件木马检测中可采用的测试覆盖率包括行覆盖率、语句覆盖率、切换覆盖率、有限状态机覆盖率等。文献[18]分析了硬件木马电路和正常电路在行覆盖率上的区别, 并验证了基于 Hoskote 算法与聚类分析的硬件木马检测方法。文献[19]提出了使用启发式算法的输入向量生成方法, 能够有效提高正常电路的代码覆盖率, 降低硬件木马检测时的误检率。文献[20]在基于测试覆盖率检测的基础上结合了等价性验证和冗余电路消除技术, 减少了疑似信号的数量, 降低了误检率。文献[21]提出了一种基于有限状态机覆盖率的动态检测方法, 使用动态验证工具进行状态提取, 状态检查与覆盖率分析。通过识别未激活状态和可疑状态检测硬件木马。

2.2.2 基于冗余电路识别的动态检测方法

硬件木马的显著特性是低触发概率, 在对目标电路进行动态仿真分析时, 硬件木马处于未激活状态。Hicks 等[22]最早提出了基于冗余电路识别(Unused Circuit Identificaiton, UCI)的动态检测方法。提取组合逻辑电路的输入输出信号为节点, 每两个节点组成一个信号对(s, t)。在动态仿真过程中, 若 t 始终等于 s, 不受其余节点数值变化的影响, 则信号对(s, t)为冗余信号, 疑似为硬件木马。Zhang 等[23]随后设计了可以躲避 UCI 检测的硬件木马, 并提出了一种更完善的动态检测方法 VeriTrust。该方法是一种基于冗余输入信号分析的动态检测方法。由于硬件木马在动态检测时未被激活, 硬件木马电路与根据动态仿真信号真值表重构的电路相比, 包含冗余输入信号, 以此为特征进行硬件木马的检测。为了降低检测的复杂度, 使用 EDA 工具以积之和(Sum of Product, SOP)或和之积(Product of Sum, POS)形式对电路进行简化[24]。通过对动态仿真过程中电路函数式 SOP 和 POS 项进行追踪分析, 提取函数式中的冗余项, 即包含硬件木马输入信号。文献[23]提出了一种基于蒙特卡罗算法的测试向量生成方法, 通过在不同测试过程中周期检测, 提高了检测的准确率, 简化了测试向量集。文献[25]提出了基于低翻转信号(Less Toggled Signals, LTS)的动态检测方法, 通过动态仿真得到 LTS 信息并构造有向图, 并利用聚类分析算法对有向图进行特征分类, 有效地识别冗余电路。

为了促进硬件木马检测, 研究人员也提出了硬件木马抵抗检测的设计方法。例如, Sturton 等[26]使用选择器替代部分组合逻辑电路使硬件木马成功逃避 UCI 检测。文献[27]提出使用 D 触发器将触发结构的多个输入信号分散到正常电路中, 通过冗余项结合正常项的方式躲避检测冗余项的动态仿真分析, 使 VeriTrust 检测失效, 称为 DeTrust。

2.3 静态逻辑分析

静态分析的检测方法, 无需考虑动态验证工具的测试覆盖率, 对网表文件的每一个信号、每一个逻辑门进行分析, 确保检测的完备性。研究人员根据对网表文件提取的不同特征, 提出了多种静态检测方法, 包括基于节点翻转概率特征检测, 基于可测试性特征检测, 基于硬件结构特征检测等。

2.3.1 基于节点翻转概率特征的静态检测方法

Waksman 等[28]通过总结常见的硬件木马特征, 发现硬件木马触发信号对输出信号的影响很弱, 造成了硬件木马电路输出节点的翻转概率较低, 由此提出了一种基于节点翻转概率特征的检测方法“FANCI”。该方法将输入信号对输出信号的影响程度量化为自定义的控制值(Control Value, CV), CV =

$counter/size(T)$。其中，$counter$ 为输入信号变化引起输出信号变化的个数，$size(T)$ 为真值表大小。FANCI 计算每一个输入信号对输出信号的控制值，并用控制值来量化输出节点翻转概率，通过选择合适的阈值，能够区分输出信号中难以控制的硬件木马信号。然而，对于时序逻辑触发的硬件木马电路，FANCI 无法计算出实际的控制值大小。因此，针对时序逻辑触发的硬件木马，FANCI 缺乏有效的检测性。

2.3.2 基于可测试性特征的静态检测方法

针对这种无法衡量时序逻辑触发特征的问题，Salmani[29]提出一种基于网表可控性和可观性分析的检测方法 COTD(Controllability and Observability hardware Trojan Detection)。可控性和可观性分析同时考虑了组合逻辑电路和时序逻辑电路。利用硬件木马低触发概率导致其内部节点可测试性较差的特性，寻找疑似硬件木马信号。可测试性特征包括时序逻辑和组合逻辑可控性(Sequential/Combinational Controllability, SC/CC)，时序逻辑和组合逻辑可观性特征值(Sequential/Combinational Observability, SO/CO)。通过 k 均值聚类算法对信号可测试性特征进行分类，可以准确区分硬件木马电路与正常电路。该方法利用了现有 EDA 工具的功能，效率较高，很多研究人员在此基础上提出了更多特征检测方法。

Xie 等[30]提出了将硬件木马与正常电路之间的聚类簇间距离作为重要特征，结合电路规模，使用 SVM 分类器进行训练，可准确识别硬件木马。文献[31]利用统计公式将可测试性特征进行了重组，提高了检测的准确性。文献[32]利用边界模型检查对基于可测试性特征的分类结果进行处理，降低了误检率。文献[33]使用基于可控性特征的检测结合侧信道和功能测试定位木马的载荷部分。文献[34]将可测试性特征结合网表结构特征生成新的特征向量，使用两种监督分类器对特征向量进行聚类，识别硬件木马。文献[35]提出了一种扩充数据集技术，使用 4 种监督分类器进行训练，能够根据可测试性特征更好地区分硬件木马和正常电路。文献[36]利用差分放大器对可测试性特征进行重定义，差分放大了特征值。采用 k 均值聚类方法对该特征值进行分类，进一步提高了检测准确性。

基于可测试性特征的检测方法在检测范围和精度上有很大优势，但该方法过于依赖可控性特征值和可观性特征值。硬件木马设计人员通过优化电路结构改变可测试性特征，可躲避基于可测试性特征的检测方法，如 DeTest。

2.3.3 基于硬件结构特征的静态检测方法

基于节点翻转概率和基于可测试性特征的检测方法针对的是硬件木马的低触发概率对内部节点或输出信号的影响。除此之外，还可以针对硬件木马实现低触发概率的电路结构特征提出检测方法。这种方法无需依赖度量值以及机器学习的分类器，提高了检测的准确率。

(1) 基于线网结构特征的静态检测方法

Oya 等[37]最早提出了基于硬件结构的检测方法，从硬件木马网表文件中提取出的 9 种线网特征，并赋予每种特征相应的权值。并结合动态仿真分析，提高了检测的准确率。Hasegawa 等[38]提出了一种通用方法构造线网特征，用基本逻辑门参数包括两级扇入个数、距寄存器输入最小级数、距寄存器输出最小级数、距主输入最小级数、距主输出最小级数构造 5 维特征向量，并通过分类器进行检测识别。但在检测过程中发现硬件木马的边界线网存在一定的漏检率。在此基础上，文献[39]扩充到 51 个线网特征，并结合随机森林算法优选出 11 个线网特征构造特征向量。文献[40]使用基于反向神经网络训练的分类器，对边界线网进行补充分析，将部分目标的检测准确率提高到了百分之百。

(2) 基于自环结构特征的静态检测方法

Yao 等[41]提出了一种基于节点连接关系特征的检测方法。只保留寄存器以及寄存器之间的连接信息，构建触发器层级信息流图(FF level information flow graph, IFG)，并基于自循环节点组数量、节点入度、自循环组数量与入度之和、节点出度 4 个特征对 IFG 进行检测。此方法的优点是建立了 IFG，将复杂的网表抽象为简单的有向连接图，大大提高了检测效率。Chen 等[42]在此基础上将 IFG 结构分析与节点翻转概率分析相结合，提高组合逻辑触发硬件木马的检测准确率。但该方法具有一定的误检率，会把乘法器中的一些器件误识别为可疑硬件木马。

(3) 基于逻辑门结构特征的静态检测方法

Chen 和 Liu[43]发现硬件木马为了实现低触发概率，组合逻辑门需按照特定结构连接。在与门(AND)、或门(OR)、与非门(NAND)、或非门(NOR)4 种基本的组合逻辑门可实现的 16 种两级连接中，构成低触发概率的有 8 种结构，即 AONN 检测模板。通过检测目标电路中低触发 AONN 模板，并对组合逻辑连接关系进行量化来识别硬件木马触发电路。之后，Liu 等[44]又提出了分别构建时序逻辑门特征和组合逻辑门特征，结合可测试性特征构造新的特征

向量。文献[45]利用基于权重的聚类分析对逻辑门特征值进行分类, 提高了检测的准确率。以上方法由于只针对 AONN 四种类型逻辑门提取特征, 忽略了时序逻辑连接, 如果在触发电路中引入D触发器, 在不改变原始木马功能的情况下, 破坏 AONN 模板匹配, 造成一定的漏检率。

通过对不同的基于网表文件的检测方法进行研究, 可以发现现有检测方法中都存在一定的漏检或误检。动态验证受限于第三方测试工具, 输入的测试向量覆盖不全面会影响检测结果。基于可测试性特征的静态检测方法依赖于特征值的量化。构建的多种硬件结构特征不具有通用性, 木马的时序逻辑连接和组合逻辑连接的触发方式不同, 无法统一进行分析。因此本文提出一种基于网表控制流分析的硬件木马检测方法, 根据其逻辑门类型及连接关系建立统一的节点控制流图。可有效检测低概率触发的硬件木马包括时序逻辑触发硬件木马和组合逻辑触发硬件木马, 以及新型的抗检测硬件木马。

3 基于网表控制流分析的硬件木马检测方法

本文中, 网表控制流的定义是: 电路网表中组合逻辑门之间, 及组合逻辑门和时序逻辑门之间数据可控性传递的数据流。

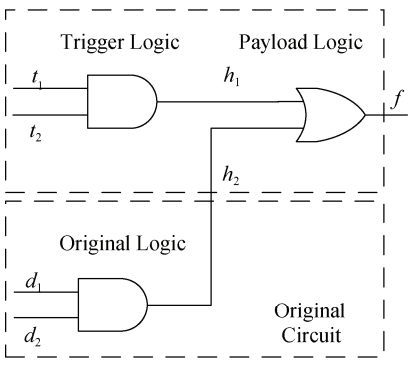
本节首先分析了典型的硬件木马检测方法存在的脆弱性, 然后针对性地提出了基于网表控制流分析的硬件木马检测方法, 最后介绍了基于网表控制流分析的硬件木马检测流程。

3.1 现有检测方法脆弱性分析

3.1.1 基于可测性特征的检测方法脆弱性分析

Salmani[29]首次提出了硬件木马低概率触发特征与电路的可控性与可观性之间的关联性, 基于可测性特征, 文献[30-36]又提出了很多检测方法。该类检测方法均依赖于硬件木马电路的可测性, 包括组合逻辑可控值(*CC1/0*)和组合逻辑可观值 *CO*。采用无监督聚类分析方法或其他机器学习算法对待测电路的可测性值进行聚类即可检测硬件木马。然而经过本文分析发现, 该检测方法具有一定脆弱性。硬件木马设计者通过对木马电路进行简单修改即可降低硬件木马电路的可测性值, 使基于可测性值分析的硬件木马检测方法失效。信号的可测性特征包括可观性和可控性, 两种特性均存在一定脆弱性, 示例如下。

图 2(a)为硬件木马电路, 图 2(b)为其对应的真值表。电路中 h_1 为触发信号, h_2 为目标信号, f 为输出信号。当满足触发条件 t_1、t_2 均为 1 时, 硬件木马触发, 无论目标信号 h_2 为何值, 均将输出信号 f 设置为 1。假设检测过程中硬件木马未触发, 触发信号 h_1 一直为 0, 其 *CC0* 值较高, 根据每个信号的可测性值, 将 h_1 标记为硬件木马电路。若硬件木马设计者在目标信号 h_2 与输出信号 f 相等时将 h_1 置为 1[图 2(b)椭圆所圈状态], 此时整个硬件木马电路逻辑并未改变, 但是触发信号 h_1 的 *CC0* 值可以有效降低, 与正常信号 h_2 的 *CC0* 值类似。由此可知, 基于可控性值分析, 无法检测硬件木马触发信号 h_1。



(a) 2输入触发硬件木马电路

Truth Table

t_1	t_2	d_1	d_2	h_1	h_2	f
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	1	1
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	0	1	1

t_1	t_2	d_1	d_2	h_1	h_2	f
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	1	1
1	1	0	0	1	0	1
1	1	0	1	1	0	1
1	1	1	0	1	0	1
1	1	1	1	1	1	1

(b) 硬件木马触发电路真值表

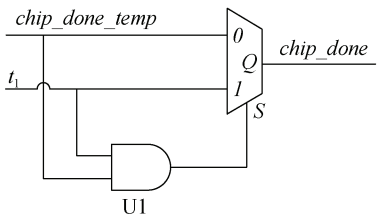
图 2 基于可控性分析的硬件木马检测脆弱性

Figure 2 Vulnerability analysis of Controllability based HT detection methods

同理, 为了降低目标信号的可观性值 *CO*, 根据可测性特征值 SCOAP 算法中计算可观值 *CO* 的公式如下:

$$CO = \min(branchCO)$$

图 3(a)中 t_1 为 *CO* 值高的硬件木马信号, 选取任一 *CO* 值低的正常信号如 *chip_done*, 插入图 3(b)中逻辑电路。当 t_1=*chip_done* 时, 由 t_1 驱动输出, 在未改变电路逻辑功能的前提下, 可有效降低信号 t_1 的 *CO* 值。使基于可观性特征值的检测方法失效。



(a) 降低可观性值的逻辑电路

Truth Table

t_1	chip_done_temp	chip_done	驱动信号
0	0	0	chip_done_temp
0	1	1	
1	0	0	
1	1	1	t_1

(b) 降低可观性值的逻辑电路真值表

图 3 基于可观性分析的硬件木马检测脆弱性

Figure 3 Vulnerability analysis of Observability based HT detection methods

3.1.2 AONN 检测脆弱性分析

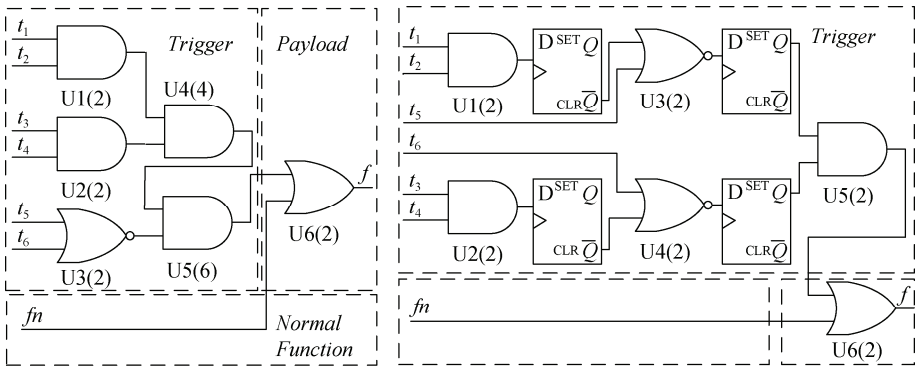
Chen 和 Liu[43]针对常见低概率触发硬件木马的结构提出了针对组合逻辑门的 AONN 检测方法。提出了 *Rare Value* (*RV*)对组合逻辑门进行量化, 相邻组合逻辑门之间满足 8 种模板即可将 *RV* 值向后续逻辑电路传递, 硬件木马触发信号的 *RV* 值比正常信号要大。然而经过本文分析发现, AONN 只将常见硬件木马组合逻辑门进行特征分析, 硬件木马设计人员如在组合逻辑门之间插入 D 触发器, 即可明显降低硬件木马的 *RV* 值。

图 4(a)为 6 输入触发硬件木马电路, 图 4(b)为对其进行抗检测优化后的电路。逻辑门名称后括号内数值为其 *RV* 值。电路中 t_1~t_6 为触发信号, 只有 t_1~t_4 均为 1 且 t_5 和 t_6 均为 0 时, 硬件木马才会触发, 触发概率为 0.016。当用 AONN 方法进行检测时, U5 的 *RV* 值为 6, 明显高于其他逻辑门, 所以被识别为硬件木马。图 4(b)所示为改进后的硬件木马电路。通过在低概率触发的组合逻辑门之间插入 D 触发器, 破坏木马电路特征, 使其不符合 AONN 判定模板, 有效降低其 *RV* 值, 躲避 AONN 检测。

3.1.3 FASTrust 检测脆弱性分析

Yao 等[41]首次将网表以 DFF 触发器为节点抽象为信息流图 IFG 进行硬件木马检测。对于时序逻辑触发的硬件木马(如计数器触发或深度状态机触发的电路结构), 其信息流图 IFG 中包含多个相连成组的自循环节点, 或多个多入度自循环节点相连的节点组。通过此特征可以检测时序触发的硬件木马。

然而经过本文分析发现, 由于 FASTrust 以 D 触发器为节点抽象生成的信息流图 IFG, 只考虑了 D 触发器之间的连接关系, 忽略了 D 触发器之间的组合逻辑门, 误检率很高。同时, 其针对计数器等时序触发电路, 以自循环 D 触发器节点数目为特征进行量化并不准确。硬件木马设计人员可以通过用两个 D 触发器双向连接结构代替 D 触发器自循环结构, 不需自循环结构即可构成多位计数器或有限状态机。



(a) 6输入触发硬件木马电路

(b) 抗检测优化后的硬件木马电路

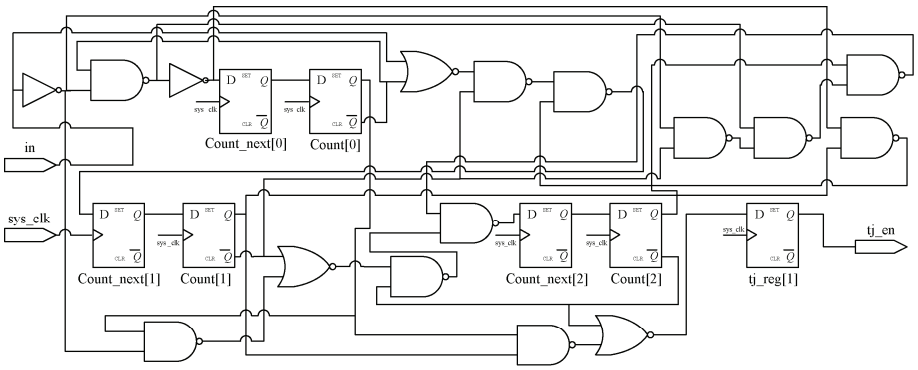
图 4 AONN 硬件木马检测脆弱性分析

Figure 4 Vulnerability analysis of AONN HT detection

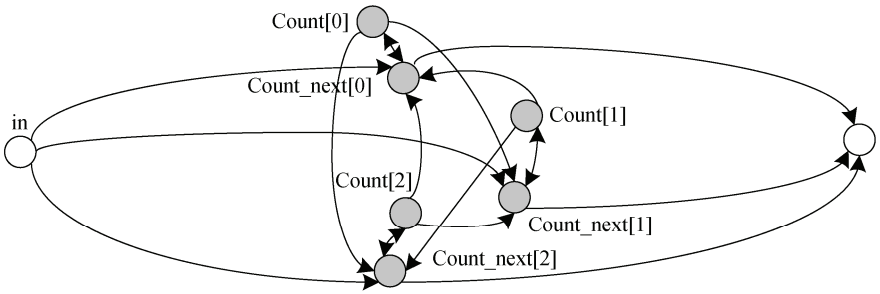
图 5(a)为通过 2 个双向连接的 D 触发器进行计数的硬件木马电路, 图 5(b)为根据 FASTrust 生成的信息流图 IFG。可以看出在提取的 8 个节点中, 不存在自循环节点, 因此, FASTrust 通过自循环节点组大小及自循环节点组入度为特征的硬件木马检测方法无法检测出该类木马, 造成漏检。

3.2 基于网表控制流分析的检测方法

对现有典型硬件木马检测方法进行脆弱性分析可以看出通过对可测性特征值进行聚类、单独分析组合逻辑特征或简单提取时序逻辑门连接关系进行



(a) 无自循环结构的3bit计数器电路



(b) 无自循环结构3bit计数器信息流图IFG

图 5 FASTrust 硬件木马检测脆弱性分析

Figure 5 Vulnerability analysis of FASTrust

触发硬件木马的检测存在很多漏洞。硬件木马设计人员很容易通过组合逻辑、时序逻辑混合设计的方法进行抗检测设计达到隐蔽的目的。因此, 本文提出了一种对网表进行更精细刻画的检测方法, 统一考虑组合逻辑及时序逻辑对电路可控值的影响, 针对隐蔽性强的抗检测硬件木马电路可有效检测。

为了统一组合逻辑电路与时序逻辑电路, 本文构建了节点控制流图(Node Control Flow Graph, NCFG)对电路信号传递特征进行更精细的刻画, 以实现低概率触发电路的检测。节点控制流图是描述网表电路数据传递路径的模型。节点控制流图由节点和边组成。整个电路的输入输出端口及具有记忆功能的 D 触发器为节点, 节点之间的组合逻辑电路抽象为边, 边是具有方向的, 代表了信号控制与驱动的方向。同时, 边也有值, 量化了前级信号对后级信号的控制能力。

本文中, 任意两个节点之间的路径为一条控制流, 控制流具有初始方向, 控制流传递过程中控制值会增长, 控制方向会改变, 因此可通过一个数组定义该控制流的每一段的控制值。当一个节点受一个控制流或多个控制流驱动时, 代表其电平受一个或多个节点影响, 通过节点汇聚值定义该节点被控值的大小。

3.2.1 针对组合逻辑电路的控制流分析

组合逻辑电路的特点是任意时刻其输出仅仅取决于该时刻的输入, 与电路原来的状态无关。硬件木马经常使用组合逻辑电路构造低概率触发部分。影响其输出结果的输入个数越多, 其触发概率越低。在控制流分析中, 其触发信号的可控值越大, 如常见的输入模板匹配型触发电路(即当多个信号同时为某一特定数值时激活硬件木马)。

与(AND)、或(OR)、非(INV)3 种组合逻辑门是组合逻辑的基本构成单元, 通过以上 3 种组合逻辑可以组成与非(NAND)、或非(NOR)、异或(XOR)、异或非(NXOR)等组合逻辑。所有组合逻辑的控制流都可以通过以上 3 种门的控制流进行计算。图 6 中第二列为传统组合逻辑可控值的计算方法, 从信号控制流的角度考虑, 信号经过一个组合逻辑门, 其可控性会发生变化, 且输出信号的可控性与输入信

逻辑门	组合逻辑可控值定义	*CF0*	*CF1*
a, *b* → *z*	$CC0(z) = min(CC0(a), CC0(b)) + 1$ $CC1(z) = CC1(a) + CC1(b) + 1$	*0*	*1*
a, *b* → *z*	$CC0(z) = CC0(a) + CC0(b) + 1$ $CC1(z) = min(CC1(a), CC1(b)) + 1$	*1*	*0*
z → *zn*	$CC0(zn) = CC1(zn) + 1$ $CC1(zn) = CC0(zn) + 1$	*CF1*	*CF0*

图 6 基础组合逻辑门的可控性定义

Figure 6 Controllability definition of basic combinational logic

号有关。以 2 输入 AND 门为例, 其输出信号 *z* 逻辑 0 可控值 *CC0*(*z*)与输入信号 *a*、*b* 的逻辑 0 可控值的最小值有关。因此, 从信号控制流角度分析, 去除固定增加值 1, 其可控值并没有发生累加。反之, 输出信号 *z* 逻辑 1 可控值等于输入信号 *a*、*b* 的逻辑 1 可控值之和。从信号控制流角度分析, 其可控值增加, 即输出值受输入值影响更大, 输出值更不易控制, 更可能为隐蔽性强的硬件木马触发信号。

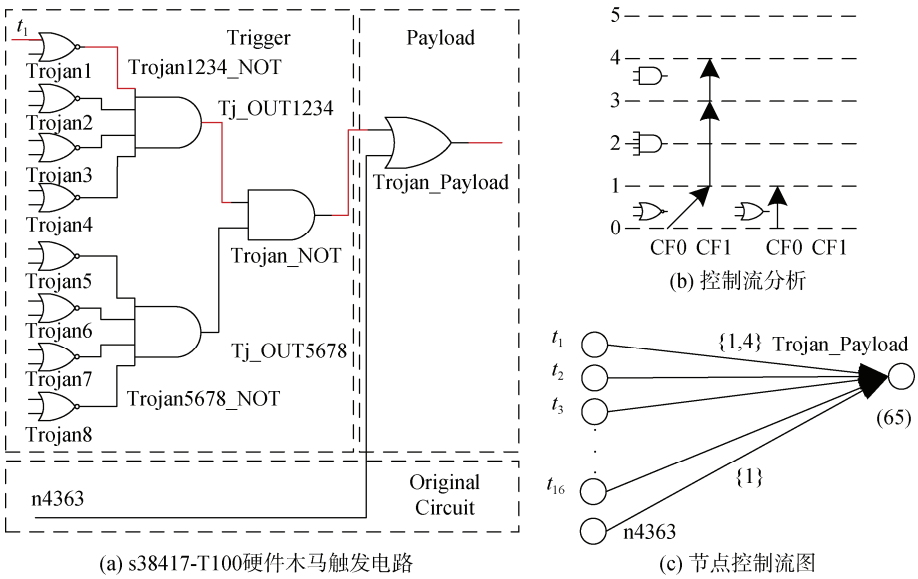
由以上分析可知, AND 门是逻辑 1 可控值增加方向, OR 门是逻辑 0 可控值增加方向。本文定义 AND 门控制流增长方向为 CF1 (Control Flow 1), OR 门控制流增长方向为 CF0 (Control Flow 0)。当一个 AND 门连接另一个 AND 门, 其控制流在 CF1 方向增长。同理, 当一个 OR 门连接另一个 OR 门, 其控制流在 CF0 方向增长。经过反相器控制流增长方向发生切换。传统计算方法每经过一个反相器 INV 其可控值都会增长 1, 但从数学角度分析, 其可控性并未改变。针对硬件木马检测, 通过寻找连续增长的控制流, 可以寻找可控值较大的信号。

因此, 本文提出的节点控制流图有如下特征。以待测电路输入、输出端口及 D 触发器为节点, 当信号从一个节点 *node_start* 开始经过组合逻辑到达另一节点 *node_stop* 截止时, 以控制流数组 *flow_list* 来定量描述信号流经组合逻辑门时控制流的增长情况。以 *flow_start* 描述控制流的起始方向, *flow_stop* 描述控制流的终止方向。当流经的第一个逻辑门为 AND 门时, 起始方向 *flow_start* 为 CF1; 当流经的第一个逻辑门为 OR 门时, 起始方向 *flow_start* 为 CF0; 若第一个逻辑门为反相器, 初始方向由下一个 AND 门或 OR 门决定。当控制流增长方向切换时, 控制流数组 *flow_list* 新增一个元素。

图 7 所示为硬件木马检测平台 TrustHub[46-47]上电路 s38417-T100 的触发电路。由图 7(a)可知, 其触发电路为组合逻辑门构成的比较器, 触发输入个数为 16, 触发概率为 1.4243e-70。以图 7(a)中红色信号为例进行控制流分析, 其驱动的第一个逻辑门为 NOR 门, 即 OR 门串接 INV, *flow_start* 为 0, 经过 NOR 门后, 控制流方向为 CF1。第 2 个逻辑门 AND 门为 CF1 增加方向, 因此控制流在 CF1 方向增加。同时, 组合逻辑门的可控性与其输入个数有关。图 7(a)中 *Trojan1234_NOT* 为 4 输入 AND 门, 若采用 2 输入 AND 门, 将经过 2 个 AND 门。因此, 控制流增加量应为 $\log_2(IN)$, *IN* 为组合逻辑输入个数。图 7(b)所示为红线穿过组合逻辑门时控制流, *flow_ start* 为 0, *flow_stop* 为 0, *flow_list* 为{4,1}。其余 15 个输入触发节点与 *Trojan_payload* 之间的控制流与红线所示控制流相同。被控信号 *n4263* 与 *Trojan_payload* 之间 *flow_list* 为{1}。*Trojan_payload* 节点为输出节点, 节点汇聚的受控值为与其相连的控制流数组 *flow_list* 最大值之和, 本例中 *Trojan_ Payload* 节点汇聚值 *P* 为 65。

3.2.2 针对时序逻辑触发的控制流分析

时序逻辑电路的特点是其任意时刻的输出不仅取决于当时的输入信号, 而且还取决于电路原来的状态, 即时序电路具有记忆性特征。硬件木马经常采用时序逻辑电路构造其触发器, 通过增加时间记忆深度、或结合逻辑触发难度与时间记忆深度构造低概率触发电路, 常见的有计数器、深度状态机等。

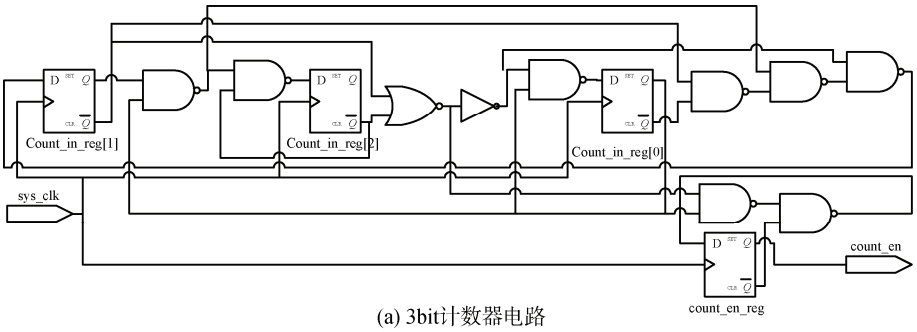


(a) s38417-T100硬件木马触发电路 (b) 控制流分析 (c) 节点控制流图

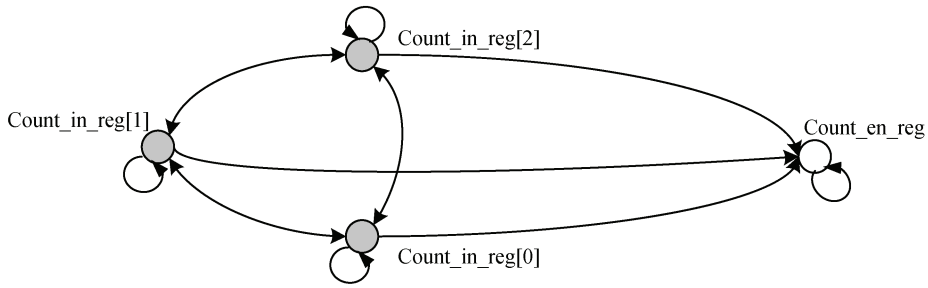
图 7 针对组合逻辑电路的控制流分析

Figure 7 Control Flow analysis of combinational logics

以图 8 所示 3bit 计数器为例, 图 8(a)是计数器的电路图, 图 8(b)为其节点控制流图。其计数单元为 3 个具有自循环结构的 D 触发器, 和一个驱动输出的 D 触发器。4 个 D 触发器均为自循环节点, 其中存储其计数状态的 3 个 D 触发器之间互相驱动, 由此可知, 时序逻辑触发电路触发概率与两两相连的自循环节点个数有关。将两两相连的自循环 D 触发器组成一个节点组, 节点组规模越大, 计数位数越多, 触发概率越低。



(a) 3bit计数器电路



(b) 3bit计数器节点控制流图

图 8　3bit 计数器电路及其节点控制流图

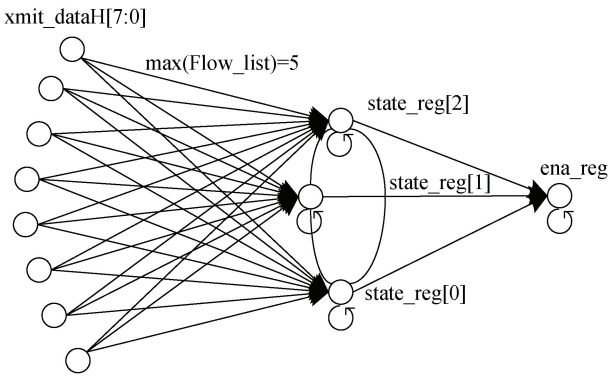
Figure 8　Schematic and NCFG of 3-bit counter

时序逻辑触发的木马电路不仅跟记忆深度有关, 还跟触发状态有关。图 9 所示为 TrustHub[15,47]上硬件木马电路 RS232-T900, 图 9(a)为其硬件木马触发电路 RTL 代码, 图 9(b)为其节点控制流图。当 8 位输入信号 xmit_data[7:0]依次为 *AA*、*55*、*22*、*FF*、*11* 时, 激活硬件木马。

```
always @ (negedge sys_rst_l or posedge xmitH) begin
    if (~sys_rst_l) state_DataSend<=0;
    else begin
      case (state_DataSend)
      s_idle: begin
                if(xmit_dataH==8'haa)
                  state_DataSend<=s_count1;
                else
                  state_DataSend<=s_idle;
              end

      s_count1: begin
                if(xmit_dataH==8'h55)
                  state_DataSend<=s_count2;
                else
                  state_DataSend<=s_idle;
              end
      s_count2: begin
                if(xmit_dataH==8'h22)
                  state_DataSend<=s_count3;
                else
                  state_DataSend<=s_idle;
              end
      s_count3: begin
                if(xmit_dataH==8'hff)
                  state_DataSend<=s_stop;
                else
                  state_DataSend<=s_idle;
              end
      s_stop: begin
                if(xmit_dataH==8'h11)
                  state_DataSend<=s_idle;
                else
                  state_DataSend<=s_stop;
              end
    endcase
  end
end
```

(a) RS232-T900 RTL代码



(b) RS232-T900节点控制流图

图 9　RS232-T900 触发电路 RTL 及节点控制流图

Figure 9　RTL and NCFG of RS232-T900 HT trigger

将其两两相连的自循环节点 *state_reg[0,1,2]*组为一组, 节点组规模大小为 3, 驱动其的节点为 *xmit_data[7:0]*, 驱动节点与受控节点的每条控制流数组中最大值 *Max(flow_list)*为 5。该自循环节点组的控制流汇聚值 *G* 为 120。

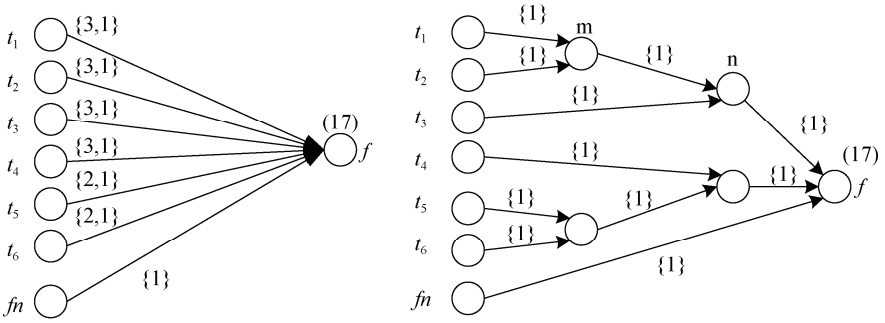
3.2.3　针对新型抗检测木马的控制流分析

常见的硬件木马检测方法分别对组合逻辑与时序逻辑触发的硬件木马进行分析检测, 建立硬件木

马特征库并进行特征检测识别。然而, 硬件木马设计者很容易将单纯的组合逻辑木马或时序逻辑木马进行抗检测优化, 例如在 3.1 节中所示, 通过在组合逻辑木马中插入触发器、在时序逻辑低概率触发的木马中将自循环触发器替换为双向连接的非自循环触发器等手段, 破坏现有硬件木马组合逻辑和时序逻辑特征, 躲避硬件木马检测。

针对现有检测木马的不足和抗检测木马的特征, 本文提出的检测方法通过以下方法检测新型抗检测木马。

以 3.1 节中图 4 所示通过插入 D 触发器切断组合逻辑控制流的新型抗检测木马为例。建立节点控制流图时, 也应同时考虑 D 触发器前后的组合逻辑, 即控制流可以通过 D 触发器进行传递, 控制流初始值与驱动该节点的前一段控制流有关。在计算两个节点间控制流数组 *flow_list* 时, 若控制流初始方向 *flow_start* 与驱动该节点的前一控制流方向一致, 控制流数组中第一段应进行累加。3.1 节图 4 节点控制流图如图 10 所示。图 10(a)为常规木马的节点控制流图, 节点 f 受节点 t_1~t_6 驱动, 节点汇聚值 P 为 16。图 10(b)为通过插入 D 触发器进行抗检测优化后硬件木马的节点控制流图, 若不考虑起始控制流的初始值, 节点 f 的节点汇聚值为 3, 与正常节点类似。因此, 为了检测组合逻辑混合型抗检测硬件木马, 在进行控制流分析时, 应考虑控制流的传递性。



(a) 原始电路的节点控制流图　　(b) 抗检测优化后电路节点控制流图

图 10　6 输入触发硬件木马电路节点控制流图

Figure 10　NCFG of 6 inputs triggered HT

以信号 t_1 驱动的控制流为例, 第一个经过的逻辑门为 AND 门, 起始控制流方向为 CF1; 经过 D 触发器反向输入后, 控制流方向为 CF0; 后经过 NOR 门, 控制流未中断, 方向仍为 CF0。因此当控制流未中断时, 应将其前序控制流进行累加。同时, 节点汇聚值 P 与驱动其的节点个数及节点控制流组有关。除定义两节点间控制流 *flow_list* 外, 还需定义累计控制流数 *flow_accumulated*。以图 10(b)为例, 节点 m 受节点 t_1、t_2 驱动, 且经过节点 m 后, 2 条控制流均未中断, 因此节点 m 累计控制流数 *flow_accumulated* 为 2。节点 m 对节点 n 受控汇聚值进行传递。

$$P(N) = P(M) + flow\_accumulate * flow\_list(m\text{-}n)=4$$

同理, 节点 t_5 流经节点 n 后, 控制流也未中断。因此节点 n 累计控制流数 *flow_accumulated* 为 3。依次计算 f 节点汇聚值为 17, 与未插入 D 触发器的电路结构一致。

由 3.1 节可知, 非自循环 D 触发器也可以构成计数器。硬件木马检测除了检测自循环节点 *self_loop_node* 外, 还应检测是否存在形成环状结构的 D 触发器组 *circle_nodes*。如图 5 所示两两相连的 D 触发器。时序触发电路的本质是通过增加存储性器件个数, 增加时间复杂度来降低触发率。状态之间的转换与当前状态及输入信号有关, 因此存储状态的触发器输出会影响其输入, 形成自循环结构, 抗检测木马可以通过插入 D 触发器破坏常见状态机的自循环节点特征, 但根据状态机的原理, 其必然存在环状结构进行状态的存储与转换, 图 5 中电路可实现状态机功能, 虽然没有自循环节点, 但存在 D 触发器构成的环。对于构成环状结构的 D 触发器组, 环内的每个 D 触发器的输入及输出视为环的输入及输出。环之间如果构成两两相连, 则组为自循环节点组。

3.3　基于控制流分析的硬件木马检测

本文提出的基于网表控制流分析的硬件木马检测方法整体框架如图 11 所示, 主要包括两个阶段: 节点控制流图生成阶段与硬件木马检测阶段。

3.3.1　节点控制流图生成阶段

节点控制流图生成阶段是硬件木马检测的第一步。由 3.3 节可知, 根据组合逻辑木马、时序逻辑木马及新型抗检测木马的特征, 生成的控制流图需包含以下特征, 首先其节点为输入输出端口或 D 触发器, 其次节点之间存在通过组合逻辑门相连的控制流。每一条控制流均规定了控制流的起点、终点、

起始方向、终止方向和控制流数组, 依次为 *node_start*、*node_stop*、*flow_start*、*flow_stop*、*flow_list*。

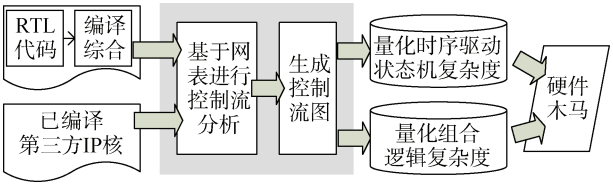


图 11 基于网表控制流分析的硬件木马检测流程

Figure 11 Procedure of control flow based HT detection

算法 1. 节点控制流图的生成

输入: 被测电路门级网表 *v*

输出: 依据电路连接关系生成的节点控制流图 NCFG, 其中每一条控制流 *g* = {*node_start, node_stop, flow_start, flow_stop, flow_list*}

1: 网表 *v* 中输入端口、输出端口及 D 触发器作为节点集合 *N*
2: FOREACH *n* in *N* Do
3: 初始化当前控制流 *g*
4: *node_start* = *n*;
5: CASE(该控制流第一个驱动的逻辑门):
6: AND gate: *flow_start* = 1
7: OR gate: *flow_start* = 0
8: INV gate: 记录反向标志位
9: FOREACH 驱动的逻辑门
10: If AND gate
11: If 控制流方向为 CF1
12: *flow_list* 当前元素增加
13: Else
14: *flow_list* 新增一个元素
15: 控制流方向改变
16: If OR gate
17: If 控制流方向为 CF0
18: *flow_list* 当前元素增加
19: Else
20: *flow_list* 新增一个元素
21: 控制流方向改变
22: If INV gate
23: 控制流方向改变
24: *flow_list* 当前元素不变
25: If *m* in *N* 控制流到另一节点终止
26: *node_stop* = *m*
27: *flow_stop* = 当前控制流方向
28: Return *g* 返回该条控制流 *g*
29: Until 网表 *v* 中连接全部计算完毕

算法 1 采取深度追踪的方法构建控制流图, 即从一个节点开始, 寻找该节点驱动的控制流, 到驱动另一个节点中止, 直到所有的路径搜索完毕。

算法的输入为被检测电路门级网表 *v*, 它描述了电路逻辑门的组成及连接关系。算法的输出为节点控制流图 NCFG, 包含很多控制流。其中每一条控制流 *g* 包括控制流起点、控制流终点、起始方向、终止方向和控制流组, 记为 *{node_start, node_stop, flow_start, flow_stop, flow_list }*。

对于任意节点 *n*, 对其驱动的逻辑门进行深度优先搜索, 如果为 AND 门, 该控制流起始方向 *flow_start* 为 1; 如果为 OR 门, 该控制流起始方向 *flow_start* 为 0; 如果为 INV 反相器, 记录反向标志位, *flow_start* 由反相器输出相连的逻辑门决定。

深度搜索过程中, 如果控制流增长方向与穿过的组合逻辑门控制流增长方向一致, 曾控制流数组当前元素增加, 否则控制流数组 *flow_list* 新增一个元素。反相器只改变控制流方向, 不改变控制流数组大小。当遇到另一个节点 *m*, 控制流终止, 返回该控制流 *g* 的参数, *node_stop* 为 *m*, *flow_stop* 为当前控制流方向, *flow_list* 为当前控制流数组。

3.3.2 硬件木马检测阶段

判断被测电路门级网表是否包含疑似木马电路, 需要根据控制流图进行硬件木马检测。

硬件木马的显著特征为超低概率触发, 由 3.3 节可知, 通过组合逻辑、时序逻辑以及两者融合可以设计出低概率触发电路。基于提取的控制流图中的信息, 定量衡量被测电路是否具有低概率触发模块。

(1) 时序逻辑触发的硬件木马检测

为了降低硬件木马的触发概率, 时序逻辑触发的硬件木马通过增加状态寄存器数目和组合逻辑复杂度来实现。因此, 硬件木马实际上也是一种有限状态机, 其触发概率远远低于普通状态机。根据节点控制流图进行硬件木马检测, 实际是检测状态机数量, 及量化每个状态机的触发概率。

算法 2. 时序逻辑触发的硬件木马检测

输入: 控制流图 NCFG

输出: 自循环节点组数量及每个组汇聚值 *G*

1: FOREACH *g* in NCFG Do
2: *start_node* 标记为已访问节点
3: If *stop_node* =*start_node*
4: *self_loop_nodes* 新增 *start_node* 标记为自循环节点
5: If *stop_node* 为已访问节点

6: 根据 *stop_node* 提取 *circle_nodes* 搜索成环的节点组

7: FOREACH node *m* in *self_loop_node*s & *circle_nodes* DO

8: If node *m* bi-direction connect node *n* in *self_loop_nodes* or *circle_nodes*

9: 将 *m n* 组成 *self_loop_group*

10: FOREACH *group* in *self_loop_group*

11: *G* = size of *group*

12: FOREACH node *s* in group

13: 计算节点的节点控制值 *P*

14: *G* = *G* + *P*

15: *G* 为节点组 *group* 的控制流汇聚值

算法 2 根据控制流图查找网表中的有限状态机数目, 并量化每个状态机控制流汇聚值 *G*。有限状态机通过状态寄存器记录当前状态, 根据组合逻辑进行状态转换。因此每个状态寄存器具有自循环特征, 同一状态机不同状态寄存器之间具有双向连接特征。

算法的输入为被检测电路门级网表抽象生成的节点控制流图 NCFG。

对于节点控制流图中的每一条路径, 如果其 *start_node* 与 *stop_node* 相同, 说明该节点为自循环节点, *self_loop_nodes* 集合中加入该节点。

由 3.1 节可知, 新型抗检测木马可以通过 2 个双向连接的寄存器、或多个寄存器连接形成的环来记录状态并控制状态转换。因此, 除了检测自循环节点, 还应检测是否存在环状结构的寄存器组。遍历节点控制流图 NCFG 中的每一条路径, 将其 *start_node* 标记为已访问节点。如果某一路径中 *stop_node* 为已标记节点, 说明该节点存在一个环中, 根据该节点连接关系, 提取一个环, *circle_nodes*。自循环节点本质上是由一个节点组成的环。

自循环节点、环状节点组均可组成状态机。对于每一个自循环节点及环状节点组, 如果其跟另外一个自循环节点或环状组之间存在双向连接, 则将其组成一个 *self_loop_group*, 即一个状态机。

对任一 se*lf_loop_group*, 其触发概率与节点组汇聚值 *G* 有关。*G* 与节点组规模及每一个节点的汇聚控制值有关, 节点汇聚控制值在算法 3 组合逻辑触发的硬件木马检测中进行详细计算。

(2) 组合逻辑触发的硬件木马检测

为了降低硬件木马的触发概率, 组合逻辑触发的硬件木马通过增加组合逻辑复杂度来实现。即一个节点的数值受多个节点控制。硬件木马检测需要量化每个节点的可控性。

算法 3. 组合逻辑触发的硬件木马检测

输入: 节点控制流图 NCFG

输出: 每个节点控制流汇聚值 *P*

1: REPEAT

2: FOREACH node *n* in NCFG Do

3: FOREACH *g* is NCFG

4: If *stop_node* = *n*

5: If *n* in *self_loop_nodes/circle_node*

6: If *start_node* not in *self_loop_nodes/circle_node*

7: *root_node*(*n*)前序节点新增元素 *start_node*

8: ELSE

9: *root_node*(*n*)新增元素 *start_node*

10: If *root_node*(*n*) = []

11: *P*(*n*) = 0

12: *n* 标记为已计算

13: If all nodes in *root_node*(*n*)已计算

14: FOREACH *r* in *root_node*(*n*)

15: If *r* 到 *n* 控制流未中断

16: *P*(*n*) = *P*(*r*) + *flow_list*(*r-n*) **flow_accumulated*(*r*)

17: *flow_accumulated*(*n*) += *flow_accumulated*(*r*)

18: Else

19: If *r* 到 *n* 起始控制流方向与 *r* 驱动控制流方向一致

20: *flow_list*(*r-n*)(*1*)= *flow_list*(*r-n*)(*1*) * *flow_accumulated(r)*

21: *P*(*n*) = Max(*flow_list*(*r-n*))

22: *n* 标记为已计算

23: Else

24: *P*(*n*) = Max(*flow_list*(*r-n*))

25: *n* 标记为已计算

26: UNTIL 所有节点汇聚值 *P* 均已计算

算法 3 根据节点控制流图计算网表中的每个节点汇聚的控制值 *P*。对于控制流图中的每一个节点 *n*, 搜索它的前序节点, 如果其没有前序节点, 将其标为已计算。如果节点 *n* 为有限状态机一部分, 即为 *self_loop_nodes* 或 *circle_nodes*, 只考虑非自循环节点组对其影响, 因为状态机内部节点之间影响已通过算法 2 状态机规模进行考虑。

当节点 *n* 的前序节点均已计算汇聚值后, 节点 *n* 的汇聚值计算遵循以下要求。

对于每一个前序节点 *r*, 如果节点 *r* 到 *n* 之间控制流未中断, 则节点 *n* 的汇聚值 *P*(*n*)为节点 *r* 的汇聚值 *P*(*r*)加上 *r* 到 *n* 之间控制流的增长。如果节点 *r* 处

本身汇聚了多个节点, 控制流的增长为 *r* 到 *n* 控制流乘以节点 *r* 处汇聚节点数 *flow_accumulated*(*r*)。同时, 节点 *n* 的汇聚节点数增加 *flow_accumulated*(*r*)。

如果节点 *r* 到 *n* 之间控制流中断, 则节点 *n* 的汇聚值为 *r* 到 *n* 之间控制流的最大值, 其中第一段控制流需考虑其是否延续了前序控制流。

4 实验及结果评估

本节通过构建硬件木马检测平台, 对基于网表控制流分析的硬件木马检测方法进行验证。下面详细介绍实验中使用的检测样本、环境搭建、实验结果及数据对比等。

4.1 检测样本

实验中使用的硬件木马检测样本来自硬件木马检测平台 TrustHub 上的良性样本(不含硬件木马)、硬件木马(包括时序逻辑触发硬件木马、组合逻辑触发硬件木马)和新型抗检测木马如 DeTrust、DeTest 等, 如表 1 所示。表中列举了检测样本的端口数量、信号数量、逻辑门数量(包括组合逻辑门数量和时序逻辑门数量)和反相器数量。

表 1 检测样本汇总

Table 1 summary of HTs used in the experiments

检测样本	电路规模					
	Port	Net	Cells	CL	SL	Inv
AES-HTfree	450	450545	424736	417328	6850	120720
RS232-HTfree	23	176	160	118	42	7
PIC16F84 -HTfree	138	3658	3521	3132	387	329
AES-T400	450	425220	424959	418082	6877	121066
AES-T700	450	424626	424496	417559	6937	120803
AES-T900	450	425149	425019	418087	6932	120881
RS232-T300	23	175	157	115	42	7
RS232-T400	23	197	182	140	42	31
RS232-T500	23	331	300	225	75	21
RS232-T600	23	195	269	223	46	40
RS232-T800	23	192	172	130	42	7
RS232-T900	23	256	226	180	46	10
PIC16F84-T100	138	3925	3796	3345	451	409
PIC16F84-T200	138	2934	2761	2411	350	120
PIC16F84-T300	138	3573	3449	3048	401	402
PIC16F84-T400	138	3566	3442	3041	401	401
DeTrust RS232-T800	23	203	183	130	53	7
DeTest RS232-T800	23	175	157	115	42	7

(注: Port 指电路端口数量; Net 指信号数量; Cell 指逻辑门数量; CL 指组合逻辑门数量; SL 指时序逻辑门数量; Inv 指反相器数量)

4.2 检测结果分析

利用 4.1 节所示检测样本, 本文首先根据算法 1 对每个检测样本刻画了节点控制流图。然后基于算法 2、算法 3 对其进行了硬件木马检测。

表 2 为本文提出的基于网表控制流分析的硬件木马检测方法的检测结果, 对比了未植入硬件木马的正常电路和植入多种触发类型的硬件木马电路。列出了节点控制流图的规模, 以及对控制流图进行时序逻辑分析及组合逻辑分析的检测结果。其中, 时序逻辑分析列出了自循环节点组的汇聚值, 组合逻辑分析列出了正常电路节点汇聚值和硬件木马电路节点汇聚值。经过人工检查, 检测样本中的所有硬件木马均能够被有效检测, 即基于网表控制流分析的硬件木马检测漏检率为 0。

4.2.1 节点控制流图刻画

第 3 列为节点控制流图中节点数量, 第 4 列为控制流数目。可以看出, 节点数量等于检测样本中输入、输出节点数量与时序逻辑门数量 *SL* 之和, 符合预期。控制流数量为信号数量 *Net* 的 2~10 倍, 这是因为不同的控制流可能会经过相同的信号线路。并且对于复杂程度高、时序逻辑多的检测样本, 其节点控制流数量更大。

如图 12 所示为检测样本中 RS232-T900 中提取的控制流图, 红色节点为自循环节点, 多个自循环节点组成自循环节点组, 实现有限状态机功能。灰色节点为普通节点。节点之间为有向连接, 2 个节点之间可能存在多条路径, 因此可能存在多条控制流。图 12 中右上角多个节点的控制流汇聚到以 3 个自循环节点组成的自循环节点组中, 为深度状态机触发硬件木马, 自循环节点组规模为 3, 图 12 中左下方为多个自循环节点构成正常状态机, 包含 11 个自循环节点, 图 12 中还包含多个单自循环节点的组, 通常为状态机的输出驱动节点。

4.2.2 时序逻辑分析

根据节点控制流图, 计算自循环节点及环状节点组数量之和 *s_node*, 根据自循环节点及环状节点组连接关系计算自循环节点组数量 *s_group*, 根据每个自循环节点组规模及汇聚的节点控制流大小计算每个自循环节点组汇聚值。可以看出, 正常电路状态机提取的自循环节点组汇聚值 *Gen_Max* 一般不超过 30, 而基于计数器触发或基于深度状态机触发的硬件木马电路其自循环节点组汇聚值为 100 以上, 与正常电路有限状态机自循环节点组汇聚值有明显区别, 因此, 自循环节点组汇聚值判断阈值取 100。硬件木马 RS232-T300 触发电路为 32bit 计数器,

表 2　基于网表控制流的硬件木马检测结果

Table 2　Experiment result of HTs detection based on NCFG

电路类型	检测样本	生成控制流图		时序逻辑分析					组合逻辑分析		
		node	flow	s_Node	s_group	Gen_count/Max	HT_count/Max	Hit	Gen_Max	HT_Max	Hit
不含硬件木马的正常电路	*RS232-HTfree*	65	327	22	10	10/23			8		
	AES-HTfree	7300	764884	0	0	0			8		
	PIC16F84-HTfree	525	39187	285	196	196/628			153		
硬件木马电路 计数器触发	*RS232-T300*	65	1331	71	11	10/23	1/528	√	8	8	
	RS232-T500	98	1893	56	11	10/23	1/519	√	8	8	
	AES-T900	7382	773270	128	1	0	1/8131	√	8	10	
组合逻辑触发	*RS232-T400*	65	276	22	10	10/23	0		8	92	√
	RS232-T800	65	510	22	10	10/23	0		8	100	√
	AES-T400	7327	765016	0	0	0	0		8	8513	√
	AES-T700	7387	765013	0	0	0	0		8	898	√
深度状态机触发	*RS232-T600*	69	491	26	12	10/23	2/141	√	8	9	
	RS232-T900	69	485	26	12	10/23	2/120	√	8	8	
	PIC16F84-T100	589	42410	348	202	201/628	1/731	√	155	179	
	PIC16F84-T200	488	39517	298	197	196/628	1/764	√	153	178	
	PIC16F84-T300	539	39875	298	197	196/628	1/764	√	153	178	
	PIC16F84-T400	539	39499	298	197	196/628	1/764	√	155	178	
抗检测型木马	*DeTrust-RS232-T800*	76	385	23	10	10/23	0		8	78	√
	DeTest-RS232-T1000	65	841	39	11	10/23	0		8	118	√

(注: *node* 指节点数量; *flow* 指控制流数量; *s_node* 指自循环节点与环状节点数量之和; *s_group* 指自循环节点组数量; *Gen_count/Max* 为正常逻辑电路自循环节点组数量及最大节点组汇聚值; *HT_count/Max* 为硬件木马电路自循环节点组数量及最大节点组汇聚值; *Hit* 为是否检测到; *Gen_Max* 为正常电路最大节点汇聚值; *HT_Max* 为硬件木马电路最大节点汇聚值)

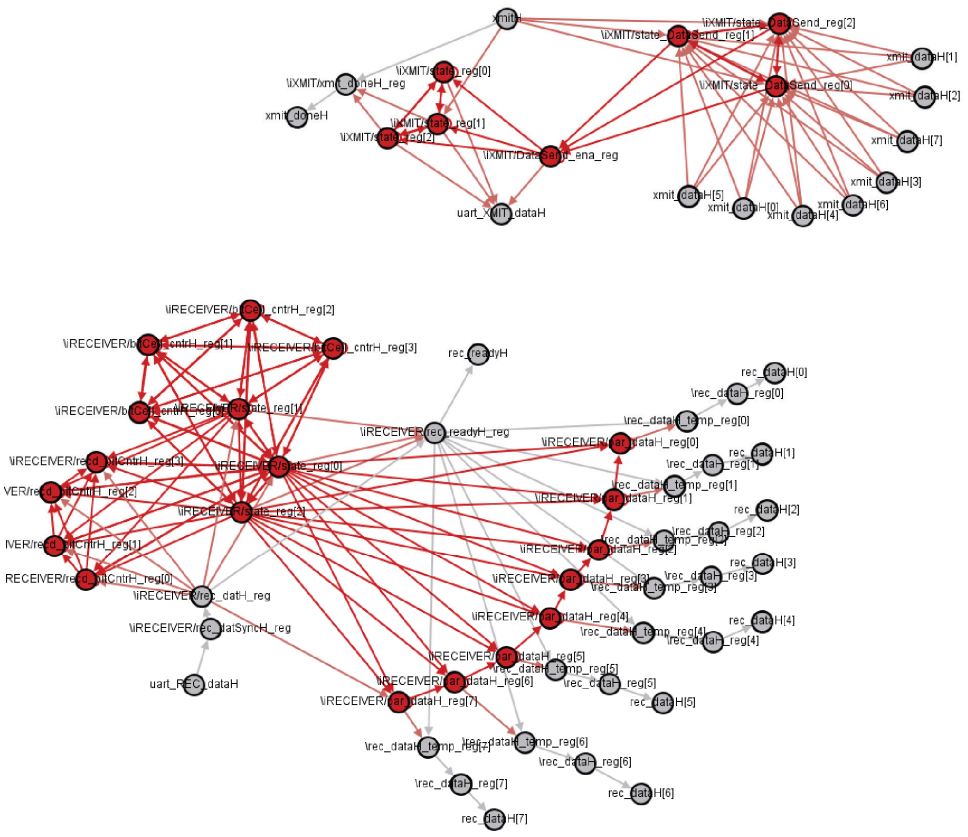


图 12　RS232-T900 节点控制流图

Figure 12　NCFG of RS232-T900

AES-T900 触发电路为 128bit 计数器, 随着计数器规模的上升, 自循环节点组汇聚值显著增加。需要指出的是在 PIC16F84 系列硬件木马检测样本中, 由于其正常电路为处理器模块, 逻辑比较复杂, 存在大量数据寄存器、指令寄存器对状态机的控制电路, 因此其正常电路状态机自循环节点组汇聚值也较高, 出现了十多个汇聚值为 600 左右的正常自循环节点组, 造成误检。但硬件木马自循环节点组汇聚值为 764, 依然大于正常电路。

4.2.3 组合逻辑分析

根据节点控制流图, 通过计算普通节点中是否存在汇聚值比较大的节点检测组合逻辑触发硬件木马。一个节点的汇聚值与驱动节点的数目及控制流的大小有关。可以看出正常电路的节点汇聚值 *Gen_Max* 一般不超过 10, 而用于输入信号模板匹配触发、内部状态模板匹配触发等类型的组合逻辑木马, 其均会有一个触发信号, 该信号节点汇聚值一般为 100 以上, 正常电路的节点汇聚值一般为 10 左右, 节点组汇聚值判断阈值取 75。与硬件木马有显著区别。

需要指出, 对于新型抗检测木马, 包括在组合逻辑触发硬件木马电路中插入 D 触发器的 DeTrust_RS232_T800 及增加木马信号翻转概率的 DeTest_RS232_T1000, 其将组合逻辑门与时序逻辑门进行了融合设计, 可以抵抗 VeriTrust、COTD 等硬件木马检测方法。但是本文提出的基于网表控制流的硬件木马检测方法, 考虑了信号在组合逻辑门和时序逻辑门之间的传递性, 其硬件木马节点汇聚值均可被正确量化, 在组合逻辑分析中我们可以看到其木马节点汇聚值分别为 78 和 118, 远高于正常电路节点汇聚值, 可以被有效检测。

4.3 与同类研究的比较

本文提出的硬件木马检测方法与近几年提出的部分硬件木马检测方法的检测结果对比如表 3 所示。

针对硬件木马检测平台 TrustHub 上的硬件木马, 各种检测方法均能进行有效检测。对于新型抗检测木马, 例如, 基于可控性、可观性分析的硬件木马检测 COTD 会出现漏检。FASTrust 对时序触发的硬件木马只考虑了自循环节点, 未考虑环状节点组, 会出现漏检。同时, 针对如 RS232-T900 正常电路部分状态机, 使用 FASTrust 时会出现误检。AONN 硬件木马检测针对组合逻辑触发的硬件木马提取了连接特征, 但插入 D 触发器后其特征容易被破坏, 造成漏检。

5 结论

集成电路芯片是电子信息系统的核心部件, 因此针对芯片设计的硬件木马检测非常重要。近年来各国学者提出了一系列硬件木马检测方法, 对常见的低概率触发的硬件木马进行检测。然而硬件木马设计与硬件木马检测互相促进发展, 通过本文对近年的硬件木马检测方法进行脆弱性分析, 发现新型抗检测可以通过组合时序逻辑混合等方法破坏其木马样本特征, 达到躲避检测的目的。

表 3 硬件木马检测方法对比

Table 3 Comparison of HT detections

	[23]	[28]	[29]	[41]	[43]	本文
TrustHub	√	√	√	√	√	√
Defeating UCI	√	√	√	√	√	√
DeTrust			√	√		√
DeTest				√	√	√
*新型抗检测木马						√

(注: 新型抗检测木马指 3.1 节中图 4、图 5 类型木马)

在此基础上, 本文提出了一种基于网表控制流分析的硬件木马检测方法。本方法根据电路网表中逻辑门属性及连接关系对网表电路进行了更精细的刻画, 提出了节点控制流图的建模方法, 并且利用生成的节点控制流图将电路触发概率进行量化, 达到准确检测低概率触发的硬件木马。

在下一步的研究工作中, 我们将研究如何降低节点控制流图检测的复杂度, 进一步增加硬件木马检测的效率和准确性。

参考文献

[1] Tehranipoor M H, Wang C. Introduction to Hardware Security and Trust[M]. New York, NY: Springer, 2012.

[2] Bhunia S, Hsiao M S, Banga M, et al. Hardware Trojan Attacks: Threat Analysis and Countermeasures[C]. *The IEEE*, 2014: 1229-1247.

[3] Skorobogatov S, Woods C. Breakthrough Silicon Scanning Discovers Backdoor in Military Chip[C]. *International Workshop on Cryptographic Hardware and Embedded Systems*, 2012: 23-40.

[4] Tehranipoor M, Koushanfar F. A survey of hardware trojan taxonomy and detection[J]. *IEEE Design & Test of Computers*, 2010, 27(1): 10-25.

[5] King S T, Tucek J, Cozzie A, et al. Designing and Implementing Malicious Hardware[C]. *The 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, 2008: 1-8.

[6] Sullivan D, Biggers J, Zhu G D, et al. FIGHT-Metric: Functional Identification of Gate-Level Hardware Trustworthiness[C]. *2014 51st ACM/EDAC/IEEE Design Automation Conference*, 2014: 1-4.

[7] Guo X L, Dutta R G, Jin Y E, et al. Pre-Silicon Security Verification and Validation: A Formal Perspective[C]. *The 52nd Annual Design Automation Conference*, 2015: 1-6.

[8] Dupuis S, Ba P S, Flottes M L, et al. New Testing Procedure for

Finding Insertion Sites of Stealthy Hardware Trojans[C]. *2015 Design*, *Automation & Test in Europe Conference & Exhibition*, 2015: 776-781.

[9] Saad W, Sanjab A, Wang Y P, et al. Hardware trojan detection game: A prospect-theoretic approach[J]. *IEEE Transactions on Vehicular Technology*, 2017, 66(9): 7697-7710.

[10] Li J, Lach J. At-Speed Delay Characterization for IC Authentication and Trojan Horse Detection[C]. *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, 2008: 8-14.

[11] Huang Y W, Bhunia S, Mishra P. MERS: Statistical Test Generation for Side-Channel Analysis Based Trojan Detection[C]. *The 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016: 130-141.

[12] Liu Y, Huang K, Makris Y. Hardware Trojan Detection through Golden Chip-Free Statistical Side-Channel Fingerprinting[C]. *2014 51st ACM/EDAC/IEEE Design Automation Conference*, 2014: 1-6.

[13] Agrawal D, Baktir S, Karakoyunlu D, et al. Trojan Detection Using IC Fingerprinting[C]. *2007 IEEE Symposium on Security and Privacy*, 2007: 296-310.

[14] Courbon F, Loubet-Moundi P, Fournier J J A, et al. SEMBA: A SEM Based Acquisition Technique for Fast Invasive Hardware Trojan Detection[C]. *2015 European Conference on Circuit Theory and Design*, 2015: 1-4.

[15] Bao C X, Forte D, Srivastava A. On reverse engineering-based hardware trojan detection[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2016, 35(1): 49-57.

[16] Fyrbiak M, Wallat S, Swierczynski P, et al. HAL—The missing piece of the puzzle for hardware reverse engineering, trojan detection and insertion[J]. *IEEE Transactions on Dependable and Secure Computing*, 2019, 16(3): 498-510.

[17] Potkonjak M. Synthesis of Trustable ICs Using Untrusted CAD Tools[C]. *The 47th Design Automation Conference*, 2010: 633-634.

[18] Jayakumar N, Purandare M, Somenzi F. Dos and Don'ts of CTL State Coverage Estimation[C]. *The 40th annual Design Automation Conference*, 2003: 292-295.

[19] Ugarte I, Sanchez P. Formal Meaning of Coverage Metrics in Simulation-Based Hardware Design Verification[C]. *Tenth IEEE International High-Level Design Validation and Test Workshop*, 2006: 221-228.

[20] Zhang X H, Tehranipoor M. Case Study: Detecting Hardware Trojans in Third-Party Digital IP Cores[C]. *2011 IEEE International Symposium on Hardware-Oriented Security and Trust*, 2011: 67-70.

[21] Qiu Y X, Li H W, Wang T C, et al. Property Coverage Analysis Based Trustworthiness Verification for Potential Threats from EDA Tools[C]. *2016 IEEE 25th Asian Test Symposium*, 2016: 43-48.

[22] Hicks M, Finnicum M, King S T, et al. Overcoming an Untrusted Computing Base: Detecting and Removing Malicious Hardware Automatically[C]. *2010 IEEE Symposium on Security and Privacy*, 2010: 159-172.

[23] Zhang J, Yuan F, Wei L X, et al. VeriTrust: Verification for Hardware Trust[C]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2015: 1148-1161.

[24] Xue M F, Hu A Q, Huang Y, et al. Monte Carlo Based Test Pattern Generation for Hardware Trojan Detection[C]. *2013 IEEE 11th International Conference on Dependable*, *Autonomic and Secure Computing*, 2014: 131-136.

[25] Zhou E R, Li S Q, Chen J H, et al. A Novel Detection Method for Hardware Trojan in Third Party IP Cores[C]. *2016 International Conference on Information System and Artificial Intelligence*, 2017: 528-532.

[26] Sturton C, Hicks M, Wagner D, et al. Defeating UCI: Building Stealthy and Malicious Hardware[C]. *2011 IEEE Symposium on Security and Privacy*, 2011: 64-77.

[27] Zhang J, Yuan F, Xu Q. DeTrust: Defeating Hardware Trust Verification with Stealthy Implicitly-Triggered Hardware Trojans[C]. *The 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014: 153-166.

[28] Waksman A, Suozzo M, Sethumadhavan S. FANCI: Identification of Stealthy Malicious Logic Using Boolean Functional Analysis[C]. *The 2013 ACM SIGSAC conference on Computer & communications security*, 2013: 697-708.

[29] Salmani H. COTD: Reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist[J]. *IEEE Transactions on Information Forensics and Security*, 2017, 12(2): 338-350.

[30] Xie X, Sun Y Y, Chen H D, et al. Hardware trojans classification based on controllability and observability in gate-level netlist[J]. *IEICE Electronics Express*, 2017, 14(18): 20170682.

[31] Sebt S M, Patooghy A, Beitollahi H, et al. Circuit enclaves susceptible to hardware trojans insertion at gate-level designs[J]. *IET Computers & Digital Techniques*, 2018, 12(6): 251-257.

[32] Dharmadhikari P, Raju A, Vemuri R. Detection of Sequential Trojans in Embedded System Designs without Scan Chains[C]. *2018 IEEE Computer Society Annual Symposium on VLSI*, 2018: 678-683.

[33] Farajipour Ghohroud N, Hessabi S. Behavioral-level hardware trust: Analysis and enhancement[J]. *Microprocessors and Microsystems*, 2018, 58: 24-33.

[34] Kok C H, Ooi C Y, Inoue M, et al. Net Classification Based on Testability and Netlist Structural Features for Hardware Trojan Detection[C]. *2019 IEEE 28th Asian Test Symposium*, 2020: 105-1055.

[35] Kok C H, Ooi C Y, Moghbel M, et al. Classification of Trojan Nets Based on SCOAP Values Using Supervised Learning[C]. *2019 IEEE International Symposium on Circuits and Systems*, 2019: 1-5.

[36] Huang K, He Y. Trigger identification using difference-amplified controllability and dynamic transition probability for hardware trojan detection[J]. *IEEE Transactions on Information Forensics and Security*, 2020, 15: 3387-3400.

[37] Oya M, Shi Y H, Yanagisawa M, et al. A Score-Based Classification Method for Identifying Hardware-Trojans at Gate-Level Netlists[C]. *Design*, *Automation & Test in Europe Conference & Exhibition*, *2015*, 2015: 465-470.

[38] Hasegawa K, Oya M, Yanagisawa M, et al. Hardware Trojans Classification for Gate-Level Netlists Based on Machine Learning[C]. *2016 IEEE 22nd International Symposium on On-Line*

Testing and Robust System Design, 2016: 203-206.

[39] Hasegawa K, Yanagisawa M, Togawa N. Trojan-Feature Extraction at Gate-Level Netlists and Its Application to Hardware-Trojan Detection Using Random Forest Classifier[C]. *2017 IEEE International Symposium on Circuits and Systems*, 2017: 1-4.

[40] Hasegawa K, Yanagisawa M, Togawa N. A Hardware-Trojan Classification Method Utilizing Boundary Net Structures[C]. *2018 IEEE International Conference on Consumer Electronics*, 2018: 1-4.

[41] Yao S, Chen X M, Zhang J, et al. FASTrust: Feature Analysis for Third-Party IP Trust Verification[C]. *2015 IEEE International Test Conference*, 2015: 1-10.

[42] Chen X M, Liu Q Y, Yao S, et al. Hardware trojan detection in third-party digital intellectual property cores by multilevel feature analysis[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018, 37(7): 1370-1383.

[43] Chen F Q, Liu Q. Single-Triggered Hardware Trojan Identification Based on Gate-Level Circuit Structural Characteristics[C]. *2017 IEEE International Symposium on Circuits and Systems*, 2017: 1-4.

[44] Liu Q, Zhao P Y, Chen F Q. A hardware trojan detection method based on structural features of trojan and host circuits[J]. *IEEE Access*, 2019, 7: 44632-44644.

[45] Zhao P Y, Liu Q. Density-Based Clustering Method for Hardware Trojan Detection Based on Gate-Level Structural Features[C]. *2019 Asian Hardware Oriented Security and Trust Symposium*, 2020: 1-4.

[46] Salmani H, Tehranipoor M, Karri R. On Design Vulnerability Analysis and Trust Benchmarks Development[C]. *2013 IEEE 31st International Conference on Computer Design*, 2013: 471-474.

[47] Shakya B, He T, Salmani H, et al. Benchmarking of hardware trojans and maliciously affected circuits[J]. *Journal of Hardware and Systems Security*, 2017, 1(1): 85-102.

张宁 于 2013 年在哥伦比亚大学电气工程专业获得硕士学位。现任中国科学院信息工程研究所工程师。研究领域为电磁空间安全。研究兴趣包括芯片安全技术、物理安全检测。Email: zhangning@iie.ac.cn

吕志强 于 2007 年在哈尔滨工业大学微电子学与固体电子学专业获得博士学位。现任中国科学院信息工程研究所副研究员。研究领域为信号收发与分析、射频系统集成。研究兴趣包括嵌入性系统安全研究。Email: lvzhiqiang@iie.ac.cn

张焱琳 于 2014 年在沈阳工业大学电子科学与技术专业获得学士学位。现在中国科学院大学网络空间安全学院计算机技术专业攻读专业硕士学位。研究领域为硬件木马检测。研究兴趣包括集成电路设计。Email: zhangyanlin@iie.ac.cn

黄伟庆 于 2002 年在北京邮电大学获得硕士学位。现任中国科学院信息工程研究所正研级高级工程师。研究领域为电磁信息安全、信号处理等。研究兴趣包括电磁泄漏发射检测、无线通信安全、电磁信息安全。Email: huangweiqing@iie.ac.cn