https://doi.org/10.1051/jnwpu/20254340813

基于属性自动提取与形式化验证的硬件木马检测方法

戴晓洁, 程梦妮, 朱嘉诚, 邰瑜

(西北工业大学 网络空间安全学院, 陕西 西安 710072)

摘 要:在大规模集成电路设计领域,硅前硬件木马检测面临两大挑战:缺乏黄金模型和检测准确率不足。研究旨在提出一种新的方法以提高集成电路设计中硬件木马的检测速度和准确性。该方法采用形式化分析,将集成电路功能轨迹中每个时钟周期内信号的跳变作为分析焦点,通过遍历匹配技术构建了候选信号集,并利用低翻转信号的特性进行属性挖掘,以实现硬件设计功能属性的自动提取。此外,通过分析和形式化验证来执行硬件木马检测,并利用验证结果中的反例来快速定位相关设计中的硬件木马。对 Trust-Hub 数据集的 AES 系列部分测试基准实验结果表明:所提方法成功自动提取了与低活跃度信号相关的定常属性,并有效地检测出了硬件木马。所提方法为大规模集成电路设计中的硬件木马检测提供了一种快速准确的解决方案。

关 键 词:硬件设计;属性提取;形式化验证;硬件木马检测

中图分类号:TP309 **文献标志码**:A **文章编号**:1000-2758(2025)04-0813-08

随着半导体供应链的全球化,为节约开发成本和时间,现代计算系统通常集成了来自不同安全等级供应商的电路设计,特别是来自不可信第三方的知识产权(intellectual property,IP)核,这些互联互通的计算资源的硬件组件构成了一个混合信任的硬件电路计算环境。硬件木马是硬件电路在设计和生产过程中可能发生的恶意设计修改,会导致电路功能异常、制造后门允许远程操控和破坏等问题[1]。例如,2012 年在军用级 FPGA 的 ProASIC3 芯片中发现了未记录的硬件木马[2],该木马通过提取密钥,使攻击者能够修改芯片的配置,从而完全控制芯片。与芯片设计制造流程相对应,硬件木马检测技术可以分为硅前检测和硅后检测。随着集成电路集成度的不断提高以及设计规模的不断增大,硬件设计硅后检测也越来越艰难,一方面设计验证需要的代价太大;另一方面测试和验证方法的局限性导致其覆盖率难以保证。因此系统中往往存在未被发现的漏洞,只有在造成重大损失后才会被发现,带来了巨大的安全风险。因此在硅前设计阶段对硬件设计进行安全性检测是十分必要的[3]。2016 年美国国家自然科学基金委(national science foundation,NSF)形式化安全方法研讨会报告指出:形式化验证是计算机系统达到安全和保密需求的唯一可靠途径,其中可验证的底层硬件安全机制是软件安全的基础[4]。因此,如何应用形式化的方法快速准确地定位到硬件木马是一个值得深入研究的课题。

近年发展起来的基于属性检查的形式化验证是一种静态的硬件设计验证方式,旨在通过工具从逻辑上检查设计文件是否满足某种属性,相对于逻辑仿真验证硬件设计,形式化验证可以极大缩短验证时间[5]。基于信息流追踪[6-7]的形式化验证方法可以检测出违反机密性、安全性以及隔离特性等安全属性的恶意逻辑,为硬件设计脆弱性检测提供了一种新的思路。Guo 等[8]研究了硬件信任评估中常用的两大类形式化方法:定理证明和等价验证,并从形式化视角概述了符号代数在等价验证中的使用。进一步,该团队提出了一种新的分层形式化验证框架

收稿日期:2024-08-26

基金项目:国家自然科学基金面上项目(U23B2041,62272389,62074131)资助

作者简介:戴晓洁(1988—),助理研究员

通信作者:邰瑜(1982—),副研究员 e-mail:taiyu@nwpu.edu.cn

用于硅前阶段的电路信任评估[9],通过形式化验证 8051 微处理器上的内存完整性属性证明所提出框架的适用性。Farzana 等[10]描述了一种属性驱动的形式化方法用于设计安全的 SoC,并证明了该方法在不同威胁模型下的有效性,将形式化方法融入 SoC 设计流程,从设计端增强安全性。

很多学者重点研究了集成电路设计的属性自动提取,并利用属性进行形式化验证。Vasudevan 等[11]通过 Goldmine 工具静态分析与决策树自动提取断言语句,利用一种形式化引擎验证断言语句的有效性。Danese 等[12]提出了一种数据挖掘与覆盖率分析相结合的方法,提取形式如 $G(\varphi \rightarrow \psi)$ 的时序断言。Ghasempouri 等[13]提出了基于度量标准评估规范的列联表相关性及“兴趣性”挖掘技术,能够对获得的断言进行排序。Malburg 等[14]阐述了一种通过图的构建来遍历动态依赖图从而推断时序属性的方法。这些研究在安全属性方面有所欠缺。Garcia 等[15]通过对硬件规范的挖掘来自动生成硬件设计属性,该属性可以与基于断言的验证技术一起使用,但仅限于在单个执行跟踪上评估的属性[16],无法显示信息流从特定源流向特定接收器。Smith 等[17]利用函数语言开展安全属性自动提取,然而该方法无法在时序上对安全属性进行挖掘,若对该方法进行时间属性的扩展会导致性能低下。Bauer 等[18]建立了基于 LTL(linear temporal logic)三值语义的属性提取框架,时刻监视硬件设计行为来提取一个不确定的结果,无法对确定的属性进行提取。工具方面,IODINE 结合静态分析与动态跟踪方法来自动查找例如信号互斥或 One-hot 编码行为的作为属性实例[19]。Zhang 等[20]采用机器学习方法自动提取硬件设计的关键安全属性,包含通过对已知漏洞进行训练获取未知漏洞和通过已知安全属性的训练提取更多的属性。Isadora 工具能够完成由跟踪属性到信息流安全属性的提取。与在多个跟踪轨迹定义的行为规范不同,规范挖掘是在执行单个跟踪定义的行为模式中进行发掘,所以只适用于对跟踪属性的提取。Deutschbein 等[21]在工具设计中加入了信息流跟踪逻辑,把安全规范应用到工具结构中,能够提取信息流信息。然而这些工具仍处于研究阶段,距离应用还存在一定差距,自动提取出的属性大多无法兼备硬件设计的功能与安全性,仍需要人工书写验证属性,当系统规模大或结构复杂时,属性的定义和验证会变得更为复杂,无法保证属性的完备性。

硬件木马检测技术代表性的方法有逻辑测试、形式验证、代码分析和机器学习方法[22]。由于芯片结构日益复杂,获取“黄金模型(经验证没有硬件木马植入的电路)”已经非常困难,需要黄金模型的方式具有很大的局限性[23]。在大规模系统上应用神经网络进行分析可能会面临计算资源和时间的挑战。通过手动的代码分析、逻辑验证等硬件木马检测方式非常耗时。因此,如何自动提取硬件电路设计属性中与木马相关的属性,并根据形式化验证结果来判别是否存在木马,对大规模电路设计的硬件木马检测具有实用意义。

1 硬件木马检测方法

本文提出了一种基于属性自动提取与形式化验证的硬件木马检测方法。首先利用仿真工具获得硬件设计电路中每个信号的仿真轨迹数据;接着对轨迹数据采用遍历查找算法找出符合特性的属性信号,实现集成电路定常属性的自动提取;然后进一步缩小需要提取属性的设计规模,方便电路设计及检测人员进行人工分析,通过人工分析对查找到的属性信号进行判别,将有效属性作为形式化验证的电路验证语句;同时采用形式化验证工具对检测出的有效属性进行验证,根据验证结果定位属性关联的设计语句,可以检测硬件木马,并获取木马触发条件。硬件木马检测过程如图 1 所示。

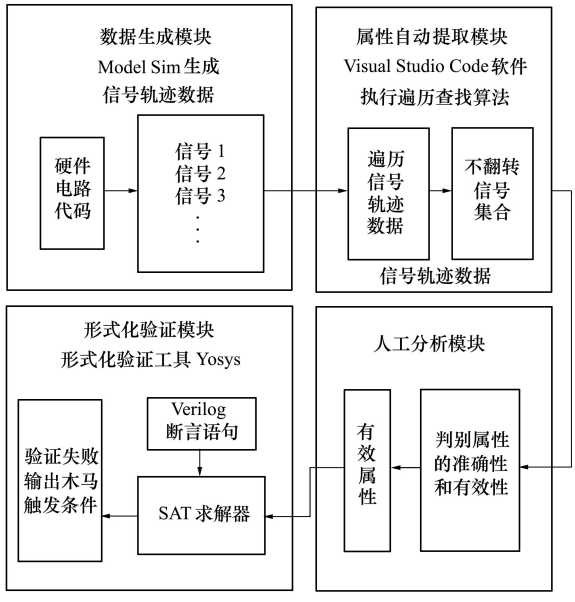

图 1 硬件木马检测过程

2　属性自动提取方法

2.1　数字电路低翻转信号特性

数字电路可以分为组合逻辑和时序逻辑 2 种,组合逻辑由逻辑门组成,其输出仅取决于当前的输入。时序逻辑包括时钟和存储元件,具有时序性,任意时刻的输出信号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。低触发信号是指在数字电路仿真信号观测值中存在明显断崖式概率分布的信号集合,以单比特信号为例,它的取值为低(高)的概率无限接近于零,而取值为高(低)的概率接近于 1,即低触发信号的取值状态极少或者永不产生跳变,当信号发生跳变时,它的熵值会增加,即信号翻转时会以某种方式传递更多的信息。例如在硬件设计源代码中嵌入一个恶意逻辑,当明文输入为特定值时,该逻辑可以泄露密钥到输出端口,该特定值出现概率极低,在有限的测试周期中很难测到其跳变情况。

2.2　定常属性

集成电路属性是集成电路设计行为的形式化或有关设计规范的约束。定常属性是对始终成立的硬件设计不变行为进行建模,例如,安全策略通常希望加密算法核完成加密的时间或者功率恒定,以防止信息从时间或能量侧信道泄露。硬件电路设计在不同输入条件下表现出的行为应该不变,检测人员可以将集成电路设计的不变特性形式化表述为定常属性。常量值和常量时间都属于定常属性。

2.3　定常属性的提取算法

利用定常属性的不变特性,可以将其看做极低概率发生变化的事件,对电路设计中仿真获得的所有信号进行遍历,对其中不发生翻转的信号进行标记,由于仿真受限于时间和规模,短时间不发生变化的信号长时间可能发生变化,但长时间不发生变化的信号在短时间内一定不会发生变化,据此可以得出,从仿真数据里得到的不翻转信号一定包含了电路设计的定常属性。

基于上面判断,可以通过遍历查找电路轨迹里的不翻转信号来挖掘定常属性,算法具体描述为:首先判断字符串的起始位是否为地址符‘/’,末尾的信号值不能为不定态‘x’或者高阻态‘z’,如果满足上述条件则将该信号加入到数据链表中,否则继续判断下一条字符串,直到整个仿真信号集都被加入到数据链表之中。接下来判断数据链表之中的每个信号是否有过翻转行为。

创建一个指向链表头部的指针 cur,令其指向第一个信号节点,创建另一个同样指向链表头部的指针 pre,在 cur 指针指向每一个信号的过程中,pre 指针不断往后搜索。当出现 cur 指针与 pre 指针所指向的节点处信号名称相同的情况时,搜索中止并判断 2 个节点的信号值是否相等,如果不相等则将该节点的翻转标志位 sig 赋值为 1,表示该信号在仿真过程中至少经过了 1 次翻转。对于每一个链表中的信号都重复上述过程,通过遍历链表来查看每一个信号值在仿真过程中是否发生翻转。最后在标记阶段完成后,搜索整个链表提取出节点标志位 sig 为 0 的信号及其取值作为低翻转信号集。提取算法如下所示。

算法 1　低翻转信号提取算法

```
1: 输入:AES-T1000 仿真数据集
2: 输出:低翻转信号集合
3:  define biaoji:// 翻转标记函数
4:  while cur is not None do// 当前节点不为空
5:    while pre is not None do// 下一节点不为空
6:    if cur ! =pre do// 如果信号数值不等
7:    cur.sig←1// 设置翻转位为 1
8:      pre←pre.next// 分析下一节点
9:      cur←cur.next// 分析下一节点
10:  define tiqu(self,difan):// 低翻转信号提取函数
11:    while cur is not None do// 当前节点不为空
12:    if cur.sig==0 do// 如果该节点标志位为 0
13:      difan← cur// 将该信号加入到低翻转信号数组中
14:      cur ← cur.next// 分析下一节点
```

3　形式化验证方法

形式化验证技术是一种基于数理逻辑论证方法检验系统是否满足给定安全规范的方法[24]。该方法采用规范化的语法语义去定义系统以及系统应该满足的属性,再进行人工或自动化的推理论证,最终可以给出系统关于给定属性的安全性或正确性结论。与软件测试这种保障软件安全的方法不同,形式化验证是用推理论证的方法“证明”软件不存在某种安全漏洞,而不是采用用例覆盖的方法去“检测”漏洞的存在性,因此形式化方法可以从根本上

保证系统的安全性。在建立形式化规范与系统模型后,需要采用形式化验证器进行辅助验证,如交互式定理证明器、行为监测引擎等。

Yosys 是基于 Verilog RTL 的开源逻辑综合和验证工具,可以处理任何可综合的 Verilog 设计[24-26],可以将 Verilog 设计通过逻辑综合转换为门级网表,并集成了 SAT 验证工具,用于实现功能和安全属性的形式化验证。

图 2 为典型属性形式化验证流程。首先使用属性描述语言以断言的形式描述需验证的属性,可以将断言语句直接嵌入原始逻辑中,然后将属性中的断言语句转换为证明约束,使用形式化求解器在证明约束下验证集成电路设计是否满足断言。若不满足断言则验证失败,提供一个反例,可用于违反策略的逻辑复现;若满足断言,则验证成功。

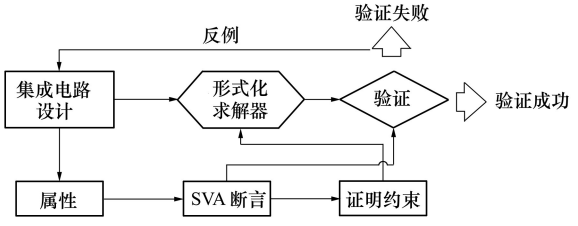

图 2 形式化验证流程图

4 实验结果及分析

4.1 仿真测试环境

实验环境配置如表 1 所示。实验中各个模块程序主要采用 Python 编写,实验采用开源电路综合优化工具软件 Yosys,安装在虚拟机的 Ubuntu 18.04.1 操作系统中,内置 SAT 求解器用于测试基准形式化安全验证。测试数据由 ModelSim SE-64 10.4 生成。

表 1 实验环境配置

实验环境	Windows 系统	Linux 系统(虚拟机内安装)
硬件配置	CPU:Intel(R) Core(TM) i7-8700 3.20 GHz GPU:Nvidia GeForce RTX 2070 (8 GB) 内存:16G 硬盘:650G	
软件配置	Windows 10(64 位) ModelSim SE-64 10.4 Python 3.6 Visual Studio Code 1.84.2 Vmware Workstation Pro	Ubuntu 18.04.1 Yosys Open Synthesis Suite 0.7

4.2 定常属性提取和分析

通过 ModelSim 对 AES-T1000 基准进行轨迹仿真,获得包含时钟信号、复位信号、明密文和密钥信号在内的信号数据集作为输入信号数据,部分仿真波形如图 3 所示。

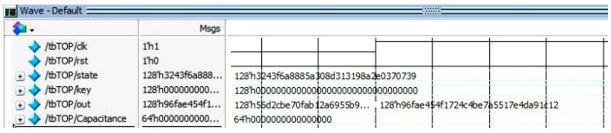

图 3 AES-T1000 仿真波形(部分)

在 Visual Studio Code 软件中执行算法 1 所示的算法过程,从 6 万条 AES-T1000 仿真轨迹数据中提取到的 6 条低翻转信号集如表 2 所示。

表 2 低翻转信号集

序号	信号名称	信号数值
1	/tbTOP/uut/Trojan/load	64'h0000000000000000
2	/tbTOP/Capacitance	64'h0000000000000000
3	/tbTOP/uut/Capacitance	64'h0000000000000000
4	/tbTOP/uut/Trigger/Tj_Trig	1'h0
5	/tbTOP/uut/Tj_Trig	1'h0
6	/tbTOP/uut/Trojan/Tj_Trig	1'h0

对实验提取出的 6 个低翻转信号进行分析:信号 Capacitance 是信号 load 在 TOP 级模块实例化中的别名,所以两者等价即信号值相等,本实验中一直为 0 表示木马在仿真周期内未被触发。表 2 中后 3 个信号是木马触发信号 Tj_Trig 在不同级别模块中的展示,可视为同一信号,表示木马未被触发。从而得到该硬件电路的定常属性为:断言木马触发信号 Tj_Trig 始终保持为 0,断言木马的负载输出信号 Capacitance 与信号 load 始终保持为 0,可通过属性描述语言表达如下:

1: set rst:=0

2: assert Tj_Trig==0

4.3 收敛速度仿真

从算法过程分析,低翻转信号个数应该是随仿真时间收敛的,为了观察具体规律,本文分别选择 AES-T1000 基准和 RSA-T300 基准设计内的 2 个模块进行仿真。

AES-T1000 设计中 TOP 模块是顶层模块,TSC

模块是该设计中木马的负载模块,仿真结果如图 4 所示,可以看到 2 个模块的低翻转信号个数分别从 451 和 279 逐渐收敛到个位数,在仿真初期收敛速度较快,随着仿真时间的增加收敛速度明显放缓。TOP 模块的低翻转信号提取结果最终收敛为木马触发信号 Tj_Trig,TSC 模块最终收敛为木马的负载信号 load 和触发信号 Tj_Trig。

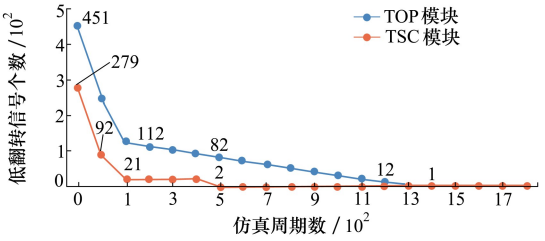

图 4　AES-T1000 低翻转信号个数随仿真时间变化情况(单个周期为 10 ns)

在 RSA-T300 设计中,选择 RSACypher 和 Modmult 模块进行分析,仿真结果如图 5 所示,可以看到 RSACypher 模块在仿真时间超过 600 个时钟周期后,低翻转信号个数从 147 收敛为 38。在检查低翻转信号集合与原始设计后,38 个信号中 7 个为使能信号 TjEnable,30 个为木马触发计数信号 TrojanCounter,1 个 eqOp 信号为触发信号。Modmult 模块在经过 200 个时钟周期的仿真后,低翻转信号个数快速下降到 5 并保持不变。检查这 5 个信号的原始设计,发现这 5 个信号均为第 33 位信号,该设计为 32 位密码核,第 33 位信号始终未使用,所以未发生反转。

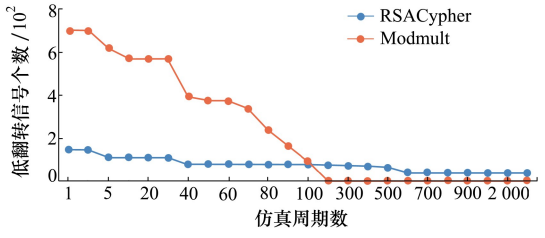

图 5　RSA-T300 低翻转信号个数随仿真时间变化情况(单个周期为 10 ns)

从图 4~5 仿真结果可以看出硬件设计内的低翻转信号个数整体上具有随时间逐渐收敛的趋势,一般在仿真初期收敛速度较快,随着仿真时间的增加收敛速度明显放缓。

4.4　形式化验证结果及分析

本文中主要通过 Verilog 断言语句以及形式化验证工具 Yosys 对集成电路属性进行形式化验证。验证前使用 Yosys 工具将包含 Tj_Trig 信号的 Trojan_Trigger 模块 RTL 级代码转换为门级,生成门级网表文件 GATELEVEL.v。具体脚本指令如下:

1: read_verilog Trojan_Trigger.v

2: hierarchy-check

3: proc; opt; fsm; memory; opt

4: write_verilog GATELEVEL.v

仿真需要证明在不复位的前提下,低翻转信号 Tj_Trig 是否始终保持为 0。具体安全约束脚本指令如下,第一条指令读入设计文件,第二条指令调用 SAT 工具并设置约束:

1: read_verilog GATELEVEL.v

2: sat-prove Tj_Trig 1'b0-set \rst 0

验证结束,SAT 求解器形式化验证失败并给出一个反例(如图 6~7 所示),AES-T1000 返回的信号表明,当满足 state 为特定值时,低翻转信号 Tj_Trig 的逻辑值将会翻转为逻辑 1。RSA-T300 返回的信号表明,只有第二位为 1,其他位均为 0 时,木马触发,与 RSA-T300 测试向量中木马设计触发条件完全一致。

图 6　AES-T1000 属性形式化验证结果

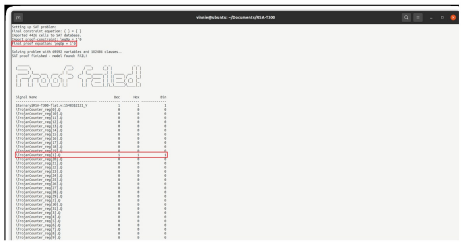

图 7　RSA-T300 属性形式化验证结果

通过观察图 6~7 中的反例可以看出:当输入明文为特定值时,Tj_Trig 信号会跳变为逻辑 1,在原始代码中进行追溯可以发现该信号为木马触发信号,当信号逻辑为 1 时会触发负载以泄露密钥;当计数

器触发条件时,即 TrojanCounter 第二位为 1 时,木马被触发。实验结果表明,本方法可以精确捕捉、定位与木马相关的触发信号或相关单元。

现有 FANCI、VeriTrust 和 GLIFT 3 种当前广泛使用的安全验证方法需要设计者重新建模,设计验证框架,个别验证规模有限。表 3 总结了 4 种方法针对 Trust-Hub 数据集部分木马测试基准的检测结果,本文方法从信息流的角度研究集成电路安全验证与木马检测方法,在门级网表的基础上描述信息流和标签的传播,无需设计者引入新的模型或框架,同样能够达到木马检测效果,相比 GLIFT 在适用范围方面具有一定优势,可以实现对多种功能不同的硬件设计的木马检测。

表 3 硬件木马检测结果对比

测试基准	触发条件	FANCI	VeriTrust	GLIFT	本文方法
AES-T400	预定义明文输入	✓	✓	✓	✓
AES-T500	预定义明文输入	✓	✓	-	✓
AES-T700	预定义明文输入	✓	✓	-	✓
AES-T800	预定义明文输入	✓	✓	-	✓
AES-T900	加密特定数量明文	✓	✓	-	✓
AES-T1000	预定义明文输入	✓	✓	✓	✓
AES-T1100	预定义明文输入	✓	✓	✓	✓
AES-T1200	加密特定数量明文	✓	✓	✓	✓
AES-T1300	预定义明文输入	✓	✓	-	✓
AES-T1400	预定义明文输入	✓	✓	-	✓

(注:其中“✓”表示可以检测,“-”表示没有报告对应方法的结果)

5 结 论

本文提出了一种基于属性自动提取与形式化验证的硬件木马检测方法,以低翻转信号特性为基础开展定常功能属性的自动提取,提取仿真轨迹中始终不变的信号,将其作为定常属性,采用 Yosys 验证工具读取集成电路 RTL 级代码设计,通过将自动提取出的定常属性转化为相应约束的方式,利用布尔可满足性分析的方法,对其进行形式化验证,如果验证失败,可以返回 1 个反例,对反例进行分析可以发现是硬件木马触发条件。本文方法能够帮助测试人员缩小需要提取属性的规模,快速识别、定位集成电路设计中的木马。

参考文献:

[1] 冯燕, 陈岚. 基于路径特征和支持向量机算法的硬件木马检测技术[J]. 电子与信息学报, 2023, 45(6): 1921-1932
FENG Yan, CHEN Lan. Hardware trojan detection based on path feature and support vector machine[J]. Journal of Electronics & Information Technology, 2023, 45(6): 1921-1932 (in Chinese)

[2] TAMZIDUL H, JONATHAN C, PRABUDDHA C, et al. Hardware IP trust validation: learn (the untrustworthy), and verify [C]//IEEE International Test Conference, New York, 2018: 1-10

[3] 陈星任, 熊焰, 黄文超, 等. 一种基于静态分析的多视图硬件木马检测方法[J]. 信息网络安全, 2023, 23(10): 48-57
CHEN Xingren, XIONG Yan, HUANG Wenchao, et al. A multi-view hardware Trojan detection method based on static analysis [J]. Netinfo Security, 2023, 23(10): 48-57 (in Chinese)

[4] CHONG S, GUTTMAN J, DATTA A, et al. Report on the NSF workshop on formal methodsforsecurity[EB/OL]. (2016-08-03) [2024-07-25]. https://arxiv.org/abs/1608.00678

[5] MOEBIUS N, STENZEL K, REIF W. Formal verification of application-specific security properties in a model-driven approach [C]//International Symposium on Engineering Secure Software and Systems, Berlin, 2010:166-181

[6] HU W, ARDESHIRICHAM A, KASTNER R. Hardware information flow tracking[J]. ACM Computing Surveys, 2021, 54(4): 1-39

[7] ARDESHIRICHAM A, HU W, MARXEN J, et al. Register transfer level information flow tracking for provably secure hardware design[C]//Design, Automation & Test in Europe Conference & Exhibition, 2017: 1691-1696

[8] GUO X, DUTTA R G, JIN Y, et al. Pre-silicon security verification and validation: a formal perspective[C]//2015 52nd ACM/EDAC/IEEE Design Automation Conference, San Francisco, 2015: 1-6

[9] GUO X, DUTTA R G, JIN Y. Hierarchy-preserving formal verification methods for pre-silicon security assurance[C]//2015 16th International Workshop on Microprocessor and SoC Test and Verification, Austin, 2015: 48-53

[10] FARZANA N, RAHMAN F, TEHRANIPOOR M, et al. SoC security verification using property checking[C]//2019 IEEE International Test Conference, Washington, 2019: 1-10

[11] VASUDEVAN S, SHERIDAN D, PATEL S, et al. Goldmine: automatic assertion generation using data mining and static analysis[C]//2010 Design, Automation & Test in Europe Conference & Exhibition, 2010: 626-629

[12] DANESE A, RIVA N D, PRAVADELLI G. A-team: automatic template-based assertion miner[C]//Proceedings of the 54th Annual Design Automation Conference, 2017: 1-6

[13] GHASEMPOURI T, PRAVADELLI G. On the estimation of assertion interestingness[C]//2015 IFIP/IEEE International Conference on Very Large Scale Integration, 2015: 325-330

[14] MALBURG J, FLENKER T, FEY G. Property mining using dynamic dependency graphs[C]//2017 22nd Asia and South Pacific Design Automation Conference, 2017: 244-250

[15] GARCIA F, OSWALD D, KASPER T, et al. Lock it and still lose it: on the(in)security of automotive remote keyless entry systems[C]//Proceedings of the 25th USENIX Security Symposium, 2016: 929-944

[16] KOZYRI E, CHONG S, MYERS A C. Expressing information flow properties[J]. Foundations and Trends in Privacy and Security, 2022, 3(1): 1-102

[17] SMITH C, FERNS G, ALBARGHOUTHI A. Discovering relational specifications[C]//Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, 2017: 616-626

[18] BAUER A, LEUCKER M, SCHALLHART C. Monitoring of real-time properties[C]//Proceedings of the 26th International Conference on Foundations of Software Technology and Theoretical Computer Science, 2006

[19] HANGAL S, NARAYANAN S, CHANDRA N, et al. IODINE: a tool to automatically infer dynamic invariants for hardware designs[C]//2005 42nd ACM/IEEE Design Automation Conference, Anaheim, 2005: 775-778

[20] ZHANG R, STANLEY N, GRIGGS C, et al. Identifying security critical properties for the dynamic verification of a processor[J]. ACM SIGARCH Computer Architecture News, 2017, 45(1): 541-554

[21] DEUTSCHBEIN C, MEZA A, RESTUCCIA F, et al. Toward hardware security property generation at scale[J]. IEEE Security & Privacy, 2022, 20(3): 43-51

[22] 黄钊, 王泉, 杨鹏飞. 硬件木马:关键问题研究进展及新动向[J]. 计算机学报, 2019, 42(5): 993-1017
HUANG Zhao, WANG Quan, YANG Pengfei. Hardware Trojan: research progress and new trends on key problems[J]. Chinese Journal of Computers, 2019, 42(5): 993-1017 (in Chinese)

[23] 史江义, 温聪, 刘鸿瑾, 等. 基于图神经网络的门级硬件木马检测方法[J]. 电子与信息学报, 2023, 45(9): 3253-3262
SHI Jiangyi, WEN Cong, LIU Hongjin, et al. Hardware Trojan detection for gate-level netlists based on graph neural network[J]. Journal of Electronics & Information Technology, 2023, 45(9): 3253-3262 (in Chinese)

[24] SRI D B, RAJAKUMAR K, MADHUSOODHANAN P, et al. A holistic approach to CPU verification using formal techniques[C]//2022 IEEE Women in Technology Conference, 2022: 1-5

[25] SHAH D, HUNG E, WOLF C, et al. Yosys+nextpnr: an open source framework from verilog to bitstream for commercial fpgas[C]//2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines, 2019: 1-4

[26] 陈春雷, 王省欣, 谭静, 等. 基于 Yosys 的硬件信息流安全验证与漏洞检测[J]. 计算机应用研究, 2021, 38(6): 1865-1869
CHEN Chunlei, WANG Xingxin, TAN Jing, et al. Hardware information flow security verification and vulnerability detection using Yosys[J]. Application Research of Computers, 2021, 38(6): 1865-1869 (in Chinese)

Hardware Trojans detection through automatic properties extraction and formal verification

DAI Xiaojie, CHENG Mengni, ZHU Jiacheng, TAI Yu
(School of Cybersecurity, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract: To address the difficulties in obtaining golden reference models and the low accuracy when conducting pre-silicon hardware Trojans detection on a large-scale integrated circuit design, a formal detection method to quickly and accurately locate hardware Trojans is proposed in this paper. The hopping signals within each clock cycle are used as the object of analysis in integrated circuit functional traces, in which the candidate set of these signals is obtained by a traversal matching approach. The properties of the candidate set are excavated by utilizing the low flip signal feature to realize the automatic extraction of the hardware design functional properties. According to the counterexamples given by the verification results, the hardware Trojans related design can be quickly located through analysis and formal verification. The experimental results of some Trust-Hub AES benchmarks show that the proposed method can automatically extract the constant properties related to low-activity signals and successfully detect hardware Trojans. This study provides a fast and accurate solution of hardware Trojans detection on a large-scale integrated circuit design.

Keywords: hardware design; property extraction; formal verification; hardware Trojans detection

引用格式: *戴晓洁, 程梦妮, 朱嘉诚, 等. 基于属性自动提取与形式化验证的硬件木马检测方法[J]. 西北工业大学学报, 2025, 43(4): 813-820*

DAI Xiaojie, CHENG Mengni, ZHU Jiacheng, et al. Hardware Trojans detection through automatic properties extraction and formal verification[J]. Journal of Northwestern Polytechnical University, 2025, 43(4): 813-820 (in Chinese)