

自适应优化的二元分类型硬件木马检测方法

薛明富^{1),2)} 王 箭^{1),2)} 胡爱群³⁾

¹⁾(南京航空航天大学计算机科学与技术学院 南京 210016)

²⁾(软件新技术与产业化协同创新中心 南京 210023)

³⁾(东南大学信息科学与工程学院 南京 210096)

摘 要 近年来,由于半导体设计和制造过程的全球化以及第三方知识产权核(Intellectual Property cores, IP cores)的广泛使用,电路越来越容易受到硬件木马的攻击。硬件木马能够使得电路功能故障、泄露机密信息或者导致其他一些灾难性的后果,因此已经引起了许多关键部门的密切关注。大多数已有的硬件木马检测工作需要参考芯片去提供参考信息,然而,获得参考芯片是极其困难的。参考芯片通常是假设通过可信的工厂生产出来或者通过很严苛的逆向工程检测验证为不含木马,两种方式的代价都是极其昂贵的。在一些情景下,参考芯片甚至是不存在的,比如掩膜在工厂被修改过。该文提出了一种自适应优化的二元分类型硬件木马检测方法,消除了对生产后的参考芯片的依赖。在电路设计过程中,通常可以假设通过详尽的硅前检测或者经过严格的设计流程,使得原电路的仿真芯片是不含木马的。该文工作尤其适用于检测在设计之后的步骤中插入的木马,比如工厂生产阶段。首先,将木马检测问题建模为二元分类问题,采用电路设计流程中的仿真信息(瞬时功耗)对算法进行训练。经过训练的算法将会生成一个分类器,该分类器能够在芯片生产后测试时自动识别不含木马的和含木马的电路。有许多代表性的分类算法,该文建模了不同的算法并分析评估了哪个算法更适合木马检测。采用的评估指标包括正确分类率、混淆矩阵和查全率。评估了不同工艺偏差下多种算法的性能,给出了在较高水平的工艺偏差下合适的算法设置。更进一步地,考虑到在电路仿真和实际的硅生产之间可能会产生一定的偏差,从而导致该方法的检测性能在实际检测中有所下降,该文提出了一些优化方法来增强这一技术:(1)分析了各分类算法错误分类的芯片编号,提出了匹配算法对的工作方式,用来提供互补的检测性能;(2)提出了关注于检测错误的算法自适应迭代优化,权重调整是基于该算法在上一轮迭代中的分类错误;(3)由于将含木马的电路判定为不含木马的代价高于将不含木马的电路判定为含木马的代价,该文引入了不同检测判决的代价,提出了代价敏感型检测。在ISCAS89基准电路上进行的验证实验结果表明,所提出的方法能够检测已知的和未知的木马,检测精度和查全率均在90%以上。该方法没有添加额外的电路到原电路设计,因此没有引入硬件开销。

关键词 硬件木马检测;硬件安全;参考芯片;分类算法;自适应优化

中图法分类号 TP301 DOI号 10.11897/SP.J.1016.2018.00439

Adaptive Optimization of Two-Class Classification-Based Hardware Trojan Detection Method

XUE Ming-Fu^{1),2)} WANG Jian^{1),2)} HU Ai-Qun³⁾

¹⁾(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016)

²⁾(Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 210023)

³⁾(School of Information Science and Engineering, Southeast University, Nanjing 210096)

Abstract Recently, because of the globalization of the semiconductor design and fabrication process as well as the widely use of third-party intellectual property cores (IP cores), integrated circuits (ICs) are becoming increasingly vulnerable to hardware Trojans (HTs). Hardware Trojan

收稿日期:2016-06-28;在线出版日期:2017-05-24。本课题得到国家自然科学基金青年基金(61602241)、江苏省自然科学基金青年基金(BK20150758)、CCF-启明星辰鸿雁科研计划(CCF-VenustechRP2016005)、CCF-绿盟科技鲲鹏基金(CCF-NSFocus2017003)、中国博士后科学基金面上资助(2014M561644)、江苏省博士后基金科研资助(1402034C)、中国民航信息技术科研基地开放课题基金(CAAC-ITRB-201405)、中央高校基本科研业务费专项资金(NS2016096)资助。薛明富,男,1986年生,博士,讲师,中国计算机学会(CCF)会员,主要研究方向为硬件木马检测、硬件安全。E-mail: mingfu_xue@nuaa.edu.cn。王 箭,男,1968年生,博士,教授,中国计算机学会(CCF)会员,主要研究领域为密码学与信息安全。胡爱群,男,1965年生,博士,教授,主要研究领域为移动网络安全和信息安全。

can make the IC malfunction, leak confidential information, or lead to other catastrophic consequences, thus has raised serious concerns from many critical communities. Most of the existing hardware Trojan detection works require golden chips to provide reference signals. However, obtaining a golden chip is extremely difficult. The golden chips are supposed to be either fabricated by a trusted foundry or verified to be Trojan-free through strict reverse engineering. Both methods are prohibitively expensive. In some scenarios, the golden chips even don't exist, e. g. , if the mask is altered at the foundry. This paper proposes an adaptive optimization of two-class classification-based hardware Trojan detection method which can eliminate the need of fabricated golden chips. In the IC design time, it can be assumed that after thorough pre-silicon detection or under strict design process, the simulated IC of the original design is Trojan-free (golden netlist). This method is particularly suitable for detecting the Trojans which are inserted in the subsequent steps after design, such as the fabrication stage. First, we formulate the hardware Trojan detection problem into a two-class classification problem. Then, we train the classification algorithms using transient power of simulated ICs during IC design flow. The trained algorithms will then form a classifier which can automatically identify fabricated ICs as Trojan-free or Trojan-inserted during test-time. There are many representative types of classification algorithms. We formulate different algorithms and figure out which algorithm is more suitable for HT detection. The metrics accuracy, confusion matrix and recall are used for evaluation. We evaluate the performance of different algorithms against process variations, and present the suitable algorithm settings in the presence of high level of process variations. Moreover, considering there may be a shift which occurs between the IC design simulation and actual silicon fabrication which may cause reduced detection performance during practical post-silicon detection, we also propose several optimized methods to enhance the technique: (1) we analyze the misclassified ICs' numbers of a certain algorithm and present the matched algorithm-pairs to provide complementary detection performance; (2) we propose adaptive iterative optimization of one algorithm by focusing on errors, in which the weight-adjusting are based on how successful the algorithm was in the previous iteration; (3) since the cost of misclassification from Trojan-inserted to Trojan-free is larger than the cost of misclassification from Trojan-free to Trojan-inserted, we use a method of altering algorithms to take into account of the costs of making different hardware Trojan detection decisions, called cost-sensitive detection. The experiment results on ISCAS89 benchmark circuits show that, after optimization, the proposed approach can not only detect known Trojans, but also be able to detect various kinds of unknown Trojans with an accuracy and recall of more than 90%. Since we didn't add any extra circuit to the original design, there is no overhead of this approach.

Keywords hardware Trojan detection; hardware security; golden chips; classification algorithms; adaptive optimization

1 引 言

近年来,对集成电路(Integrated Circuits, ICs)的恶意修改,又称为硬件木马(Hardware Trojan, HT),已成为网络空间安全领域一个新兴的焦点问题^[1-2]. 由于半导体设计和制造过程的全球化,以及

第三方知识产权核(Intellectual Property cores, IP cores)的广泛使用,电路变得越来越容易受到硬件木马的攻击. 硬件木马能够使得电路功能故障、泄露机密信息或者导致其他一些灾难性的后果,严重威胁着上层信息系统的安全性,因此已经引起了工业界、政府、军方和许多关键部门的密切关注^[3-5]. 硬件木马检测技术亟待研究,以确保芯片和片上系统

(System-on-Chip, SoC)的安全与可信。

已有的硬件木马检测方法根据其工作于电路生命周期中所处阶段的不同,可以分为两大类:测试时检测(test-time detection)和实时检测(run-time detection)。前者工作于芯片生产后的出厂检测阶段,后者工作于芯片销售后部署进系统中的实时运行阶段。大多数已有的硬件木马检测工作属于测试时检测,而实时检测工作则相对较少。

在测试时检测(test-time detection)中,测试者将电路的逻辑输出或旁道参数信息与参考电路的相应输出或参数进行对比,一些安全性设计(design-for-security)也相继被提出用来促进测试时的检测。

(1)逻辑测试。攻击者往往巧妙地将硬件木马设计成由电路内部的稀有事件触发,这使得传统的逻辑功能测试很难激活或检测到木马。文献[6]通过多次激励电路内部节点的稀有逻辑条件来增加逻辑测试方法的检测几率。这种方法很难触发有着复杂触发条件的大型木马。

(2)旁道参数分析。许多旁道参数分析方法^[7-9]也先后被提出,通过提取电路的旁道参数信息,如泄露电流、暂态电流、功耗和时延用于检测硬件木马。然而,这些方法易受到工艺偏差(Process Variations, PV)和噪声的影响。此外,面对百万门级的现代电路中插入的细小木马,该方法检测精度大幅度下降。为此,文献[10]提出了基于探索式分区和测试向量生成的硬件木马检测方法,能够实现可控的区域化测量,并促进木马电路的转换活动,提高木马检测精度。

(3)安全性设计。一些区域化激活的方法也被提出用于改进旁道参数分析方法的检测精度^[11-13]。然而,这些方法在检测时采用了大量的随机测试向量,同时还使用了计算耗时的训练过程用于测试模式选择^[14]。文献[15-16]提出将时延结构或者电流传感器电路嵌入到电路中以增强其检测能力。文献[17]通过插入填充细胞到电路的空闲区域来阻止硬件木马的插入。文献[18]提出了电路混淆结构来阻碍木马的插入。然而,这些实现需要在设计阶段改动原电路设计并增加更多的电路,这将会增加电路设计过程的复杂性并产生可观的开销,尤其是对于大规模电路而言。文献[19]提出了基于时域约束估计和主成分投影的子空间硬件木马检测方法,可以在较高水平的工艺偏差和噪声情况下检测出微小木马。

还有一些实时检测(run-time detection)工作,通过增加一些安全结构去监控电路中非预期的逻辑、信号或者行为的改变来检测木马^[20]。这些实时检测工作可以作为测试时检测的一个补充去覆盖芯片的整个生命周期。然而,其主要的缺点是引入了巨大的硬件开销。

大多数已有的硬件木马检测工作需要生产后的参考芯片来提供参考信息。然而,获得一个参考芯片是极其困难的。在某些情形下,参考芯片甚至不存在^[4]。在文献[21]中,我们已初步探索了将硬件木马检测问题建模为分类问题的可能性。本文提出了一种自适应优化的二元分类型硬件木马检测方法,能够以自动化的方式检测硬件木马,同时不需要参考芯片。在电路设计过程中,通常可以假设通过详尽的硅前检测或者经过严格的设计流程,使得原电路的仿真IC是不含木马的。本文工作可以消除对生产后参考芯片的依赖,尤其适用于检测在设计之后的步骤中插入的木马,比如工厂生产阶段。

首先,本文将硬件木马检测问题建模为二元分类问题。然后,利用IC设计流程中的IC仿真信息(瞬时功耗)对分类算法进行训练。训练后的算法将会生成一个分类器,它能够在测试时自动识别生产后的电路是否含有木马。本文评估了不同算法对抗工艺偏差的检测性能,给出了在高水平工艺偏差情形下的合适的算法设置。更进一步地,考虑到在电路仿真和实际的硅生产之间可能会产生一定的偏差,从而导致本方法的检测性能在实际检测中有所下降,本文提出了一些可选的优化方法来增强这一检测技术以更好地应用于实际检测:(1)分析了特定算法的分类错误的IC编号,并给出了匹配算法对的工作方式;(2)提出了关注检测错误的分类算法自适应迭代优化,其权重调整是基于该算法在上一次迭代中的检测错误情况;(3)考虑不同类型硬件木马检测判决的代价,提出了代价敏感型检测。在ISCAS89基准电路上的实验结果表明,基本的分类方法经过上述优化之后,不仅能够检测出已知的木马,还能够检测不同种类的未知木马,检测率较高(90%~100%)。本方法没有添加额外的电路到原电路设计,因此没有引入硬件开销。

本文第2节介绍动机、相关工作分析和问题建模;第3节介绍所提出的基于二元分类的硬件木马检测方法;第4节介绍增强优化的检测方法;第5节给出实验结果;第6节给出总结。

2 动机和问题建模

2.1 动机及已有工作分析

大多数已有的硬件木马检测工作需要参考芯片去提供参考信号. 然而, 获得参考芯片是极其困难的^[4]. 参考芯片通常是假设通过可信的工厂生产出来或者通过很严苛的逆向工程检测验证为不含木马. 两种方式的代价都是极其昂贵的. 在一些情景下, 参考芯片甚至是不存在的, 比如掩膜在工厂被修改过.

近来, 有少量的工作开始研究不需要参考芯片的硬件木马检测方法^[22-25]. 文献[22]通过比较电路自身在两个不同时间窗的暂态电流来识别木马, 然而该方法仅仅针对时序型木马. 文献[24]采用蒙特卡洛仿真提取工艺偏差下的电路指纹来识别木马, 该方法依赖于工艺偏差的精确模型, 而这可能很难获得. 文献[25]通过比较电路自身中相似路径的时延来识别木马的效果. 然而, 它需要原始网表用于生成测试模式, 并且, 当木马对关键路径时延影响较小时, 该方法检测精度降低^[23]. 一般来说, 这些方法都有一些局限性, 它们往往需要较高的计算量、复杂的工艺偏差模型和大量的测试去确保精确度, 尤其对于大规模电路而言^[15]. 这促使本文去提出一个不需要参考芯片的硬件木马检测方法, 同时以自动模式工作, 且具备实用性和低开销.

已有极少量的工作开始探索引入机器学习方法用于检测硬件木马^[26-30]. 文献[26]采用机器学习的方法来使得基于逆向工程破解的硬件木马检测方法中的布局(layout)识别步骤得以自动化执行, 可以减少人工工作量. 然而, 基于逆向工程破解的硬件木马检测方法极其耗时、成本极高并且需要人工参与, 另外, 逆向破解工程只能鉴定被破解的芯片是否可信, 对于大量其他同批次的芯片, 无法确保其可信度, 因为木马极有可能只植入进了部分芯片. 该文的方法中还需要参考芯片进行训练, 而参考芯片难以获得.

文献[27-28]分别将一元支持向量机(Support Vector Machine, SVM)和 K-means 两种机器学习方法应用到基于逆向工程破解的硬件木马检测方法中, 可以消除对参考芯片的依赖. 然而, 基于逆向工程破解的硬件木马检测方法极其耗时、成本极高并且需要人工参与, 且无法确保其他未经逆向破解检测的芯片是否可信.

文献[29]提出面向多核处理器平台的基于 SVM 的实时硬件木马检测方法. 然而, 该训练方法需要木马行为级的特征, 这需要木马触发产生实际的攻击行为, 而在实际硬件木马检测情形中, 木马是保持潜伏状态难以触发的, 并不会产生攻击行为. 此外, 有些木马, 如参数改变型木马和隐通道型木马, 仅仅会改变电路的细微参数, 并不会产生任何实际的攻击行为, 无行为特征可言. 并且该方法对已有电路增加实时检测结构, 会引入较大的硬件开销, 也会增加电路实时运行时的时间开销. 相比之下, 本文的方法是生产后检测, 没有引入硬件结构, 所以没有引入硬件开销, 且可以在木马潜伏状态下检测到木马. 文献[30]在文献[29]的基础上, 用实时的木马行为级特征对训练算法进行自适应学习. 然而, 木马通常是保持潜伏的, 并不会产生攻击行为. 此外, 有些木马仅是改变了电路的细微参数, 并不会产生任何实际的攻击行为. 并且, 该方法用来自适应反馈训练的“实时的木马行为”, 并不能确定就是木马行为, 很有可能只是些偶然的或者特殊的电路行为, 该方法很有可能采用误判的“实时的木马行为”来对算法进行了自适应训练, 使得算法训练错误, 进而导致检测错误.

2.2 问题建模

在测试时检测(test-time detection)中, 一批已生产的 IC 有待检测. 测试者想要判定哪个 IC 是不含木马的, 哪个 IC 是含有木马的. 从机器学习的角度, 木马检测问题可以建模为一个二元分类问题, 定义如下.

定义 1. 给定 N 个生产的 IC, 定义为 $IC_i (1 \leq i \leq N)$, 每个 IC 有多个特征和一个目标变量(Trojan-free 或 Trojan-inserted), 硬件木马检测问题可定义为判定 IC_i 的正确分类, 将它们分类为不含木马的或者含有木马的两类实例, 分别定义为 C_f (Trojan-free) 和 C_i (Trojan-inserted).

在这个基于分类算法的方法中, 需要对算法进行训练来检测木马. 训练的算法将会生成一个分类器, 该分类器能够在测试时识别不含木马的芯片和插入木马的芯片. 本检测方法中的关键元素如下(可对照图 1):

(1) 训练(或学习). 用预处理的数据训练检测算法(分类器).

(2) 训练集. 在本文的模型中, 训练的实例是 IC 设计流程中仿真 ICs 的参数/指纹, 其中一些插入了不同的木马. 具体而言, 在这些仿真 ICs 中, m 个

ICs 是不含木马的, 而 n 个 ICs 是插入木马的, 每个 IC 插入了一种木马(定义为 $HT_1 \sim HT_n$).

(3) 训练算法. 本文评估了许多分类算法, 并进行了优化增强以取得更好的检测精度.

(4) 实例. ICs 的参数, 包括训练集中仿真的 ICs 和测试集中生产的 ICs. 每个实例有多个特征和一个额外的目标值(Trojan-free 或 Trojan-inserted).

(5) 测试集. 生产的 ICs 的参数. 在本文的模型中, 有 $p+q$ 个生产的 ICs 待检测, 其中 q 个生产的 ICs 是插入木马的. 注意到, 由于在实际检测情景中, 人们需要检测未知的木马, 这些测试集中的木马($HT_{1u} \sim HT_{qu}$)不同于训练集中的木马($HT_1 \sim HT_n$).

(6) 检测(测试). 给定 N 个生产的 ICs, 经过训练的分类器将每个 IC 分类为含木马的或不含木马的.

3 基于二元分类的硬件木马检测方法

3.1 总体流程

一般而言, 人们需要检测未知的木马, 这意味着不能假设在训练时能够获得待检测的木马. 本文引入微小的电路改动到原电路设计来表征木马. 对于一个特定电路, 每次引入一种类型的微小改动, 从而生成多个含木马的样本电路. 所插入木马的种类和数量详见实验部分. 虽然这些微小的改动不是真正有恶意功能的木马, 但它们有利于评估本方法的检测性能, 因为如果本方法能够检测出这些微小的改动, 它更加能够检测出带有恶意功能、更大体积的真实木马. 已有工作对硬件木马进行了分类研究, 本文以基本的分类方法——组合型木马和时序型木马为例进行了实验, 结果表明通过少量木马的训练, 本方法可以检测多种未知木马.

本文提出的基于二元分类的硬件木马检测方法的总体流程如图 1 所示, 可划分为两个阶段. 第一阶段是学习阶段(训练时), IC 设计流程中得到的仿真 IC 的瞬时功耗信息(含对工艺偏差的建模)用于训练学习算法. 一些仿真 IC 是不含木马的, 而其他一些 IC 则插入了不同种类的木马. 所提方法的具体实验流程如图 2 所示. 在学习阶段(训练时), 对于一个特定的基准电路, 仿真产生 $m+n$ 个芯片, 其中, m 个 ICs 是不含木马的, 而其他 n 个 ICs 是插入木马的, 每个 IC 插入一种木马(定义为 $HT_1 \sim HT_n$). 每个 IC 的瞬时功耗曲线(指纹)被采样为 500 个点, 作为机器学习中的 500 个属性, 并增加了一个属性以标识它是否含有木马. 注意到, 可以将 IC 指纹采样

为更多的点数以更精确地表征原旁路参数指纹, 然而这会增加检测算法的处理时间. 实验表明采样为 500 个点可以较好地表征原旁路指纹并且算法处理时间较快. 对于不同的 FPGA、ASIC 芯片, 这些参数信息或者属性信息设置方式是通用的. 所采用的学习算法是一些代表性的分类算法, 并进行了建模、评估和分析. 训练之后, 分类算法将会生成分类器用于木马检测阶段.

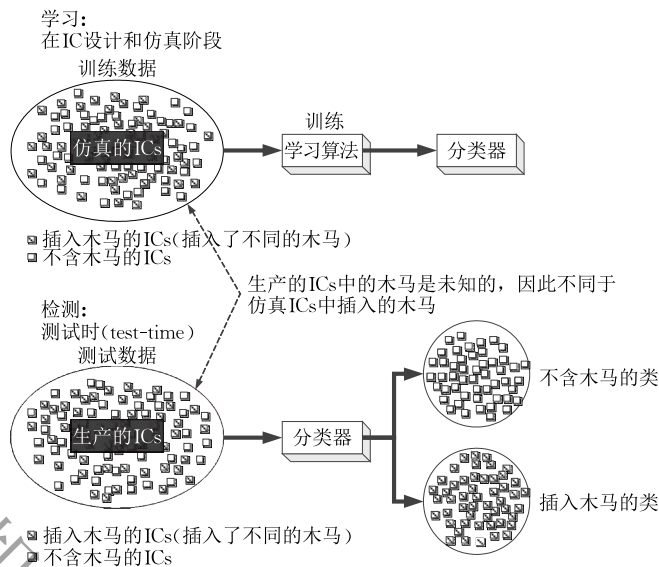


图 1 自适应优化的二元分类型硬件木马检测方法总体流程

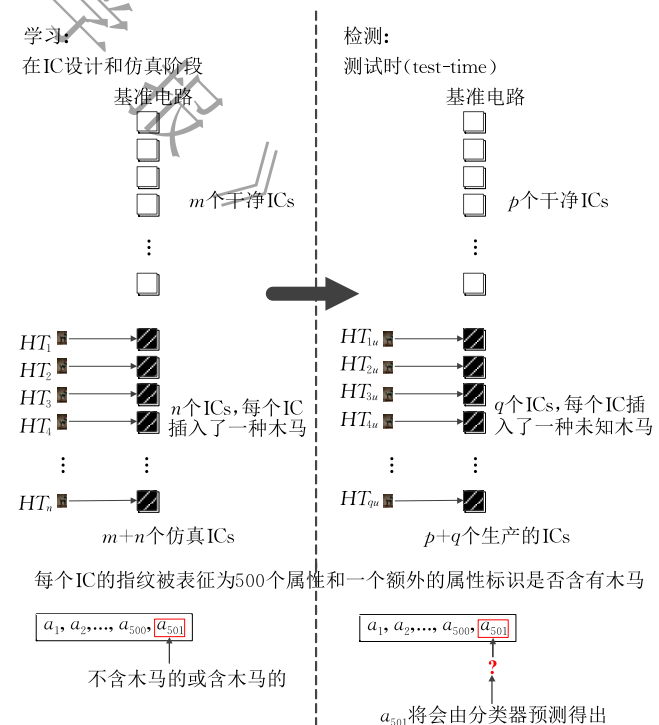


图 2 本文方法的实验流程示意图

第二阶段是检测阶段(测试时), 测试数据是已生产的 IC 的瞬时功耗. 分类器将每个 IC 分类为不

含木马的或者插入木马的,不失一般性,生产的 IC 中的木马最有可能是未知的木马,因此不同于仿真 IC 中的木马. 具体而言,在检测阶段,对于一个特定的基准电路,有 $p+q$ 个生产的 ICs 待检测. 在这些生产的 ICs 中, p 个 ICs 是不含木马的,而其他 q 个 ICs 是插入木马的,每个 IC 插入了一种未知木马(定义为 $HT_{1u} \sim HT_{qu}$). 未知木马意味着这些木马不同于训练阶段的木马. 每个 IC 的指纹也被表征为 500 个属性,而第 501 个属性将会由所提出的方法预测得出. 本方法的一个关键特征是经过仿真阶段一些木马的训练后,该分类器能够在实际测试时自动检测不同类型的未知木马. 需要注意的是,针对某特定电路的训练,不能用于另一种电路的木马检测,这在目前的研究现状下,尚无法达到.

3.2 分类算法

如上所述,本文已经将硬件木马检测问题建模为一个分类问题. 然后,需要确定使用哪种算法,更确切地说,哪种算法更适合硬件木马检测. 一些代表性的分类算法如下: (1) k -Nearest Neighbors; (2) Decision Tree; (3) Naive Bayes; (4) Logistic Regression; (5) Support Vector Machines; (6) Meta-algorithms. 每个算法有各自的优点、缺点、性能和应用情景. 即使缩小了算法的范围后,也没有一个万能的解——哪个算法会得出最好的木马检测结果. 本文将会实践不同的算法并分析评估它们的性能. 此外,本文还将研究一些能够提升特定学习算法性能的方法.

采用这些算法构建具体方法时有一些通用的步骤^[31]. 首先,收集数据. 在本文的实现中,数据是 IC 的旁道参数(瞬时功耗),它可以通过 IC 设计工具或者其他测试设备获得. 第二,准备输入数据,在这步中,本文将会编辑数据为机器学习可使用的格式. 第三,分析输入数据,可以观察这些数据,去分析是否有明显的模式或者规律. 将数据绘图或者使用一些数据可视化的方法,有利于分析输入数据. 第四,训练算法. 用设计阶段的仿真 IC 参数来训练检测算法. 第五,测试算法. 本文用已知的木马和未知的木马测试这些检测算法. 最后,实际使用该检测算法. 所提出的方法能够以自动化的方式用于实际的硬件木马检测.

3.3 评估指标

本文用以下指标评估所提出的方法. 正确分类率(即准确率 *accuracy*),它表示在所有测试的 IC 中算

法正确分类的 IC 所占的比例. 为了反映出 IC 是如何被错误分类的,本文还引入混淆矩阵. 硬件木马检测的情景是一个二元分类问题,因此其混淆矩阵如表 1 所示. 如果算法正确地将一个插入木马的 IC 判定为含木马的,则称为一个真阳性 TP (True Positive). 当算法正确地将不含木马的 IC 判定为不含木马的,则称为一个真阴性 TN (True Negative). 另外两种可能的情形分别是假阴性 FN (False Negative)和假阳性 FP (False Positive),如表 1 所示.

表 1 硬件木马检测问题的混淆矩阵

		预测值	
		含木马	不含木马
实际值	含木马	真阳性 TP	假阴性 FN
	不含木马	假阳性 FP	真阴性 TN

在实际应用中,检测出含木马的 IC 比检测出不含木马的 IC 更为重要,因此本文还使用了查全率 $Recall = \frac{TP}{(TP+FN)}$. 查全率表示所有含木马的 IC 中,分类器正确识别为含木马的 IC 所占的比例,即木马检测率.

4 自适应优化检测方法

本文在基准电路上进行了验证实验,实验结果显示,所提出的基本分类方法能够检测出已知和未知的木马,检测率较高(在 90% 以上). 更进一步地,考虑到在电路仿真和实际的硅生产之间可能会产生一定的偏差,从而导致本方法的检测性能在实际检测中有所下降,本文还提出了一些可选的优化方法去增强上述基本方法的效果.

本文评估了不同分类算法的硬件木马检测性能,这些算法有各自的优点和缺点,一个很自然的改进想法是组合多个算法或者自适应迭代优化某一个算法. 此外,考虑到在硬件木马检测场景中,不同的检测判决的代价不同,引入了代价敏感型检测.

4.1 匹配算法对

组合多个分类算法的预测比使用单一算法能够获得更好的判决. 本文将会验证组合算法能够超越单个分类算法的性能,前提是这些分类算法之间有明显的差别. 这个差别可以是算法本身不同或者应用到算法的数据不同^[31]. 首先,采用各算法独立进行检测,自动记录各算法不正确分类的 IC 编号. 然后,基于已经训练的算法的性能(检测错误 FP 、 FN),选择新的算法提供互补的检测能力. 最后,先

后采用原算法和匹配算法进行检测,达到互补的检测效果.这一改进使得新的分类器更关注于之前分类器错误分类的 IC 或木马,可达到更高的正确分类率.

4.2 单个算法的自适应迭代优化

自适应优化硬件木马检测方法的基本思想是通过关注于检测算法的分类错误(FP、FN)来迭代改进该算法.本文采用一个特定的分类算法作为基分类器,用权重向量对它的输入数据进行加权.权重不是相等的,而是基于上一轮迭代中分类器的检测错误.在第一次迭代中,测试的 IC 是等值加权的,但是在后续的迭代中,先前被错误分类的芯片将会被赋予更多的权重.一种类似的方法叫做 AdaBoost^[31].不像 Adaboost 依赖于一系列不同的分类算法,本文的思想是自适应迭代一个特定算法,优点是在实际应用中只需要实现单个算法,而不是多个算法.如果是硬件实现本系统(如 FPGA 等嵌入式系统),可以迭代复用单个算法结构,而不用消耗大量硬件资源去实现多个算法.

该方法如下工作^[31]:为训练集中的每个 IC 定义权重向量 $\mathbf{W}$.最初,这些权重是等值的.首先用训练数据对分类算法进行训练,并计算分类器的错误.第二次依然采用同一个数据集对分类器进行训练,此时,调整训练集的权重,降低第一次分类中正确分类的 IC 的权重,增加第一次迭代中错误分类的 IC 的权重. AdaBoost 分配 α 值到分类器的每次迭代,其中, α 值基于分类器的错误率 ϵ 计算得出.错误率 ϵ (error rate)表示错误分类的 IC 数目占总的待测试 IC 数目的比例,错误率 ϵ 如下给出:

$$\epsilon = \frac{FP + FN}{number_of_instances} \tag{1}$$

而 α 如下给定:

$$\alpha = \frac{1}{2} \ln \left(\frac{1-\epsilon}{\epsilon} \right) \tag{2}$$

当计算 α 之后,更新权重向量 $\mathbf{W}$,以使得正确分类的 IC 的权重减少,而错误分类的 IC 的权重会增加, $\mathbf{W}$ 如下给定:

$$\begin{cases} W_i^{(t+1)} = \frac{W_i^{(t)} e^{-\alpha}}{Sum(\mathbf{W})}, & \text{如果 IC 预测正确} \\ W_i^{(t+1)} = \frac{W_i^{(t)} e^{\alpha}}{Sum(\mathbf{W})}, & \text{如果 IC 预测不正确} \end{cases} \tag{3}$$

t 是迭代的数目.计算 $\mathbf{W}$ 之后,开始新一轮迭代.重复该迭代训练和权重调整,直到该算法训练的错误为 0 或者达到用户自定义的阈值.

4.3 代价敏感型检测

在硬件木马检测这一情景中,对于含木马的电路和不含木马的电路而言,错误分类的代价是不同的.由于检测含木马的 IC 比检测不含木马的 IC 更重要,本文已经引入了查全率 *recall* 这一指标去评估分类器的性能.更进一步地,可以考虑不同检测判决的代价,为检测器的判决增加一个代价函数,称为代价敏感型检测.

本文采用两种代价矩阵来对比分析.具有相等判决代价的硬件木马检测代价矩阵如表 2 所示.该代价矩阵的总代价如下计算: $TP \times 0 + FN \times 1 + FP \times 1 + TN \times 0$.具有不相等判决代价的硬件木马检测代价矩阵如表 3 所示.这个代价矩阵的总代价计算如下: $TP \times (-1) + FN \times 1 + FP \times (10) + TN \times 0$.采用不相等的代价矩阵,两种错误的木马检测分类会产生不同的代价(1 和 10).相似地,两种正确的分类也会有不同的收益(-1 和 0).当构建一个分类器时,可以设置这些代价参数,从而选择一个最小代价的分类器.有许多方式可以将代价函数包含在分类算法中,比如在 AdaBoost 中,可以基于代价函数调整分类错误的 IC 的权重向量 $\mathbf{W}$ ^[31].

表 2 具有相等判决代价的硬件木马检测代价矩阵

		预测值	
		含木马	不含木马
实际值	含木马	0	1
	不含木马	1	0

表 3 具有不相等判决代价的硬件木马检测代价矩阵

		预测值	
		含木马	不含木马
实际值	含木马	-1	1
	不含木马	10	0

5 实验结果

5.1 实验设置

在 ISCAS89 基准电路上对本方法进行了实验评估.设计了多个木马(包含组合木马和时序木马)分别插入在基准电路上,用于训练和测试的目的.两个基本的木马是一个 4 bit 的比较器(组合型木马 T_1)和一个 3 bit 的计数器(时序型木马 T_2).其他木马是在这两个基本木马的基础上每次增加 2~5 个逻辑门.这些木马的输出是悬空的,使得它们不影响电路的正常输出. T_2 仅占用了 s38417 大小的 0.005%, T_1 更小.这些硬件木马仅是对电路微小的改动,用于评估检测方法的目的.国际上已有的木马

标准 Trust-Hub Benchmarks^①也是可行的,然而,本文实验需要对一个电路分别插入大量的木马进行实验,而 Trust-Hub Benchmarks 是一个电路对应一个特定木马.因此,为了便于实验和自动化处理,本文采用了上述的木马植入方式.尽管这些微小的改动没有实际的攻击功能,它们还是有利于评估本方案的检测性能.因为如果本方法能够检测如此微小的改动,它更加能够检测有着恶意功能和更大体积的真实木马.

原设计和插入木马的设计都采用 Synopsys Design Compiler 在 65 nm TSMC 工艺下进行了合成.用 Synopsys HSPICE 进行了不同工艺偏差 PV 下的电路仿真,并获得了瞬时功耗曲线.每个 IC 的暂态功耗曲线被采样为 500 个点,作为该 IC 的 500 个特征.本文为 IC 增加了第 501 个特征,用来表征该 IC 是不含木马的或者插入木马的.在测试时,IC 的

第 501 个特征是由算法预测得出的目标值.在本文的实验中,晶粒间(inter-die)和晶粒内(intra-die)工艺偏差是 10%、15%、20%(三种实验设置).在 HSPICE 中采用 Monte Carlo 仿真对各种 PV 设置进行了仿真.实验流程如 3.1 节所述,参见图 2.

5.2 基本方法的实验结果

表 4 给出了在三种 PV 设置下 s38417 的检测结果,包含两种实验情形——检测已知木马和检测未知木马.正确分类率(即 *accuracy*)和查全率 *recall* (即木马检测率)的含义如 3.3 节所述.训练集是 100 个仿真 IC 的瞬时功耗曲线,其中 50 个 IC 插入了木马(每个 IC 插入了一种木马).测试集是 100 个 IC 的瞬时功耗曲线,其中 50 个 IC 插入了木马.在实验设置 I 中,测试集中的 50 个木马和训练集中的 50 个木马是相同的,表示检测已知木马.而在实验设置 II 中,测试集中的 50 个木马不同于训练集中

表 4 s38417 在不同工艺偏差设置下的检测结果

算法	木马	PV 10%		PV 15%		PV 20%	
		正确分类率/%	木马检测率/%	正确分类率/%	木马检测率/%	正确分类率/%	木马检测率/%
BayesNet	已知	94	86	95	90	98	96
	未知	91	82	90	80	89	80
NaiveBayes	已知	91	82	89	78	94	88
	未知	90	80	91	82	89	78
NaiveBayes-Updateable	已知	91	82	89	78	94	88
	未知	90	80	91	82	89	78
RBFNetwork	已知	96	92	99	98	99	98
	未知	95	90	95	90	92	84
SimpleLogistic	已知	100	100	100	100	100	100
	未知	96	92	89	84	83	72
SMO	已知	100	100	100	100	100	100
	未知	96	92	93	86	89	80
SPegasos	已知	100	100	100	100	100	100
	未知	98	96	96	92	96	92
IBK	已知	100	100	100	100	100	100
	未知	93	88	87	74	76	70
AttributeSelect-Classifier	已知	98	100	99	98	98	98
	未知	84	90	74	72	61	74
Decorate	已知	100	100	100	100	100	100
	未知	92	84	88	76	81	74
LogitBoost	已知	100	100	100	100	100	100
	未知	93	88	80	74	74	74
HyperPipes	已知	100	100	100	100	100	100
	未知	95	90	94	98	74	94
FT	已知	100	100	100	100	100	100
	未知	96	92	84	82	79	72
LMT	已知	100	100	100	100	100	100
	未知	96	92	89	84	83	72
RandomForest	已知	100	100	100	100	100	100
	未知	95	90	91	82	88	76

的 50 个木马,表示检测未知木马.如表 4 所示,当检测已知的或未知的木马时,几乎所有的分类算法都能够正确鉴别测试的 IC,正确分类率较高(90%以

上).木马检测率(*recall*)与正确分类率(*accuracy*)大致相当,但不是完全相等,这取决于在所有错误分

① <http://www.trust-hub.org/>

类的 IC 中,有多少不含木马的 IC 被错误地分类为含木马的,以及有多少含木马的 IC 被错误地分类为不含木马的.

表 4 还显示了 s38417 在三组工艺偏差设置下的检测结果.可以看出,当检测已知木马时,随着工艺偏差的增加,这些算法的检测性能是相对稳定的.当检测未知木马时,大部分算法的检测性能随着工艺偏差的增加而有所降低.也有一些算法表现出了较好的抗工艺偏差能力,随着工艺偏差增加,其检测性能没有显著下降,比如 BayesNet、NaiveBayes、Logistic、RBFNetwork、SPegasos 等.

表 5 给出了本文工作与已有相关工作的平均正确分类率(*accuracy*)对比,由表可知,本文方法的正确分类率相比文献[26-27,30]有明显提升.文献[28]中,SVM 的正确分类率高于本文方法,然而,文献[28]

采用的基于逆向工程的硬件木马检测方法极其耗时、成本昂贵并且需要人工参与,且无法确保其他未经逆向破解检测的芯片是否可信.

表 5 本文方法与相关工作的平均正确分类率对比

本文方法	[26]	[27]	[28]		[30]	
			K-means	SVM	K-NN	SVM
95%	83.8%	90.67%	92.4%	100%	63.5%	70%

图 3 给出了 s35932 的分类结果.可以看出,当检测已知木马时,分类错误为 0,这意味着所有的 IC 都被正确地分类.当检测未知木马时,6 个 IC 是错误分类的,其中一个不含木马的 IC 被分类为含木马的,而 5 个含木马的 IC 被分类为不含木马的.因此,对于已知木马的正确分类率是 100%,而对于未知木马的正确分类率是 94%.

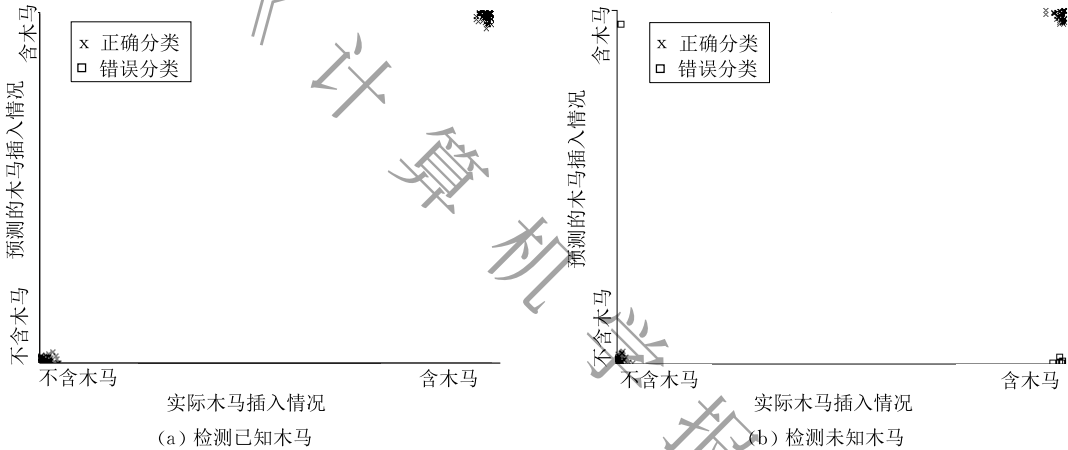


图 3 s35932 的分类结果(采用 SimpleLogistic 算法为例)

5.3 优化方法的实验结果

本文分析了含木马的 IC 被错误分类为不含木马的 IC 编号.在此基础上,给出了特定算法的匹配算法对. s38417 在 10% PV 情形下的匹配算法对如表 6 所示,第 3 列表示含木马的 IC 被错误分类为不

含木马的 IC 的编号.对于某特定算法而言,并不局限于第 4 列中的匹配算法项.第 4 列中的括号表示采用此匹配算法时仍有个别含木马的 IC 逃脱了检测.没有括号表示采用此匹配算法时检测出了所有不含木马的 IC.

表 6 匹配算法对

算法	木马检测率/%	假阴性(FN)的 IC#	匹配算法对
BayesianLogisticRegression	88	2,3,4,7,83,93,94,100	SPegasos (2,100),Logistic
BayesNet	82	2,3,4,70,83,84,93,94,100	SPegasos (2,100),Logistic
NaiveBayes	80	2,3,4,7,70,83,84,93,94,100	SPegasos (2,100),Logistic
Logistic	86	3,21,29,84,93,94,100	LWL (21),SMO
RBFNetwork	90	2,3,93,94,100	LWL (2),OneR
SimpleLogistic	92	2,93,94,100	LWL (2),SMO,J48
SMO	92	2,93,94,100	SPegasos (2,100)
SPegasos	96	2,100	LWL (2),OneR,PART
IB1	88	2,3,4,93,94,100	SPegasos (2,100)
LWL	88	2,4,21,70,83,84	SPegasos (2),VFI
AttributeSelectClassifier	90	2,4,83,84,100	FilteredClassifier (100)
Bagging	88	2,3,4,70,83,84	BayesNet (2)
Dagging	80	2,3,4,7,83,84,93,94,100	SPegasos (2,100)

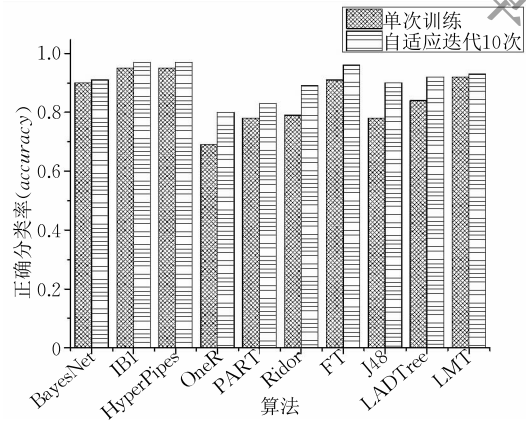
(续 表)

算法	木马检测率/%	假阴性(FN)的 IC#	匹配算法对
ClassificationViaRegression	94	3,93,100	OneR
FilteredClassifier	94	3,93,100	OneR
LogitBoost	88	2,4,83,84,94,100	ClassificationViaRegression (100)
MultiBoostAB	92	2,83,84,93	SPegasos (2)
MultiClassClassifier	86	2,21,29,84,93,94,100	SPegasos (2,100)
HyperPipes	90	2,3,93,94,100	SPegasos, FilteredClassifier
ConjunctiveRule	86	2,8,15,21,70,94,100	SPegasos, FilteredClassifier
DecisionTable	86	2,4,21,70,83,84,100	ClassificationViaRegression
NNge	92	2,93,94,100	OneR (2),J48,PART
OneR	88	2,4,21,70,83,84,	ClassificationViaRegression
PART	86	2,3,4,21,83,84,100	SPegasos (2,100)
Ridor	82	2,3,4,7,70,77,83,84,94	ClassificationViaRegression (3)
ADTree	92	2,83,93,100	SPegasos (2,100)
BFTree	82	2,3,4,12,19,70,83,84,93	SPegasos (2),J48graft
DecisionStamp	88	3,8,21,80,93,100	SPegasos (100),J48graft
FT	92	2,93,94,100	SPegasos (2,100),OneR
J48	88	2,4,70,83,84,100	ClassificationViaRegression (100)
J48graft	88	2,4,21,70,83,84	ClassificationViaRegression
REFTree	84	2,3,4,7,64,70,83,93	SPegasos (2)
SimpleCart	80	2,3,4,12,19,21,70,83,84,93	SPegasos (2),J48

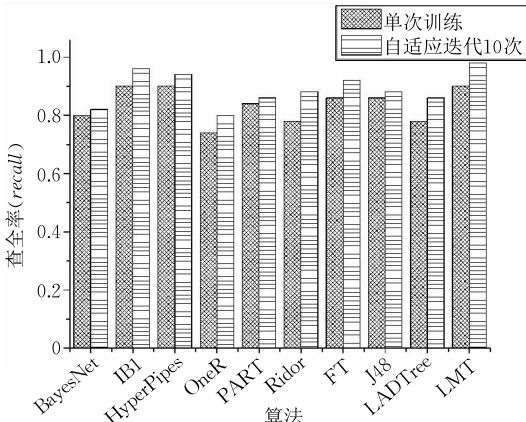
s35932 的自适应优化检测结果如图 4 所示,给出了一个算法单次测试的性能与该算法自适应迭代 10 次的性能对比.可以看出通过该算法的权重调整

迭代,正确分类率和查全率(即木马检测率)都明显地提升了.

本文评估了采用两种代价矩阵设置情形下的代价敏感型硬件木马检测.相等的代价矩阵如表 2 所示,不相等的代价矩阵如表 3 所示. s38417 在 10% PV 下检测未知木马的代价敏感型检测结果如图 5 所示.可以看出,虽然这些算法有大致相当的正确分类率,它们的检测代价存在一定的差异,尤其是在代价矩阵 $\mathbf{II}$ 之下.有些算法的总体判决代价为负,表明该算法判决的收益大于代价.当构建一个分类器时,测试者可以评估木马检测代价,从而选择一个最小代价的分类器,或者对检测率和代价取折衷的选项.



(a) 正确分类率对比



(b) 查全率对比

图 4 s35932 单次算法检测结果对比自适应迭代 10 次的检测结果(检测未知木马)

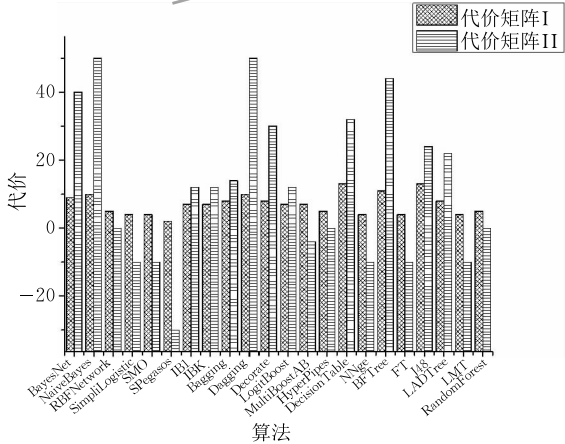


图 5 s38417 的代价敏感型检测结果(PV 10%,检测未知木马)

6 总 结

本文提出了一个自适应优化的二元分类型硬件

木马检测方法,可以消除已有方法对生产后的参考芯片的依赖。首先,将硬件木马检测问题建模为一个二元分类问题,然后用 IC 设计流程中的仿真 IC 训练分类算法。训练的算法将会生成一个分类器,该分类器能够在测试时自动识别不含木马的和含木马的 IC。评估了不同算法对抗工艺偏差的检测性能,给出了在高水平工艺偏差情形下的合适的算法设置。更进一步地,考虑到在仿真和实际的硅生产之间可能会产生一定的偏差,从而导致本方法的检测性能在实际检测中有所下降,本文还提出了一些可选的优化方法来增强这个技术:(1)分析了某特定算法错误分类的 IC 编号,提出了匹配算法对的工作方式;(2)提出了关注于检测错误的单个分类算法自适应迭代优化,权重调整是基于该算法在上一次迭代中的检测错误情况;(3)考虑硬件木马检测不同判决的代价,提出了代价敏感型检测。在基准电路上的实验结果表明,所提出的优化方法能够检测已知的和未知的木马,检测率较高(90%以上)。由于本方法没有增加额外的电路到原电路设计,因此没有引入硬件开销。

参 考 文 献

- [1] Bhunia S, Hsiao M S, Banga M, Narasimhan S. Hardware trojan attacks: Threat analysis and countermeasures. *Proceedings of the IEEE*, 2014, 102(8): 1229-1247
- [2] Rostami M, Koushanfar F, Karri R. A primer on hardware security: Models, methods, and metrics. *Proceedings of the IEEE*, 2014, 102(8): 1283-1295
- [3] Swierczynski P, Fyrbiak M, Koppe P, Paar C. FPGA Trojans through detecting and weakening of cryptographic primitives. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2015, 34(8): 1236-1249
- [4] Bhunia S, Abramovici M, Agarwal D, et al. Protection against hardware Trojan attacks: Towards a comprehensive solution. *IEEE Design & Test of Computers*, 2013, 30(3): 6-17
- [5] Salmani H, Tehranipoor M M. Vulnerability analysis of a circuit layout to hardware Trojan insertion. *IEEE Transactions on Information Forensics and Security*, 2016, 11(6): 1214-1225
- [6] Chakraborty R S, Wolff F, Paul S, et al. MERO: A statistical approach for hardware Trojan detection//*Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems (CHES)*. Lausanne, Switzerland, 2009: 396-410
- [7] Agrawal D, Baktir S, Karakoyunlu D, et al. Trojan detection using IC fingerprinting//*Proceedings of the IEEE Symposium on Security and Privacy (SP)*. Berkeley, USA, 2007: 296-310
- [8] Nowroz A N, Hu Kangqiao, Koushanfar F, Reda S. Novel techniques for high-sensitivity hardware trojan detection using thermal and power maps. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2014, 33(12): 1792-1805
- [9] Potkonjak M, Nahapetian A, Nelson M, Massey T. Hardware Trojan horse detection using gate-level characterization//*Proceedings of 46th Design Automation Conference (DAC)*. San Francisco, USA, 2009: 688-693
- [10] Xue Ming-Fu, Hu Ai-Qun, Wang Jian. A novel hardware Trojan detection technique using heuristic partition and test pattern generation. *Acta Electronica Sinica*, 2016, 44(5): 1132-1138(in Chinese)
(薛明富, 胡爱群, 王箭. 基于探索式分区和测试向量生成的硬件木马检测方法. *电子学报*, 2016, 44(5): 1132-1138)
- [11] Narasimhan S, Du D, Chakraborty R S, et al. Hardware Trojan detection by multiple-parameter side-channel analysis. *IEEE Transactions on Computers*, 2013, 62(11): 2183-2195
- [12] Salmani H, Tehranipoor M. Layout-aware switching activity localization to enhance hardware Trojan detection. *IEEE Transactions on Information Forensics and Security*, 2012, 7(1): 76-87
- [13] Wei S, Potkonjak M. Scalable hardware Trojan diagnosis. *IEEE Transactions on Very Large Scale Integration Systems*, 2012, 20(6): 1049-1057
- [14] Xue M F, Hu A Q, Huang Y, et al. Monte Carlo based test pattern generation for hardware Trojan detection//*Proceedings of the 11th IEEE International Conference on Dependable, Autonomous and Secure Computing (DASC2013)*. Chengdu, China, 2013: 131-136
- [15] Cao Y, Chang C H, Chen S S. A cluster-based distributed active current-sensing circuit for hardware Trojan detection. *IEEE Transactions on Information Forensics and Security*, 2014, 9(12): 2220-2231
- [16] Lamech C, Plusquellic J. Trojan detection based on delay variations measured using a high-precision, low-overhead embedded test structure//*Proceedings of the International Symposium on Hardware-Oriented Security and Trust (HOST)*. San Francisco, USA, 2012: 75-82
- [17] Xiao K, Forte D, Tehranipoor M. A novel built-in self-authentication technique to prevent inserting hardware Trojans. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2014, 33(12): 1778-1791
- [18] Xue M F, Wang J, Wang Y D, et al. Security against hardware Trojan attacks through a novel chaos FSM and delay chains array PUF based design obfuscation scheme//*Proceedings of the International Conference on Cloud Computing and Security*, Springer-Verlag's Lecture Notes in Computer Science (LNCS). Nanjing, China, 2015, 9483: 14-24
- [19] Xue M F, Liu W, Hu A Q, et al. Detecting hardware Trojan

- through time domain constrained estimator based unified subspace technique. *IEICE Transactions on Information and Systems*, 2014, E97-D(3): 606-609
- [20] Abramovici M, Bradley P. Integrated circuit security-new threats and solutions//*Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies (CSIIRW)*. Oak Ridge, USA, 2009, article 55: 1-3
- [21] Xue M F, Wang J, Hu A Q. An enhanced classification-based golden chips-free hardware Trojan detection technique //*Proceedings of the IEEE Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. Taipei, China, 2016: 1-6
- [22] Narasimhan S, Wang X, Du D, et al. TeSR: A robust temporal self-referencing approach for hardware Trojan detection//*Proceedings of the 4th International Symposium on Hardware-Oriented Security and Trust (HOST)*. San Diego, USA, 2011: 71-74
- [23] Zheng Y, Yang S, Bhunia S. SeMIA: Self-similarity based IC integrity analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2016, 35(1): 37-48
- [24] Liu Y, Huang K, Makris Y. Hardware Trojan detection through golden chip-free statistical side-channel fingerprint//*Proceedings of the Design Automation Conference (DAC)*. San Francisco, USA, 2014: 1-6
- [25] Yoshimizu N. Hardware Trojan detection by symmetry breaking in path delays//*Proceedings of the International Symposium on Hardware-Oriented Security and Trust (HOST)*. Arlington, USA, 2014: 107-111
- [26] Nasr A A, Abdulmageed M Z. Automatic feature selection of hardware layout: A step toward robust hardware Trojan detection. *Journal of Electronic Testing*, 2016, 32(3): 357-367
- [27] Bao C, Forte D, Srivastava A. On application of one-class SVM to reverse engineering-based hardware Trojan detection //*Proceedings of the 15th International Symposium on Quality Electronic Design*. Santa Clara, USA, 2014: 47-54
- [28] Bao C, Forte D, Srivastava A. On reverse engineering-based hardware Trojan detection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2016, 35(1): 49-57
- [29] Kulkarni A, Pino Y, Mohsenin T. SVM-based real-time hardware Trojan detection for Many-Core platform//*Proceedings of the 17th International Symposium on Quality Electronic Design*. Santa Clara, USA, 2016: 362-367
- [30] Kulkarni A, Pino Y, Mohsenin T. Adaptive real-time trojan detection framework through machine learning//*Proceedings of the International Symposium on Hardware-Oriented Security and Trust (HOST)*. McLean, USA, 2016: 120-123
- [31] Harrington P. *Machine Learning in Action*. Shelter Island, USA: Manning Publications, 2012



XUE Ming-Fu, born in 1986, Ph.D., assistant professor. His research interests include hardware Trojan detection, hardware security and information security.

WANG Jian, born in 1968, Ph.D., professor. His research interests include cryptography and information security.

HU Ai-Qun, born in 1965, Ph.D., professor. His research interests include mobile network security and information security.

Background

Recently, the malicious alteration of integrated circuits (IC), also referred as hardware Trojans (HTs), has become an emerging security problem in the hardware community. Hardware Trojan can make the IC malfunction, leak confidential information, or lead to other catastrophic consequences, thus has raised serious concerns from many critical communities.

Most of the existing works require fabricated golden chips to provide reference information. However, obtaining a golden chip is extremely difficult. In some scenarios, the

golden chips even don't exist. A few methods are proposed to detect HT without golden chips by using self-authentication techniques. However, these methods always need expensive computations, sophisticated process variation models and a large number of measurements to ensure accuracy for large designs.

Recently, a rare few works use machine learning methods to detect hardware Trojans. One kind of this work is using machine learning to facilitate the layout identification step of reverse engineering-based detection. However, destructive

reverse engineering is very difficult with expensive cost and lots of manual efforts. Moreover, reverse engineering can only be used in a sample of chips while the trust of the remaining chips cannot be guaranteed. Another work is using hardware Trojan’s behaviors to train the machine learning algorithms which is applied for run-time detection. However, the hardware Trojans are typically latent which is difficult to trigger, thus it is difficult to obtain hardware Trojans’ behavior features. Moreover, some types of Trojans only modify the parameters of the IC or construct a covert channel. These types of Trojans have no behaviors to characterize.

This paper presents a novel adaptive optimization of two-class classification-based hardware Trojan detection method which can eliminate the need of fabricated golden chips. We formulate the hardware Trojan detection problem into a two-class classification problem. Then, we train the classification algorithms using simulated ICs during IC design flow. The trained algorithms will then form a classifier which can automatically identify fabricated ICs as Trojan-free or

Trojan-inserted during test-time. Moreover, we also propose several optional optimized methods to enhance the technique, including matched algorithm-pairs, adaptive iterative optimization of one algorithm by focusing on errors, and cost-sensitive detection. The experiment results on ISCAS89 benchmark circuits show that, after optimization, the proposed approach can detect both known Trojans and various kinds of unknown Trojans with a high accuracy (more than 90%).

This work is supported by the National Natural Science Foundation of China (No. 61602241), the Natural Science Foundation of Jiangsu Province (No. BK20150758), the CCF-Venustech Hongyan Research Plan (No. CCF-VenustechRP2016005), the CCF-NSFocus Kunpeng Foundation (No. CCF-NSFocus2017003), the Chinese Postdoctoral Science Foundation (No. 2014M561644), the Jiangsu Province Postdoctoral Science Foundation (No. 1402034C), and the Fundamental Research Funds for the Central Universities (No. NS2016096).