doi:10.3772/j.issn.1002-0470.2022.02.001

可调节跳变概率硬件木马检测方法①

卢新元②* ** ***　许　超****　陈华军****　章隆兵* ** ***　王　玥*****

(* 计算机体系结构国家重点实验室(中国科学院计算技术研究所)　北京 100190)
(** 中国科学院计算技术研究所　北京 100190)
(*** 中国科学院大学　北京 100049)
(**** 龙芯中科技术有限公司　北京 100190)
(***** 黑龙江省公安厅　黑龙江 150008)

摘　要　研究了芯片设计和制造过程中的硬件木马植入方法和检测技术,考虑到现有的检测方法存在木马激活时间较长或面积开销较大的问题,提出一种可调节跳变概率的加速硬件木马检测方法。该方法根据电路拓扑结构,采用权值替换策略动态选择插入实现跳变概率调节的二路选择器的顺序,提高电路中稀有节点的跳变概率,降低木马激活时间,加速硬件木马检测,优化了面积开销。在 ISCAS'89 基准电路的实验结果表明,同现有的加速木马检测方法相比,本文方法的面积开销节省了 44.1% ~68.9%,稀有节点的平均跳变概率提高了 19.0% ~49.1%,且电路规模越大,效果越明显。

关键词　硬件木马; 跳变概率; 稀有节点; 二路选择器; 加速木马检测结构

0　引　言

随着芯片设计和制造的全球化,芯片供应链上的各个阶段由不同的工厂或机构分工完成,在实现芯片产业高速发展的同时,也为攻击者植入硬件木马电路[1]提供了机会,芯片正面临着日趋严重的安全威胁。硬件木马隐藏在原始电路中,在特定条件下触发并造成信息泄露或停止服务等严重问题,同时很难被传统的测试方法检测,因此如何有效地防范和检测硬件木马,成为近年来芯片安全领域研究的重点。根据硬件木马种类和电路规模的不同,研究人员提出了多种硬件木马检测方法[2-3],其中非破坏性检测方法可分为侧信道分析技术和逻辑测试技术。侧信道分析技术[4-6]通过分析木马电路被激活或部分激活后导致的功耗、时序、电磁等侧信道信息异常来检测待测电路中是否存在硬件木马,适合大规模电路,但木马检测的准度容易受工艺偏差和测量噪声等因素的影响。逻辑测试技术[7-11]利用可测试性技术对电路施加信号激励,通过对比输出结果与期望值来判断电路中是否被植入木马,有效地避免了工艺偏差和测量噪声等因素的影响。逻辑测试技术是目前最直接有效的木马检测方法,但通常需要生成大量的测试向量来激活硬件木马,只适合小规模电路。

针对逻辑测试技术中硬件木马难以激活的问题,文献[12-16]提出了基于跳变概率提高的加速硬件木马检测方法,通过在电路中添加加速木马检测结构,提高了稀有节点到达稀有值状态的概率,增加了木马的激活概率,减少了木马激活时间。文献[12]通过在电路稀有节点处插入与或门和虚拟扫描触发器(dummy scan flflip-flflop,DSFF)结构,有效地缩短

① 国家自然科学基金(61521092),中国科学院战略性先导科技专项(XDC05020000)和中国科学院重点部署(ZDRW-XH-2017-1)资助项目。
② 男,1994 年生,博士生;研究方向:计算机系统结构,芯片验证与测试;联系人,E-mail: luxinyuan@ict.ac.cn。
(收稿日期:2021-01-15)

了木马激活时间,但该方法存在较大的面积、延迟及功耗开销。文献[13]用二路选择器(2-to-1 multiplexer,MUX)代替与或门,并优化插入选择算法,减少了插入点个数,但该结构依然存在一定程度的面积开销,且算法的时间复杂度较高,不适用于较大规模电路。文献[14]同样通过插入二路选择器提高稀有节点的跳变概率,选择将电路中部分节点的值替换为常值 0 或 1,但该方法缺乏插入点选择过程的优化,只是通过降低阈值来满足预设定的面积开销。文献[15]提出了权值随机向量的方法,有效地提高了稀有节点的跳变概率,但权值随机向量生成和选择过程进一步增加了面积和测试时间开销。文献[16]提出了基于最大布尔可满足性加速木马检测的方法,该方法能够最大化稀有节点的跳变概率,但无法检测到非稀有节点作为触发节点之一的木马。文献[13,14]中加速木马检测结构的概率替换方式较为单一,可以一定程度上提高稀有节点的跳变概率,但针对复杂的电路拓扑结构,无法最大化稀有节点的跳变概率,并且存在较大的面积开销。

针对以上方法存在的问题,本文提出一种可调节跳变概率的加速硬件木马检测方法,通过在芯片设计阶段插入特殊的二路选择器来提高稀有节点的跳变概率,加速硬件木马检测。该方法根据电路的拓扑结构分析稀有节点的扇入扇出,采用权值替换策略动态选择最优的插入点顺序,最大化电路中稀有节点的跳变概率,降低木马激活时间,且面积和时间开销可忽略不计。此外,该方法适用于较大规模电路。

1　背景介绍

1.1　硬件木马原理

硬件木马电路由触发模块和负载模块组成,其中触发模块负责激活木马电路,负载模块则是在木马激活后负责攻击的电路模块。触发模块分为数字触发和模拟触发,本文主要讨论和研究数字触发的硬件木马电路问题。攻击者通常会选取电路中的稀有值状态作为木马触发模块的条件,以文献[12]中展示的电路结构为例,电路输入端口的激励值默认完全随机,即输入端口值为 0 或 1 的概率都为 0.5,电路中各个节点状态值的概率通过概率分析计算后如图 1 所示。

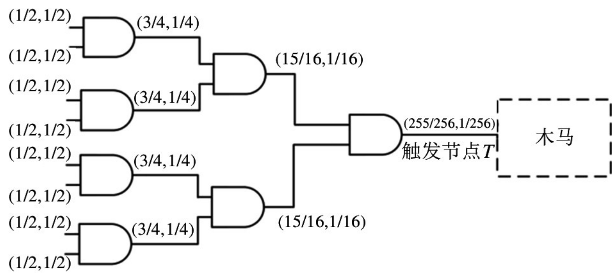


图 1　电路中各节点状态值的概率

图中节点 T 值为 1 的概率为 1/256,满足稀有值的条件,因此攻击者通常会选取该节点 T 作为木马触发模块的输入,当节点 T 满足稀有值状态 1 时,木马被激活。文献[12]将节点值为 0 的概率 $P0$ 与值为 1 的概率 $P1$ 的乘积定义为该节点的跳变概率 TP,并将电路中跳变概率小于阈值的节点定义为稀有节点。节点的跳变概率越低表明该节点从非稀有值状态跳变为稀有值状态所需的时钟周期数越长,由该节点作为输入触发木马被激活所需的时间越长,因此提高稀有节点的跳变概率可以有效地降低木马电路激活所需时间,加速硬件木马检测。

1.2　经典的加速木马检测结构

旨在提高跳变概率的加速硬件木马检测方法是在稀有节点处插入特殊的电路结构,将电路中稀有节点的跳变概率提高到阈值以上,通过缩短稀有节点到达稀有值状态的时钟周期数来降低硬件木马触发节点激活所需时间。文献[13]提出的加速木马检测结构如图 2 所示。

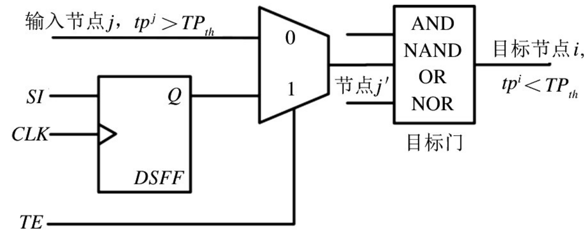


图 2　文献[13]中的加速结构

图中目标节点 i 的跳变概率小于阈值,满足稀有节点条件,根据插入算法选择在目标节点 i 扇入

门的输入节点 j 处插入加速木马检测结构。该结构由 1 个 2-to-1 MUX 和 1 个 DSFF 组成,MUX 的输入端分别连接输入节点 j 以及扫描触发器的输出端 Q,MUX 的输出端同目标门相连,MUX 的选择信号由使能信号 TE 控制。TE 为 0 时,电路处在功能模式下,选择节点 j 作为 MUX 输出,电路的原有逻辑保持不变。当 TE 切换为 1 后,电路进入测试模式,此时 MUX 输出结果同扫描触发器 DSFF 的值保持一致,节点 j' 值为 0 和 1 的概率($P_{j'}0$, $P_{j'}1$)被替换为(0.5,0.5)。该结构可以保证在功能模式下,电路的功能逻辑不受影响,且在测试模式下,替换节点的状态值由扫描触发器进行控制,从而提高目标节点的跳变概率。

1.3 权值替换分析

文献[13]选择将所有需替换节点的概率统一替换为(0.5,0.5),但这并不是适合所有情况的最优权值替换选择,以图 3 中 3 种不同的情况为例进行说明。

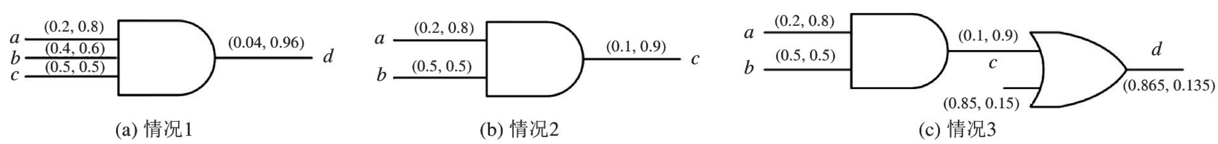


图 3　不同情况下的权值替换分析

图 3(a)中三输入与门输出节点 d 的跳变概率可计算为 0.0384,假定跳变概率阈值设为 0.1,则需要提高输出节点 d 的跳变概率,使其满足阈值要求。根据文献[13]中的方法,选取该与门所有输入中值为 1 的概率最低的节点 a 进行加速木马检测结构插入,将节点 a 状态值的概率替换为(0.5,0.5),输出节点 d 的跳变概率被提高为 0.09,但依然小于阈值,则需要进一步在节点 b 处插入加速木马检测结构,这显然增加了插入点的数量。由跳变概率计算公式可知,若使得输出节点 d 值为 1 的概率大于 0.113,则可以保证 d 的跳变概率满足阈值,即输入节点 a 在替换后值为 1 的概率应大于 0.563。例如将节点 a 的概率 (P_a0, P_a1) 替换为(0.8,0.2),则节点 d 的跳变概率为 0.1344,大于阈值。

图 3(b)中两输入门的输出节点 c 的跳变概率计算为 0.09,阈值设定为 0.1,同样需要在输入节点 a 处插入加速木马检测结构。在这种情况下将概率替换为(0.5, 0.5)或(0.8, 0.2)都可以满足输出节点 c 的跳变概率大于阈值,但替换为(0.8,0.2)是更优的选择,因为跳变概率越大,木马触发节点被激活所需的时钟周期数越短。

但并非所有的情况下将输入节点的概率值取反都是最优的权值替换选择,例如图 3(c)中的结构。同样地,将节点 a 的概率替换为(0.5,0.5)或(0.8, 0.2),节点 c 的跳变概率都能大于阈值,但同时会影响节点 d 的跳变概率,计算后发现需要用(0.5,0.5)替换才可以满足节点 d 的跳变概率大于阈值,为减少插入点数量,降低加速木马检测结构的面积开销,这种情况下将输入节点 a 的概率修改为(0.5, 0.5)是最优的权值替换选择。

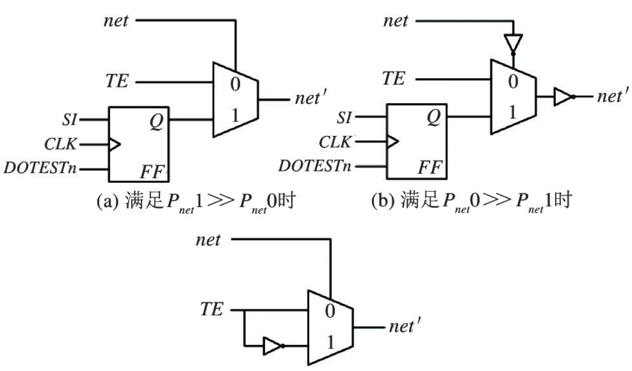
2　可调节跳变概率的加速木马检测方法

本文提出了一种可调节跳变概率的加速木马检测结构,并设计了相应的权值选择和插入点选择算法,根据电路的拓扑结构选择合适的插入结构,并动态规划插入点顺序。

2.1 可调节跳变概率的结构实现

可调节跳变概率的加速木马检测结构如图 4(a)和 4(b)所示,当替换节点 net 的概率满足 $P_{net}1 >> P_{net}0$ 时,选择插入图 4(a)中结构;当替换节点 net 的概率满足 $P_{net}0 >> P_{net}1$ 时,则选择图 4(b)中结构进行插入。图 4(a)中结构由一个扫描触发器和一个 2-to-1 MUX 组成,触发器的 Q 连接 MUX 的一个输入,测试模式控制信号 TE 则和 MUX 的另外一个输入端相连。MUX 的选择信号由电路中替换节点 net 控制,MUX 的输出则作为替换后的节点 net',图 4(b)中结构是在图 4(a)中结构的基础上额外添加了 2 个反相器。根据图 4(a)和 4(b)中的加速木马检测结构,替换后节点 net' 的概率计算公式如

表 1所示。



(a) 满足 $P_{net}1 \gg P_{net}0$ 时　(b) 满足 $P_{net}0 \gg P_{net}1$ 时

(c) 当替换方式是概率值取反，优化后的加速木马检测结构

图 4　本文提出的加速木马检测结构

以替换节点 *net* 值的概率 $P_{net}1 >> P_{net}0$ 的情况为例,替换后节点 *net′* 的概率计算公式如表 1 中式(1.1)和(1.2)所示。当电路处在功能模式下,使能信号 *TE* 为 0,则可简化为式(1.3)和(1.4),将扫描触发器的值置为1并保持不变,此时替换后节点 *net′* 的值同节点 *net* 的值完全一致,该结构可以保证电路的功能逻辑不受影响。因此,在电路进入功能模式之前,工程师需要通过复位信号 *DOTESTn* 将新增扫描触发器的值全部置 1,来保证功能模式下电路的功能逻辑不变。当电路进入测试模式时,使能信号 *TE* 为 1,替换后节点 *net′* 的概率计算公式可简化为式(1.5)和(1.6),若扫描触发器 *Q* 端值的概率($P_{FF}1$, $P_{FF}0$)满足(0.5, 0.5),则替换后节点 *net′* 的概率($P_{net'}0$, $P_{net'}1$)趋近于(0.5, 0.5);若扫描触发器 *Q* 端值的概率($P_{FF}1$, $P_{FF}0$)满足(0,1),即保持为常值 0,则替换后节点 *net′* 值的概率满足 $(P_{net'}1, P_{net'}0) = (P_{net}0, P_{net}1)$。因此,该加速木马检测结构既可以将替换节点 *net* 值的概率修改为(0.5, 0.5),也可以修改为节点 *net* 值的概率取反后的值,需要根据电路的拓扑结构进行判断,选择最优的修改方式,在保证提高稀有节点跳变概率的前提下,最大化减少插入点的数量,降低面积开销。

表 1　替换后节点 *net′* 的概率计算公式

	当 $P_{net}1 >> P_{net}0$ 时,选择插入图 4(a)中结构		当 $P_{net}0 >> P_{net}1$ 时,选择插入图 4(b)中结构	
概率计算公式	$P_{net'}0 = P_{net}0 \times P_{TE}0 + P_{net}1 \times P_{FF}0$	(1.1)	$P_{net'}0 = P_{net}0 \times P_{FF}1 + P_{net}1 \times P_{TE}1$	(2.1)
	$P_{net'}1 = P_{net}0 \times P_{TE}1 + P_{net}1 \times P_{FF}1$	(1.2)	$P_{net'}1 = P_{net}0 \times P_{FF}0 + P_{net}1 \times P_{TE}0$	(2.2)
功能模式, *TE* = 0	$P_{net'}0 = P_{net}0 \times 1 + P_{net}1 \times P_{FF}0$	(1.3)	$P_{net'}0 = P_{net}0 \times P_{FF}1$	(2.3)
	$P_{net'}1 = P_{net}1 \times P_{FF}1$	(1.4)	$P_{net'}1 = P_{net}0 \times P_{FF}0 + P_{net}1$	(2.4)
测试模式, *TE* = 1	$P_{net'}0 = P_{net}1 \times P_{FF}0$	(1.5)	$P_{net'}0 = P_{net}0 \times P_{FF}1 + P_{net}1$	(2.5)
	$P_{net'}1 = P_{net}0 \times 1 + P_{net}1 \times P_{FF}1$	(1.6)	$P_{net'}1 = P_{net}0 \times P_{FF}0$	(2.6)

当选择用替换节点概率值取反的方式进行修改时,可以将该加速木马检测结构进一步优化,优化后的结构如图 4(c)所示。同图 4(a)和 4(b)中结构相比,图 4(c)中选择用反相器代替扫描触发器,反相器会将 *TE* 值取反并同 2-to-1 MUX 输入端相连。该优化后的结构既可以保证测试模式下将替换节点处的概率值修改为取反后的值,又可以保证功能模式下电路的功能逻辑不变,并且一定程度上减少了需要添加扫描触发器的数量,进一步降低了面积开销。

根据图 4(a)和 4(b)中结构可知,当扫描触发器 *Q* 端值的概率($P_{FF}1$, $P_{FF}0$)满足(0.5, 0.5)时,节点 *net′* 概率并非为(0.5, 0.5),实际概率为通过式(1.5)、(1.6)、(2.5)和(2.6)计算后所得。为了便于描述,本文定义:

(1) 采用图 4(a)和 4(b)中结构修改后得到的概率值称为平均权值,该替换方式称为平均权值替换。此时扫描触发器 *Q* 端值的概率($P_{FF}1$, $P_{FF}0$)需满足(0.5, 0.5)。

(2) 采用图 4(c)中结构修改后得到的概率值称为反向权值,该替换方式称为反向权值替换。

2.2　权值选择算法

为了保证在替换节点处能够选择最优的加速检测木马结构进行替换,需要对电路中稀有节点的扇出情况进行分析,为此本文提出如算法 1 所示的权值选择算法。本文定义:节点的扇出路径上同该节点相连的门定义为该节点的扇出门;节点的扇入路径上同该节点相连的门定义为该节点的扇入门。节

点所有扇出门的输出端口定义为该节点的扇出节点;节点扇入门的所有输入端口定义为该节点的扇入节点。

算法 1 权值选择算法

输入:稀有节点 NET_{rare},网表 *Netlist*,跳变概率阈值 TP_{th}

输出:替换权值 *WSP*

//初始化:

1. 获取稀有节点 NET_{rare} 的扇入节点集合 *FIN*,扇出节点集合 *FOUT* 及扇入门类型
2. 从扇入节点集合 *FIN* 中挑选替换节点

   If(扇入门类型为与门/与非门)

   　　选择 *FIN* 中值为 1 的概率最小且未被替换过的节点作为替换节点 *net*

   Else if(扇入门类型为或门/或非门)

   　　选择 *FIN* 中值为 0 的概率最小且未被替换过的节点作为替换节点 *net*

   End if

//计算跳变概率:

3. 分别用平均权值和反向权值这两种替换权值修改替换节点处的概率值,并计算稀有节点 NET_{rare} 的跳变概率 $tp_{WSP_{0.5}}$ 和 $tp_{WSP_{\mathrm{invert}}}$

//选择替换权值:

4. 比较 $tp_{WSP_{0.5}}$ 和 $tp_{WSP_{\mathrm{invert}}}$ 同阈值的大小关系

   If($tp_{WSP_{0.5}}$ 大于或等于阈值且 $tp_{WSP_{\mathrm{invert}}}$ 小于阈值)

   　　选择平均权值作为替换权值

   Else if($tp_{WSP_{\mathrm{invert}}}$ 大于或等于阈值且 $tp_{WSP_{0.5}}$ 小于阈值)

   　　选择反向权值作为替换权值

   Else if($tp_{WSP_{\mathrm{invert}}}$ 和 $tp_{WSP_{0.5}}$ 都大于或等于阈值)

   计算两种权值替换下集合 *FOUT* 中各扇出节点的跳变概率,统计跳变概率大于或等于阈值的扇出节点总个数,选择使得总数更大的权值作为替换权值,若总个数相同,选择使得稀有节点 NET_{rare} 跳变概率更大的权值作为替换权值

Else

选择使得稀有节点 NET_{rare} 跳变概率更大的权值作为替换权值

End if

返回替换权值 *WSP*

算法 1 的输入为稀有节点 NET_{rare},插完扫描链[17]的门级网表以及预先设定的跳变概率阈值 TP_{th},输出则为最优的替换权值 *WSP*。首先需要获取该稀有节点的扇入节点集合 *FIN*、扇出节点集合 *FOUT* 以及扇入门类型,可以通过电子设计自动化(electronic design automatic,EDA)综合工具从门级网表中提取。然后根据扇入门类型从扇入集合 *FIN* 中选取一个扇入节点作为最优的替换节点。选取替换节点的原则是:当扇入门为与门或与非门时,选择扇入节点中值为 1 的概率最小的且未被替换过的节点作为替换节点;当扇入门为或门或或非门时,则选择扇入节点中值为 0 的概率最小的且未被替换过的节点作为替换节点。该选取原则的目的是最大化稀有节点的跳变概率。选定替换节点后,用平均权值和反向权值这两种权值进行替换,重新计算并得到稀有节点的跳变概率 $tp_{WSP_{0.5}}$ 和 $tp_{WSP_{\mathrm{invert}}}$。当 $tp_{WSP_{0.5}}$ 满足阈值且 $tp_{WSP_{\mathrm{invert}}}$ 小于阈值时,选择平均权值作为更优的替换权值;反之,则选择反向权值作为替换权值。若 $tp_{WSP_{0.5}}$ 和 $tp_{WSP_{\mathrm{invert}}}$ 都满足阈值,则需要进一步计算该稀有节点的所有扇出节点的跳变概率,并统计扇出节点中跳变概率满足阈值的总个数,选择使得总数更大的权值作为替换权值。若满足阈值的扇出节点的总数依然相同,则选择使得稀有节点跳变概率更大的权值,这是因为更大的跳变概率代表该节点被激活到稀有值状态所需时钟周期数更少。若替换完成后,该稀有节点的跳变概率依然小于阈值,则需要在它的扇入节点中,按照相同的原则继续选择最优的替换节点,直到该稀有节点的跳变概率大于阈值。

2.3 插入点选择算法

电路中节点的跳变概率由它扇入路径上的所有节点共同决定,因此为了最小化插入点个数,节点扇入路径上的所有稀有节点都应优先于该节点进行加速结构的插入,对于不存在扇入扇出关系的稀有节点,它们的插入顺序则不会相互影响。为了最小化跳变概率的计算次数,节点的跳变概率计算只需要在它扇入路径上的所有节点加速结构插入完成后计算,为此本文提出了如算法 2 所示的插入点选择算法。本文将节点的全部扇入节点的个数定义为该节点的扇入度,显然电路中输入端口的扇入度均为 0。

算法 2 的输入是插入扫描链后的门级网表以及预先设定的跳变概率阈值,输出为插入加速木马检测结构后的网表 *UpdateNetlist*。首先从门级网表中提取电路中各个节点 *NET* 的扇入门类型集合 G_{NET}、扇入节点集合 FIN_{NET} 和扇出节点集合 $FOUT_{NET}$。通

过统计每个节点对应的扇入节点集合中节点个数可以得到该节点的扇入度 D_{NET}，然后将扇入度为 0 的节点放入集合 A 中，剩余节点都放入集合 B 中。循环开始后，每次从集合 A 中取出一个节点，根据该节点扇入门类型计算其跳变概率，并判断是否大于阈值。若该节点为稀有节点，则根据算法 1 选择替换节点并判断采用哪种权值进行替换，在替换节点处插入对应的加速木马检测结构，然后将该替换节点标记为已替换。将该节点 i 的所有扇出节点的扇入度减 1，更新集合 A 和 B，将扇入度为 0 的扇出节点从集合 B 移出，并放入集合 A 中，重复此循环，直到集合 B 为空，最终返回加速木马检测结构全部插入完成后的网表。

算法 2　插入点选择算法

输入：网表 *Netlist*，跳变概率阈值 TP_{th}

输出：插入加速木马检测结构后的网表 *UpdateNetlist*

//初始化：

1. 获取电路中各节点 NET 的扇入节点集合 FIN_{NET}，扇出节点集合 $FOUT_{NET}$ 及扇入门类型集合 G_{NET}
2. 统计各节点扇入节点集合 FIN_{NET} 中节点个数，得到各节点的扇入度 D_{NET}

//插入加速木马检测结构：

3. 将所有扇入度 D_{NET} 为 0 的节点放入集合 A 中，并将剩下的节点全部放入集合 B 中

　While（集合 B 非空）do

　　　从集合 A 中取出一个节点 i，$i \in A$，并根据该节点扇入门类型计算其跳变概率 tp_{net}

　　While（tp_{net} 小于阈值）do

　　　　根据算法 1 选取替换节点 j 及替换权值，$j \in FIN_{NETi}$，将该替换节点 j 标记为已替换，根据替换权值在节点 j 处添加对应的加速木马检测结构，并重新计算节点 i 的跳变，概率 tp_{NET}

　　End while

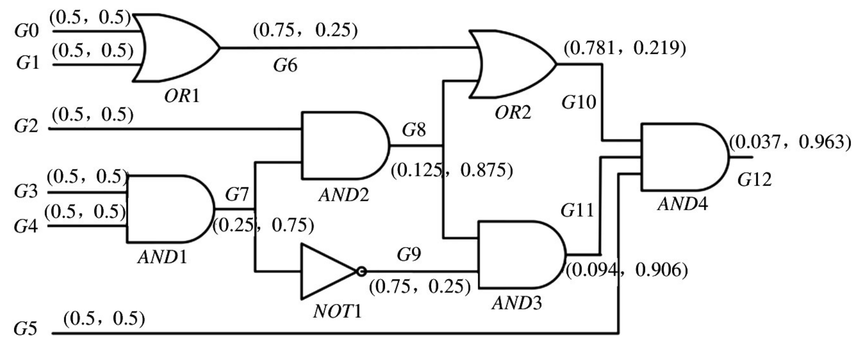
　　　　将节点 i 的扇出节点集合 $FOUT_{NETi}$ 中所有节点的扇入度减 1，移出集合 B 中

　　　　所有扇入度为 0 的节点，并放入集合 A 中

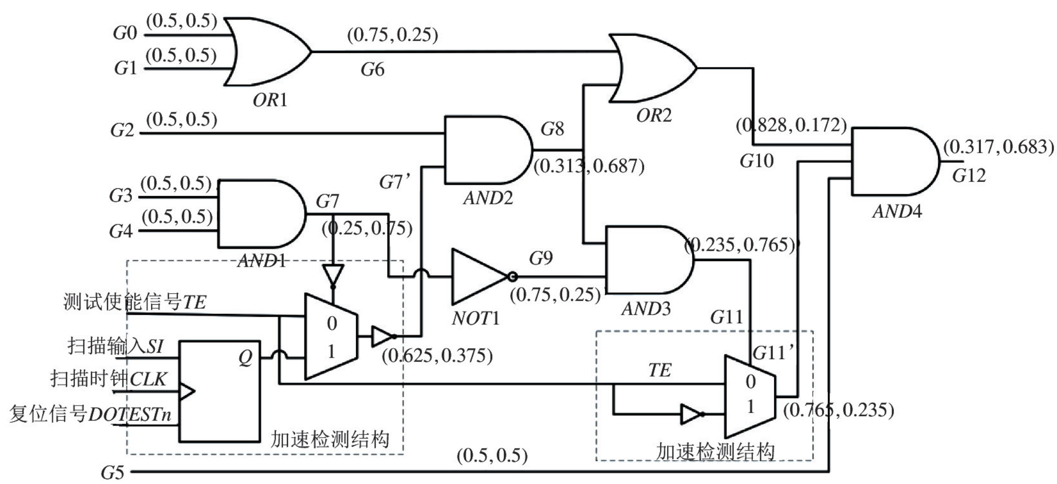
End while

4. 返回插入加速木马检测结构后的网表 *UpdateNetlist*

以图 5 为例介绍完整的权值选择和插入点选择过程。图 5（a）中所示为初始电路，设定电路输入端口值为 0 和 1 的概率均为 0.5，电路中各节点为 1 和 0 的概率如图中所示，设定跳变概率阈值为 0.14。为了更好地理解权值选择和插入点选择过程，将图 5（a）中初始电路简化为图 6（a）中所示的拓扑结构，带有箭头的线所示为稀有节点。根据插入点选择算法，首先将电路中所有扇入度为 0 的节点放入集合 A 中，即 $A = \{G0, G1, G2, G3, G4, G5\}$，如图 6（a）中虚线所示节点，剩余节点放入集合 B 中。依次从集合 A 中取出节点 $G5$、$G4$、$G3$，并更新集合 B 中对应扇出节点的扇入度。当 $G3$ 从集合 A 中取出后，B 中节点 $G7$ 的扇入度变为 0，此时更新集合 $A = \{G0, G1, G2, G7\}$，如图 6（b）所示。继续从 A 中取出节点 $G7$，由扇入门类型计算可知，$G7$ 的跳变概率大于阈值。更新 $G8$ 和 $G9$ 的扇入度，并将扇入度为 0 的 $G9$ 放入集合 A 中，如图 6（c）所示，随后取出 $G9$ 并计算其跳变概率，$G9$ 的跳变概率同样大于阈值。更新节点 $G11$ 的扇入度，此时集合 B 中没有节点满足扇入度为 0，集合 A 和 B 保持不变。继续从集合 A 中取出节点 $G2$，更新节点 $G8$ 扇入度，随后将 $G8$ 放入集合 A 中，如图 6（d）所示。接着取出 $G8$，并计算可知其跳变概率小于阈值，根据权值选择算法选取值为 1 概率更小的节点 $G7$ 为替换节点，计算两种权值替换下 $G8$ 的跳变概率。计算可知，两种权值替换下 $G8$ 的跳变概率都大于阈值，则需要进一步计算 $G8$ 所有扇出节点的跳变概率，即 $G10$ 和 $G11$。计算可知，当采用反向权值替换后，即 $G7$ 的概率值修改为（0.75，0.25），此时 $G10$ 的跳变概率小于阈值，而采用平均权值替换后，$G10$ 和 $G11$ 的跳变概率均大于阈值，因此选择平均权值，并在 $G7$ 处插入如图 5（b）左下角中虚线框内的加速木马检测结构，此时 $G11$ 变为如图 6（e）中所示的虚线。随后依次从集合 A 中取出节点 $G11$、$G2$、$G1$、$G6$、$G10$、$G12$，每取出一个节点后，判断其跳变概率是否大于阈值，并将更新后集合 B 中扇入度为 0 的节点放入集合 A 中，分别如图 6（e）、6（f）、6（g）、6（h）所示。计算可知 $G12$ 的跳变概率小于阈值，同样根据权值选择算法将 $G11$ 作为替换节点，带入两种替换权值计算后可知，$G12$ 的跳变概率均满足阈值，但反向权值使得 $G12$ 的跳变概率更大，因此选择反向权值为替换权值，并插入对应的加速结构。需要特别强调的是，由于节点

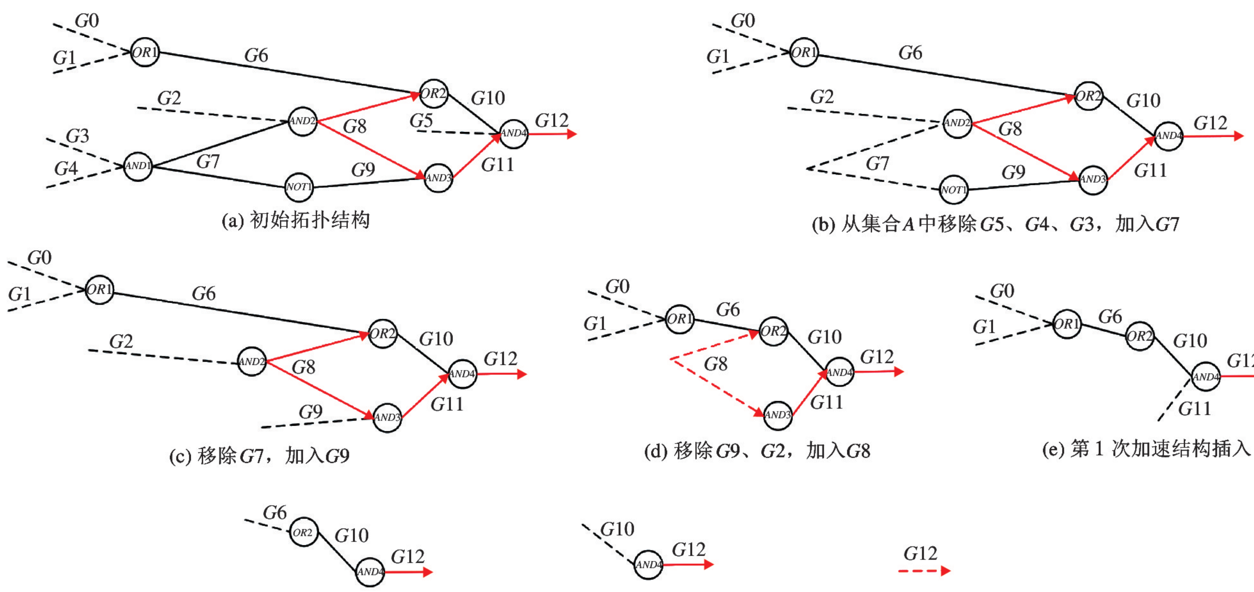


（a）初始电路



（b）插入加速木马检测结构后的电路

图 5　示例电路



(a) 初始拓扑结构

(b) 从集合 A 中移除 G5、G4、G3，加入 G7

(c) 移除 G7，加入 G9

(d) 移除 G9、G2，加入 G8

(e) 第 1 次加速结构插入

(f) 移除 G11、G1、G0，加入 G6

(g) 移除 G6，加入 G10

(h) 第 2 次加速结构插入

图 6　拓扑结构图

G9 先于 G8 从集合 A 中取出，为了不影响 G9 及其扇出路径上节点的跳变概率，加速结构只会添加在门 AND1 和 AND2 之间。最终当集合 B 为空时，插入点选择过程结束，此时电路中全部节点的跳变概

率满足阈值要求,插入加速结构后的电路如图 5(b)所示。

3　加速硬件木马检测流程

综上所示,本文提出的可调节跳变概率的加速硬件木马检测流程如图 7 所示。

首先,分析和统计电路中各个节点的扇入扇出节点集合及扇出门类型。其次,根据扇入扇出结构和权值选择规则,查找稀有节点并判断需插入的加速木马检测结构类型,并根据插入点选择规则,动态规划稀有节点的优化顺序。最后,根据稀有节点及对应的加速结构类型,在替换节点处插入对应的加速木马检测结构直至满足程序设定的结束条件。加速木马检测结构插入完成后,需将新增的扫描触发器全部放到扫描链上,保证功能模式下电路逻辑不变,测试模式下实现加速木马检测。

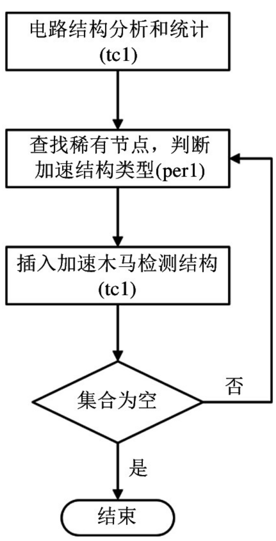


图 7　加速硬件木马检测流程图

4　实验结果及分析

为了评估本文所提可调节跳变概率的加速木马检测方法对降低木马激活时间及面积开销的效果,本实验选取 ISCSA'89 基准电路作为评估对象,并使用 DC 工具基于 STM 65 nm 工艺进行综合、可测试性设计和加速木马检测结构插入,生成最终的网表。实验通过 perl 语言搭建算法执行环境,并使用 VCS 进行随机向量仿真验证。上述的电子 EDA 工具都在 Linux 环境下运行,服务器的 CPU 为 Intel Xeon Platinum 8176,主频为 2.1 GHz,内存容量为 64 GB。在仿真测试过程中,电路所有输入端口值为 0 和 1 的概率均设定为 0.5。

本文从跳变概率、木马激活周期数及面积开销 3 个方面进行分析和评估,实验结果如下。

4.1　跳变概率

考虑到实验的适用性和完备性,本文分别在 4 种阈值条件下,选择 ISCSA'89 基准电路中的 5 个基准电路进行实验。同初始电路以及文献[13]中经典方法相比,稀有节点的平均跳变概率统计结果如表 2 所示。

由表中结果可知,根据本文方法插入加速木马检测结构后,5 种基准电路中稀有节点的平均跳变概率均显著提高,且远大于阈值。当阈值设定为 1.0E-1时,同初始电路相比,稀有节点的平均跳变概率提高了约 4 ~6 倍;当阈值设定为 1.0E-4 时,稀有节点的平均跳变概率则提高了约几百到几千倍。

表 2 的最后 1 列是本文方法同文献[13]中方法在 4 种阈值情况下的实验对比结果。从结果来看,5 种基准电路在采用本文方法后稀有节点的平均跳变概率都在文献[13]中方法的基础上有了明显提高。其中,在电路 s5378 和 s38417 上的优化效果最为明显,尤其是 s38417 中稀有节点的平均跳变概率提高了约 50% 。

4.2　木马激活验证

为了直接验证本文方法能否有效地加速硬件木马检测,实验选择对初始电路 s5378 和已插入加速木马检测结构的电路分别植入相同的硬件木马电路,选择 10 000 条随机向量进行仿真,统计并比较木马激活次数。考虑到硬件木马规模的多样性,实验采用文献[13]中设计的 2 种硬件木马电路,分别如图 8(a)和 8(b)所示。图中虚线框内是木马激活模块,当木马输入端口分别满足 1101 和 110 101 时,硬件木马电路被激活。实验设定跳变概率阈值为 1.0E-1,并从电路 s5378 的稀有节点中随机选择节点作为 2 种硬件木马的输入端口,并分别随机生成 1000 组木马电路。

表 2　稀有节点平均跳变概率统计

电路	跳变概率阈值	稀有节点的平均跳变概率			平均跳变概率优化	
		初始电路	文献[13]方法	本文方法	同初始电路相比/倍数	同文献[13]方法相比/%
s5378	1.0E-1	0.0358	0.1586	0.1898	4.3X	19.7
	1.0E-2	0.0023	0.0619	0.1114	47.4X	80.3
	1.0E-3	0.0003	0.0106	0.0153	50X	44.3
	1.0E-4	4.884E-5	0.0786	0.0946	1935.9X	20.4
	平均					41.2
s9234	1.0E-1	0.0228	0.1688	0.1805	6.9X	6.9
	1.0E-2	0.0022	0.1126	0.1156	51.5X	2.7
	1.0E-3	0.0003	0.0734	0.0829	275.3X	13.0
	1.0E-4	5.293E-5	0.1216	0.1864	3520.6X	53.3
	平均					19.0
s13207	1.0E-1	0.0179	0.1575	0.1769	8.9X	12.3
	1.0E-2	0.0009	0.1276	0.0732	80.3X	-42.6
	1.0E-3	0.0003	0.0396	0.0638	211.7X	61.1
	1.0E-4	4.372E-5	0.007	0.0129	294.1X	84.3
	平均					28.8
s15850	1.0E-1	0.0272	0.1751	0.1866	5.9X	7.3
	1.0E-2	0.0021	0.1263	0.0972	45.3X	-23.0
	1.0E-3	0.0005	0.1194	0.1565	312X	31.1
	1.0E-4	1.441E-5	0.0078	0.0138	956.7X	76.9
	平均					23.1
s38417	1.0E-1	0.0332	0.1618	0.1919	4.8X	18.6
	1.0E-2	0.0017	0.0625	0.1127	65.3X	80.3
	1.0E-3	0.0003	0.0413	0.0712	236.3X	72.4
	1.0E-4	2.801E-5	0.0323	0.0404	1441.3X	25.1
	平均					49.1

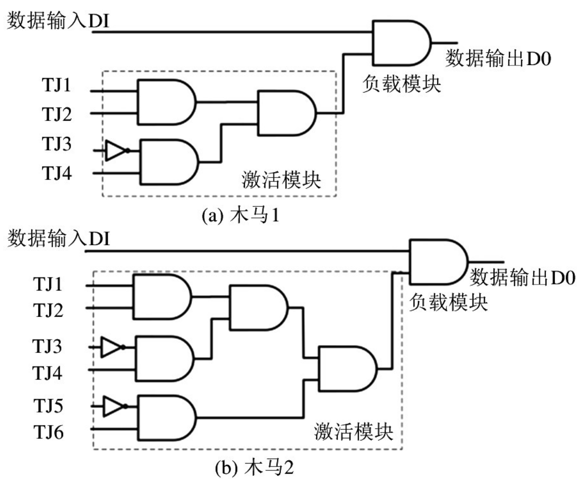


图 8　两种木马电路

随机向量仿真完成后,木马电路的平均激活次数统计如表 3 所示。数据表明,对于未插入任何加速木马检测结构的初始电路,木马电路 HT1 和 HT2 几乎无法被激活。采用本文方法和文献[13]的方法后,两种木马电路都能被成功激活。相比于文献[13]方法,本文方法将木马平均激活次数分别提高至 86.3 和 10.8 次,这说明木马被激活所需的时间周期数更短,木马检测速度的提升效果明显。

表 3　木马平均激活次数对比

木马电路	木马平均激活次数		
	初始电路	文献[13]方法	本文方法
HT1	0.2	25.7	86.3
HT2	0	1.8	10.8

4.3　时间及面积开销评估

4.3.1　时间开销分析

本文方法同经典加速木马检测方法相比,时间复杂度对比如表 4 所示。其中 n 表示电路中节点的

个数,w 表示文献[15]中权值的数量,r 和 d 分别表示文献[16]中稀有节点的个数和电路逻辑深度,g 表示电路中门的个数。

表4 几种经典方法的时间复杂度对比

方法	插入结构	时间复杂度
文献[12]	DSFF + AND/OR	$O(n^2)$
文献[13]	DSFF + MUX	$O(n^2)$
文献[15]	DSFF + MUX	$O(n^2w)$
文献[16]	AND/OR	$O(rd)$
本文方法	MUX + DSFF/INV	$O(n+g)$

同文献[12,13,15]中方法相比,本文方法的时间复杂度更低,更适合较大规模电路。

4.3.2 面积开销分析

实验分别在3种阈值条件下,对文献[13]中方法和本文方法的基准电路新增面积开销进行评估,结果如表5所示。表中第3列和第4列分别对应的是文献[13]中方法和本文方法相比初始网表的面积增加比例。从实验结果可以看出,本文方法的新增面积开销占比约为0.7% ~12.1%,面积开销很小,且面积开销占比随着阈值的减少而逐渐降低。设计者可以根据实际的面积开销要求设定适合的跳变概率阈值。表5中最后1列是本文方法同文献[13]中方法相比,新增面积开销降低的百分比。显然,3种阈值条件下,5种基准电路在采用本文方法后新增面积开销都在文献[13]方法的基础上有了明显降低。其中,在电路s5378和s13207上的优化效果最为明显,尤其是s5378中新增面积开销平均优化了68.9%。

表5 新增面积开销统计

电路	跳变概率阈值	新增面积开销占比		新增面积开销优化/%
		文献[13]方法	本文方法	
s5378	1.0E-1	0.365	0.121	66.8
	1.0E-2	0.076	0.03	60.5
	1.0E-3	0.034	0.007	79.4
		平均		68.9
s9234	1.0E-1	0.154	0.108	29.9
	1.0E-2	0.038	0.019	50.0
	1.0E-3	0.021	0.01	52.4
		平均		44.1
s13207	1.0E-1	0.18	0.11	38.9
	1.0E-2	0.094	0.023	75.5
	1.0E-3	0.037	0.016	56.7
		平均		57.0
s15850	1.0E-1	0.204	0.106	48.0
	1.0E-2	0.071	0.034	52.1
	1.0E-3	0.054	0.03	44.4
		平均		48.2
s38417	1.0E-1	0.202	0.073	63.9
	1.0E-2	0.045	0.024	46.7
	1.0E-3	0.018	0.011	38.9
		平均		49.8

5 结 论

硬件木马检测一直是业内研究的热点。本文在调研相关技术的基础上提出了可调节跳变概率的加速硬件木马检测方法。该方法从芯片的门级网表中准确提取出稀有节点的拓扑结构,分析其结构得到最优的权值替换策略,依据插入点选择规则动态规划稀有节点的修改优先级,提高稀有节点的跳变概率;并根据权值类型,优化插入的加速木马检测结构,降低硬件开销,最小化加速木马检测结构对芯片性能带来的影响,同时保证功能模式下原有电路逻辑不变。实验结果表明,采用本文所提的加速硬件木马检测方法后,电路中稀有节点的跳变概率明显提高,木马激活时间大幅降低。同经典加速木马检测方法相比,本方法在电路规模较大时效果更为明显,设计者可以根据实际需求,设定合适的跳变概率阈值,从而达到加速检测的同时降低面积开销的目的。

参考文献

[1] 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

[2] 刘华锋, 罗宏伟, 王力纬. 硬件木马综述[J]. 微电子学, 2011, 41(5):709-713

[3] LYU Y, AHMED A, MISHRA P. Automated activation of multiple targets in RTL models using concolic testing[C]//2019 Design, Automation & Test in Europe Conference and Exhibition, Florence, Italy, 2019: 354-359

[4] SALMANI H, TEHRANIPOOR M. Layout-aware switching activity localization to enhance hardware Trojan detection[J]. *IEEE Transactions on Information Forensics and Security*, 2011, 7(1): 76-87

[5] HUANG Y, BHUNIA S, MISHRA P. MERS: statistical test generation for side-channel analysis based Trojan detection[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 2016: 130-141
[6] WU T F, GANESAN K, HU Y A, et al. TPAD: hardware Trojan prevention and detection for trusted integrated circuits[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2015, 35(4): 521-534
[7] AHMED A, FARAHMANDI F, ISKANDER Y, et al. Scalable hardware Trojan activation by interleaving concrete simulation and symbolic execution[C]//2018 IEEE International Test Conference,Phoenix,USA,2018:1-10
[8] HUANG Y, BHUNIA S, MISHRA P, et al. Scalable test generation for Trojan detection using side channel analysis [J]. *IEEE Transactions on Information Forensics and Security*, 2018, 13(11): 2746-2760
[9] LYU Y, MISHRA P. Efficient test generation for Trojan detection using side channel analysis[C]//2019 Design, Automation & Test in Europe Conference & Exhibition, Florence, Italy, 2019: 408-413
[10] CHAKRABORTY R S, WOLFF F, PAUL S, et al. MERO: a statistical approach for hardware Trojan detection [C]//International Workshop on Cryptographic Hardware and Embedded Systems, Berlin, Germany, 2009: 396-410
[11] SAHA S, CHAKRABORTY R S, NUTHAKKI S S, et al. Improved test pattern generation for hardware Trojan detection using genetic algorithm and boolean satisfiability [C]//International Workshop on Cryptographic Hardware and Embedded Systems, Berlin, Germany, 2015: 577-596
[12] SALMANI H, TEHRANIPOOR M, PLUSQUELLIC J. A novel technique for improving hardware Trojan detection and reducing trojan activation time[J]. *IEEE Transactions on Very Large Scale Integration Systems*, 2011, 20(1): 112-125
[13] ZHOU B, ZHANG W, THAMBIPILLAI S, et al. A low cost acceleration method for hardware Trojan detection based on fan-out cone analysis[C]//Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis, Uttar Pradesh, India, 2014: 1-10
[14] 方凯, 王可可, 陈鑫, 等. 一种基于转换概率分析的硬件木马检测方法[J]. 微电子学, 2019, 49(2):242-248
[15] ZHOU B, ZHANG W, THAMBIPILLAI S, et al. Cost-efficient acceleration of hardware Trojan detection through fan-out cone analysis and weighted random pattern technique[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2015,35(5):792-805
[16] SHABANI A, ALIZADEH B. PMTP: A MAX-SAT based approach to detect hardware Trojan using propagation of maximum transition probability[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018: 25-33
[17] 许超,陈华军,郝守青,等.基于电路结构的测试捕获功耗优化方法[J]. 高技术通讯,2019,29(5):413-422

Hardware Trojan detection method with adjustable transition probability

LU Xinyuan[*,**,***], XU Chao[****], CHEN Huajun[****], ZHANG Longbing[*,**,***], WANG Yue[*****]
([*] State Key Laboratory of Computer Architecture,Institute of Computer Technology, Chinese Academy of Sciences, Beijing 100190)
([**] Institute of Computing Technology,Chinese Academy of Sciences, Beijing 100190)
([***] University of Chinese Academy of Sciences, Beijing 100049)
([****] Loongson Technology Corporation Limited, Beijing 100190)
([*****] Department of Public Security of Heilongjiang Province, Heilongjiang 150008)

Abstract

The hardware Trojan implementation and detection methods in the design and fabrication process of integrated circuit are studied,and an acceleration method of hardware Trojan detection with adjustable transition probability is proposed to avoid long Trojan activation time and large area overhead. According to the topology structure of circuit, the method adopts the weight replacement strategy to dynamically select the order of inserting the 2-to-1 multiplexers (2-to-1 MUX) that adjust transition probability, improves the transition probability of rare nodes in the circuit, reduces the activation time of hardware Trojan, accelerates hardware Trojan detection, and optimizes area overhead. The results of the test conducted on ISCAS'89 platform demonstrate that the average transition probability of rare nodes is increased by 19.0% to 49.1% and the area overhead is reduced by 44.1% to 68.9% compared with the existing acceleration methods of Trojan detection, and optimization effect of the proposed method is more obvious for large scale circuits.

Key words: hardware Trojan, transition probability, rare node, 2-to-1 MUX, accelerate Trojan detection structure