可信设计技术的脆弱性分析与防御

崔晓通*①② 秦蔚蓉① 程克非① 吴 渝①
①(重庆邮电大学网络空间安全与信息法学院 重庆 400065)
②(汽车噪声振动和安全技术国家重点实验室 重庆 401122)

摘 要：片上系统 (SoC) 设计人员通常使用第三方知识产权 (3PIP) 核来实现特定功能。由于这些 3PIP 核不受信任，所搭建的 SoC受到了硬件木马 (HT) 的威胁。作为可信设计技术的一个子集，多样性冗余机制在使用不可信3PIP建立可信计算方面具有较好的应用前景。然而，该文发现通过探索激活序列所设计的硬件木马能够破坏多样性冗余机制的安全性。鉴于此，该文提出一种改进的基于检查点的多样性冗余机制来防御此类攻击。

关键词：硬件木马；可信设计；检查点；序列提取
中图分类号：TP309.5; TN402 **文献标识码**：A **文章编号**：1009-5896(2021)09-2482-07
DOI: 10.11999/JEIT210624

The Vulnerability Analysis of Design-for-trust Technique and Its Defense

CUI Xiaotong①② QIN Weirong① CHENG Kefei① WU Yu①
①(*School of Cyber Security and Information Law, Chongqing University of Posts and Telecommunications, Chongqing* 400065, *China*)
②(*State Key Laboratory of Vehicle NVH and Safety Technology, Chongqing*, 401122 *China*)

Abstract: System-on-Chip (SoC) designers typically use third Party Intellectual Property(3PIP) cores to implement target functions. As these 3PIP cores are not trusted, the underlying SoC suffers from the threat of Hardware Trojans(HTs). As a subset of design-for-trust techniques, the diversified redundancy is promising in establishing trustworthy computings of SoCs. However, It is shown that the diversified redundancy can be defeated by HTs that explores triggering patterns. Therefore, an adapted diversified redundancy technique is proposed to defend against such kind of attacks.

Key words: Hardware Trojans(HTs); Design-for-trust; Checkpoints; Sequence extraction

1 引言

片上系统 (System-on-Chip, SoC) 由一系列必要组件构成，其中包括数字、模拟或混合信号知识产权 (Intellectual Property, IP) 核。由于单个公司维护SoC生命周期的高成本开销，基于IP的SoC设计流程已经成为一种全球性的趋势[1]。从图1中可以看到，基于 IP 的 SoC 生命周期涉及几个阶段：首先，SoC 集成者/设计者创建设计规范，其中定义了目标功能和性能，然后其确定一个 IP 块列表以实现预定义的规范，这些IP根据其来源可以分为两类，即内部开发的IP和从第三方IP(third Party Intellectual Property, 3PIP)供应商处购买的IP。3PIP 供应商的IP可根据它们的应用阶段进一步分为软 IP、固件 IP 和硬 IP。集成所有这些IP后，集成商将最终的图形化设计系统II (Graphic Design System II, GDSII) 布局文件发送给代工厂进行制造。

然而，上述 SoC设计开发流程近年来引发了安全问题：3PIP 供应商可能不可信，他们可能会在其 IP 中插入硬件木马。硬件木马被定义为对原始电路的恶意修改，可能导致信息泄漏、性能下降、逻辑错误和系统崩溃等结果[2]。且由于硬件木马的隐蔽性和“黄金模型”的缺失，检测 3PIP 内的硬件木马一直是一项具有挑战性的任务[3–5]。

代码覆盖率分析、结构分析、形式验证和功能分析等硅前测试阶段的检测技术通常被用于验证

收稿日期：2021-06-28；改回日期：2021-08-12；网络出版：2021-08-27
*通信作者： 崔晓通 xiaotong.sd@gmail.com
基金项目：重庆市教委科学技术项目(KJQN201900641)，计算机体系结构国家重点实验室开放课题(CARCH201902)，汽车噪声振动和安全技术国家重点实验室开放课题(NVHSKL-202114)
Foundation Items: The Science and Technology Research Program of Chongqing Municipal Education Commission (KJQN201900641), The State Key Laboratory of Computer Architecture Research Fund (CARCH201902), The State Key Laboratory of Vehicle NVH and Safety Technology Research Fund (NVHSKL-202114)

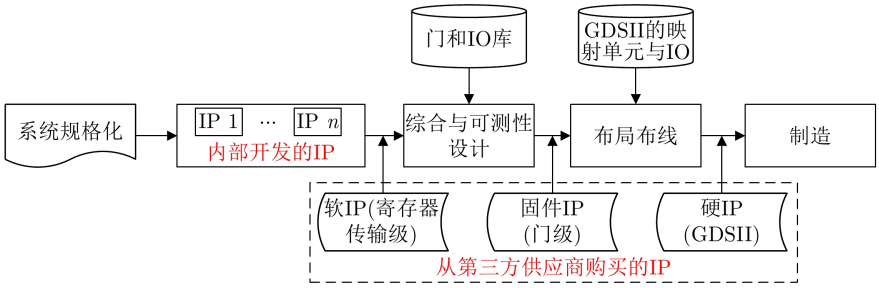


图 1 基于IP的SoC设计开发流程

3PIP核的可信度，其介绍如下。

代码覆盖率分析通常与功能验证配合使用，其中测试期间未执行的代码行被标记为可疑，并且可能是 硬件木马的一部分[6]。然而，即使代码覆盖率为100%，也不能保证检测到硬件木马，因为硬件木马可以设计为从执行时的代码行接收触发信号。

形式化验证使用携带证明代码 (Proof-Carrying Code, PCC) 来正式验证一组预定义的属性，其局限性在于证明PCC和 VHDL/Verilog 的等效过程是十分耗时的。

结构/功能分析将具有低转移概率的门或信号标记为可疑信号。这是因为硬件木马以隐蔽性作为第一设计原则，因此它们的输入通常来自不活跃的信号。然而，由于搜索空间巨大，遍历所有相关的非活跃触发组合来检测硬件木马几乎是不可能的。

可以看到，上述硅前测试阶段的检测技术在检测静默的、规模小的硬件木马非常困难。相对而言，可信设计技术在检测硬件木马方面更具有应用前景，这是因为其可以在设计阶段提前对硬件木马进行分析和规划。这些技术将在设备运行阶段发挥作用，其通常有两种目的，其一是对硬件木马进行检测，例如插入一些传感器以监控运行阶段的异常结果或影响；其二是预防硬件木马的插入，例如混淆设计使攻击者难以理解芯片的功能，从而削弱其攻击的能力。

作为可信设计技术的一个子集，运行阶段的异常监控方法可以与恢复机制配合使用，从而利用不受信任的组件实现可信计算，这类技术也被称为多样性冗余机制，其使用了来自不同的3PIP供应商的IP核，然后根据预定义的安全策略将运算子任务绑定到来自不同供应商的3PIP上。这些安全策略的目的即是检测由 硬件木马引起的恶意出错。隐藏在3PIP 中的硬件木马以单独或串通的方式进行攻击。串通攻击方式由来自同一供应商的多个3PIP包含的分布式硬件木马协同完成，从而产生预定的错误。基于多样性冗余机制的相关工作介绍如下。

Amin 等人[7]提出了一种多数投票技术，该技术使用来自多个供应商的奇数个不受信任的IP核来防止功能中断和拒绝服务(DoS)攻击。由于同时使用至少3个不同供应商的IP核，并且每个周期都需要比较IP核的输出，这种技术会产生很大的计算资源开销。

基于 IP 核的多样性和冗余计算范式，Rajendran等人[8]定义了两个从运算任务到IP核的绑定规则来检测硬件木马攻击并防止来自同一供应商的IP核之间的串通攻击。文献[9]详细说明了IP 核内的硬件木马进行攻击的方式。上述技术的缺点是它只提供对硬件木马的检测，因而无法满足关键任务型实时系统的安全需求。Cui等人[10]通过添加额外的恢复阶段规则扩展了Rajendran等人的工作，并基于整数线性规划 (Integer Linear Programming, ILP) 模型，在安全规则的前提下对IP核的数量和采购成本进行了系统优化。然而，由于ILP模型执行穷举搜索，因此在解决大规模问题时效率低下。他们进一步将硬件木马检测问题映射到了图论模型上，从而解决了效率问题，并提出在硬件木马检测之后通过被感染IP核的定位与替换来进一步强化安全方案[11]。

为了防止恶意IP核访问机密数据，Gundabolu等人[12]提出了一种基于访问控制的数据保护技术，但该技术无法检测由硬件木马引发的逻辑错误。Sayed-Ahmed等人[13]设计了一个子系统来监控一系列连续状态，一旦IP核的行为违反了安全规则，那么该IP核的片上网络连接将被断开从而将其隔离，显然，这种方法不能保证业务连续性，也不适用于关键任务型系统。

尽管上述基于多样性冗余机制的技术在实现可信计算方面很有前景，但仍然存在可以被攻击者利用的漏洞。通过对漏洞的分析，本文展示了如何从攻击者的角度通过探索硬件木马的触发序列，从而破坏多样性冗余机制的安全性，并针对该攻击场景，提出了一种基于检查点的改进方案以防御该类攻击。

本文的组织结构如下：第2节介绍了多样性冗

余机制；第3节探讨了多样性冗余机制的漏洞，并分析了如何设计硬件木马，以打破现有技术的安全策略；第4节提出了一种基于检查点的改进方案来防御该类攻击；第5节为本文总结。

2 多样性冗余机制

不受信任的3PIP核可能包含硬件木马，一旦将不受信任的供应商的 IP 核集成到SoC中，这些IP核中的硬件木马(如果存在)可能会在芯片运行期间损害系统的功能，SoC设计人员的目标是使用这些不受信任的组件来构建一个值得信赖的计算系统。然而，由于硬件木马的内在机制与环境引发的硬件的临时/永久故障大不相同，因此硬件木马所造成的逻辑错误无法通过传统故障模型进行建模[14]。

硬件的临时故障，例如由空中辐射粒子造成的单粒子翻转效应，受影响的硬件单元在很短的时间之后依然可以被正常使用[15]。因此，硬件的临时故障可以基于时间冗余机制，通过在同一硬件单元上重新执行任务，或基于空间冗余机制，使用另一个硬件单元并行执行任务，来达到消除临时故障的目的。而硬件的永久性故障，例如固定0(stuck-at-0)的单粒子锁定效应，则不会消失。因此，受影响的单元被认为因故障而不能正常使用[16]，硬件的永久性故障只能基于空间冗余机制，使用另一个硬件单元并行执行任务才能恢复其引发的错误。

由此可见，可以通过时间/空间冗余机制，来恢复硬件单元的临时/永久故障引发的逻辑错误。相对而言，由硬件木马引发的逻辑错误在其触发条件保持为真时将一直存在，而当其触发条件失效时，错误将消失。这意味着如果硬件木马的触发条件成立，则在具有时间冗余的同一硬件单元上重新执行任务将无效；而基于空间冗余使用另一个硬件单元并行执行任务也可能没有效果，因为来自同一供应商的硬件单元将包含同样的硬件木马，它们将在同样的条件下被激活。

针对上述问题，研究者提出了多样性冗余防御机制以使得系统能够正常、安全地运行。该机制利用来自不同供应商的IP核，虽然并没有确定不含硬件木马的“黄金”IP核用于对比，但可以通过比较来自不同供应商的具有相同功能的两个IP核的输出以发现异常行为。多样性冗余机制发挥作用的基本前提是“来自不同供应商的IP核往往实现方式不同，它们同时都包含硬件木马的可能性非常低，即使都包含硬件木马，但由于其实现方式不同，这些硬件木马以相同的触发条件被激活的可能性也几乎为零”。

一个完整的多样性冗余方案通常由检测阶段和恢复阶段组成。其中，检测阶段包含正常计算和重计算，正常计算被定义为一系列运算任务的连续执行，最终将实现目标应用的功能。例如，要得到$f=a_1x_1+a_2x_2$的最终结果，需要进行一系列的计算；重计算被定义为在相同或不同的硬件单元上重新执行正常计算的子任务。对于关键任务型的应用，正常计算和重计算是容错方案的必要组成部分。在硬件木马检测场景中，根据安全策略对正常计算和重计算进行调度和映射，然后将正常计算和重计算的结果进行比较，以识别是否存在硬件木马。如果结果不一致，系统将切换到恢复阶段执行恢复计算。

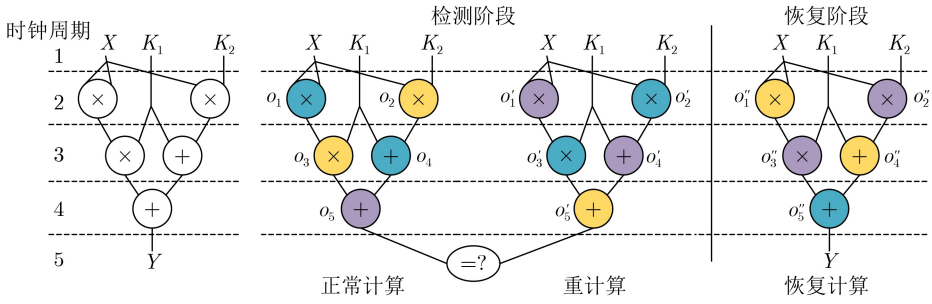
以图2(a)中的5个任务节点的应用为例，多样性冗余机制的安全策略可以被总结为以下两条运算任务到IP核的绑定规则：(1)正常计算、重计算和恢复计算中对应的任务需要绑定到来自不同供应商的IP核执行，如运算节点o_1, o_1', o_1''被绑定到不同供应商的IP核上；(2)具有父子节点关系、相关输入关系以及拥有相同子节点的运算任务需要被绑定到来自不同供应商的IP核上执行，如o_1和o_3, o_1和o_2, o_3和o_4所绑定的IP核分别对应上述关系。安全策略部署后的结果如图2(b)所示，其中不同颜色的节点表示其被映射到了不同供应商的IP核上进行执行。

接下来本文还以图2为例，展示上述安全策略的工作原理。图2(a)表示的函数是$f=K_1X^2+K_2X+K_1$。假设$K_1=2$, $K_2=3$, $X=5$，并且所有IP核都不含硬件木马，那么正常计算和重计算的输出都是62。在这种情况下，恢复计算不会被执行。但是，如果o_3所在的IP核包含硬件木马，且在运行阶段输入为(25,2)时被激活，并且它将o_3的输出更改为30。那么，正常计算的输出将与重计算的输出不一致，从而执行恢复计算。所有中间结果如表1所示，其中粗体表示的值是由于硬件木马被激活后产生的错误输出。

由于o_2,o_1''和o_3的IP核来自同一供应商且类型相同，因此它们可能包含相同类型硬件木马，但o_2和o_1''的IP 核中的硬件木马由于输入条件不同而不会被激活。另一方面，由于恢复计算中没有任何硬件木马被激活，因此其输出结果是正确的。

3 多样性冗余机制的脆弱性探索

基于多样性冗余机制工作的基本前提，本文发现其安全规则具有一个明显的限制，即在所有的计算中(正常计算、重计算和恢复计算)，每次只能容忍一个激活的硬件木马。例如，如果o_3和o_3'的IP核包含相同的硬件木马，并且它们由相同的输入序列激活，那么正常计算和重计算的输出将是相同的错



(a) 5 节点的应用　　(b) 安全策略部署后的结果

图 2 多样性冗余机制中的安全策略部署

表 1　当1个硬件木马被激活时计算中的中间结果展示

时钟周期	正常计算					重计算					恢复计算				
	o_1	o_2	o_3	o_4	o_5	o_1'	o_2'	o_3'	o_4'	o_5'	o_1''	o_2''	o_3''	o_4''	o_5''
1(6)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	25	15	N/A	N/A	N/A
2(7)	25	15	N/A	N/A	N/A	25	15	N/A	N/A	N/A	25	15	50	17	N/A
3(8)	25	15	**30**	17	N/A	25	15	50	17	N/A	25	15	50	17	67
4	25	15	**30**	17	**47**	25	15	50	17	67					
5	$o_5 \neq o_5'$时进入恢复计算														

误结果，因此被现有安全机制视为正确结果，这是违反安全策略的一种情况，本文将其称为violation 1，即将错误的结果当成正确的结果。另外一个例子是，如果o_3和o_3'的IP核包含不同的硬件木马并且它们都被激活，那么大多数情况下正常计算和重计算的输出将是不同的错误结果，那么恢复计算的输出也将与它们的输出不同，从而违反了冗余原则，即由于3个输出不同，本文无法确定哪个输出是正确的，这是违反安全策略的另一种情况，将其称为violation 2，即无法从3个不同的结果中分辨出正确的结果。因此，可以用$\sum A_i \leq 1$来描述安全策略仍然有效的情况，其中$i=\in\{1,2,3\}$，A_i表示在第i个运算中是否存在被激活的硬件木马，$A_i=1$表示存在，否则为$A_i=0$。

作为攻击者，他可以用$\sum A_i>1$来打破多样性冗余机制的安全性，即在所有计算中分布式布置多个木马并将它们激活。从基本前提可知，不同供应商的IP很难串通，因此，这里假定潜在的攻击者仅具有控制来自一个供应商的 IP核的能力。下面，将分析攻击者如何通过使$\sum A_i>1$来破坏安全策略，其分为两个部分，第1个部分是攻击者仅通过控制检测阶段中的IP核打破安全策略，第2个部分是通过控制检测和恢复两个阶段中的IP核打破安全策略。

3.1　在检测阶段中击败安全策略

仅在检测阶段使$\sum A_i>1$意味着 $A_1=A_2=1$。以图2(b)为例，所有运算任务都分配给了来自3个供应商的IP核，分别用蓝色、紫色和黄色表示。来自一个3PIP供应商中的攻击者可以在执行乘法的IP核中插入硬件木马，例如执行o_1,o_2'和o_3'的IP核。

当其中的硬件木马被激活使得$A_1=A_2=1$时，根据正常计算和重计算的输出是否相同，可以分成2种情况：第1种情况是正常计算和重计算的输出是相同的错误结果，显然，它遵循violation 1并违背了安全策略；由于o_1,o_2'和o_3'的IP核包含相同的硬件木马并在计算中占据不同的位置，即它们接收来自不同运算的输出，并为不同的运算提供输入，因此第2种情况是最常见的，即正常计算和重计算的输出是不同的错误结果，这种情况遵循violation 2并破坏了安全策略。

实现violation 1或violation 2的关键是如何设计硬件木马以使得$A_1=A_2=1$。可以看出，原始安全策略已经排除了来自同一供应商的IP核通过相同/相关输入激活其相同硬件木马的情况。因此，攻击者需要通过不同的输入来激活相同的硬件木马，本文将这种攻击称为触发序列探索。

第1步是让隐藏在不同IP核中的相同硬件木马接收到相同的触发序列，这可以通过在硬件木马中实现序列提取结构来实现。比如，假设o_1和o'_3的输入分别是677和165，它们都有二进制形式的10100101序列，因此这个序列可以用作硬件木马的触发序列。实际上，通过选定特定的比特，序列提取结构可以提取任意类型的序列。当然，为了减少被误激活的概率，在设计硬件木马时还可以加入时序逻辑，如计数器。

3.2 在检测与恢复阶段击败安全策略

通过检测和恢复阶段的IP核联合来击败安全策略意味着对于检测阶段$A_1+A_2\geq1$，对于恢复阶段$A_3=1$。由于$A_1=A_2=1$即$A_1+A_2>1$的情况已经能够击败安全策略，且在上一节中已经考虑过，所以这里只考虑$A_1+A_2=1$和$A_3=1$的情况。更为具体地，以$A_1=1$和$A_3=1$为例，考虑在正常计算和恢复计算中同时存在相同的被激活硬件木马。

仍以图2(b)为例，某3PIP供应商的攻击者可以在其执行加法的IP核中插入硬件木马，例如o_4和$o"_5$所绑定的IP核。当其中的硬件木马被激活使得$A_1=A_3=1$时，根据正常计算和恢复计算的输出是否相同，同样有2种可能的情况：第1种是它们的输出是相同的错误结果，在这种情况下，在系统切换到恢复阶段之前，正常计算和重计算的输出已经不一致，这时由于无法在检测阶段判断哪个输出是正确的，因此需要依靠恢复计算的输出来确定正确结果。但是，由于正常计算和恢复计算的输出是相同的错误结果，现有安全机制系统会将错误结果视为正确结果，显然，它符合violation 1从而破坏了安全策略；第2种情况是正常计算和恢复计算的输出是不同的错误结果，在这种情况下，它符合violation 2从而破坏了安全策略。

为了实现$A_1+A_2\geq1$和$A_3=1$，同样可如上一节所示，通过探索触发序列设计相应的硬件木马，从而达到攻击的目的。

4 防御机制探索

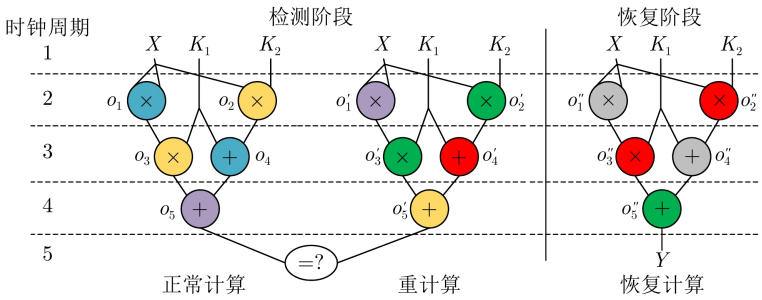
在上述攻击场景中，安全策略被分布在不同计算中的硬件木马所攻破，这些硬件木马在运行阶段被激活，且其所在的IP核来自同一供应商。那么应该如何防御此类攻击呢？

4.1 增加新的3PIP供应商？

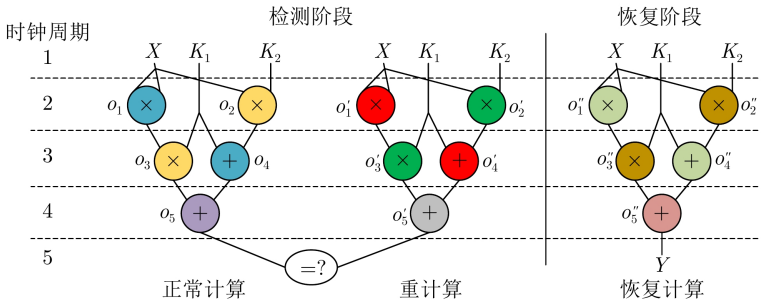
在图2(b)中，如果o_1和o'_3的IP核中的硬件木马都被激活，那么安全策略就被破坏了。由于序列提取结构的存在，无法保证它们不会被同时激活，可以想到的一种直接防御方法是使这些IP核来自不同的供应商，从而重新满足多样性冗余机制。

但是，这种加强防御的方法，代价是非常大的。假设每个供应商的乘法和加法IP核的许可费用分别为O_{mul}和O_{add}。那么建立图2(b)中多样性冗余方案将使用来自3个供应商的IP核并花费$3\times(O_{\text{mul}}+O_{\text{add}})$。相对而言，如果为了达到防御上述攻击的目的而进一步增加IP核多样性，则加强的防御方案将使用来自6个供应商的IP核并花费$6\times(O_{\text{mul}}+O_{\text{add}})$，如图3(a)所示。

此外，通过硬件木马的序列抽取结构，即使是同一供应商的不同类型的IP核也能够破坏安全策略，这将导致使用更多3PIP多样性进行防御，如图3(b)所示，其共使用了9个供应商的IP核。除了开销很大的缺陷之外，这种防御方法还会受到3PIP多样性的限制，即无法找到足够的IP核供应商来实现该方案。显然，在SoC中部署这样一种安全强化的、极大开销的安全策略是不切实际的。



(a) 防御来自同一厂商同一种类型IP核的加强方案



(b) 防御来自同一厂商所有类型IP核的加强方案

图 3 通过IP核多样性增强防御方案

4.2 插入检查点？

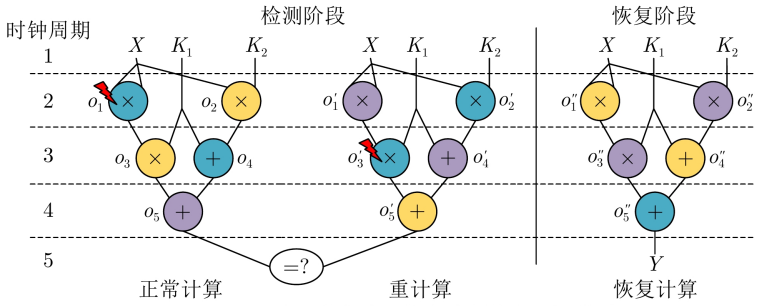
再次回顾攻击场景：来自同一供应商含有硬件木马的IP核分布在不同计算中并被激活。不失一般性，假设o_1和o_3'的IP核所含的硬件木马在正常计算和重计算中被激活，如图4(a)所示。那么正常计算和重新计算的输出都是错误的结果，可能符合violation 1或violation 2。但是，在事先不知道硬件木马位置的情况下，即使在恢复计算之后也无法识别或确定哪个计算的输出是正确的。

但是当查看具体的计算过程时，就会注意到上述攻击方法的局限性。在图4(b)中，用“W(Wrong)”和“R(Right)”来标记中间结果是否正确，那么可以通过与其他两个计算的相应节点的中间结果进行比较来识别o_1的输出是错误的。这样，就可以逐步消除所有包含错误结果的数据路径，从而建立完整、正确的计算过程。

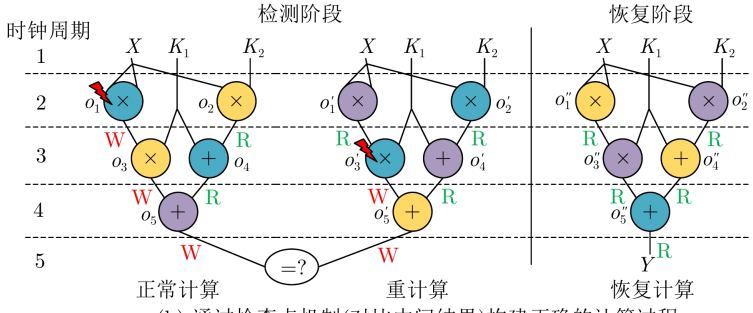
基于检查点的安全机制和原有安全机制的不同之处在于：(1)为了消除错误的中间结果，构建正确的计算过程，有必要将正常计算、重计算和恢复计算的中间结果进行比较，由于它们是在不同阶段获得的，因此需要对检测阶段的中间结果进行存储；(2)在原有的多样性冗余方案中，当正常计算和重新计算的输出不一致时，系统会切换到恢复计算，但是在新的攻击场景下，正常计算和重计算的输出可能是相同的错误结果，从而阻止系统切换到恢复计算，因此，需要将原始安全策略进行调整以适应新的攻击场景，即在将正常计算和重计算的对应运算节点的中间结果存储之前进行比较，一旦两者不一致，无论正常计算和重计算的最终输出是否一致，系统都会切换到恢复计算。

当系统运行时，基于图4(b)中的检查点防御机制中，检测阶段和恢复阶段的调度机制如表2所示。

可以看到，基于检查点的安全机制仍然具有一定的局限性：(1)其继承了原有安全机制的多样性冗余策略，因此可能会受到IP核多样性的限制，这也是多样性冗余安全策略的固有缺陷；(2)检查点机制的引入涉及中间结果的存储与比较，需要额外的控制结构和比较计算开销，可能会使片上系统上



(a) o_1和o_3所在的IP核中的硬件木马被激活



(b) 通过检查点机制(对比中间结果)构建正确的计算过程

图 4 通过检查点机制增强防御方案

表 2　基于检查点的多样性冗余机制算法

算法1 基于检查点的多样性冗余机制
设置恢复计算切换标志位flag=0
(1)在检测阶段，对于每对正常计算、重计算中对应的任务节点，比较并存储其运算结果；
(2)如果存在运算结果不一致的情况，则将flag置为1
(3)当检测阶段结束时，如果flag=0，则正常计算的最终输出为正确输出，否则进入恢复阶段。
(4)对于恢复计算中的每个任务节点，将其结果与检测阶段存储的结果进行比较，根据多数投票原则构建正确计算，并获取最终计算结果。

的资源使用更为紧张，对相应开销的进一步分析如下。

在基于检查点的防御机制中，涉及到比较等运算，需要额外的运算单元，主要包括用于比较的比较器和在比较器在不同时钟周期复用时的多路复用器。在检测阶段，假设在一个时钟周期共有N_c对任务节点的结果需要被比较，那么共需要N_c个比较器；而在恢复阶段，为了确定正确的运算节点，一个任务节点的输出需要同时与正常计算和重计算任务节点的存储结果进行比较，如果假设一个时钟周期有R_c个任务节点，那么需要$2R_c$个比较器。考虑到比较器的复用，需要对N_c和$2R_c$的值进行比较，以最终确定比较器的个数。另一方面，如果一个比较器共被复用λ次，那么其至少1个输入需要连接1个λ-输入的多路复用器。以上比较器和多路复用器需要结合调度方案、片上系统资源等条件确定与优化。

5 结论

可信设计技术在检测或防止硬件木马等方面很有前景。作为为可信设计技术的一个子集，多样性冗余机制的目标是使用不受信任的组件构建可信计算系统。然而，本文发现多样性冗余机制存在硬件木马的触发序列探索漏洞。本文探索了上述攻击的防御方法，并在现有的多样性冗余方案中加入了检查点机制，从而能够对此类攻击进行防御。

参 考 文 献

[1] BHUNIA S and TEHRANIPOOR M. Hardware Security: A Hands-on Learning Approach[M]. Cambridge: Morgan Kaufmann Publishers, 2018.

[2] XIAO K, FORTE D, JIN Y, *et al*. Hardware Trojans: Lessons learned after one decade of research[J]. *ACM Transactions on Design Automation of Electronic Systems*, 2016, 22(1): 6.

[3] ZHANG Jiliang and QU Gang. Recent attacks and defenses on FPGA-based systems[J]. *ACM Transactions on Reconfigurable Technology and Systems*, 2019, 12(3): 14.

[4] LU Renjie, SHEN Haihua, FENG Zhihua, *et al*. HTDet: A clustering method using information entropy for hardware Trojan detection[J]. *Tsinghua Science and Technology*, 2021, 26(1): 48–61. doi: 10.26599/TST.2019.9010047.

[5] HU Nianhang, YE Mengmei, and WEI Sheng. Surviving information leakage hardware Trojan attacks using hardware isolation[J]. *IEEE Transactions on Emerging Topics in Computing*, 2019, 7(2): 253–261. doi: 10.1109/TETC.2017.2648739.

[6] ZHANG Xuehui and TEHRANIPOOR M. Case study: Detecting hardware Trojans in third-party digital IP cores[C]. 2011 IEEE International Symposium on Hardware-Oriented Security and Trust, San Diego, USA, 2011: 67–70.

[7] AMIN H A M, ALKABANI Y, and SELIM G M I. System-level protection and hardware Trojan detection using weighted voting[J]. *Journal of Advanced Research*, 2014, 5(4): 499–505. doi: 10.1016/j.jare.2013.11.008.

[8] RAJENDRAN J, ZHANG Huan, SINANOGLU O, *et al*. High-level synthesis for security and trust[C]. The 19th International on-Line Testing Symposium, Chania, Greece, 2013: 232–233.

[9] RAJENDRAN J J V, SINANOGLU O, and KARRI R. Building trustworthy systems using untrusted components: A high-level synthesis approach[J]. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2016, 24(9): 2946–2959. doi: 10.1109/TVLSI.2016.2530092.

[10] CUI Xiaotong, MA Kun, SHI Liang, *et al*. High-level synthesis for run-time hardware Trojan detection and recovery[C]. The 51st ACM/EDAC/IEEE Design Automation Conference, San Francisco, USA, 2014: 1–6.

[11] CUI Xiaotong, ZHANG Xing, YAN Hao, *et al*. Towards building and optimizing trustworthy systems using untrusted components: A graph-theoretic perspective[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, To be published. doi: 10.1109/TCAD.2021.3086765.

[12] GUNDABOLU S and WANG Xiaofang. On-chip data security against untrustworthy software and hardware IPs in embedded systems[C]. 2018 IEEE Computer Society Annual Symposium on VLSI, Hong Kong, China, 2018: 644–649.

[13] SAYED-AHMED A, HAJ-YAHYA J, and CHATTOPADHYAY A. SoCINT: Resilient system-on-chip via dynamic intrusion detection[C]. The 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems, Delhi, India, 2019: 359–364.

[14] CUI Xiaotong, SAEED S M, ZULEHNER A, *et al*. On the difficulty of inserting Trojans in reversible computing architectures[J]. *IEEE Transactions on Emerging Topics in Computing*, 2020, 8(4): 960–972.

[15] KARNIK T and HAZUCHA P. Characterization of soft errors caused by single event upsets in CMOS processes[J]. *IEEE Transactions on Dependable and secure Computing*, 2004, 1(2): 128–143. doi: 10.1109/TDSC.2004.14.

[16] GAILLARD R. Single event effects: Mechanisms and classification[M]. NICOLAIDIS M. Soft Errors in Modern Electronic Systems. Boston: Springer, 2011: 27–54.

崔晓通：男，1991年生，博士，讲师，研究方向为硬件安全、容错计算.

秦蔚蓉：女，1997年生，硕士生，研究方向为硬件安全、无线电测试.

程克非：男，1974年生，博士，教授，研究方向为网络安全、嵌入式系统.

吴 渝：女，1970年生，博士，教授，研究方向为网络舆情与管控.

责任编辑：余 蓉