持续性攻击背景下的片上系统再生

（正在进行的工作）Ahmad T Sheikh、Ali Shoker 和 Paulo Esteves-Ver´ ıssimo弹性计算与网络安全中心（*RC3*）、计算机、电气与数学科学及工程学部（*CEMSE*）、阿卜杜拉国王科技大学（*KAUST*），图瓦勒 *23955-6900*，沙特阿拉伯王国。

arXiv:2301.08018v1 [cs.CR] 19 Jan 2023

摘要—为应对日益增多且动态变化、适应性不断增强的持续性攻击威胁，研究人员正在硬件层面研究故障与入侵容错（FIT），以提高关键系统的弹性。基于状态机复制，当副本遭到入侵并且彼此独立地发生故障时，FIT 被认为是有效的。这需要在软件和硬件层面采用不同的多样化方式。本文介绍首个基于硬件的再生框架，并将其命名为*Samsara*。该框架能够创建架构各异的新计算核心（FIT 副本运行于这些核心之上）。借助配备现场可编程门阵列（FPGA）的多处理器片上系统（MPSoC）的可编程和可重构特性，这一目标得以实现。针对某个核心进行分析并利用其漏洞的持续性攻击，将无法得逞，因为系统能够足够迅速地切换至不同核心架构，实现再生。我们探讨了该设计的可行性，并将实证评估留待未来工作。

索引词— 再生，多处理器片上系统，可重构计算，故障与入侵容错（FIT），拜占庭一致性 nt

1 引言

THERE 人们日益依赖硬件加速器（包括 GPU、专用集成电路、数据处理器、通用图形处理器和现场可编程门阵列）来提升关键数字计算嵌入式应用程序的性能与安全性，例如物联网服务、信息物理系统和自动化领域中的应用。近来，人们对通过在云端 FPGA 实例中部署数百个核来加速云应用程序也越来越感兴趣 [1]。性能提升归功于消除应用程序所依赖的软件栈并采用多核处理（例如图像处理和人工智能应用中的处理方式）；同时，人们认为，由于可编程硬件面对软件攻击时具有不可变性，其安全性几乎无懈可击。这推动了两方面的发展：一是具备隔离和加密功能的专用硬件安全元件，例如安全飞地、硬件安全模块、保险库等 [2]–[5]，；二是基于更小、因而更易于验证的安全抽象来强化现代复杂嵌入式系统 [6], [7]。遗憾的是，随着通用可编程硬件计算（如通用图形处理器和现场可编程门阵列）的出现，不可变性的优势已不复存在。这使得相应的系统和应用程序更容易受到入侵攻击以及硬件设计本身漏洞的影响 [8]–[10]。因此，有必要引入新技术，以提升这些可编程硬件加速器抵御入侵攻击和运行时错误的安全性与弹性。

作为硬件/软件栈中的最后一道防线，可编程和不可编程硬件加速器的安全性一直是学术界和工业界关注的焦点 [11], [12]。采用的方法主要包括：（1）隔离计算；（2）总线隔离；（3）频繁进行 DRAM 刷新和随机化 [13]，；（4）比特流加密 [14] 等。尽管这些技术通常安全可靠，但仍不足以防御：（1）难以在实践中检测的入侵攻击，因为逻辑结构的设计空间巨大，可能存在隐蔽逻辑和终止开关 [15]，；（2）由设计错误、灰尘、老化和过热引起的毛刺 [16], [17]，；以及（3）实时操作系统中的漏洞 [18]。此外，硬件验证过程极其艰巨且成本高昂，因此难以用于大规模生产 [15]。

为克服入侵检测效率低下的问题，一种近期方案是采用受分布式计算中的状态机复制（SMR） [19], [20], 启发的入侵屏蔽。为容忍最多 t 个恶意或异常副本，入侵屏蔽要求并发运行某个数量（n）的进程副本，从而构成一个复制状态机，在此情况下，这些副本运行于多个处理核心上。状态机的输出是由未受损的副本法定人数（包括软件和底层硬件核心）达成的一致结果。通过运行一种入侵一致性协议变体来达成一致，这类协议通常称为拜占庭一致性（BA） [22]。尽管这种方法无需事先定义或了解相关情况，即可抵御（非故意的）毛刺和入侵，但它有两个值得注意的局限。第一个局限是，假设副本会彼此独立地遭到破坏或发生故障，也就是说，不存在共同漏洞或毛刺

Ahmad T. Sheikh 是沙特阿拉伯 *RC3 KAUST* 的博士后研究员。（电子邮件：*ahmad.sheikh@kaust.edu.sa*）*Ali Shoker* 是沙特阿拉伯 *RC3 KAUST* 的资深研究科学家。（电子邮件：*ali.shoker@kaust.edu.sa*）*Paulo Esteves-Verissimo* 是沙特阿拉伯 *KAUST* 的计算机科学教授兼 *RC3* 主任。（电子邮件：*paulo.verissimo@kaust.edu.sa*）

但副本之间若不满足这一假设，就可能导致副本发生共因失效，从而违反法定人数不变量（因为 $n - t' < n - t$）。第二个局限是，这些系统的规模固定（n）。因此，面对可变威胁严重程度级别时，它们无法动态响应；在这种情况下，系统可能遭受超过t 次同时入侵。事实上，若持续性对手有足够长的时间，可能会导致系统资源耗尽 [23]，即攻陷超过 t 个副本。

本文介绍了首个用于故障与入侵容错（FIT）的片上系统再生框架，我们将其称为*Samsara*。*Samsara* 利用多处理器片上系统的现场可编程门阵列部分可重配置区域（PRR）特性，生成新的中央处理器核心，称为瓦片，以供 FIT 副本运行。该框架针对 FIT 协议的两大主要局限：（1）支持热插拔式扩缩容，动态增减承载副本的中央处理器核心数量 n，从而调整已受入侵副本的入侵韧性 t；（2）通过生成不同的中央处理器核心，提高故障独立性以及对高级持续性攻击的韧性。这些核心可通过多种软核模板库生成，例如由多家供应商提供的模板 [24]。

再生的概念在 FIT 中并不新鲜，已有研究建议采用这一方法 [25] 来提高基于软件的系统的弹性。然而，这一概念从未以动态方式应用于嵌入式硬件层。原因显而易见：不可编程多核硬件的架构在制造阶段就已预先定义并固定，因此无法在生产过程中使这些核心多样化。幸运的是，诸如现场可编程门阵列之类的可重配置硬件，使采用不同供应商模板实例化多样化核心成为可能。另一方面，尽管已有文献 [26], [27] 提出了控制多核系统中线程级并行的能力，但这种方法既非动态，也不实用，因为它需要重启整个系统；而 *Samsara* 则可在运行时生成并重启中央处理器核心和副本。

在本文中，我们介绍了 *Samsara* 框架的初步架构（如图 3 所示），作为赛灵思（AMD）Zynq 架构 [28]的一部分。尽管如此，所提出的架构采用通用的模块化设计，可集成到采用现场可编程门阵列的类似多处理器片上系统架构中。这种模块化设计也支持未来进一步扩展和优化，例如采用多种软核以及周期性、主动式和反应式触发策略。未来的工作包括开展实证评估，以验证其在实际应用中的行为、安全性和性能。

本文其余部分安排如下：第 2 节介绍容错与抗入侵（FIT）以及硬件多处理器片上系统加速器的背景。第 3 节描述所提出的框架，包括威胁模型、架构和工作流程。随后，我们在第 4 节开展可行性讨论，最后在第 5 节总结全文。

2 背景

本节将简要介绍多处理器片上系统（MPSoC），例如 Zynq 架构 [29], 以及故障与入侵容错（FIT），以便更顺畅地理解后续概念。

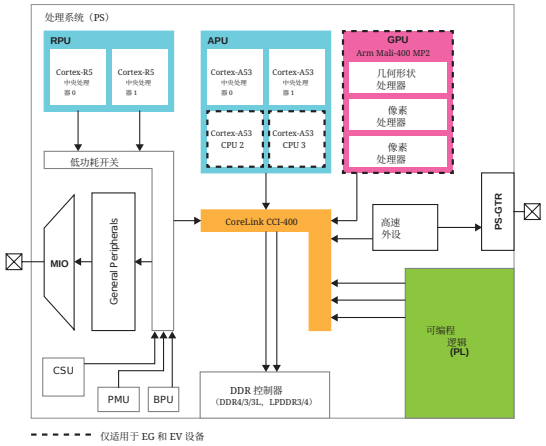

图 1：Zynq 器件的简化架构 [29], [33]

2.1 多处理器片上系统架构

多处理器片上系统（MPSoC）是一种包含多个微处理器的片上系统（SoC），常用于嵌入式设备。典型的现代 MPSoC [29] 包含多个已制造的处理核心（称为硬核）以及可编程硬件，称为现场可编程门阵列（FPGA）。后者能够在制造完成后，仍可使用任意定制硬件逻辑对底层逻辑结构进行重新配置。常见的做法是基于位文件模板（称为瓦片）生成一个仿真处理核心。对于中低产量，开发和部署成本微不足道，但单台设备的成本较高。专用集成电路（ASIC）是逻辑电路不可变的芯片。由于单台设备成本较低，大规模制造时通常会优先选择专用集成电路。FPGA 则提供了一个合适的折中方案，可在将设计送往制造工厂流片前，用于设计测试和验证。

我们通过介绍赛灵思在 2016 [29] 提出的知名 Zynq MPSoC 架构，概述多处理器片上系统。Zynq MPSoC 是 Zynq-7000 [30] 系列器件的更新版本，增加了处理单元并改进了现场可编程门阵列逻辑结构。其他现场可编程门阵列或多处理器片上系统变体 [31],[32] 与 Zynq 主体设计相似，尽管在实现上存在一些差异。

2.1.1 Zynq 架构案例研究

图1展示了 Zynq UltraScale+ MPSoC 器件的简化架构。Zynq MPSoC 器件由集成在同一芯片上的处理系统（PS）和可编程逻辑（PL）组成。PS 包含各种处理单元，负责管理片上系统的功能和安全性。PS 的一个重要组件是应用处理单元（APU），其中集成了 Arm 处理核心。这些核心可以使用 C/C++ 语言进行编程。PS 还包括图形处理单元（GPU）、实时处理单元（RPU），以及连接外部和内部外设的接口。配置安全单元（CSU）由多个模块组成；其中，处理器配置访问端口（*PCAP*）可用于由 PS 在运行时动态或静态地对 PL 进行编程。

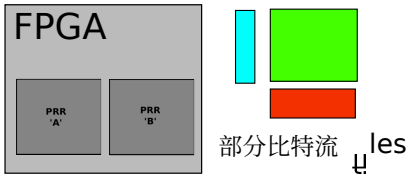

图 2：PR 基本概念

PL 由可重新配置的现场可编程门阵列逻辑结构组成，可通过编程加速任意功能。PL 可以直接配置，也可以由 PS 配置。用于配置 PL 逻辑结构的程序称为比特流或比特文件。PL 一经编程便不可变，也就是说，若不重新配置整个逻辑结构，就无法进一步更改 PL 中空闲且可重新配置的区域。PL 包含大量可配置逻辑块（CLB）、DSP 模块、输入/输出引脚、可编程互连、块随机存取存储器（BRAM）等。不过，与 PS 相比，其资源数量有限。因此，我们旨在节约利用 PL 资源，并计划根据需要通过 PS 动态地重新配置这些资源，即部分重配置（*PR*）。赛灵思将 PR 更名为动态功能交换（*DFX*） [34] 。

2.1.2 部分重配置（*PR*）
FPGA 技术支持现场编程和重配置，无需将设计发送到制造工厂进行修改。PR 更进一步，使人们能够使用部分比特流对 PL 中称为部分可重配置区域（ PRR ）的特定区域进行编程。完整比特流用于配置 FPGA 器件并使其为加速做好准备；随后可将部分比特流下载到 PL 中，仅修改 PRR，而不影响 PL 逻辑结构其余部分中运行的应用程序的完整性。图 2 展示了一个简单的 PR 示例，其中 FPGA 被划分为两个动态区域。部分位文件以不同颜色显示，借助 DFX，它们可在运行时部署到任意动态区域。PR 的最大优势之一，是能够对底层硅资源进行时间复用，以处理不同任务。这将缩减设计面积并降低功耗——这是任何数字设计中最重要的两个因素。不过，彼此正交的加速器也可以转换为用于 PR 的部分位文件。

可编程逻辑既可以通过处理系统进行编程，也可以由可编程逻辑在内部进行编程。通过处理系统对可编程逻辑进行编程时，处理器配置访问端口接口会借助 DMA控制器，从外部 DDR存储器传输完整或部分比特流。可编程逻辑也可以通过原生的内部配置访问端口 (ICAP) 进行编程。ICAP 的重新配置速度远高于 PCAP。

2.2 故障与入侵容错

故障与入侵容错（FIT） [19] 这一概念已在分布式系统领域得到充分提出和研究，旨在当系统存在任意故障（通常称为拜占庭故障 [35]）或恶意入侵时，提升系统状态的完整性。其思路是复制系统过程，使并发副本构成一个确定性状态机 [21] ，确保各方对唯一的最终状态达成一致，即采用拜占庭一致性协议 [36]。FIT 的正确性取决于一组正确（即无故障或恶意行为）节点组成的法定人数，他们能够达成一致／共识，并维护操作的全序。若要容忍数量为 t 的受损副本，系统通常需要总计 $n = 3t + 1$ 个副本 n [36], [37]。

FIT 协议的基本假设是副本能够独立失效，即副本节点之间不存在共同漏洞；然而，若无法满足这一假设，则可能导致共因失效。通过在不同抽象层级对副本进行多样化设计，可以实现故障与入侵独立性，例如，使用 N 版本编程技术 [38], [39]开发结构不同但功能等效的代码，或采用具有互不重叠的故障漏洞的不同操作系统组合 [40]。

3 SAMSARA 故障与入侵容错框架

3.1 系统与威胁模型

我们考虑一种片上系统，由一个静态处理部分（称为处理系统 (PS)）、一个可重构部分（即类似现场可编程门阵列可编程逻辑 (PL)的部分）以及一个专用集成电路(专用集成电路)芯片部分组成。所有部分都通过同一芯片上的可靠硬件总线连接。为增强安全性，*Samsara* 的主逻辑位于 ASIC 部分，因此不可变；而其他优化模块则位于 PS 中并在其中运行。不过，PL 部分是可变的，用于根据软核模板生成新的瓦片，以承载 FIT 副本。要生成的瓦片数量取决于所采用的 FIT协议 [7], [26], [36]。本文不假定使用某种特定的 FIT 协议。PL 内的瓦片通过可靠总线连接，该总线能够将消息传送至目的地。（在恶劣环境中，可以假设硬件总线不可靠，可能会丢弃或篡改交换的消息，例如受外部因素影响；不过，为简化起见，我们不考虑这种情况。）瓦片之间的通信可以是同步的，也可以是部分同步的 [41] （即在未知的时间上限内传送）。我们假设瓦片具备适当程度的遏制或隔离，因此异常或漏洞不会直接影响其他核。

此外，我们假设存在一个强大且先进的持续性对手，它能够攻击整个片上系统，目的是破坏系统状态的完整性（该状态受 FIT协议保护）。因此，攻击者可能会利用瓦片（位于 PL 中）或上述软件栈中的漏洞（通常是无需操作系统栈的小型操作）。由于我们关注的是硬件多样性，因此假设系统具备一定程度的软件多样性 [25]。不过，硬件多样性由 *Samsara* 提供，它通过不同软核模板（例如，来自不同供应商的模板）创建多样化的瓦片。然而，我们假设在参考时间 T_a内，被攻陷的核不超过t 个。在此时间范围内，预期 *Samsara* 会更新核。我们假设 PS 中的 *Samsara* 模块经过加密并以安全方式运行，例如在 TEE 飞地中运行

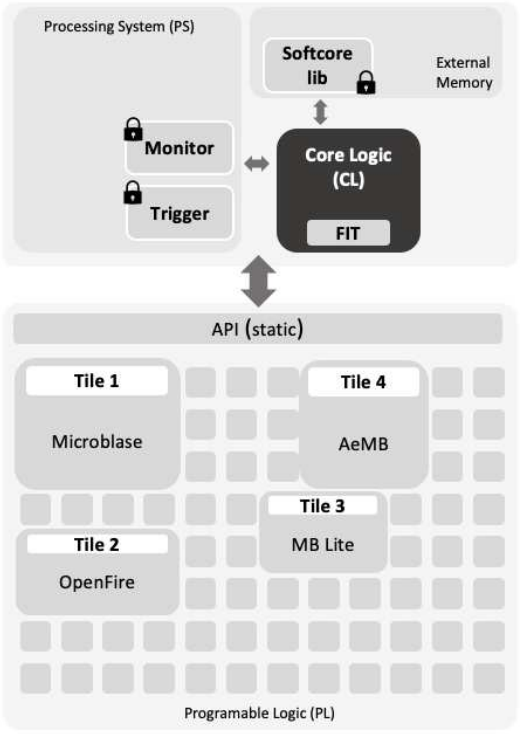

图 3：通用 Zynq 开发板上 Samsara 再生容错与入侵容忍框架的架构 [28]。四个瓦片由四个已知软核生成 [24]，以展现多样性。

（例如 *ARM TrustZone* [3]）。其目的是保护代码免遭篡改（包括复位时和执行期间）。另一方面，Samsara 的主要逻辑通过专用集成电路实现，因此不可更改。只要我们将 *Samsara* 的 代码占用空间保持得足够小，这一假设就是合理的。最后，我们假设攻击者可以攻陷副本的密码学身份验证密钥，但无法通过暴力破解来攻破密码学抽象（例如数字签名和哈希函数）。

3.2 概念

Samsara 中的再生理念，是基于从不同内部架构的软核位代码 模板中，重新启动 FPGA 内多样化的计算实例（即瓦片）。其目标是通过多样性以及对不同威胁等级的适应能力，提升嵌入式系统中基于硬件的故障与入侵容错（FIT）的有效性。遗憾的是，FIT协议的有效性取决于副本的多样程度：副本的内部实现或设计各不相同，但规格相同，以确保其外部功能和行为一致。如果副本（包括硬件层和软件层）完全相同，那么它们很可能会同时出错，原因可能是（1）遇到共同的异常，或（2）持续性对手成功访问这些副本后，在多个副本上同时利用共同的漏洞。*Samsara* 专注于硬件层的再生。它有助于以动态且可热插拔的方式，从不同模板实例化多样化的瓦片，即使在遭受攻击时，也能提高计算单元的多样性。具体而言，*Samsara* 框架有以下两个主要目标：

1. 能够启动同功能、不同构建的计算核心（瓦片）以确保多样性；2. 能够在运行时修改瓦片数量。

3.3 架构

我们在图 3 中展示了 Samsara 的高层架构。Samsara 由三个主要部分组成：(1) 核心逻辑（CL），即 ASIC 逻辑结构；(2) 处理系统（PS），即硬核处理单元；(3) 可编程逻辑（PL），即现场可编程门阵列。PS 和 PL 是大多数多处理器片上系统（例如 Zynq-7000 的逻辑结构）中的典型组成部分，而 CL 则是一种需要集成到该逻辑结构中的新型 ASIC 组件。

控制逻辑控制着再生的主要逻辑，并运行 FIT 协议。由于控制逻辑至关重要，因此建议将其部署在 ASIC硬件电路结构上，使其不可变，从而免受针对可重配置部分（PL）的攻击。只要控制逻辑占用的资源较少，这一方案便可行。控制逻辑可以访问软核库，其中包含一组多样化的 CPU 核心模板，以位文件的形式安全地存储在外部存储器中。此外，控制逻辑还可以访问 PS 中的监控器和触发器辅助模块（下文将详细说明）。这些模块以模块化方式扩展*Samsara* 的响应能力。

PL 是承载所创建瓦片的工作区部分，嵌入式操作在其中执行，因此再生也将在此发生。具体而言，PL 分为静态区域和动态区域。静态区域包含不可变逻辑及 API，用于连接 PS 和 PL。动态区域由可重配置区块（pblock）组成，可在运行时重新编程。

3.4 工作流

Samsara 中的典型执行工作流如下。嵌入式应用程序（其状态完整性至关重要）通过调用控制逻辑来执行操作。控制逻辑使用已实现的 FIT 协议，以副本方式运行该操作。根据 FIT 协议所需的副本数量 n，控制逻辑会在可编程逻辑中实例化 n 相应的瓦片。具体做法是从软核库中随机选择（因此具有多样性）软核模板。一个简单的 FIT 协议可以让控制逻辑将每项操作发送到不同瓦片并发执行，然后收集各自的结果进行比较。最终将一致（匹配）的结果返回给调用应用程序。可以使用复杂度和保证各不相同的 FIT 协议；不过，具体的 FIT 协议不在本文讨论范围内。

为使核心多样化，控制逻辑会随机选择一个正在运行的瓦片进行再生。默认的再生频率 $\mu \in [0, T]$是随机的，以混淆攻击者的模式识别。该再生协议的灵感来自 [42], [43] 中的软件再生。不过，我们设想在这种基于硬件的环境中可以进行优化，并计划在未来加以研究。简而言之，再生协议包括以下步骤：

- 在可编程逻辑中使用来自软核库的随机软核模板创建一个新的瓦片；
- 从其他瓦片传输状态，以启动新的瓦片；
- 销毁待再生的退役瓦片；以及
- 启动新创建的瓦片，以替代退役的瓦片。

3.5 优化

为简化起见，并保持较小的代码占用空间，选择要退役的瓦片以及用于生成新瓦片的软核时，默认采用随机策略，因此这些策略在 ASIC 逻辑结构的控制逻辑中实现。不过，也可以采用触发器模块中指定的更复杂策略，并使用监控器模块；这两个模块均在处理系统部分中加密并执行。监控器起到看门狗的作用：它使用启发式方法和/或其他系统检测故障、恶意或异常行为，并在检测到此类行为时触发再生。另一方面，以下触发策略也值得关注：

- 周期性：以周期性方式选择目标。
- 反应式：这是一种基于事件的策略，由监控器模块触发，例如检测到攻击或异常时。
- 主动式：这是最先进、最复杂的策略，会在不良事件（如攻击）发生之前触发再生。它可能会使用启发式方法或基于 AI 的智能。

监控器的另一个用途是在 FIT 协议中增加 n 瓦片数量时，在检测到更高威胁级别的情况下，即需要容忍 $t' > t$ 。更高的威胁可能来自能够访问软核代码（有时是开源代码）的攻击者。如果时间和资源充足，持久攻击者可能会识别软核中的漏洞，并在软核被实例化为瓦片时加以利用。在这种情况下，*Samsara*可以在 PL 区域中生成新的瓦片，而无需退役任何其他瓦片。同样，新创建的核心是从软核库中的不同模板克隆而来。我们将在未来的工作中详细说明这些内容。

4 可行性讨论

故障与入侵容忍正日益用于构建弹性系统，尤其是随着区块链的出现。人们曾认为，FIT SMR 协议在硬件层面过于复杂 [26]。事实上，FIT 协议通常被认为计算开销很大，原因包括：（1）副本之间为达成共识而交换的消息数量呈二次方增长（$O(n^2)$）；（2）大量使用密码学。然而，随着强大的硬件组件问世，可信抽象（即混合架构）的应用得以实现 [6], [7],，密码学计算效率不断提高，空间复杂度可降低至 $N = 2f + 1$，最终只需在副本之间进行相对简单的消息交换 [7], [20]。因此，正如本文所探讨的那样，FIT 正日益受到片上多核嵌入式系统领域的关注和研究。

同样，再生也被提出用于使软件多样化，作为 FIT SMR 协议的一部分 [42]。然而，在传统的多核架构中，FIT 协议无法受益于硬件再生 [26] 。得益于可重配置/可重编程硬件（例如，多处理器片上系统通过现场可编程门阵列提供的硬件 [29]），硬件再生变得易于实现。此外，现场可编程门阵列能够精细控制某个核心的重启或再生，并以热插拔方式进行，即无需重启整个系统。这取决于是否有多个核心模板可用，且这些模板都与单一多处理器片上系统平台兼容。

Samsara 面临的一项安全挑战是，控制器侧（核心逻辑）和可编程侧位于同一芯片上。这使攻击者能够攻陷控制器，阻碍再生，进而轻易攻击各个瓦片。为此，必须使用芯片上的专用集成电路硬件实现，对*Samsara's* 核心逻辑（控制逻辑）进行高度保护。我们的挑战是让控制逻辑保持精简，以便高效地用硬件实现。为此，鉴于优化模块（监控器和触发器）会带来开销，我们的设计将它们分离出来。在最坏情况下，控制逻辑本身受到保护，并可使用默认的再生配置，使瓦片更能抵御入侵攻击。

最后，这项概念分析需要通过实证实验来验证预期的正确性和性能。目前，我们正在赛灵思 Zynq 开发板 [29]上实现*Samsara* 概念验证原型。不过，我们的目标是保持*Samsara*架构的通用性，从而支持其他片上系统。值得测量的指标包括：代码占用空间、处理开销、FIT 吞吐量和延迟，以及再生时间。我们还计划调整所用协议，并提供相应的正确性和安全性证明。

5 结论

我们介绍了 *Samsara*，这是首个面向故障与入侵容错（FIT）多样化的片上系统再生框架。*Samsara* 利用多处理器片上系统的可编程计算资源，生成可运行 FIT 副本的新型多样化计算核心（瓦片）。通过从不同的软核模板实例化瓦片，可使其实现多样化；这些模板很可能由不同供应商提供。该框架的核心逻辑不可变，并由硬件保护；而瓦片则利用再生来多样化其底层计算逻辑。我们进行了概念分析，表明基于初步框架，在硬件上实现再生是可行的。同时，我们正在推进实现工作和实证评估，以便在未来的论文中验证分析结果。

参考文献

[1] S. Karandikar, H. Mao, D. Kim, D. Biancolin, A. Amid, D. Lee, N. Pemberton, E. Amaro, C. Schmidt, A. Chopra 等，“Firesim：公有云中基于FPGA加速的周期精确可扩展系统仿真”，载于 *2018*年*ACM/IEEE*第*45*届计算机体系结构年度国际研讨会（*ISCA*）。IEEE，2018年，第29–42页。

[2] V. Costan 和 S. Devadas，“英特尔 SGX 解析”，技术报告，麻省理工学院，2016年。 [在线]。获取地址：https://eprint.iacr.org/2016/086.pdf

[3] ARM，“Armtrustzone”，*ARM*，访问于2022年8月。[在线]。获取地址：https://www.arm.com/products/security-on-arm/trustzone [4] S. L. Kinney，《可信平台模块基础：在嵌入式系统中使用 TPM》。爱思唯尔，2006年。[5] M. Sabt、M. Achemlal 和 A. Bouabdallah，“可信执行环境：它是什么，又不是什么”，载于 *2015*年*IEEE Trust- com/BigDataSE/ISPA*，第1卷。IEEE，2015年，第57–64页。[6] P. E. Ver´ıssimo，“穿越虫洞：重新审视分布式系统模型”，*SIGACT News*，第37卷，第1期，第66–81页，2006年3月。 [在线]。获取地址：https://doi.org/10 1145 1122480 1122497 。/。G. S. Veronese、M. [7] Correia、A. N. Bessani、L. C. Lung 和 P. Verissimo，“高效的拜占庭容错”，*IEEE*计算机汇刊，第62卷，第1期，第16–30页，2013年。[8] Y. Kim、R. Daly、J. Kim、C. Fallin、J. H. Lee、D. Lee、C. Wilkerson、K. Lai 和 O. Mutlu，“无需访问内存即可翻转其中的比特：DRAM扰动错误的实验研究”，载于 *2014*年*ACM/IEEE*第*41*届国际计算机体系结构研讨会，2014年，第361–372页。[9] P. Kocher、J. Horn、A. Fogh、D. Genkin、D. Gruss、W. Haas、M. Hamburg、M. Lipp、S. Mangard、T. Prescher、M. Schwarz 和 Y. Yarom，“幽灵攻击：利用推测执行”，载于 *2019*年*IEEE*安全与隐私研讨会，2019年，第1–19页。[10] M. Lipp、M. Schwarz、D. Gruss、T. Prescher、W. Haas、J. Horn、S. Mangard、P. Kocher、D. Genkin、Y. Yarom、M. Hamburg 和 R. Strackx，“熔断漏洞：从用户空间读取内核内存”，*Commun. ACM*，第63卷，第6期，第46–56页，2020年5月。[在线]。获取地址：https://doi.org/10 1145 3357033 。/。A. P. Fournaris、 [11] L. Pocero Fraile 和 O. Koufopavlou，“利用硬件漏洞攻击嵌入式系统设备：强大微架构攻击综述”，*Electronics*，第6卷，第3期，第52页，2017年。[12] P. Prinetto 和 G. Roascio，“硬件安全、漏洞与攻击：全面的分类体系。”载于 *ITASEC*，2020年，第177–189页。[13] T. Zhang 和 R. B. Lee，“用于测量侧信道泄漏的安全缓存建模”，技术报告，普林斯顿大学，2014年。[14] 2022，“使用加密和身份验证保护超大规模/UltraScale+ FPGA比特流”，赛灵思，2022年。 [在线]。获取地址：https://www.xilinx.com/content/dam/xilinx/support/document

[15] S. Adee，《寻找终止开关》，*IEEE Spectrum*，第45卷，第5期，第34–39页，2008年。[16] R. L. Merlino 和 J. A. Goree，《实验室、工业和太空中的尘埃等离子体》，《今日物理》，第57卷，39，2004年7月，第32期，第–页。[17] J. R. Celaya、P. Wysocki、V. Vashchenko、S. Saha 和 K. Goebel，《用于预测功率半导体器件状态的加速老化系统》，载于*2010*年*IEEE Autotestcon*会议。IEEE，2010年，第1–6页。[18] A. Al-Boghdady、K. Wassif 和 M. El-Ramly，《物联网低端设备操作系统中安全漏洞的存在、趋势及成因》，《传感器》，第21卷，第7期，2021年。 [在线]。可访问：https://www.mdpi.com/1424 8220 21 7 2329 - / / / P. E. [19] Veríssimo、N. F. Neves 和 M. P. Correia，《入侵容忍架构：概念与设计》，载于可信赖系统架构，R. de Lemos、C. Gacek 和 A. Romanovsky 编。柏林、海德堡：施普林格·柏林·海德堡，2003年，第3–36页。[20] I. P. Gouveia、M. V¨olp 和 P. Esteves-Verissimo，《最后一道防线背后——在片上系统故障和入侵中幸存》，2020年。[ ] 2005 04096 在线。可访问：[21] https://arxiv.org/abs/。F. B. Schneider，《使用状态机方法实现容错服务：教程》，《*ACM*计算调查》（*CSUR*），第22卷，第4期，第299–319页，1990年。[22] C. Cachin、K. Kursawe 和 V. Shoup，《君士坦丁堡中的随机预言机：利用密码学实现实用的异步拜占庭协议》，《密码学杂志》，第18卷，246，2005年，第3期，第219页。[23] P. Sousa、N. Neves 和 P. Verissimo，《分布式故障/入侵容忍系统的弹性如何？》，载于*2005*年国际可信赖系统与网络会议（*DSN'05*），2005年，第98–107页。[24] M. Makni、M. Baklouti、S. Niar、M. W. Jmal 和 M. Abid，《面向嵌入式多核系统的现场可编程门阵列软核比较与性能评估》，载于*2016*年第*11*届国际设计与测试研讨会（*IDT*）。IEEE，2016年，第154–159页。

[25] P. Sousa、N. F. Neves 和 P. Ver´ıssimo，《通过架构混合化实现主动韧性》，载于*2006*年*ACM*应用计算研讨会论文集，2006年，第686–690页。[26] J. Behl、T. Distler 和 R. Kapitza，《面向多核的可扩展拜占庭容错：基于参与者的分解与面向共识的并行化》，载于第*10*届*USENIX*系统可信性热点会议论文集，丛书：HotDep'14。美国：USENIX 协会，2014年，第9页。[27] Z. Guo、C. Hong、M. Yang、D. Zhou、L. Zhou 和 L. Zhuang，《Rex：以多核速度进行复制》，载于第九届欧洲计算机系统会议论文集，丛书：EuroSys '14。美国纽约州纽约市：计算机协会，2014年。[在线]。可访问：https://doi.org/10.1145/2592798.2592800[28] 2020，《Zynq Ultrascale器件技术参考手册》，赛灵思，2020年。[在线]。可访问：https://docs.xilinx.com/v/u/en-US/ug1085-zynq-ultrascale-trm[29] 赛灵思2019，《Ug1085：Zynq Ultrascale器件技术参考手册》，赛灵思，2019年。[30] 2021，《Zynq-7000片上系统技术参考手册》，赛灵思，2021年。[在线]。可访问：https://docs.xilinx.com/v/u/en-US/ug585-Zynq-7000-TRM[31] 2022，《Zynq Ultrascale器件技术参考手册》，赛灵思，2022年。[在线]。可访问：https://www.xilinx.com/content/dam/xilinx/publications/product-briefs/xil

[32] ——，“Versal ACAP 技术参考手册（am011，”赛灵思，2022年。[在线]。网址：https://docs.xilinx.com/r/en-US/am011-versal-acap-trm/Introduction [33] L.Crockett、D. Northcote、C. Ramsay、F. Robinson 和 R. Stewart，探索 *Zynq MPSoC*：借助 *PYNQ* 和机器学习应用。格拉斯哥：斯特拉斯克莱德学术传媒，2019年4月。[在线]。网址：https://www.zynq-mpsoc-book.com/[34]赛灵思，“Ug909：Vivado 设计套件用户指南：动态功能交换”，赛灵思，2018年。[在线]。网址：https://docs.xilinx.com/r/2021.1-English/Vivado-Design-Suite-User-Guide-D

[35] L. Lamport、R. Shostak 和 M. Pease，“拜占庭将军问题”，*ACM*《编程语言与系统汇刊》，第 4 卷，第 3 期，第 382–401 页，1982 年 7 月。 [在线]。网址：https://doi.org/10 1145 357172 357176 。/。M. Castro 和 B. [36] Liskov，“实用拜占庭容错与主动恢复”，*ACM*《计算机系统汇刊》，第 20 卷，第 4 期，第 398–461 页，2002 年 11 月。 [在线]。网址：https://doi.org/10 1145 571637 571640 。/。R. Kotla、L. Alvisi、M. [37] s/应用说明/xapp1267-encryp-efuse-program.pdf Dahlin、A. Clement 和 E. Wong，“Zyzzyva：推测式拜占庭容错”，*ACM*《计算机系统汇刊》，第 27 卷，第 4 期，2010 年 1 月。 [在线]。网址：https://doi.org/10 1145 1658357 1658358 。/。K. Kanoun、M. Kaaniche、 [38] C. Beounes、J.-C. Laprie 和 J. Arlat，“容错软件的可靠性增长”，*IEEE*《可靠性汇刊》，第 42 卷，第 2 期，第 205–219 页，1993 年。[39] S. Hosseinzadeh、S. Rauti、S. Laur´en、J.-M. M¨akel¨a、J. Holvitie、S. Hyrynsalmi 和 V. Lepp¨anen，“软件安全的多样化与混淆技术：系统性文献综述”，《信息与软件技术》，第 104 卷，第 72–93 页，2018 年。 [在线]。网址：https://www.sciencedirect.com/science/article/pii/S0950584918301484 [40] M. Garcia、A. Bessani 和 N. Neves，“Lazarus：拜占庭容错系统中多样性的自动管理”，载于第 *20* 届国际中间件会议论文集，丛书名 Middleware '19。美国纽约州纽约：计算机协会，2019 年，第 241–254 页。 [在线]。网址：https://doi.org/10 1145 3361525 3361550 。/。M. Castro 和 B. Liskov，“实 [41] 用拜占庭容错”，载于第三届操作系统设计与实现研讨会论文集，丛书名 OSDI '99。美国：USENIX 协会，1999 年，第 173–186 页。[42] Y. Huang、C. Kintala、N. Kolettis 和 N. D. Fulton，“软件再生：分析、模块与应用”，载于第二十五届国际容错计算研讨会论文摘要。IEEE，1995 年，第 381–390 页。[43] D. Silva、R. Graczyk、J. Decouchant、M. Volp 和 P. Esteves-Verissimo，“威胁自适应拜占庭容错状态机复制”，载于*2021* 年第 *40* 届国际可靠分布式系统研讨会（*SRDS*）。美国加利福尼亚州洛斯阿拉米托斯：IEEE 计算机学会，2021 年 9 月，第 78–87 页。 [在线]。网址：https://doi.ieeecomputersociety.org/10.1109/SRDS53918.2021.00017