TASPDetect: Reviving Trust in 3PIP By Detecting TASP Trojans

Chidhambaranathan Rajamanikkam, Kurt Brenning, Andrew Deakin, Sanghamitra Roy, Koushik Chakraborty

USU BRIDGE LAB, Electrical and Computer Engineering, Utah State University, United States

A R T I C L E   I N F O

Keywords:
Hardware security
3rd party intellectual property
System-on-Chip

A B S T R A C T

Trustworthiness is an emerging concern in the development of Multi-Processor System-on-Chip (MPSoC) systems, which often include Third-Party Intellectual Property (3PIP) cores. Malicious hardware trojans in these 3PIP components can create a plethora of system vulnerabilities. In this paper, we explore an imminent threat that originates from a 3PIP core: architectural state preserving trojans (TASP). We illustrate that a malicious TASP can affect the performance of a core without altering its architectural state, thereby disrupting the availability of on-chip resources in a MPSoC system. We propose three complementary techniques to detect active TASP cores with overheads of less than 5%.

1. Introduction

The growing popularity of Third Party Intellectual Property (3PIP) components in Multi-Processor System-on-Chips (MPSoCs) is posing significant challenges for secure hardware design [21]. Malicious functionality in a 3PIP, or hardware trojans, can be used for causing a range of harms at the system/application level covering the three central pillars of security assurance: confidentiality, integrity, and availability. For example, the confidentiality of a MPSoC can be compromised if a hardware trojan gives the attackers back door access to sensitive information [22]. Likewise, other harmful effects of trojans include functionality change, performance degradation, and denial of service [16].

In this paper, we explore 3PIP architectural state preserving trojans (TASP). These are trojans embedded in a 3PIP core that preserve the architectural state of the processor similar to a trojan free core. Fig. 1 illustrates the concept of a TASP relative to typical trojans. While most commonly studied trojans corrupt portions of the externally visible architectural states, including the registers and the memory, a TASP does not. For example, a TASP might flip the prediction bit of branch prediction, or increase access latency in the cache. Such actions do not corrupt the architectural state of the processor. Consequently, a carefully designed TASP may evade detection, while continuously degrading the performance delivered to an end-user.

A TASP brings unique challenges in detection and mitigation, despite its apparent similarity with performance degrading trojans described in the trojan taxonomy [16]. Two of the key challenges include: a) limited gate level visibility inside a 3PIP core, and b) false positives generated by noise in the system. First, most 3PIP cores are delivered as “black boxes” to protect the intellectual property of the third party [1]. Consequently, malicious functionality embedded in these components can easily bypass many existing techniques for hardware trojan detection that rely on gate-level inspection [23]. Second, false positives can be flagged due to performance fluctuation from process variation effects and non-deterministic architectural events in a healthy core.

To detect such a potent threat in modern MPSoCs, we present three detection techniques covering a spectrum of TASP attacks. These comprise: (1) an analytical model based method to detect always-on TASP; (2) a runtime technique to detect intermittent TASP using another core; and (3) a runtime method to detect intermittent TASP without using other cores.

We make following contributions in this paper.

• We explore a potent threat model as 3PIPs grow in prominence in modern MPSoCs. We evaluate the performance impact of TASP on a modern processor (Section 2).
• We propose a Specification-centric Performance-aware Analytical Representation (SPAR) that uses an analytical model to detect always-on TASP (Section 3.2).
• We propose two runtime techniques: Covert Performance Monitoring System (CPMS) and Employing Side-channel analysis for TASP Detection (ESTD) to detect intermittent TASP during the lifetime of the chip (Sections 3.3 and 3.4).
• We demonstrate the efficacy of SPAR, CPMS, and ESTD (Section 5).
2. TASP: A Conceptual Overview

In this section, we first present a broad overview of a TASP, its relevance (Section 2.1) and its major classifications (Section 2.2). We then discuss the design of the TASP and its resulting damaging impact in ctions 2.3 and 2.4, respectively. Finally, we evaluate its design footprint in Section 2.5.

2.1. TASP Relevance in IC supply chain

Fig. 2 illustrates the TASP threat model in the context of a typical supply chain for 3PIP integration in a MPSoC. To foster a compelling competitive edge, MPSoC integrators rely heavily on 3PIP vendors for common MPSoC blocks such as processing elements. 3PIP vendors may deliver these blocks as either technology independent soft IP (exposed RTL) or technology dependent hard IP (protected RTL). The hard IP—a preferred mode for 3PIP vendors-offers only a functional signature (input, output, and functional description) to the MPSoC integrator. The 3PIP vendors then provide the fully exposed RTL directly to the fabrication house, where the entire design can be seamlessly integrated and fabricated. This widely popular model has serious implications towards circuit level vulnerability. Without a golden model and gate level visibility, a TASP can easily bypass existing post-silicon verification methods implemented by the MPSoC integrator. A vendor or a malicious design engineer in the 3PIP design house can use this potent model to harm a MPSoC integrator, or other clients such as government agencies [6].

2.2. Types of TASP Trojans

TASPs can be classified into two major categories based on their activity period: always-on and intermittent.

2.2.1. Always-on

An always-on TASP is active throughout the lifetime of the chip. The malicious designer in the 3PIP design house activates the TASP before the 3PIP core is delivered to MPSoC integrator. Moreover, these types of trojans are difficult to detect in a MPSoC system due to the absence of the golden model.

2.2.2. Intermittent

These TASPs are active sporadically, and thus interleave between inactive and active phases. Their active phase may be triggered through a variety of conditions, such as certain input patterns or external conditions (e.g., temperature). The period of active and inactive phases are assumed to be random as it is depends on discretion of that attacker. For the purpose of this work, we focus on the impact of intermittent TASPs when active and explore techniques to detect them irrespective of the trigger mechanisms.

An intermittent TASP can have several major operational phases: (i) Dormant; (ii) Activation; (iii) Operational. In the dormant phase, the trojan is simply awaiting a specific trigger mechanism to become active [21]. Delaying this activation essentially allows the TASP to evade post-silicon detection. After activation, a TASP becomes operational, where it performs its intended function by altering internal data and control signals in the 3PIP core.

2.3. Design of a TASP

Fig. 3(a) illustrates the design and operation of TASP-BPU and TASP-cache. TASP Monitoring Unit (TMU) monitors the instruction sequence being fetched from instruction cache. The TASP in the BPU, once activated, flips the branch prediction outcomes. As a result, incorrect instructions are fetched from I-cache and go through the various stages of pipeline for execution. At the end of execute stage, the branch detection model, shown in Fig. 3(a), identifies the incorrect branch prediction and therefore, the instructions present in the fetch, decode and execute pipe stages are flushed. Then, the correct instructions is fetched from the I-cache and propagated through the pipeline. Similarly, TASP in L1 I-cache and L1 D-cache (shown in Fig. 3(a)), once activated, increases the L1 cache access time. When the memory instructions, such as load and store, are executed, TASP-cache inserts additional latency when the L1 cache is accessed. As a consequence, the memory access time of those instructions are significantly affected. Both TASP-BPU and TASP-cache affects the performance of the 3PIP core without altering its architectural states such as register files and memory.

2.4. Performance impact of a TASP

We outline our methodology (Section 2.4.1) and results (Section 2.4.2) to demonstrate the performance impact of a TASP.

2.4.1. Evaluation methodology

Using the Sniper simulator [7], we inject TASPs in the Branch Prediction Unit (BPU) and the L1 cache of a modeled single core Intel

![Fig. 2. TASP threat model in the context of the existing supply chain of a 3PIP block.](image-url)
دریافت فوری
متن کامل مقاله

امکان دانلود نسخه تمام متن مقالات انگلیسی
امکان دانلود نسخه ترجمه شده مقالات
پذیرش سفارش ترجمه تخصصی
امکان جستجو در آرشیو جامعی از صدها موضوع و هزاران مقاله
امکان دانلود رایگان ۲ صفحه اول هر مقاله
امکان پرداخت اینترنتی با کلیه کارت های عضو شتاب
دانلود فوری مقاله پس از پرداخت آنلاین
پشتیبانی کامل خرید با بهره مندی از سیستم هوشمند رهگیری سفارشات