Article

U-Can-Inject-Errors (UCIe): A Protocol-Aware Hardware Trojan for FPGA Chiplet Links

Harshvadan Mihir *, Arsalan Ali Malik, Sharath Pendyala, Aydin Aysu

Posted Date: 22 September 2026

Keywords: heterogeneous FPGA security; UCIe; hardware trojan; targeted misclassifications; silent data corruptions

 doi:10.20944/preprints202609.1942.v1

Article

U-Can-Inject-Errors (UCIe): A Protocol-Aware Hardware Trojan for FPGA Chiplet Links

Harshvadan Mihir *, Arsalan Ali Malik, Sharath Pendyala and Aydin Aysu

North Carolina State University, Raleigh, NC, USA
* Correspondence: hmihir@ncsu.edu

Abstract

Chiplet integration enables designers to assemble dies from multiple vendors and connect them through standardized interconnect protocols, such as universal chiplet interconnect express (UCIe). This multi-vendor model, however, redraws the trust boundary and exposes inter-chiplet links to hardware Trojan insertion. Yet this attack surface remains largely unexplored, especially in FPGA-based chiplet ecosystems. In this work, we present the first hardware Trojan that stealthily bypasses the cyclic redundancy check (CRC)-based integrity mechanism of the UCIe protocol. The Trojan deliberately flips data bits in ways that preserve the original checksum, allowing corrupted inter-chiplet traffic to pass CRC validation undetected. We model the Trojan in a multi-die FPGA AI inference engine and show that these evasive corruptions can induce targeted misclassification, including (i) input-class suppression, (ii) forced-class promotion, and (iii) conditional class redirection. The Trojan incurs an overhead ranging from 18–35 and flip-flops (FFs) ranging from 14–17 when implemented on a Kintex-7 FPGA. Our results expose a critical gap in UCIe's integrity mechanism. Although CRC remains effective for random error detection, its linear structure enables protocol-aware Trojans to inject checksum-preserving corruptions. The integrity gap identified in our work motivates the need to move beyond error-detection codes to secure UCIe-based chiplet interconnects.

Keywords: heterogeneous FPGA security; UCIe; hardware trojan; targeted misclassifications; silent data corruptions

1. Introduction

Heterogeneous integration is changing how modern systems are built. Rather than manufacturing a large monolithic die, chip designers now combine smaller dies called *chiplets*, often sourced from different vendors and integrated over a common substrate [1,2]. Field-programmable technology adopted this technique almost a decade before the advent of chiplet systems. Xilinx's Virtex-7 2000T field programmable gate array (FPGA) released in 2011 connected four reconfigurable fabric dies over a die-to-die (D2D) fabric before the industry formalized the chiplet paradigm [3,4].

This heterogeneity, however, introduces a trust boundary that monolithic FPGA devices never had to contend with: *the active interposer*. Unlike passive interposers that provide only metal routing, active interposers integrate logic, buffers, and D2D interface controllers, enabling plug-and-play integration [5]. As the interposer is physically distinct from the FPGA fabric die(s), it may be fabricated at a different foundry or integrated by an external packaging house, making it an ideal target for a man-in-the-middle attack via Trojan insertion [6].

The universal chiplet interconnect express (UCIe)[1] protocol defines the communication protocol governing these D2D links [7]. It is rapidly emerging as the de facto standard for next-generation multi-die FPGA assemblies [8]. As the adoption of UCIe grows, so does the importance of understanding its

[1] The playful expansion of UCIe in the title is a mnemonic for our technical contribution and is not intended to disparage the standard or its consortium.

 doi:10.20944/preprints202609.1942.v1

security properties. Yet the security implications, particularly its integrity mechanisms, remain largely unexplored [9]. UCIe relies on a cyclic redundancy check (CRC) to detect transmission errors [10]. CRC is well-suited against random noise, yet it has not been tested against a *protocol-aware* adversarial fault injection.

In this work, we show the *first-ever* hardware Trojan capable of inducing targeted, evasive corruption on UCIe links. We explore the construction of minimal, protocol-aware Trojan that flips *only four payload bits*, the minimum needed to cause a 'CRC collision'[2]. This causes the receiver's integrity check to get bypassed unconditionally, while the Trojan operates with zero added latency. We study this threat in the context of a hypothetical multi-die FPGA-based artificial intelligence (AI) inference engine, in which carefully chosen corruptions induce targeted misclassifications while evading detection.

Our key contributions are as follows:

- **Novel UCIe-tailored Trojan.** We reveal an unexamined attack surface by introducing a novel hardware Trojan tailored to UCIe links. The Trojan is designed to exploit the algebraic structure of the protocol's CRC to corrupt chiplet-to-chiplet traffic while evading detection.
- **Theoretical analysis of UCIe's integrity mechanism.** We derive the minimal strategic bit-flip patterns required for CRC collision and show that just *four* bit flips are sufficient. We also prove that a single-cycle Trojan operation is insufficient and that Trojan activity must span at three distinct clock cycles to flip the required four bits.
- **End-to-end attack case study.** We model the Trojan in a multi-die FPGA AI inference engine and demonstrate targeted misclassification, including (i) input-class suppression, (ii) forced-class promotion, and (iii) conditional class redirection, achieving success rates of 100%, 53.56%, and 14.14% respectively. The Trojan requires an overhead of look-up tables (LUTs) ranging from 18 to 35 and flip-flops (FFs) ranging from 14 to 17 when implemented on a Kintex-7 FPGA.
- **Flit[3] mode extension and countermeasure study.** We demonstrate that the Trojan can be extended across two UCIe Flit modes with negligible redesign. We further evaluate three countermeasure classes namely (i) wider-polynomial CRC, (ii) keyed CRC, and (iii) cryptographic message authentication code (MAC), to show that cryptographic authentication represents a practical path toward closing the discussed vulnerability.

The rest of the paper is organized as follows: Section 2 covers the UCIe protocol stack, the identified integrity gap, and existing works in the field of Trojans in heterogeneous systems. Section 3 defines our threat model. Section 4 derives the attack payload and describes the Trojan operation. Section 5 presents the AI inference case study. Section 6 reports implementation results, stealth analysis, and the extension to other Flit formats. Section 7 discusses its resilience against existing detection schemes and evaluates possible countermeasures, and finally Section 8 concludes our work.

2. Background and Related Work

In this section, we present the UCIe protocol stack and the identified structural integrity gap. We follow this up with a discussion on hardware Trojan research in heterogeneous FPGA-based systems to situate our attack surface.

2.1. UCIe Protocol

UCIe defines a serial, layered on-package interconnect that specifies a complete protocol stack comprising three distinct layers [10]. They are defined as follows: (i) Protocol layer encapsulates either one of peripheral component interconnect express (PCIe), compute express link (CXL), or streaming data into UCIe Flit. (ii) D2D adapter generates the final Flit by appending a 2-byte header and 2-byte CRC checksum, and (iii) Physical layer (PHY) defines serial link parameters like lane count, data rates, and byte-to-lane mapping.

[2] CRC collision is when different inputs produce the same checksum.
[3] A Flit (flow control unit) is a fixed-size data packet used by UCIe for flow control, scheduling, and die-to-die transmission.

 doi:10.20944/preprints202609.1942.v1

2.2. Integrity Gap

The CRC detects random transmission errors. However, a CRC is *algebraically predictable*. Hence, a man-in-the-middle adversary aware of the CRC specifications can construct error patterns that produce a CRC collision, corrupting in-flight data while passing the integrity check unconditionally. The UCIe specification defers application-layer security to the Protocol layer, while assuming the physical link to be a reliable medium. This division creates a security responsibility gap: higher-layer protocols assume the link faithfully delivers what was sent, while the link layer assumes it needs to only defend against noise. Neither layer is designed to counter an informed adversary residing in the link path itself. Notably, the newest version of the specification (UCIe 3.0) introduces a formal security architecture, but this applies exclusively to management-plane traffic [11]. The mainband data Flits—the very channel our Trojan targets—is not addressed.

Although PCIe and CXL define their own integrity fields, bypassing the D2D Adapter CRC alone is sufficient to compromise link integrity, because UCIe explicitly disables these higher-layer checks at the Protocol layer. UCIe mandates that the Protocol layer drive the link CRC (LCRC) bytes to all zeros and that receivers ignore them [10]. Indeed, the specification assumes the D2D Adapter provides reliable Flit transport. Frame errors detected by the Protocol layer are attributed to uncorrectable internal errors, not link-level tampering [11]. PCIe and CXL do define an optional end-to-end CRC (ECRC), but this requires hardware support in every chiplet and appends 4 extra bytes to every packet, adding latency that directly contradicts UCIe's own design recommendations for latency-sensitive integrations. In sum, the D2D Adapter CRC may operate as the *only* recommended integrity check on the mainband data path. This is the singular, algebraically predictable check that our Trojan exploits.

2.3. Hardware Trojans in Reconfigurable and Chiplet Systems

Traditional hardware Trojan research in reconfigurable systems spans bitstream manipulation, fabric routing exploitation, and runtime delivery via partial reconfiguration [12]. At the bitstream level, prior works demonstrated Trojan insertion by modifying unencrypted configuration files and by reverse-engineering proprietary bitstream formats [13,14]. Within the fabric, malicious logic and routing modifications have been shown to cause a Trojan to remain dormant until post-programming activation [15–17]. Post-deployment, dynamic partial reconfiguration has been exploited to deliver a Trojan at runtime without disturbing the remainder of the design [18]. All these works consider intra-die as the attack surface.

Multi-die Trojan research extends this surface to the packaging boundary [19]. Chacon et al. demonstrate coherence-oriented Trojans inserted into a chiplet or interposer that corrupt traffic or establish covert channels in cache-coherent multi-die systems [20]. Zhang and Yu model vertically distributed triggers in 3D-IC stacks, showing that splitting trigger and payload logic across tiers substantially improves stealth [21]. These works establish an inter-die attack surface, but address custom coherence mechanisms or generic 3D stacks rather than standardized die-to-die protocols.

To our knowledge, no prior work has analyzed Trojans that exploit the protocol and integrity-layer semantics of a standardized D2D interface such as UCIe, leaving this attack surface uncharacterized in the hardware security literature.

3. Threat Model

This section solidifies the threat model by defining the attack scenario, primary threat, assumptions, and adversarial objectives that dictate the analysis presented in our work.

We consider a multi-die FPGA manufactured with an *active interposer*. Unlike passive interposers that provide only physical routing, an active interposer incorporates on-package logic including network-on-chip (NoC) routing, power delivery, and design-for-test support [22]. The primary threat in this context is a hardware Trojan embedded in one of the router modules. This Trojan becomes a man-in-the-middle between inter-die data exchanges, enabling attacks including payload corruption, traffic injection, and flow control manipulation [23].

 doi:10.20944/preprints202609.1942.v1

Figure 1 illustrates our system's assumed routing configuration. The NoC employs a daisy-chain topology in which each die connects to an associated router [24]. We assume that the dies are interconnected via an **x64 UCIe interface**.[4] We model the adversary as an *untrusted foundry* commissioned to manufacture the active interposer, capable of inserting a register transfer language (RTL)-level Trojan into the interposer's routing logic at fabrication time [20,25].

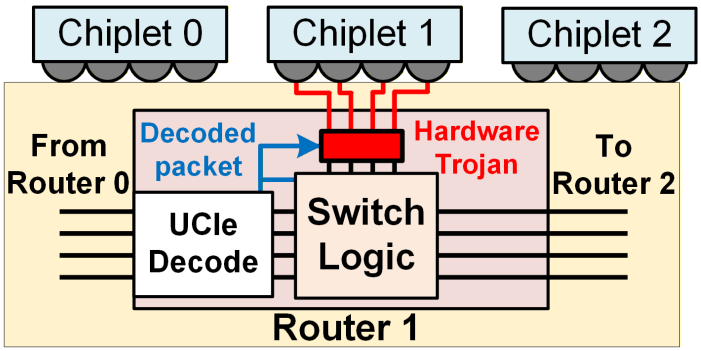

Figure 1. Threat model emphasizing a malicious Trojan insertion at the active interposer's network-on-chip (NoC) component. A Trojan is inserted at one of the NoC routers that possesses access to the incoming decoded packet used to evaluate the packet destination. It intercepts the serial data lanes directed to Chiplet 1 and injects the evasive faults.

The attacker's knowledge required for executing this silent manipulation demands no reverse engineering. UCIe is an open, published standard, and the D2D Adapter CRC is the sole mainband integrity mechanism, as established in Section 2.2. The knowledge of the die roles, bump assignments, and I/O configurations is shared with the interposer foundry during package integration [26]. The payload structure should be derivable from the chiplet I/O specifications included in the same integration package.

With access to decoded in-flight packets and knowledge of the UCIe payload structure, the adversary can identify and selectively corrupt specific bytes in the outbound data stream. The Trojan remains dormant until it detects the anticipated transfer patterns, activating only within predictable windows to perform the corruption. By exploiting the protocol's reliance on CRC-based error detection, the Trojan achieves **targeted, evasive corruptions** of inter-die traffic. The receiver's integrity check is structurally incapable of distinguishing these corruptions from legitimate traffic.

The proposed Trojan targets the mainband interface; therefore, we consider the sideband interface, link initialization, and training out of scope. Additionally, since our threat model focuses on D2D communication on the same package, we exclude the forward error correction (FEC) mechanism typically used in UCIe retimers to facilitate off-package links.

4. Hardware Trojan Architecture and Design

This section derives the CRC-colliding error patterns that the Trojan injects, proves the minimum Trojan activity bound required for evasion, and describes the faulting mechanism.

4.1. Attack Payload Derivation

UCIe computes a CRC-16/UMTS checksum over a fixed 128-byte input formed by zero-padding the Flit data [10]. Over Galois field GF(2), the checksum is the remainder of polynomial division of the input by the generator $G(x)$ specified by the protocol as:

$$G(x) = (x+1)(x^{15}+x+1) = x^{16}+x^{15}+x^2+1. \quad (1)$$

[4] An x64 UCIe link utilizes 64 serial data lanes, a pair of half-rate sampling clocks (`clk_fwd_p`/`clk_fwd_n`), and a `valid` signal for packet framing.

 doi:10.20944/preprints202609.1942.v1

Because addition is XOR in GF(2), XORing any *multiple* of $G(x)$ into the data does not change the remainder:

$$E(x) = M(x) \cdot G(x) \Rightarrow \mathrm{CRC}\big(D(x) \oplus E(x)\big) = \mathrm{CRC}\big(D(x)\big), \tag{2}$$

where $E(x)$ is an error polynomial XORed with the message under protection $D(x)$, and $M(x)$ is a non-zero multiplier. UCIe processes input bits least significant bit (LSB) first, and uses the raw remainder without post-processing [27]. This is equivalent to performing CRC division with the reciprocal (bit-reversed) of the original generator $G'(x)$:

$$G'(x) = x^{19} + x^{17} + x^4 + x^3 = x^3 \cdot F(x). \tag{3}$$

From (3), we derive the *fundamental polynomial*:

$$F(x) = x^{16} + x^{14} + x + 1. \tag{4}$$

Therefore, all nonzero multiples of $F(x)$ are the required CRC-colliding error patterns: when XORed with the original message, they yield the same checksum as the original, and the receiver is structurally unable to distinguish the corrupted from legitimate data. To realize such error patterns during transmission, the corresponding polynomial terms must be mapped onto the specific physical lanes and clock cycles imposed by the UCIe PHY interface.

Figure 2 shows UCIe's strict serialization scheme: over each lane, a byte of serial data is transmitted across eight cycles of the receiver's system clock. Byte 0 (B_0) maps to lane 0, byte 1 (B_1) to lane 1, and so forth. For a byte $n \in \{0, \ldots, 63\}$, bit 0 ($B_n[0]$) is transmitted during cycle 0, bit 1 ($B_n[1]$) during cycle 1, and so on. Let $k \in \{0, \ldots, 7\}$ denote the cycle number, a single-cycle activation at cycle k restricts the Trojan to manipulating a fixed bit index ($B_n[k]$) within every byte simultaneously.

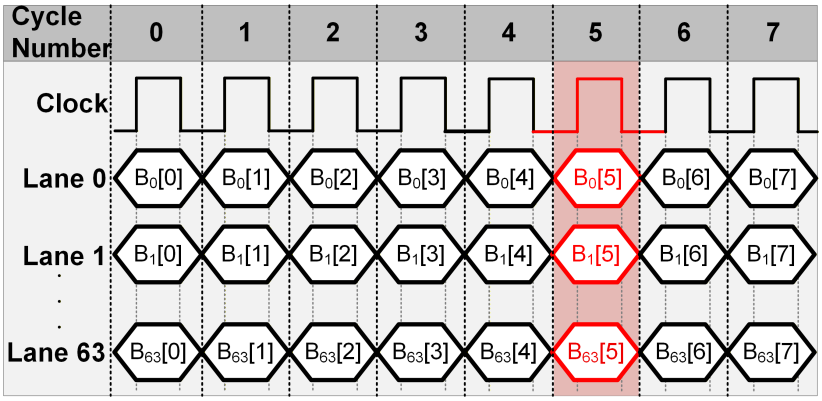

Figure 2. Timing diagram of UCIe serial data transmission scheme over an x64 link. Each lane carries serial data over 8 clock cycles. Lane k transmits byte k (shown as B_k). The Trojan activation cycle (cycle 5 in this case) and the corresponding bits affected ($B_k[5]$) are marked in red.

The UCIe CRC detects any error pattern with up to three flipped bits. Consequently, an error must flip at least four bits to evade detection. This matches the four nonzero terms of (4). The Trojan is designed to inject bit-shifted instances of $F(x)$. These are the *minimal* CRC-colliding error patterns. Theorem 1 formalizes the physical constraints required to realize these patterns on the UCIe x64 interface.

Theorem 1 (Insufficiency of Single-Cycle Logic and Minimum Activity Bound for Evasion)**.** *A hardware Trojan restricted to a single-cycle operation cannot generate any nonzero error polynomial $E(x)$ that is a multiple of the fundamental polynomial $F(x) = x^{16} + x^{14} + x + 1$. Two-cycle operation is likewise insufficient for any minimal CRC-colliding pattern. The minimum Trojan activity required to inject an undetected corruption spans at least three distinct clock cycles.*

 doi:10.20944/preprints202609.1942.v1

Proof. The UCIe protocol serializes data by transmitting each byte least significant bit (LSB)-first over 8 consecutive clock cycles. Any data bit represented by the polynomial term x^e is transmitted during clock cycle $c = e \bmod 8$; we refer to these eight temporal windows as 'cycle slots.' A Trojan operating for a single clock cycle can only flip bits occupying the same cycle slot. To evade detection, the Trojan must inject an error polynomial $E(x)$ that is a multiple of $F(x)$. However, any bit-shifted version of $F(x)$ contains at least two terms whose exponents differ exactly by 1 (due to the x^1 and x^0 terms), and two consecutive exponents can never share the same value modulo 8 ($n \not\equiv n+1 \pmod 8$). A valid error pattern necessarily requires flipping bits in at least two different cycle slots, which contradicts the single-cycle constraint. This proves that a single-cycle Trojan cannot trigger an undetected corruption.

The minimal CRC-colliding patterns are the shifted versions of $F(x)$, which has Hamming weight four. Mapping all its exponents $\{0, 1, 14, 16\}$ to their cycle slots via modulo-8 yields the unique target cycle slots $\{0, 1, 6\}$. A bit-shifted instance $x^k F(x)$ has exponents $\{k, k+1, k+14, k+16\}$, mapping to cycle slots $\{k \bmod 8, (k+1) \bmod 8, (k+6) \bmod 8\}$. The three intra-pattern offsets are always distinct regardless of k. No shift of $F(x)$ fits within two cycle slots. Since one- and two-cycle operations are insufficient, the lower bound on the required Trojan activity is three cycles. □

4.2. Trojan Triggering and Execution

We now describe the design of a Trojan targeting a UCIe x64 link operating in 68-byte PCIe Flit mode. Extensions to other Flit modes are presented in Section 6.3. In this mode, the 68-byte logical Flit exceeds the 64-byte physical link width. This results in data boundaries not aligning with transfer boundaries. The first transfer carries the 2-byte Flit header and 62 bytes of Flit 1 data payload, while the second carries the remaining 4 bytes of Flit 1 (including its 2-byte CRC) followed by 60 bytes of Flit 2. This 4-byte offset produces a repeating alignment pattern with a period of $68/4 = 17$ transfers.

UCIe supports idling via pause data stream (PDS) events for power saving or flow control. The transmitter terminates the active stream with a PDS header, zero-pads data to the nearest 64-byte boundary, and then transmits at least two full 64-byte transfers of zeros. Finally. It de-asserts the `valid` signal for at least 16 system clock cycles. As elaborated ahead, this behavior is central to the Trojan's triggering mechanism.

Figure 3 illustrates the complete working of the Trojan through a timing diagram. First, the Trojan monitors the *valid* signal received from the UCIe link. Using a D flip-flop, the current and previous cycle values are compared to assert `edge` on each rising edge and a `low` signal when `valid` is absent. The `low_count` increments every cycle `valid` remains low and resets when it goes high. When it reaches 16, the `PDS` signal is asserted. An AND gate combines `PDS` with `edge` to produce the `resume` signal indicating resumption of normal operation. Specifically, UCIe PHY calibration de-asserts `valid` throughout link initialization and bring-up, so `edge` never arrives until steady-state traffic is established.

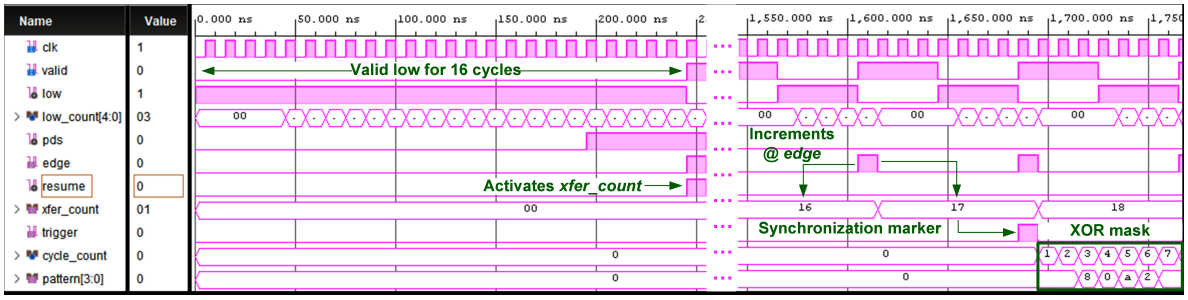

Figure 3. Timing diagram demonstrating the working of the proposed Trojan. The `valid` signal is monitored to be low for 16 cycles, which launches a counter to count transactions till they reach a multiple of 17. This becomes the trigger condition for the Trojan, which in turn generates a 4-bit mask to be XORed to the outgoing serial data in-flight, resulting in undetectable errors at the receiver side.

The `resume` signal acts as a reset for the `xfer_counter` that starts counting the number of times `edge` goes high. Each occurrence indicates a 64-byte transfer. Upon resumption, the link reset guar-

 doi:10.20944/preprints202609.1942.v1

antees that the next 64-byte transfer follows the fixed format of a 2B header and 62B data, which our Trojan uses as a synchronization marker.

As the same Flit-boundary pattern repeats after every 17 transfers, the `trigger` signal is asserted when the `xfer_count` reaches $17 \times M$, where M is an adversary-defined multiplying factor that controls attack frequency and balances impact against detection risk. In this example, we use $M = 1$ for demonstration.

Upon `trigger` assertion, the `cycle_counter` starts tracking the current bit-slot within each 8-cycle UCIe transmission interval. The Trojan receives the router's decoded 64-byte packet and a read enable signal (not shown). It uses `cycle_count` to compute the CRC-colliding XOR mask for the active cycle slot, generating a 4-bit `pattern[3:0]` corresponding to the four nonzero terms of $F(x)$ established in Theorem 1. There are four XOR gates placed in the datapath of the targeted serial lanes (not shown). The gates apply `pattern[3:0]` to the relevant portion of the 64 data lanes, producing the corrupted serial data with zero cycle latency.

5. Trojan Customization

This section explains the testbed to evaluate the effectiveness of our interposer Trojan. We explain our multi-die FPGA case study model, adversarial goals, and the adopted attack strategy to achieve them.

5.1. System Architecture

We consider a hypothetical multi-die FPGA accelerator for deep neural network (DNN) inference based on [28]. As established in previous sections, the Trojan resides in the active interposer. Figure 4 shows the system under attack. In it, a quantized INT8 DNN trained on MNIST executes sequentially across: Convolution → MaxPool and Flatten → Dense → Argmax chiplets, connected to shared memory via memory-mapped PCIe transactions over 68-byte UCIe Flits [29].

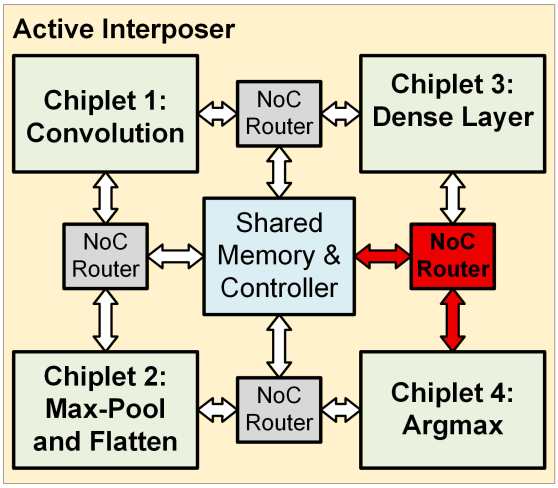

Figure 4. Chipletized deep neural network architecture consisting of four logic dies (Convolution, Max-Pool and Flatten, Dense, Argmax) connected via a NoC and shared memory. The CRC-colliding interposer Trojan, embedded in the NoC router connecting the Dense Layer chiplet to Shared Memory, intercepts and corrupts class scores (logits) before they reach the Argmax chiplet for the final decision.

The Trojan intercepts the UCIe Flit stream on the D2D link between the shared memory controller and the Argmax chiplet (Chiplet 4). This is the highest-leverage insertion point, since logit values at this stage are consumed directly by the Argmax decision with no further computations. Corrupting at other interposer links (e.g., Convolution→MaxPool) is equally feasible but requires more bit flips to propagate misclassifications through subsequent chiplets, consistent with adversarial fault-injection findings in the literature [28,30].

 doi:10.20944/preprints202609.1942.v1

We use a small-scale architecture that is hypothetical by design, but the attack generalizes to any multi-die architecture. This system can either have individual dies that host individual layers or coarser processing-element arrays that host multiple layers on one die [31]. Because all chiplets read and write through a common memory controller, logit values stored in the shared memory must cross a UCIe link to reach the Argmax chiplet. This is a structural property of shared-memory DNN accelerators. That traversal is the attack surface, regardless of how layers are partitioned across dies.

5.2. Experiment Framework

We evaluate the interposer Trojan using a hardware-software co-simulation workflow. The target DNN is trained in Python with Keras and TensorFlow. To generate realistic D2D traffic, Dense-layer memory transactions are captured and packetized into 68-byte UCIe Flits. The appended protocol headers and software-computed CRC checksums produce a complete traffic log of all D2D interactions between the shared memory controller and Chiplet 4. These Flits are replayed through an RTL UCIe TX/RX model with the Trojan inserted in the transmission path. Receiver-decoded logits are then compared with baseline software logits to quantify the impact of misclassification.

5.3. Adversarial Goals

The CRC-colliding interposer Trojan targets *controlled, class-specific* misclassification rather than generic accuracy degradation. Each 64-byte transaction layer packet within a UCIe write transaction contains 8 bytes of metadata and a 4-byte memory address, with the remaining bytes carrying the write payload [32]. The Trojan restricts all injections to the payload region exclusively for two reasons: (i) corrupting metadata produces malformed Flits that trigger receiver-side protocol-level detections, effectively neutralizing the attack; and (ii) modifying the address field redirects traffic to unauthorized memory regions, activating the access-control monitors described in Section 7.1.

The shared memory controller emits 8-bit logits quantized in INT8 sign-magnitude representation, packed sequentially into the payload of a single UCIe write transaction. This fixed packing allows the interposer Trojan to locate and corrupt specific class scores within the serial Flit stream as they traverse the link. Three adversarial objectives are evaluated:

- **Input-class suppression**: suppress a specific class so it is never predicted (e.g., preventing any image from being recognized as Class '1').
- **Forced-class promotion**: force the model to predict a target class regardless of input (e.g., classifying every image as Class '3').
- **Conditional class redirection**: selectively convert predictions from a source class to a target class (e.g., redirecting Class '1' to Class '3') without disturbing the classification of other categories.

5.4. Attack Methodology

Each Trojan instance is synthesized for a fixed adversarial objective targeting a fixed class or class pair. The interposer Trojan monitors the UCIe Flit stream in transit, decodes the payload byte corresponding to the target class logit, and selects a CRC-colliding error polynomial satisfying the bit-flip constraints derived in Section 4.1. To suppress a class, the Trojan forces the two most significant bits (MSBs) of the logit to '11', and to promote a class, it attempts to set the three MSBs to '011'.

Input-class suppression is unconditional: the Trojan monitors the target byte and selects an error polynomial that flips the current MSBs to '11', after which the XOR gates inject the pattern and minimize the class score. By contrast, *forced-class promotion* and *conditional class redirection* face stricter limitations. While the adversary ideally targets the '011' MSB pattern for promotion, the fixed bit-spacing of valid CRC-colliding error patterns restricts the available flip combinations. Figure 5 details the valid error vectors for a Class '1'→ Class '3' redirection, which requires a single error pattern that simultaneously minimizes the source class logit and maximizes the target class logit. As shown, pattern 'A' achieves this only if the current MSBs of Class '1' and Class '3' are '10' and '010', respectively; patterns 'B' and 'C' apply under the other indicated bit configurations, making these attacks *conditional* on the instantaneous payload state.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 22 September 2026 doi:10.20944/preprints202609.1942.v1

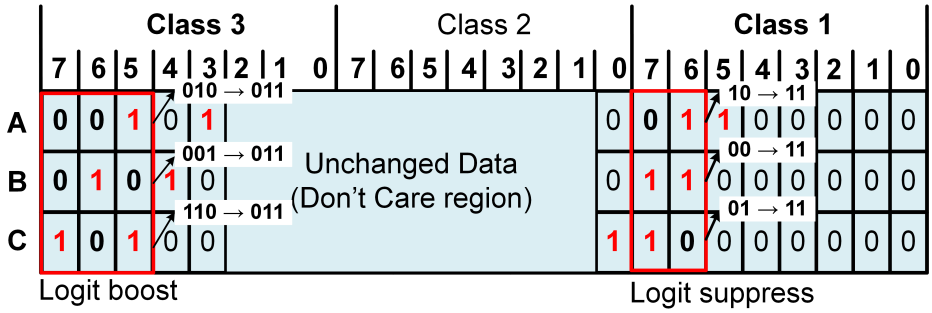

Figure 5. Error patterns for a Class '1' → Class '3' redirection. Rows A, B, and C show valid error vectors derived from fundamental polynomial $F(x)$ to bypass UCIe CRC verification. Red markers highlight bit-flips that simultaneously boost the target (Class '3') and suppress the source (Class '1'). Annotations (e.g., 010 → 011) define required existing bit values, illustrating the attack's dependency on specific D2D payload patterns.

6. Implementation Results

This section reports attack efficacy, stealth metrics, generalization to other Flit formats suggested by UCIe, and hardware overhead for our proposed Trojan.

6.1. Attack Efficacy

We ran approximately 50k inferences with the Trojan in place to measure the impact of Trojan activations on class distributions and to evaluate overall attack efficacy using **success rate**: the fraction of activations that successfully achieve the core adversarial objective. Figure 6(a) provides the baseline reference distribution for an uncompromised system.

For input-class suppression (Figure 6(b)), Class '1' predictions drop from 311 to 13, a suppression of over 95% demonstrating the highest reliability with a 100% success rate upon activation. Conversely, forced-class promotion (Figure 6(c)) raises Class '3' predictions from 312 to 1485, an approximately five-fold increase, yet succeeds in only 53.56% of activated cases, confirming the restrictive impact of the polynomial constraints discussed in Section 5.4. Finally, conditional class redirection (Figure 6(d)) pulls Class '1' down to 236 while driving Class '3' up to 399, yielding the lowest success rate of 14.14%, reflecting the scarcity of data patterns that simultaneously satisfy the suppression and promotion constraints within the valid error patterns. Crucially, the receiver accepted every corrupted packet, validating the Trojan's ability to bypass UCIe's integrity checks unconditionally.

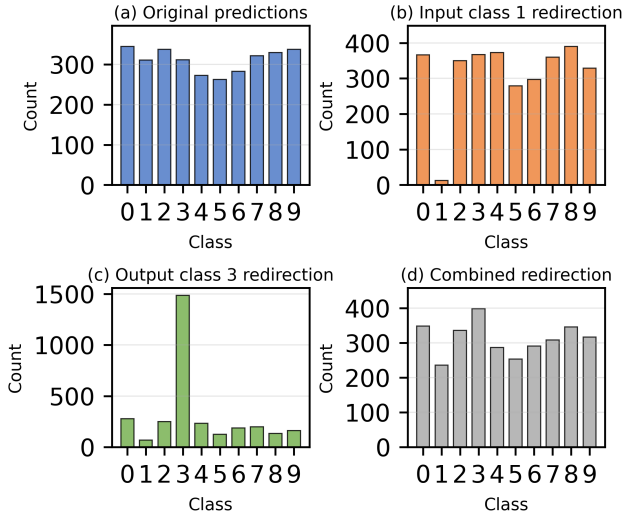

Figure 6. Prediction distributions under Trojan activation on the FPGA SiP D2D link. (a) Baseline without Trojan interference. (b) Input-class suppression: predictions suppressed for Class '1'. (c) Forced-class promotion: Class '3' frequency increased. (d) Conditional class redirection: Class '1' suppressed and Class '3' elevated simultaneously. All attacks evade UCIe CRC verification while producing controlled, class-specific misclassification.

 doi:10.20944/preprints202609.1942.v1

6.2. Trojan Stealth Analysis

We evaluate Trojan stealth at both the application and protocol levels. At the application level, we evaluate **percentage unaffected images**: the fraction of inferences that do not suffer misclassification. This serves as the primary stealth metric, with the multiplying factor M acting as the principal control knob. Figure 7(a) shows that increasing M raises the fraction of unaffected images from 94.54% to 99.31%, comparable to prior neural-network Trojans (99.58%) [30]. Notably, Figure 7(b) shows that this increased stealth comes with a hardware cost: larger values of M require wider bit-counters, raising flip-flop demand and overall resource utilization. Hence, the adversary tackles this area vs. stealthiness trade-off via M.

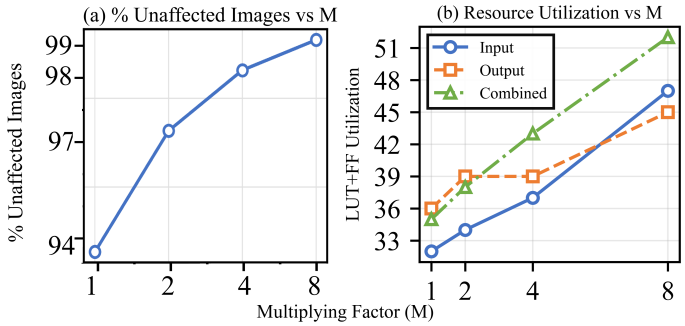

Figure 7. Effects of the multiplying factor M on Trojan behavior. (a) As M increases, the fraction of unaffected inferences rises from 94.54% to 99.31%, reducing detection likelihood. (b) Larger values of M simultaneously increase resource utilization, revealing an explicit area–stealthiness trade-off.

At the protocol level, synthetic compliance testing was carried out based on publicly available resources [33,34]. The results showed that the Trojan maintains 100% compliance across critical checks, including header validity, packet length, and address alignment. This is expected because it confines all modifications to deterministic payload XOR patterns and leaves protocol-visible metadata unchanged. In the maximal-attack setting ($M = 1$), the Trojan introduces only a 1.07% payload mismatch rate. This error rate will decrease proportionally as M increases. In the minimal-attack case ($M = 8$), the mismatch rate falls further to just 0.13%, which highlights the limited protocol-level footprint of the attack.

6.3. UCIe 3.0 and the 256-Byte Flit Mode

UCIe specification revision 3.0 advances the standard by formally extending support to higher data rates and 3D packaging while maintaining backward compatibility with prior generations [11]. A principal architectural change in this revision is the treatment of the 256-byte (256B) Flit format. While previous specification versions supported the 256B structure as an optional capability, revision 3.0 mandates it as the primary mode of operation. The proposed Trojan is not limited to the 68B Flit mode. It can also be extended to the 256B Flit mode. The reason is that the attack depends on the CRC polynomial, not on the specific Flit size. UCIe 3.0 still uses the same 16-bit CRC algorithm, so the CRC-colliding error pattern derived in Section 4.1 remains valid.

The 256B mode divides the Flit into two CRC-protected regions. CRC0 covers the first 128 bytes, and CRC1 covers the rest. Since both checks use the same polynomial, the Trojan can apply the same type of CRC-colliding corruption within either region. Moreover, the data link layer packets (DLLP) bytes that are native to PICe are driven by the D2D adapter, native to UCIe. UCIe omits the standalone LCRC present in these bytes to subsume the integrity of the entire packet.

The main aspects of Trojan redesign stem from the difference in alignment patterns in both Flit modes. In 68B mode, the relevant Flit boundary pattern repeats every 17 transfers as derived in Section 4.2. This is why the Trojan triggers every $17 \times M$ transfers. In 256B mode, one Flit is transmitted as four consecutive 64-byte transfers. Therefore, the corresponding trigger period becomes $4 \times M$ transfers.

 doi:10.20944/preprints202609.1942.v1

Moreover, the 256B mode enables corruption across both checksum boundaries within a single Flit. Hence, activating during the first and third 64-byte transfers targets the regions protected by CRC0 and CRC1, respectively, allowing both checksum checks to pass. We pick the first and third transfers as these are driven by the Protocol layer and may carry payload write data rather than the metadata fields we aim to avoid.

Thus, moving from 68B to 256B mode does not require a new payload construction. It mainly changes the trigger schedule from a 17-transfer alignment period to a 4-transfer alignment period. This supports the broader conclusion that the vulnerability comes from the algebraic predictability of the CRC, rather than from a peculiarity of the 68B Flit format.

6.4. Trojan Resource Utilization

To quantify resource utilization, we synthesized the design for a Xilinx Kintex-7 (XC7K160T) FPGA using Vivado 2025.1. Depending on M which is varied from 1 to 8, the implementation requires approximately 18–35 LUTs and 14–17 FFs for the 68B-mode Trojan. For the 256B-mode variant, we vary M from 4 to 32 for a fair comparison in terms of transactions per trigger. The implementation requires approximately 17–36 LUTs and 12–15 FFs. These numbers indicate that the algebraically derived zero-cycle latency attack methodology derived by us requires minimal logic footprint.

7. Countermeasure Analysis

This section covers the existing Trojan detection schemes and the resilience of our proposed Trojan against them. We further evaluate three potential countermeasure classes, tracing the limitations that motivate each transition.

7.1. Trojan Resilience Against Existing Detection Schemes

We now discuss potential resilience against representative detection approaches [35]. Under our threat model, standard destructive, non-destructive, and runtime techniques each encounter certain obstacles. Techniques such as scanning-electron-microscopy-based netlist reconstruction or side-channel power and timing analysis rely on a verified golden reference. These would not work here because the adversary manufactures the active interposer, making reliable golden references difficult to obtain [36,37]. Furthermore, since the Trojan targets payload data rather than memory-access permissions, it is designed to bypass policy-based monitors that do not inspect internal data integrity [38]. A similar limitation applies to delay-based physically unclonable function (PUF) schemes: the Trojan operates with zero-cycle latency and therefore introduces no timing deviations that such methods are intended to detect [39]. Taken together, these properties make the CRC-colliding interposer Trojan difficult to expose with standard detection mechanisms in the absence of stronger trust anchors or direct payload-integrity checks.

Verification-based techniques target malicious logic during pre-silicon validation [40]. Yet universal standards for chiplet security remain absent [9]. Typical validation prioritizes protocol compliance over exhaustive payload testing. Specifically, the lack of systematic methods for deriving identification tests tailored to the unique interfaces, partitioning, and interactions of chiplet architectures [41].

7.2. Countermeasure Analysis for the Proposed Trojan

C1 — Wider Polynomial (CRC-32). The first proposal extends the check from 16 to 32 bits using the CRC-32/ISO-HDLC polynomial, the same used in PCIe LCRC [42]. This provides a detection guarantee of up to 4 bits, compared to 3 for its CRC-16 counterpart. It reduces the probability of undetected random errors from 2^{-16} to 2^{-32} at the cost of approximately 4 extra bytes per Flit and $\approx 2\times$ the CRC logic area. However, *C1* does not address the root cause. The collision-causing error patterns for the new polynomial are still pre-computable offline, and the Trojan simply re-derives a valid collision vector for the wider polynomial before deployment. *C1* hardens the link against random errors, but offers no resistance to a polynomial-aware adversary.

 doi:10.20944/preprints202609.1942.v1

C2 — Keyed CRC. The fundamental weakness of *C1* is the absence of a secret. Krawczyk shows that a CRC can be converted into a keyed authentication function by using a shared secret to select the active polynomial from a parameterized family [43]. This makes the effective integrity check unpredictable to any party that does not hold the key. This secret can be negotiated at link bring-up. With an appropriately sized key, the probability of undetected random errors falls to 2^{-28} for any party without the key, at the cost of two additional bytes per CRC field.

A sophisticated interposer Trojan may defeat *C2* by intercepting and storing the key during the bring-up handshake, which traverses the same D2D link the Trojan monitors. With the key in hand, the Trojan can identify the active polynomial and compute a valid collision using the same linear algebra. The secret selects which polynomial is used, but does not change the fact that the colliding pattern is still analytically computable. *C2* therefore reduces to key secrecy, and the Trojan may obtain that knowledge by spoofing at bring-up.

C3 — Cryptographic MAC (SipHash-2-4). The weakness of *C2* is that CRC linearity is preserved regardless of polynomial selection: once the Trojan obtains the key, forgery reduces to a closed-form algebraic computation. A non-linear MAC removes this property entirely. SipHash-2-4 is a representative keyed hash built from 64-bit add-rotate-XOR (ARX) operations [44]. A single 64-bit tag replaces all CRC fields, which adds between $32 - 48$ bits of bandwidth overhead. The probability of undetected random errors is reduced to 2^{-64} without the key. A synthesized FPGA implementation of SipHash-2-4 on a Xilinx UltraScale+ device requires 2355 LUTs and 1395 flip-flops [45].

Defeating *C3* requires one of two approaches. The first is to capture the session key and recompute a valid tag over every corrupted Flit in real time. This demands embedding the full MAC datapath on the interposer and executing it at link rate. The second is to corrupt the payload and find, by trial and error, which additional bit flips produce the same tag output without knowing the key. This leads to a search over 2^{64} candidates with no analytical shortcut. Cryptographic integrity checks of this class therefore force the adversary into a position where no lightweight attack path exists, regardless of whether the session key is compromised.

8. Conclusions

This work shows that the die-to-die interconnect, overlooked by prior FPGA Trojan research focused on the programmable fabric, is an exploitable attack surface. We demonstrate that UCIe's CRC, the sole integrity check on the mainband data path, is algebraically predictable and therefore cannot defend against a protocol-aware interposer adversary. Our CRC-colliding interposer Trojan exploits this property with four bit-flips and zero added latency, inducing targeted misclassifications in an FPGA AI inference engine at a footprint of 18 to 35 LUTs and 14 to 17 FFs on a Kintex-7 FPGA. The same attack carries over to the 256-byte Flit mode mandated by UCIe 3.0, which shows that the weakness is inherent to predictable linear integrity codes rather than specific to one Flit format. Our countermeasure analysis indicates that neither a wider polynomial nor a keyed CRC closes this gap, since both leave the Trojan free to derive a collision offline. A cryptographic MAC removes this freedom by forcing a full tag recomputation per Flit at link rate, or a 2^{64} brute-force search over tag-preserving bit patterns. Neither of which should be ideally feasible for a Trojan operating under real-time link constraints. We therefore argue that securing UCIe-based FPGA heterogeneous systems against supply-chain adversaries calls for revisiting the integrity model at the D2D Adapter layer, moving beyond error-detection codes.

References

1. Chen, W.; Bottoms, W. Heterogeneous integration roadmap. In Proceedings of the 2017 International Conference on Electronics Packaging (ICEP). IEEE, 2017, pp. 302–305.
2. Lau, J.H. Recent advances and trends in heterogeneous integrations. *Journal of Microelectronics and Electronic Packaging* **2019**, *16*.
3. Dorsey, P.; et al. Xilinx stacked silicon interconnect technology delivers breakthrough FPGA capacity, bandwidth, and power efficiency. *Xilinx White Paper: Virtex-7 FPGAs* **2010**, pp. 1–10.

 doi:10.20944/preprints202609.1942.v1

4. Narashiman, S.; Joshi, D.; Sridhar, D.; Rajesh, H.; Sattva, S.; Manjunath, V.; et al. Chiplets on wheels: Review paper on holistic chiplet solutions for autonomous vehicles. *arXiv preprint arXiv:2406.00182* **2024**.
5. Calzada, P.E.; Sami, M.S.U.I.; Azar, K.Z.; Rahman, F.; Farahmandi, F.; Tehranipoor, M. Heterogeneous integration supply chain integrity through blockchain and CHSM. *ACM Transactions on Design Automation of Electronic Systems* **2023**, *29*, 1–25.
6. Wang, J.; Guo, S.; Chen, Z.; Zhang, T. A benchmark suite of hardware Trojans for on-chip networks. *IEEE Access* **2019**, *7*, 102002–102009.
7. Sharma, D.D.; Pasdast, G.; Qian, Z.; Aygun, K. Universal chiplet interconnect express (UCIe): An open industry standard for innovations with chiplets at package level. *IEEE Transactions on Components, Packaging and Manufacturing Technology* **2022**, *12*, 1423–1431.
8. QuickLogic Corporation. QuickLogic and YorChip Partner to Develop Low-Power, Low-Cost UCIe FPGA Chiplets. Press Release.
9. Vashistha, N.; Rahman, M.L.; Haque, M.S.U.; Uddin, A.; Sami, M.S.U.I.; Shuo, A.M.; Calzada, P.; Farahmandi, F.; et al. ToSHI-Towards Secure Heterogeneous Integration: Security Risks, Threat Assessment, and Assurance. *Cryptology ePrint Archive* **2022**.
10. UCIe Consortium. Universal Chiplet Interconnect Express (UCIe) Specification. https://www.uciexpress.org/specifications, 2022–2024.
11. Universal Chiplet Interconnect Express, Inc. *Universal Chiplet Interconnect Express (UCIe) Specification*, revision 3.0, version 1.0 ed., 2025.
12. Mal-Sarkar, S.; Krishna, A.; Ghosh, A.; Bhunia, S. Hardware Trojan attacks in FPGA devices: Threat analysis and effective counter measures. In Proceedings of the Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI, 2014, pp. 287–292.
13. Chakraborty, R.S.; Saha, I.; Palchaudhuri, A.; Naik, G.K. Hardware Trojan Insertion by Direct Modification of FPGA Configuration Bitstream. *IEEE Design & Test* **2013**, *30*, 45–54. https://doi.org/10.1109/MDT.2013.2247460.
14. Ender, M.; Swierczynski, P.; Wallat, S.; Wilhelm, M.; Knopp, P.M.; Paar, C. Insights into the Mind of a Trojan Designer: The Challenge to Integrate a Trojan into the Bitstream. In Proceedings of the Proc. 24th Asia and South Pacific Design Automation Conference (ASP-DAC), Tokyo, Japan, 2019; pp. 1–8. https://doi.org/10.1145/3287624.3288742.
15. Krieg, C.; Wolf, C.; Jantsch, A. Malicious LUT: A Stealthy FPGA Trojan Injected and Triggered by the Design Flow. In Proceedings of the Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Austin, TX, USA, 2016; pp. 1–8. https://doi.org/10.1145/2966986.2967054.
16. Ahmed, Q.A.; Wiersema, T.; Platzner, M. Malicious Routing: Circumventing Bitstream-level Verification for FPGAs. In Proceedings of the Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE), 2021, pp. 1–6. https://doi.org/10.23919/DATE51398.2021.9474026.
17. Ahmed, Q.A.; Wiersema, T.; Platzner', M. Post-configuration Activation of Hardware Trojans in FPGAs. *Journal of Hardware and Systems Security* **2024**, *8*, 79–93. https://doi.org/10.1007/s41635-024-00147-5.
18. Johnson, A.P.; Chakraborty, R.S.; Mukhopadhyay, D. A Novel Attack on a FPGA based True Random Number Generator. In Proceedings of the Proc. Workshop on Embedded Systems Security (WESS), co-located with ESWEEK, Amsterdam, Netherlands, 2015; pp. 1–8. https://doi.org/10.1145/2818362.2818368.
19. Suzano, J.; Abouzeid, F.; Di Natale, G.; Philippe, A.; Roche, P. On hardware security and trust for chiplet-based 2.5D and 3D ICs: Challenges and Innovations. *IEEE Access* **2024**, *12*, 29778–29794.
20. Chacon, G.A.; Williams, C.; Knechtel, J.; Sinanoglu, O.; Gratz, P.V.; Soteriou, V. Coherence attacks and countermeasures in interposer-based chiplet systems. *ACM Transactions on Architecture and Code Optimization* **2024**, *21*, 1–25.
21. Zhang, Z.; Yu, Q. Modeling hardware trojans in 3D ICs. In Proceedings of the 2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). IEEE, 2019, pp. 483–488.
22. Cadence System Analysis. (2023) Overview of Active Interposers. Cadence Design Systems. Accessed: March 3, 2026. [Online]. Available: https://resources.system-analysis.cadence.com/blog/overview-of-active-interposers.
23. Beaumont, M.R.; Hopkins, B.D.; Newby, T. Hardware Trojans - Prevention, Detection, Countermeasures (A Literature Review). 2011.

 doi:10.20944/preprints202609.1942.v1

24. Coudrain, P.; Charbonnier, J.; Garnier, A.; Vivet, P.; Vélard, R.; Vinci, A.; Ponthenier, F.; Farcy, A.; Segaud, R.; Chausse, P.; et al. Active interposer technology for chiplet-based advanced 3D system architectures. In Proceedings of the 2019 IEEE 69th Electronic Components and Technology Conference (ECTC). IEEE, 2019, pp. 569–578.
25. Chacon, G.A.; Williams, C.; Knechtel, J.; Sinanoglu, O.; Gratz, P.V. Hardware trojan threats to cache coherence in modern 2.5D chiplet systems. *IEEE Computer Architecture Letters* **2022**, pp. 133–136.
26. Park, H.; Kim, J.; Chekuri, V.C.K.; Dolatsara, M.A.; Nabeel, M.; Bojesomo, A.; Patnaik, S.; Sinanoglu, O.; Swaminathan, M.; Mukhopadhyay, S.; et al. Design flow for active interposer-based 2.5-D ICs and study of RISC-V architecture with secure NoC. *IEEE Transactions on Components, Packaging and Manufacturing Technology* **2020**, *10*, 2047–2060.
27. Schmidt, T. CRC Generating and Checking. Application Note AN730, Microchip Technology Inc., 2000.
28. Malik, A.A.; Aydin, F.; Aysu, A. Precision Strike: Targeted Misclassification of Accelerated CNNs with a Single Clock Glitch. In Proceedings of the 2025 IEEE Physical Assurance and Inspection of Electronics (PAINE). IEEE, 2025, pp. 1–7.
29. Sharma, D.D.; Choudhary, S.; Onufryk, P.; Pelt, R. On-Package Memory with Universal Chiplet Interconnect Express (UCIe): A Low Power, High Bandwidth, Low Latency and Low Cost Approach. In Proceedings of the 2025 IEEE Symposium on High-Performance Interconnects (HOTI). IEEE, 2025, pp. 87–96.
30. Clements, J.; Lao, Y. Hardware trojan design on neural networks. In Proceedings of the 2019 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2019, pp. 1–5.
31. Shao, Y.S.; Clemons, J.; Venkatesan, R.; Zimmer, B.; Fojtik, M.; Jiang, N.; Keller, B.; Klinefelter, A.; et al. SIMBA: Scaling deep-learning inference with multi-chip-module-based architecture. In Proceedings of the IEEE/ACM international symposium on microarchitecture, 2019, pp. 14–27.
32. PCI-SIG. *PCI Express 2.0 Base Specification*. PCI-SIG, 2006. Revision 0.9.
33. Haque, A. Test Plan : PCIe Transaction Layer (TL). LinkedIn Pulse, 2024. Accessed: 2025-03-07.
34. crusader2000. pcie_common.sv. GitHub repository: PCIE-Transaction-Layer-Verification, 2024. Accessed: 2025-03-07.
35. Chakraborty, R.S.; Narasimhan, S.; Bhunia, S. Hardware Trojan: Threats and emerging solutions. In Proceedings of the 2009 IEEE International high level design validation and test workshop. IEEE, 2009, pp. 166–171.
36. Courbon, F.; Loubet-Moundi, P.; Fournier, J.J.; Tria, A. A high efficiency hardware trojan detection technique based on fast SEM imaging. In Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2015, pp. 788–793.
37. Zhang, J.; Su, G.; Liu, Y.; Wei, L.; Yuan, F.; Bai, G.; Xu, Q. On Trojan Side Channel Design and Identification. In Proceedings of the IEEE/ACM International Conference on Computer-Aided Design. IEEE, 2014, pp. 278–285.
38. Nabeel, M.; Ashraf, M.; Patnaik, S.; Soteriou, V.; Sinanoglu, O.; Knechtel, J. 2.5 D root of trust: Secure system-level integration of untrusted chiplets. *IEEE Transactions on Computers* **2020**, *69*, 1611–1625.
39. Deric, A.; Holcomb, D. Know time to die–integrity checking for zero trust chiplet-based systems using between-die delay PUFs. *IACR Transactions on Cryptographic Hardware and Embedded Systems* **2022**, pp. 391–412.
40. Bhunia, S.; Hsiao, M.S.; Banga, M.; Narasimhan, S. Hardware Trojan attacks: Threat analysis and countermeasures. *Proceedings of the IEEE* **2014**, *102*, 1229–1247.
41. Sami, M.S.U.I.; Zhang, T.; Shuvo, A.M.; Haque, M.S.U.; Calzada, P.E.; Azar, K.Z.; et al. Advancing trustworthiness in system-in-package: A novel root-of-trust hardware security module for heterogeneous integration. *IEEE Access* **2024**, *12*, 48081–48107.
42. Koopman, P. 32-Bit Cyclic Redundancy Codes for Internet Applications. In Proceedings of the Proc. International Conference on Dependable Systems and Networks (DSN), 2002, pp. 459–468. https://doi.org/10.1109/DSN.2002.1028931.
43. Krawczyk, H. LFSR-based Hashing and Authentication. In Proceedings of the Advances in Cryptology — CRYPTO 1994. Springer, 1994, Vol. 839, *Lecture Notes in Computer Science*, pp. 129–139. https://doi.org/10.1007/3-540-48658-5_15.

 doi:10.20944/preprints202609.1942.v1

44. Aumasson, J.P.; Bernstein, D.J. SipHash: A Fast Short-Input PRF. In Proceedings of the Progress in Cryptology — INDOCRYPT 2012. Springer, 2012, Vol. 7668, *Lecture Notes in Computer Science*, pp. 489–508. https://doi.org/10.1007/978-3-642-34931-7_29.
45. Welte, B.; Zambreno, J. An FPGA implementation of SipHash. In Proceedings of the 2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). IEEE, 2023, pp. 63–70.