Hardware Trojan Threats to Cache Coherence in Modern 2.5D Chiplet Systems

Gino A. Chacon, Charles Williams, Johann Knechtel, Ozgur Sinanoglu, and Paul V. Gratz

Abstract—As industry moves toward chiplet-based designs, the insertion of hardware Trojans poses a significant threat to the security of these systems. These systems rely heavily on cache coherence for coherent data communication, making coherence an attractive target. Critically, unlike prior work, which focuses only on malicious packet modifications, a Trojan attack that exploits coherence can modify data in memory that was never touched and is not owned by the chiplet which contains the Trojan. Further, the Trojan need not even be physically between the victim and the memory controller to attack the victim’s memory transactions. Here, we explore the fundamental attack vectors possible in chiplet-based systems and provide an example Trojan implementation capable of directly modifying victim data in memory. This work aims to highlight the need for developing mechanisms that can protect and secure the coherence scheme from these forms of attacks.

Index Terms—Hardware security, cache coherence, hardware trojans, 2.5D integration, trusted computing, chiplets

1 INTRODUCTION

Computing systems are moving toward 2.5D designs that source various hard IPs, called chiplets, from multiple vendors and integrate them using an interposer. Industry has demonstrated that 2.5D designs lower manufacturing costs, enabling further scaling post-Moore’s Law [1]. Future 2.5D designs may leverage standards such as Compute Express Link [2] to interoperate via a shared memory system. While 2.5D designs provide many benefits, we show they also increase the risk of Trojan attacks, specifically targeting the coherence system. Here, we demonstrate several novel Trojans attacking cache coherence in 2.5D designs. We illustrate the risks for these systems and hope to excite the architecture community to address these risks. Though we focus on 2.5D integrated systems, note that these attacks also apply to general cache-coherent systems integrating closed-source or hard IP blocks from various vendors.

Hardware Trojans, or Trojans for short [3], are a threat in which an attacker infiltrates some level of the design or fabrication process to insert malicious circuitry into a design. Trojans can cause disastrous system failures via confidentiality, integrity, and/or availability violations. Prior work has shown that Trojans can leak data from memory [4], disrupt cryptographic security features [5], and induce denial-of-service attacks [6]. As industry moves towards 2.5D designs integrating multiple vendor chiplets, specific chiplets used in building these systems may be untrustedworthy. Even if the IP vendor is trustworthy, the manufacturing process may not be, leading to infiltration and the insertion of Trojans.

In 2.5D designs, memory coherence is crucial to allow each component and chiplet to maintain an up-to-date view of the system’s memory. We identify this system as an ideal target for Trojan attacks as coherence mechanisms control how all components communicate data updates. Existing coherence schemes do not enforce existing virtual/physical memory permissions, thus, a Trojan connected to the coherence scheme can directly manipulate any memory region in the full system regardless of memory permissions or physical location. Unlike prior packet-level NoC attacks, Trojans on cache coherence do not need to be physically on the path between the victim and the memory controller to launch effective attacks. Despite this attractiveness, there is a lack of works deeply exploring coherence exploits and their defense in 2.5D systems or otherwise.

Here, we propose several new Trojan attacks that leverage the coherence system protocol to maliciously manipulate the victim process’ memory. We first describe a set of new fundamental attacks that a Trojan can mount on coherence systems, passive reading, masquerading, modifying, and diverting attacks, according to Basak et al.'s taxonomy [7]. Here we assume an attacker implements these coherence system attacks in hardware through compromised design or manufacturing. While each of these attacks individually violates a system’s security, we further show that adversaries can orchestrate them together to perform complex Forging attacks that modify any process’ memory. These purely hardware attacks cannot be thwarted by contemporary software defense mechanisms since all exploited coherence interactions are transparent to software and legal within the coherence protocol. Further, no prior work considers such attacks on coherence systems, neither in the context of 2.5D systems with chiplets nor traditional 2D systems.

Contributions. This work provides new insights into how Trojans can manipulate coherence systems to violate the security of a chiplet system. We present a simulated example of a substantial attack that can directly manipulate memory in an address space other than that of the compromised chiplet. This work makes the following contributions:

- We present potential attack stages available to a Trojan designers exploiting coherence systems.
- We demonstrate how to use these fundamental stages to orchestrate a complex Trojan attack in a chiplet-based system.
- We provide suggestions for future work on hardening modern chiplet designs.

2 BACKGROUND

2.1 Hardware Trojans

Hardware Trojans are malicious hardware inserted by an attacker during a device’s design or manufacturing process. Chiplet-based devices have a complex multistage design flow that can be compromised at many levels, especially as multiple 3rd party vendors emerge to provide chiplets from separate foundries and design teams. This design flow makes verification much more difficult and expensive than traditional System-on-Chip (SoC) manufacturing.

Detecting Trojans is challenging as chiplet-based systems contain multiple complex IPs sourced from various vendors. Testing a chiplet’s functionality may occur during the manufacturing or integration stage, which requires a reference model or device [3]. However, if the 3rd party IP’s source is untrustworthy, the reference model itself may incorporate the Trojan, or the IP may camouflage the Trojan as a correct implementation. Attackers can conceal a Trojan by only allowing it to trigger under specific conditions. For example, the Trojan we describe in Section 3.2 only activates when it observes references to a specific address. These properties make the Trojan difficult to detect by simply testing the functionality of
the chiplet. For this work, we assume that an attacker infiltrates some stage within the design or manufacturing process to target the system's coherence mechanisms.

Prior art focuses on infiltrating the NoC of a target design to cause deadlocks [6], leak information [4], or disrupt security features [5]. However, NoC-based attacks require the Trojan to directly attack NoC packets, limiting the Trojan to only packets traversing a particular path in the NoC. Prior attacks would not work in a 2.5D integrated system because attacking chiplets are not on the path between victim chiplets and the memory controller.

2.2 Coherence Protocols

Multi-processor systems incorporate cache coherence protocols to ensure data coherency across processors’ private caches. The coherence system has a complete vantage point and control over the memory system. All communication between cores and main memory follows the coherence protocol, making it an ideal target for a Trojan co-located with a processor’s private caches. At this location, a Trojan can snoop on coherence messages produced by other processors, manipulate those messages, or generate messages without incurring exceptions and remaining invisible to software running on the system. Further, coherence schemes do not enforce virtual/physical memory permissions, thus any Trojan connected to the coherence scheme can directly manipulate any memory region in the full system, without regard to memory permissions or physical location.

We target the MOESI Hammer coherence protocol, a hybrid broadcast-directory system used in many AMD processors [8]. Though our focus is MOESI Hammer, our attack scenarios can easily be ported to other coherence schemes. In schemes without broadcast, the Trojan simply needs to register “S” or “X” state with the directory to ensure that update requests on the given line are seen.

3 Trojans Targeting Coherence Systems

Here we propose a new set of “basic” Trojan attacks on the coherence scheme that follows the general, not coherence specific, taxonomy for Trojan attacks by Basak et al. [7]: passive reading, masquerading, modifying, and diverting. These basic attacks can adversely affect the system but cannot provide an attacker with control the memory system. We then propose a novel, more sophisticated and powerful “Forging” attack to modify data belonging to another core, even on a different chiplet than that holding the Trojan.

Target System. We demonstrate our attacks on a 64-core processor with eight chiplets, eight cores per chiplet, based on the Rocket-64 architecture [9]. Each core has private L1 and L2 caches with a unified cache controller. An NoC connects each chiplet and four memory controllers that maintain a portion of the global state directory. The cache controllers generate coherence messages that are injected into the NoC as network packets.

3.1 Basic Trojan Attacks on the Coherence Systems

Fig. 1 illustrates our proposed basic coherence attacks. We assume a Trojan is placed at a core’s cache controller and can intercept coherence messages from the network interface ahead of the state directory. While these attacks target the MOESI Hammer hybrid protocol, the basic principles of the attacks, are consistent with any coherence protocol. Passive Reading (Fig. 1a). Trojans passively reading (snooping) observe incoming coherence messages from the chiplet’s network-on-chip (NoC) sub-system as they reach the L2’s state directory. The Trojan may buffer messages, identify specific request patterns, and facilitate a covert communication channel. The Trojan does not affect the system’s state but may trigger a more complex Trojan.

Masquerading (Fig. 1b). Masquerading (spoofing) occurs when a Trojan modifies the packet’s sender field such that the packet appears to originate from a different core. If the target packet is a request, such an attack can result in a deadlock since all responses from the directory or other cores are sent to the incorrect core. If the target packet is a response, the Trojan may block it and respond with a different core ID, requesting core proceeds, leaving local caches incoherent. Diverting (Fig. 1d). Trojans can launch diverting attacks by blocking the local state directory from observing a request and then resending the request with a different destination field. This results in the compromised core and the original requester becoming incoherent with respect to the rest of the memory system.

Limitations of Basic Attacks. Any of the above attacks can individually result in incoherence or deadlocks but cannot directly manipulate a victim’s data. Only combining these attacks allows for a more complex set of attack vectors that would enable a Trojan to pose a significant security threat.

3.2 Trojan Design

In the remainder of this paper, we propose the Forging Attack, a novel attack that manipulates legal coherence transactions to allow a Trojan to write to a target address in a different process operating in a different chiplet. The compromised chiplet containing the
The data is successfully modified in the data array. The Trojan immediately sends a response to the directory to indicate that it is "evicting" modified data. The directory forwards the data to all cores in the network, waiting to receive all acknowledgments (ACKs) before updating the data. Once all ACKs are received, the directory sends the final ACK, allowing the Trojan to proceed with "evicting" the maliciously changed dirty data.

The directory responds with an ACK response containing the malicious data. The Trojan blocks the local directory from seeing any response from the directory or cores, waiting to receive all ACKs. Once all ACKs are received, the Trojan can access the data, since the directory considers the compromised core as the data owner.

Phase 2, Writing Malicious Data. Once access permissions are acquired, the global directory assumes that the Trojan’s core is the exclusive owner of the data. Fig. 2(bottom) illustrates Phase 2 of the attack. This phase allows the Trojan to mimic the legal operations that enable writing to main memory as if the core was evicting the target address. The steps of the attack are as follows: Once the Trojan receives the final ACK, the requests to the target address are unblocked. The Trojan immediately sends a PUTX to the directory to indicate that it is “evicting” modified data. The directory responds with a WRITEBACK_ACKNOWLEDGEMENT, allowing the Trojan to proceed with “evicting” the maliciously changed dirty data. The Trojan responds to the WRITEBACK_ACKNOWLEDGEMENT with a WRITEBACK_EXCLUSIVE_DIRTY response containing the malicious data. The data is written to memory.

3.4 Results

We evaluate the Trojans in gen5, targeting a victim which iterates over an array to set each value to ‘1’ or ‘0’ and then reads the array to compute a sum. Fig. 3a shows the data the victim process observes without the Trojan enabled. The victim writes ‘0’ or ‘1’ to various locations in its data array and then re-reads these locations, seeing the expected data values. Fig. 3b shows the data the victim receives when it attempts to read the data array after modifying it. The steps of the attack are as follows: Once the Trojan receives the final ACK, the requests to the target address are unblocked. The Trojan immediately sends a PUTX to the directory to indicate that it is “evicting” modified data. The directory responds with a WRITEBACK_ACKNOWLEDGEMENT, allowing the Trojan to proceed with “evicting” the maliciously changed dirty data. The Trojan responds to the WRITEBACK_ACKNOWLEDGEMENT with a WRITEBACK_EXCLUSIVE_DIRTY response containing the malicious data. The data is written to memory.

Unlike prior work, which focuses on Trojans modifying packets [10], [11], [12], [13], we leverage the coherence mechanism itself to modify data in memory that is never touched and not owned by the chiplet containing the Trojan. Our attack does not require the data to be in the compromised core’s caches. Generating and blocking specific coherence messages allows the Trojan to mislead the global directory about the state and ownership of the targeted data.
CONCLUSION AND FUTURE DEFENSES

As industry moves toward chiplet-based designs, hardware Trojans pose a significant threat to security. These systems will rely heavily on coherence to ensure that data remains up-to-date in all components, making the coherence protocol an attractive target. Critically, unlike prior work, which focuses only on packet modifications, we show that a coherence-centric Trojan attack can modify memory that is not even owned by the compromised chiplet. We provide an example of a complex Trojan implementation that modifies memory without relying on malicious software components. This work highlights the need for mechanisms to protect the coherence scheme from these novel attacks.

Detecting Trojans during chiplet manufacturing is challenging considering the complexity of individual IPs. Defenses against hardware Trojan exploiting a system’s coherence mechanisms could implement runtime monitoring to identify malicious behavior originating from a particular chiplet. A benefit of 2.5D integration is that the components are usually sourced from vendors and then integrated onto an interposer layer at a separate foundry than each IP’s manufacturing [9]. Requiring an interposer’s manufacturing and integration by a trustworthy facility could allow the 2.5D interposer to act as a hardware root of trust that can embed security features. Embedding the security features into the interposer layer could allow defenders to observe coherence packets and securely control data flow freely. We plan to explore these themes in our future work.

REFERENCES

[1] S. Naffziger et al., “Pioneering chiplet technology and design for the AMD EPYC™ and Ryzen™ processor families: Industrial product,” in Proc. ACM/IEEE 48th Annu. Int. Symp. Comput. Archit., 2021, pp. 57–70.

[2] Compute Express Link (CXL), 2022. Accessed: May 18, 2022. [Online]. Available: www.computeexpresslink.org

[3] S. Bhunia and M. Tehranipoor, The Hardware Trojan War. Berlin, Germany: Springer, 2018.

[4] M. N. I. Khan, A. De, and S. Ghosh, “Cache-out: Leaking cache memory using hardware Trojan,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 28, no. 6, pp. 1461–1470, Jun. 2020.

[5] M.-M. Bidmeshki, G. R. Reddy, L. Zhou, J. Rajendran, and Y. Makris, “Hardware-based attacks to compromise the cryptographic security of an election system,” in Proc. IEEE 34th Int. Conf. Comput. Des., 2016, pp. 153–156.

[6] M. Kim, S. Kong, B. Hong, L. Xu, W. Shi, and T. Suh, “Evaluating coherence-exploiting hardware Trojan,” in Proc. Des. Automat. Test Europe Conf. Exhib., 2017, pp. 157–162.

[7] A. Basak, S. Bhunia, T. Tkacik, and S. Ray, “Security assurance for system-on-chip designs with untrusted IPs,” IEEE Trans. Inf. Forensics Secur., vol. 12, no. 7, pp. 1515–1528, Jul. 2017.

[8] P. Conway, N. Kalyanasundaram, G. Donley, K. Lepak, and B. Hughes, “Cache hierarchy and memory subsystem of the AMD Opteron processor,” IEEE Micro, vol. 30, no. 2, pp. 16–29, Mar./Apr. 2010.

[9] J. Kim et al., “Architecture, chip, and package co-design flow for 2.5D IC design enabling heterogeneous IP reuse,” in Proc. 56th ACM/IEEE Des. Automat. Conf., 2019, pp. 1–6.

[10] D. M. Ancajas, K. Chakraborty, and S. Roy, “Fort-NoCs: Mitigating the threat of a compromised NoC,” in Proc. 51st ACM/EDAC/IEEE Des. Automat. Conf., 2014, pp. 1–6.

[11] T. Boraten and A. K. Kodi, “Mitigation of denial of service attack with hardware Trojans in NoC architectures,” in Proc. IEEE Int. Parallel Distrib. Process. Symp., 2016, pp. 1091–1100.

[12] M. H. Khan, R. Gupta, J. Jose, and S. Nandi, “Dead flit attack on NoC by hardware Trojan and its impact analysis,” in Proc. 14th Int. Workshop Netw. Chip Archit., 2021, pp. 10–15.

[13] N. Prasad, R. Karmakar, S. Chattopadhyay, and I. Chakrabarti, “Runtime mitigation of illegal packet request attacks in networks-on-chip,” in Proc. IEEE Int. Symp. Circuits Syst., 2017, pp. 1–4.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.