Evaluating Coherence-exploiting Hardware Trojan

Minsu Kim†, Sunhee Kong†, Boeui Hong†, Lei Xu‡, Weidong Shi‡, and Taeweon Suh†*
† Department of Computer Science and Engineering, Korea University, Seoul, Republic of Korea
Email: {koggiri1990, nickong, boyhong, suhtw}@korea.ac.kr
‡ Department of Computer Science, University of Houston, Houston, TX, USA
Email: {lxu13, wshi3}@central.uh.edu

***Abstract*—Increasing complexity of integrated circuits and IP-based hardware designs have created the risk of hardware Trojans. This paper introduces a new type of threat, a coherence-exploiting hardware Trojan. This Trojan can be maliciously implanted in master components in a system, and continuously injects memory transactions onto the main interconnect. The injected traffic forces the eviction of cache lines, taking advantage of cache coherence protocols. This type of Trojans insidiously slows down the system performance, incurring Denial-of-Service (DoS) attack. We used a Xilinx Zynq-7000 device to implement the Trojan and evaluate its severity. Experiments revealed that the system performance can be severely degraded as much as 258% with the Trojan. A countermeasure to annihilate the Trojan attack is proposed in detail. We also found that AXI version 3.0 supports a seemingly irrelevant invalidation protocol through ACP, opening a door for the potential Trojan attack.**

Keywords*—Hardware Trojan; DoS attack; Cache Coherence; AXI Protocol; Zynq-7000

I. Introduction

IC chips typically go through several steps before being finally introduced to the market: Planning, Design, Manufacturing, and Validation. In the design phase, IC chip vendors utilize diverse approaches such as Virtual platform and Intellectual Property (IP)-based design, to improve competitiveness with shortened time-to-market (TTM). Especially, the IP-based design methodology is nowadays an irrevocable choice and trend for TTM. However, it makes IC chips vulnerable to the malicious hardware insertion and/or modification, namely referred to as Hardware Trojan (HT). If IP cores are supplied by the untrusted or even trusted third-party IP vendors, HT can insidiously be embedded in the IP, be incorporated in the final design, and be shipped to customers. The Manufacturing step is also considered untrustworthy as the number of companies outsourcing the fabrication facility is increasing due to the high cost of constructing and maintaining the fab. Once the design layout is handed to a foundry company, it is exposed to an environment where HT may be obliviously embedded. The planning and validation steps may be considered trusty because these steps are usually carried out by themselves [1]. However, if evil insiders exist, the whole process can be compromised as well.

Compared to the other hardware components in computer or smartphone systems, CPUs are much more complex. For example, the Intel's latest processor, Skylake, boasts 1.75 billion transistors in a single chip [2]. The number of engineers in a design team easily exceeds a few hundreds. Thus, it would not be feasible to audit the work from every single engineer as long as the product functions as intended. It indicates that hardware components are not free from malicious modifications and/or insertions via evil insiders. Even worse, there are some speculations that some products are intentionally modified to include a kill switch [3]. Some backdoor in CPU requires only 1341 additional gates [4]. Unlike traditional CPU companies such as Intel and AMD, some companies such as ARM license soft IP cores to hundreds, if not thousands, of partner companies. The licensed soft cores are integrated and synthesized with many other IPs for the SoC design. It could make soft IPs more vulnerable to the HT insertion.

Nowadays, an increasing number of CPU companies provide cache coherent interface for hardware accelerators. The closer and direct cooperation between CPU and hardware accelerator provides significant advantages in performance and energy consumption [5]. ARM supports two coherent interfaces: Accelerator Coherency Port (ACP) and AXI Coherency Extension (ACE). The ACP provides a unidirectional interface from hardware accelerator to CPU, whereas the ACE provides a bidirectional interface between hardware accelerator and CPU with the snoop connections. Intel and AMD also provide proprietary coherent interfaces: Intel QuickPath Interconnect (QPI) and AMD HyperTransport. Those have been mostly used for homogeneous multi-processor configuration. More recently though, soon after acquiring Altera, Intel announced that they will also provide a coherence interface using QPI for the XEON+FPGA configuration [6]. While the open coherence interfaces provide good opportunities for the tuning application's performance and energy consumption, they also open a door for the evil HT addition.

This paper focuses on a new type of threat, a coherence-exploiting hardware Trojan incurring the Denial-of-Service (DoS) attack. More specifically, the Trojan injects the malicious traffic onto the main interconnect. It does not alter the system state and simply slows down the system performance, taking advantage of cache coherence protocols. We implemented and evaluated the coherence-exploiting hardware Trojan using Xilinx Zynq-7000 on Zedboard. This paper demonstrates the severity of the Trojan attack with benchmark programs running on μC/OS-III. The countermeasures to mitigate or completely annihilate the Trojan attack are also proposed in detail. While researching on this type of Trojan, we have surprisingly found that AXI 3.0 supports a seemingly irrelevant invalidation protocol through ACP, opening a door for the potential coherence-exploiting Trojan attack.

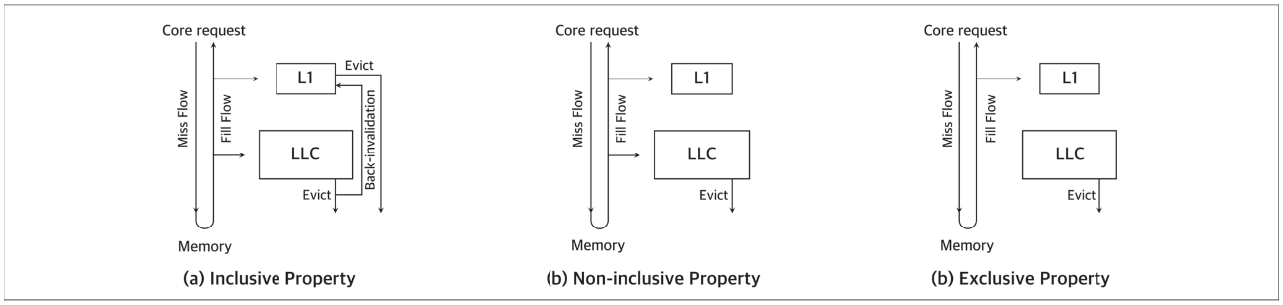


Fig. 1. Cache behaviors according to caching properties [14]

The paper is organized as follows: Section II summarizes related work. Section III presents the theoretical background of coherence-exploiting hardware Trojan. Section IV describes the experiment environment. Section V presents the experiment outcome and its implications. Countermeasures are discussed in Section VI, and Section VII summarizes and concludes our paper.

II. Related Work

A. Software based cache attack

Cache side-channel attacks exploiting cache coherence mechanism are an emerging topic. Yarom et al. [7] presented the cache timing attack exploiting different access latencies between L1 cache and Last Level cache (LLC) to extract the private encryption keys from victim program on the Intel processors. This attack takes advantage of the inclusive property of LLC to exploit back-invalidation. Lipp et al [8] proposed a new cache timing attack on ARM-based smartphones. This attack also utilizes different access latencies between remote cores and DRAM to monitor not only tap and swipe events but also cache activities in the secure world from the normal world in ARM TrustZone. Recently, Allan et al. [9] used software-based cache invalidation to slow down an encryption process for increasing the amount and quality of information gained in the cache timing attack.

B. Hardware trojan

Hardware Trojan has emerged as a major security concern not only in the industry but also in the government [10]. The reason why it has attracted attention is its covertness and powerfulness. A suspected actual case is the failure of Syrian radar [3], where the state-of-the-art Syrian military radar could not detect an enemy bombing due to the radar's temporary malfunction. It was suspected that a backdoor implanted in commercial off-the-shelf microprocessors in the Syrian raider was triggered during the attack. In a recent article published by New York Times [11], the US National Security Agency (NSA) has implanted HT in nearly 100,000 computers around the world, targeting not only Russian military networks but also trade institutions inside the European Union. The HT working with malware allows US to conduct surveillance on these computers.

Due to such threats, researchers have been actively investigating HT and its defense techniques. Wang et al. [12] classified hardware Trojan into three categories according to their physical, activation, and action characteristics. King et al. [4] designed and implemented the Illinois Malicious Processor to demonstrate that small malicious modifications can compromise the security of the entire computer system. Jin et al. [13] demonstrated several hardware Trojans in RTL level to examine the possibility of designing hardware Trojans that can evade state-of-the-art detection methodologies. Various hardware Trojan detection techniques have also been proposed. Typical HT detection techniques rely on side-channel signal analysis. HTs generally affect some chip signatures such as path delay, power consumption and performance. Side-channel signal analysis tries to detect HTs by comparing with trusted golden ICs for side-channel signatures [1] even though the ICs used in the research are commonly small in size.

TABLE I. Caching properties of renowned Processors

	Cache levels		
	L1 Cache	***L2 Cache***	***L3 Cache***
Intel Sandy bridge [15]	32KB	Non-inclusive (256KB)	Inclusive (1MB to 20MB)
AMD Shanghai [16]	64KB	Exclusive (512KB)	Non-inclusive (6MB to 12MB)
ARM Cortex-A9 [17]	32KB	Non-inclusive or Exclusive (128KB to 8MB)	N/A
ARM Cortex-A53 [17]	32KB	Non-inclusive - Data Inclusive – Instruction (128KB to 2MB)	N/A
ARM Cortex-A57 [17]	32KB	Inclusive - Data Non-inclusive – Instruction (128KB to 2MB)	N/A

III. Theory of operations

Modern processors employ multi-level caches in memory hierarchy. L1 caches, typically small and private, are divided into instruction and data caches. LLC typically large and shared among all the cores, is unified for instruction and data. There are three caching properties in multi-level caches: Inclusive, Exclusive, and Non-Inclusive. Fig. 1 illustrates the multi-level cache behaviors upon cache miss and upon eviction, depending on the caching properties. The property indicates whether the contents in the high level cache (for example, L1) should be a subset of the low level cache (for example, LLC). If a memory block in the high level cache is also present in the low level, then the low level cache is called an inclusive cache. It implies that, if a specific line in the low level cache is evicted, the same line in the high level if present is evicted as well. It is called the back-invalidation. The exclusion property guarantees that a specific memory block resides either in the high level or in the low level cache, not in both. If the low level cache is neither inclusive nor exclusive, it is called non-inclusive. TABLE I presents the commodity processors' caching properties and sizes. The caches with inclusive property are simple to design in multi-core configuration because only LLC should snoop bus transactions for coherence. However, the overall cache capacity is equal to the size of LLC. With the exclusive property, the cache capacity is equal to L1+LLC. However, both L1 and LLC should snoop bus transactions, incurring the design complexity. Generally, the choice of the property is dependent on the ratio of the size of the high level cache to the low level cache [14].

The coherence-exploiting hardware Trojan can be embedded inside master components inside computer or smartphone systems. CPUs or hardware accelerators interfaced with the system bus or main interconnects are the target places for the Trojan insertion. This type of Trojan can adversely influence the cache behavior to slow down the system performance, resulting in DoS attack. We introduce 2 types of attack models.

A. *Trojan model #1*

The Trojan model #1 exploits the back-invalidation property of an inclusive cache to slow down the system performance. Fig. 2 illustrates its operation. A hardware Trojan embedded inside a hardware accelerator is located at the same level as CPU core, and it shares an LLC with the CPU. The HT in the hardware accelerator continuously injects read transactions targeting a particular set of LLC through the main interconnect (Fig. 2. ①). Then, if a LLC miss happens, a memory block is brought into the LLC from memory, and the LLC should evict a cache line to make a room for the fresh memory block (Fig. 2. ②). As a cache line is evicted from the LLC, the same line in the L1 cache if present should be evicted as well, to satisfy the inclusion property (Fig. 2. ③). If applications running on CPU require the evicted cache line, it should be brought back from main memory. This type of Trojan is extremely stealthy because it does not alter anything in computer systems and simply slows down the system performance by elongating the memory access latency. This Trojan model works more badly against inclusive caches, compared to exclusive or non-inclusive caches because exclusive cache allows data to be placed only in one cache. That is, even though HT injects a read transaction to LLC evicting an LLC cache line, it is not able to touch the L1 cache content.

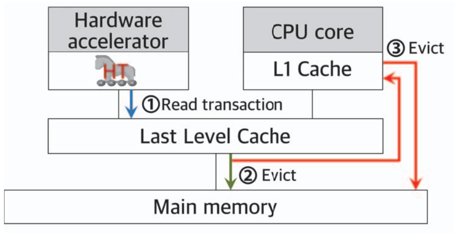


Fig. 2. Trojan model #1: Injection of malicious read transaction (Inclusive cache property)

B. *Trojan model #2*

Unlike the Trojan model #1, the Trojan model #2 injects write invalidation transactions to the main interconnect. While a malicious write transaction (not invalidation) changes the system state and most likely results in the system crash, the write invalidation transaction causes the caches to simply invalidate the corresponding cache line. Fig. 3 illustrates Trojan model #2. The HT in the hardware accelerator injects the write invalidation transaction targeting a specific cache line (Fig. 3. ①). Since it informs that a data has been updated on the hardware accelerator side, the corresponding cache line in L1 cache if present is invalidated even with the exclusive cache property (Fig. 3. ②). It also only changes the cache state but does not alter the system state. In the shared bus architecture, the write invalidation transaction is typically implemented with read-for-ownership. The similar protocol is implemented in the point-to-point interconnection architecture such as the AXI protocol. In this paper, we used a Zynq-7000 FPGA for experiments, which is equipped with Cortex-A9 CPU and ACP interface. The write invalidation transaction is injected to the LLC via the ACP interface. As detailed in Section IV C, we have found that AXI 3.0 supports a seemingly irrelevant invalidation protocol through the ACP, opening a door for the potential coherence-exploiting Trojan attack through the malicious write transaction.

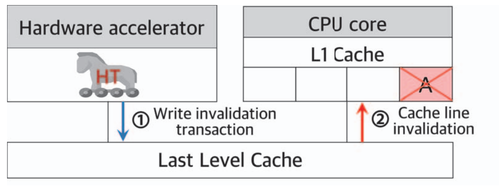


Fig. 3. Trojan model #2: Injection of malicious write invalidation transaction (Exclusive cache property)

IV. EXPERIMENT

A. *Experiment Setup*

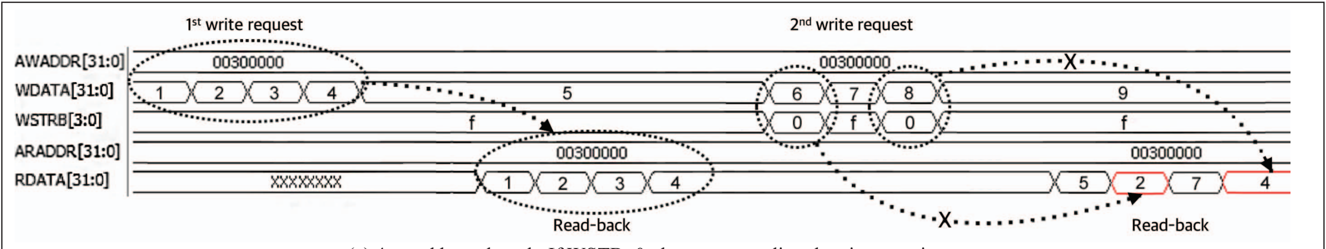
We used the Zynq-7000 FPGA on Zedboard [18] to implement and evaluate the coherence exploiting hardware Trojan. Zynq-7000 is based on Xilinx All Programmable SoC architecture. It is composed of two major blocks: Processing System (PS) with a dual-core ARM Cortex-A9 MPcore, and the typical FPGA fabric called programmable logic (PL). The Xilinx Vivado 2015.4 was used for synthesizing and place&routing the hardware Trojan, which was developed using Verilog-HDL. To measure the performance impact, we ported μC/OS-III on Zynq-7000 and used three benchmark programs: Sobel filter, K-means and Fast Fourier Transform (FFT) running on μC/OS-III. The Xilinx SDK was used for compiling and downloading benchmark programs to the main memory on Zedboard. To collect cache-related statistics, the performance counters in performance monitor unit inside Cortex-A9 were used. The operating clock frequencies of the major components on Zedboard are summarized in Table II.

TABLE II. OPERATING CLOCK FREQUENCIES

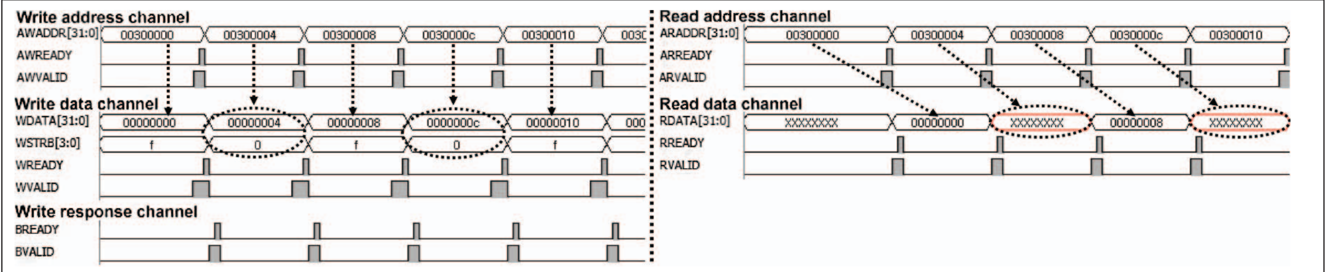
	Hardware components on Zedboard		
	PS (Cortex-A9s)	***PL***	***Memory (DDR3)***
Frequency	667MHz	100MHz	533MHz

B. *Zynq-7000 Coherence Architecture*

Fig. 4 shows the Zynq-7000 block diagram focusing on CPUs, memory hierarchy, and cache coherence. Zynq-7000 has two Cortex-A9s. Each processor has separate 32KB L1 instruction and data caches and provides two 64-bit AXI master interfaces for independent instruction and data accesses to the Snoop Control Unit (SCU). This interface includes the snoop signals to maintain coherence between the L1 data caches via SCU. The cache coherence protocol is MESI. Since AXI is based on point-to-point interconnection instead of shared-bus, SCU keeps records of information stored in the L1 data cache with Tag RAM for coherence support. The Tag RAM acts as a local directory that includes information about coherent cache



(a) 4-word burst length: If WSTB=0, the corresponding data is not written



(b) 1-word burst length: if WSTB=0, the corresponding data is not written

Fig. 5. Write burst transaction and Read burst transaction

lines held in the L1 data caches. The two Cortex-A9s share the unified 512KB L2 cache for instruction and data. The Zynq-7000 also provides the coherence interface called Accelerated Coherence Port (ACP). The ACP allows a master component located in the PL to share data coherently with caches in the PS section. Via the ACP interface, the master component in the PL can read the most up-to-date data in the PS section whether the data is located in L1 data cache, L2 cache or main memory. Similarly, a write transaction from ACP to PS causes SCU to look up the Tag RAM to maintain coherence with the PS section. ACP does not provide the snoop interface for the master in PL to monitor transactions on ACP slave, meaning that ACP is based on the unidirectional connection from a master component in PL to PS. The Zedboard is equipped with a 512MB DDR3 memory.

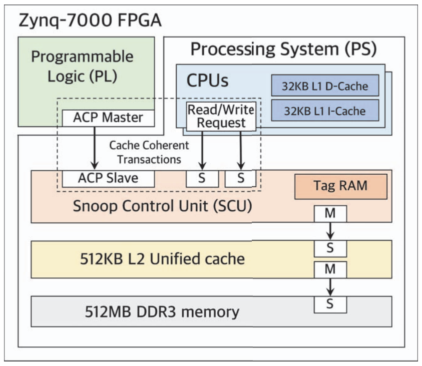


Fig. 4. Zynq-7000 block diagram focusing on memory hierarchy and cache coherence

C. Write Invalidation Injection (Trojan model #2) via ACP

The AXI protocol is lengthy and complex. We explain only relevant signals and protocols for the understanding of this paper. We implemented the Trojan model #2 since the caches in Zynq-7000 with Cortex-A9 support non-inclusive or exclusive property. The AXI protocol [17] has five independent channels for read and write: read address, read data, write address, write data and write response channels. All transaction channels use the *VALID* and *READY* signals for handshaking. The source first asserts the *VALID* to indicate when data is available. Then, the destination asserts the *READY* to indicate its readiness. The data transfer occurs only when both the *VALID* and *READY* signals are asserted (HIGH). A write transaction utilizes three channels in the following order: write address, write data, and then write response channels. A read transaction utilizes two channels in the following order: read address and read data channels.

The AXI supports several auxiliary signals for special purposes. For example, the 4-bit *AWCACHE* signal indicates the detailed caching policy. *AWCACHE* = 4'b1111 denotes that data will be cached with write-back and write-allocate policy. The *WSTRB[n:0]* specifies valid byte lanes of the data bus; Each bit in *WSTRB[n:0]* indicates the validity of each 8-bit in the data bus. Thus, *WSTRB[n]* corresponds to an 8-bit data in *WDATA[(8n)+7: (8n)]*. Fig. 5 illustrates the role of WSTRB signal in the burst transaction. The data size is 32-bit, meaning that *WSTRB[3:0]* is required. The master component generates several write requests with data and then read-backs from the same location. Fig. 5. (a) shows the waveform with the burst length set to 4 words. The master component writes data with all 4-strobe bits asserted (*WSTRB*=4'1111), meaning that all 32-bit data are valid. A read-back from the same location confirms the correctness of the previous write transaction. The next write transaction alternates *WSTRB* with 4'1111 and 4'0000. The read-back confirms that only the 1st and 3rd data are correctly stored in memory.

A write transaction with *WSTRB* set to zero is a legitimate transaction in burst mode as illustrated in the 2nd write burst transaction (alternating writes) of Fig. 5. (a). However, if a burst length is set to a 1-word and if *WSTRB[3:0]*=4'b0000, it implies the write transaction is simply for invalidating a cache line in the other cache for the corresponding address. In a typical multi-core configuration with the snoop-capability, it can be perceived as invalidating the cache line for coherence since the master is about to write data to its local cache. Then, the written data should be able to be supplied to other cache upon a snoop request. Nevertheless, ACP does not allow snoop from other cache (or CPU) since ACP is the unidirectional interconnect from PL to

PS, as mentioned in Section IV B. Therefore, the 1-word burst write with invalid data must be an illegal transaction in ACP. However, surprisingly enough, we have found that the transaction is operated the same way, as shown in Fig. 5. (b). It opens a door for contriving Trojans and injecting malicious invalidation traffic. Once the transaction is injected, SCU takes an action to simply invalidate the corresponding cache line either in L1 or L2 cache. It does not alter the system state and simply slows down the system performance, resulting in DoS attack. We exploit this weakness to contrive the coherence-exploiting Trojan. In summary, the Trojan injects write transactions with *AWCACHE*=4'b1111, the burst length=1, and *WSTRB*=0.

D. Trojan Implementation and Validation

The Trojan model #2 was implemented with Verilog-HDL. To inject write invalidation transactions targeting a specific address or a memory region according the ACP protocol, a simple FSM was used for the design. Xilinx Vivado 2015.4 was used to synthesize the design, and the implemented Trojan reports the resource utilization summarized in Table III. For the inspection of write invalidation protocol via ACP and correctness checkup, we have deployed Xilinx's Integrated Logic Analyzer (ILA) to PL and tapped into the AXI interconnect. The waveforms of AXI signals were captured while the system is running with injecting the Trojan. The correct operations depicted in Fig. 5 were observed with both a 4-word burst write transaction and a 1-word write invalidation transaction.

TABLE III. RESOURCE UTILIZATION OF IMPLEMENTED TROJAN

	PL resources in Zynq-7000		
	LUTs	***LUTRAMs***	***FFs***
Utilization (#)	426	64	463

V. TROJAN SEVERITY EVALUATION AND ANALYSIS

The Trojan generates memory transactions with physical address via ACP. The experiments were performed with two scenarios; In the first case, the Trojan injects write transactions only to one hot cache line where applications access intensively. In the second case, the Trojan injects write transactions to a particular region of memory ranging from 32B to 8KB. The 32B indicates that one cache line is accessed continuously because the cache line size is 32B in Cortex-A9. The 128B indicates that four cache lines are accessed in a round-robin manner. We have executed three benchmark programs 10 times and report the averages in all experiments. The baseline is the case with no Trojan. We have checked the correctness by comparing outputs after running applications with and without Trojan.

Fig. 6 shows the miss rates of the L1 data cache with and without Trojan for the case #1. Since the data set of the three benchmark programs almost fit into the cache, the miss rates without the Trojan are negligible. However, with Trojan implanted in ACP, the miss rates in all benchmarks significantly increase regardless of the caching property. Among the benchmarks, the FFT exhibits the smallest increase in miss rate since its temporal locality is comparably low. The increases in miss rates of all benchmarks have led to the performance degradation. Fig. 7 shows the ratio of execution times for the case #1, compared to the baseline. The execution times were dramatically increased across all benchmark programs with the Trojan. Especially the system performance in K-means with exclusion property was degraded as much as 258%, compared to the baseline. Fig. 8 shows the ratios of execution times for the case #2, compared to the baseline. The case #2 targets a particular region of memory, varying its size from 32B to 8KB. As the region size increases, the performance degradation tends to be relieved, yet as severe as at least 100% degradation, as seen in 8KB region with FFT. It means that it would be more an effective attack if the Trojan targets a hot cache line instead of a wider range of memory region.

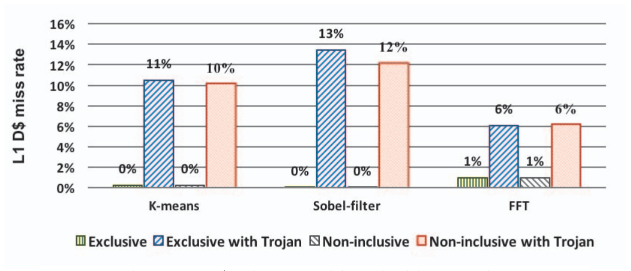


Fig. 6. L1 D$ miss rate with and without Trojan

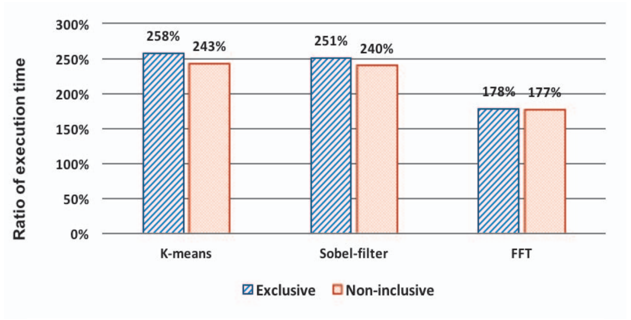


Fig. 7. Ratio of execution times with Trojan over baseline (Case #1)

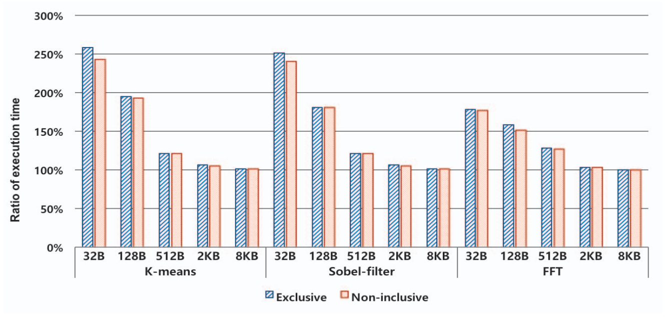


Fig. 8. Ratio of execution times with Trojan over baseline (Case #2)

The Trojan injection frequency is also a major factor influencing its severity on performance. FPGAs provide great environment for prototyping and evaluation. Nevertheless, due to its inherent nature, FPGA-based systems run much slower than actual systems. In our experiments, PL runs at 100MHz. Thus, the Trojan injection was synchronized with the same 100MHz via ACP. As the injection frequency increases, the Trojan's impact will be much more severe since cache lines are invalidated at a much faster rate. One more factor to consider in evaluation is the difference in memory wall in FPGA systems, compared with actual computer/smartphone systems. In FPGA-based systems, the performance disparity between CPU and main memory is relatively small, while the opposite is true for actual systems. For example, in Zynq-7000-based Zedboard, Cortex-A9 operates at 667MHz while the main memory runs at 533MHz, as shown in Table II. Thus, taking into account the actual memory wall, this type of Trojan would influence more significantly in the actual computer or smartphone systems. Caches with inclusion property are more prone to the Trojan attack since naive and harmless read transactions from coherence port can insidiously pull down active cache lines from caches, resulting in severe slow-down of system performance.

VI. Countermeasures

The coherence-exploiting Trojans are extremely insidious and hard to detect once deployed. However, it is feasible to mitigate or completely avoid the attack. First, the mitigation can be achieved by adding more dirty bits per a cache line and SCU Tag RAM in the Zynq-7000 case. Since the L1 data cache and SCU Tag RAM in Znyq-7000 have one dirty bit per a cache line, one invalidation transaction nullifies one whole cache line. However, if multi-dirty bits per a cache line are implemented, only a particular sub-block is invalidated per an invalidation, mitigating the Trojan's impact on performance. Nevertheless, it would not be satisfactory especially in real-time and mission-critical applications such as military equipment.

The complete removal of the Trojan impact can be achieved with collaboration of hardware and software. Fig. 9 depicts the system architecture, where a pair of memory-mapped filtering registers exists in SCU. The key point is to place the filtering registers in an IP block separate from the 3rd party hardware IP. In the Zynq-7000 case, the filtering registers located in SCU are separated from the 3rd party IP located in PL. Thus, evil hardware IP developers are oblivious to the existence of the filtering registers. The filtering registers should be programmed with the start and end addresses for active coherence operation. The memory area specified with the registers are eligible for coherence support (Fig. 9. ⓐ). Coherence transactions targeting other areas are filtered out (Fig. 9. ⓑ).

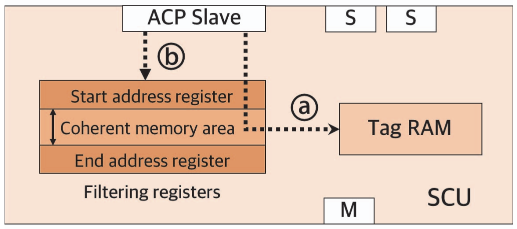


Fig. 9. Countermeasure block diagram to annihilate the Trojan attack: Filtering registers should be programmed for coherence operation

VII. Conclusions

This paper studied the coherence-exploiting hardware Trojan, which continuously injects memory transactions onto the main interconnect. The experiment results show that the injected traffic causes a significant performance degradation as it increases the miss rate in L1 data cache. This type of attack can be made extremely insidious as the invalidation write transactions do not alter the system state and simply slows down the system performance, thus most likely resulting in DoS attack. The system with snoop interface and inclusive caches could more easily be exploited. We have discussed the countermeasures to mitigate or completely remove the impact of this attack. The complete removal of the Trojan impact can be achieved with placing memory-mapped filtering registers in an IP separate from the 3rd party IP, and programming the registers to make it active. We have also found that AXI 3.0 provides a seemingly irrelevant write invalidation protocol in the ACP interface, opening a door for the insidious Trojan implantation. Other proprietary coherence interface standards such as ARM's AXI ACE, Intel's QPI and AMD's HyperTransport could be potential targets for the exploitation, as IP-based design is nowadays a norm and the hardware accelerator support via system interconnect becomes gaining popularity.

Acknowledgment

This work was supported by NATO Science for Peace and Security Programme (G4919).

References

[1] M. Tehranipoor and F. Koushanfar, "A survey of hardware Trojan taxonomy and detection," IEEE Des. Test Comput., pp. 10–25, 2010.

[2] "7th Generation Intel® Core™ i7 Processors", Intel, 2015. [Online]. Available: https://www-ssl.intel.com/content/www/us/en/processors/core/core-i7-processor.html.

[3] Sally Adee, "The Hunt for the Kill Switch". IEEE Spectrum. Vol. 45, Issue 5, pp 34-39, May 2008.

[4] S.T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou, "Designing and Implementing Malicious Hardware," in Proceedings of the First USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET '08), April 2008.

[5] M. Sadri, C. Weis, N. When, and L. Benini, "Energy and performance exploration of accelerator coherency port using xilinx ZYNQ," presented at the 10th FPGAWorld Conf., Copenhagen, Denmark, Sep. 2013.

[6] "Using OpenCL for FPGAs and Preview of Xeon+FPGA architecture", Using OpenCL for FPGAs and Preview of Xeon+FPGA architecture, 2016. [Online]. Available: https://cpufpga.wordpress.com/.

[7] Yuval Yarom and Katrina Falkner. FLUSH+RELOAD: A high resolution, low noise, L3 cache side-channel attack. In USENIX Security Symposium, 2014.

[8] Lipp, Moritz, et al. "ARMageddon: Cache Attacks on Mobile Devices." *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association.

[9] T. Allan, B. B. Brumley, K. Falkner, J. van de Pol, and Y. Yarom, "Amplifying side channels through performance degradation," In Annual Computer Security Applications Conference, Los Angeles, CA, US, December 2016.

[10] "Trusted Integrated Circuits (TRUST)", Darpa.mil, 2007. [Online]. Available: http://www.darpa.mil/program/trusted-integrated-circuits.

[11] D. Shanker, "N.S.A. Devises Radio Pathway Into Computers", Nytimes.com, 2014. [Online]. Available: http://www.nytimes.com/2014/01/15/us/nsa-effort-pries-open-computers-not-connected-to-internet.html.

[12] X. Wang, M. Tehranipoor, and J. Plusquellic, ''Detecting Malicious Inclusions in Secure Hardware: Challenges and Solutions,'' Proc. IEEE Int'l Workshop Hardware-Oriented Security and Trust (HOST 08), IEEE CS Press, 2008, pages. 15-19.

[13] Y. Jin, N. Kupp, and Y. Makris, "Experiences in hardware Trojan design and implementation," in Proc. IEEE Int. Workshop Hardware-Oriented Security Trust (HOST), Jul. 2009, pp. 50–57.

[14] A. Jaleel, E. Borch, M. Bhandaru, S. C. Steely Jr., and J. Emer. Achieving non-inclusive cache performance with inclusive caches: Temporal locality aware (tla) cache management policies. In MICRO-43, MICRO '43, 2010.

[15] "Intel® 64 and IA-32 Architectures Optimization Reference Manual", Intel. [Online]. Available: http://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-optimization-manual.html.

[16] D. Hackenberg et al. Comparing Cache Architectures and Coherency Protocols on x86-64 Multicore SMP Systems. In MICRO, pages 413– 422. IEEE, 2009.

[17] "ARM Information Center", Infocenter.arm.com. [Online]. Available: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0488f/index.html.

[18] "Zynq-7000 All Programmable SoC Technical Reference Manual,", 2015. [Online]. Available: http://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf.