

Article

A Survey of of Side-Channel Attacks and Mitigation for Processor Interconnects

Jie Yuan ^{1,2}, Jing Zhang ^{1,2}, Pengfei Qiu ^{1,2,*}, Xinghai Wei ^{1,2}  and Dongxiao Liu ^{1,2} 

¹ School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China; yuanjie@bupt.edu.cn (J.Y.); wuweimo@bupt.edu.cn (J.Z.); junjuntvt@bupt.edu.cn (X.W.); liudongxiao@bupt.edu.cn (D.L.)

² Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education, Beijing 100876, China

* Correspondence: qp@bupt.edu.cn

Abstract: With advancements in chip technology, the number of cores in modern commercial processors continues to rise, leading to increased complexity in interconnects and on-chip networks. This complexity, however, exposes significant security vulnerabilities, primarily in the form of side-channel and covert-channel exploits. Unlike other microarchitectural side-channel attacks, those leveraging on-chip interconnects utilize unique characteristics, allowing attackers to develop novel methods that can bypass existing effective defenses. In this paper, we present a comprehensive survey of current side-channel and covert-channel attacks based on processor-on-chip interconnects. We categorize these attacks into three types: contention-based, distance-based, and layout-based, according to the specific interconnect characteristics they exploit, and discuss corresponding countermeasures for each. Finally, we provide an outlook on future development trends in processor interconnect side channels. This survey is the first to specifically focus on interconnect-based side channels in processors.

Keywords: processor microarchitecture; interconnect; network-on-chip; side channel; covert channel



Citation: Yuan, J.; Zhang, J.; Qiu, P.; Wei, X.; Liu, D. A Survey of Side-Channel Attacks and Mitigation for Processor Interconnects. *Appl. Sci.* **2024**, *14*, 6699. <https://doi.org/10.3390/app14156699>

Academic Editor: Andrea Prati

Received: 7 July 2024

Revised: 27 July 2024

Accepted: 27 July 2024

Published: 31 July 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In recent years, security issues in processor microarchitectures have garnered increasing attention. The discovery of the Spectre [1] and Meltdown [2] vulnerabilities in 2018 propelled discussions on computer architecture security to unprecedented levels, leading to a continual emergence of attacks and defense measures surrounding various microarchitectural components and features. Due to their ease of implementation, covert nature, and direct threat to sensitive data, software-based side-channel and covert-channel attacks have become significant directions in microarchitecture security [3]. In microarchitectural side-channel attacks, adversaries monitor the activities of victim programs by observing the side effects produced when they share hardware resources, thus inferring the victim's secrets. These attacks can be leveraged to leak encryption keys [4–7], web browsing activities [8,9], user keystrokes [10,11], and other sensitive information [12,13]. In some typical microarchitectural attacks, covert channels can complement speculative execution and other microarchitectural features to completely bypass memory isolation and leak arbitrary data [1,2].

Over the past two decades, many microarchitectural components have been exploited to construct microarchitectural attacks [11], including branch predictors [1], caches [14,15], prefetchers [16,17], memory management units [18,19], among others. In response, many practical measures to mitigate microarchitectural side-channel issues have been implemented, such as prohibiting programs from different security domains from being scheduled on the same physical core, partitioning shared caches spatially, or flushing cache contents during context switches.

However, on the one hand, these defensive measures inevitably incur additional performance overhead, while on the other hand, there are numerous undocumented components within the microarchitecture that may provide opportunities for side-channel attacks, but not all aspects have been sufficiently discussed [3]. In recent years, on-chip interconnect side-channel attacks have begun to attract attention. These interconnects are used to connect various resources on the chip, including physical cores, LLC, DRAM controllers, and I/O ports. On-chip interconnects are highly accessible; specifically, most accesses to the last-level cache and all accesses to DRAM must go through the on-chip interconnect. Therefore, restricting access to on-chip interconnects, similar to caches, is very challenging, implying a high correlation between on-chip interconnects and microarchitectural security. However, for a long time, especially since the emergence of side-channel attacks based on ring buses and mesh interconnects [20–22], there has been little discussion about defensive measures against on-chip interconnects.

Since the advent of large-scale integrated circuit technology, people have been exploring ways to integrate more and smaller electronic components into one. Chips, as one of the most important technological products and application forms, have also seen their manufacturing processes shrink. The most advanced chip manufacturing process has reached 3 nm [23]. However, this is believed to be close to the physical limit, with factors such as quantum effects, tunneling effects, and electron migration posing significant challenges. While a further reduction in manufacturing process sizes may be limited in the near future, the pursuit of higher-performance processors has not ceased. Instead of further shrinking the manufacturing process, improvements are being made by integrating more cores, increasing chip areas, and developing more complex on-chip interconnects in multi-core processors.

The 2022 International Roadmap for Devices and Systems (IRDS) projects that lithography technology will soon reach its limits, necessitating the introduction of new 3D structures in chip design and manufacturing [24]. This shift implies larger spatial scales and more cores, along with complex interconnect structures and their corresponding optimizations. As of now, one of the latest multi-core processor architectures, Intel's Sierra Forest (Intel, Santa Clara, CA, USA), features a product with 144 cores: the Intel Xeon 6700 series, aimed at providing enhanced AI performance and handling compute-intensive workloads for high-throughput cloud servers [25]. According to Intel's official reports, the Xeon 6900 series, featuring 288 cores, is expected to be released in the first quarter of 2025 [26]. Additionally, Kalray's 256-core CPU (Kalray, Montbonnot-Saint-Martin, France), the MPPA2, is widely used in data centers to accelerate data processing.

In recent years, with the increasing complexity of on-chip systems and the rapid expansion of applications, security issues related to on-chip interconnects and Network-on-Chip have gradually received attention. Although some relatively well-known attacks have emerged, discussions on side-channel and covert-channel attacks on on-chip interconnects and in the SoC security field remain scarce and outdated, lacking a comprehensive investigation of state-of-the-art attacks and defenses.

In this survey, we focus on some of the most exploitable characteristics of processor-on-chip interconnect architectures, including contention, distance, and planar layout. Specifically, we analyze three types of side-channel attacks and corresponding countermeasures found in the existing literature:

- (1) **Contention:** Traffic generated when cores communicate with LLC or IMC located on other tiles causes interconnect congestion.
- (2) **Distance:** Signal propagation delay created by the distance between cores constructs a channel.
- (3) **Planar Layout:** Thermal radiation from heavily loaded cores to neighboring cores constructs a channel.

The subsequent sections of this paper will provide a detailed discussion of each of these exploitable characteristics. Each section will introduce the attacks within the context of on-chip interconnects, provide an overview of threat models, and discuss mitigation

strategies. We will compare and contrast various defense mechanisms, considering both their advantages and disadvantages. This survey primarily includes papers published in the past decade. Papers related to processor-on-chip interconnect or NoC security that do not focus on side-channel attacks, such as NoC denial-of-service attacks and information leakage, are not included in this survey.

The structure of this paper is as follows: Sections 2 and 3 provide an overview of on-chip interconnects and processor microarchitectural side channels, respectively. Sections 4–6 describe these attacks and their corresponding countermeasures in the context of processor-on-chip interconnects. Section 7 discusses the future development trends of interconnect side channels. Section 8 concludes this survey.

2. Interconnect and Network-on-Chip

In integrated circuits (ICs), interconnects refer to the structures that electrically connect two or more circuit elements, such as transistors. The design and layout of these interconnects are crucial for the normal operation, performance, power efficiency, reliability, and manufacturing yield of ICs. On a chip, components need to communicate with each other through these interconnects. To facilitate this communication, designers have developed on-chip communication subsystems based on network principles, commonly referred to as Network-on-Chip (NoC) in certain digital circuit categories [27].

NoC is most commonly found among modules in System-on-Chip (SoC). Modules on ICs are typically semiconductor IP cores that systematize various functionalities of computer systems and are designed to be modular in a network science sense. NoCs are packet-switched networks based on routers between SoC modules. With advancements in chip manufacturing technology, computer architects have been able to integrate an increasing number of processors and other heterogeneous components onto the same chip.

Early SoC architectures were based on buses and crossbars. In this traditional bus architecture, each signal has a dedicated line for point-to-point transmission. Early SoC had fewer heterogeneous modules, making this simple and easy-to-implement bus architecture cost-effective. Buses were utilized in many mature architectures, with typical examples including ARM's AMBA (Advanced Microcontroller Bus Architecture) bus [28] and IBM's CoreConnect [29].

In contrast, buses struggled to adapt to architectural changes or effectively leverage advancements in silicon technology, leading this architecture to become a critical bottleneck in SoC performance as the number of SoC modules and interconnect complexity increased. This is primarily because buses cannot classify activities based on their characteristics; for example, buses cannot categorize transactions, transfers, and physical layer behaviors. Additionally, bus architectures exhibit significant disadvantages in scalability, power consumption, and reusability, along with drawbacks such as unstable line delays and rapidly increasing verification costs.

In pursuit of more complex SoC, researchers, inspired by the internet, proposed the concept of Network-on-Chip as an alternative to bus architectures. NoCs are miniature versions of wide-area networks consisting of routers, packets, and links, proposed as a new solution for inter-module communication in highly complex SoC modules [30,31]. This new paradigm describes a communication method between IPs, including routing protocols, flow control, switching, arbitration, and buffering, offering higher scalability, resource reuse, and overall performance, while also reducing costs, design risks, and time to market. Leveraging these advantages, NoCs have successfully replaced buses as the solution for complex SoCs requiring scalable interconnect architectures.

Cores and non-core units inside CPUs are connected via a ring or mesh interconnect, constituting an NoC as well. Additionally, different CPUs are also connected via an interconnect known as Ultra Path Interconnect (UPI) [32]. While these interconnects offer significant advantages in terms of latency and bandwidth in multi-core CPUs, they may leak information about program memory access patterns due to time differences caused by congestion on the interconnects.

Figure 1 shows the mesh interconnect architecture of an Intel 16-core processor as an example, consisting of multiple processing elements interconnected by routers and wires (links). Processing elements can be any components such as microprocessors, specialized integrated circuits, or intellectual property blocks performing specific tasks. In the context of processors, processing elements are typically processor cores or non-core components such as integrated memory controllers, UPI controllers, I/O units, etc. Routers on the processor are commonly referred to as stops and, depending on the organization of the processor interconnect, can also be referred to as ring stops [20] or mesh stops [21,22].

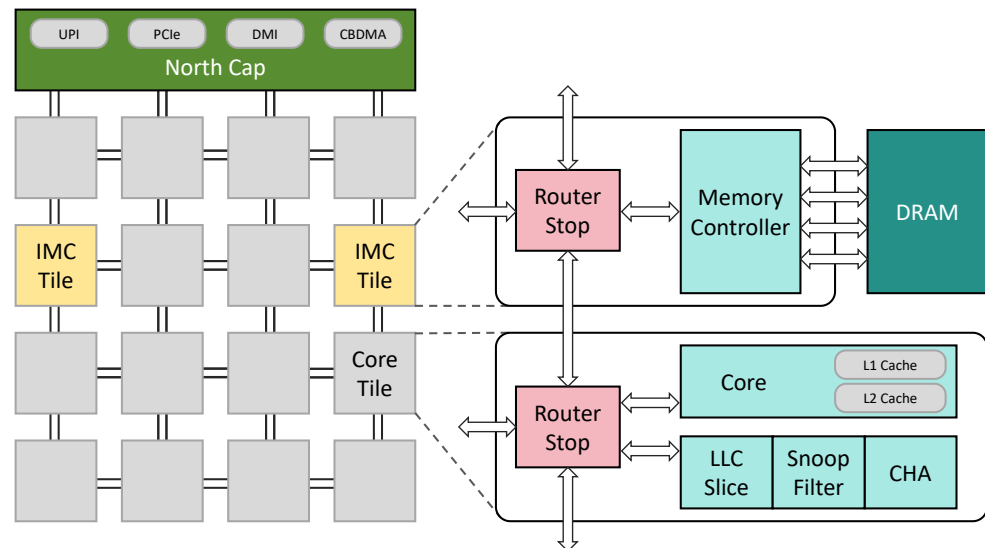


Figure 1. Mesh interconnect architecture of an Intel 16-core processor.

Nodes of the NoC composed of stops and processing elements on the processor are called tiles. Tiles on the processor are differentiated based on the processing elements they contain, such as core tiles and IMC tiles [33]. As shown in Figure 1, a core tile contains a CPU core (including L1 and L2 caches), an LLC slice, and a stop. The LLC slice comprises data, a directory (snoop filter), and a control unit called Cache/Home Agent (CHA), responsible for maintaining cache coherence among blocks. The stop handles injection, forwarding, and receiving of traffic in the interconnect. It is also known as a node router or station. An IMC tile includes a stop and an integrated memory controller, which connects to off-chip DRAM modules.

The NoC interconnect architecture employs a packet-based communication approach. Requests or responses sent to caches or off-chip memory are divided into packets, which are then transmitted and injected into the network. Flit is the smallest unit of flow control in the NoC. A packet can consist of one or more flits. For instance, when a load instruction at the core tile at position s in Figure 2 is executed, it ① first checks the private cache and ② LLC cache, and if there is a cache miss, the required data must be fetched from memory. Therefore, ③ a memory fetch request message is created and sent to the network interface (NI) over the appropriate virtual network. The NI transforms it into a network packet based on the packet format, filters the packet, and ④ sends it into the network via the local router. The network ⑤ is then responsible for routing these packets to their destination, which is the IMC tile at position d . Packets can be routed along the same path or different paths, depending on the routing protocol. The NI at position d ⑥ receives flits and creates packets based on them, then ⑦ forwards the request to the IMC, and finally ⑧ initiates the memory fetch request.

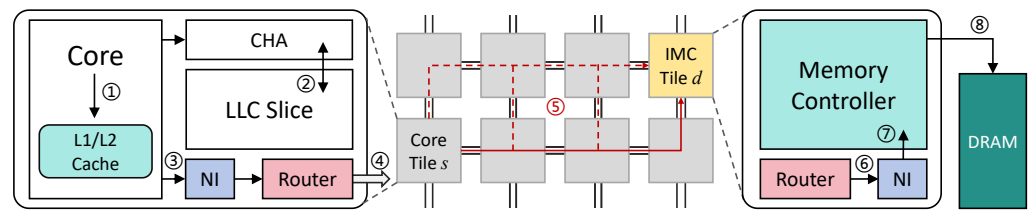


Figure 2. The control flow transmission sequence of a load instruction in an NoC.

3. Microarchitectural Side Channel

CPU microarchitectural side-channel attacks are techniques for obtaining sensitive information by monitoring and analyzing the physical implementation of computer processors. These attacks exploit side channels generated by the processor during instruction execution and memory access, such as timing, power consumption, electromagnetic radiation, etc., to infer internal operations and data of the processor. Most microarchitectural channels can be categorized into attacks based on sharing (also known as persistent state or residual state) and attacks based on contention (also known as transient state).

Sharing-based side channels, also referred to as stateful side channels, operate on the basis of sharing microarchitectural resources between attackers and victims. Typically, attackers proactively adjust and configure these resources shared with the victim, deliberately inducing the resources into a known state, and then await changes in the shared resource state resulting from the victim's normal operation. Subsequently, attackers can reevaluate the state of the shared resources to infer secrets about the victim's execution. Despite strict isolation of data and code at the software and hardware levels, side effects of victim code execution still leave traces observable by attackers from different security domains. The fundamental reason behind these attacks lies in the sharing of limited microarchitectural resources across different security domains, such as private caches [34], TLB [35], Branch Target Buffer (BTB) [36,37], and last-level caches (LLCs) [15,38].

Contention-based channels, also known as stateless side channels, also require attackers and victims to share certain microarchitectural resources, but the construction of these channels does not rely on traceable side effects of victim execution in the microarchitectural state. Instead, these attacks necessitate attackers to passively monitor the latency of their access to shared resources and infer secrets about the victim's execution based on changes in the latency. Therefore, in these attacks, the side effects of victim execution are observable by attackers only while the victim is actively executing. In contrast to stateful side channels, stateless side channels shift the focus from "sharing" resources to their "limited" availability. Typical resources used for contention-based channels include functional units [39], execution ports [40,41], cache banks [42], memory buses [43], and random number generators [44].

From the perspective of information leakage scope, attacks can be classified into preemptive scheduling attacks, Simultaneous Multithreading (SMT) attacks, and cross-core attacks. Preemptive scheduling attacks (also known as time-sharing attacks) [16,37,45] involve attackers and victims having their programs reside on the same core and executing alternately in time, with the aim of disrupting temporal boundaries. SMT attacks [34,35,39], similar to preemptive scheduling attacks, require victims and attackers to execute concurrently on different logical cores within the same physical core. Cross-core attacks [38,46,47], being the most general, involve victims and attackers running on different physical cores.

Numerous defense mechanisms against microarchitectural side channels have been proposed, primarily focusing on two main strategies: isolation and decoupling.

Isolation aims to mitigate information leakage by disrupting the sharing of resources and components within the microarchitecture that side channels depend on. For instance, attacks targeting SMT can be mitigated by disabling SMT, which is currently the most prevalent approach adopted by cloud providers and end-users [48]. Additionally, a primary strategy involves eliminating "sharing" by partitioning and isolating shared resources to

ensure attackers cannot observe the usage of victim resources [49–51]. Another mitigation technique for preemptive scheduling and SMT side channels involves refreshing the microarchitectural state during context switches to clear any potential traces left by the victim. For example, several works have proposed CPU cache refreshing during context switches [49,52,53].

Decoupling involves eliminating the correlation between secret data and the observable leaked information. For widely prevalent timing side channels, the most general mitigation strategy is the application of constant-time programming principles [54–56]. This approach ensures that the execution time of instruction sequences remains consistent, regardless of the different microarchitectural states or execution paths they may take.

However, all these mitigations effectively reverse certain microarchitectural optimizations, inevitably leading to some degree of performance degradation. Thus, the existing defenses against microarchitectural side channels represent a trade-off between efficiency and security.

4. Interconnect Contention Side Channel

Before delving into contention-based interconnect side channels and covert channels, it is necessary to understand the cache architecture of the CPU. These interconnect side channels and covert channels rely on the congestion generated by the traffic injected into the interconnect by the attacker. There are various methods for injecting traffic into the interconnect, with the most common and effective being sending requests from one core to an LLC slice located in a different tile.

4.1. CPU Cache Architecture

Modern CPU caches feature a multi-level structure. Taking the most common Intel x86 microarchitecture cache architecture as an example, CPU caches are divided into L1, L2 (present in most microarchitectures), and L3 (often referred to as the last-level cache or LLC). L1 and L2 caches are typically referred to as private caches, integrated with each CPU core on the processor, thus providing very fast access (4 to 12 cycles) but with smaller capacities, primarily used to store local data and instructions. In contrast, the LLC, compared to private caches, is slower (40 to 60 cycles) but has a larger capacity and is shared among the cores. Starting from the Nehalem-EX [57] architecture, the LLC has been divided into equally sized LLC slices, where each LLC slice is typically integrated with a core set from an interconnect perspective.

4.1.1. Non-Uniform Cache Access Architecture

Large-scale multi-core processors are equipped with a large last-level cache (LLC) to mitigate expensive off-chip accesses. For example, AMD Ryzen (AMD, Santa Clara, CA, USA) features a 256 MB LLC [58], while the Intel Xeon 6900 series has a 108 MB LLC [59]. This LLC is distributed across multiple banks (or tiles, for Intel) interconnected via an NoC to reduce access latency and improve core isolation, known as Non-Uniform Cache Access (NUCA) architecture. The Non-Uniform Cache Access (NUCA) architecture aims to reduce memory access times and improve cache hit rates by increasing cache capacity [58,60]. However, due to the limited bandwidth and port count of individual physical cache banks, multi-core processor chips employ physically separate banks to share the LLC, which are interconnected with each other through the NoC. Therefore, in the NUCA architecture, cache access latencies to different cache banks exhibit differences [61], as memory operations involve network protocols and message passing within the NoC.

4.1.2. Cache Inclusivity

Most Intel CPU caches are inclusive, meaning data contained in L1 and L2 caches must also reside in the LLC [62]. Private caches and LLCs are set-associative, meaning they are divided into fixed numbers of cache sets, each containing a fixed number of cache ways. Each cache way can be mapped to a cache line, with the cache line size typically

being 64 bytes, constituting the basic unit of cache transactions. Cache sets and the LLC slice storing each cache line are determined by their addresses. Since private caches usually have fewer sets than LLC, cache lines mapped to different LLC sets may map to the same L2 or L1 set. When a core executes a load from a memory address, it first checks if the data associated with that address is available in L1 and L2. If available, the load results in a hit in the private cache, and the data are serviced locally. If not, it results in a miss in the private cache, and the core checks if the data are available in the LLC. If there is an LLC miss, the data need to be fetched from DRAM via the memory controller. Memory controllers are integrated into system agents responsible for managing communication between the main memory and the CPU. Some Intel processors also implement hardware prefetching, which may result in additional cache lines being copied from memory or LLC to the private cache during a single load operation.

4.2. Ring Bus Contention

The ring interconnect, commonly referred to as a ring bus, is a high-bandwidth on-chip interconnect that first appeared in Intel's Nehalem-EX microarchitecture [57]. As illustrated in Figure 3, it facilitates internal communication among the cores, LLC, system agent (or Uncore), and GPU within the processor. When a core executes a load operation and experiences a cache miss in its private cache (L1-L2), the load must traverse the ring interconnect to reach the LLC and/or memory controller. Different CPU consumers/producers communicating via the ring interconnect are known as ring agents [63]. Each ring agent participates in ring communication through a ring stop (sometimes referred to as interface block, node router [64,65], or ring station [66]). Each core shares its ring stop with an LLC slice. To minimize latency, traffic on the ring interconnect always follows the shortest path, meaning the ring stop can inject/receive traffic in both clockwise or counterclockwise directions and always selects the direction of the shortest path to the destination (clockwise or counterclockwise in the figure). The communication protocol on the ring interconnect utilizes four physical rings: request, snoop, acknowledge, and data rings.

The contention-based interconnect side-channel attack relies on a crucial observation: the ring stations always prioritize existing traffic on the ring bus over new traffic entering from their agents. When new ring traffic is delayed due to existing ring traffic, contention occurs in the ring. Hence, When one agent bombards an LLC slice with requests, other agents loading from the same slice observe a delay.

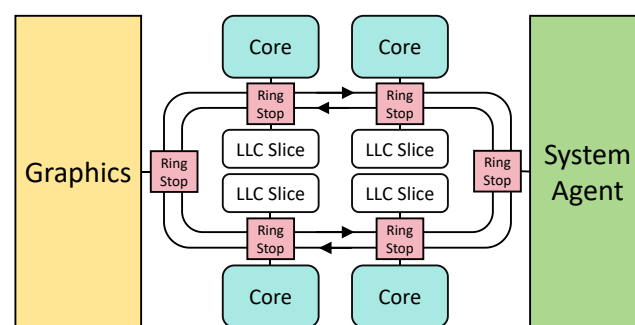


Figure 3. Ring bus architecture.

Covert Channel. In the scenario of cross-core covert channels, the sender and receiver share a segment of the ring interconnect. When sending a bit “1”, the sender sends loads to cause contention on the ring interconnect, while no action is taken when sending “0”. The receiver decodes the signal by timing the loads passing through the ring interconnect segment, which are prone to contention due to the sender’s loads. Therefore, when the sender sends “1”, the receiver’s load experiences a delay. To differentiate between “0” and “1”, the receiver can simply use the average load delay: smaller load delays are assigned to “0”, while larger load delays are assigned to “1”. Synchronization between

both parties can be achieved using a shared timestamp counter or extended to use other clock-independent techniques.

Side Channel. For side channels, the attacker (receiver) sends loads through a fixed segment of the ring interconnect and measures their delays. The attacker collects the load delay of each measurement as a sample and refers to the collection of many samples (i.e., one run of the attacker/receiver) as a trace. If, during the attack, the victim (sender) performs a memory access operation that meets certain conditions, the attacker will measure a longer load delay. Typically, slices accessed by unsuspecting victims are evenly distributed across the entire LLC. Many implementations of encryption algorithms select different functions based on the key bits, a common pattern exploited in many existing side-channel attacks. Different functions may exhibit different memory access patterns, resulting in varying degrees of load on the ring bus, which increases the delay of the attacker's load, allowing the attacker to infer the key.

4.3. Mesh Interconnect Contention

Due to the proximity to physical limits, improving chip manufacturing processes has become increasingly challenging, making the integration of more cores on a single CPU chip the only reliable option for manufacturers seeking higher-performing processors. However, the further increase in processor cores poses challenges for ring interconnects due to their scalability and routing scheme limitations. In ring interconnect architectures, the latency between cores may linearly increase with the number of cores, as communication between two cores may need to route through all other cores. To address this issue, Intel introduced mesh interconnect as an alternative to ring interconnect in the Skylake-SP server CPU family released in 2017 [33], which has since been widely adopted in Intel processors and holds promising prospects for the future. In addition to Intel CPUs, processors from other manufacturers also employ mesh interconnect, such as tile processors [67,68] and ARM server CPUs (United Kingdom) [69].

Essentially, mesh interconnect involves organizing the chip on a plane into a two-dimensional matrix consisting of several tiles. Each tile is either composed of a core along with an LLC slice or a non-core component such as an IMC (integrated memory controller), UPI (Ultra Path Interconnect) controller, or I/O unit, as illustrated in Figure 1. Each tile is equipped with a mesh stop and is connected to its vertical and horizontal neighbors, with the mesh stop responsible for managing traffic in each direction. In mesh interconnects, the number of hops between any two tiles is proportional to the square root of the tile count, effectively limiting the core-to-core latency on modern processors with rapidly increasing core counts.

Similar to contention side-channel attacks on the ring interconnect, contention side-channel attacks on the mesh interconnect also exploit contention on the same mesh route, as shown in Figure 4. As data movement on the mesh constantly occurs during various program activities, if an attacker's and victim's programs happen to share a mesh route, the victim's activities, including cache access, may be inferred. During this process, the attacker can only access their own resources.

Covert Channel. In the scenario of covert channels, the receiver anchors itself to a given core and accesses addresses mapped to a given LLC slice to monitor the interconnect. These accesses travel through the mesh interconnect and may experience delays due to interconnect contention from other applications. The receiver can then time these accesses to determine if contention has occurred. To generate reliable accesses to a given LLC slice, the receiver's accesses need to miss in L2 and hit in LLC while avoiding L2 hits and DRAM accesses. The sender's configuration is similar to the receiver's, avoiding or causing interconnect contention by alternating accesses to its own LLC slice and the target LLC slice to encode messages. When the sender sends a bit "1", it creates accesses through the mesh interconnect to the given LLC by evicting entries from the private L2 cache, which will contend with the receiver's accesses, resulting in higher latency observed by the receiver; conversely, when the receiver does not observe higher latency, indicating that the sender

has not accessed the distant LLC, the receiver decodes the signal in that time slice as a bit “0”.

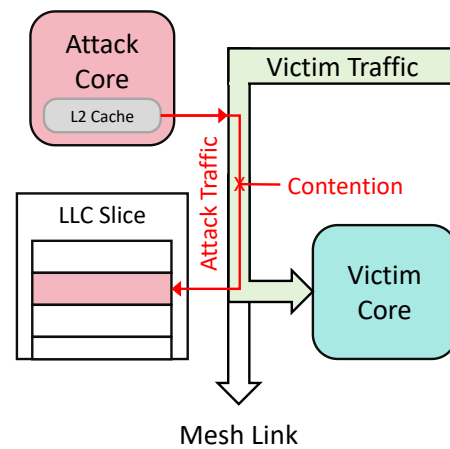


Figure 4. Mesh interconnect contention.

Side Channel. Similarly, side-channel attacks on the mesh interconnect also exploit the characteristic of encryption algorithms executing paths dependent on key bits. Unlike side-channel attacks on the ring interconnect, in the mesh interconnect, the victim’s memory accesses are typically evenly distributed across all LLC slices due to the design of slice hash functions. Therefore, attackers need to select an optimal position to maximize observable contention and send probing loads to multiple mesh routes as much as possible. To prevent victims from ceasing to generate contention after loading secrets into private caches, some side-channel attacks [21] also require specific defenses against side-channel attacks targeting private caches to be enabled system-wide, such as refreshing private caches on context switches, giving attackers the opportunity to trigger victim context switches to refresh the victim’s private cache, thereby requiring the victim’s next memory access in the same time slice to still generate interconnect traffic. Other work [22] assumes attacks can be executed on cloud VMs, where attackers cannot pin processes to their desired cores or learn which cores the victim application occupies, meaning attackers cannot select optimal routes for contention. However, even so, their evaluation shows a high chance of contention opportunities on random routes. As the victim application runs, attackers immediately sequentially access each address in the L2 eviction set concentrated in different LLCs upon receiving responses to previous requests and record timestamps for each request (e.g., using the RDTSCP instruction to read CPU counters). The interval between consecutive requests reflects cache transaction latency. When the interval increases, the victim should have performed related cache transactions simultaneously.

4.4. Mitigation

Ring Interconnect. Side-channel attacks based on ring bus contention share many characteristics with traditional microarchitectural side-channel attacks, the most fundamental of which is the exploitation of architecturally promised operations to create observable timing differences. Current mitigation strategies for such side channels are predominantly software-based, with the primary approach being adherence to constant-time programming principles. Although there is no precedent for attacking constant-time code in the existing Interconnect-based side channel literature, it is important to note that current constant-time programming techniques are mainly designed to counter side channels related to cache and instruction execution paths. Their effectiveness against interconnect-based microarchitectural side channels remains questionable. Furthermore, the implementation of truly constant-time code requires programmers to have a comprehensive and in-depth understanding of hardware optimizations, significantly increasing the complexity of constant-time code implementation. Reports indicate that Intel CPUs perform hardware memory

disambiguation between private caches and the ring interconnect [70]. This optimization can potentially disrupt constant-time programming by making ring contention a function of cache line content.

In pure hardware mitigation, designs based on spatial partitioning and static scheduling arbitration policies can ensure that there is no ring contention between processes from different security domains [71]. However, this approach does not address slice contention issues and still requires the introduction of additional mechanisms for mitigation. Another approach is to limit the range of cores capable of executing certain specific code, such as deploying trusted and untrusted code on separate core clusters while isolating the LLC slices of the two clusters. This less intrusive hardware–software co-design requires careful restriction of cross-cluster traffic generation, such as LLC miss, for ring bus architectures.

In terms of software mitigations, since proposed ring contention side channels heavily rely on the attacker (or receiver in a covert channel scenario) constantly missing in the private cache to load from the target LLC slice, purely software-based anomaly detection techniques can be developed. These techniques use hardware performance counters to monitor bursts of load requests to individual LLC slices. However, these techniques are only useful if they maintain a very low false-positive rate [20]. Some research has considered mitigations under multi-level security models, using principles similar to Quality of Service (QoS) optimization in the Internet by assigning higher priority to low-security domain traffic to ensure it is not affected by high-security domain traffic [72]. Nonetheless, this approach offers one-way protection against information leakage and does not meet the security requirements of today’s commonly used cloud server trusted execution environments.

Mesh Interconnect. Mitigation measures against mesh interconnect contention can be also categorized into software and hardware approaches. Software mitigation strategies, based on reverse engineering of the mesh interconnect, propose placing victim programs on cores less susceptible to such side-channel attacks, such as centrally located cores [21], or isolating the LLC to prevent attackers from generating mesh traffic [22]. Additionally, restricting the location of attacker cores can also be effective.

Hardware-level mitigation measures propose segregating traffic flows from different security domains. Wang and Suh studied a domain-aware priority arbitration strategy [72], which prioritizes low-security traffic over high-security traffic on routers. Wassel et al. proposed a time-multiplexed scheduling strategy [71], where network links can only carry traffic from predefined security domains at each moment. While these methods are effective, they require hardware modifications to the interconnect and once deployed, cannot be adjusted to accommodate more security domains.

5. Distance-Based Interconnect Side-Channel

The increase in the number of processor cores and the complexity of interconnects inevitably leads to greater differences in distances between different cores. As the lengths of interconnect links reach a certain scale, the time differences exhibited by electrical signals propagating over links of different lengths can be significantly observed and utilized to construct distance-based side-channel attacks [73].

5.1. Distance-Based Side Channel

Multi-core processors employ distributed LLCs to enhance memory-level parallelism by allowing concurrent requests to different banks. However, from a negative perspective, the latency of accessing different banks may vary depending on the distance from the requesting core. Hence, attackers can exploit addresses with significant differences in LLC hit times to construct side-channel or covert channels. Specifically, adjacent entries in a large array may be mapped to different LLC slices based on physical addresses and slice hash algorithms, and hence located in banks at different distances. By accessing these two adjacent entries, the attacker can infer sensitive information through measurable differences in LLC access times.

Covert Channel. In covert channels, the sender and receiver may agree upon a shared timer for the receiver to measure the time taken by the sender's cache access operations. The sender induces a private cache miss and sends requests to LLC slices located on banks at different distances. When sending a bit "1", the sender accesses array entries mapped to LLC slices located farther away, causing the receiver to observe longer access delays through the timer. Conversely, when the sender accesses local or closer LLC slices, the receiver observes lower delays, decoding this signal as a bit "0".

Side Channel. Distance-based side-channel attacks require attackers to determine the timing of sensitive operations in the victim's program through interaction with the victim, such as message exchange via networks or inter-process communication (IPC), detecting changes in shared variables, or utilizing other side channels. By exploiting timing differences, attackers can deduce whether the victim's program accessed the LLC on a closer or farther bank, implying access to different cache locations, as shown in Figure 5. In some encryption algorithms, such access pattern information can be used to infer critical secrets, such as AES algorithm key bits. Compared to contention-based side-channel attacks, distance-based side-channel attacks do not require the construction of contention or interference but demand the ability to directly determine the timing of victim operations and acquire fine-grained timing information about individual memory accesses or at least memory access groups.

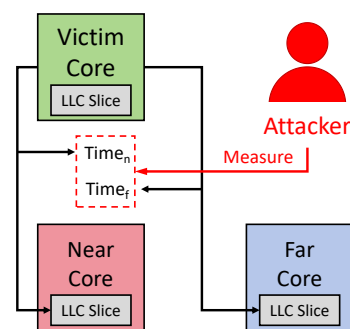


Figure 5. Distance-based side-channel attack.

5.2. Mitigation

Architecturally defending against distance-based side-channel attacks can be achieved most directly by ensuring uniform access time to all nodes simultaneously, a configuration feasible in many network setups. Additionally, artificial delays can be introduced in shorter path communications to prevent attackers from inferring cache access patterns based on the timing of communication between nearby and distant nodes. However, this approach implies that response times for all nodes are bounded by the response time of the furthest node, resulting in significant performance loss. Alternatively, leveraging the contention-based interconnect side-channel principles mentioned earlier, injecting a certain level of noise into the interconnect can induce contention, causing delays observed by attackers to be influenced by both distance and contention, thus preventing attackers from inferring sensitive information about victims.

6. Layout-Based Interconnect Side-Channel

In order to systematically organize processor cores and other components on a single chip, it is necessary to design the layout of these components on the plane. For processors, in such layouts, spatially adjacent tiles can also create side channels. This type of side channel no longer relies on wired interconnects but instead on a wireless interconnect pathway formed by thermal radiation, known as thermal channels [74]. This chapter will begin with an exploration of thermal management in modern computers and then introduce processor thermal-channel attacks.

6.1. Thermal Management

Thermal management is critical to the safe and reliable operation of modern computer systems. Many system components, such as hard disk drives, DRAM, GPUs, motherboards, and processor chips, integrate thermal sensors [75]. The pursuit of high performance in modern processors inevitably leads to an increase in chip power density, making it increasingly challenging to ensure the thermal stability of processors. Major processor vendors (such as Intel, AMD, and VIA) integrate thermal sensors in their processor products for real-time monitoring of chip temperature. ARM processors also include on-chip internal thermal sensors for power and temperature management.

Early thermal management was statically performed in hardware, akin to common circuit protections, mainly involving mechanisms like powering off the processor to prevent melting. However, power-off incurs significant costs to processor performance, leading to the evolution of more sophisticated dynamic frequency and voltage scaling techniques in thermal management, regulating temperature rise by altering processor frequencies [76,77]. Nowadays, popular thermal management techniques combine both software and hardware approaches, where the operating system polls temperature sensors and uses sensor data to manage cooling mechanisms like processor frequency scaling and fan speed [78].

Some processor manufacturers specify a maximum junction temperature for their products, representing the highest temperature at which the processor can operate safely. Exceeding this threshold may cause irreversible silicon damage to the processor. To avoid such scenarios, some processor manufacturers integrate a Digital Thermal Sensor (DTS) into each core of the processor to monitor processor temperature. Each DTS records the difference between the current temperature of the processor core and its maximum junction temperature [79], with the absolute value of this difference representing the current temperature of the processor core in Celsius. The accuracy of DTS varies across different processor products, with Intel's DTS typically having a resolution of ± 1 °C.

There is a recent trend towards opening up thermal management data to users. Some operating systems allow users to implement customized thermal management strategies based on actual application needs; for example, newer Linux systems permit users to configure thermal policies [80,81]. This user-centric thermal management philosophy aims at personalized and maximized utilization of processor performance but also provides opportunities for thermal-channel-based side-channel and covert-channel attacks. For instance, thermal data from DTS can be read from specific CPU registers of the corresponding core; in Linux systems, data from all sensors is exposed using the `coretemp` kernel module [82]. This information can be accessed from user space through the `sysfs` file system refreshed every 2 milliseconds.

Furthermore, different processor vendors and series typically have different thermal sensor topologies. For example, Intel and VIA processors use distributed sensors to monitor temperature data for each core, while AMD processors (such as the Opteron K10 series) only allow monitoring of the overall chip temperature using sensors within the processor socket [82]. Optimizing the number and topology of thermal sensors on processors remains an active research topic [83,84].

6.2. Thermal-Channel Attack

When processor cores operate under high-load conditions, their temperatures significantly increase due to resistive heating. This excess heat radiates and spreads to spatially adjacent cores, where it can be detected by dedicated temperature sensors, as shown in Figure 6. Covert and side channels constructed based on this characteristic are referred to as thermal channels. Unlike previously discussed side-channel attacks, these channels do not exploit on-chip interconnects but rather leverage another critical property of the NoC: the planar layout of processor cores on the chip.

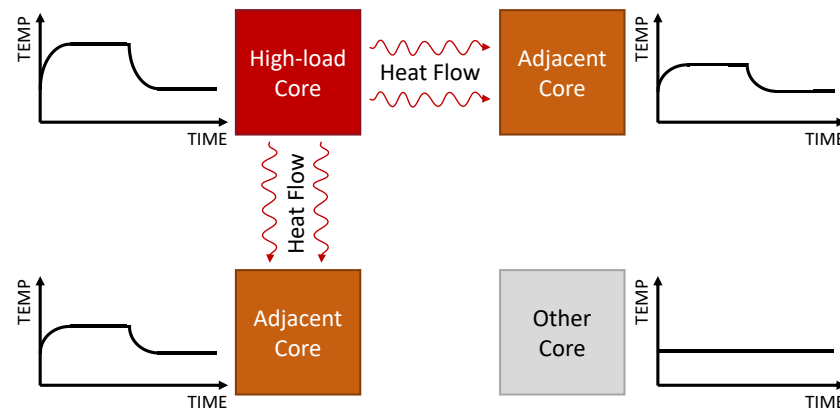


Figure 6. Heat propagation from a neighbouring core.

It is easy to imagine that even if programs on different cores can only access temperature information from their respective cores, the presence of thermal radiation enables attackers to easily construct cross-core covert channels. The sender and receiver can pre-arrange a fixed-length time slice and the positions of their cores in the layout. As long as the sender and receiver cores are sufficiently close in the layout, the sender can execute computational operations at a high intensity within the fixed time slice to increase the temperature of its own core and the surrounding cores. The receiver continuously monitors the temperature changes in its own core, and when the average temperature within a time slice is significantly higher than usual, it means the sender has transmitted a bit “1”; otherwise, it sends a bit “0”.

It is worth mentioning that, unlike time-based covert-channel attacks, thermal channels can not only bypass spatial isolation of program resources but also temporal isolation. This is because the core performing high-load computations does not immediately return to normal temperature levels after the program ends, due to the residual heat effect. Consequently, processes following this computation in the execution schedule will have the opportunity to obtain information about that computation.

Although thermal channels for covert communication have been extensively studied, it is difficult to relate the temperature changes in victim cores to specific secrets to construct side-channel attacks due to their coarse granularity in time. Nevertheless, thermal channels have been demonstrated to leak information about the computational nature of victim programs to processes on other cores, which can be used to complement other attack techniques, such as determining the timing of execution of attack code.

6.3. Challenges in Mitigating Heat-Channel Attacks

Heat-channel-based side-channel and covert-channel attacks rely entirely on temperature information exposed to software, making the natural solution to this issue limiting access to temperature sensors on the system. However, this requirement contradicts the original intent of thermal management and its design flexibility for pursuing optimal performance. Moreover, centralized monitoring and control of processor thermal states have become increasingly challenging to adapt to the growing number of processor cores in modern commercial processors. Modern commercial processors can now contain hundreds of cores, each hosting numerous autonomous processes. For instance, Intel’s SCC platform [85], a research prototype, allows subsets of cores to independently manage their frequency and voltage to enhance power efficiency, making it impossible to hide temperature information, a crucial input in the decision-making process, within this architecture. Balancing performance and security poses a challenge for all processor microarchitecture optimization measures, including thermal management.

In addition to temperature sensors, many other parameters may still leak information about the system’s thermal state, such as clock skew, fan speed, and even processor frequency in systems that allow dynamic frequency scaling. Since all these parameters are

typically shared among cores or subsets of cores within the processor chip, even if access to temperature sensors is restricted or prohibited, these parameters can still be used to implement side-channel and covert-channel attacks.

7. Future Directions for Interconnect Side-Channel Attacks

Despite the emergence of several highly threatening side-channel attacks targeting on-chip interconnects, microarchitecture designers and security researchers need to be particularly vigilant about the evolving trends of processor interconnect side channels. In this chapter, we analyze the seriousness of future interconnect side-channel threats and the necessity for related security research from three perspectives: the development of interconnect side channels, the combination of interconnects with traditional processor microarchitecture side channels, and the integration of interconnect side channels with other SoC security threats.

7.1. Development of Interconnect Side Channels

Examining the history of on-chip interconnects and their security research, it is evident that the increasing complexity of interconnect structures with the expansion of SoC or CPUs has provided new opportunities for side-channel attacks. During the era of dedicated signal line buses, interconnects were generally considered immune to side-channel exploitation. However, with the adoption of Network-on-Chip (NoC) concepts to accommodate complex SoC, contention-based timing side channels began to emerge. As processors continued to scale, the increased distance between cores introduced observable timing differences and distance-based interconnect side channels. The appearance of contention-based side-channel attacks from ring-to-mesh interconnects underscores the longstanding neglect of interconnect side-channel issues.

The advent of higher computational power chips is an irreversible trend, necessitating further optimization of NoC Quality of Service (QoS), arbitration mechanisms, and data flows to support larger processors. This will lead to more complex interconnects with new structures or characteristics, which are highly susceptible to side-channel exploitation. Examples include novel physical side channels introduced by emerging technologies such as wireless NoC [86,87] and optical NoC [88–90]. Even existing interconnect structures harbor potential side-channel attack opportunities, such as routing table blocking in mesh interconnects.

7.2. Integration of Interconnect and Traditional Processor Microarchitecture Side Channels

Although processor microarchitecture side channels have received extensive attention and discussion, the vulnerabilities introduced by microarchitectural optimizations continue to emerge, with mitigation measures generally lagging behind. The rise of interconnect side channels provides new avenues for exploiting these vulnerabilities. Interconnect characteristics that have been or could potentially be used by side channels can enhance existing side-channel or other microarchitectural attacks, allowing them to bypass deployed defenses. For instance, reducing the timer precision required for cache side-channel attacks or transient execution attacks can circumvent restrictions on access to high-precision timers.

7.3. Combination of Interconnect Side Channels with Other SoC Security Threats

Before the security issues of interconnects on processors were recognized, security researchers had already been discussing the security of NoC or interconnects on SoC. However, research related to side-channel and covert-channel security has been relatively lagging, as SoCs face unique security threats such as malicious IP cores, and many of the assumptions made about security in SoCs are not as relaxed as those for processors. This means that SoCs need to deal with more complex and diverse security issues, such as eavesdropping attacks, deception attacks, and denial-of-service attacks. However, processor interconnect side-channel attacks will provide new insights for existing SoC attacks, such

as the lesser-discussed timing-based attacks on SoC interconnects that provide new attack opportunities for eavesdropping.

8. Conclusions

In this paper, we summarize the side-channel and covert-channel vulnerabilities based on interconnects and outline the most advanced attacks and defense measures. We particularly discuss the three key characteristics of interconnects that are most easily exploited by side-channel and covert-channel attackers: lines contention, core distance, and spatial layout, focusing on their attack methods and separately discussing the countermeasures that have been proposed and their limitations. Finally, we analyze the future direction of interconnect side-channel.

To date, there remains insufficient attention from both industry and academia on the security implications of interconnects. Traditional defense mechanisms against other microarchitectural side-channels are largely ineffective against interconnect-based attacks, making interconnects an increasingly important target for microarchitectural attacks. In the foreseeable future, the pursuit of processor performance will continue unabated. Whether through further improvements in chip fabrication processes or the emergence of super processors integrating more cores, this inevitably entails new interconnect architectures and the associated security vulnerabilities.

Current research surveys on NoC security issues remain limited, and those available are either outdated [91] or lack comprehensive discussion on side-channel vulnerabilities [92]. This paper, being the first review focusing on processor interconnect side channels and covert channels, aims to summarize and reflect on existing attacks and defenses. Through this, we hope to elevate the academic focus on the security of processor and SoC interconnects, aiding future research endeavors. However, this survey is limited in scope; it discusses only a few impactful interconnect side-channel attacks, confines its discussion to the NoC of commercial processors, and treats interconnect side channels as a subset of microarchitectural side channels without delving into broader SoC interconnect security issues. As described in Section 7, interconnect side channels have significant potential to combine with other microarchitectural side-channel attacks or exploit other SoC security vulnerabilities, thereby amplifying threat levels or creating new attacks. We plan to address other SoC security issues based on or utilizing interconnect characteristics in future research.

Author Contributions: Methodology, J.Y.; Formal analysis, D.L.; Investigation, X.W.; Writing—original draft, J.Z.; Writing—review & editing, P.Q. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kocher, P.; Horn, J.; Fogh, A.; Genkin, D.; Gruss, D.; Haas, W.; Hamburg, M.; Lipp, M.; Mangard, S.; Prescher, T.; et al. Spectre Attacks: Exploiting Speculative Execution. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2019. [\[CrossRef\]](#)
2. Lipp, M.; Schwarz, M.; Gruss, D.; Prescher, T.; Haas, W.; Fogh, A.; Horn, J.; Mangard, S.; Kocher, P.; Genkin, D.; et al. Meltdown: Reading kernel memory from user space. In Proceedings of the 27th Security Symposium Security 2018, Baltimore, MD, USA, 15–17 August 2018.
3. Zhang, J.; Chen, C.; Cui, J.; Li, K. Timing Side-Channel Attacks and Countermeasures in CPU Microarchitectures. *ACM Comput. Surv.* **2024**, *56*, 1–40. [\[CrossRef\]](#)
4. Wang, Y.; Paccagnella, R.; Shacham, H.; Fletcher, C.; Kohlbrenner, D. Hertzbleed: Turning Power Side-Channel Attacks into Remote Timing Attacks on x86. In Proceedings of the 31st USENIX Security Symposium (USENIX Security 22), Boston, MA, USA, 10–12 August 2022.

5. Gast, S.; Juffinger, J.; Schwarzl, M.; Saileshwar, G.; Kogler, A.; Franza, S.; Köstl, M.; Gruss, D. SQUIP: Exploiting the Scheduler Queue Contention Side Channel. In Proceedings of the 2023 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 22–25 May 2023; pp. 2256–2272. [\[CrossRef\]](#)
6. Kumar, S.; Dasu, V.A.; Bakshi, A.; Sarkar, S.; Jap, D.; Breier, J.; Bhasin, S. Side Channel Attack On Stream Ciphers: A Three-Step Approach To State/Key Recovery. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2022**, *2022*, 166–191. [\[CrossRef\]](#)
7. Goy, G.; Loiseau, A.; Gaborit, P. A New Key Recovery Side-Channel Attack on HQC with Chosen Ciphertext. In *Post-Quantum Cryptography: Proceedings of the 13th International Workshop, PQCrypto 2022, Virtual Event, 28–30 September 2022*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2022; pp. 353–371. [\[CrossRef\]](#)
8. Ridder, F.; Frigo, P.; Vannacci, E.; Bos, H.; Giuffrida, C.; Razavi, K. SMASH: Synchronized Many-sided Rowhammer Attacks from JavaScript. In Proceedings of the 30th USENIX Security Symposium (USENIX Security 21), Virtual, 11–13 August 2021.
9. Zhang, R.; Kim, T.; Weber, D.; Schwarz, M. ({M} WAIT} for It: Bridging the Gap between Microarchitectural and Architectural Side Channels. In Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23), Anaheim, CA, USA, 9–11 August 2023; pp. 7267–7284.
10. Kurth, M.; Gras, B.; Andriesse, D.; Giuffrida, C.; Bos, H.; Razavi, K. NetCAT: Practical Cache Attacks from the Network. In Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 18–21 May 2020. [\[CrossRef\]](#)
11. Javed, A.R.; Beg, M.O.; Asim, M.; Baker, T.; Al-Bayatti, A.H. AlphaLogger: Detecting motion-based side-channel attack using smartphone keystrokes. *J. Ambient. Intell. Humaniz. Comput.* **2023**, *14*, 4869–4882. [\[CrossRef\]](#)
12. Gulmezoglu, B.; Zankl, A.; Tol, M.C.; Islam, S.; Eisenbarth, T.; Sunar, B. Undermining User Privacy on Mobile Devices Using AI. In Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, Auckland, New Zealand, 9–12 July 2019. [\[CrossRef\]](#)
13. Yan, M.; Fletcher, C.; Torrellas, J. Cache Telepathy: Leveraging Shared Resource Attacks to Learn DNN Architectures. In Proceedings of the 29th USENIX Security Symposium (USENIX Security 20), Berkeley, CA, USA, 12–14 August 2018.
14. Guo, Y.; Zigerelli, A.; Zhang, Y.; Yang, J. Adversarial Prefetch: New Cross-Core Cache Side Channel Attacks. In Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 23–26 May 2022; pp. 1458–1473. [\[CrossRef\]](#)
15. Schwarz, M.; Lipp, M.; Gruss, D.; Weiser, S.; Maurice, C.; Spreitzer, R.; Mangard, S. Key Drown: Eliminating Software-Based Keystroke Timing Side-Channel Attacks. In Proceedings of the 2018 Network and Distributed System Security Symposium, San Diego, CA, USA, 18–21 February 2018. [\[CrossRef\]](#)
16. Wang, D.; Qian, Z.; Abu-Ghazaleh, N.; Krishnamurthy, S.V. PAPP. In Proceedings of the 56th Annual Design Automation Conference 2019, New York, NY, USA, 2–6 June 2019. [\[CrossRef\]](#)
17. Xiao, C.; Tang, M.; Guille, S. Exploiting the microarchitectural leakage of prefetching activities for side-channel attacks. *J. Syst. Archit.* **2023**, *139*, 102877. [\[CrossRef\]](#)
18. Wang, Z.; Taram, M.; Moghimi, D.; Swanson, S.; Tullsen, D.; Zhao, J. NVLeak: Off-Chip Side-Channel Attacks via Non-Volatile Memory Systems. In Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23), Anaheim, CA, USA, 9–11 August 2023; pp. 6771–6788.
19. Zhao, Z.N.; Morrison, A.; Fletcher, C.W.; Torrellas, J. Binoculars: Contention-Based Side-Channel attacks exploiting the page walker. In Proceedings of the 31st USENIX Security Symposium (USENIX Security 22), Boston, MA, USA, 10–12 August 2022; pp. 699–716.
20. Paccagnella, R.; Luo, L.; Fletcher, C. Lord of the Ring(s): Side Channel Attacks on the CPU On-Chip Ring Interconnect Are Practical. In Proceedings of the 30th USENIX Security Symposium (USENIX Security 21), Virtual, 11–13 August 2021.
21. Dai, M.; Paccagnella, R.; Gomez, M.; Mit, G.; McCalpin, J.; Mengjia, T.; Mit, Y. Don't Mesh Around: Side-Channel Attacks and Mitigations on Mesh Interconnects. In Proceedings of the 31st USENIX Security Symposium (USENIX Security 22), Boston, MA, USA, 10–12 August 2022.
22. Wan, J.; Bi, Y.; Zhou, Z.; Li, Z. MeshUp: Stateless Cache Side-channel Attack on CPU Mesh. In Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 22–26 May 2022.
23. Wu, S.Y.; Chang, C.; Chiang, M.; Lin, C.; Liaw, J.; Cheng, J.; Yeh, J.; Chen, H.; Chang, S.; Lai, K.; et al. A 3 nm CMOS FinFlex™ Platform Technology with Enhanced Power Efficiency and Performance for Mobile SoC and High Performance Computing Applications. In Proceedings of the 2022 International Electron Devices Meeting (IEDM), San Francisco, CA, USA, 3–7 December 2022.
24. IEEE. International Roadmap for Devices and Systems (IRDS) 2023 Update. Available online: https://irds.ieee.org/images/files/pdf/2023/2023IRDS_Litho.pdf (accessed on 29 April 2024).
25. Intel. Intel® Xeon® 6780E Processor. Available online: <https://www.intel.com/content/www/us/en/products/sku/240362/intel-xeon-6780e-processor-108m-cache-2-20-ghz/specifications.html> (accessed on 6 June 2024).
26. Intel. Intel Xeon 6 Processors. Available online: <https://download.intel.com/newsroom/2024/client-computing/Xeon-6-Fact-Sheet.pdf> (accessed on 6 June 2024).
27. Network on a Chip. Available online: https://en.wikipedia.org/wiki/Network_on_a_chip (accessed on 5 June 2024).
28. ARM. Amba Specification. Available online: <https://developer.arm.com/products/architecture/amba-protocol> (accessed on 29 April 2024).
29. IBM Microelectronics. CoreConnect Bus Architecture. Available online: http://www.scarpaz.com/2100-papers/SystemOnChip/ibm_core_connect_whitepaper.pdf (accessed on 29 April 2024).

30. Benini, L.; De Micheli, G. Networks on chips: A new SoC paradigm. *Computer* **2002**, *35*, 70–78. [\[CrossRef\]](#)
31. Dally, W.; Towles, B. Route packets, not wires: On-chip interconnection networks. In Proceedings of the 38th Design Automation Conference (IEEE Cat. No.01CH37232), Las Vegas, NV, USA, 22 June 2001. [\[CrossRef\]](#)
32. Intel Ultra Path Interconnect. Available online: <https://en.wikipedia.org/wiki/IntelUltraPathInterconnect> (accessed on 1 May 2024).
33. WikiChip. Mesh Interconnect Architecture—Intel. Available online: https://en.wikichip.org/wiki/intel/mesh_interconnect_architecture (accessed on 1 May 2024).
34. Lipp, M.; Hadžić, V.; Schwarz, M.; Perais, A.; Maurice, C.; Gruss, D. Take a Way: Exploring the Security Implications of AMD's Cache Way Predictors. In Proceedings of the 15th ACM Asia Conference on Computer and Communications Security, Taipei, Taiwan, 2–9 October 2020. [\[CrossRef\]](#)
35. Gras, B.; Razavi, K.; Bos, H.; Giuffrida, C. Translation leak-aside buffer: Defeating cache side-channel protections with TLB attacks. In Proceedings of the 27th USENIX Security Symposium (USENIX Security 18), Baltimore, MD, USA, 15–17 August 2018.
36. Evtvushkin, D.; Ponomarev, D.; Abu-Ghazaleh, N. Jump over ASLR: Attacking branch predictors to bypass ASLR. In Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Taipei, Taiwan, 15–19 October 2016.
37. Evtvushkin, D.; Ponomarev, D.; Abu-Ghazaleh, N. Understanding and Mitigating Covert Channels Through Branch Predictors. *ACM Trans. Archit. Code Optim.* **2016**, *13*, 1–23. [\[CrossRef\]](#)
38. Disselkoe, C.; Kohlbrenner, D.; Porter, L.; Tullsen, D. Prime+abort: A timer-free high-precision L3 cache attack using intel TSX. In Proceedings of the 26th USENIX Security Symposium (USENIX Security 17), Vancouver, BC, Canada, 16–18 August 2017.
39. Wang, Z.; Lee, R. Covert and Side Channels Due to Processor Architecture. In Proceedings of the 2006 22nd Annual Computer Security Applications Conference (ACSAC'06), Miami Beach, FL, USA, 11–15 December 2006. [\[CrossRef\]](#)
40. Bhattacharyya, A.; Sandulescu, A.; Neugschwandtner, M.; Sorniotti, A.; Falsafi, B.; Payer, M.; Kurmus, A. SMoTherSpectre: Exploiting speculative execution through port contention. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, London, UK, 11–15 November 2019. [\[CrossRef\]](#)
41. Gras, B.; Giuffrida, C.; Kurth, M.; Bos, H.; Razavi, K. ABSynthe: Automatic Blackbox Side-channel Synthesis on Commodity Microarchitectures. In Proceedings of the 2020 Network and Distributed System Security Symposium, San Diego, CA, USA, 23–26 February 2020. [\[CrossRef\]](#)
42. Yarom, Y.; Genkin, D.; Heninger, N., CacheBleed: A Timing Attack on OpenSSL Constant Time RSA. In *Cryptographic Hardware and Embedded Systems—CHES 2016*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2016; pp. 346–367. [\[CrossRef\]](#)
43. Wu, Z.; Xu, Z.; Wang, H. Whispers in the hyper-space: High-speed covert channel attacks in the cloud. In Proceedings of the USENIX Security Symposium, Bellevue, WA, USA, 8–10 August 2012.
44. Evtvushkin, D.; Ponomarev, D. Covert Channels through Random Number Generator. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016. [\[CrossRef\]](#)
45. Evtvushkin, D.; Riley, R.; Abu-Ghazaleh, N.C.A.E.; Ponomarev, D. BranchScope. In Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems, Williamsburg, VA, USA, 24–28 March 2018. [\[CrossRef\]](#)
46. Sullivan, D.; Arias, O.; Meade, T.; Jin, Y. Microarchitectural Minefields: 4K-Aliasing Covert Channel and Multi-Tenant Detection in IaaS Clouds. In Proceedings of the 2018 Network and Distributed System Security Symposium, San Diego, CA, USA, 18–21 February 2018. [\[CrossRef\]](#)
47. Yan, M.; Sprabery, R.; Gopireddy, B.; Fletcher, C.; Campbell, R.; Torrellas, J. Attack Directories, Not Caches: Side Channel Attacks in a Non-Inclusive World. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2019. [\[CrossRef\]](#)
48. Armasu, L. OpenBSD Will Disable Intel Hyper-Threading to Avoid Spectre-Like Exploits (Updated). Available online: <https://www.tomshardware.com/news/openbsd-disables-intel-hyper-threading-spectre,37332.html> (accessed on 1 May 2024).
49. Sprabery, R.; Evchenko, K.; Raj, A.; Bobba, R.B.; Mohan, S.; Campbell, R. Scheduling, Isolation, and Cache Allocation: A Side-Channel Defense. In Proceedings of the 2018 IEEE International Conference on Cloud Engineering (IC2E), Orlando, FL, USA, 17–20 April 2018. [\[CrossRef\]](#)
50. Wang, Y.; Ferraiuolo, A.; Suh, G.E. Timing channel protection for a shared memory controller. In Proceedings of the 2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA), Orlando, FL, USA, 15–19 February 2014. [\[CrossRef\]](#)
51. Zhou, Z.; Reiter, M.; Zhang, Y. A software approach to defeating side channels in last-level caches. In Proceedings of the Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016.
52. Ge, Q.; Yarom, Y.; Cock, D.; Heiser, G. A Survey of Microarchitectural Timing Attacks and Countermeasures on Contemporary Hardware. *J. Cryptogr. Eng.* **2018**, *8*, 1–27. [\[CrossRef\]](#)
53. Ge, Q.; Yarom, Y.; Heiser, G. No Security without Time Protection. In Proceedings of the 9th Asia-Pacific Workshop on Systems, Jeju Island, Republic of Korea, 27–28 August 2018. [\[CrossRef\]](#)
54. Guarnieri, M.; Kopf, B.; Reineke, J.; Vila, P. Hardware-Software Contracts for Secure Speculation. In Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 24–27 May 2021. [\[CrossRef\]](#)
55. Barbosa, M.; Barthe, G.; Bhargavan, K.; Blanchet, B.; Cremers, C.; Liao, K.; Parno, B. SoK: Computer-Aided Cryptography. In Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 24–27 May 2021. [\[CrossRef\]](#)

56. Fei, S.; Yan, Z.; Ding, W.; Xie, H. Security Vulnerabilities of SGX and Countermeasures. *ACM Comput. Surv.* **2022**, *54*, 1–36. [CrossRef]
57. Kottapalli, S.; Baxter, J. Nahalem-EX CPU architecture. In Proceedings of the 2009 IEEE Hot Chips 21 Symposium (HCS), Stanford, CA, USA, 23–25 August 2009. [CrossRef]
58. AMD. AMD Ryzen. Available online: <https://www.amd.com/en/processors/ryzen> (accessed on 2 May 2024).
59. Intel. Intel Xeon PHI. Available online: <https://ark.intel.com/content/www/us/en/ark/products/series/75557/intel-xeon-phi-processors.html> (accessed on 2 May 2024).
60. Jaleel, A.; Mattina, M.; Jacob, B. Last Level Cache (LLC) Performance of Data Mining Workloads On a CMP—A Case Study of Parallel Bioinformatics Workloads. In Proceedings of the The Twelfth International Symposium on High-Performance Computer Architecture, Austin, TX, USA, 11–15 February 2006. [CrossRef]
61. Kim, C.; Burger, D.; Keckler, S.W. An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches. In Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems, San Jose, CA, USA, 5–9 October 2002. [CrossRef]
62. Intel. Intel 64 and IA-32 Architectures Optimization Reference Manual. September 2019. Available online: <https://www.intel.com/content/dam/doc/manual/64-ia-32-architectures-optimization-manual.pdf> (accessed on 2 May 2024).
63. Kanter, D. Intel’s Sandy Bridge Microarchitecture. Available online: <https://www.realworldtech.com/sandy-bridge/8/> (accessed on 3 May 2024).
64. Ausavarungnirun, R.; Fallin, C.; Yu, X.; Chang, K.K.W.; Nazario, G.; Das, R.; Loh, G.H.; Mutlu, O. Design and Evaluation of Hierarchical Rings with Deflection Routing. In Proceedings of the 2014 IEEE 26th International Symposium on Computer Architecture and High Performance Computing, Paris, France, 22–24 October 2014. [CrossRef]
65. Fallin, C.; Yu, X.; Nazario, G.; Mutlu, O. A High-Performance Hierarchical Ring On-Chip Interconnect with Low-Cost Routers. *Comput. Archit. Lab. Carnegie Mellon Univ. Tech. Rep.* **2011**, *7*, 2011.
66. Saini, S.; Chang, J.; Jin, H., Performance Evaluation of the Intel Sandy Bridge Based NASA Pleiades Using Scientific and Engineering Applications. In *High Performance Computing Systems. Performance Modeling, Benchmarking and Simulation*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2014; pp. 25–51. [CrossRef]
67. Vangal, S.; Howard, J.; Ruhl, G.; Dighe, S.; Wilson, H.; Tschanz, J.; Finan, D.; Iyer, P.; Singh, A.; Jacob, T.; et al. An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS. In Proceedings of the 2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers, San Francisco, CA, USA, 11–15 February 2007. [CrossRef]
68. Bell, S.; Edwards, B.; Amann, J.; Conlin, R.; Joyce, K.; Leung, V.; MacKay, J.; Reif, M.; Bao, L.; Brown, J.; et al. TILE64—Processor: A 64-Core SoC with Mesh Interconnect. In Proceedings of the 2008 IEEE International Solid-State Circuits Conference—Digest of Technical Papers, San Francisco, CA, USA, 3–7 February 2008. [CrossRef]
69. Gelas, J.D. New ARM IP Launched: CMN-600 Interconnect for 128 Cores and DMC-620, an 8Ch DDR4 IMC. 2016. Available online: <https://www.anandtech.com/show/10711/arm-cmn-600-dmc-620-128-cores-8-channel-ddr4> (accessed on 3 May 2024).
70. Downs, T. Hardware Store Elimination. 2020. Available online: <https://travisdowns.github.io/blog/2020/05/13/intel-zero-opt.html> (accessed on 3 May 2024).
71. Wassel, H.M.G.; Gao, Y.; Oberg, J.K.; Huffmire, T.; Kastner, R.; Chong, F.T.; Sherwood, T. SurfNoC. In Proceedings of the 40th Annual International Symposium on Computer Architecture, Tel-Aviv, Israel, 23–27 June 2013. [CrossRef]
72. Wang, Y.; Suh, G.E. Efficient Timing Channel Protection for On-Chip Networks. In Proceedings of the 2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip, Lyngby, Denmark, 9–11 May 2012. [CrossRef]
73. Mahmud, F.; Kim, S.; Chawla, H.; Tsai, C.C.; Kim, E.; Muzahid, A. Attack of the Knights: A Non Uniform Cache Side-Channel Attack. In Proceedings of the 39th Annual Computer Security Applications Conference, Austin, TX, USA, 4–8 December 2021.
74. Masti, R.; Rai, D.; Ranganathan, A.; Müller, C.; Thiele, L.; Capkun, S. Thermal covert channels on multi-core platforms. In Proceedings of the 24th USENIX Security Symposium (USENIX Security 15), Washington, DC, USA, 14–16 August 2015.
75. Open Hardware Monitor. Available online: <http://openhwaremonitor.org/> (accessed on 3 May 2024).
76. Advanced Micro Devices. Cool and Quiet Technology Installation Guide for AMD Athlon 64 Processor Based Systems. Available online: http://www.amd.com/Documents/Cool_N_Quiet_Installation_Guide3.pdf (accessed on 3 May 2024).
77. Intel Corporation. Intel SpeedStep FAQ. Available online: <http://www.intel.com/support/processors/sb/CS-032349.htm?wapkw=intel+speedstep> (accessed on 3 May 2024).
78. CPU Frequency and Voltage Scaling Code in the Linux(TM) Kernel. Available online: <https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt> (accessed on 3 May 2024).
79. Intel. CPU Monitoring with DTS/PECI. Available online: <https://hww.ru/wp/wp-content/uploads/2020/06/cpu-monitoring-dts-peci-paper.pdf> (accessed on 3 May 2024).
80. Daniel, A.; Guittot, V. RL A Simplified Thermal Framework for ARM Platforms. Available online: https://elinux.org/images/2/2b/A_New_Simplified_Thermal_Framework_For_ARM_Platforms.pdf (accessed on 2 May 2024).
81. Brown, L.; Seshadri, H. Cool hand linux* handheld thermal extensions. In *Linux Symposium*; Citeseer: Princeton, NJ, USA, 2007; p. 75.
82. CoreTemp. Available online: <http://www.alcpu.com/CoreTemp/> (accessed on 3 May 2024).
83. Lee, K.J.; Skadron, K.; Huang, W. Analytical model for sensor placement on microprocessors. In Proceedings of the 2005 International Conference on Computer Design, San Jose, CA, USA, 2–5 October 2005. [CrossRef]

84. Memik, S.O.; Mukherjee, R.; Ni, M.; Long, J. Optimizing Thermal Sensor Allocation for Microprocessors. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2008**, *27*, 516–527. [[CrossRef](#)]
85. Corporation, I. SCC External Architecture Specification. Available online: https://en.wikichip.org/w/images/f/f9/SCC_EAS.pdf (accessed on 3 May 2024).
86. Sarihi, A.; Patooghy, A.; Khalid, A.; Hasanzadeh, M.; Said, M.; Badawy, A.H.A. A Survey on the Security of Wired, Wireless, and 3D Network-on-Chips. *IEEE Access* **2021**, *9*, 107625–107656. [[CrossRef](#)]
87. Shruthi, R.; Shashidhara, H.R.; Deepthi, M.S. Comprehensive Survey on Wireless Network on Chips. In *Proceedings of the International Conference on Paradigms of Communication, Computing and Data Sciences*; Springer: Singapore, 2022; pp. 203–218. [[CrossRef](#)]
88. Kavitha, T.; Maheswaran, G.; Maheswaran, J.; Pappa, C.K. Optical network on chip: Design of wavelength routed optical ring architecture. *Bull. Electr. Eng. Inform.* **2022**, *12*, 167–175. [[CrossRef](#)]
89. Sharma, K.; Sehgal, V.K. Improved and Efficient Optical-NoC Architecture. In *Proceedings of the 2021 6th International Conference for Convergence in Technology (I2CT)*, Maharashtra, India, 2–4 April 2021. [[CrossRef](#)]
90. Baharloo, M.; Abdollahi, M.; Baniasadi, A. System-level reliability assessment of optical network on chip. *Microprocess. Microsyst.* **2023**, *99*, 104843. [[CrossRef](#)]
91. Fiorin, L.; Silvano, C.; Sami, M. Security aspects in networks-on-chips: Overview and proposals for secure implementations. In *Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*, Lubeck, Germany, 29–31 August 2007; pp. 539–542.
92. Charles, S.; Mishra, P. A Survey of Network-on-Chip Security Attacks and Countermeasures. *ACM Comput. Surv.* **2022**, *54*, 1–36. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.