Mitigating Timing-Based NoC Side-Channel Attacks With LLC Remapping

Anurag Kar, Xueyang Liu, Yonghae Kim, Gururaj Saileshwar, Hyesoon Kim, and Tushar Krishna

***Abstract*—Recent CPU microarchitectural attacks utilize contention over the NoC to mount covert and side-channel attacks on multicore CPUs and leak information from victim applications. We propose NoIR, a dynamic LLC slice selection mechanism using slice remapping to obfuscate interconnect contention patterns. NoIR reduces contention variance by 92.18% and mean IPC degradation due to cache invalidation is limited to 7.38% for SPEC CPU 2017 benchmarks for a 1000-access threshold. While previous defenses focused on redesigning the NoC and routing algorithms, we show that a top-down system-level approach can significantly raise the bar for a NoC security vulnerability with minimal modifications to the NoC hardware.**

***Index Terms*—Network-on-chip, security, side-channel attacks.**

I. Introduction

SIDE-CHANNEL attacks in processors allow an attacker to leak information from a victim process by exploiting contention for shared micro-architectural resources. While traditionally, cache side-channel attacks have relied on introducing contention for shared cache-sets, recent attacks have exploited contention over the on-chip interconnect to mount powerful stateless side-channel attacks which evade existing defenses [1], [2].

In these attacks, the victim sends traffic over the on-chip interconnect, while the attacker creates contention with the victim traffic and monitors the timing of its accesses to infer secret data. Previous attacks [1], [2], [3], [4] demonstrate that sensitive information like RSA keys and keystroke events can be leaked with this mechanism. Thus, for holistic defenses, the focus shifts to securing the interconnect as well. Previous works on defending against NoC timing side channels have involved temporal partitioning, static scheduling, and reversing priority [5], [6], [7], [8], but have high performance and latency overheads or only provide unidirectional protection.

We propose NoIR (Noc with Integrated Remapping) which randomizes the physical-address to slice mappings with remapping. We use an encryption-based mechanism that randomizes PA to LLC slice mapping and an encryption key change remaps the LLC slices. However, we posit that any remapping technique could be employed. All prior works on cache randomization [9] use randomization of set indices within a cache slice, and are thus vulnerable to NoC-based attacks. To our knowledge, this is the first work that uses a dynamic LLC slice remapping technique to randomize NoC traffic. NoIR cuts contention variance by 92.18% and its performance overhead due to cache invalidation is limited to 7.38% compared to the baseline.

Manuscript received 21 March 2023; revised 23 April 2023; accepted 7 May 2023. Date of publication 16 May 2023; date of current version 24 May 2023. *(Corresponding author: Anurag Kar.)*

Anurag Kar, Xueyang Liu, Yonghae Kim, Hyesoon Kim, and Tushar Krishna are with the Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: akar34@gatech.edu; xliu791@gatech.edu; yonghae@gatech.edu; hyesoon@cc.gatech.edu; tushar@ece.gatech.edu).

Gururaj Saileshwar is with the University of Toronto, Toronto, ON M5S, Canada, and also with Nvidia, Santa Clara, CA 95051 USA (e-mail: gururaj@cs.toronto.edu).

Digital Object Identifier 10.1109/LCA.2023.3276709

II. Background and Motivation

A. NoC Side-Channel Attacks

While attack scenarios on the NoC such as snooping are known, side-channel attacks on NoCs that exploit contention of network traffic on the interconnect have recently gained notoriety for their stealthiness. However, in most NoC designs efficient resource sharing underpins their practicality and scalability, making completely eliminating network contention implausible. NoC contention is categorized into two types:

Slice Contention. Occurs when a victim and an attacker are both sending packets to the same LLC slice. The input buffer of the destination router gets congested resulting in increased latency for the attacker's traffic.

Interconnect Contention. Occurs when a victim and an attacker are sending packets to different slices but their paths contend in the middle, causing a timing difference.

B. Prior Demonstration of Attacks

In prior attacks, the attacker profiles the latency of packets on the interconnect with and without contention and uses the timing difference to mount covert and side-channel attacks.

Lord of the Ring(s) [1]: LoTR is the first microarchitectural side channel that leverages contention on the SoC ring interconnect. The authors reverse-engineer the communication protocols on the ring interconnect and extract key bits from vulnerable EdDSA and RSA implementations, as well as keystroke events. LoTR achieves a data rate of over 4 Mbps.

MeshUp [2]: MeshUp exploits the timing difference resulting from interconnect congestion on a mesh interconnect. MeshUp generates traffic to contend with the victim's traffic on a mesh interconnect and leaks sensitive data by monitoring the contention. To recover the 2048-bit private RSA key, the paper achieved 47% chances to recover 2040 bits correctly.

Leaky Buddies [3]: Leaky Buddies is the first cross-component CPU-GPU covert channel attack. It proposes two attacks, of which one uses the contention on the interconnect to mount covert channel attacks in a heterogenous system with an iGPU. This channel delivers up to 400 kbps with a 0.8% error rate.

Don't Mesh Around [4]: Don't Mesh Around provides an improved covert-channel attack on the mesh interconnect with a capacity of over 1.5 Mbps. It also demonstrates a side-channel attack and proposes a software-only mitigation to the attack. However, the paper doesn't provide any security analysis of the mitigation or any performance numbers. We contend that a hardware-based mechanism will be faster than a software-based mechanism and will preserve binary compatibility.

C. Temporal Partitioning of NoC as a Defense

Time-Division Multiplexing (TDMA): The naive time-division multiplexing [5] approach allows packets from only one security domain to pass through the router in each cycle. The security domain allowed to propagate alternates in a round-robin fashion, isolating traffic across security domains temporally but causing large delays at each router.

Surf Scheduling: SurfNoC [6] proposes reducing the large latency of TDMA by scheduling the flits in a wave-like manner. The flits are allowed to move forward whenever they are in the wavefront corresponding to their security domain.

While they reduce the contention observed, neither entirely eliminate it. Their latency overheads are significantly worse than the baseline and scale linearly with network size. They also suffer from potential under-utilization of bandwidth since a packet has to wait for its time slot even in an idle network.

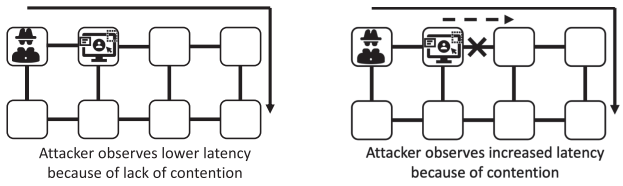
III. NoIR: NoC With Integrated Remapping

We make the observation that NoC side-channels become feasible only when an attacker can cause deliberate network contention. This assumption is met when: 1) the attacker knows the location of the victim traffic's destination slice, or 2) the attacker knows the interconnect path where contention will occur between the victim and attacker's network packets.

Even in modern CPUs such conditions can be easily satisfied. For example, most commercial Intel CPUs use static hash functions to map physical addresses (PAs) to LLC slices [10]. Hence, with a sufficient number of trials, the attacker can reverse-engineer a PA to LLC slice mapping and create a contention-based NoC side channel. These hash functions are generally the same across SoCs in the same architecture generation rendering all SoCs using that architecture vulnerable.

A. Threat Model

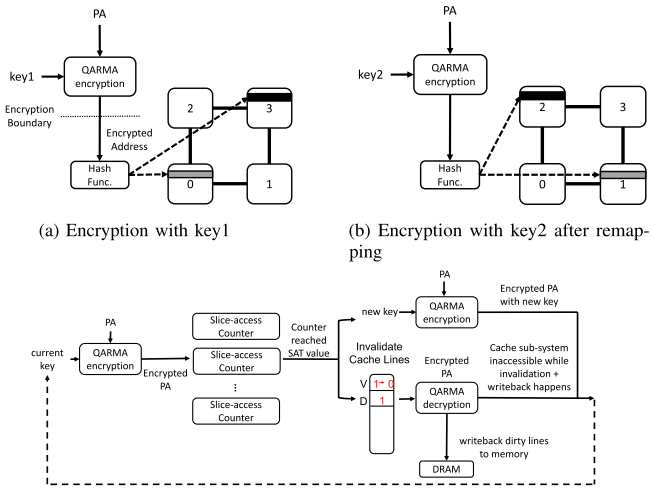
We assume the attacker has code execution capability on the same system as a victim process. The attacker and victim run on different cores, and the attacker attempts to leak information about the victim's execution through side-channels. We assume the attacker uses NoC side-channels to leak information about the victim. For instance, the attacker introduces contention on the shared on-chip interconnect with the victim, by transferring its own packets via cache accesses, as shown in Fig. 1. The basic mechanism to transmit a 0 or a 1 in an attack scenario is shown in Fig. 1(a) and (b). We assume stateful shared cache side-channels, that rely on contention for cache space (e.g., conflict-based attacks or flush-based attacks), to be defended using techniques like cache set randomization [9] and focus on the *stateless* NoC side-channel attacks.



(a) Transmitting Bit '0' in an attack scenario

(b) Transmitting Bit '1' in an attack scenario

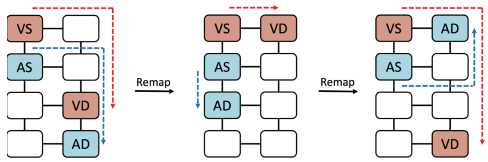
Fig. 1. Attack model with bit transmission demonstration. 'X' denotes the contended resource (LLC slice or interconnect link).



(a) Encryption with key1

(b) Encryption with key2 after remapping

(c) Flowchart showing NoIR's mechanism including cache invalidation and writeback



(d) Demonstration of obfuscation generated by NoIR (*AS=Attack Source, AD=Attack Destination, VS=Victim Source, VD=Victim Destination*)

Fig. 2. Randomized slice mappings using encryption and dynamic remapping proposed with NoIR.

B. NoIR Mechanism

We propose NoIR, a randomized mapping mechanism that maps PAs to LLC slices using encrypted addresses. We encrypt the physical addresses using a QARMA block cipher, which is a low-latency block cipher already in use in Arm CPUs for Pointer Authentication Code generation. We then use this encrypted address to decide the LLC slice. Address encryption randomizes the physical addresses (PA) as well as the output of the hash function which decides the LLC slice. We periodically change the encryption key to change the PA to LLC slice mapping after an LLC slice has been accessed a set number of times (Fig. 2(a) and (b)) This obfuscates network contention and raises the bar to orchestrate an NoC side-channel attack.

We change the encryption key after a set number of accesses at a per-slice granularity, which we shall call the *Slice Access Threshold* (SAT), by using one counter per router (since each LLC slice is co-located with 1 router). When any LLC slice has been accessed more than the SAT value, the encryption key is

changed and remapping is performed. At each remapping event, the lines in the LLC will need to be invalidated and dirty lines are written back. Fig. 2(c) and (d) demonstrate the NoIR mechanism flowchart and how an attack is obfuscated.

IV. Security Analysis

1) Analysis for Consecutive Hit Probability: We take the number of LLC slices to be equal to the number of CPUs and call it N. Thus at any remapping event, the probability that the attacker and victim destination cache-line are mapped to the same LLC slice (slice contention) is given by:

$$P(E_1) = \frac{N}{N^2} = \frac{1}{N} \tag{1}$$

Now we consider the probability that the attacker and the victim will be mapped to the same slice again in the next remapping event given they are already in the same slice:

$$P(E_2/E_1) = \frac{1}{N} \times \frac{1}{N} = \frac{1}{N^2} \tag{2}$$

Similarly, for a continuous sequence of n such remapping events, it's easy to show that the probability will be given by:

$$P(E_n/E_1 \cap E_2 \cap \cdots \cap E_{n-1}) = \frac{1}{N^n} \tag{3}$$

In our evaluations $N = 16$, hence we would have a 6.25% chance of the slices overlapping randomly. That is an improvement of 93.75% over the baseline with no remapping. The chances of the slice being remapped to fall on the same slice keep decreasing exponentially with the number of trials shown by (3) (n = number of trials). E.g., for $N = 16$, the probability reduces to 0.39% and 0.024% for 2 and 3 remapping events respectively. Fig. 2(d) shows how NoIR obfuscates a NoC side-channel attack with remapping events.

2) Analysis for NoC Traffic: Randomization of LLC to PA mapping would effectively transform any traffic pattern into uniform-random over a large enough period of time. We assume the attacker and the victim only inject attack traffic.

i: average injection rate of NoC traffic for non-attack CPUs

P_{avg}: average no. of packets reaching a slice in the network because of background traffic

This gives us:

$$P_{avg} = \frac{N-2}{N} \times i = \left(1 - \frac{2}{N}\right) \times i \tag{4}$$

For $N = 16$ and $i = 0.3$ (low-load), $P_{avg} = 0.2625$ i.e., a 26.25% obfuscation added due to background traffic.

V. Evaluation and Results

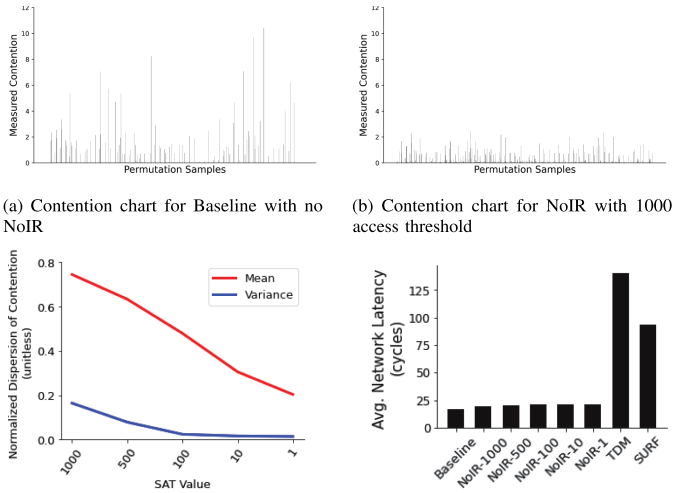
A. Experimental Setup

Network Evaluation: We use the Garnet2.0 [11] NoC simulator for our network evaluation and use a 4x4 mesh with 16 routers in a mesh configuration with XY Routing. Mathematically, we define: $Contention = \frac{latency_{attack} - latency_{baseline}}{latency_{baseline}}$

Performance Evaluation: We model the performance degradation due to the cache invalidation and writeback using the MacSim [12] microarchitecture simulator. We model the cache-flushing penalty by invalidating the cache lines and making the cache unavailable for a certain period of time. Table I summarizes the simulation configurations. We evaluate NoIR with SAT=1000, TDMA and SurfNoC all running SPEC CPU 2017 benchmarks in rate-mode.

TABLE I
MacSim Simulator Configuration for Performance Measurements

Processor	
Cores	16 cores, 3.0 GHz ,x86, out-of-order, 5-stage pipeline, 4 issue width
L1 Cache (Private)	64KB, 8-way, 64B line size, 3 cycles, inclusive
L2 Cache (Private)	256KB, 8-way, 64B line size, 8 cycles, inclusive
Last Level Cache	
LLC slice	16 slices, 2MB, 32-way, 64B line size, 30 cycles
Total LLC size	32MB
DRAM	
Bus Frequency	800MHz (DDR 1.6 GHz)
Channels/Ranks/Banks	2/1/8 , 16KB Rowbuffer



(a) Contention chart for Baseline with no NoIR

(b) Contention chart for NoIR with 1000 access threshold

(c) Mean and variance of contention vs SAT values normalized to baseline

(d) Average network latency

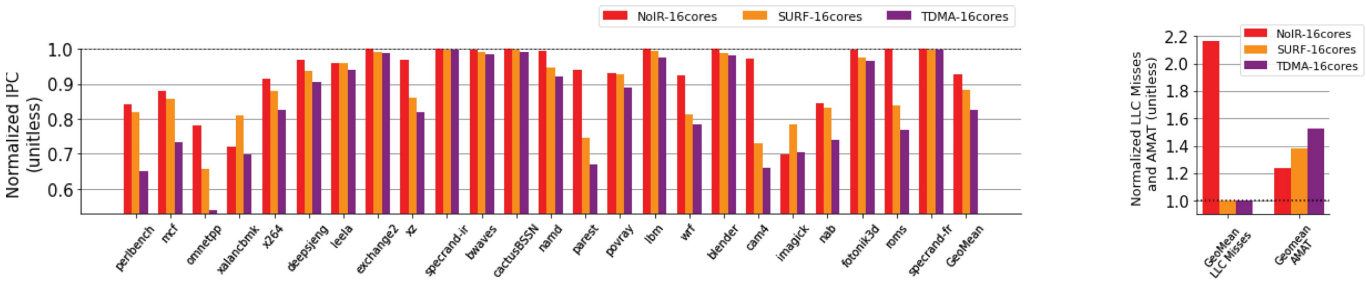
Fig. 3. Contention charts, mean and variance of contention and average network latency for NoIR.

B. Mean and Variance of Contention

We generate contention charts by injecting synthetic traffic into the NoC with two sources and two destinations for all possible permutations and measuring contention. In the contention chart for the case of NoIR disabled and NoIR enabled with SAT=1000 in Fig. 3(a) and (b), we see that the chart with no NoIR is 'peakier' while the chart for NoIR-1000 is spread out with less discernible peaks, amortizing the contention over the NoC. Our conclusion is supported by the variance plot in Fig. 3(c) where we see a 92.18% reduction in variance for SAT=1000. Using variance as a measure of dispersion allows us to demonstrate the amortization of contention using NoIR in a quantifiable manner.

C. Network Latency

The major weakness with SurfNoC was that the latency overhead was too high for practically sized NoCs. NoIR gives us a modest network latency overhead of 12% in an attack scenario(Fig. 3(d)) because unlike TDM and SurfNoC, NoIR incurs negligible packet transmission penalties only because of



(a) Normalized Instructions-per-cycle (IPC)

(b) Normalized LLC Misses and AMAT

Fig. 4. IPC, LLC Misses and AMAT evaluated for SPEC CPU 2017 benchmark and normalized against the baseline with no defenses.

the QARMA block cipher (4 cycles), which every packet will go through before being redirected to its destination slice.

D. Storage, Area and Power Overheads

Storage: For *SAT*=1000, we need a 10-bit counter per LLC slice giving us a tiny 160 bits of overhead for 16 slices.

Area and Power: We implemented the QARMA block cipher using the Verilog HDL and synthesized our design at 1 GHz clock frequency using the Synopsys Design compiler and the 45 nm FreePDK library. We estimate a 0.0717 mm^2 area overhead, dynamic power overlead of 248.84 μW and leakage power overhead of 345.78 μW, which are insignificant compared to the overall power budget of the SoC.

E. Performance Overhead, LLC Misses and AMAT

The IPC degradation measured with NoIR-1000 is 7.38% compared to the baseline, which is caused primarily by the increase in LLC miss rate as well as LLC inaccessibility due to the writebacks caused by NoIR's regular remapping. Our overheads are smaller than SurfNoC (11.70%) and TDMA (17.35%) and will scale better than both with increased core counts. The detailed results are shown in Fig. 4(a).

Fig. 4(b) shows that the LLC misses from NoIR are 2x the baseline while TDMA and SurfNoC don't show any increase. Despite that, we see that the *AMAT* (Average Memory Access Time) for NoIR is lower than TDMA and SurfNoC. This is because while NoIR incurs a much larger number of LLC misses, the LLC latency is only 4 cycles more than the baseline.

VI. Limitations and Conclusion

In conclusion, we propose NoIR, a system-level randomized LLC-slice selection technique to obfuscate network contention. NoIR amortizes contention and spreads it over the NoC raising the bar significantly against NoC side-channel attacks. We show a variance reduction of 92.18% using NoIR with *SAT*=1000, network latency overhead of 12%, and a modest performance degradation of 7.38% for SPEC CPU 2017 benchmarks.

While NoIR raises the bar significantly for a NoC side-channel attack, it is possible that an attacker targeting all LLC slices equally can still create contention on the NoC with enough trials. In that case an attacker can possibly denoise the contention and still leak out useful information. Moreover setting the proper SAT value is crucial since an attacker could make accesses below the threshold and be undetected. Extending NoIR to be completely secure remains part of our future work.

References

[1] R. Paccagnella, L. Luo, and C. W. Fletcher, "Lord of the ring(s): Side channel attacks on the CPU on-chip ring interconnect are practical," in *Proc. 30th USENIX Secur. Symp.*, USENIX Assoc., 2021, pp. 645–662. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/paccagnella

[2] J. Wan, Y. Bi, Z. Zhou, and Z. Li, "MeshUp: Stateless cache side-channel attack on CPU mesh," in *Proc. IEEE Symp. Secur. Privacy*, Los Alamitos, CA, USA, 2022, pp. 1396–1414. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP46214.2022.00081

[3] S. B. Dutta, H. Naghibijouybari, N. Abu-Ghazaleh, A. Marquez, and K. Barker, "Leaky buddies: Cross-component covert channels on integrated CPU-GPU systems," in *Proc. IEEE 48th Annu. Int. Symp. Comput. Architecture*, Valencia, Spain, 2021, pp. 972–984. [Online]. Available: https://doi.org/10.1109/ISCA52012.2021.00080

[4] M. Dai, R. Paccagnella, M. Gomez-Garcia, J. McCalpin, and M. Yan, "Don't mesh around: Side-Channel attacks and mitigations on mesh interconnects," in *Proc. 31st USENIX Secur. Symp.*, Boston, MA, USA: USENIX Assoc., 2022, pp. 2857–2874. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/dai

[5] M. Schoeberl, F. Brandner, J. Sparsø, and E. Kasapaki, "A statically scheduled time-division-multiplexed network-on-chip for real-time systems," in *Proc. IEEE/ACM 6th Int. Symp. Netw.-on-Chip*, 2012, pp. 152–160.

[6] H. M. G. Wassel et al., "SurfNoC: A low latency and provably non-interfering approach to secure networks-on-chip," in *Proc. 40th Annu. Int. Symp. Comput. Architecture*, New York, NY, USA, 2013, pp. 583–594. [Online]. Available: https://doi.org/10.1145/2485922.2485972

[7] A. Shalaby, Y. Tavva, T. E. Carlson, and L.-S. Peh, "Sentry-NoC: A statically-scheduled NoC for secure SoCs," in *Proc. IEEE/ACM 15th Int. Symp. Netw.-on-Chip*, New York, NY, USA, 2021, pp. 67–74. [Online]. Available: https://doi.org/10.1145/3479876.3481595

[8] Y. Wang and G. E. Suh, "Efficient timing channel protection for on-chip networks," in *Proc. IEEE/ACM 6th Int. Symp. Netw.-on-Chip*, 2012, pp. 142–151.

[9] M. K. Qureshi, "CEASER: Mitigating conflict-based cache attacks via encrypted-address and remapping," in *Proc. IEEE/ACM 51st Annu. Int. Symp. Microarchitecture*, 2018, pp. 775–787.

[10] Y. Yarom, Q. Ge, F. Liu, R. B. Lee, and G. Heiser, "Mapping the Intel last-level cache," Cryptology ePrint Archive, Tech. Rep. 2015/905, 2015. [Online]. Available: https://ia.cr/2015/905

[11] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha, "GARNET: A detailed on-chip network model inside a full-system simulator," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, 2009, pp. 33–42.

[12] H. Kim, J. Lee, N. Lakshminarayana, J. Sim, J. Lim, and T. Pho, "MacSim: A CPU-GPU heterogeneous simulation framework user guide," Georgia Inst. Technol., Atlanta, GA, USA, 2012. [Online]. Available: https://github.com/gthparch/macsim/blob/master/doc/macsim.pdf