Packet Header Attack by Hardware Trojan in NoC based TCMP and its Impact Analysis

Vedika J. Kulkarni, Manju R., Ruchika Gupta, John Jose, Sukumar Nandi
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati, India

ABSTRACT

With the advancement of VLSI technology, Tiled Chip Multicore Processors (TCMP) with packet switched Network-on-Chip (NoC) have been emerged as the backbone of the modern data intensive parallel systems. Due to tight time-to-market constraints, manufacturers are exploring the possibility of integrating several third-party Intellectual Property (IP) cores in their TCMP designs. Presence of malicious Hardware Trojan (HT) in the NoC routers can adversely affect communication between tiles leading to degradation of overall system performance. In this paper, we model an HT mounted on the input buffers of NoC routers that can alter the destination address field of selected NoC packets. We study the impact of such HTs and analyse its first and second order impacts at the core level, cache level, and NoC level both quantitatively and qualitatively. Our experimental study shows that the proposed HT can bring application to a complete halt by stalling instruction issue and can significantly impact the miss penalty of L1 caches. The impact of re-transmission techniques in the context of HT impacted packets getting discarded is also studied. We also expose the unrealistic assumptions and unacceptable latency overheads of existing mitigation techniques for packet header attacks and emphasise the need for alternative cost effective HT management techniques for the same.

CCS CONCEPTS

• **Computer systems organization** → Interconnection architectures; • **Security and privacy** → Hardware security implementation; • **Networks** → Network reliability.

KEYWORDS

Hardware Trojan, Network-on-Chip Security, Secured TCMP design, Trojan Impact, Packet Header Attack

ACM Reference Format:
Vedika J. Kulkarni, Manju R., Ruchika Gupta, John Jose, Sukumar Nandi. 2021. Packet Header Attack by Hardware Trojan in NoC based TCMP and its Impact Analysis. In *International Symposium on Networks-on-Chip (NOCS '21), October 14–15, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3479876.3481597

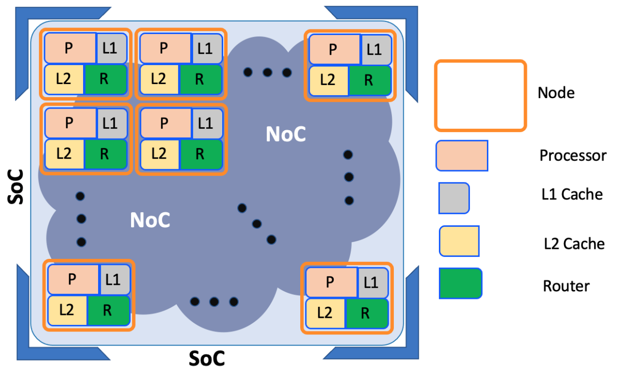


Figure 1: TCMP with Homogeneous Nodes in a Single SoC

1 INTRODUCTION

In order to reduce time-to-market and overall cost of microprocessors used in embedded systems and Internet-of-Things sectors, System-on-Chips (SoCs) started housing third party Intellectual Property(IP) blocks. Due to the high design cost, many chip manufacturing industries rely on outsourcing design automation, fabrication, and testing of integrated circuits [16]. Functional and logical security of such devices is at stake due to the involvement of untrusted third parties during various phases of chip manufacturing. Malicious circuits, known as Hardware Trojan (HT) implanted inside a genuine blueprint design can go untraceable during the verification and testing phase of SoC [12]. HTs can alter the system behavior to deploy attacks such as information leakage, unauthorized access, functional errors, and delay-of-service [8]. Some sophisticated HTs easily bypass the root-of-trust technique and stay dormant till the required input is fed. This makes it extremely difficult to trace their existence [21]. Such intermittent HTs remain active only for a short span of time, thereby HT detection becomes strenuous [17].

Tiled Chip Multicore Processors (TCMP) uses Network on Chip (NoC) for inter tile communication [2]. NoC is a packet switched framework where packets travel in the form of a series of flow control units called flits. Head flit (packet header) carries the control information required to forward the packet to its destination, while body flits carry the data. Packets between two tiles are routed through a set of intermediate routers based on an embedded routing algorithm. Fig. 1 shows a typical TCMP connecting homogeneous tiles (nodes). Modern TCMPs like Intel Xeon Phi and AMD Ryzen Threadripper have similar architectures [6].

NoC being the communication backbone of TCMP is a prime spot for mounting HT attacks. Since, NoC has access to data that travel between tiles, an HT infected router is capable of performing data corruption, stealing sensitive information and impacting QoS. Detection, localization, and mitigation of HTs in NoC has always been challenging and exhibit enormous research potential. Prior

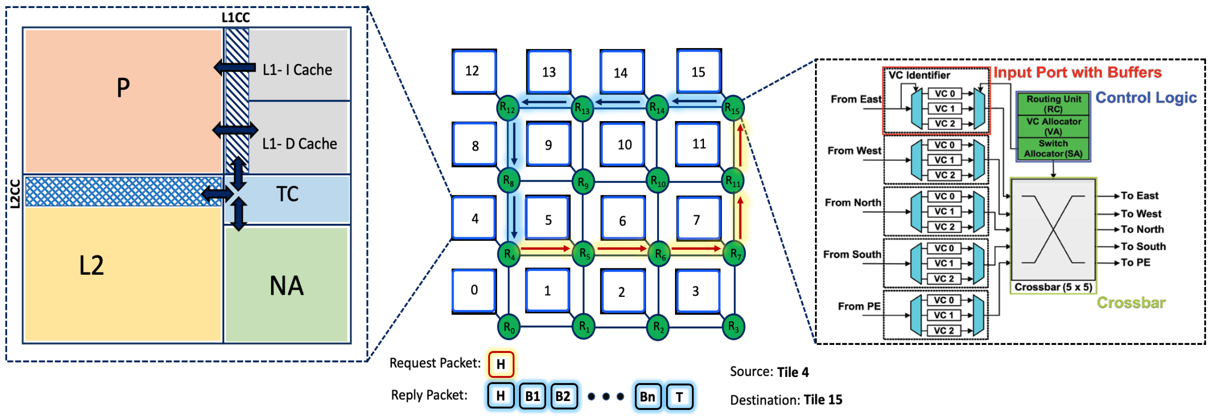


Figure 2: Internal architecture of tile and router in a 4×4 TCMP. Tile consists of Processor (P), L1 and L2 Caches, L1 Cache Controller (L1CC), L2 Cache Controller (L2CC), Tile Controller (TC), and Network Adapter (NA)

research works have outlined techniques for the detection and localization of such attacks [16],[20],[15]. Denial of Service (DoS) attack by HT causes network congestion, high communication latency, and throughput degradation [7],[18]. HTs can be localized by monitoring unusual traffic behaviour against DoS attack [5]. Bit shuffling and Error Correcting Codes are advantageous in solving HTs issue that corrupt data [11], [4]. Security zones managed by a centralized security manager can protect sensitive information stealing by malicious agents [19]. Data scrambling, packet authentication using sophisticated ciphers, and node obfuscation methods are proposed to protect data from compromised NoC [1]. Trojan aware routing can bypass an HT infected router using cost effective shielding [13]. Random forest algorithm of machine learning is used to detect DoS attack in an online mode [22].

We specifically focus our attention on HTs that can manipulate packet header. Modification of control fields in the packet by an HT for leaking packet data to malicious applications is explored recently [10]. To mitigate this type of HT, the authors suggest an authentication mechanism where control fields of the packet are tagged with a dynamic random value and the tag is scrambled with the packet data. The suggested mitigation technique that involves tag generation and key distribution among various cores make unrealistic assumptions and involves unacceptable processing time in NoC based systems. Moreover, they do not explore the HT impact in terms of vital quantitative and qualitative metrics. To the best of our knowledge, no other work explored packet header attacks in NoC based systems and conducted in depth study on the impacts created by such attacks. This motivated us to work further in this direction and we make the following contributions in this paper.

a. We identify a suitable location in the NoC router where an HT that can manipulate packet content can be inserted without violating the router's basic operation.
b. We model an HT that modifies the destination address of a packet and simulate it in a TCMP, whose effect leads to packet drop upon reaching a wrong destination tile.
c. To the best of our knowledge, we are the first to demonstrate the impact of an HT that modifies the destination address field of an NoC packet header. We conduct an in-depth study of its impact at core level, cache level, and NoC level.
d. We also model and study the impact of a packet re-transmission technique for NoC that works on the principle of Automatic Repeat Request (ARQ).
e. We prove that the existing mitigation techniques have unrealistic assumptions and unacceptable latency overheads. Hence we emphasize the need for having a cost effective mitigation technique for packet header attacks.

2 BASELINE TCMP ARCHITECTURE

Internal architecture of tile and router in a 4×4 TCMP is shown in Fig. 2. A tile consists of Processor (P), L1 Instruction Cache, L1 Data Cache, L2 Cache, L1 Cache Controller (L1CC), L2 Cache Controller (L2CC), Tile Controller (TC), and a Network Adapter (NA). L1CC and L2CC provide interface to the corresponding caches while TC act as an intermediary between the cache controllers and NA. Since, L2 cache is shared and distributed among all 16 tiles, total sets in L2 cache are uniformly partitioned across tiles in sequential fashion. NoC router consists of Input Port buffers, Routing Unit, Virtual Channel Allocator (VC Allocator), Switch Allocator, and a Crossbar.

Whenever there is an L1 cache miss, the L1CC forwards it to TC, and the TC computes corresponding L2 tile number. The request is served locally by forwarding to L2CC if the L2 tile number is same as of the tile number of TC. Otherwise, the request is forwarded to NA in order to contact the remote tile. NA, based on message type (L1 miss request, L2 miss reply, L1 write back etc) creates and forwards the packet to router attached to the tile. A miss request packet consists of a single head flit (H), whereas a reply packet consists of a head flit followed by multiple body flits and a single tail flit (H, B1, B2, ..., Bn, T). NoC follows wormhole switching where body flits follow head flit. Router computes the appropriate output port based on underlying routing technique.

We assume the flit format as shown in Fig. 3. The control channel that runs in parallel to the flit channel carries common prefixes, namely, FT and VCID for the head flit and other flits of the same packet. FT distinguishes the type of the flit. VCID is assigned for head flit specifying the VC identifier the head flit should occupy upon reaching the downstream router. VCID is then inherited by all the subsequent flits of the same packet. PID uniquely identifies a packet in the network. The address of the source tile and the

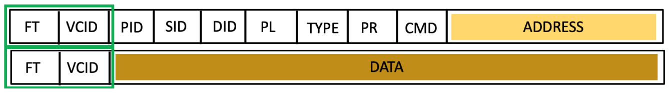


Figure 3: Head Flit and Body Flit Format

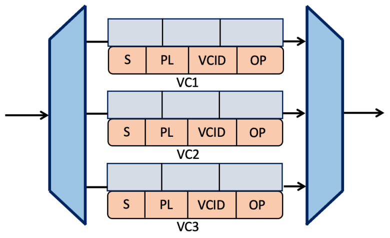


Figure 4: Internal Architecture of Input Port Buffers

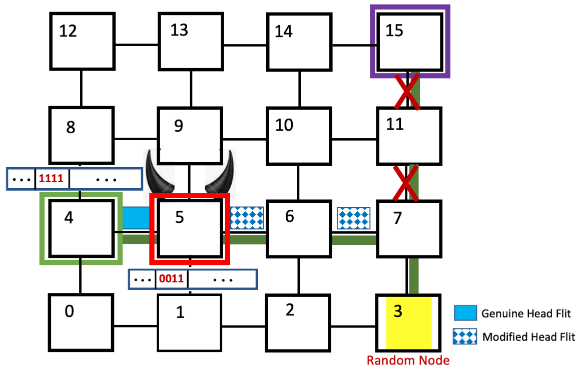


Figure 5: NoC Packet with SID=4 and DID=15 is changed to DID=3 by HT at Router 5

destination tile is given by SID and DID, respectively. PL indicates how many non-head flits are there in a given packet. Since NoC carries all categories of packets exchanged between tiles, the TYPE field specifies the packet type. Every packet is assigned a priority in the PR field, used for ranking packets during port conflicts. CMD can be utilized for storing additional metadata about a packet. ADDRESS field carries the physical address that needs to be communicated between memory levels.

Input port of a router consisting three VCs is shown in Fig. 4. Here, each VC is a FIFO and can accommodate a maximum of three flits at a time. Every VC separately maintains a control buffer consisting Status (S), Packet Length (PL), Virtual Channel Identifier (VCID), and Output Port (OP). When a flit reaches a router, the input port demultiplexer extracts the VCID from the common prefix of the incoming flit and places the flit into the designated VC. If it is a head flit, the PL field of the flit is copied to the PL of the control buffer, and S is set to busy. The route computation unit extracts DID from the head flit to compute the next outgoing port and OP is updated accordingly. Hence, once the routing is done for a head flit, then OP holds the next outgoing port information for all the subsequent flits of that packet. Therefore, even if the head flit advances to the next router, other flits still inherit OP and move forward thereby facilitating wormhole routing. When the tail flit leaves the router, the OP field get reset.

For every head flit, the VC allocator allocates a new VC based on the VC availability at the downstream router. VC availability of downstream routers is updated every cycle by credit exchange between the neighboring routers. PL gets decremented for every non-head flit arriving in the VC and PL counter reaching zero indicates end of a packet and S is set to free.

3 THREAT MODEL

Though data encryption (encrypting payload fields of body and tail flits) is gaining popularity in NoC, head flit is forwarded through the network in an un-encrypted format due to the essential requirement of its contents by the intermediate routers for taking routing decisions. This unavoidable plain text transmission of the head flit makes it vulnerable to HT attacks. In this study, we assume the HT as a small circuit mounted on the input buffers of one of the router. The HT attack, which is randomly triggered, manipulates the head flit when it is residing in the input port VC of the HT infected router. Once the OP and VCID is computed and updated by route computation and VC allocator, respectively, the HT modifies DID field of the L1-cache miss request packets. To facilitate this, the HT checks the TYPE field of the head flit and if it is an L1 miss request packet, it modifies the packet's DID to a random DID such that the new DID can be still be reached by underlying XY routing technique without any turn violations. We choose L1 miss request packet for HT attack because L1 miss penalty impacts processor throughput to a larger extent.

We take an illustrative example to explain HT and its working using NoC given in Fig. 5. We assume that the router in tile 5 is HT infected. Consider an L1 miss occurring at tile 4. It generates an L1 miss request packet, injects it into router 4 and is to be routed to destination tile 15. Therefore, the DID in the head flit is 15 (1111 in binary). As per XY routing, this packet passes through router 5. If the HT in 5 is already triggered, it modifies DID to a random router number, for instance 3 (0011 in binary). Since the HT change the DID after the routing and VC Allocation in router 5, the packet moves to 6 (original path to DID=15). However, at router 6 routing is done for DID=3. Accordingly, the packet takes a south turn at router 7, reaches router 3 and subsequently gets ejected to tile 3.

Whenever an L1 miss request packet reaches a tile, it extracts the ADDRESS field in the head flit. As per cache memory design concepts, address bits are split into Tag, Index, and Offset. The most significant bits of Index represent the destination tile address. In the mentioned example, the destination tile value calculated from ADDRESS at tile 3 does not match with DID of the head flit. Hence, the packet is dropped at tile 3 without any further processing. Moreover, the source of the packet (tile 4) expects a cache miss reply from tile 15 which never comes back, while tile 15 is unaware about such happenings.

We observe that changing DID to a random DID can cause XY routing turn violations and leads to a deadlock in due course. In the example discussed before, note that if the DID is changed to any router residing in the same or the right side column of the router 5, the packet can still reach the new DID simply by following XY routing without any turn violation. On the contrary, if DID is changed to any router in the left side column(s) of router 5, it needs 'U'-turns or prohibited turns in order to reach the new DID. For instance, if the HT modifies DID to router 8, the packet that reaches router 6 has to be forwarded back to 5 in order to reach router 8. This is a violation of XY routing leading to deadlock. To

Table 1: Workload Details of SPEC CPU 2006 Benchmark Mixes

Workload	Benchmark Name (Number of Instances)				Miss Characteristics of Benchmarks
WL1	*soplex* (8)		*cactusADM* (8)		100% High MPKI
WL2	*gromacs* (8)		*hmmer* (8)		100% Low MPKI
WL3	*soplex* (4)	*cactusADM* (4)	gromacs (4)	*hmmer* (4)	50% High MPKI, 50% Low MPKI
WL4	*soplex* (6)	*cactusADM* (6)	*gromacs* (2)	*hmmer* (2)	75% High MPKI, 25% Low MPKI
WL5	*soplex* (2)	*cactusADM* (2)	*gromacs* (6)	*hmmer* (6)	25% High MPKI, 75% Low MPKI

avoid this, we design the HT to ensure that the modified DID never leads to turn restriction violations in XY routing. This selective DID modification guarantees a deadlock free routing and makes it really challenging to detect and localize the HT at the same time. In our threat model, even though HT is always active, it gets triggered and modifies packet header only with probability *p* while stay dormant otherwise, whose implementation details are discussed in the next section.

4 EXPERIMENTAL SETUP

We model a 16-tile TCMP with a 4×4 mesh NoC using gem5 [3], an event driven and cycle accurate simulator. Each tile consists of an Out-of-Order CPU with ALPHA architecture and dynamic instruction scheduling. Two level cache hierarchy; 16 KB, 4-way set associative private L1 Instruction, Data cache each and a 2 MB, 8-way set associative shared L2 cache is used. We use an 8 GB main memory. SNUCA technique maps L2 cache sets to all the 16 tiles. Ruby module integrated in gem5 is used to simulate memory module and two level MESI protocol models cache coherence operations. NoC operations are modelled in Garnet 2.0 integrated with gem5. NoC routers use XY routing algorithm and have three VCs per input port. We use single flit request packet, 5 flit reply packets, and 64-bit flit channel. In our simulations, L1-Miss Status Holding Register (MSHR) per core can accommodate 256 entries and Reorder Buffer per core has 192 entries.
We consider the following architectures for evaluation:

- **NHT:** Baseline architecture without any HT.
- **HT:** Baseline architecture with an active HT located at router 5 having an attack probability of *p=0.1*.
- **HTR1:** Trojan model HT defined above and having an L1 cache MSHR level re-transmission using ARQ technique with timer expiry T1=1000 clock cycles.
- **HTR2:** Trojan model HT defined above and having an L1 cache MSHR level re-transmission using ARQ technique with timer expiry T2=200 clock cycles.

The timers values T1 and T2 used in re-transmission models, HTR1 and HTR2 are empirically fixed based on the experimental analysis of the worst case miss penalty for L1 miss requests that result in L2 miss and L2 hits, respectively. This paper focuses on the scenario with HT fixed at router 5. Router 5 is central router with larger traffic passing through it. This allows us to study the worst case results for HT of this nature.

It can be noted that, although the HT is at the NoC, its impact is seen on the core and cache metrics dominantly, as will be discussed in Section 5 in detail. Due to this, the impact is not very visible with synthetic traffic patterns since they do not represent real request-response traffic. Hence, we analyze HT behaviour with the real application workloads consisting SPEC CPU 2006 benchmarks [9].

Based on the nature of benchmark and its miss rate, we categorize the benchmarks into High MPKI (greater than 20 misses per 1000 instructions) and Low MPKI (less than 10 misses per 1000 instructions). Based on the experimental values of MPKI obtained [9], we pick *soplex* (38) and *cactumADM* (24) under High MPKI group together with *gromacs* (9) and *hmmer* (3) under Low MPKI group. To run a simulation on a 16-tile TCMP, an application has to be scheduled in each core for the execution. Accordingly, we create workloads, each of which consists of 16 benchmark instances. We setup five workloads classifications, namely, WL1, WL2, WL3, WL4, and WL5 as shown in Table 1. During simulation, the execution is first fast forwarded for 1,000,000 instructions followed by detailed execution of 500,000 instructions to collect the core, cache, and NoC statistics using for each of the mentioned four architectures. After the fast forwarding phase, HT in router 5 is activated for a period of 10 consecutive cycles for a window of 100 cycles while maintaining HT attack probability *p=0.1.*

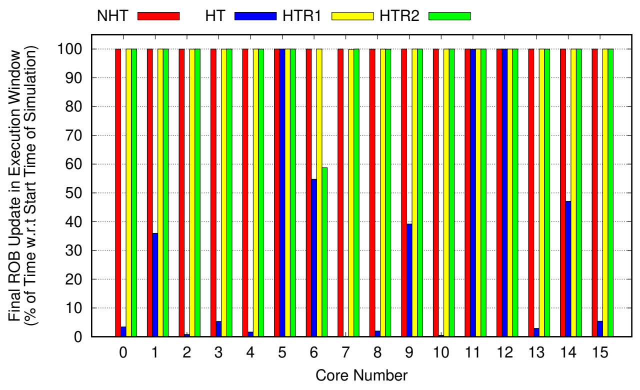
5 EXPERIMENTAL RESULTS

In this section, we analyze the impact of the packet header attack by the proposed HT using various TCMP performance metrics including instruction execution throughput, number of L1 cache misses and its average miss penalty, NoC packet classification, and average packet latency in comparison with NHT, HTR1, and HTR2 architectures.

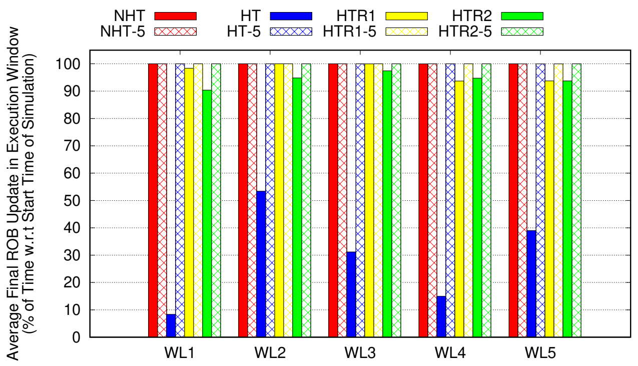
5.1 Impact on Core

Reorder Buffer (ROB) plays an important role in ensuring in-order commit of instructions in speculative dynamic scheduled out-of-order processors. ROB gives an illusion to the user that the program executes in-order. An instruction is issued only if two necessary conditions are met. (i) a free entry available in ROB (ii) a free reservation station entry in the functional unit where the operation is to be executed. At any given point in time, the ROB holds all instructions that are issued, yet not committed. A completed instruction gets committed only when it reaches the head of ROB.

LOAD instructions accessing L1-D cache may encounter a cache miss that leads to an NoC packet creation. As HT is attacking a fraction of these L1 miss request packets, there are adequate chances that some of the issued LOAD instructions with an ROB entry get affected by HT attack. The HT disallows the L1 miss request packet to reach its correct destination and consequently miss reply packet is not created. Hence, the ROB entry of HT impacted LOAD

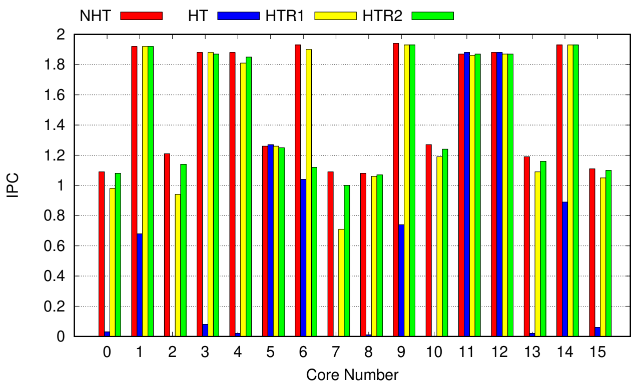


(a) Core-wise analysis for WL3

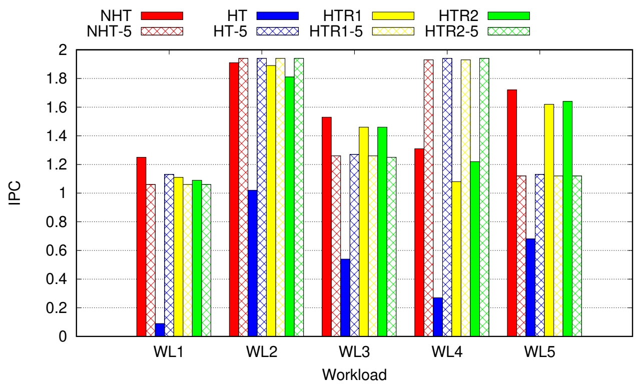


(b) Workload wise analysis

Figure 6: Comparison of final ROB update across various architectures. The height of the bar indicates when the ROB is updated last, as a fraction of instruction execution window.



(a) Core-wise analysis for WL3



(b) Workload wise analysis

Figure 7: Comparison of average IPC across various architectures

instruction remains forever in the 'issue' state and never advances to 'complete' state. It incurs an indefinite wait for Head of the ROB which in turn blocks commit of other ROB entries. There can be other subsequent instructions having data dependence over LOAD. A stalled LOAD instruction also introduces indefinite waiting for those data dependent instructions in their respective functional units to receive the data that is never written back. On the other hand, instructions that are non-dependent on LOAD are completed, however, could not commit as ROB commit is an in-order process. Consequently, instruction issue gets stalled when ROB becomes full which eventually brings down the core to a complete halt.

Fig. 6 shows the comparison of final ROB update time represented as a fraction with in the execution time window. Bar touching 100 represents ROB is updated till the end of execution indicating absence of stall during instruction issue. The final update status of ROB in various cores of WL3 is given in Fig. 6a. We observe that all cores except 5, 11, and 12 report their last ROB update at a very early stage in execution time (refer low HT bar in blue). Since HT is mounted on router 5, packets from 5 are not impacted. We also find that during the execution window, L1 miss request packets generated by tile 11 and 12 do not pass through router 5, when HT is in attack mode. Hence, ROB is not getting stalled in these cores. similarly, we also study the final ROB update statistics across all five workloads and the results are shown in Fig. 6b. Across all workloads, from the average final update of ROB across all cores over various architectures (NHT, HT, HTR1, HTR2) and the final ROB update of tile 5 across various architectures (NHT-5, HT-5, HTR1-5, HTR2-5) we can see that HT leads to stalling of application very early bringing the application to complete halt where as application in core 5 is running without any trouble.

We know that ROB level stalling leads to drastic reduction in number of instructions committed in the execution window. Instruction per Cycle (IPC) is a standard indicator used to analyze the performance of a core. We measure the IPC across all the 16 cores of WL3 and plot the results in Fig. 7a. We can notice that in NHT (architecture without any HT), applications with Low MPKI like *hmmer* running on cores 1, 4, 9, 12 and *gromacs* running on cores 3, 6, 11, 14 show high IPC (above 1.8) where as applications with high MPKI like *soplex* running on cores 0, 5, 8, 13 and *cactusADM* running on cores 2, 7, 10, 15 show low IPC (below 1.3) as

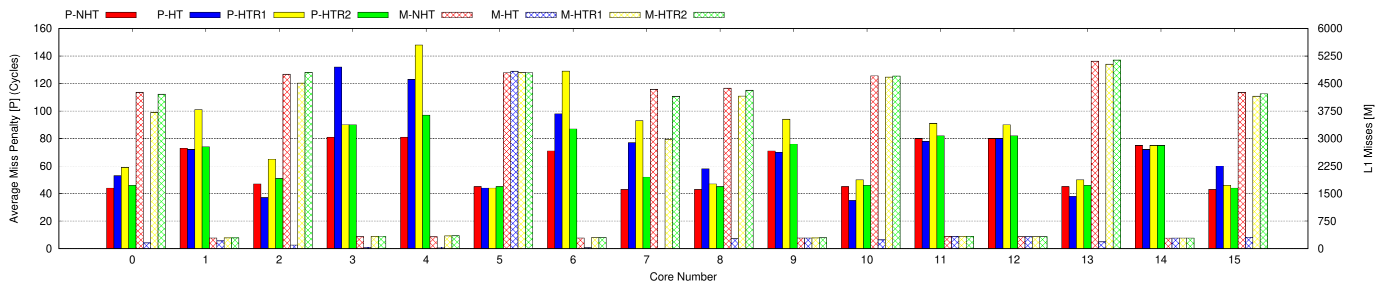


Figure 8: Core wise L1 cache miss statistics of WL3. (i) Y-axis on LHS: Average L1 cache miss penalty, (ii) Y-axis on RHS: Number of L1 cache misses generated. Bars with prefix 'P' - average miss penalty, prefix 'M' - L1 cache misses

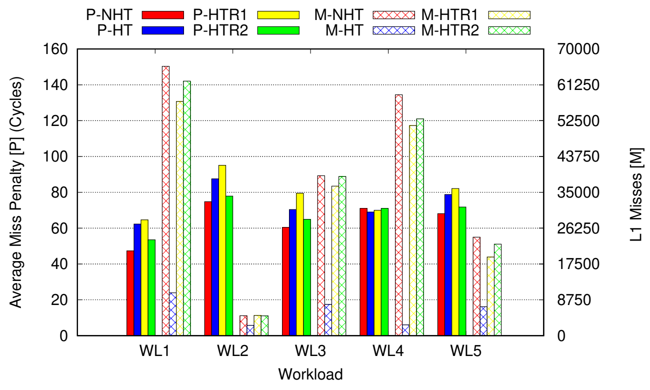


Figure 9: Average Miss Penalty for all Workloads

expected. When HT is activated these IPC values dip significantly except in core 5. But if we do re-transmissions of lost packets, we could improve IPC. We can also observe that where ever there is an ROB level stall (low HT bars in Fig. 6a) there is a a proportionate reduction in IPC (low HT bars in Fig. 7b).

Based on the experiments we conduct across various workloads, we summarize the average IPC across all cores and IPC of core 5 independently in refer Fig. 7a. The similar trend of ROB update and its correction in IPC value can be observed here too. Tile with HT infected router is benefited significantly while all other cores are kept busy waiting to receive the response against the miss request packet that is eventually dropped at a wrong destination tile. Therefore, we deduce that the DID modification by an HT mounted on an NoC router can create serious IPC impact in cores. In this way Trojan infected NoC can affect performance of cores that pass all hardware security tests.

Ideally, an NoC based TCMP is assumed to have a lossless communication network and considers no instance of an HT. Even though re-transmission is a costly approach in NoC based systems, we model the architectures HTR1 and HTR2 to show how much IPC improvement we gain if we could afford a re-transmission circuit. Because of the random nature of HT we see that almost 99% of lost packets upon re-transmission was not impacted by the HT.

5.2 Impact on Cache

We study the HT impact at cache level by analyzing L1 cache miss penalty and number of L1 cache misses generated. L1 cache miss penalty is the additional number of cycles needed to fetch the missed data block into the L1 cache from L2 or main memory. Since HT impacts only L1 cache miss packets, the infected packets are dropped after reaching the wrong destination tile. Because of this, the HT impacted request packets never reach the original destination tile and hence reply packets are not generated. This leads to ROB level stalling of instructions as we discussed before.

Fig. 8 shows core wise L1 cache miss statistics of WL3. We plot two parameters here for each of the four architectures. Y-axis on left side shows the average L1 cache miss penalty and the data is plotted with solid bars. Y-axis on right side shows the number of L1 cache misses generated and the data is plotted with dotted bars. We can see that the number of cache misses is significantly reduced in HT architecture as instruction execution itself is stopped in the core leading to zero access to D-cache. This reduction in number of cache misses is clearly visible on the cores 0, 8, and 13, where high MPKI benchmark *soplex* is running and on the cores 2, 7, 10, and 15, where high MPKI benchmark *cactusADM* is running. Cores showing higher miss penalty in NHT are those that run low MPKI benchmarks *hmmer* and *gromacs*, however they have a significant fraction of L1 misses resulting in L2 misses. We do not count miss penalty for L1 miss request that are impacted by HT and are subsequently dropped.

Similarly, we can see that when re-transmission is enabled (HTR1 and HTR2), packet stalling is reduced thereby giving chance for more instructions to execute and subsequently more cache misses are generated. However, due to the re-transmission, the average miss penalty is increased due to the high round trip latency for HT impacted miss requests. Since HTR1 uses a 1000 cycles expiry timer, its miss penalty is higher than HTR2 that uses 200 cycle expiry timer. We can also observe that the average miss penalty and number of misses in core 5 is same for all the four architectures which shows that core 5 is not impacted at all whereas all other cores suffer from cache performance degradation.

We conduct analysis across various workloads and summarize the L1 cache miss statistics in Fig. 9. For ease of reference we use solid bars to represent L1 cache miss penalty and dotted bars to represent count of L1 cache misses. The number of L1 cache misses generated in various architectures is inline with the workload characteristics discussed in the last column of Table 2. We also conduct a study on cache memory related packets in NoC and its classifications for each of the workload given in Table 3. We observe that WL2 has more L2 to main memory traffic and vice versa than other workloads which indicates that a bigger fraction of L1 cache misses reach main memory. This leads to higher miss penalty for WL2 under all four architectures and clearly observed in Fig. 9.

Table 2: Detailed Classification of L1 Miss Request Packets for all Workloads A: Total generated in NoC, B: Passing through HT Router, C: Modified by HT Router

Workload	NHT			HT			HTR1			HTR2		
	A	B	C	A	B	C	A	B	C	A	B	C
WL1	157060	11756	0	24476	363	23	135659	8609	683	147407	10460	791
WL2	14623	1109	0	7538	386	26	14601	1203	99	14327	1134	85
WL3	95500	6093	0	19741	422	24	88596	5657	428	94160	6371	497
WL4	142779	12617	0	6786	317	25	122833	9560	832	126746	8995	733
WL5	60346	4850	0	17880	428	26	48118	2501	125	56067	3984	207

Table 3: NoC Packet Classification (in %) between Memory Hierarchies

WL	L1 to L2	L2 to L1	L2 to MM	MM to L2	Others
WL1	41.9	41.9	4.4	4.4	7.5
WL2	33.3	33.3	9.5	9.5	14.5
WL3	40.9	40.9	3.9	3.9	10.4
WL4	41.2	41.2	3.3	3.3	10.9
WL5	39.9	39.9	4.3	4.3	11.7

5.3 Impact on NoC

Since our HT is designed to act on L1 cache miss request packets alone, we categorize the NoC packets in all the workloads to find out the fraction of such packets among the total NoC traffic. We comprehend our study on NoC packet classification in Table 3 and find that L1 to L2 traffic (L1 miss request packets) constitutes 33% to 42% of overall NoC traffic. This variation across workloads is due to the variation in spatial and temporal locality of benchmarks constituting various workloads. We further analyze the number of packets passing through the HT router and those that are impacted by the HT leading to DID modification in all four architectures. Table 2 summarises the findings. For each architecture, we list out three parameters listed as A, B, and C in the table. A represents total number of L1 miss request packet, B represents packets traveling through HT router, and C represents the packets that are modified by the HT router. We can clearly see that C is roughly 10% of B which confirms out HT attack probability of *p=0.1*. When re-transmission is enabled, we see the performance and number of packets getting closer to NHT model, showing that impact of HT gets nullified.

We study the average packet latency in the network and find that latency is not much impacted as the network is never getting congested. However, HTR1 and HTR2 show up to 3% increase in average packet latency due to the packet re-transmissions. We observe that even though there is no noticeable latency variations to packets at NoC level due to HT impact, we can see significant performance degradation at cache and core levels. Thus even a side channel analysis on NoC traffic will not reveal the real impact of the proposed HT there by making it more difficult to diagnose from network level parameters.

5.4 Hardware Overhead of the Trojan Circuit

To verify the proposed HT, we use ProNoC [14] that facilitates prototyping of NoC based systems. We implement the proposed HT in Verilog HDL with an integration to ProNoC for mounting the HT attack. To analyze the timing constraints RTL code is generated using ProNoC and fed to Cadence Incisive simulator for functional verification. The designs are synthesised in 90nm technology using Cadence Genus. Since the HT circuitry is not operating in the critical path, we verify that the HT mounted NoC router meets all timing constraints related to delay analysis compared to baseline router without an HT. We observe that the area overhead of the HT circuit is 0.9% of the baseline router. Power overhead is very minimal as the HT remains dormant for 90% of time.

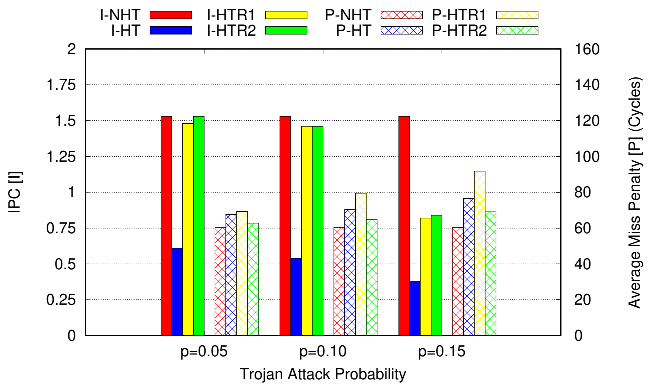


Figure 10: Impact of HT over IPC and L1 Cache Miss Penalty for varying HT Attack Probabilities for WL3

5.5 Impact on Variation in HT Attack Probabilities

All the results mentioned so far is evaluated for HT attack probability *p=0.1*. From our experiments, we find that when packets are re-transmitted it shows 99% success rate. To understand the impact on IPC and average miss penalty, we model HT with three varying attack probabilities; *p=0.05*, *p=0.1*, and *p=0.15*. We plot the results in Fig. 10. IPC and average miss penalty value is same for NHT architecture as HT attack probability is not impacting its performance. We can observe that IPC value reduces with increase in HT attack probability(refer I-HT bar) as ROB level stalling happen earlier with higher *p*. When *p=0.15*, IPC is reduced even for HTR1 and HTR2 architectures. Upon careful analysis, we notice that a small fraction of re-transmitted packets are again modified by HT and are subsequently dropped at wrong destinations leading to instruction stalling at ROB and hence lower IPC. We conclude that the success rate of re-transmission is limited to maximum HT attack probability *p=0.1*. As expected, the average miss penalty also increases with the increase in HT attack probability.

5.6 Challenges and Way Forward

NoCs are designed as lossless communication framework and techniques for handling packet loss are not implemented in conventional TCMP architectures. Since the HT forces packets reach wrong destinations, it leads to packet discarding. We have already seen the impact of such packet discard at the core, cache, and NoC levels. These HTs operating at the NoC level has the potential to stall applications running on cores that are considered to be secure. In this paper we prove that even though cores are secure, HTs at NoC can make cores vulnerable and it opens up big challenges to SoC designers. By modelling NoC level ARQ technique, we prove that 99% of re-transmitted packets are reaching their respective destinations for an HT attack probability, $p<0.1$ due to the true random nature of HT attack. However, this success rate with ARQ technique is limited to cases where the cores run low injection rate applications. We observe that due to higher packet traffic, re-transmitted packets are modified leading to application stalling for high MPKI workloads. We have not looked into circuit level implementation challenges of ARQ technique in NoC. The timer circuits and associated logic may pose higher overheads and we feel ARQ based approach may not be the best HT mitigation approach.

Mitigation techniques in recent research works [10] exploring hash based authentication and secure key distribution cannot prevent HT from modifying the packet. It can only help in the detection of packets modified by HT. Moreover, implementation of cryptographic and authentication techniques in NoC without considering the latency of such secure hardware circuits limits its acceptability. From an HT designer perspective, the attack probability should be low enough for hiding the presence of HT. At the same time, HT attack should be random and sporadic to make it difficult to detect. Attack on NoC packets other than L1 cache miss request packets and its mitigation is an interesting problem to work on.

6 CONCLUSION

This paper presented an HT attack on packet header and its impact analysis in NoC based TCMP. We first identified the location in the NoC router where the proposed HT can reside and manipulated packet header contents without violating basic router operations. We demonstrated the HT behaviour by modelling an HT that modified the DID field of packet which resulted in packet dropping at wrong destination tile. We studied the impact of the proposed HT at core level, cache level, and at NoC level. We also experimentally proved that the proposed HT mounted on an NoC router holds enough potential to degrade overall system performance by stalling applications running on cores. Our results showed that in spite of average reduction in overall system performance, the HT node alone benefited with preferential access to shared resources in the system. We evaluated the performance of ARQ based packet re-transmission to understand its impact and feasibility. We proved that due to unrealistic assumptions and expensive latency overheads the existing mitigation technique is ineffective to handle the proposed HT attack.

Employing the re-transmission circuitry alongside every core is way too expensive resort to safeguard the affected cores against such HT attacks due to excessive area and power overhead. The natural alternative is to go for a cost effective technique that involves detection, localization, and shielding which holds a promising scope as the future work.

ACKNOWLEDGMENTS

The authors acknowledge the support and funding from Information Security Education and Awareness (ISEA) Project Phase-II, MeitY, Government of India.

REFERENCES

[1] Dean Michael Ancajas, Koushik Chakraborty, and Sanghamitra Roy. 2014. Fort-NoCs: Mitigating the threat of a compromised NoC. In *Design Automation Conference*. 1–6.

[2] Luca Benini and Giovanni De Micheli. 2002. Networks on chips: A new SoC paradigm. *Computer* 35, 1 (2002), 70–78.

[3] Nathan Binkert et al. 2011. The gem5 simulator. *ACM SIGARCH Computer Architecture News* 39, 2 (2011), 1–7.

[4] Travis Boraten and Avinash Kodi. 2018. Mitigation of Hardware Trojan based Denial-of-Service attack for secure NoCs. *Journal of Parallel and Distributed Computing* 111 (2018), 24–38.

[5] Subodha Charles, Yangdi Lyu, and Prabhat Mishra. 2019. Real-time detection and localization of DoS attacks in NoC based SoCs. In *Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 1160–1165.

[6] George Chrysos. 2014. Intel® Xeon Phi™ coprocessor - The Architecture. *Intel Whitepaper* 176 (2014), 43–50.

[7] Abhijit Das, Sarath Babu, John Jose, Sangeetha Jose, and Maurizio Palesi. 2018. Critical packet prioritisation by slack-aware re-routing in on-chip networks. In *International Symposium on Networks-on-Chip*. IEEE, 1–8.

[8] Farimah Farahmandi, Yuanwen Huang, and Prabhat Mishra. 2019. *System-on-Chip Security: Validation and Verification*. Springer Nature.

[9] John L Henning. 2006. SPEC CPU2006 benchmark descriptions. *ACM SIGARCH Computer Architecture News* 34, 4 (2006), 1–17.

[10] Mubashir Hussain and Hui Guo. 2017. Packet leak detection on hardware-trojan infected NoCs for MPSoC systems. In *International Conference on Cryptography, Security and Privacy*. 85–90.

[11] Manoj Kumar JYV, Ayas Kanta Swain, Sudeendra Kumar, Sauvagya Ranjan Sahoo, and Kamalakanta Mahapatra. 2018. Run time mitigation of performance degradation hardware trojan attacks in network on chip. In *Computer Society Annual Symposium on VLSI*. IEEE, 738–743.

[12] He Li, Qiang Liu, and Jiliang Zhang. 2016. A survey of hardware Trojan threat and defense. *Integration* 55 (2016), 426–437.

[13] R Manju, Abhijit Das, John Jose, and Prabhat Mishra. 2020. SECTAR: Secure NoC using Trojan Aware Routing. In *International Symposium on Networks-on-Chip*. IEEE, 1–8.

[14] Alireza Monemi, Jia Wei Tang, Maurizio Palesi, and Muhammad N Marsono. 2017. ProNoC: A low latency network-on-chip based many-core system-on-chip prototyping platform. *Microprocessors and Microsystems* 54 (2017), 60–74.

[15] Sadia Moriam, Elke Franz, Paul Walther, Akash Kumar, Thorsten Strufe, and Gerhard Fettweis. 2021. Efficient Communication Protection of Many-Core Systems against Active Attackers. *Electronics* 10, 3 (2021), 1–31.

[16] Nachiketh Potlapally. 2011. Hardware Security in Practice: Challenges and Opportunities. In *International Symposium on Hardware-Oriented Security and Trust*. IEEE, 93–98.

[17] Manju Rajan, Abhijit Das, John Jose, and Prabhat Mishra. 2021. Trojan Aware Network-on-Chip Routing. *Network-on-Chip Security and Privacy* (2021), 277–310.

[18] Anderson Camargo Sant'Ana, Henrique Medina, and Fernando Gehm Moraes. 2021. Security Vulnerabilities and Countermeasures in MPSoCs. *IEEE Design & Test* (2021).

[19] Johanna Sepúlveda, Daniel Flórez, and Guy Gogniat. 2015. Reconfigurable security architecture for disrupted protection zones in NoC-based MPSoCs. In *International Symposium on Reconfigurable Communication-centric Systems-on-Chip*. IEEE, 1–8.

[20] Gaurav Sharma, Georgios Bousdras, Soultana Ellinidou, Olivier Markowitch, Jean-Michel Dricot, and Dragomir Milojevic. 2021. Exploring the security landscape: NoC-based MPSoC to Cloud-of-Chips. *Microprocessors and Microsystems* 84 (2021), 1–16.

[21] Kan Xiao, Domenic Forte, Yier Jin, Ramesh Karri, Swarup Bhunia, and Mohammad Tehranipoor. 2016. Hardware Trojans: Lessons learned after one decade of research. *ACM Transactions on Design Automation of Electronic Systems* 22, 1 (2016), 1–23.

[22] Jiaqi Yao, Ying Zhang, Zhiming Mao, Sen Li, Minghui Ge, and Xin Chen. 2020. On-line Detection and Localization of DoS Attacks in NoC. In *Joint International Information Technology and Artificial Intelligence Conference*, Vol. 9. IEEE, 173–178.