Handling Packet Blocking Attack in NoC using Traffic Snooping

Manju Rajan, Arya Phadke, Syam Sankar, and John Jose
Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, Assam, India

2024 IEEE International Conference on Omni-layer Intelligent Systems (COINS) | 979-8-3503-4959-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/COINS61597.2024.10622599

***Abstract*—Modern multiprocessors use Network-on-Chip (NoC) architecture for inter-core communication. As NoCs usually come as reconfigurable intellectual property blocks, malicious circuits such as Hardware Trojans (HT) in the NoC logic can alter the system's behaviour to deploy attacks like denial of service (DoS), bandwidth depletion, information leakage, etc. This paper examines a potential hardware Trojan attack on NoC routers, specifically targeting the switch arbitration phase. The proposed attack model introduces random delays for certain packets by disrupting the output of the switch arbitration phase. This imbalance in communication can result in denial of service attacks, causing performance degradation in IP cores. To counter this, we introduce a weighted delay-aware rerouting technique to detect and mitigate such HT attacks. The impact of these attacks includes increased latency and potential bottlenecks within the NoC router. Experimental results show that the latency of HT-infected packets tends to increase by 20%, and the mitigation technique can maintain performance similar to baseline.**

***Index Terms*—NoC Security, Packet Blocking, Hardware Trojan, Traffic Snooping**

I. Introduction

Modern Multiprocessor System-on-Chips (MPSoCs) devices utilise Network-on-Chip (NoC) Intellectual Property (IP) to implement on-chip communication between processing cores and various IP blocks like graphics processing units (GPUs), memory blocks, peripheral, and acceleration units [1]. Figure 1 shows an 8×8 mesh NoC. Here, the various IP blocks are linked to the NoC router through a network interface, which, upon receiving a packet (request/reply) from an IP, converts into a standardized flow control unit called flit and stores it in input buffers known as Virtual Channels (VCs). Each packet is segmented into a head flit, a set of body flits, and a tail flit. The head flit undergoes route computation, VC allocation, switch arbitration, and switch traversal phases of the router pipeline. During these phases, the head flit determines the output direction and the corresponding input VC it must use in the next hop. When multiple head flits request the same output port, the switch arbitration phase decides which head flit wins access to the output port. All body flits, which carry the data to be transmitted, follow the same routing path, implementing wormhole switching. The tail flit indicates the end of the packet [2].

The design and implementation of NoCs rely heavily on automated tools, and some NoCs support dynamic reconfiguration to optimize performance. Consequently, any malicious modifications, such as Hardware Trojan (HT) in the NoC logic, can be exploited by an adversary to alter the system behaviour that may lead to attacks like denial of service (DoS), bandwidth depletion, information leakage, etc [3] [4] [5] [6]. These malicious modifications can be introduced during manufacturing or even at design stages [7]. Even though packet delay Trojans have been proposed in the past [8], in this work, we propose the possibility of one such HT attack on NoC where the attackers can modify the output of the switch arbitration phase to create random delay for certain packets. This causes disruption in the communication balance and potentially leads to DoS attacks, which degrade the performance of IP cores. We also highlight how such HT attacks can be detected and mitigated by proper rerouting technique.

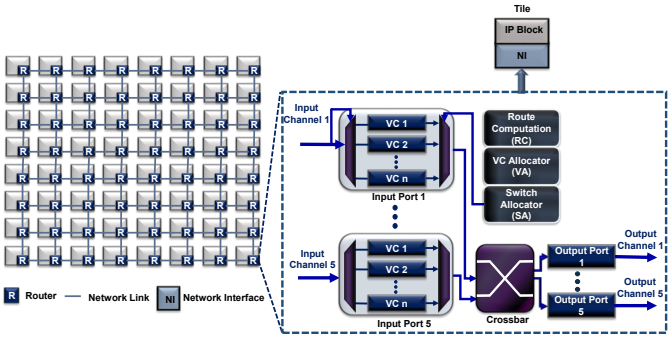


Fig. 1: 8×8 Mesh NoC

The key contributions of this paper can be summarized as follows:

- We propose a novel HT attack that targets the switch arbitration phase in NoC routers, causing random delays for packets staying in the VC buffers thereby causing disruptions in packet traversal.
- We analyze the effects of the proposed HT attack, demonstrating how it leads to increased packet latency and potential DoS attacks.
- We propose a cost-effective approach to detect such HT attacks by monitoring and analyzing anomalies in packet delay patterns using traffic snooping.
- We demonstrate an HT mitigation approach by dynamically adjusting the routing paths to maintain communication balance.

979-8-3503-4959-7/24/$31.00 ©2024 IEEE

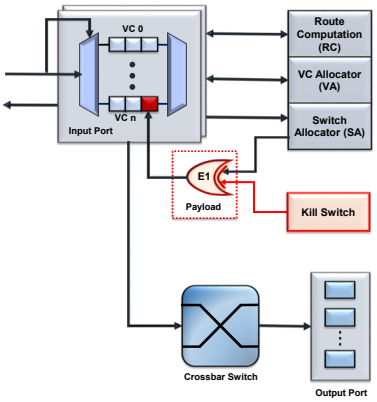


Fig. 2: Design of Arbitration Logic HT Attack Model

II. Packet Blocking Trojan Model

In an NoC router, switch arbitration (SA) is critical stage that manages multiple head flits contending for the same output port. In every cycle, the switch arbitration logic determines which flit proceeds to an output port based on predefined rules such as round-robin, priority, or quality of service (QoS) requirements. In this paper, we introduce an HT that targets the switch arbitration phase. Figure 2 shows the position of the HT circuit in an NoC router. A commonly used backdoor Kill Switch (KS) is employed to activate HT and trigger the attack [9]. This ensures that the HT remains dormant until a specific condition is met or a command is fired, making its detection more challenging. When activated, this HT randomly blocks head flits from one of the input ports that win the SA phase from further switch traversal. This blocking persists for a few random number of cycles, during which the HT continues to disrupt normal switch arbitration by blocking different flits in randomly chosen directions that win subsequent arbitration phases. This intermittent and random blocking introduces significant delays, increases packet latency, and can potentially lead to a DoS attack by continuously disrupting packet flow.

To understand the impact of the HT, consider an a mesh NoC, where a router ***R*** in it located at the center of the mesh. When head flits arrive at each input port, they are allocated VCs and proceed through the router based on the switch arbitration logic. During normal operation, the router efficiently directs these flits, maintaining low latency and high throughput. For example, assume the North input port has a flit destined for the South output port, the East input port has a flit also destined for the South output port, and the West input port has a flit requesting the North output port. The switch arbiter receives these requests and decides which flit will be granted access to each output port. Suppose the arbiter uses a round-robin scheme. The current cycle grants the North input port's request to access the South output port and the West input port's request to access the North output port. In the next cycle, the arbiter processes new requests and, following the fair arbitration scheme, might grant the East input port access to the South output port.

Now, consider a scenario where the proposed HT is triggered in router ***R***. Once activated, the HT blocks packet traversal from one of the input ports that won the SA stage, in this case, the North and West input ports. If the HT selects the North input port, it prevents the head flit in the North input port's VC from advancing to the South output port. In the next cycle, when the SA logic processes new requests, the East input port might win access to the South output port. However, the HT could randomly block the East input, preventing its flits from advancing to the South output port. This blocking continues for certain cycles, affecting different input ports as they win the SA. The HT randomly selects which input port to block among the winning ports from different directions (North, South, East, West). Once the HT goes dormant, the SA resumes normal operations. However, during the active period, the cumulative effect of the HT's random blocking leads to significant performance degradation.

The primary consequence of the proposed HT attack is increased latency for packets traversing through the affected VCs. As head flits are delayed, the rest of the packet including the body and tail flits also experience delays, leading to higher packet latency, especially for HT-infected packets. With VCs being intermittently blocked, the throughput of the NoC router decreases. The router cannot utilize its full capacity, resulting in inefficient packet flow and potential bottlenecks. Also, prolonged or frequent activation of the HT can lead to a DoS attack. Due to persistent blocking, critical data packets might be delayed beyond a threshold causing application level stalling. Moreover, the unpredictable nature of the HT activation and duration of blocking cycles can affect the system's performance.

III. Motivation

The complexity and criticality of NoC also make it a prime target for HT attacks. Specifically, HT attacks targeting the arbitration logic within NoC to delay data packets selectively can disrupt the communication balance and can affect the overall system efficiency. The selective delay of data packets can throttle the performance of essential IP cores, causing bottlenecks in the data flow and slowing down the entire system. By selectively delaying packets, attackers can effectively launch DoS attacks, making critical system components halt their operations. This can be an undesirable scenario and can be catastrophic on systems like autonomous vehicles, medical devices, and financial systems. For industries relying on reliable and secure SoC operations, such as aerospace, defense, and telecommunications, these attacks can result in costly downtime, repair, and loss of reputation.

Figure 3 shows the impact of the proposed HT in average packet latency while running a Uniform Random synthetic traffic pattern in an 8×8 mesh NoC. We can see that with HT in the active stage, even though the impact on average packet latency shows only 1%, its effect on HT-affected packets is $2.3\times$ higher than the baseline architecture with no HT. In some cases, such delayed packets might lead to missed security checks or timeout errors, allowing unauthorized entities to access sensitive system parts without being detected. For instance, encryption keys, authentication tokens,

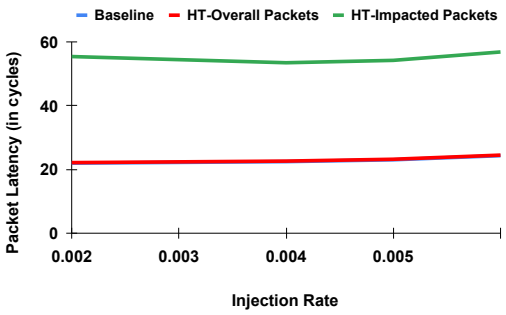


Fig. 3: HT impact in packet latency on an 8×8 Mesh NoC

and secure handshake processes require prompt data transfers. When these processes are delayed, the protocols might fail, thus weakening security policies. Another possible threat can be on autonomous vehicles that rely on a NoC to manage communications between its navigation, sensor, and control systems. Here, an HT attack that delays critical data packets could lead to sensor data delays where the vehicle might receive outdated or incorrect sensor information, impairing its ability to make real-time decisions. It can also create navigation errors, leading to misinterpretation of navigation data, causing the vehicle to take incorrect routes or fail to respond to obstacles.

IV. Related Works

DoS attack by an HT on NoC mainly aims at resource depletion, including bandwidth. The HT can deploy such an attack by flooding the network with frequent and useless packets [9] [10] in such a way that the genuine packet will never reach its actual destination or get delayed [11] [12]. Consequently, the victim packets suffer from buffer and link unavailability, delaying or halting the entire NoC communication [13]. Deadlock occurs in NoC when packets keep some resources like channels and buffers while requesting other resources. HTs in the route compute module of an NoC router can activate an attack by changing the routing algorithm. In some cases, these modifications on routing done by an HT can create deadlocks and livelocks. HTs can also misroute packets, thus denying them to reach their destination [14] [15]. The purpose of misrouting HT is to send the packets away from the destination, which can lead to DoS attacks. Such misrouting can be done by techniques like changing the packet's source or destination address to an illegal address [16], transmitting wrong information about the availability of the buffers in an NoC router, creating routing loops, etc. This behaviour of the HT makes the victim packets suffer from livelock, thus wasting the resources without any productivity. Recent research works discuss the possibilities of such HT attacks in NoC circuits that eventually tamper the quality of service of the applications running in TCMPs [17] [18] [8] [19] .

V. Trojan Detection and Mitigation

To detect and mitigate HTs that can cause packet blocking, we propose a Traffic Snoop Manager (TSM) module to be part of each NoC router. It consists of Detection and Redirection units. Once the Detection unit is active, the traffic is monitored using a weighted moving average algorithm to identify irregular packet delay patterns. When routers detect unintended packet delays, the Redirection unit is activated to mitigate the effect of the HT. The working of the TSM module is elaborated in the following sections.

Algorithm 1: Weighted Moving Average for Packet Delay Analysis

Input: Sequence of packet delays $d_1, d_2, \ldots, d_n$
Parameters: Window size w, Weights $w_1, w_2, \ldots, w_w$ **Output:** Sequence of weighted moving averages $M_1, M_2, \ldots, M_n$
for $i \leftarrow 1$ *to* n **do**
  if $i < w$ **then**
    /* Initial window calculation */
    $M_i \leftarrow \frac{\sum_{j=1}^{i} w_j \cdot d_{i-j+1}}{\sum_{j=1}^{i} w_j}$
  else
    /* Weighted moving average */
    $M_i \leftarrow \frac{\sum_{j=1}^{w} w_j \cdot d_{i-j+1}}{\sum_{j=1}^{w} w_j}$
Return $M_1, M_2, \ldots, M_n$

A. Trojan Detection Using Traffic Snooping

Once the HT detection unit is active, each NoC router continuously monitors the packet traffic within the network. This phase involves a detailed analysis of the time each packet spends at each router and helps in identifying any irregularities in the traffic flow. All NoC routers are equipped with mechanisms to observe and record traffic patterns. They specifically track and store each packet's buffering time in the router before being forwarded to the next hop. This is computed using the entry and exit times of packets at each router. Under normal operating conditions, when a packet arrives at a router, it experiences a standard processing delay of about three cycles (router pipeline delay). This standard delay includes the time required for the packet to be buffered, processed, and transmitted to the next router in the network. When the network traffic is high, VC availability is an issue that will incur additional congestion-induced delays at intermediate routers. VCs are critical for managing multiple packet flows through the same physical channel without interference. When VCs are blocked, packets may experience additional delays of a few more cycles. This delay occurs because the packet has to wait until a VC becomes available for it to proceed. However, such VC blocking is not a frequent scenario as the NoC is designed with proper traffic management techniques to minimize delays and smooth packet flow. This includes strategies like adaptive routing, load balancing, and congestion control. These mechanisms work together to ensure that all packets are fairly treated, reducing the likelihood of unusual delays even under high-traffic conditions.

Algorithm 2: Packet Redirection with Re-routing

```
1  Input: ht_dir, expected_outport, destination_y,
     current_y, current_id, num_columns
2  Output: New outport and intermediate destination
3  if ht_dir == expected_outport then
4      call ReRoutingAlgorithm()
5  ReRoutingAlgorithm()
6  if expected_outport == East then
7      if destination_y > current_y then
8          new_outport ← North
9          intermediate_dest ←
             current_id + num_columns
10     else
11         new_outport ← South
12         intermediate_dest ←
             current_id − num_columns
13 else if expected_outport == West then
14     if destination_y > current_y then
15         new_outport ← North
16         intermediate_dest ←
             current_id + num_columns
17     else
18         new_outport ← South
19         intermediate_dest ←
             current_id − num_columns
20 else if expected_outport == North then
21     new_outport ← Local
22     intermediate_dest ←
         current_id + 2 × num_columns ± 1
23 else if expected_outport == South then
24     new_outport ← Local
25     intermediate_dest ←
         current_id − 2 × num_columns ± 1
```

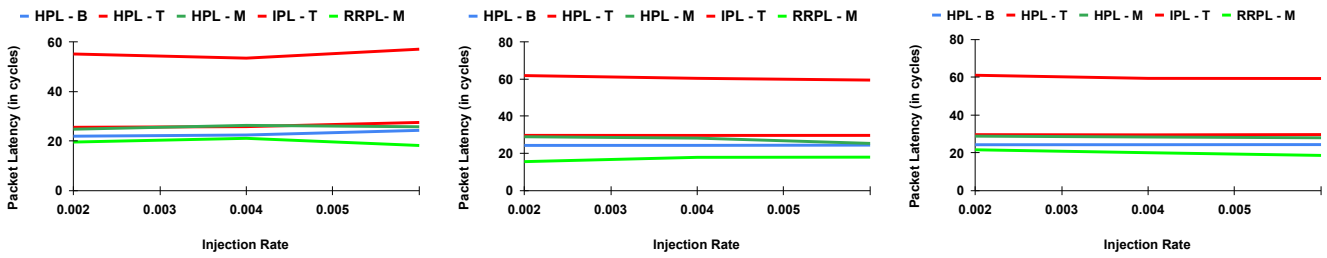
We use a weighted moving average algorithm as presented Algorithm 1, to differentiate between expected delays and potential issues. It highlights any significant deviations from normal behaviour. The algorithm calculates the average delay over a defined window of recent packet delays, giving more weight to the more recent delays. This helps in capturing the current traffic conditions more accurately. The system can detect irregular patterns by comparing the current delay against the moving average. Here, the delay analysis begins by taking a sequence of packet delays $d_1, d_2, \ldots, d_n$ as input, along with a specified window size w and corresponding weights $w_1, w_2, \ldots, w_w$. The goal is to produce a sequence of weighted moving averages $M_1, M_2, \ldots, M_n$. We use w = 5, and weights are assigned linearly. For each packet delay d_i, the algorithm calculates the weighted moving average. If the current index i is less than the window size w, it performs an initial window calculation by considering all available delays up to the current packet (Line No. 4)). For indices i greater than or equal to w, the algorithm uses a fixed window size w for the calculation (Line No. 6). Finally, the algorithm returns the sequence of weighted moving averages $M_1, M_2, \ldots, M_n$. This approach helps in smoothing out the packet delay data and identifying any irregular patterns that might indicate issues within the NoC. The use of weights allows the algorithm to give more importance to certain delays, typically the more recent ones, thus providing a more responsive and accurate analysis of the network's performance.

B. Trojan Mitigation Using Packet Redirection

Algorithm 2 represents how packet redirection is done when irregular delays are detected by an NoC router. When a router detects an unintended delay for a packet coming from an NoC router in a particular direction by analysing the weighted moving average of the time a packet spent in previous routers, it checks whether the direction of the router with the HT-induced delay (ht_dir) matches the expected outport for an incoming packet. If so, the TSM activates the redirection unit which then operates a re-routing algorithm. Consider a case for a packet where the expected outport direction is East or West. In that case, the algorithm compares the destination Y-coordinate with the current Y-coordinate to determine whether to reroute North or South, setting the intermediate destination accordingly. For North or South outport directions, the packet is redirected to the intermediate destination based on the number of columns in the NoC mesh. Unlike existing techniques such as dynamic caging [8] and shielding ring [15], our approach involves all routers detecting packet delays (both in HT neighbours and other victim routers experiencing delays due to backpressure) and rerouting packets accordingly. This rerouting ensures that packets avoid the compromised router while maintaining progress toward their final destination, thereby preventing buffer blockage and reducing congestion due to back pressure.

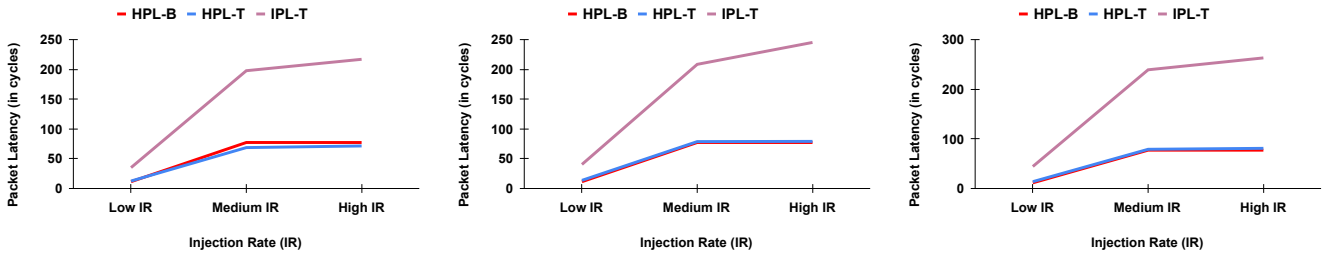
VI. Experimental Analysis and Discussion

We assess the performance of TSM by analyzing packet latency and VC utilization. We utilize the event-driven simulator, gem5 [20], to implement and evaluate the baseline system (a standard NoC without any HTs), a NoC with a HT infected router, and the proposed TSM. The garnet framework within gem5's ruby memory model is used for NoC implementation. Our baseline system consists of a traditional 8×8 2D mesh NoC, featuring five virtual channels (VCs) per input port and a 128-bit flit channel for inter-router communication, employing XY routing. To simulate the HT, we alter the switch arbitration module to ensure the presence of a single HT router in the NoC at any given time. The TSM module is integrated into garnet, incorporating all necessary micro-architectural and functional specifications. To evaluate performance and NoC-specific parameters, we execute standard synthetic traffic patterns, Uniform Random and Transpose, and Tornado with varying injection rates.



(a) Uniform Random Synthetic Traffic (b) Transpose Synthetic Traffic (c) Tornado Synthetic Traffic

Fig. 4: Impact on average packet latency while running synthetic benchmarks: Uniform Random, Transpose, and Tornado with the HT activation probability of 0.6



(a) At Probability = 0.3 (b) At Probability = 0.6 (c) At Probability = 0.9

Fig. 5: Impact on APL under different HT activation probabilities for Uniform Random synthetic benchmark at low, medium, and high injection rates.

A. Impact on Packet Latency

In NoC, average packet latency (APL) is the average time a packet travels from its source router to its destination router. This latency often includes all delays (router and queuing) the packet encounters during its journey. For evaluation, we consider the following cases for average packet latency

- HPL: APL of all packets passing through HT
- IPL: APL of all packets impacted by HT
- RRPL: APL of all packets rerouted by the neighbours due to the proposed mitigation technique.

Figure 4 presents the average packet latency metrics under different injection rates while running Uniform Random, Transpose and Tornado traffic. Here, HPL-B provides a reference point for understanding normal latency conditions without HT (baseline case). Whereas HPL-T and IPL-T highlight the latency of the system when HT is enabled. HPL-M and RRPL-M show the latency values once the mitigation technique mode is in place. For Uniform Random, as shown in Figure 4a, at an injection rate of 0.002, the presence of a HT increases the HPL-T by 16% over HPL-B. The application of the mitigation technique reduces the average latency to 24% (HPL-M). We can see that the IPL-T has 2.5 increase over the baseline, indicating the impact of HT. However, the latency for rerouted packets (RRPL - M) shows a 10% reduction compared to the baseline, demonstrating the effectiveness of the mitigation. While analysing the HPL-T of injection rates 0.004 and 0.006, we find that HPL-T is approximately 14% above the baseline. With the application of the mitigation technique, the average latency increase of HPL-M is reduced by 4% from the HPL-T. The latency for packets directly impacted by the Trojan (IPL-T) shows an average of 2.3 times increase over the baseline, indicating severe impact. However, the latency for rerouted packets (RRPL-M) shows an average decrease of 64% over IPL-T. Similarly, as shown in Figure 4b and Figure 4c, the presence of a packet blocking Trojan significantly increases packet latency across both transpose and tornado traffic patterns, with Trojan-affected latency rising by approximately 21% and impacted packet latency by 1.4x compared to the baseline. The mitigation technique effectively reduces the latency, lowering HT-affected latency by about 5% on average. Rerouted packets show a substantial latency decrease, averaging a reduction of 22% compared to the baseline.

B. Impact of Trojan Activation Probability

Figure 5 presents the likelihood of HT's impact at different injection rates with varying probability (p) of HT activation. As shown in Figure 5a, at $p = 0.3$, HPL-T increases latency by approximately 9% at low IR. However, it decreases latency by approximately 11% at medium IR and by 7% at high IR. For IPL-T, the latency increase approximately $2\times$ at low IR, $1.5\times$ at medium IR, and $2\times$ at high IR when compared to the HPL-B. At $p = 0.6$ as presented in Figure 5b, the presence of HPL-T results in an average latency increase of approximately 19% at low IR, 2% at medium IR, and 2% at high IR. Packets directly impacted by the Trojan (IPL-T) show substantial increases in latency, approximately $2.5\times$ at low IR, $1.7\times$ at medium IR, and $2.1\times$ at high IR over

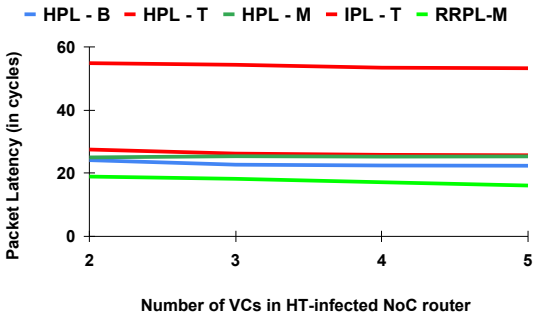


Fig. 6: HT impact on packet latency with varying VC

HPL-B. At $p = 0.9$, the Trojan consistently increases latency across all IRs (refer to Figure 5c). HPL-T shows an increase of approximately 20% at low IR, 2% at medium IR, and 4% at high IR. For IPL-T, the increases are notably severe, approximately 2.8× at low IR, 2× at medium IR, and 2.4× at high IR compared to the HPL-B.

C. Impact of VC Count Variation

Our proposed HT is located at the SA phase of the router pipeline and impacts flits residing in VC. As the VC count changes the overall impact also can vary. We study this impact by modeling routers with different VC counts. Figure 6 illustrates the APL of packets passing through HT-infected routers with varying numbers of VCs. At VC=2, the presence of the Trojan (HPL-T) increases latency to a 14% rise from HPL-B. Implementing mitigation techniques (HPL-M) reduces this latency to a 9% decrease from HPL-T with a 3.6% increase from HPL-B. Packets directly impacted by the Trojan (IPL-T) experience significantly higher latency measure 1.2× increase compared to HPL-B. In contrast, rerouted packets (RRPL-M) exhibit lower latency demonstrating a 21% decrease from HPL-B. While analysing VC=3 and VC=4, HPL-T indicates an average of 15% rise from HPL-B. Mitigation efforts (HPL-M) reduce latency and average of 3% decrease from HPL-T. IPL-T shows latency reflecting a 1.3× increase over HPL-B, while RRPL-M records latency of average 21% decrease from HPL-B. At VC=5, HPL-T shows a latency increased of 14.8% over HPL-B. HPL-M reduces this to 1.7% decrease from HPL-T. Whereas IPL-T shows a 1.38× increase over HPL-B, while RRPL-M shows latency decrease of 28% over HPL-B. Overall the presence of Trojan consistently elevates latency levels, mitigated to some extent by strategic rerouting and mitigation techniques.

VII. Acknowledgement

This work is supported by the research grant from SERB Project-CRG/2021/007400, DST, and ISEA Project Phase- II, Government of India.

VIII. Conclusion

The advancement in NoC-based systems has introduced susceptibility to HT attacks. In this paper, we presented a novel HT attack targeting the switch arbitration phase within NoC routers, aiming to introduce random packet delays. To mitigate these threats, we proposed a Traffic Snoop Manager, which detects anomalous delay patterns indicative of HT presence in the network. Upon detection, immediate actions are initiated where neighbouring routers neutralize the infected router by dynamically rerouting packets, thus preventing performance degradation in the system.

References

[1] "A comparison of Network-on-Chip and Busses," 2005.
[2] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the Design Automation Conference*, pp. 684–689, 2001.
[3] T. Boraten and A. Kodi, "Mitigation of Hardware Trojan based Denial-of-Service attack for secure NoCs," *Journal of Parallel and Distributed Computing*, vol. 111, pp. 24–38, 2018.
[4] S. Charles and P. Mishra, "A Survey of Network-on-Chip Security Attacks and Countermeasures," *ACM Computing Surveys*, vol. 54, pp. 1–36, 05 2021.
[5] D. Fang, H. Li, J. Han, and X. Zeng, "Robustness Analysis of Mesh-based NoC Architecture under Flooding-Based Denial of Service Attacks," in *International Conference on Networking, Architecture and Storage*, pp. 178–186, 2013.
[6] V. Y. Raparti and S. Pasricha, "Lightweight Mitigation of Hardware Trojan Attacks in NoC-based Manycore Computing," in *Proceedings of the Annual Design Automation Conference*, pp. 1–6, 2019.
[7] C. Pilato, K. Basu, M. Shayan, F. Regazzoni, and R. Karri, "High-Level Synthesis of Benevolent Trojans," in *Design, Automation & Test in Europe Conference Exhibition*, pp. 1124–1129, 2019.
[8] R. Gupta, V. J. Kulkarni, J. Jose, and S. Nandi, "Securing On-Chip Interconnect against Delay Trojan Using Dynamic Adaptive Caging," in *Proceedings of the Great Lakes Symposium on VLSI*, pp. 411–416, 2022.
[9] T. Boraten and A. K. Kodi, "Mitigation of Denial of Service Attack with Hardware Trojans in NoC Architectures," in *International Parallel and Distributed Processing Symposium*, pp. 1091–1100, 2016.
[10] P. Mishra and S. Charles, *Network-on-Chip Security and Privacy*. 2021.
[11] V. J. Kulkarni, R. Manju, R. Gupta, J. Jose, and S. Nandi, "Packet Header Attack by Hardware Trojan in NoC based TCMP and its Impact Analysis," in *IEEE/ACM International Symposium on Networks-on-Chip*, pp. 21–28, 2021.
[12] M. H. Khan, R. Gupta, J. Jose, and S. Nandi, "Dead Flit Attack on NoC by Hardware Trojan and Its Impact Analysis," in *Proceedings of the International Workshop on Network on Chip Architectures*, pp. 10–15, 2021.
[13] R. Manju, M. Choksey, and J. Jose, "Runtime Detection of Time-Delay Security Attack in System-an-Chip," in *IEEE/ACM International Workshop on Network on Chip Architectures*, pp. 1–6, 2022.
[14] L. Daoud and N. Rafla, "Routing Aware and Runtime Detection for Infected Network-on-Chip Routers," in *International Midwest Symposium on Circuits and Systems*, pp. 775–778, 2018.
[15] R. Manju, A. Das, J. Jose, and P. Mishra, "SECTAR: Secure NoC using Trojan Aware Routing," in *IEEE/ACM International Symposium on Networks-on-Chip*, pp. 1–8, 2020.
[16] N. Prasad, R. Karmakar, S. Chattopadhyay, and I. Chakrabarti, "Runtime mitigation of illegal packet request attacks in Networks-on-chip," in *IEEE International Symposium on Circuits and Systems*, pp. 1–4, 2017.
[17] M. Hussain, A. Malekpour, H. Guo, and S. Parameswaran, "EETD: An Energy Efficient Design for Runtime Hardware Trojan Detection in Untrusted Network-on-Chip," in *IEEE Computer Society Annual Symposium on VLSI*, pp. 345–350, 2018.
[18] R. JayashankaraShridevi, D. M. Ancajas, K. Chakraborty, and S. Roy, "Security Measures Against a Rogue Network-on-Chip," *Journal of Hardware and Systems Security*, vol. 1, pp. 173–187, 2017.
[19] S. Sankar, R. Gupta, J. Jose, and S. Nandi, "Trop: Trust-aware opportunistic routing in noc with hardware trojans," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 29, no. 2, 2024.
[20] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The Gem5 Simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, pp. 1–7, 2011.