Hardware Trojan Mitigation for Securing On-chip Networks from Dead Flit Attacks

Mohammad Humam Khan*, Ruchika Gupta*†, Vedika J. Kulkarni*, John Jose*, Sukumar Nandi*
*Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, India
†Department of Computer Science and Engineering, Chandigarh University, India

***Abstract*—With the advancements in VLSI technology, Tiled Chip Multicore Processors (TCMP) with packet switched Network-on-Chip (NoC) have emerged as the backbone of the modern data intensive parallel multi-core systems. Tight time-to-market and cost constraints have forced chip manufacturers to use third-party IPs in sophisticated TCMP designs. This dependence over third party IPs has instigated security vulnerabilities in inter-tile communication that cannot be detected at manufacturing and testing phases. This includes possibility of having malicious circuits like Hardware Trojans (HT). NoC is the likely target of HT insertion due to its significance and positional advantage from system and communication standpoints. Recent research shows that HTs can manipulate control fields of NoC packets and leads to dead flit attacks that has the potential to disrupt the on-chip communication resulting in application level stalling. In this paper, we propose run time detection of such dead flit attacks by analyzing packet movement behaviours. We also propose a cost effective mitigation mechanism by re-routing the packets around the HT infected router. Our experimental study with real benchmarks on 8x8 mesh TCMP evaluates the effectiveness of the proposed solution.**

***Index Terms*—Network-On-Chips, Hardware Trojans, Packet Flow Analysis, Dead Flit Attack, Run-time Mitigation**

I. INTRODUCTION

The continuous effort towards scaling the transistor technology to lower dimensions enabled the semiconductor industry to densely pack them in a given chip area. Over the years, this led to the emergence of powerful processors. Due to the limitations on performance scaling capability of the unicore processors, hardware architects introduced multicore processors where multiple cores run in lower frequencies and provide higher throughput with enhanced system performance. Tiled Chip Multi-core Processor (TCMP) is a class of System-on-Chip (SoC) designed for data intensive parallel systems. The multiple tiles hosted on a chip communicate with each other with the help of an interconnection network called Network-on-Chip (NoC) [1]. In a TCMP, each tile consists of an out of order processing core, private L1 cache and a slice of L2 cache (or bank). The L2 cache is equally shared among all the tiles. Modern TCMPs like Intel Xeon Phi and AMD Ryzen Threadripper have similar architectures [2]. For each L1 cache miss that gets transmitted as a request packet through underlying NoC, the requested block is fetched from its home-bank (or remote L2 tile) as reply packets.

These days TCMP systems have become more complex and in order to meet the demands and time-to-market constraints, TCMP manufacturers have started turning towards outsourcing parts of the design [3]. These practises expose the design to various security issues including possibility of having malicious circuits like Hardware Trojans (HT). HTs can alter the system behavior to deploy attacks such as information leakage, unauthorized access, functional errors, and delay-of-service [4]. NoC is the plausible candidate of HT insertion due to its exposure to overall system and communication. Moreover, during inter-tile communication, messages are routed through NoC routers, making NoC an apt choice for employing HT.

A detailed classification of HTs based on various aspects including its insertion, effect on system, activation period, location of HT and more are explored [5]. HTs can cause eavesdropping or sniffing attacks to steal information from TCMP via NoC [6] [7] [8]. Spoofing and data integrity attacks can corrupts the data travelling through NoC to produce live locks and application stalling [9] [10]. Recent works on HT impacts on packet header attacks [11] [12] do not provide any mitigation approach to handle those HTs.

This paper makes following contributions:

1) We analyse the impact of dead flit attacks caused by the modification of flit type field of NoC packets by HTs.
2) We develop a mechanism that detects the presence of such dead flits at run time.
3) We propose a cost-effective mitigation strategy for by-passing such HT routers by re-routing the NoC traffic.
4) We quantitatively evaluate the performance of our techniques and prove that the suggested technique is effective in restoring system performance.

II. DEAD FLIT ATTACK IN NoC-BASED-TCMP

Recent studies show the impact of HT attacks on modifications of control fields of NoC packets [11] [12]. Fig. 1 shows an instance of the dead flit attack in a 4×4 mesh TCMP. As per the threat model [12], the HT is a tiny circuit accessing the input buffers of a router. When a head flit enters the infected router, before routing operation is performed the flit type field is modified from head to body. In normal scenario, after the head flit gets buffered in its assigned VC, the routing unit extracts the destination ID and computes the output port and subsequently the OP field present in the control buffer is updated to facilitate wormhole routing. A body flit gets the same output port and VCID of preceding head flit. In this case the VCID and OP field are not updated as the head flit is never processed. The HT modified body flit stays in the router

978-1-6654-9005-4/22/$31.00 ©2022 IEEE

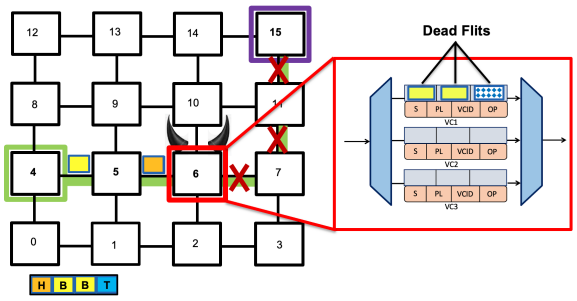

Fig. 1. Dead Flit Attack instance in a 4×4 mesh TCMP

waiting for a valid VCID and OP. Hence the body flit occupies the buffer indefinitely making itself a dead flit. Presence of dead flits cause congestion and delay in flit movements around the HT infected router.

III. DETECTION AND MITIGATION OF HT ROUTER

We propose an additional security wrapper module to facilitate detection and mitigation of HTs that create dead flits. Since the location of the HT in the NoC is unknown, every router in NoC should contain a Run-Time Mitigation Module (RTMM) as shown in Fig. 2. The run-time mitigation of HT consists of 3 phases: (1) Trojan Detection and Localization by Detection and Localization Unit (DLU); (2) Run-Time caging of HT node based on traffic congestion around the HT node by Caging Control Unit (CCU); and (3) Re-Routing of packets by Packet Re-Routing Unit (PRU) such that it reaches its destination without passing through HT routers.

A. Phase-I: HT Detection and Localization

HT detection is done by real time statistics collection of buffer occupancy of flits in routers. Here, we integrate the circuits to calculate BOP (Buffer Occupancy Period) of current router, ABO (Average Buffer Occupancy) of a flit across all routers it traveled and BOAT (Buffer Occupancy Anomaly Threshold). For smooth functioning of RTMM, we propose two more fields in the head flit structure to store BOP and ABO values. These values are used and updated by DLU to check for anomalous packets.

When a packet reaches a router, the values of BOP and ABO are extracted from head flit by DLU and the difference (BOP - ABO) is compared BOAT. $anomalous\_packet\_count$ is incremented if the difference is greater than BOAT. These packets are classified as potentially anomalous packets due to their higher buffer occupancy in immediate upstream routers compared to the average buffer occupancy across all routers they travelled. This anomaly checking procedure is done for every head flit passing through the router.

To localize the HT node, DLU uses another threshold known as the Congestion Threshold (CT) which provides a threshold for the number of anomalous packets required to declare a potentially malicious router as infected by HT. After each $time\_epoch$ (consisting of E cycles), the DLU compares $anomalous\_packet\_count$ with CT. If $anomalous\_packet\_count$ is larger than CT i.e. a significantly large number of packets are identified as potentially

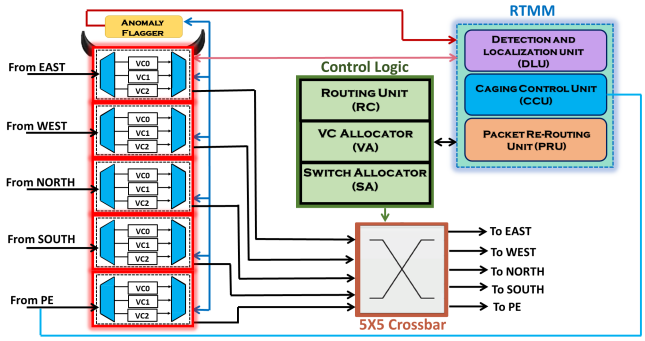

Fig. 2. Organization of NoC router integrated with proposed RTMM

anomalous in current $time\_epoch$, the DLU becomes sure that this particular neighboring router is HT infected and updates the $alert\_flag$ and $direction\_flag$ to point towards the direction of anomaly. Once the router identifies that one of its neighbors is a suspected HT, it invokes the CCU to take necessary action and initiate the cage formation to prevent the packets from passing through the HT router. The working of DLU, operation of various counters, and its association with different thresholds is shown in Algorithm 1.

Algorithm 1: Run-Time Detection of Dead Flit Trojan

Data: BOP, ABO, BOAT, CT
Result: Localization of HT Node (if any)

```
anomalous_packet_count = 0;
if (BOP - ABO) ≥ BOAT then
    anomalous_packet_count += 1
end
// After Every E cycles
if cycle_number % E = 0 then
    if anomalous_packet_count ≥ CT then
        alert_flag = 1;
        direction_flag = m_dir;
        Send MF to Neighboring Routers;
    end
    anomalous_packet_count = 0
end
```

In baseline NoC without HT, sometimes high network load can lead to VCs unavailability creating back-pressure due to which the packets passing through its neighbors can suffer an additional delay. This can be detected as anomalous packets in this region. The proposed detection mechanism ensures the elimination of these false positives by employing a continuous two-level packet monitoring with BOAT and CT. We observe that packets incurring delay due to congestion cannot sustain through two levels of threshold.

B. Phase-II: Congestion Dependent Caging of HT Node

When a particular router detects the possibility of an HT in its neighboring router, it raises the appropriate bit in $direction\_flag$ to depict the direction of malicious node with

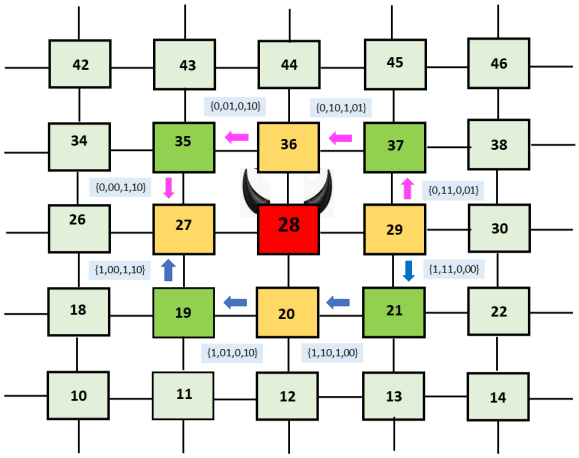

Fig. 3. Illustration of Cage Formation around HT node using Messenger Flits

respect to it and triggers CCU to initiate cage formation around the HT node using a special flit called Messenger Flit (MF) similar to Trojan Aware Routing [13]. The caging process involves informing all the 8 surrounding routers about the presence of HT. To achieve this, the router detecting the presence of HT sends two MFs containing the required information: one in clockwise direction and other in anticlockwise direction.

The MF is a special head flit containing four additional fields providing necessary information required for cage viz. $propagation\_dir$, hop_count, $cage\_router\_type$, and $alert\_dir$. $propagation\_dir$ is a 1-bit field depicts the direction of MF propagation with 0 representing anti-clockwise direction and 1 representing clockwise direction. The 2-bit hop_count field indicates the required number of hops the MF needs to be propagated further. At every intermediate router, hop_count is decremented and MF stops propagating as hop_count reaches 0. 1-bit $cage\_router\_type$ field indicates whether the router receiving the MF is a cage-corner router (0) or a cage-edge router (1). Cage-edge routers are direct neighbor of HT node. The $cage\_router\_type$ is flipped at every router before propagating the MF to the next router. The last field $alert\_dir$ is 2-bit field (with encoding '00': North, '01': South, '10': East, '11': West) indicating the direction of HT with respect to the router receiving the MF and is updated only at the cage-edge routers.

It is possible that multiple HT neighbors detect the presence of HT more or less at the same time and generate separate MF to alert the neighbors. To avoid the propagation of separate MF initiated by two different routers to cage the same HT node, a prior verification is done at CCU before propagating the MF that discards it if a received MF is alerting about the same node as HT, the current router is already aware about.

Consider a 8x8 NoC mesh with router 28 as HT infected router as shown in Fig. 3. For cage formation illustration, we assume that router 29 (east neighbor of 28) detects the presence of HT using the Trojan Detection Algorithm as described in Phase-I. Accordingly, it sets the value of $direction\_flag$ to '11' to convey that its West neighbor is malicious and triggers the CCU. In response, CCU of router 29, generates two MFs to inform all the 8 routers surrounding node 28. The first MF is forwarded to router 37 and the second to router 21. The MF travelling towards router 37 from router 29 contains four special fields as $\{0, 11, 0, 01\}$ conveying the information that MF has to be forwarded in anti-clockwise direction $\{0\}$, has to be propagated for 3 more hops $\{11\}$, the recipient router 37 is a cage-corner router $\{0\}$, and that HT is present in the South of router 37 $\{01\}$. Router 37 upon receiving MF, processes the MF in its CCU, performs required MF modifications, updates the values in the local router, and finally forwards the MF to its neighboring router in anti-clockwise direction (to router 36) with special field values as $\{0, 10, 1, 01\}$. The hop_count field is decremented from 3 $\{11\}$ to 2 $\{10\}$ and $cage\_router\_type$ is updated from $\{0\}$ to $\{1\}$ indicating that the new recipient (router 36) is a cage-edge router which is the direct neighbor of HT router. As MF moves forward, the CCU of all the routers in its path viz. 36, 35 and 27 process MF to reflect necessary changes required to register 28 as an HT router. Similarly, the second MF generated by router 29 travels in clockwise direction and reaches router 27 with field values as shown in Fig. 3.

Cage formation is an run-time adaptive process that involves a combination of both self learning and received updates. For instance, when router 36 receives an MF from router 29, one of the 3 cases can arise: Case 1: Router 36 receives MF from router 29, and it has also detected some malicious activity in router 28 itself but less than the threshold to consider it as an alarming situation. Case 2: Router 36 has already initiated MF in one of the previous cycles, and Case 3: When 36 has not yet observed any malicious behaviour in router 28.

Case 1 advances to Case 2 if 36 observes similar abnormalities from HT router in subsequent cycles, however, no other incoming packets other than coming from 30 are re-routed in Case 3. 36 only dispenses the re-routing cooperation to 29 if some abnormality from 28 is registered by it also. The cage formation completes only if all the cage-edge routers detect similar malicious behaviour of HT infected router.

The proposed caging process is dynamic in nature. It is capable of dissolve the cage after a specific Caging Time and re-cage if packet movement delay is observed again due to HT behaviour. Also, the proposed RTMM is capable of detecting HT irrespective of its location in the NoC, provided there exists only one active HT at any given point in time that does not reside on the edge router or a corner router of NoC mesh.

C. Phase-III: Re-Routing the Packet

When a packet moving in the direction of HT infected router reaches to cage-edge router, the Packet Re-Routing Unit (PRU) of RTMM determines an alternative path for the packet to reach its destination bypassing the HT router, if the cage is erected. The routing decision is taken only for the head flit and body and tail flits follow the same path as per wormhole routing. When the cage is active, the PRU of cage-edge routers block their output port in the direction of HT infected router. The Re-routing procedure for two packets P_1 (from S_1 and D_1) and P_2 (from S_2 and D_2) is shown in Fig. 4. It can

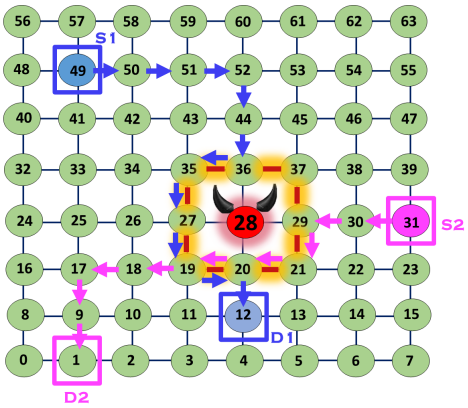

Fig. 4. Illustration of Packet Re-routing when caging is active

be noticed that the cage is active and consequently packets reaching the cage-edge routers are re-routed in both the cases of packet as shown in blue and pink colour.

Speculative re-routing leads to violation of underlying XY routing rule giving rise to a possibility of deadlocks, unless addressed at the deign level. In Fig. 4, when P_1 moves from router 49 to 36 and then to 35, it initially travels south and takes a west turn, which is against the XY routing rules and can leave the system in a deadlock. Hence, to prevent the scenario of deadlocks, the PRU uses an already established concept of intermediate destination, ejection, and re-injection [14]. Due to caging, packets are prevented from travelling through HT router and hence do not experience any additional delay due to non-availability of VCs. However, packets destined to HT or packets originating at HT node are not affected by caging. Since detection happens dynamically at run-time by observing the anomaly in waiting period of flits, few packets might get still affected by HT before caging is activated.

IV. Experimental Setup

To study the impact of our proposed mitigation technique, we use gem5 [15], an event driven cycle accurate simulator to model a 64-tile TCMP with a 8x8 mesh NoC. Each tile consists of an Out-of-Order super-scalar processor with ALPHA architecture and dynamic instruction scheduling. Each tile also houses a two level inclusive cache hierarchy consisting of a 4 KB, 4-way set associative, private L1 Instruction as well as Data caches and a shared 1 MB, 8-way set associative L2 cache with SNUCA mapping technique. We use Ruby module of gem5 with two level MESI protocol for modeling the memory hierarchy. NoC operations are modelled in Garnet 2.0 integrated with gem5. NoC routers use XY routing algorithm and have 4 VCs per input port. We use single flit request packet, 5 flit reply packets with a 64-bit flit channel.

We consider the following architectures in our study:

- **NHT**: Baseline architecture without any HT.
- **HT-HB**: Baseline architecture with an HT on any random router that modifies FT field of common prefix from head flit to body flit with an attack probability *p=0.05*.

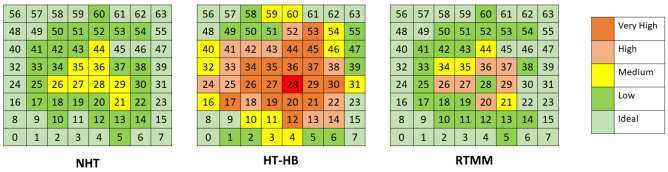

Fig. 5. Heat Map for Cumulative Buffer Occupancy

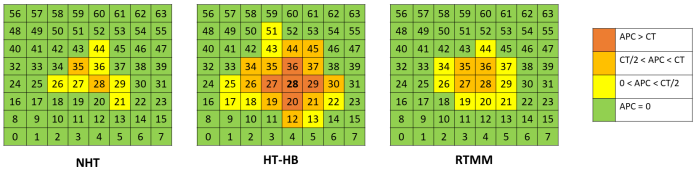

Fig. 6. Heat Map for Packet Count

- **RTMM**: Baseline architecture integrated with proposed mitigation module RTMM.

SPEC CPU 2006 benchmarks are used to evaluate the performance and effectiveness of the proposed mitigation technique. The benchmarks are categorized based on the Misses Per Kilo Instructions (MPKI) into high (more than 25 MPKI) and low (less than 20 MPKI) benchmarks. Five workloads WL1, WL2, WL3, WL4, and WL5 with different combinations of benchmarks, each consisting of 64 instances are considered as shown in Table I. This classification and subsequent analysis across various workloads help us evaluate the effectiveness of proposed mitigation technique under applications of varying NoC injection rate and cache miss behaviors. During simulation, the execution is first fast forwarded to attain warm-up followed by detailed execution to collect the relevant statistics for each of the architectures considered for study.

V. Experimental Results

We analyze the impact of the threat on an 8x8 mesh NoC and then study the effectiveness of the proposed mitigation scheme w.r.t. Buffer Occupancy Period (BOP), Average Packet Latency (APL), Hop Count, and L1 Average Miss Penalty (AMP).

A. Buffer Occupancy and Congestion Analysis

Anomalous Packet Count (APC) is defined as number of packets that are flagged as anomalous by the Detection and Localization Unit (DLU). We define two new metrics viz. Cumulative Buffer Occupancy (CBO) and Anomalous Packet Count (APC) for each router. CBO of router R_i is defined as the sum of Buffer Occupancy Period (BOP) of all the flits passing through R_i divided by total number of flits passing through R_i indicated by N[R_i]

$$CBO[R_i] = \sum_{k=1}^{N[R_i]} BOP_k \; / \; N[R_i]$$

We record CBO values for each router for WL3 and plot as a heat map considering router 28 as an HT infected router as shown in Fig. 5. It can be observed that for NHT architecture, CBO ranges from very low to medium. For the routers in the

TABLE I
WORKLOAD CHARACTERISTIC DETAILS OF SPEC CPU 2006 BENCHMARK MIXES

Workload	Benchmarks Characteristics	High MPKI benchmark instances				Low MPKI benchmark instances			
		lbm	milc	soplex	cactusADM	namd	hmmer	gromacs	libquantum
WL1	100% Low MPKI	0	0	0	0	16	16	16	16
WL2	25% High MPKI, 75% Low MPKI	4	4	4	4	12	12	12	12
WL3	50% High MPKI, 50% Low MPKI	8	8	8	8	8	8	8	8
WL4	75% High MPKI, 25% Low MPKI	12	12	12	12	4	4	4	4
WL5	100% High MPKI	16	16	16	16	0	0	0	0

center of NoC mesh, the CBO values are higher compared to other routers since network traffic is higher for center routers which leads to some delay incurred by packets passing through them. This additional delay is normal and happens due high network traffic for high MPKI benchmarks. We also observe in the heat map of HT-HB that network congestion is very high in the surrounding region of HT node. This is due to the malicious activity of HT that create delay for packet movement due to dead flits residing in VC of 28. This behaviour is anomalous and is captured by our mitigation technique to detect and localize the HT node. Consequently, it can be observed that the heat map of RTMM is almost similar to heat map of NHT case. However, there are still some routers in the neighboring region of HT node for which CBO values are higher than normal. The CBO values of routers cooperating in the re-routing mechanism are slightly higher than normal due to the increased traffic generated by re-routed packets.

Similarly, APC values are plotted as a heat map depicting variations on 8x8 NoC mesh as shown in Fig. 6. The color differences show the variation in the number of flits moving through various routers. We can observe that in NHT case some packets passing through center routers are flagged as anomalous however, enough anomalous packets are not found to mark any given router as HT. As explained before, this us due to high network traffic in the center of mesh as compared to other regions. In contrast, in HT-HB case the APC values of direct neighbor of HT router are much higher and crossed the CT. This behaviour is used by RTMM to flag router 28 as the HT node and inform other neighbors about its presence. The heat map of RTMM shows a more balanced flit flow traffic. Altogether, the heat map analysis of CBO and APC elucidates the relationship of CBO and APC values with Trojan action and utilization of these values for mitigation purpose.

B. Impact on Packet Latency and Hop Count

Since the proposed threat model blocks all the VCs in HT infected router except one, this creates congestion if multiple packets arrive simultaneously resulting in additional delay experienced by packets passing through HT router. Consequently, Average Packet Latency (APL) and Average Hop Count (AHC) of all the packets passing through the infected region (HT node and its surrounding routers) increases. The proposed mitigation technique uses re-routing that leads to packets travelling additional routers when caging is active, increasing number of hops as a consequence. Fig. 7 shows a comparison of APL and AHC for different workloads as

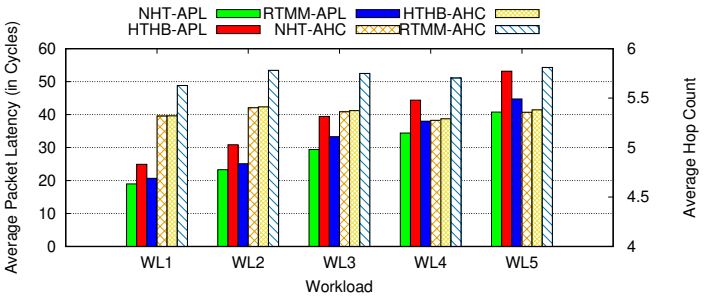

Fig. 7. Comparison of APL and AHC for all Workloads

mentioned in the Table 8. Solid bars represent APL for all three architectures NHT-APL, HTHB-APL and RTMM-APL considered in experimental study. Similarly, check bars viz. NHT-AHC, HTHB-AHC and RTMM-AHC show the Average Hop Count for different workloads. We observe that APL values have increased drastically for HT-HB compared to baseline NHT architecture. This is due to additional delay that packets passing through the infected region suffer due to Trojan action. An increase in Packet Latency of these packets consequently increases the overall packet latency of the workload. It can be noted from the plot that bars corresponding to RTMM-APL are lower than HTHB-APL which shows the effectiveness of our proposed mitigation technique. However, they are still higher than the baseline case i.e. NHT which is due to the fact that some of the packets are re-routed which may take few more additional hops and cycles to reach the destination. It can also be observed that as we move from WL1 to WL5, the APL also increases. This is because the percentage of high MPKI benchmarks in the workloads increase from WL1 to WL5. Due to this the network traffic increases leading to a natural increase in packet latency as well as queuing latency.

Average Hop Count (AHC) for NHT and HT-HB is almost same for all the workloads as observed in Fig. 7. Due to HT effect even though the packets passing through the infected region suffer additional delay however the hop count in both the cases are same as the packets are moving in the same path. The observed slight difference in NHT and HT-HB is due to the difference in number of packets that are able to reach their destinations in the simulation time. The AHC for RTMM has increased comparatively. This is due to the fact that re-routed packets have to travel through additional routers or a possible non-minimal route determined by the re-routing algorithm leading to an increase in hop count for these packets, hence increase in AHC.

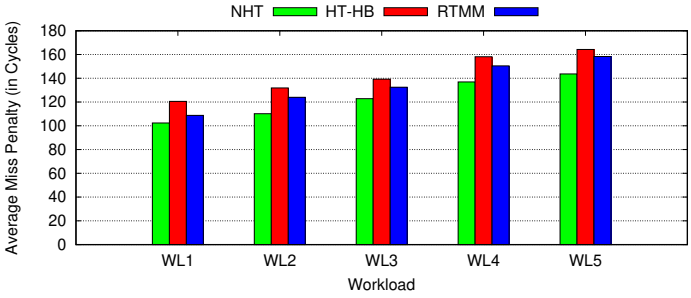

Fig. 8. Comparison of AMP for all Workloads

C. Impact on Average Miss Penalty

We analyze the impact of HT and investigate the performance of proposed mitigation technique at cache level using L1 cache miss penalty. L1 cache miss penalty is the number of extra cycles required to fetch the missing data block into the L1 cache from L2 cache or main memory. This is equal to the time spent by an outstanding miss request in MSHR. L1 request-response packets and L2 request-response packets are involved to fulfill L1 and L2 miss demands. Fig. 8 shows the variation of Average Miss Penalty (AMP) values across all the five workloads for different architectures. As we move from WL1 to WL5, the ratio of high MPKI benchmarks in the workload increases leading to more NoC packets creation and hence more impacted packets due to trojan resulting into increased height of AMP bars. Additionally, note that the proposed RTMM technique lowers down the AMP bars across all the workloads demonstrating the effectiveness the mechanism. The RTMM bars are slightly higher than the baseline (NHT) as the re-routed packets travel through a non-minimal path adding to the the packet latency which consequently increase the AMP for the cache misses. Hence, a portion of decrease in AMP due action of RTMM is compensated by an increase in AMP of the re-routed packets.

D. Overhead Analysis

We use ProNoC [16] that facilitates prototyping of NoC based systems for the implementation of the proposed RTMM module. We implement both the HT behaviour of dead flit creation and RTMM module in Verilog HDL integrated to ProNoC. To analyze the timing constraints, RTL code is synthesised in 90nm technology using Cadence Genus. Since the RTMM circuitry is not operating in the critical path, we verify that the proposed NoC meets all timing constraints related to delay analysis compared to the baseline router without an RTMM. The area and power overhead of the RTMM circuit is 1.8% and 2.3% of the baseline router, which is intuitive expected as all packets now undergo the BOP and ABO verification.

VI. Conclusion

This paper presents a run-time self-learning based mitigation technique for dead flit attacks on NoC-based-TCMPs. The proposed RTMM module assures that NoC routers can mitigate the impact of HT that causes dead flits by detecting, localising, caging, and re-routing packets travelling via HT infected routers. We experimentally demonstrated the effectiveness of the proposed mitigation technique using various performance metrics like Average Packet Latency, Cumulative Buffer Occupancy, Anomalous Packet Count, and Average Miss Penalty of L1 cache misses. Through this paper We re-emphasize the need for additional security modules like RTMM to safeguard the NoCs from malicious HT attacks.

Acknowledgement

The authors acknowledge the support and funding from DST-SERB Core Research Grant [CRG/2021/007400] and ISEA Project Phase-II, MeitY, Government of India.

References

[1] L. Benini and G. De Micheli, "Networks on chips: A new SoC paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, 2002.

[2] G. Chrysos, "Intel® xeon phi™ coprocessor - The Architecture," *Intel Whitepaper*, vol. 176, pp. 43–50, 2014.

[3] N. Potlapally, "Hardware security in practice: Challenges and opportunities," in *International Symposium on Hardware-Oriented Security and Trust (HOST)*. IEEE, 2011, pp. 93–98.

[4] F. Farahmandi, Y. Huang, and P. Mishra, *System-on-Chip Security: Validation and Verification*. Springer Nature, 2019.

[5] S. Moein, J. Subramnian, T. A. Gulliver, F. Gebali, and M. W. El-Kharashi, "Classification of hardware trojan detection techniques," *2015 Tenth International Conference on Computer Engineering & Systems (ICCES)*, pp. 357–362, 2015.

[6] D. M. Ancajas, K. Chakraborty, and S. Roy, "Fort-nocs: Mitigating the threat of a compromised noc," in *Proceedings of the 51st Annual Design Automation Conference (DAC)*, 2014, pp. 1–6.

[7] T. Boraten and A. K. Kodi, "Packet security with path sensitization for nocs," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 1136–1139.

[8] 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 *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2018, pp. 345–350.

[9] J. Sepúlveda, A. Zankl, D. Flórez, and G. Sigl, "Towards protected mpsoc communication for information protection against a malicious noc," *Procedia computer science*, vol. 108, pp. 1103–1112, 2017.

[10] Q. Yu and J. Frey, "Exploiting error control approaches for hardware trojans on network-on-chip links," in *2013 IEEE international symposium on defect and fault tolerance in VLSI and nanotechnology systems (DFTS)*. IEEE, 2013, pp. 266–271.

[11] V. JK, M. R, R. Gupta, J. Jose, and S. Nandi, "Packet header attack by hardware trojan in noc based tcmp and its impact analysis," in *2021 15th IEEE ACM International Symposium on Networks on Chip (NOCS)*. IEEE/ACM, 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 14th International Workshop on Network on Chip Architectures (NoCArc)*, 2021, pp. 10–15.

[13] R. Manju, A. Das, J. Jose, and P. Mishra, "Sectar: Secure noc using trojan aware routing," in *2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*. IEEE, 2020, pp. 1–8.

[14] A. Das, S. Babu, J. Jose, S. Jose, and M. Palesi, "Critical packet prioritisation by slack-aware re-routing in on-chip networks," in *2018 Twelfth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*. IEEE, 2018, pp. 1–8.

[15] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, "The gem5 simulator," *ACM SIGARCH computer architecture news*, vol. 39, no. 2, pp. 1–7, 2011.

[16] A. Monemi, J. W. Tang, M. Palesi, and M. N. Marsono, "ProNoC: A low latency network-on-chip based many-core system-on-chip prototyping platform," *Microprocessors and Microsystems*, vol. 54, pp. 60–74, 2017.