Securing On-chip Interconnect against Delay Trojan using Dynamic Adaptive Caging

Ruchika Gupta
Dept. of CSE, IIT Guwahati, India

Vedika J. Kulkarni
Dept. of CSE, IIT Guwahati, India

John Jose
Dept. of CSE, IIT Guwahati, India

Sukumar Nandi
Dept. of CSE, IIT Guwahati, India

ABSTRACT

With the progressive innovation of VLSI technology, Tiled Chip Multicore Processors (TCMP) have surfaced up as the backbone of the modern data intensive parallel multi-core systems. Network-on-Chip (NoC) is considered as the most preferred choice for on-chip communication. Manufacturers have begun to investigate the prospects of using third-party IP in sophisticated TCMP designs due to strict time-to-market limitations. The inflated reliance over third party IPs induced security vulnerabilities in inter-tile communication. In this paper, we implement a novel Hardware Trojan (HT) called as Delay Trojan (DT) placed in an NoC router. Proposed DT adds random delay to flits going through it, while other NoC routers merely experience regular congestion, making DT detection difficult. As a result, packets of latency-critical applications stalls impacting system performance and throughput. Further, we propose a dynamic adaptive learning framework embedded in NoC routers that detects DT with reasonable accuracy and alerts neighboring routers. We also propose a caging technique to re-route packets. Our experimental study evaluates the impact of DT and the effectiveness of the proposed solution.

CCS CONCEPTS

• **Computer systems organization → Embedded hardware**.

KEYWORDS

Hardware Trojan, Congestion, Re-routing, Adaptive Caging

ACM Reference Format:
Ruchika Gupta, Vedika J. Kulkarni, John Jose, and Sukumar Nandi. 2022. Securing On-chip Interconnect against Delay Trojan using Dynamic Adaptive Caging . In *Proceedings of the Great Lakes Symposium on VLSI 2022 (GLSVLSI '22), June 6–8, 2022, Irvine, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3526241.3530333

1 INTRODUCTION

Multi-core technology was created to improve system performance by allowing multiple processes to execute simultaneously on multiple cores. Over the last decade, Tiled Chip Multicore Processors (TCMP) has emerged as the most promising design framework for high performance computing systems. Network-on-chip (NoC), the communication backbone of TCMP, is a packet switched framework in which data is packetized and sent through the network as a series of flow control units called flits. NoC consists of switching units called routers interconnected by bidirectional links. Figure 1 depicts a typical TCMP architecture connecting homogeneous tiles. A tile consists of a processor, private L1 caches, and a slice of shared distributed L2 cache [1]. Communication between multiple memory levels contributes to the major share of inter-tile messages travel as NoC packets. A cache miss request packet contains a single head flit, while a cache miss reply packet contains a single head flit, numerous body flits, and a single tail flit (H, B1, B2, ..., Bn, T). NoC follows wormhole switching where B follows H. The router calculates the relevant output ports for each incoming packet.

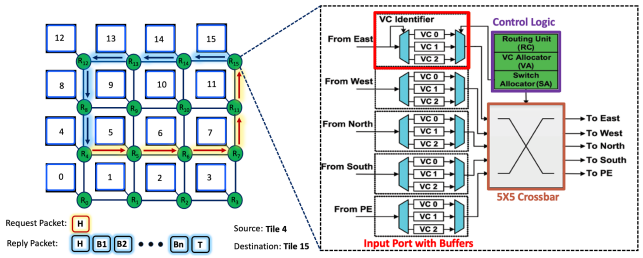


Figure 1: 4x4 mesh NoC based TCMP

Network congestion is one of the major factors affecting system performance. Multiple packets competing for the same output port in a router [2] [3] results in the buffering of the packets that lose switch arbitration. Such port conflicts increase the packets buffer occupancy leading to the formation of congestion hot-spots. Congestion causes delay as packets now take longer than usual to reach destinations. If packet delay of the latency critical applications exceeds a specific threshold, it can lead to the application level stalling and timeout. Congestion at one router not only affects packets passing through that router, but also creates back pressure over other neighboring routers. This can impact packets travelling through the neighbors of congestion hot-spots. VoIP, real-time video streaming, IPTV, Video on Demand, Video Conferencing, and multi-player games are examples of latency sensitive applications that can get impacted by NoC level congestion [4].

Outsourcing IP to third-party intellectual property (3PIP) providers has a potential security risk. NoC is the likely target of HT insertion in 3PIP since it has total system visibility from both a system and communication standpoint. Furthermore, during inter-tile communication, messages are routed through NoC routers, making

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
GLSVLSI '22, June 6–8, 2022, Irvine, CA, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9322-5/22/06…$15.00
https://doi.org/10.1145/3526241.3530333

them an apt choice for employing Hardware Trojans (HT). Due to wide range of impact NoC routers are prime target for attacks in TCMPs [5] [6]. Possibilities of HT attacks at NoC level are gaining attention [7] [8]. HTs can alter the system behaviour by mounting various attacks such as leakage of critical data, unauthorised access, delay-of-service, and even system failure. Study of threat impact, associated challenges, and proposing techniques for the detection and localization of such attacks is gaining popularity in NoC domain [9] [10].

In this paper, we implement an HT in NoC router called as Delay Trojan (DT), which introduces additional random delay to flits passing through it. Additional flit delay keeps virtual channels (VCs) engaged for longer periods of time, causing network congestion. Proposed DT creates effects similar to the congestion in NoC routers that makes it hard to detect and challenging to mitigate. As a result, latency-critical application packets incur delay, causing stalls and eventually deteriorates system performance and throughput. To the best of our knowledge, modeling and impact analysis of DT have not been explored before. We make the following contributions:

a. We design and implement a novel DT that intermittently delays flits held in the input buffer of NoC routers.
b. We model an instance of DT in an 8X8 mesh TCMP and study its impact at core, cache, and NoC level.
c. We propose a dynamic adaptive learning framework embedded in NoC routers that detects DT with reasonable accuracy and alerts neighboring routers also for subsequent remedial measures.
d. We also propose a caging technique to re-route packets around routers responsible for creating abnormal packet delays either by a DT or by genuine congestion hot-spots.

2 DELAY TROJAN ATTACK IN TCMP

Existing studies in the literature assess the variation in congestion caused by a variety of underlying routing algorithms and traffic patterns. Average packet latency can be increased up to a factor of 15 due to congestion [11] [12]. Motivated by these findings, we propose a malicious implant known as Delay Trojan in an NoC router that introduces a random intermittent delay to selected packets passing through it. We assume that at any given point in time there exists at most one DT infected router in the NoC. While there maybe possibility of multiple DTs in the NoC, our threat model is limited to a single DT active at a time. In case there are multiple DTs, that would have noticeable effect on the performance which will not help the DT to remain active unnoticed. In our threat model, even though the DT is considered to be always active, it mount attacks on the packets only with the probability p.

The proposed DT is a tiny circuit that can attack packets when they reside in the input port buffer of the infected router, similar to the one shown in input buffer of Figure 2. The DT facilitates this by blocking the control signals that trigger the route computation activity, which disables the routing operation. After a random number of delay cycles, the control signal is activated again, causing the delayed packet to be routed and arbitrated, and the packet to be sent to its destination. The DT increases the miss penalty of critical L1 cache misses by creating a congestion in the path of a few random cache miss request packets. For the HT attack, we choose the L1 miss request packets since the L1 miss penalty has a greater influence on the performance.

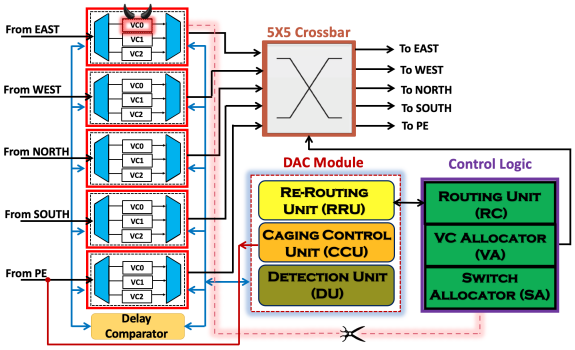


Figure 2: NoC router with the proposed DAC module

3 TROJAN DETECTION AND MITIGATION

Since the mitigation module is unaware about the DT location, every router in the NoC has a proposed Dynamic Adaptive Caging (DAC) module to detect, localize, and eliminate the impact of potential DT. DAC technique consists of three phases: Trojan Detection by Detection Unit (DU), Dynamic Caging by Caging Control Unit (CCU), and Packet Re-routing unit (RRU) that takes care of the alternative route such that packet reaches its destination. The interaction among various units of DAC with other components of router is shown in the Figure 2.

3.1 Phase 1: Trojan Detection

The presence of DT is detected by a Delay Comparator (DC) associated with the input ports and DU that coordinates it. DAC proposes a minor modification in the head flit by adding two fields; *Average Time spent per Router* (ATR) and *Time spent in Previous Router* (TPR), updated in the head flit by DC as it leaves the router.

DT detection is done with the help of ATR and TPR. The DU contains a *delay_counter*, *alert_counter*, and a 4- bit flag *flag_NSEW* to represent a direction. When a new packet arrives, the ATR and TPR values are extracted from the head flit and processed by DC. If the difference (TPR-ATR) is larger than a specified *anomaly_threshold*, DC updates the DU to increment the *delay_counter*. We call such packets as anomaly packets. The comparison of ATR and TPR and subsequent updation (if any) of *delay_counter* takes place for every incoming head flit. This process helps to comprehend the number of packets experiencing significantly larger delay in the previous router alone when compared to the average delay experienced across all routers they traveled so far.

At the end of a *time_epoch*, DU checks if *delay_counter* is larger than the *counter_threshold*. If so, *alert_counter* is incremented, else it is reset. In short, *alert_counter* is incremented only if there is a significant number of anomaly packets identified inside a *time_epoch*. Next step is to check whether the *alert_counter* is incremented in every subsequent *time_epoch* indicating the presence of many packets getting delayed in the previous router. If such behaviour is not noticed for at least one *time_epoch*, *alert_counter* is cleared and we conclude that the previous router is genuine. However, if

alert_counter exceeds an *alert_threshold*, DU identifies the neighbor as a prime suspect of DT and the appropriate direction flag in *flag_NSEW* is set. Once the router identifies that one of its neighbors is a suspected DT (as one of the bits in *flag_NSEW*=1), it invokes the CCU to take necessary action and initiate the creation of cage to protect the packets from entering the DT router. The working of DU, operation of various counters, and its association with different thresholds is shown in Algorithm 1.

Due to the blocked VCs creating congestion, leads to back-pressure and DT radiates delay effect over the packets passing through its neighbors while boosting the possibility of finding anomaly packets in that region. This is a case of false positives. Our mechanism ensures the elimination of false positives by employing a continuous multilevel packet monitoring with *anomaly_threshold*, *counter_threshold*, and *alert_threshold*. Packets incurring delay due to the back-pressure and usual NoC congestion cannot sustain through three levels of thresholds, hence sets up the *flag_NSEW*.

Algorithm 1: DT Detection

```
Input  : Incoming packet information ATR and TPR
Output : Alert with DT node detection
T: Time Epoch Length;
ANT: anomaly_threshold; CNT: counter_threshold; ALT: alert_threshold
ICT: Initial Counter Threshold; m_dir: Input Port Direction;
flag_alert: Alert Flag; flag_NSEW: Alert Direction;
/*Delay Comparator */
if (TPR − ATR) > ANT then
    delay_counter = delay_counter + 1;
/*At the end of T cycles */
if delay_counter > CNT then
    alert_counter = alert_counter + 1;
    if alert_counter ≥ ALT then
        flag_alert = 1; flag_NSEW = m_dir;
        Send MF to neighbors
        alert_counter = 0; CNT = ICT;
    else
        CNT = CNT/2;
else
    alert_counter = 0; CNT = ICT;
delay_counter=0 /*RESET at the end of every time interval */
```

3.2 Phase 2: Dynamic Adaptive Caging

When a router detects the possibility of a DT in its neighboring router, it raises the appropriate bit in *flag_NSEW* and triggers CCU to initiate dynamic adaptive caging process. CCU uses a special flit called *messenger_flit* (*MF*) to build a cage around the DT router. The caging process involves informing all 8 of the surrounding routers about the HT. To facilitate this, the router detecting DT sends two *MF* containing required information; one in clockwise and other in anticlockwise direction. The presented caging approach is slightly inspired from the HT shielding mechanism [13] [14].

Apart from the mandatory fields of a head fit, the *MF* additionally consists of four fields; namely, *propagation_dir*, *hop_count*, *alert_flag*, and *alert_dir*. 1-bit *propagation_dir* denotes the direction of *MF* propagation; clockwise (1) and anticlockwise (0). A 2-bit *hop_count* indicates required number of hops the *MF* needs to be propagated further. At every intermediate router (cage-corner routers) *hop_count* is decremented and *MF* propagation stops as it reaches 0. 1-bit *alert_flag* indicates whether the router receiving

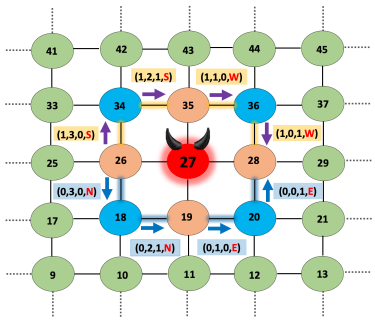


Figure 3: Illustration of exchange of *messenger_flits* (*MF*) to build a cage around DT in router 27.

the *MF* is a cage-edge router (1), or a cage-corner router (0). Cage-edge routers are always the direct neighbor of DT. The *alert_flag* is flipped every time before propagating the *MF* to the next router. The last field *alert_dir* indicates the direction of DT with respect to the router receiving the *MF* and is updated only at the cage-edge routers.

We show an illustrative example using Figure 3 to explain the process of erecting a cage in an 8x8 mesh NoC assuming router 27 as DT infected router. Let 26 be the west neighbor of 27 detecting DT using trojan detection mechanism discussed in last section and sets the 'E' bit of *flag_NSEW*. Thus, 26 identifies its east neighbor as DT. The CCU of 26 generates two *MF* to inform all of the 8 routers surrounding 27. The first *MF* is forwarded to 34 while the second is forwarded to 18. Note that *MF* comprises of four special fields and accordingly the value {1, 3, 0, S} is sent from 26 to 34 interpreted as follows. The *MF* has to be forwarded in the clockwise direction {1}, propagated for another 3 more hops {3}, the recipient 34 is a cage-corner router for the cage {0}, and DT is in the south of 34 {S}. 34 upon receiving the *MF*, get it processed in its CCU, perform necessary updates in local router, make required modifications in *MF*, and forwards to its neighbor in clockwise path with final field values of {1, 2, 1, S}. The *hop_count* field is decremented from 3 to 2 and *alert_flag* is flipped from 0 to 1 indicating new recipient 35 is a cage-edge router and the direct neighbor of DT. The CCUs of 35, 36, and 28 now process the *MF* and reflect necessary updates to register 27 as a DT router. Similarly the other *MF* initiated by 26, propagates through 18, 19, 20 and finally reaches 28 with the field values shown in Figure 3. Thus, the DU and CCU together pass essential information to all 8 surrounding routers and erects a cage such that movement of packets through DT can be stopped.

Cage building is an adaptive process that involves the comparison of the self learning and received learning before cage formation. For instance, 19 receiving *MF* can have one of the three possible cases. Case 1, when 19 receives *MF* from 26 and also has its own observation about experiencing the abnormal behaviour by 27 but less than required threshold count. Case 2, when 19 has already initiated *MF* in one of the previous cycles, and Case 3, when 19 yet not register any discrepancy from 27. The first case advances to second if 19 observes the similar abnormalities from DT router in subsequent cycles, however, no other incoming packets other than coming from 26 are re-routed in the third case. 19 only dispenses the re-routing cooperation to 26 if no abnormality from 27 is registered

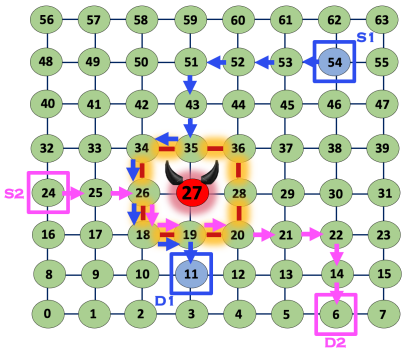


Figure 4: Illustration of Packet Re-Routing

by it. The cage formation completes only when all the cage-edge router experiences the similar behaviour from DT node.

The DT is intermittent in nature and caging design is capable of releasing the cage after a specific *Caging Time* and re-cage if DT behaviour continues. Proposed DAC is capable of detecting DT irrespective of its location, provided there exists only one active DT at any given point in time and not resides at an edge or a corner router. It is also possible that multiple DT neighbors detect the DT presence more or less at the same time and generate separate *MF*. Due to the propagation of separate *MF* initiated by two different routers to cage the same DT, we perform a cross verification at CCU before propagation that discards it if a received *MF* is alerting the same DT the current router is already aware about.

3.3 Phase 3: Packet Re-Routing

RRU of DAC module handles such packets by facilitating an alternate travel path to destinations. The active caging status and *flag_NSEW* helps a router to distinguish the output port leading to DT router that must be disabled. Accordingly, if the computed and disabled output port of a packet is same then the packet is re-routed by assigning a new output port.

Re-routing for two packets P1 and P2 is shown in Figure 4. We notice that the cage is erected and accordingly packet reaching the cage-edge routers (26, 35, 19, and 28) is now re-routed. Consider an instance of P1 traveling from S1 to D1, upon reaching 35 is assigned south output port which is disabled by the CCU, henceforth, to perform re-routing either west or east is chosen randomly to forward P1. P1 is marked as re-routed packet such that the downstream routers facilitate the re-routing instead of applying XY routing. Likewise P1 follows the path shown in blue color and finally reaches D1. This is a classical example of re-routing leading into a non-minimal alternative path for the packet.

Consider another case when P2 moves from S2 to D2 and upon reaching cage-edge router 26, the computed and disabled output port is come out to be same, hence P2 is assigned with south port. Other cage routers recognises P2 as a re-routed packet assist the re-routing as shown with pink arrows in the figure. At router 18, computed output port is distinct from input port usual routing is employed and the P2 follows rest of the shown path to reach D2. Due to caging, packets are prevented from travelling through DT and hence do not experience any trojan induced delay. However, packets destined to DT are not affected by caging. Similarly, packets generated by DT tile also remain unaffected. Since detection happens dynamically at run-time by observing the anomaly in packet delays, few packets might get still affected by DT before caging is activated. However, since routing decision is taken only for the head flit and all other flits of the packet follows the same path causing no violation of wormhole routing. Speculative re-routing leads to underlying XY routing violation giving rise to a possibility of deadlocks, unless addressed at the deign level. For example, when P1 moves from router 43 to 35 and then to 34, it initially travels south and takes a west turn, which is against the XY turn rules and can leave the system to a deadlock. To prevent deadlock, we use an already established concept of intermediate destination, ejection, and re-injection [15].

4 EXPERIMENTAL SETUP

We use the event-driven cycle accurate simulator gem5 [16] to model a 64-tile TCMP with an 8×8 mesh NoC. Every tile consists of an Out of Order CPU with ALPHA architecture and dynamic instruction scheduling. The memory system consists of a 4 KB, 4-way set associative private split (Instruction and Data) L1 cache, and a 1024 KB, 8-way set associative shared distributed L2 cache. We fix 12 GB of main memory for our system. We use Ruby memory model, MESI two level cache coherence protocol and Garnet framework for NoC that uses XY routing. We have 3 VCs per input port for a 64-bit flit channel. We use a single flit request packet and 5 flit reply packets.

We compare the performances of the following architectures:

- **BL**: A baseline NoC without any HT.
- **DT**: An NoC with one DT in a randomly chosen router as proposed in Section II with an attack probability of *p= 0.15* and induced delay= 128 cycles.
- **DAC**: An NoC with proposed DAC module mentioned in Section III.

To evaluate performance of the above architectures, we use SPEC CPU 2006 benchmarks. Based on the Misses Per Kilo Instructions (MPKI), we categorize benchmarks into high (more than 24 MPKI) and low (less than 18 MPKI) benchmarks considering latency sensitive applications are running on cores. The High MPKI benchmarks group consists of *lbm*, *milc*, *soplex*, and *cactusADM* . The Low MPKI benchmarks group contains *namd*, *hmmer*, *gromacs*, and *libquantum*. For experimentation, we set up five workloads with different combinations of benchmarks as shown in Table I. After fast forwarding the execution to the region of interest, DT and DAC are activated, and statistics are collected. We define region of interest as set of nine routers including DT and its eight surrounding routers.

5 EXPERIMENTAL RESULTS

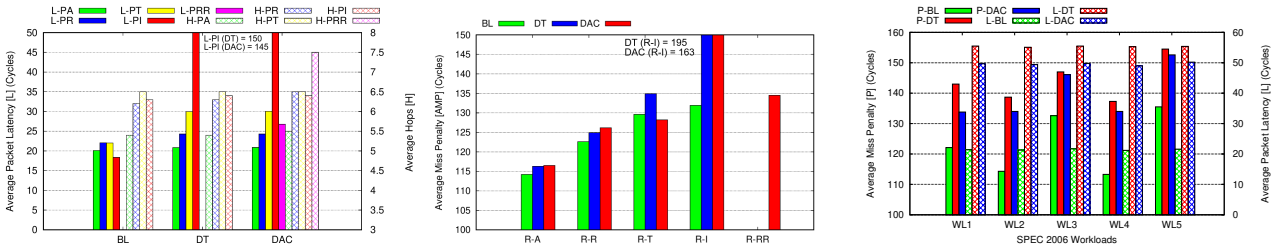
We analyse the impact of DT on NoC and also evaluate the performance of our proposed mitigation system based on the metrics namely; average packet latency, average miss penalty, average hop count, and buffer occupancy at each router.

5.1 Impact on Packet Latency and Hop Count

Since our DT is attacking flits of the packets traveling through the NoC, we study its impact at NoC level by collecting average packet latency and average number of hops. Figure 5 shows the

Table 1: Workload Characteristic Details of SPEC CPU 2006 Benchmark Mixes

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



(a) Comparison of APL and Hops for WL1

(b) Comparison of AMP for WL1

(c) Comparison of APL and AMP across all Workloads

Figure 5: Performance Statistics (Average Packet Latency - APL, Average Miss Penalty - AMP and Hops) for all Workloads

comparison of average packet latency and the hop count for the packets traveling through the NoC for baseline (BL), NoC with a DT and an NoC with caging based mitigation approach using WL1 mentioned in Table 1. The solid bars represent latency values and pattern bars represent hop counts. We notice five solid bars, indicating average latency of all the packets passing through the NoC (L-PA), packets passing through the DT router alone (L-PT), packets passing through the region of interest (L-PR), packets that are impacted by DT (L-PI), and packets that are re-routed by the DAC mechanism (L-PRR). Similarly, in terms of average hops too we define H-PA, H-PT, H-PR, H-PI, and H-PRR.

The effect on L-PA in BL and DT is same since trojan delays less than 1% of total packets in the NoC. As expected, due to delay of movement of packet in DT and subsequent back pressure created by it, L-PR and L-PT are slightly higher in DT and DAC than in BL. We observe that packets passing through the region of interest experience a slightly higher delay on average because of the created congestion. However, the HT effect is clearly visible on the impacted packets (refer L-PI bar of DT and DAC), with an increase of 800% as compared to BL. When we apply DAC mitigation, we can see that the latency comes down and is close to that of BL. It clearly demonstrates the effectiveness of our proposed detection and mitigation system. The L-PRR of DAC is slightly higher than L-PA of BL because when we apply re-routing, the packets need to travel additional hops to bypass the DT router. We also conduct analysis for all workloads given in Table 1.

Average hops comparison for WL1 is also shown in Figure 5a. The average number of hops is close to 6 in each case, except for the re-routed packets (H-PRR) in DAC where it is around 7.5. This is due to the additional hops and possible non-minimal route some packets take due to DAC's re-routing algorithm. This shows the efficiency of DAC's re-routing that enables a packet to reach the destination without much delay.

The pattern bars in figure 5c shows the comparison of Average Packet Latency (APL) across all five workloads. Irrespective of the workloads, we observe that APL is same. This is because of the fact the DT is not distinguishing the packets based on its source and number of NoC packets created by that source which is the MPKI property of the benchmark. However, we can notice that the miss penalty bars (solid bars) shows variation across the workloads discussed in next subsection.

5.2 Impact on L1 Cache Miss penalty

Average Miss Penalty (AMP) of an L1 request is calculated as the time taken for the request to receive its reply at the L1 cache. This is the amount of time an outstanding miss entry stays in the Miss Status Handling Register of L1 cache. This includes the latency of L1 miss request packet, access time of L2 cache, and latency of miss reply packet. If this L1 miss results in an L2 miss, additional round-trip delay over NoC from L2 to main memory and main memory access time has to be added. AMP of all L1 requests (R-A), L1 requests through the region of interest (R-R), L1 requests through the trojan router (R-T), impacted L1 requests (R-I), and rerouted L1 requests (R-RR) is compared across BL, DT, and DAC in Figure 5b. The R-A and R-R in DT architecture (blue bar) is only slightly higher than that of respective value in BL (green bar) that too due to the congestion created by DT. However, when we compare the R-T value we see that the difference increases and the effect is clearly visible in R-I where DT causes the AMP to rise to 150% of its value in BL. The effectiveness of DAC is observed in R-RR, DAC restores the AMP of rerouted packets close to the BL value.

The comparison of AMP across all packet classifications is shown in Figure 5c for all five workloads. The solid bars in figure 5c shows the comparison of AMP across all five workloads. We can see that DAC lowers the AMP across all workloads. Since WL5 consists of high MPKI benchmarks, it generates more NoC packets leading to more number of impacted packets and hence high AMP bars.

At the same time, the AMP bars in WL4 are relatively low due to the presence of low MPKI benchmarks. As expected, the AMP bars of WL1, WL2, and WL3 exhibit proportional rise based on the percentage of high MPKI benchmarks in the workload.

5.3 Buffer Occupancy and Flit Flow Analysis

We collect the Cumulative Buffer Occupancy (CBO) and Cumulative Flit Count (CFC) of all flits passing through a router. We plot CBO and CFC as a heat map for WL1 when DT is present in router 27 as shown in Figure 6. Variation in buffer occupancy and flit count is shown in a range; green (very low) to red (ultra high). In terms of CBO, more the colour is on red side, more the number of cycles flits stayed in buffer of the respective router, whereas in terms of CFC, more red colour shows more number of flits traveled through the respective router. As expected, the center of the mesh indicates higher CBO and CFC even in BL as flits travel through center region more. The DT router stands out as the only red node in Figure 6b, whereas it changes to green in Figure 6c by the effective caging that blocks packets to DT. The orange colour effect is registered due to the back pressure caused by congestion in Figure 6b. Moreover, we observe that the caging do not put much pressure on the buffer stay on the surrounding routers. The CFC analysis in Figure 6f exhibits heavy flit traffic around the DT router and it shows that our caging performs re-routing correctly. Overall the heat map study with CBO and CFC comparison gives us a clear picture of the delay vs flit flow relation in the proposed DT and its mitigation technique.

5.4 Overhead Analysis

We implement the proposed DT and DAC module in Verilog HDL and integrate to ProNoC [17] that facilitates prototyping of NoC to verify our designs. The designs are synthesised in 90 nm technology using Synopsys DC. Since neither DT nor the DAC module operates in the critical path, the NoC with DT and DAC can be operated at the same frequency as that of BL. DAC circuit uses counters and flags incurring an additional storage of 62 bits per router. We estimate additional area and power incurred by the DT and DAC module using Orion 3.0 and find that there is an area overhead of 0.06% for DT implementation and 1.6% for the DAC module with respect to the BL while power overhead is up to 0.75%.

6 CONCLUSION

This paper introduced the HT attack possibility that incurred delay to selected packets. The delay created is comparable to the one caused by normal congestion such that the neighboring router do not suspect anything abnormal. We demonstrated that the proposed DT located on the NoC router exhibited the ability to degrade the system performance by stalling the cores running latency sensitive applications. We also suggested a threshold statistics-based detection and localization technique, as well as a dynamic adaptive caging mechanism that created a cage around the DT router. We then employed a packet re-routing method to bypass the DT router. We are able to recover the system's performance at a minimum cost in terms of average packet latency, average miss penalty, area and power overhead. Measuring system performance for a variety of impacted message types, thresholds, and workload combinations holds a potential future work to examine.

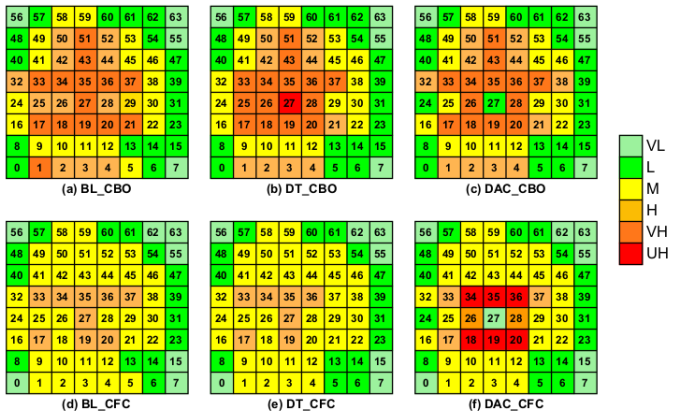


Figure 6: Heat Map Analysis for Cumulative Buffer Occupancy (CBO) and Cumulative Flit Count (CFC)

7 ACKNOWLEDGEMENT

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

REFERENCES

[1] G. Chrysos, "Intel® xeon phiTM coprocessor - The Architecture," Intel Whitepaper, vol. 176, pp. 43–50, 2014.
[2] S. H. S. Rezaei, A. Mazloumi, M. Modarressi, and P. Lotfi-Kamran, "Dynamic resource sharing for high-performance 3-d networks-on-chip," IEEE Computer Architecture Letters, vol. 15, no. 1, pp. 5–8, 2015.
[3] Z. Shi and A. Burns, "Real-time communication analysis for on-chip networks with wormhole switching," in Second ACM/IEEE International SymposiumonNetworks-on-Chip(nocs2008). IEEE,2008,pp.161–170.
[4] M. Podlesny, Networking Mechanisms for delay-sensitive applications. Washington University in St. Louis, 2009.
[5] T. H. Boraten and A. K. Kodi, "Securing nocs against timing attacks with non-interference based adaptive routing," in 2018 Twelfth IEEE/ACM International Symposium on Networks-on-Chip (NOCS). IEEE, 2018, pp. 1–8.
[6] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxon- omy and detection," IEEE design test of computers, vol. 27, no. 1, pp. 10–25, 2010.
[7] H. Li, Q. Liu, and J. Zhang, "A survey of hardware trojan threat and defense," Integration, vol. 55, pp. 426 – 437, 2016.
[8] 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.
[9] 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, 2014, pp. 1–6.
[10] K. Xiao and M. Tehranipoor, "Bisa: Built-in self-authentication for preventing hardware trojan insertion," in 2013 IEEE international symposium onhardware oriented security and trust(HOST). IEEE,2013,pp.45–50.
[11] N. Aswathy, R. R. Raj, A. Das, J. Jose, and V. Josna, "Adaptive packet throttling technique for congestion management in mesh nocs," in International Symposium on VLSI Design and Test. Springer, 2017, pp. 337–344.
[12] C. Wang, W.-H. Hu, and N. Bagherzadeh, "Congestion-aware network- on-chip router architecture," in 2010 15th CSI International Symposium on Computer Architecture and Digital Systems. IEEE, 2010, pp. 137– 144.
[13] R. Manju, A. Das, J. Jose, and P. Mishra, "Sectar: Secure NoC using trojan aware routing," in International Symposium on Networks-on-Chip. IEEE, 2020, pp. 1–8.
[14] M. Rajan, A. Das, J. Jose, and P. Mishra, "Trojan aware network-on-chip routing," Network-on-Chip Security and Privacy, pp. 277–310, 2021.
[15] A. Das, S. Babu, J. Jose, S. Jose, and M. Palesi, "Critical packet priori- tisation 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.
[16] N. Binkert et al., "The gem5 simulator," ACM SIGARCH Computer Architecture News, vol. 39, no. 2, pp. 1–7, 2011.
[17] 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.