TampML: Tampering Attack Detection and Malicious Nodes Localization in NoC-based MPSoC

Haoyu Wang, *Student Member, IEEE*, and Basel Halak, *Member, IEEE*

***Abstract*—The relentless growth in demand for computing resources has spurred the development of large-scale, high-performance chips with diverse, innovative architectures. The Network-on-Chip (NoC) paradigm has become a predominant system for on-chip communication within Multi-Processor System-on-Chip (MPSoC) designs. However, the increasing complexity and the reliance on outsourced Third-Party Intellectual Properties (3PIPs) introduce non-negligible risks of Hardware Trojan (HT) insertions by untrusted IP vendors. One of the most critical threats posed by HTs is the tampering with communication data packets. In this paper, we introduce a comprehensive framework for the detection of tampering attacks and localization of HTs within NoCs. This framework is incorporated into a novel distributed monitoring architecture that leverages the NoC structure. Utilizing a machine learning model for malicious flit detection and a high-precision algorithm for HT node localization, the framework's efficacy has been substantiated through tests with real PARSEC benchmark workloads. Achieving an impressive detection accuracy and precision of 99.8% and 99.5% respectively, the framework can localize HT nodes with up to 100% precision and recall in most cases. Furthermore, the data cost of localization is on average only 3.7% of tampered flits, which is significantly more efficient—up to 11 times faster—than our initial methods. As a comprehensive and cutting-edge security solution for combating communication data tampering attacks, it accomplishes the expected performance while maintaining minimal power and hardware overhead.**

***Index Terms*—Hardware Security, NoC, MPSoC, Hardware Trojan, Data Tampering, Machine Learning.**

I. Introduction

THE significant rise in the outsourcing level in the hardware supply chain has led to the emergence of new security threats at the circuit level, for instance, HT (Hardware Trojan) [1] [2]. HTs are malicious modifications in the circuit or architecture, especially in the VLSI (Very Large-Scale Integration) systems. The Network-on-Chip (NoC) architecture is widely employed to facilitate efficient communication among Intellectual Properties (IPs) in Multi-Processor Systems-on-Chip (MPSoCs). However, this architecture is susceptible to the injection of Hardware Trojans (HTs), posing significant risks to communication systems and protocols. Given the challenges in modern chip design, it is often difficult to avoid the use of third-party IPs (3PIPs) from untrusted vendors.

This paper was produced by the IEEE Publication Technology Group. They are in Piscataway, NJ.

Manuscript received January 10, 2024; revised May 30, 2024.

Three out of the five security issues in NoC-based MPSoCs summarized in [3] can be induced by HTs, namely spoofing, Denial-of-Service (DoS), and buffer overflow or memory extraction. Such attacks can take place at various stages of the hardware development cycle, including IP development, system integration, physical implementation, and IC (Integrated Circuits) fabrication. Detecting hardware Trojan insertion during development using verification methods is highly challenging and often ineffective [5]. This is because these methods are designed to verify circuit functionality under expected operational conditions, whereas stealthy hardware Trojans are typically activated either internally or under rare conditions, which are unlikely to be covered by standard verification or validation methods. This is especially true in heterogeneous multi-core systems with highly complex functionality that can easily mask malicious behavioral deviations caused by a Trojan. Therefore, the application of artificial intelligence technology in this context is on the rise. For instance, the use of supervised machine learning algorithms for Trojan detection has been proposed, as demonstrated by the authors of [7], showing potential for achieving the outstanding performance including high detection accuracy and low overhead and power.

HTs in NoC architectures can sabotage functionality or facilitate sensitive information leakage. A common tactic is communication data tampering, allowing system cores to be spoofed and data stolen. HTs are hypothesized to be injected into a router, Network Interface (NI), or an outsourced IP core, as discussed in [6], enabling traffic data manipulation. HTs in routers can alter both source and routing-path traffic data, while those in NIs and IPs are limited to tampering with source data. Existing HT detection methods focus on system feature monitoring. The study in [7] utilized an SVM (Support Vector Machine) algorithm for detecting traffic attacks in a custom NoC for biomedical applications. Gradient Boosting classifiers were used in [8], and [9] proposed a novel router architecture with an embedded DetectANN module for HT detection and malicious data mitigation. In [12], timing violations of packets were employed for HT detection and localization in NoC. Yao's study in [13] presented a framework for both detecting and localizing traffic attacks in NoC, treating the source node as malicious. However, this method overlooks nodes on the malicious packet's routing path. To our knowledge, there is no existing work using machine learning or similar advanced techniques for detecting and localizing tampering attacks.

This paper is the first to propose a machine learning-based approach for detecting and localizing HT-based data tampering attacks in NoC, surpassing existing methods. Building on our preliminary work in [14], this paper presents significant enhancements and extended experiments. Our results show that the ANN (Artifical Neural Network) model and our proposed algorithm effectively parses traffic data to detect and localize HT-affected nodes with high overall performance in a novel distributed monitoring system. The primary contributions of this work are detailed subsequently.

- A complete and enhanced communication data tampering attack detection & HT-injected nodes localization framework has been proposed. The detection stage has been optimized to enhance accuracy and reduce runtime and overhead. This innovative framework combines ML with flit attributes, marking the first time such an approach has been used to detect tampered data communication and accurately localize HT nodes.
- A novel, distributed monitoring system is proposed for the real-time collection of traffic data, integrating and operationalizing the security framework. This system has been explored and assessed across three conventional in-router monitoring schemes.
- An improved experimental flow was established to validate the effectiveness of the proposed framework under real workloads. Both static and dynamic validation methods were implemented using Gem5 simulators. The proposed solution was rigorously evaluated using Synthetic Traffic Pattern (STP) and PARSEC benchmarks, which mimic real-world traffics and workloads. It was extensively compared with existing approaches, demonstrating its robustness and superiority.

The structure of this paper is organized as follows: Section II provides a review of the related work. The proposed framework is detailed in Section IV. Section V introduces the monitoring scheme and architecture. Section VI discusses the experimental setups and results. Finally, conclusions are drawn in Section VII.

II. Background and Related work

In [7], a supervised Support Vector Machine (SVM) algorithm was trained to identify three common traffic attacks: traffic diversion, router looping, and core address spoofing. This training utilized packet content features for offline pre-training. Hussain, in [10] introduced the Energy Efficient Trojan Detection (EETD) design, emphasizing high energy efficiency in HT detection within NoC systems. Charles, in [12], developed an HT detection method that continuously monitors the timing of packet arrivals and other factors in NoC. This method introduced two boundary parameters, Packet Arrival Curves (PACs) and Destination Packet Latency Curves (DLCs), as thresholds to differentiate between normal and malicious packets. Charles also proposed a real-time localization technique for malicious IPs by assessing whether their DLCs fall within a specific timing confidence interval, leading to a protocol for identifying malicious IPs. Ke, in [9], designed a DetectANN module within a proposed router architecture to serve as an HT detection unit. By analyzing network activities such as buffer and link utilization, and local operational temperature, Ke's approach achieved a notable detection accuracy of 97%. Yao et al. in [13] applied machine learning to localize HT, directly classifying the source node of malicious packets as HT-injected, without considering elements along the routing path of these packets. In a subsequent study, Charles et al. utilized machine learning (specifically, XGBoost) in [8] to detect flooding Denial of Service (DoS) attacks in real-time, achieving an impressive accuracy of approximately 99.9%. The Sniffer system, detailed by Sinha in [11], employed a perceptron-based machine learning model to evaluate the traffic status of routers. This system could detect and localize Malicious 3-Party IPs (M3PIPs) initiating flooding attacks.

Machine learning algorithms commonly used for detecting or localizing Hardware Trojans (HTs) in Network on Chip (NoC) systems include SVM [7], ANN [9], Random Forest [13], XGBoost [8], and perceptron-based ML [11]. The study in [9] highlighted the effectiveness of deploying an ANN model for HT detection in NoC, leading to its adoption in our semi-supervised learning approach for processing packet data and providing foundational predictive values for further analysis and decision-making within our framework. Moreover, studies like [11], [8], and [9] have utilized machine learning to detect attacks, leveraging runtime-engineered system performance features such as timing, latency, utilization, temperature, packet counts, and injection/ejection rates. These features are easily collectible and can indirectly indicate normal or abnormal system states. However, they are subject to external influences, such as temperature variations due to environmental conditions, or time and latency alterations due to changes in system frequency, voltage, and other factors, which can potentially lead to inaccurate HT detection. Conversely, Amey's research [7] effectively used packet content to train SVMs, demonstrating significant success in detecting Denial-of-Service (DoS) attacks, including traffic diversion and core spoofing. Packets' attributes are particularly sensitive to specific attacks, especially malicious tampering, and are less likely to be affected by external interferences not related to HT attacks.

In the realm of detecting and localizing traffic attacks such as DoS and related HTs in NoC systems, ML has been employed predominantly in scenarios like flooding attack detection. However, one specific type of DoS attack, known as the tampering attack, has not been extensively addressed using ML techniques, but rather through traditional methods. For instance, Frey [15] introduced a countermeasure against HT tampering attacks using Physical Unclonable Function (PUF). This method involved the deployment of a random vector generator in routers, enabling each router to have an independent local random-vector generator. This generator was crucial in providing evidence for selecting the flit permutation pattern, leading to a collaborative dynamic permutation and anti-tampering flit protection based on the PUF technique. Furthermore, to counteract complex bit-triggered HTs capable of compromising packets, Kumar in [16] utilized bit shuffling and Hamming coding methods. These methods were effective in mitigating specific types of HTs that tamper with

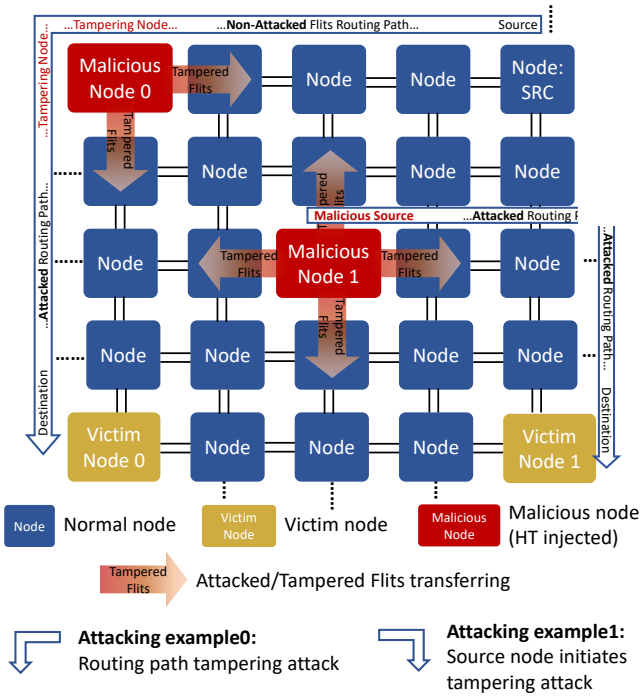


Fig. 1: Illustration of threats modeling.

packet content. Additionally, Charles in [6] proposed a novel lightweight and trust-aware routing algorithm. This algorithm was designed to enhance system protection by circumventing malicious outsourced IPs during communication, thereby providing a defensive mechanism against tampering attacks in NoC environments.

None of the related state-of-the-art studies have introduced a method for detecting and localizing packet tampering HT. Our earlier work, as presented in [14], was the first to propose a comprehensive technique specifically addressing the detection and localization of packet tampering attacks. In this paper, we aim to extend and refine this technique, focusing on enhanced detection and more precise localization of packet tampering attacks.

III. Threat Modelling

This work specifically addresses HT-based communication data tampering attacks in NoC systems, focusing on the manipulation of flits or packets. Consequently, malware and external attacks fall outside the scope of this research.

Starting with modelling attacking paths. Two distinct attacking patterns are considered in this work, as depicted in two examples in Fig. 1. The victim nodes are located in the bottom left and bottom right corners, targeted by malicious nodes 0 and 1, respectively. Example 0 involves malicious node 0 and victim node 0, illustrating flit tampered by a routing-path node. In this scenario, flits originated from the top right corner pass through and are tampered by malicious node 0 before reaching victim node 0. Example 1 showcases an attack from a malicious source node 1, where flits are sent directly from this node to victim node 1, affecting all intermediate nodes on the malicious routing path. The localization technique in the complete framework is specifically designed to address these

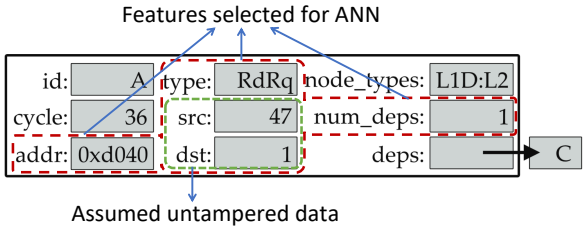


Fig. 2: Packet example to illustrate features selected and assumption in [14].

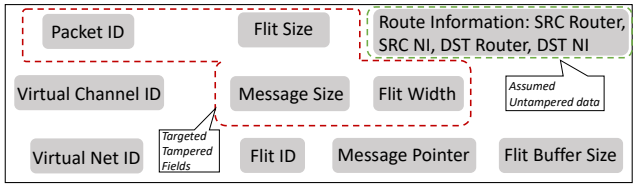


Fig. 3: Flit format to illustrate features selected and assumption.

attack patterns. Its aim is to swiftly and accurately identify the malicious node involved in the attack.

[14] primarily explored the modification of packets' feature data, such as memory address, packet type, and number of dependencies, as depicted in Fig. 2. These modifications can be considered alterations of packets' global features. However, each packet is composed of flits, including the head flit, body flit, and end flit. In practical hardware implementations and runtime communication, the process of constructing flits is more susceptible to tampering by inserted HTs. Since there are more data bit operations and calculations during flit construction rather than during transfer, such as ID (packet ID and virtual channel ID) calculations, message formatting (message size and message pointer), and buffer size determination, HT could be triggered during these operations to maliciously interfere with the information being injected into the flit. Therefore, this work targets flits as the objects of attack instead of packets, providing a more realistic scenario. We have identified specific fields within a flit frame for tampering, including packet ID, flit size, message size, and flit width, as shown in Fig. 3. The rationale for selecting these features is twofold: they do not significantly disrupt normal traffic and communication in the NoC, yet they pose a high risk of spoofing or fuzzing victim IPs, which is a major threat posed by data tampering attacks. Additionally, by selecting a diverse range of data types as targets, the research covers a broader spectrum of real attack scenarios, thereby enhancing the validation of the proposed framework. Consistent with our initial work, we assume that the source ID and destination ID, integral to routing information, are not maliciously modified. This assumption is crucial to avoiding introducing new traffic attacks such as traffic diversion, routing loops [7], and flooding [8] attacks, which could lead to unwanted side-effects from non-targeted attacks. As such, all flits are expected to be routed normally by the selected routing algorithm, such as the XY-routing algorithm.

In addition, the HT is specifically injected into the input

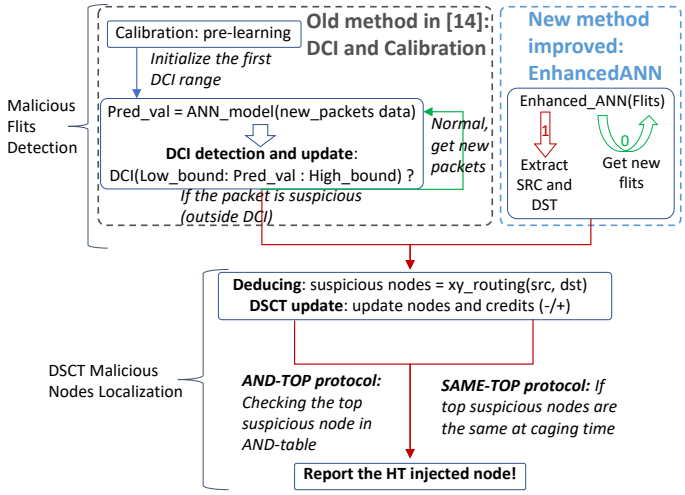


Fig. 4: The working flow of the improved proposed Hardware Trojan detection and localization framework.

unit of the router. This location is chosen due to its critical role in numerous functions including flit formatting, sending, diverting, routing calculation, and receiving. Nodes infected by the HT can maliciously modify the information of normal flits passing through these compromised nodes.

IV. Proposed TampML framework

A. Overview of proposed framework

The enhanced Hardware Trojan (HT) detection and localization framework, dubbed TampML, derives its name from a fusion of "Tampering Attack Detection" and "Malicious Nodes Localization," while also symbolizing the use of machine learning for tampering attack detection. It comprises fewer phases compared to our first design: the **Malicious Flits Detection Phase** and the **Dynamic Security Credit Table (DSCT) HT Localization Phase**, as shown in Fig. 4. This framework's detection phase is tasked with identifying malicious flits. An effective ANN model, incorporated into the initial detection phase, processes selected real-time flit features (see Fig. 3). Identified suspicious flits are then communicated to the DSCT. The DSCT initially identifies suspect nodes and concurrently updates security credits across two DSCT tables, facilitating the dynamic localization of the HT node. Two concurrent localization protocols aim to pinpoint HT-compromised nodes. The first, SAME-TOP, introduced in our prior work [14], is complemented by a new protocol in this paper, AND-TOP. AND-TOP serves as an adjunct method, improving the localization's effectiveness and efficiency. Further details are elucidated in subsequent sections.

B. Detection: Malicious Flits Detection using EnhancedANN

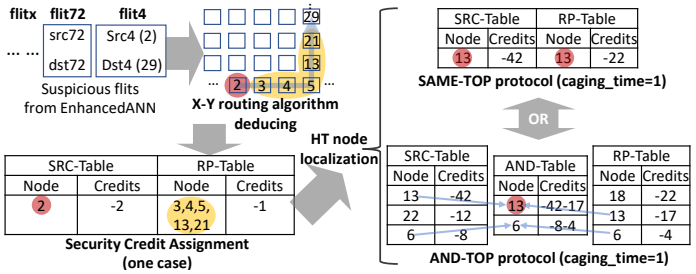
A variety of machine learning models have been utilized to address security issues in NoCs, among which neural networks stand out as more powerful and efficient compared to other fundamental models. This superiority is attributed to their robust performance with non-linear datasets and proficiency in binary classification tasks. Consequently, in this stage, an ANN-based ML model is utilized, focusing on flits' contents to identify tampering attacks. The threat model centers on flits' contents being tampered with in real-time by HTs in the input units of malicious routers. Key features for detecting abnormal flits, highlighted in red in Fig. 3, were chosen for the framework. Due to flits containing more communication and routing information compared to packets, feature selection requires careful consideration. The selected features for detecting tampering include packet ID, flit size, message size, and flit width. Other elements, such as virtual channel ID, virtual net ID, message pointer, flit buffer size, and routing information, primarily influence communication but can lead to unintended consequences in detecting targeted attacks. For example, altering the virtual channel ID can cause flit congestion by directing traffic to busy channels, and tampering with routing details such as the destination ID can misroute flits to incorrect IPs, resulting in data loss for the intended receiver. Unlike methods that monitor abnormal system performance variations, our framework preemptively reacts by directly extracting and monitoring flits' contents, closely aligning with the attack mechanism. Furthermore, flit tampering attacks are not related to the chip's physical characteristics including area temperature or electrical noise, but are strongly associated with the data itself.

Algorithm 1 Malicious Flit Detection using EnhancedANN

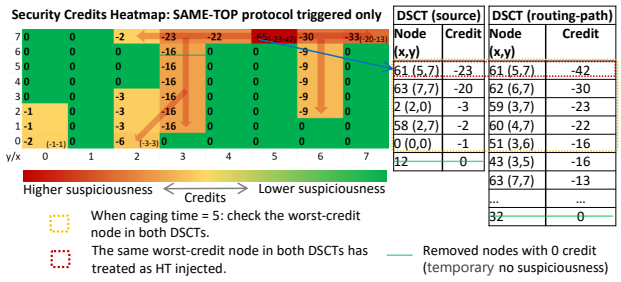
Require: Input Attributes of Flits: Pkt ID, Flit Size, Msg Size, Flit Width

1: **while** A flit inputs **do**
2: $Classification \leftarrow EnhancedANN(Flit\_Attributes)$
3: **if** $Classification == 1$ **then**
{//Suspicious Flit, Report SRC and DST to DSCT.}
4: **else**
{//Normal Flit, Getting Next}
5: **end if**
6: **end while**=0

An ANN model was trained using supervised learning, with real workload flits labeled as normal (0) or tampered (1). This training enables the model to quickly discern whether a flit is normal or abnormal. The model's architecture consists of three layers: an input layer, a hidden layer, and an output layer, balancing between model size and performance (including accuracy, precision, recall, and inference speed). The model was iteratively developed, starting from the smallest size and expanding until satisfactory performance was achieved, resulting in the final three-layer structure. The classification outcomes of this ANN model can be directly used without further processing, such as the Dynamic Confidence Interval (DCI) method mentioned in [14]. The latter approach suffers from significant disadvantages, including prolonged processing time for a single packet, increased computational demands, and heightened randomness. A classification result of 1 indicates a suspicious flit, prompting the reporting of its source and destination ID to the second stage of DSCT HT node localization. Conversely, a result of 0 signifies a normal flit, and the model then examines the next flit. This process, depicted in the right corner of Fig. 4, is termed EnhancedANN.



(a) DSCT workflow: from suspicious flits to malicious node localization



(b) DSCT example: AND-TOP protocol only, and the heat map of the HT-injected node simulated and localized

Fig. 5: Dynamic Security Credit Table illustration and the AND-TOP example

It streamlines the malicious data detection stage and enhances detection efficiency and performance.

The entire process of malicious flit detection is more efficient and streamlined compared to the Dynamic Confidence Interval (DCI) method, as illustrated in Algorithm 1. Specifically, it has removed the calibration stage, which costs around 500 packets (equivalent to more than 500 flits), DCI checking inside or outside of the dynamic interval, and DCI updating the curve. Only the ANN classification stage is retained, with an added if-else statement to check the classification result. It is estimated that at least 80% of computation and time is optimized by this newly proposed detection approach.

C. Localization: Dynamic Security Credit Table

The flit tampering behavior, including maliciously modified data, is not directly related to the location of HT injection in the NoC. Thus the machine learning model is nearly impractical on HT node localization. To address this, we developed the Dynamic Security Credit Table (DSCT) algorithm, which is both precise and rapid. DSCT's primary function is to identify suspicious nodes, assign and update their security credits, rank nodes based on these credits, and ultimately localize the HT. The DSCT algorithm hinges on two crucial elements: the node and its associated credit. Additionally, to enhance the original scheme mentioned in [14], we incorporated an extra localization protocol, functioning as a compensatory trigger. The localization triggering protocols are named SAME-TOP and AND-TOP. The 'TOP' in their names denotes the highest level of suspicion, represented by the worst credit value in the DSCT table.

Before delving into the new two localization protocols, it is important to outline some preliminary algorithms of the DSCT. The DSCT approach utilizes two separate tables for tracking suspicious activity: one for source nodes and another for routing-path nodes involved in suspicious flits transmission, as shown in the threat modeling in Fig. 1.

In Algorithm 2, suspicious source nodes are identified by the originating node of suspicious flits and are added to the *DSCT_src* table with an initial -2 security credit in *CreSrc*. Each time a source node already listed in *DSCT_src* generates a new suspicious packet, its credit score decreases by 2. Conversely, the *DSCT_rtp* table catalogs suspicious routing-path nodes, excluding the flit's source node. The updating of *DSCT_rtp* involves utilizing the X-Y routing algorithm to infer routing-path nodes based on source and destination IDs, subsequently updating their credits in *CreRtp* with a value of -1. Nodes in both tables receive a credit reward of +1 if they are not involved in any suspicious activity during a detection cycle. A node is removed from either DSCT table (green strikethroughs in Fig. 5b) when its credit reaches 0 or a positive value. The distinct credit assignment mechanisms in the two DSCTs ensure that malicious nodes are identified in every inspection of suspicious packets. Flits tampered with by source nodes have a more significant impact compared to those

Algorithm 2 Dynamic Security Credit Table Algorithm

Require: Source ID and Destination ID of suspicious flits reported by EnhancedANN

while Suspicious flit inputs **do**
  {//DSCT_src check and update:}
  if $src\_node \notin DSCT\_src$ **then**
    $DSCT\_src \leftarrow DSCT\_src \cup src\_node$
  end if

  $CreSrc[src\_node] \leftarrow CreSrc[src\_node] - 2$
  $CreSrc[other\_nodes] \leftarrow CreSrc[other\_nodes] + 1$
  {//DSCT_rtp check and update:}
  $rtp\_nodes \leftarrow XY\_routing(src\_node, dst\_node)$

  if $rtp\_nodes \notin DSCT\_rtp$ **then**
    $DSCT\_rtp \leftarrow DSCT\_rtp \cup rtp\_nodes$
  end if

  $CreRtp[rtp\_nodes] \leftarrow CreRtp[rtp\_nodes] - 1$
  $CreRtp[other\_nodes] \leftarrow CreRtp[other\_nodes] + 1$
  {//AND-TOP table: Picking same nodes in both tables}
  $DSCT\_AND \leftarrow DSCT\_src \cap DSCT\_rtp$
  $CreAnd \leftarrow CreSrc \cap CreRtp$

  $Caging\_SAME \leftarrow (DSCT\_src[worst\_credit] == DSCT\_rtp[worst\_credit])$
  $Caging\_AND \leftarrow (len(DSCT\_AND) >= 3)$
  if $(Caging\_SAME$ **or** $Caging\_AND)$ at caging_time **then**
    {//Report an HT-injected node!}
  end if
end while=0

altered by routing-path nodes; the former affects all nodes between the source and destination, while the latter influences only nodes past the tampering HT-injected node.

In the example depicted in Fig. 5a, a suspicious flit, specifically flit 4, is being processed by the DSCT localization module. The source ID (2) and destination ID (29) of this flit are first extracted and analyzed using an inverse XY-routing algorithm to deduce the routing-path nodes. In this scenario, nodes 3, 4, 5, 13, and 21 are identified as suspicious routing-path nodes. Subsequently, both the source node and these routing-path nodes are added to their respective tables: the SRC-Table for the source node and the RP-Table for the routing-path nodes. Each type of node is assigned different security credits upon entry, with the source node receiving -2 credits and the routing-path nodes each receiving -1 credit.

1) SAME-TOP Trigger Protocol

The localization of HT nodes takes place at the conclusion of the DSCT algorithm. We begin with an explanation of the SAME-TOP protocol. A concept called 'caging time', plays a crucial role here. Caging time, though not an actual time measurement but an integer, represents the total count of suspicious nodes in each DSCT simultaneously. It determines the optimal moment for localizing the HT-injected node, thus significantly reducing the chances of erroneous HT localization. However, this work has optimized it further, owing to the new dataset derived from flits data, which slightly differs from packet data. As a result, the SAME-TOP protocol can achieve excellent localization performance.

In the SAME-TOP localization protocol, localization is triggered under one condition: if the same suspicious node ID appears with the worst credits (highest suspiciousness) in both DSCT tables. When this condition is met, the top-suspicious node is reported as an HT-injected node to a security unit or users. Essentially, the SAME-TOP protocol identifies an HT-injected node as the one that concurrently tops in suspiciousness in both DSCT tables.

Fig. 5b illustrates this with an HT-injected node 61 as an example. The two tables in this figure correspond to *DSCT_src* and *DSCT_rtp* as defined in Algorithm 2. The heatmap in Fig. 5b shows the total credits calculated by summing the same nodes' credits in the two DSCTs. For instance, node 61 at coordinates (5,7) has a total credit of -65, derived from -23 in DSCT(source) and -24 in DSCT(routing-path). The orange square indicates it is time to activate the localization function due to the presence of a sufficient number of suspicious nodes, and the red square includes the same nodes with the lowest credits in both tables, which are then identified as HT-injected nodes. The heatmap also visually represents the threat propagation from the corrupt node 61 to its neighbors and packet receivers, indicated by red arrows originating from node 61. Similarly, in Fig. 5a, if the SAME-TOP protocol is triggered at the final stage of DSCT, node 13 is identified as a malicious node due to its lowest credits of -42 and -22 in the separate source-node and routing-path-node tables.

2) AND-TOP Trigger Protocol

The AND-TOP protocol, designed as a supplementary localization method, is more straightforward. It was specifically developed to address situations where the SAME-TOP protocol may not be effective or responded promptly, particularly in cases involving HT-injected nodes that are highly inconspicuous. AND-TOP operates by leveraging the two DSCT tables to identify intersectional nodes and aggregate their credits from both tables, thereby creating a new DSCT table named the AND-Table. This table is dynamically updated in real-time based on the latest entries in both the source-node DSCT table and the routing-path DSCT table.

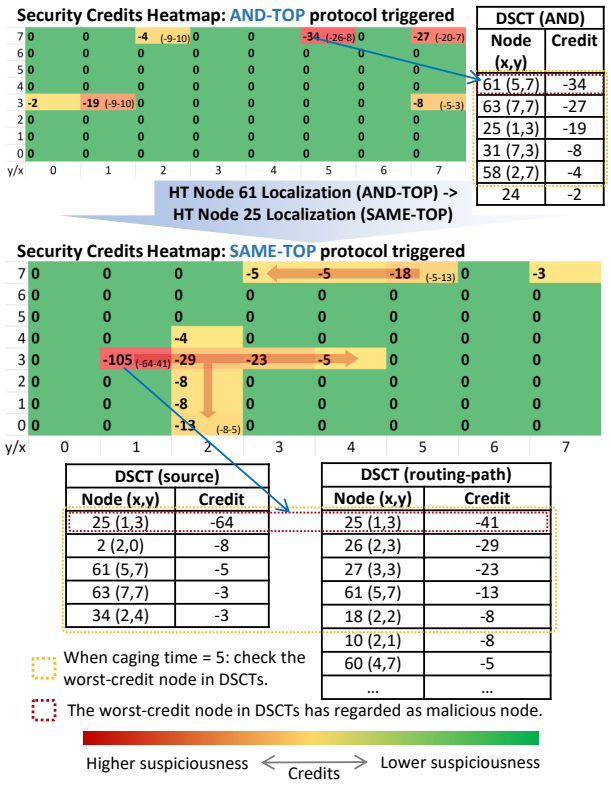


Fig. 6: DSCT examples: two malicious nodes (nodes 61 & 25) and both protocols triggered.

A configurable caging time, which must not be set to zero, dictates the timing for localizing malicious nodes within the AND-TOP protocol. If set to zero, the top node would be incorrectly flagged as an HT-injected node every time a new suspicious node is added. As shown in Fig. 5a, the AND-Table is updated by selecting the same nodes that appear in both the SRC-Table and the RP-Table, and their corresponding credits are summed and recorded in the AND-Table. Once the caging time for the AND-TOP protocol is reached, the node with the highest level of suspiciousness (in this example, node 13 with a combined credit score of -42-17) is identified as an HT-injected node. As a compensatory measure, the AND-TOP protocol is integrated into the DSCT localization algorithm alongside the SAME-TOP protocol, using an OR operation. This means either protocol can be triggered, but not both simultaneously. However, in practice, the SAME-TOP protocol is given higher priority due to the potential risk of the AND-TOP protocol incorrectly identifying innocent nodes. The process of extracting intersectional nodes and their credits from the two tables is detailed in lines 14 and 15 of Algorithm 2.

In the example illustrated in Fig. 6, both the AND-TOP and

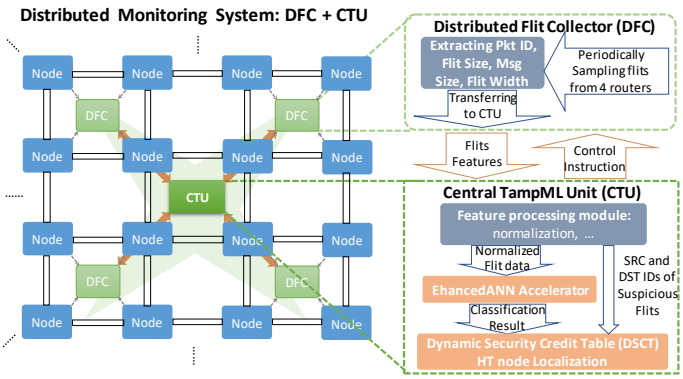


Fig. 7: Distributed Monitoring System Design. (An example of 4x4 NoC)

SAME-TOP protocols were activated, successfully localizing two simulated HT-injected nodes. The AND-TOP protocol initially localized malicious node 61, concurrently marking node 24 with a low credit value, indicative of suspicious activity around it. Subsequently, node 25 was localized by the SAME-TOP protocol, leading to updates across all DSCT tables. For instance, node 25 escalated to the top position in both the source and routing-path tables due to its worst credit scores, while node 61's credits improved as fewer packets passed through and were compromised by it.

An interesting observation from the credit heatmaps in Fig. 6 is that the paths of attack or impact, characterized by tampered flits, are more prominently visible in the heatmap associated with the SAME-TOP protocol than with the AND-TOP protocol. This phenomenon is evident in the attack paths originating from nodes 25 and 61 during the localization process of node 25. A similar pattern is noticeable in Fig. 5b, where multiple attack paths from node 61 are clearly discernible. The reason for this discrepancy lies in the AND-TOP protocol's focus on nodes common to both tables, potentially overlooking other suspicious nodes. Consequently, the AND-TOP heatmap may show inconsistent nodes without clear attack paths, whereas the SAME-TOP heatmap consistently highlights suspicious nodes with more defined paths of attack. This distinction underscores the complementary nature of the two protocols, with each offering unique insights into the network's security status.

The enhanced HT localization framework we have developed, building upon the previously proposed model, enables swift and precise localization of HT-infected nodes. The ability to rapidly and accurately identify HT nodes is crucial, as it affords the system more valuable time to defend against HT attacks. This enhanced response time ensures that subsequent workloads can be executed reliably and smoothly. Therefore, achieving high precision in HT node localization is not just beneficial but essential for the overall security of the system.

V. Distributed Monitoring System Architecture Design in Network-on-Chip

To facilitate real-time flit collection and traffic monitoring, we have proposed a distributed monitoring system integrated within the NoC architecture. This system, depicted in Fig. 7, comprises two main functional modules: the **Distributed Flit Collector (DFC)** and the **Central TampML Unit (CTU)**. The scalability of the DFC allows for an adjustment between 4 to 16 nodes under monitoring, depending on the size of the NoC being monitored. In our experimental setup involving a 64-core NoC system, we utilized 4 DFCs, with each DFC monitoring a cluster of 16 nodes. Typically, a single CTU suffices for the entire system.

A. Distributed Flit Collector (DFC)

In the top-right corner of Fig. 7, the functionalities of the DFC are showcased. The DFC's operation is tailored to the security sensitivity requirements of the system, collecting flits at an adjustable frequency, referred to as the sampling rate. Once the flit data frames are obtained, an extractor component within the DFC selects the relevant data needed for the ML model classification. The final responsibility of the DFC is to transmit all pertinent flit features to the CTU module for further processing.

B. Central TampML Unit (CTU)

TampML is our proposed tampering attack detection and localization approach, integrated into the CTU module, which is represented by orange blocks in Fig. 7. The CTU is tasked with critical functions including data pre-processing, EnhancedANN classification, and malicious node localization. The DFCs periodically forward the extracted flit data to the CTU, which then undertakes the sorting, selecting, and normalizing of this data. Once the preliminary processing is completed, the EnhancedANN model takes over, classifying the flit data. This classification process is facilitated by a machine learning accelerator embedded in the actual chip, which significantly boosts detection efficiency while maintaining low hardware and power costs. As outlined in chapter IV, the DSCT localization module receives routing information of suspicious flits identified by EnhancedANN. It then executes the localization algorithm and protocols at a logical level, achieving a balance in power, performance, and area (PPA) considerations. The localization outcomes are subsequently sent to the system's main processing unit (CPU) for further security-related decision-making.

C. Design Considerations

This integrated approach, combining TampML with the CTU and DFCs, ensures a streamlined and effective response to potential security threats within the system. However, it is important to note that typically, security designs can impose redundancy on the power, performance, and area (PPA) aspects of the actual SoC. To mitigate this issue and minimize any undesired impact of our proposed monitoring system, we have introduced two key design considerations.

The first consideration is an adjustable flit sampling rate. This feature allows the system to maintain a balance between obtaining sufficient flit data for security analysis and minimizing power consumption. By carefully controlling the sampling rate, the CTU can receive the necessary data without significantly impacting the overall system efficiency.

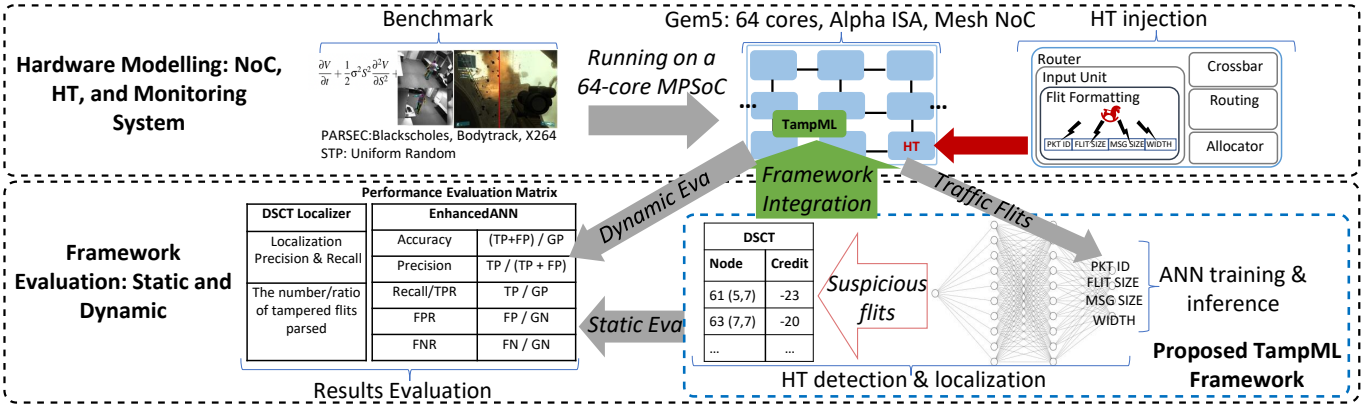


Fig. 8: Experiment flow.

The second consideration involves an architectural level strategy, where we isolate the security modules from the NoC system itself. As depicted in the architecture overview of Fig. 7, the DFCs and the CTU are designed to 'collect', 'extract', and 'select' flit data but do not engage in NoC communication. This means that the monitoring system does not introduce additional traffic, delay the normal transmission of any flit, or alter any flit data. This approach of isolating the security functions from the main NoC operations significantly reduces the risk of interference with the system's primary functions. Moreover, the monitoring system is safeguarded against attacks from the tampering HT within NoCs. This enhanced security is due to the system's limited role in merely reading and parsing flit data, without actually utilizing this data for any operational purposes. Similarly, the HT is unable to probe or interact with our monitoring system, thus preventing them from executing targeted attacks against it.

These design considerations, particularly the second, position our architecture as more advanced compared to existing state-of-the-art secure NoC architectures, such as those referenced in [1], [5], [9], [11]. This advancement is primarily due to our system's ability to provide robust security monitoring without impeding the NoC's core operational efficiency.

VI. Experiment and Results

In the experiments conducted as part of the study in [14], a purely static simulation and validation approach was employed. The tool Netrace [17] was used for capturing and parsing packet data. Additionally, a script-based (Python) HT-injection module was designed to statically tamper with the packet dataset. A key aspect of these experiments was the exclusive use of a single PARSEC workload as the benchmark for evaluating the framework. In contrast, this work expands upon the experimental methodology. We have established a more comprehensive experiment flow to assess our framework, which is divided into two main segments as depicted in Fig. 8: **Hardware Modelling** and **Framework Evaluation**. This bifurcated approach allows for a more thorough and nuanced evaluation of the framework, encompassing both the simulation of hardware conditions and the direct assessment of the framework's performance and effectiveness in various scenarios.

1) Hardware Modelling

A 64-core NoC-based MPSoC was constructed using the Garnet tool in the Gem5 simulator. This setup included our distributed monitoring system within the modeled NoC. For application deployment, we utilized three real workloads from the PARSEC benchmark collection to generate authentic flit datasets. Additionally, the typical Synthetic Traffic Patterns (STP) were simulated to assess the detection capabilities of our framework. The uniform random case within the STP was utilized to compile the training dataset for EnhancedANN, whereas other benchmarks served as verification datasets for it. On the hardware side, we implemented a tampering HT into the input units of selected routers. This HT could randomly alter specific fields in flits that were either routed through or originated from these routers. For this study, HTs were injected into four strategically chosen routers within the NoC, identified by IDs 11, 32, 44, and 59, to ensure a representative spread across the network. It should be noted that in the STP benchmark, the HT is triggered continuously, whereas in the PARSEC benchmark, it is activated only during the Region-of-Interest (ROI). This approach is designed to mirror the most realistic scenarios and to demonstrate the most serious effects of HT activation.

2) Framework Evaluation

We retained the static validation flow to expedite the collection of metric data. For realistic MPSoC scenarios, our proposed security architecture, depicted in Fig. 7, was integrated into the modeled NoC, enabling runtime dynamic evaluation. We calculated key metrics such as accuracy, precision, recall, False Positive Rate (FPR), and False Negative Rate (FNR) to gauge detection performance. Due to the unique aspects of our localization issue and method, precision and recall are especially relevant in reflecting true performance. Equally important is the ratio of flits used for localization in each scenario, as this directly indicates the speed of localization.

A. Monitoring System Exploration and Evaluation

The final design of our monitoring system was chosen from three proposed schemes: Distributed Flit Collector (DFC), Scanning Monitors in Routers, and Fixed Monitors in Routers. While the DFC is detailed in Fig. 7, the latter two schemes use internal router monitors to collect flits that are either routed

Evaluation Metrics	Synthetic Traffic Pattern (STP)					PARSEC			Average
	Uniform Random	Neighbor	Shuffle	Tornado	Transpose	Blackschole	X264	Bodytrack	
Accuracy	1	1	1	1	1	1	0.9984	1	**0.9984**
Precision	0.99	1	0.99	0.99	0.99	1	1	1	**0.9950**
Recall	0.99	1	1	1	1	1	0.9934	1	**0.9979**
FPR	0	0	0	0	0	0	0	0	**0**
FNR	0.01	0	0	0	0	0	0.66	0	**0.0837**

TABLE I: Malicious Flits detection results on STP and PARSEC.

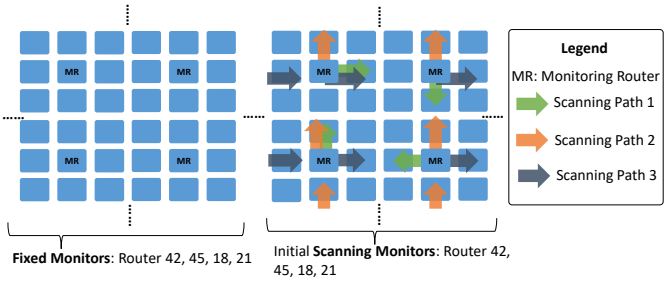


Fig. 9: Fixed and Scanning monitoring schemes.

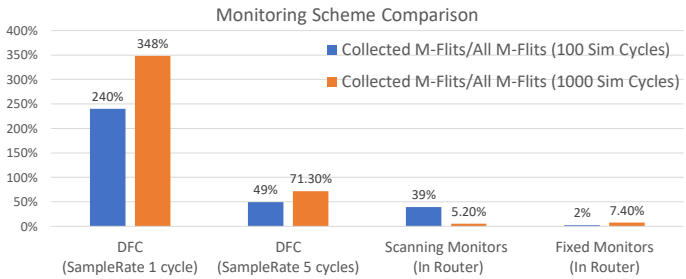


Fig. 10: Monitoring scheme comparison. (M-Flits: Malicious Flits)

through or originate from the router, thus limiting the monitoring range. In selecting routers for monitoring, our goal was to cover as extensive a routing path as possible. Consequently, routers 42, 45, 18, and 21 were equipped with monitors in both the fixed and initial scanning monitoring schemes. The Fixed Monitors scheme is relatively straightforward, involving constant monitoring by the selected routers. The Scanning Monitoring scheme, on the other hand, involves periodically enabling monitors in different routers in a specific sequence, as depicted in the 'scanning path' of Fig. 9. This method allows for a more diverse coverage, encompassing a wider range of routing flits.

We conducted the uniform random traffic pattern of STP benchmark on the NoC system with HT injection to gather tampered flits, calculating the ratio of collected tampered flits by all monitoring routers to all tampered flits in the system to assess the efficiency of our monitoring scheme. Fig. 10 demonstrates the performance of all schemes, showing that our DFC scheme performs well even with a 5-cycle sampling period. With a 1-cycle sampling period, the DFC scheme captures more flits, including numerous repeats, as a flit's routing path may pass through multiple clusters monitored by several DFCs. Although the 1-cycle sampling provides the most comprehensive data for the TampML framework, the 5-cycle sampling represents a more balanced approach, optimizing the trade-off between a number of tampered flits collected and runtime power consumption. Additionally, the scanning monitoring scheme outperforms the fixed monitoring, aligning with expectations. Clearly, the DFC scheme is the preferred choice in this work, ensuring sufficient input data for the TampML framework.

B. Malicious Flits Detection Results

The performance of malicious flit detection is indicated by the EnhancedANN classification results, detailed in Table I. This assessment includes all benchmarks, such as STP and PARSEC, since the detection stage primarily concentrates on the contents of flits rather than aspects including traffics and routing paths. Most of the metrics align with expectations. However, an anomaly was noted in the X264 workload, which exhibited a False Negative Rate (FNR) of 0.66. The X264 is a pipelined media processing workload with high data dependency, which is different from Blackschole and Bodytrack. It also features both high data sharing and high data exchange. Consequently, this anomaly can be attributed to the unique communication traffic characteristics and data usage of the X264 workload, which led to frequent misclassification of malicious flits as normal. Despite this, the average metrics across all benchmarks reached a satisfactory level.

C. HT-injected Node Localization Results

Given the differences in the experimental flow and dataset from the [14], a direct comparison of results is not feasible. We conducted retests of the localization process using the new flits dataset, but only with the early established SAME-TOP protocol. This retesting is depicted in Fig. 11. While the precision of the SAME-TOP protocol remained high, achieving 0.93, the proportion of flits required for a single localization was not consistently low, averaging around 18.7%, and at times reaching as high as 24.5% (nearly one-forth of all tampered flits). This high flit cost for localization is not acceptable in practical scenarios. Therefore, to address this issue and enhance the localization process without compromising precision, the AND-TOP protocol was introduced. This additional protocol aims to improve the efficiency of the localization process, reducing the number of flits needed for accurate localization. The next two subsections are presenting the overall results on the complete framework. it is important to note that our assessment focused exclusively on real PARSEC workloads, as most benchmarks involving STPs feature artificially predefined traffics, rather than naturally occurring,

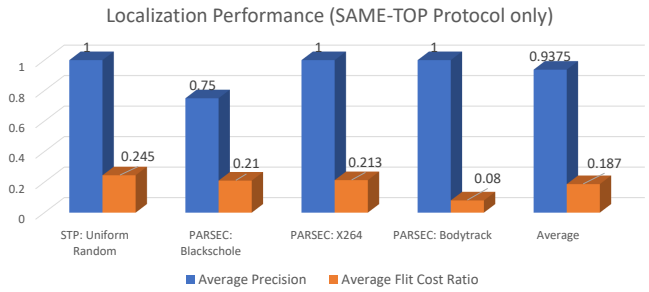


Fig. 11: Localization results with SAME-TOP protocol only.

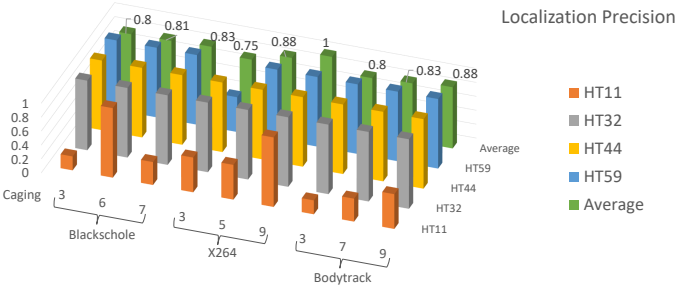


Fig. 12: HT node localization precision results.

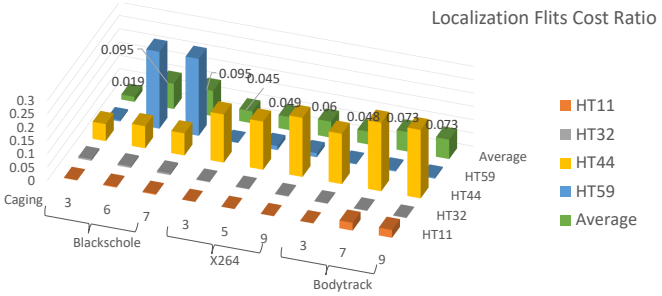


Fig. 13: Flits cost ratio in localization.

application-specific traffic. This distinction is crucial because the artificial nature of STP traffic does not provide a realistic or authentic environment for evaluating our DSCT localization method. Therefore, to ensure a valid and realistic assessment of the DSCT's performance, we relied on real-world workload scenarios as presented in the PARSEC benchmarks.

1) Localization Results

Fig. 12 presents the precision results of our newly implemented DSCT. The 'Z' axis represents the precision of localization, the 'X' axis shows different predefined caging values, and the 'Y' axis displays different node IDs injected by HTs in our experiment. The concept of 'caging number', as introduced in Section IV-C2, refers to the timing for initiating the localization process. A smaller caging number implies a faster localization of the HT-injected node due to increased sensitivity. However, this also leads to a higher likelihood of false positives, which, while still within acceptable limits, reduces the overall precision. It is noteworthy that among the HT nodes, only node 11 tends to trigger more false positives. This exception is attributed to node 11's proximity to node 0, which plays a significant role in controlling the CPU and thus experiences a higher volume of instruction and data transfers. Consequently, when an HT is injected into a node near the main CPU (node 11 in our case), there is an increased chance of false positives. Overall, the average precision is approximately 84% when both the SAME-TOP and AND-TOP protocols are enabled, and all HT nodes are successfully localized.

Another observation is the inverse relationship between the caging number and precision: a larger caging number improves precision but also increases the flit cost, indicating slower localization. This scenario highlights a trade-off between precision and flit cost. The subsequent subsection will delve more into the implications of this trade-off, particularly focusing on the results related to flit cost. In Fig 12, we selected three typical caging numbers for each benchmark, which demonstrate a great balance between precision and flit cost.

2) Localization Efficiency

Fig. 13 illustrates the flit cost per localization event in our proposed architecture (see Fig. 7). When compared to the flit cost ratio under the SAME-TOP protocol alone (as shown in Fig. 11), notable reductions in flit usage are observed across different benchmarks. Specifically:

- For the Blackschole workload, there is a reduction in flit usage by approximately **2.2 to 11 times**.
- In the case of X264, the reduction is between **3.6 to 4.6 times**.
- For Bodytrack, the reduction ranges from **1.1 to 1.65 times**.

These reductions are influenced by the selected caging number, which varies based on the specific traffic characteristics of different workloads. It was observed that a caging number in the range of 5 to 9 tends to be optimal, balancing precision (up to 100%) with a flit cost of no more than 10%.

When comparing localization efficiency among different HT injection nodes, HTs in nodes 32 and 11 are typically localized more rapidly, possibly due to their critical roles in global communication traffic. Conversely, HTs in nodes 44 and 59 generally take longer to localize. This variation may be influenced by our flit collection strategy as well, where the Central TampML Unit (CTU) often receives flits primarily from the first clusters (1 and 2), leading to earlier localization of HTs in nodes with IDs smaller than 32. This issue could potentially be addressed by another architectural approach, which will be discussed in the next subsection.

D. Distributed localization scheme exploration

In addition to our initial monitoring scheme shown in Fig. 7, we explored a more distributed approach that integrates four complete detection and localization TampML frameworks into the four Distributed Flit Collectors (DFCs). In this architecture, each DFC is tasked with the detection of malicious flits and localization of HT nodes within its respective node cluster. While this setup incurs greater hardware and power overhead, it significantly enhances the efficiency of detection and localization, as evidenced by the results in Fig. 14. Compared to Fig. 13, the localization performance improvement is apparent. Focusing specifically on localization efficiency, the performance for HT nodes 11 and 32 remains consistent with

Comparative works	[12]	[11]	[22]	**[Our work]**
HT & Attacks	Flooding	Flooding	Path Collision	**Data Tampering**
ML model	N/A	Perceptron-based ML	N/A	**ANN**
Detection Method	PAC, DLC	ML using BWT, IFI, VCL	CPRD Architecture	**EnhancedANN**
Accuracy	N/A	97 %	N/A	**99.84%**
Precision	N/A	97.6 %	N/A	**99.5%**
Localization Method	Event Handler for Router	MIP Algorithm	CPDD Architecture	**DSCT Algorithm**
Precision	N/A	96.7%	N/A	**88% (S+A) ∼ 93% (S)**
Recall	N/A	N/A	N/A	**100%**
Timing or Data Cost	8∼24us@1.4GHz	30∼140 Cycles	97∼1118 Cycles	**3.7% of tampered data**

PAC: Packet Arrival Curves. **DLC**: Destination Packet Latency Curves. **ML**: Machine Learning. **BWT**: Buffer Waiting Time. **IFI**: Inter-Flit-Interval. **VCL**: Virtual Channel Occupancy **MIP**: Malicious Intellectual Property. **CPRD**: Collision Point Router Detection. **CPDD**: Collision Point Direction Detection. **ANN**: Artificial Neural Network. **DSCT**: Dynamic Security Credit Table. **S+A**: SAME-TOP and AND-TOP protocols. **S**: SAME-TOP protocol only

TABLE II: Comparison between related works with our work

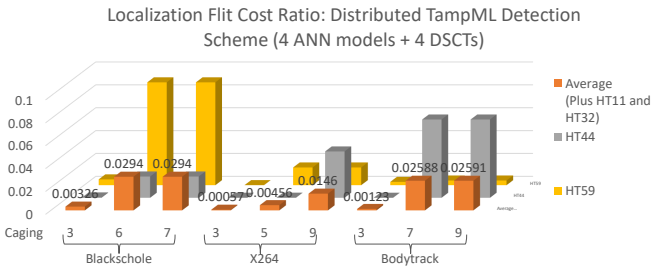


Fig. 14: Flits cost ratio in localization based on the more distributed TampML scheme (4 ANN models plus 4 DSCTs for 4 node clusters).

the outcomes discussed in the early section, so they are not included. However, there is a remarkable improvement in the localization of HT node 44. In the Blackscholes workload, HT node 44 is localized up to **213 times faster**; in X264, the improvement reaches up to **6114 times**; and in Bodytrack, it is up to **3508 times faster**. For HT node 59, there's an increase in localization speed by up to **3.2 times** in the Blackscholes benchmark, while the performance in other benchmarks remains unchanged.

Averagely, this scheme can be **faster as 20.6 times** as the centralised scheme with single ANN and DSCT. This more distributed scheme demonstrates that by dedicating complete detection and localization frameworks to individual clusters within the NoC, we can achieve significantly faster response times, particularly in localizing certain HT nodes, albeit at the cost of increased hardware and power requirements.

E. Overall comparison

Compared to the state-of-the-art, our work is pioneering in proposing and implementing a secure monitoring framework capable of detecting malicious internal communication data and localizing HT-injected nodes within NoC-based MPSoC context. Nonetheless, direct comparison of metrics with other works that focus on different types of attacks presents certain challenges.

Our work demonstrates superior performance metrics in table II. Employing an ANN model, the EnhancedANN detection method achieves an exceptional accuracy of 99.84% and a precision of up to 99.5%, significantly surpassing the benchmarks set by other methods. The DSCT Algorithm used for localization ensures a recall rate of 100%, indicating full detection of relevant instances. Notably, our framework's efficiency is underscored by its minimal data cost, utilizing averagely only 3.7% of tampered data for localization. This efficiency represents a marked improvement over other studies and our previous work in [14], which report lower data or timing costs. Collectively, these results position our work as a leading solution in the field of secure network-on-chip architectures, offering a highly efficient and precise approach to tackling data tampering threats.

VII. Conclusion

In conclusion, this paper introduces a secure framework with a highly efficient monitoring scheme for Network-on-Chips (NoCs) based Multi-Processor System-on-Chip (MPSoC), capable of detecting tampered flits and localizing Hardware Trojan (HT)-injected nodes. The framework demonstrates a detection performance with 99% accuracy and precision on the PARSEC benchmark's real workloads. Localization performance varies between 88% to 93%, contingent on whether the new AND-TOP trigger protocol is enabled. With the AND-TOP protocol activated, localization efficiency improves by up to 11 times compared to our earlier design. Moreover, we have examined a more distributed monitoring and detection scheme, which, despite higher power and hardware resource demands, achieves up to 20.6 times the localization efficiency of the singular framework approach. This framework not only enhances security in NoC-based MPSoCs against communication data tampering attacks but also has the potential to be adapted for other communication bus architectures, such as AMBA systems, where the data packet structure is amenable to ML model training and the malicious IPs can be pinpointed using the DSCT algorithm.

References

[1] C. Liu, J. Rajendran, C. Yang and R. Karri, "Shielding Heterogeneous MPSoCs From Untrustworthy 3PIPs Through Security- Driven Task Scheduling," in *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 4, pp. 461-472, Dec. 2014.

[2] B. Halak, "CIST: A Threat Modelling Approach for Hardware Supply Chain Security," in *Hardware Supply Chain Security: Threat Modelling, Emerging Attacks and Countermeasures*, 1st ed., pp. 3-65, 2021.

[3] S. Charles and P. Mishra, "A survey of network-on-chip security attacks and countermeasures," in *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1-36, 2021.

[4] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware Trojans: Lessons Learned after One Decade of Research," in *ACM Transactions on Design Automation of Electronic Systems*, vol. 22, no. 1, Article 6, May 2016, pp. 1-23.

[5] N. Prasad, R. Karmakar, S. Chattopadhyay, and I. Chakrabarti, "Runtime mitigation of illegal packet request attacks in Networks-on-Chip," in *2017 IEEE International Symposium on Circuits and Systems*, 2017, pp. 1-4.
[6] S. Charles and P. Mishra, "Lightweight and Trust-Aware Routing in NoC-Based SoCs," in *2020 IEEE Computer Society Annual Symposium on VLSI*, 2020, pp. 160-167.
[7] A. Kulkarni, Y. Pino, M. French, and T. Mohsenin, "Real-Time Anomaly Detection Framework for Many-Core Router through Machine-Learning Techniques," in *ACM Journal on Emerging Technologies in Computing Systems*, vol. 13, no. 1, Dec. 2016, pp. 1-22.
[8] C. Sudusinghe, S. Charles, and P. Mishra, "Denial-of service attack detection using machine learning in network-on-chip architectures," in *Proceedings of the 15th International Symposium on Networks-on-Chip*, ACM, Virtual Event, 2021, pp. 35-40.
[9] K. Wang, H. Zheng, and A. Louri, "TSA-NoC: Learning-Based Threat Detection and Mitigation for Secure Network-on-Chip Architecture," in *IEEE/ACM International Symposium on Microarchitecture*, vol. 40, no. 5, Sept. 2020, pp. 56-63.
[10] 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)*, Jul. 2018, pp. 345-350.
[11] M. Sinha, S. Gupta, S. S. Rout, and S. Deb, "Sniffer: A machine learning approach for DoS attack localization in NoC-based SoCs," in *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, no. 2, 2021, pp. 278-291.
[12] S. Charles, Y. Lyu, and P. Mishra, "Real-time detection and localization of distributed DoS attacks in NoC-based SoCs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 12, pp. 4510–4523, 2020.
[13] J. Yao, Y. Zhang, Z. Mao, S. Li, M. Ge, and X. Chen, "On-line Detection and Localization of DoS Attacks in NoC," in *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, Chongqing, China, 2020, pp. 173–178.
[14] H. Wang and B. Halak, "Hardware Trojan Detection and High-Precision Localization in NoC-based MPSoC using Machine Learning," in *28th Asia and South Pacific Design Automation Conference (ASPDAC '23)*, Tokyo, Japan, Jan. 16-19, 2023, pp. 1-6.
[15] J. Frey and Q. Yu, "A hardened network-on-chip design using runtime hardware Trojan mitigation methods," *Integration*, vol. 56, pp. 15-31, 2017.
[16] M. K. JYV, A. K. Swain, S. Kumar, S. R. Sahoo, and K. Mahapatra, "Run time mitigation of performance degradation hardware trojan attacks in network on chip," in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Jul. 2018, pp. 738-743.
[17] J. Hestness and S. W. Keckler, "Netrace: Dependency-tracking traces for efficient network-on-chip experimentation," The University of Texas at Austin, Dept. of Computer Science, Tech. Rep., 2011.
[18] L. L. Woo, M. Zwolinski, and B. Halak, "Early detection of system-level anomalous behaviour using hardware performance counters," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, Mar. 2018, pp. 485-490.
[19] H. Zheng and A. Louri, "Agile: A learning-enabled power and performance-efficient network-on-chip design," *IEEE Transactions on Emerging Topics in Computing*, 2020.
[20] K. Wang, A. Louri, A. Karanth, and R. Bunescu, "IntelliNoC: A holistic design framework for energy-efficient and reliable on-chip communication for manycores," in *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2019, pp. 1-12.
[21] J. Y. Won, X. Chen, P. Gratz, J. Hu, and V. Soteriou, "Up by their bootstraps: Online learning in artificial neural networks for CMP uncore power management," in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, February 2014, pp. 308-319.
[22] C. G. Chaves, S. P. Azad, T. Hollstein, and J. Sepúlveda, "DoS attack detection and path collision localization in NoC-based MpsoC architectures," *Journal of Low Power Electronics and Applications*, vol. 9, no. 1, p. 7, 2019.

Haoyu Wang is currently pursuing a Ph.D. degree at the University of Southampton in the United Kingdom. He earned an MSc degree in Analogue and Digital Integrated Circuit Design from Imperial College London, UK, and a BEng degree with first-class honors in Electronic Engineering from a joint bachelor's program offered by the University of Central Lancashire (UK) and the Beijing Institute of Technology (China). Before commencing his doctoral studies, Haoyu held the position of Senior Silicon Design Engineer in the GFX IP design group at AMD. His research interests encompass secure NoC-based SoC design, computer architecture, deep learning accelerators, and FPGA technology.

Basel Halak is currently an Associate Professor in secure electronics and the Director of the Cyber Security Academy, University of Southampton. He is also a Visiting Scholar with the Technical University of Kaiserslautern, an Industrial Fellow of the Royal Academy of Engineering, a Senior Fellow of the Higher Education Academy, and a National Teaching Fellow of Advance HE U.K. He has published more than 100 refereed conference and journal papers and authored five books on the security and reliability of electronic devices and systems. His research interests include hardware security, digital design, and embedded systems. He is with several technical program committees, such as HOST, IEEE DATE, DAC, IVSW, ICCCA, ICCCS, MTV, and EWME. He is a member of the Hardware Security Working Group of the World Wide Web Consortium (W3C). He is an Associate Editor of IEEE ACCESS and the Editor of the IET Circuits, Devices and Systems Journal.