Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 6.199

IJCSMC, Vol. 8, Issue. 7, July 2019, pg.188 – 202

A Non-Invasive Approach to Detect and Mitigate Hardware Trojan in Network on Chip

Musharraf Hussain[1]; Naveed Khan Baloach[2]; Aamir Anwar[3]

[1]Faculty of Computer Science, The University of Lahore – Islamabad Campus, Pakistan
[2]Faculty of Computer Engineering, University of Engineering and Technology Taxila, Pakistan
[3] Faculty of Computer Science, The University of Lahore – Islamabad Campus, Pakistan
[1] musharraf1990@gmail.com; [2] naveed.khan@uettaxila.edu.pk; [3] aamir.anwar@cs.uol.edu.pk

Abstract— Due to the globalization in the semiconductor industry, the malevolent modification made in the circuitry of the hardware is known as the hardware Trojan (HTs) which have made the security of the chip very critical. To detect and mitigate these HTs in the general integrated circuits the researchers have proposed many methods across the decade. However, this much effort is not made for hardware Trojans (HTs) in the network on chip. In this work, we implement a countermeasure to congeal the network on chip hardware design to prevent any person to do changes in the network on chip design. We propose a collaborated method which performs flit integrity and dynamic flit permutation to eliminate the hardware Trojan inserted into the router of NoC by a disloyal person in the company or a 3rd party vendor corporation. The proposed method has improved the number of receive packets up to 10% more than that of the existing methods which contain HTs in the destination address of the flit. Our proposed method has also decreased the average latency for the hardware Trojan inserted in header, tail and destination field of the flit up to 14.7%, 8%, and 3% respectively over the Runtime HT mitigation method.

Keywords— Hardware Trojan (HT), Denial of service (DoS) attack, Dynamic flit permutation (DFP), Flit De-Permutation (FDeP), Network-on-chip (NOC), Bandwidth depletion

I. INTRODUCTION

As numbers of hardware threats are available in which hardware Trojan is one the most visualize one threat. As the majority of the company are outsourcing fabrication, testing and also the assembly of the chip. Due to these outsourcing of different processes of the chip component the security of the hardware has gone in a very challenging environment [1-4]. Hardware Trojan is the modification in the circuitry of the chip. At the time of fabrication, the ion beams can be used to insert the dormant Trojan which may disturb the functional behaviour of the chip. These kinds of attacks are called the multilevel attack which may defect the overall functionality of the chip. Taking the case study of the crypto module from where it is clear that the attacks poses on the crypto module is more dangerous than that of a single adversely attack. By taking an example if an adversary makes an

attack on the AES module by putting faults in it with the aim to get the cyber key. This can be only maliciously modified by adversary who is a part of that conspiracy. Trojan can be inserted locally or can be distributed in the whole network. The payload and the trigger part of the Trojan can be digital or analog as malicious modification can be done when the temperature of the chip changes [5-10].

The payload circuit defines the threat of the hardware Trojan which may be a denial of service (DoS), may alter the normal operation of the chip or will provide the privileges to the attacker to access the classified memory gap [11-13]. Researcher community for hardware which is the Semiconductor Research Corporation (SRC) they have reviewed that fewer amounts of detection methods of the hardware Trojans are present which are creating the challenging environment for the security of the hardware. The existing methods can be divided into two main categories which are destructive and non-destructive methods for the detection of hardware Trojan. The chips containing Hardware Trojan will harm the overall system accordingly the malicious circuits. Network on the chip is a concept in which reliable and high-performance router for multiple IP cores are equipped with single silicon chips [14-15]. The next paper sections are categorized as follows. In section 2 we have discussed that how the hardware Trojan can be inserted in the NoC. The summarized related hardware Trojan detection and mitigation methods for the network on the chip and our contribution in it are discussed in section 3. The proposed technique is discussed in section 4 which contains the hardware Trojan detection and Network on chip integrity check, the unique and unpredictable permutation selector signal. The experimental results are discussed in section 5. The conclusion and future work are discussed in section 6.

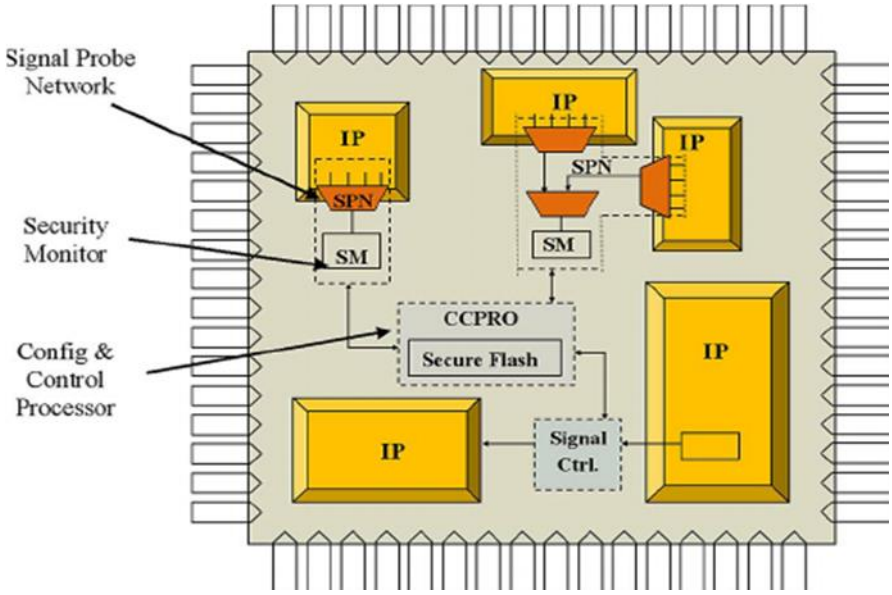

Figure 1: Insertion of Hardware Trojan

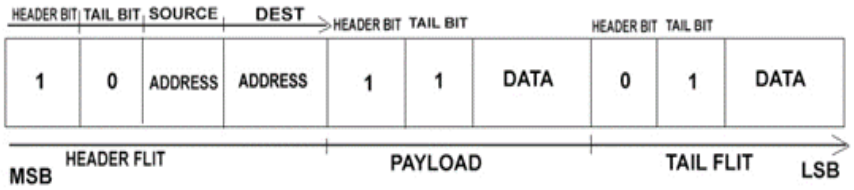

Figure 2: Packet Format Used in This Work

II. HARDWARE TROJAN INSERTION METHOD IN NOCS

A. *Attacking Scenario*

HTs are specifically created with the intent to deliver numerous attacks which include DoS Attack, attacks aimed at leaking information, attacks that maliciously manipulate data or attacks that degrade system performance. HTs can be set in next to the RTL description [16], or directly into the gate level netlist through manipulations in the designing process, which can ultimately lead to logical assaults going on the system. Whereas the designs and arrangements can be changed and personalized to include an HT at the prototyping/fabrication stage by changing the characteristics of the contained circuitry, among those characteristics one may be a postponement. The countermeasure planned in this effort aims to resolve the HTs which has been introduced into the NoC by the unfaithful& fraudulent team members inside the NoC design house, or the 3rd-party company that does the multiple-processors (MPSoC) integration, as shown in Fig 1. Security monitors (SMs) are used to check the functionalities of the System on chip to enable the real time functionality monitoring system [17]. SMs based monitoring systems are used to develop finite state machines (FSMs) and the functionalities of these FSMs are to check the behaviour of interested signal. Signal Probe Network (SPN) [18] is feed by these signals. Security monitor are configuring in such a way that it can't interrupt the functionality of the other Security monitors. To check these behaviours of Trojan the normal operation and trigger circuitry operation are perform concurrently when any deviation is detected from the normal operation.

B. *NoC Baseline Architecture*

The fundamental building block that control the flow transfer in a NoC is a Flit, i.e., the header fits the tail flit, and the many payload flits. Initial bit indicates the header flit and high logic of this bit indicate being there of header flits. The next bit indicates the tail flit and the high logic indicate the entrance of tail flit and the remaining flits contain information which may be routing protocols or data bits, source identifier, destination identifier. The source address flit, destination address flit and the information flit type govern how the NoC router will convey the flits on numerous hop routing passage ways. The packet layout is shown in Fig 2. For the purpose of this work we will be using a simplified version of the packet format that the Intel's NoC use [19]. Malicious amendments of the header flit possibly will result in damaging the integrity of the data, it may also cause the misrouting of the packets, or the flit might be lost altogether. The misrouted packets may violate the deadlock-free routing rules and may cause a deadlock in the network. Deadlock, live-lock and flit letdown will lead to the bandwidth depletion.

C. *Trojans Attacks in NoCs*

NoC which is based on the MPSoCs when the HTs are inserted in it then it may steal the secret information and hijacking of more comfortable in the processing element than that of the NoC routers. The main idea of this work is to strengthen the network on chip design, that why we are considering the type of the HTs attacks which may form the denial of service attacks in the routers of the network on chip. Due to the denial of service attacks which may cause the depletion of the bandwidth, live-lock and misroute in the network on the chip as well as the deadlock in NoC [19]. The HTs in the NoC may also change the destination/source address of the packets and may also change the type of the flit. If the destination HT is inserted, then it may change the destination of the flit and lead the flit to another destination. When the header or the tail HTs are inserted in the NoC, then the flit may keep hold of in the router until some of the act may retune the router. The detail about HT is discussed in [19].

III. RELATED WORK

The security of the software level is discussed in the [20, 21]. The compromised NoCs detection methods at the firmware level are discussed in [22]. To aggravate the attacks the encryption authentication framework method is used from the Network on chip [23]. The security of the NoC from HTs the network interfaces is occupied or then the modules which are directly connected to the IP cores of the chip [23, 24]. To check the memory address method which is to prevent the unsecured zone to communicate with the secure zone is presented in [25, 26]. The method used for the memory protection which is based on the lookup table is used which may provide the authentication against if any malicious packets are present in NoC [27, 28]. To prevent the packets coming from the secure zone to the secure communication barriers are used in [26]. Key keeper and security wrapper are used for encryption of the public and private keys at the core level to prevent the cryptography attacks at the core level. The existing method for the HTs is considering security at the NIs which may have the ability to prevent the illegal memory address and also packet communication at NIs, but there is no method for depletion of bandwidth to be mitigated in the NoC. Some methods of the hardware trojans are implemented in the NoC microarchitecture. For the controlling of the Denial of service attacks when an HTs has

inserted in the NoC wange et al. [29] have proposed a method which will set the upper throughput limit at the switch allocators of the NoC. To the output and input arbiters, some of the control prevent the low-security flits from exceeding their static limits of the security channels which have low limits. A user defined function named counter is used to calculate the throughput of the flits, and if the counter is modified maliciously, then the defence of the security method will be failed.

To get Baron et al [30] have used the camouflaged and denial of services attacks in the NoC a two security wrappers method. The one wrapper has been used for the comparison of the destination address of the ongoing packets from the router to check the legality of the destination address of the packets. The other wrapper [31] which is for counting the number of the cycles that when the packet is inserted in the NoC so that it can postpone the other packets injection if there is any HT present in the network. So by studying this approach, it may block a packet from entering at the non-secure zone to that of the secure zone but if a person who may have access to the security wrapper can add a malicious packet in the network. This method also has the drawback of the latency and area overhead.

The hardware Trojan inserted into the network on chip will get information leakage, unauthorized memory access and denial of service attack which may be due to the incorrect path routing, livelock or deadlock which is due to the malicious modification in NoC. The existing methods which are used for the detection of hardware Trojans are designed for general ICs [14, 17]. However, there are less active numbers of methods which may harden the network on chip design. The work is presented in this need. In this work, we proposed a method which will harden the network on chip against any tempering due to hardware Trojan. In contrast to the existing approaches, the proposed method also contains the full features of the network on the chip which are the parallel transmission channels, high scalability, and high modularity when we strengthen the network on chip against hardware Trojans attacks.

A method by which the restricting address register is applied to the address decoder which may stop bus master/ slaves which have been maliciously modified can't access the restricted address range proposed by Kim et al. [32]. As the drawback of this method is that the restricted address is not fully able to be seen by the master bus and it also has the ability to be changed from the external port so it may be readily modified maliciously by the attackers. Kim et al. [24] have presented a secret code method that the authorized personnel may also easily find the packets which have been modified maliciously but they did not get the risk and cost of that authenticated code, unfortunately. For the bus arbiter a time-based counter and the statistical analyser is used in [33] to find the uncertain bus mastership but the methods which based on the time to detect the HTs are consider wasted when the Trojan is triggered the reason is that it will create the depletion of the packet when a maliciously modified packet is travelling in the network on chip.

Previously work presented in [33, 34] on the assessment of the range to access the memory has been considered the key method that it may be considered to be secured. Although these methods have the drawback to check the range of the memory by itself which will need the protection and when the packets are passed to the memory range, then there is no protection of the packets at the routers and NI. However, there is also a chance that the router and the NI can also be negotiated at the stage of the route computation. So, a person can quickly change the memory address because it has been stored in the output buffers. There are some methods for the HTs to be detected which are present in the NIs but not present in the routers of the network on chip communication [30]. As we have the knowledge that every router has 5 connection nodes with its neighbouring router nodes but only the Network interface has the connection with the IP core and has a link to the router. So, the reason we can think that the router is more prone to denial of service attacks that of NIs.

IV. THE METHOD PROPOSED FOR HT MITIGATION IN NOC ROUTERS

A. *Message Authentication Code (MAC)*

MAC is a procedure based on well-established cryptographic primitives, i.e. symmetric key cryptography; MAC provides message authentication. To accomplish MAC, process the sender and receiver will share a symmetric key which is mostly denoted by K. MAC is encrypted checksum which is generated by the underlying message and which is sent along with that message to ensure the message transferring authentic. MAC algorithm at sender side uses the message which is to send and a secret key K and produces a MAC value which is sent along with the message. Just like HASH, MAC also compresses a high input to an arbitrarily fixed length output. However, the significant difference in MAC and HASH is that the MAC algorithm uses secret key during the compression to occur after that the sender sends the message along with that MAC value. At this stage, we consider the message clear that there is no fault at a stage not confidentially if we need confidential MAC then we have to encrypt the message. On the receiver side, the message and secret key are used to regenerate the

MAC value. Then the MAC received at the receiver side and regenerated MAC at receiver are compared if they both are equal then the receiver will accept the message and also guarantee that the message is sent from an authentic sender. If the received one MAC and regenerated MAC which is calculated at receiver side did not match then the receiver will not accept that message and consider it faulty.

B. *HMAC*

A hash function is a kind of function that maps arbitrary size of data to a fixed size of data. The value which is return by a Hash function is known as hash value, hash codes or simply hashes. One use of the hash function is the hash table which is widely used in different computer software for the lookup of data rapidly. The primary function of the hash function is that it speeds up database or table lookup to detect duplicated data in a large file. To find similar stretches in DNA sequences is an example of Hash Function. A hash function is also useful in cryptography. The cryptographic Hash function has the responsibilities to ensure that the input data will map to an already stored hash value. If the input data is unknown, then it is difficult to construct a hash value for it because it is more difficult to reconstruct the hash value for the unknown input as this is used for assuring the integrity of the transmitted data and this is the building blocks of HMACs because HMAC provides message authentication.

HMAC (keyed- hash-based message authentication code) is a kind of message authentication code (MAC) which contains cryptographic hash function and also a secret key for cryptography. HMAC is mainly used for both data integrity and message authentication. The strength of the HMAC depends upon the length of the underlying Hash function, the size of the hash output and also on the size and the quality of the key used.

Primarily to generate HMAC it has to passes two hash computation. The secret key is used to generate the inner and outer secret key. The first pass of this algorithm generates an internal hash value from the secret key and message and the second pass is used to generate the final HMAC from the inner hash result and outer key. This algorithm provides good immunity against the length, extension, and data integrity attacks.

C. *Implementation of HMAC*

The following pseudo-code explains the implementation of the HMAC algorithm.

If the length of key > block size then
    Shrinkthek the key size by applying a hash function
Else if the length of key< block size then
    Enlarge the key size by the padding of 0's
Compute outer key
Compute inner key
Add inner key, outer key and the message

In this algorithm first, we check the length of the key if the length of the key is higher than the block size then by applying the hash function, we shrink the size of the key. If the size of the key is less than that of the block size, then we enlarge the size of the key by adding 0' s to the key. After that, we calculate the outer key and the inner key which is used in HMAC algorithm. Moreover, at last we return H (outerkey // H(innerkey // message)) it means we return a value in which the inner key and message is concatenated, and then we apply hash function on it, and then this result is concatenated with the outer key and at last we again apply hash function on the overall result and return that value from the HMAC algorithm. So by applying these kinds of restriction, the integrity of the message can be achieved.

V. OVERVIEW OF PROPOSED FRAMEWORK

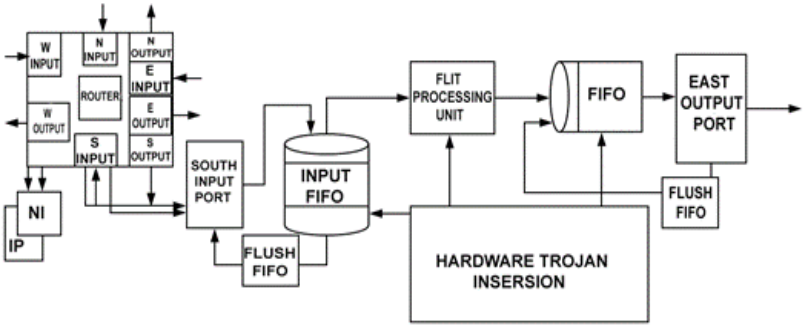

Figure 3: High Level View of Proposed Technique

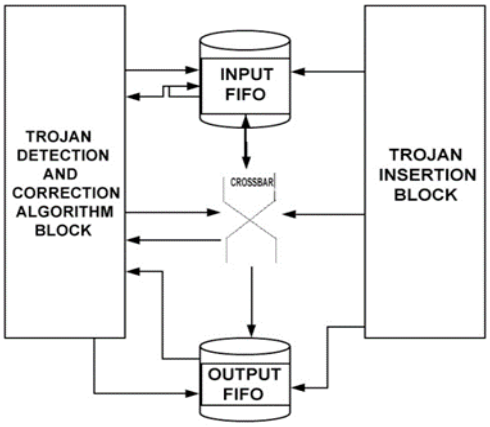

Figure 4: Router Architecture Used in this Work

The proposed methodology is aimed at the detection and mitigation of HTs embedded in NOC at different stages of design and fabrication which maliciously modify the type, destination, integrity of a flit. These attacks will cause bandwidth depletion and are strongly considered in this study. The study assumes to the links of the NoC routers are believed trusted (which is guaranteed by some method presented at [13]). Suitable for this reason of large area getting by the FIFOs it is considered as the FIFO is easier to get compromised rather than that of a submodule of the router just like flit processing unit.

The left side of Fig 3 general NoC is shown, which depicts links and routers with 5 ports which are labelled west, south, east, north and local i/o ports. At the input side of HTC1 module which is dynamically permuting the NoC flit bits before to reach it to FIFO unit. Due to this reason, the NoC flit which is stored in the input of the FIFO is jumbled. Because of the dynamicity & the randomness in the permutation patterns so due to this reason, it is challenging for the malicious attacker to exploit the content of the flit via HT into something meaningful. We are defending our router proactively than that of static protection of the router. The underlying phenomenon of this work is to minimize the chances of the HT that are inserted at FIFO of the router which may change the most significant bits of the NoC packet. The other HTC2 is used for the examining of the flit integrity which can be destroyed via the HTs Inserted at input FIFO of NoC router. The focal purpose behind HTC2 is to stop the malevolent flits from inflowing the network and drops the flits and then flushes out the input FIFO unit if it is necessary. HTs may also have the possibility to be present at the flit processing unit and also at the east output port of the FIFO. So due to this reason, we proposed the HTC3 module which stops the malicious flit from exiting the router in question. These 3 HTC modules which are presented in Fig 3 provide the defence against the HTs to enter and damage the integrity of the flit, and so it reduces the NoC bandwidth.

The detailed explanation of HTC modules of Fig 3 is presented in Fig 4. It is imperative that the critical fields remain unchanged to make sure that a packet will reach its original destination, fields such as destination/source address and the flits type bits. To guarantee that the integrity of the flits should not be changed although in the presence of HTs we have proposed that first, the message should pass through HMAC encoding module. Then in the next step, we are applying a DFP mechanism to the buffers of the packet so that the bits which are considered critical should be randomly reordered before to send that packet to input FIFO. If the attacker doesn't have the knowledge of the dynamic permutation pattern, then the HT inserted by he/she will not work/get triggered as they are expected or to perform the mission as they have been designed. Similarly, the DFDeP is functional by the side of the output port of the FIFO. Later than De-Permutation, the HMAC unit will regenerate another HMAC value which will be compared with the sent HMAC value which will guarantee the correctness of the message.

A pattern selection vector is responsible for achieving randomness in the permutation, and this randomness in the permutation is generated by local vector generator dynamically. The random number generator which is integrated into every router is tasked to produce a selection vector to get solitary of flit permutation outline in favour of the Dynamic Flit Permutation & dynamic flit De-Permutation units.

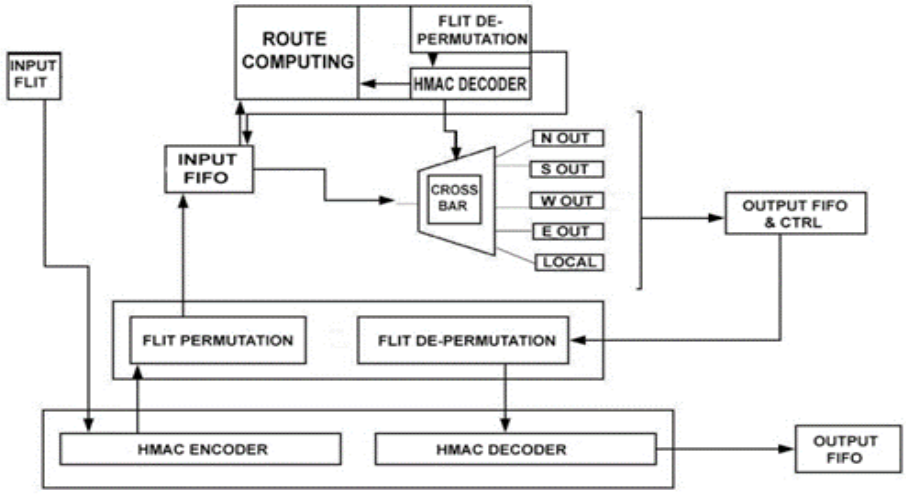

Figure 5: Proposed Hardware Trojan Detection and Mitigation Method

Routing record of DFP for the Dynamic Flit Permutation and Dynamic Flit De-Permutation changes over time& also differs from one router to other. As we know, the routing history of the packet in every router is dependent on NoC. Which appliance the side of operation stage, and the local random generator will have changeable output at the design stage of the NoC. So due to this reason the opponent cannot easily set any HTs that may result in successful attacks.

In the proposed routing computation unit the permuted flits content to get the exact output port of the next router to which is designed. To reduce the cost of the hardware, we are using partial FDP and comparing the HMAC keys to get exact inner and outer signals which may be the status signals and write request of the FIFO buffer.

A. *Proposed Technique for The Dynamic Permutation and Integrity of Flit*

The proposed methodology includes an HMAC algorithm which is used to check the integrity of the data. The overall mechanism used in this paper is overviewed in Fig 5 above. In this algorithm the critical bit like tail, destination address and header field with HMAC algorithm for the purpose to minimize the cost of the hardware and to firmly control the HT inserted in the NoC. We then combine the critical bits with non-critical bits and the code word gotten from HMAC algorithm is permuted before storing in the FIFO buffer inputs. Then in the method proposed for the routing, we partially de-permute the critical bits and the HMAC decoding for the route computation normally. This is due to this reason that the biased de-permutation and the HMAC decoder be used in the direction of to get a valid packet output. If trouncing of the integrity of the flit occurs then we drop the packet and update the permutation packet.

The flit permutation technique used in this paper can prevent the flit content so that attacker cannot quickly produce any HT. This may modify the flits content and also due to the permutation technique the DoS attack can also be prevented. So due to this, not all the critical bits can be changed by the HT and HMAC algorithm to remove the bandwidth depletion and the integrity of the flits.

As shown in Fig 5 the exact same PPS vector is applied to the PFDP, FDP, and FP. In order to further decrease the transparency this is induced via the selection vector generator so that someone can apply the selection vector to the five output and input ports per router. The technique presented in this study can be un-mitigated which can dynamically alter the configuration permutation if the error is perceived in the field of the vital flit area. More especially baseline to compute the route for this module in Fig 5 which is presented in this work [34].

B. *Dynamic PPS Update*

The internal router signals and the time relative of the PPS vector construction. When we want to alter the PPS vector, then we will hang around awaiting the tail flit reach destination which means that flit fully arrives. When the tail flit gets there, then the input barrier checker set the full buffer signal en route for high this is due to this reason that more buffers cannot be entered into the buffer. Signal en route to being high, in sort to discontinue additional flits as of incoming the input buffers.

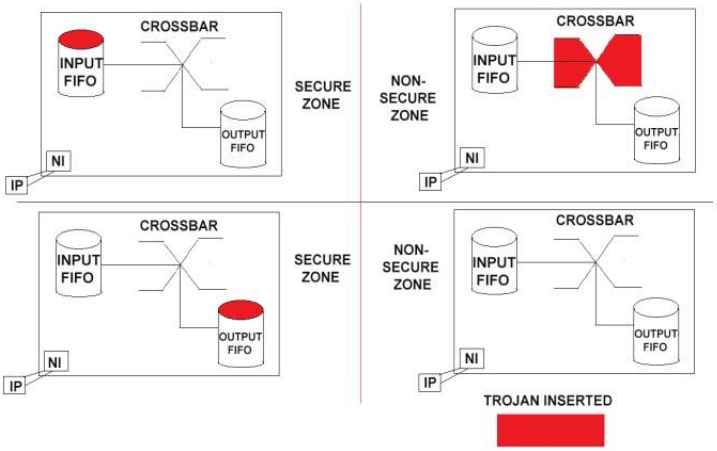

Figure 6: NoC Divided into Secure and Non-Secure Zone

Then network which is responsible for the generation of the PPS vector is turn on, and then it allows entering the next upcoming vector to PPS vector. As the Output of the buffers is got emptied, then the evolution occurs. The evolution then directs to create the next clock pulse in favor of PPS vector production to get afterward random PPS vector by the side of the in-progress time. Then after this, the depressing boundary of clock pulsate is leading to the alterations, that will twist of PPS set-up plus after all, it will set the buffer signals to standard in that order.

VI. EXPERIMENTAL RESULTS

A. *Experimental Setup*

In our proposed method we have presented a 4x4 Mesh NoC which is using XY routing technique. Input and output FIFOs of depth 8 are used. The system is simulated using booksim simulator. Round Robin arbiter is used for the directional precedence in the crossbar. A 3-bit PPS vector is used in favour of the random vector selection of 8 different permutation patterns. Overall NoC is divided into secure and unsecure zones which are shown in Fig 6. IP cores within a secure zone are considered as trusted plus authenticated whereas the IP cores in unsecure zones may have some un-trusted modules. The packet broadcast is limited to secure and non-secure zone under contact rules. The probability of sending and receiving packets for each protected and unprotected zone is equal. We have painstaking dissimilar packet insertion rates (λ) varying from 0.075 en route for 0.2 packets per cycle per node.

B. *Throughput*

We compared our throughput with Jonathan Frey et al. [13] by checking the whole quantity of received suitable packets via NoC NIs during the known simulation moment. The suitable packet is that which reach its original destination somewhat than that of the adapted destination address. By evaluating the performance which is shown in Fig 7(a), 7(b), 7(c) the results show that the anticipated method take delivery of an identical number of packets at identical traffic vaccination rate, in spite of the HTs that are introduced. The method in [36] drops the packet as HTs are inserted. Our technique increases the number of acknowledged valid packets via up to 8% and 12% over Runtime Hardware Trojan mitigation method and baseline respectively when a single HT is inserted. Whereas for 3 DEST HT improvements in some acknowledged valid packets are up to 10% above Runtime Hardware Trojan mitigation method.

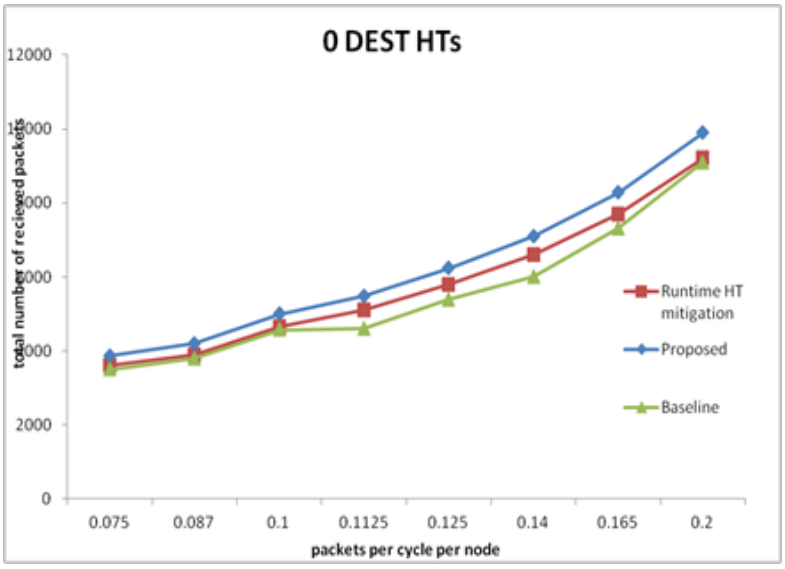

Figure 7 (a): No HT

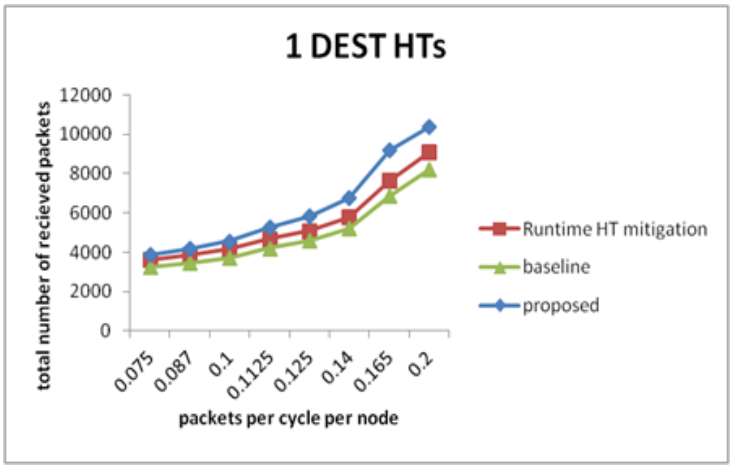

Figure 7 (b): One DEST HT

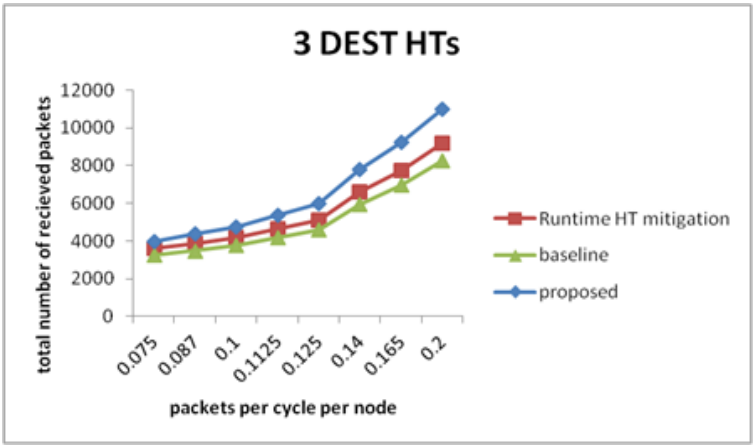

Figure 7 (c): Three DEST HT

The amount of acknowledged valid packets as impacted by the HEAD HTs is shown in the Fig 8. Our proposed technique has a good performance in this situation as well because the Runtime Hardware Trojan mitigation method did not recover as much header flits as our proposed method has done. The number of received valid packets through our proposed method is 14.7% more than that of Runtime Hardware Trojan mitigation method when 3 HEAD HTs are inserted.

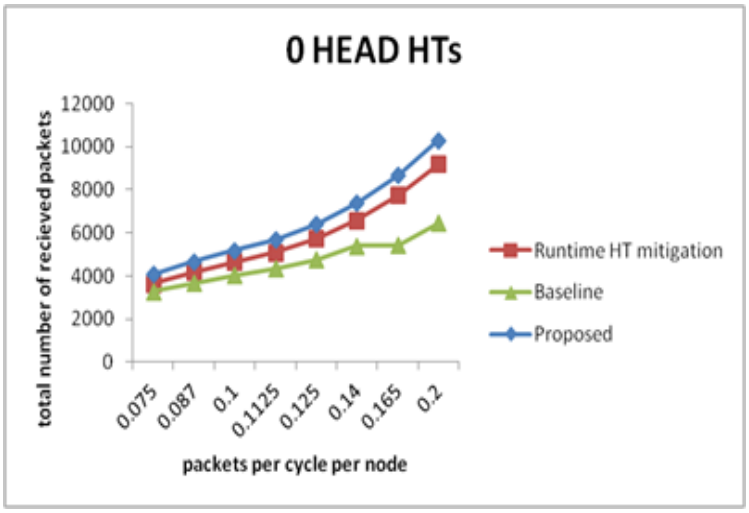

Figure 8 (a): No HEAD HT

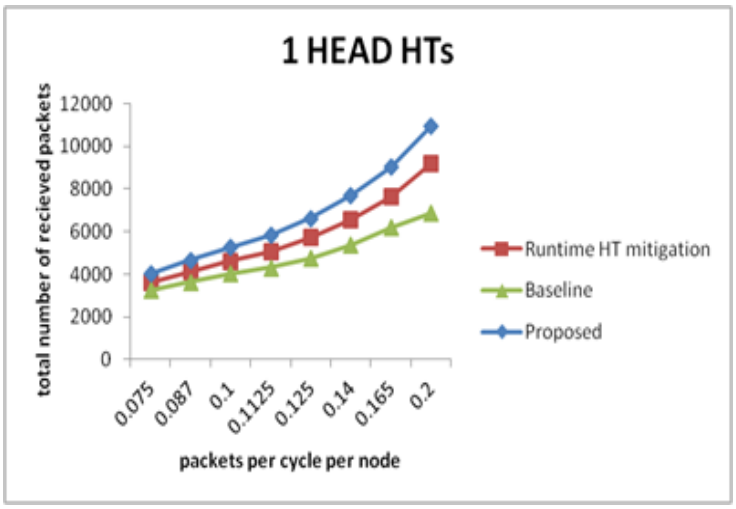

Figure 8 (b): One HEAD HT

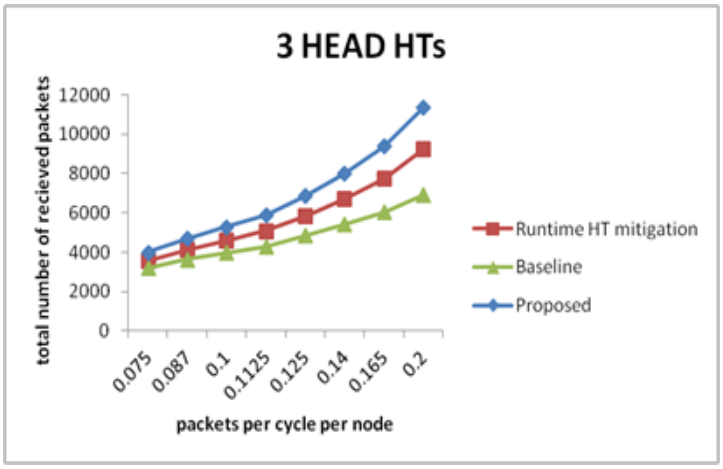

Figure 8 (c): Three HEAD HT

When the TAIL HT is inserted it supplies the tail flit of the packet to misroute. Our proposed method has good performance over the Runtime HT mitigation method. Our method has 3% more received valid packets than of the Runtime HT mitigation method as shown in Fig 9. The performance drop due to tail flit is more than that of the HEAD HT case. This is because when the tail flit is dropped the packets will not release the network resource which will have more effect on the bandwidth depletion.

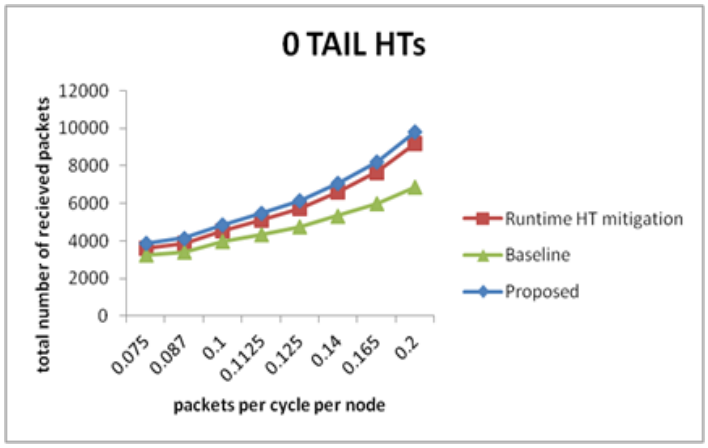

Figure 9 (a): No TAIL HT

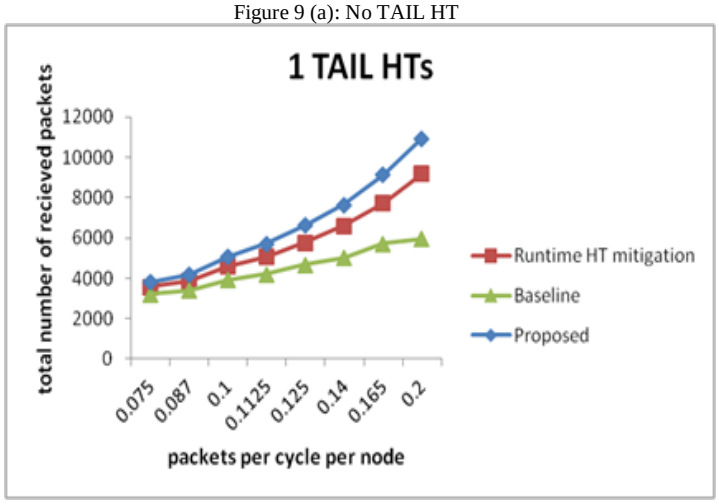

Figure 9 (b): One TAIL HT

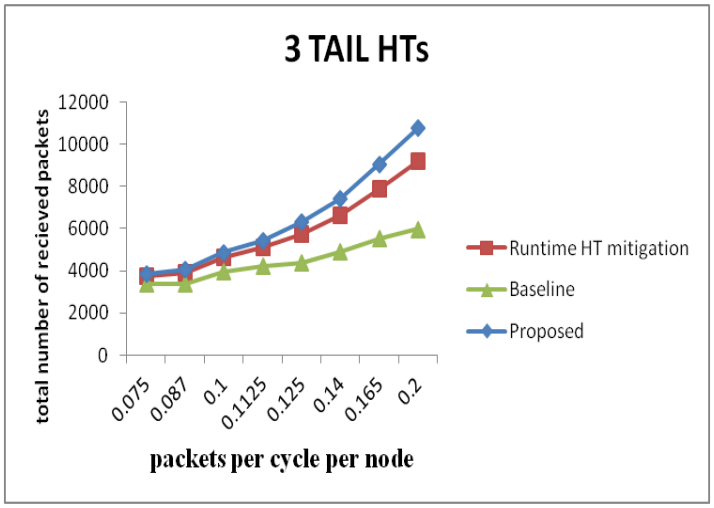

Figure 9 (c): Three TAIL HT

When the tail flit is lost due to the TAIL HT, then we are observing the very significant change in the numbered of the received packets as compare to the NOC when there is no HT present. As it is clear from the Fig 9, our method performs excellent result as compared to the other runtime hardware Trojan mitigation method in the form of getting more valid packets when the TAIL HTs are inserted in the routers. Although when the numbers of HT are increased, then there is a significant change in the throughput of our method and the Runtime HT mitigation method. As it is clear from the Fig 9(b), the performance drop which is more harmful because when the tail flit does not arrive, then it will not release the path through which it is transferring the flits. As we are increasing the number of HTs which are being inserted, then the numbers of routers will wait for the smash to place in tail flits which will prevent new packets to be inserted in the network. Therefore, it will decrease a high number of getting valid packets.

C. *Traffic Hotspot Migration and Bandwidth Depletion*

To calculate approximately the switching task of every router, we have observed the entrance of every new flit at the output port of the router. The number of flits that are transferred to all routers is first collected, and then we draw it in a node type grid using MATLAB which is shown in Fig 10. The switching activities of one router are represented by a square block with the correspondent coordinates in real NoC execution. Different colours represent different intensities of the number of output port switching happen in the whole simulation process. The result shown in Fig 10 is switching tasks for the period of two DESTINATION HTs and two HEAD HTs presents in the NoC. Then we compare the result with the baseline where no HTs is present and with runtime HTs mitigation method where two DESTINATION HTs and two HEAD HTs are present.

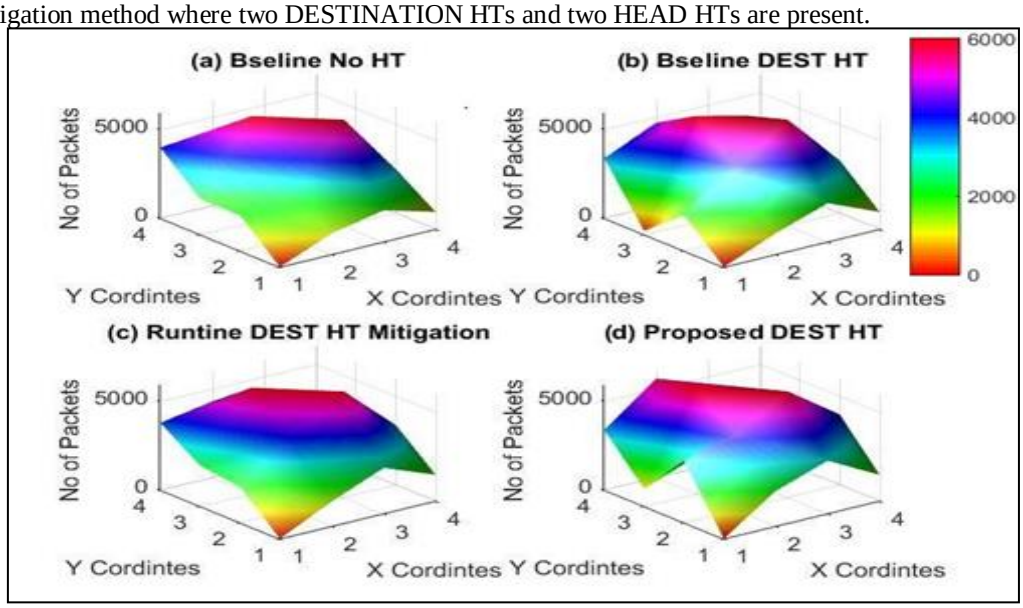

Figure 10 (a): Traffic Hotspot Migration and Bandwidth Depletion Induced by DEST HTs

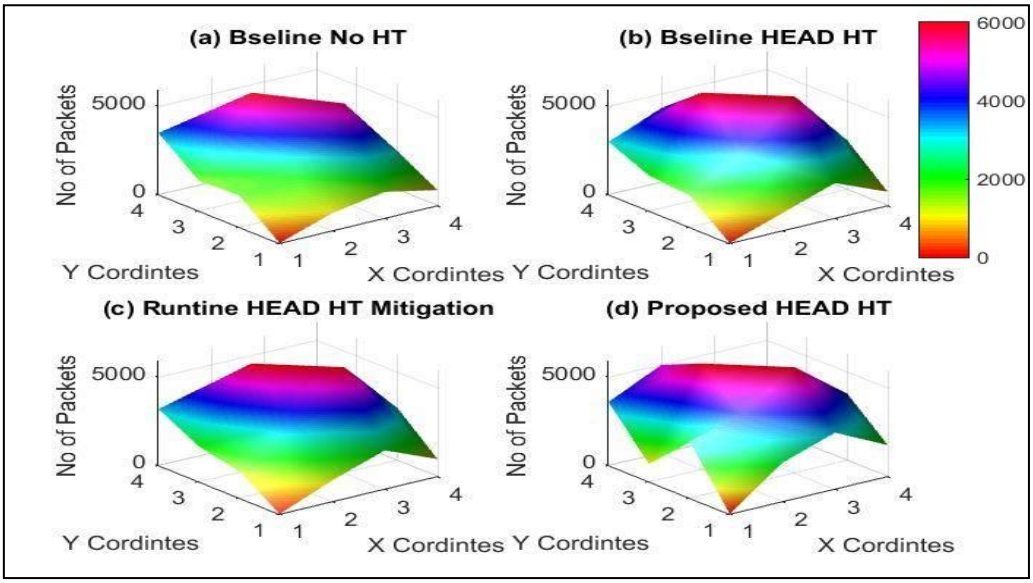

Figure 10 (b): Traffic Hotspot Migration and Bandwidth Depletion Induced by HEAD HTs

Fig 10(a) shows that the proposed technique has the closest outputs as compared to the baseline which contains 0 HTs. As compared to the other method the switching activity is less than that of the NI term because it transmits packets on the counter basis.

In a baseline NoC where two DEST HTs are present, differs from when there is 0 HTs present. It is because the data is being in retreat to the incorrect local port. As an illustration consequently router 1 (Y=1 and X=2) of the baseline NoC has far further switching task when the two HTs are injected as compared to when 0 HTs are present, and this is due to R1's neighbour router, i.e., R0 (Y=1 and X=1), and R5 (Y=2 and X=2) are the vaccination points of the HTs and traffic mitigation are also seen due to the DEST HTs on router R4(Y=2 and X=1).

In case of HEAD HTs, the Runtime HT mitigation method has less effect on preventing the HEAD HT as compared to our technique. The Runtime HT mitigation method receives lesser packets than the baseline NoC when 0 HT is present as shown in the Fig 10b.

D. *Effective Average Packet Latency*

Latency is the time taken by a packet since its injection in the network to the point when its tail arrives at the destination address of the network. The average packet latency of planned technique is shown in Fig 11, which doesn't change when the number of the HTs increases, but it varies in accordance with the increase in the injection ratio of the network.

As compared to the Runtime mitigation method effective average latency of our proposed method is low. The effective average latency is the multiplication of the average latency, and that fraction of the valid packets that are acknowledged by the baseline when there are no HTs presents over valid packets unruffled by the method beneath test as expressed below.

$$Effective_{avg\ Latency} = Avg\ Latency * \frac{Valid\ Packets_{\ Baseline\ without\ HT}}{Valid\ Packets_{\ method\ under\ test}}$$

We are using effective average latency to deem the contact of HTs on the latency comparatively. As shown in Fig 11 our proposed technique perks up the effectual average latency via up to 8%, 14.7% and 3% over the Runtime HTs mitigation method for 3 DEST HTs, 3 HEAD HTs, and 3 TAIL HTs inserted respectively.

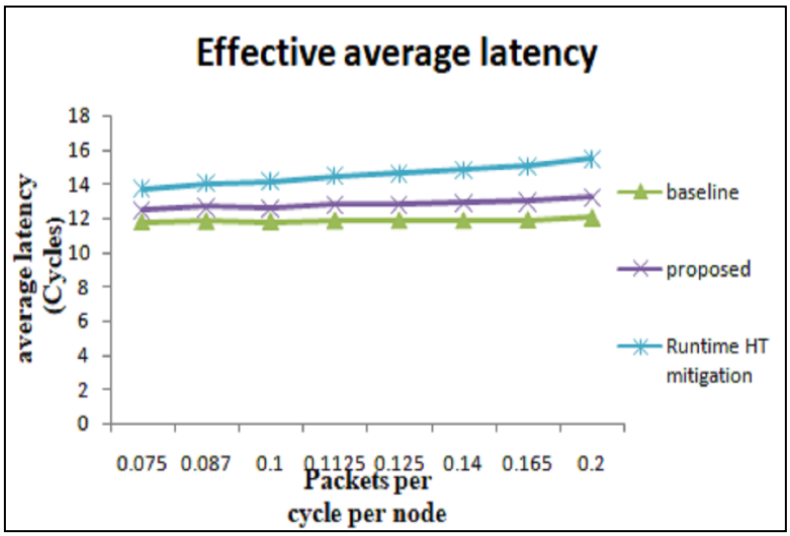

Figure 11: Effective Average Latency

VII. CONCLUSION

Securities of the NoC just like Hardware Trojans are receiving the considerable attention of the researchers. Although some methods used for detecting and mitigation Hardware Trojans are present in NoC. In this work, we propose a method which hardens the NoC for Hardware Trojan attacks. Previous methods are considering

the Network Interfaces, not the routers. To minimize this gap, a dynamic flit permutation and flit integrity methodology that stops an attacker from changing the flits contents in routers is used. A random vector generation is used to achieve the flit permutation dynamicity, which is capable of choosing different permutation for each router based on the history of that particular router. As each router has a different permutation pattern, so it is so difficult for an HT attacker to modify the content in different routers successfully.

The results are evidence that our method performs better for cases involving address filtering, packet termination the NIs and address filtering, packet termination regarding bandwidth depletion, average latency, throughputs in the routers. The result shows that our method performed very well when different HTs are injected into the router HEAD, TAIL, and DEST as compared to the Runtime HT mitigation methods. Along with the increase in the HTs injection and for different variations of the HTs injection our proposed method performs very well. The percentage improvement of invalid packets is up to 14.7% as compared to that of Runtime HTs mitigation method. Our method improves the effective average latency by up to 8%, 14.7% and 3% over the Runtime mitigation method for 3 HTs in the destination address, Header flit and tail flit of the packet respectively.

One limitation of our method is that we are considering the links between routers as trustworthy and in upcoming time we will assess the NoC routine by concerning with a method to remove this limitation.

REFERENCES

1. D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in Security and Privacy, 2007. SP'07. IEEE Symposium on, 2007, pp. 296-310.
2. 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.
3. M. Banga and M. S. Hsiao, "VITAMIN: Voltage inversion technique to ascertain malicious insertions in ICs," in Hardware-Oriented Security and Trust, 2009. HOST'09. IEEE International Workshop on, 2009, pp. 104-107.
4. S. Baron, M. S. Wangham, and C. A. Zeferino, "Security mechanisms to improve the availability of a Network-on-Chip," in Electronics, Circuits, and Systems (ICECS), 2013 IEEE 20th International Conference on, 2013, pp. 609-612.
5. L. Benini and G. De Micheli, "Networks on chips: A new SoC paradigm," computer, vol. 35, pp. 70-78, 2002.
6. S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware Trojan attacks: threat analysis and countermeasures," Proceedings of the IEEE, vol. 102, pp. 1229-1247, 2014.
7. R. S. Chakraborty and S. Bhunia, "HARPOON: an obfuscation-based SoC design methodology for hardware protection," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 28, pp. 1493-1502, 2009.
8. G. K. Contreras, M. T. Rahman, and M. Tehranipoor, "Secure split-test for preventing IC piracy by untrusted foundry and assembly," in Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2013 IEEE International Symposium on, 2013, pp. 196-203.
9. W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in Design Automation Conference, 2001. Proceedings, 2001, pp. 684-689.
10. S. Evain and J.-P. Diguet, "From NoC security analysis to design solutions," in Signal Processing Systems Design and Implementation, 2005. IEEE Workshop on, 2005, pp. 166-171.
11. L. Fiorin, S. Lukovic, G. Palermo, and P. di Milano, "Implementation of a reconfigurable data protection module for NoC-based MPSoCs," in Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on, 2008, pp. 1-8.
12. L. Fiorin, G. Palermo, S. Lukovic, V. Catalano, and C. Silvano, "Secure memory accesses on networks-on-chip," IEEE Transactions on Computers, vol. 57, pp. 1216-1229, 2008.
13. J. Frey and Q. Yu, "A hardened network-on-chip design using runtime hardware Trojan mitigation methods," Integration, the VLSI Journal, vol. 56, pp. 15-31, 2017.
14. C. H. Gebotys and R. J. Gebotys, "A framework for security on NoC technologies," in VLSI, 2003. Proceedings. IEEE Computer Society Annual Symposium on, 2003, pp. 113-117.
15. C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," Proceedings of the IEEE, vol. 102, pp. 1126-1141, 2014.

16. Y. Jin, N. Kupp, and Y. Makris, "Experiences in hardware Trojan design and implementation," in Hardware-Oriented Security and Trust, 2009. HOST'09. IEEE International Workshop on, 2009, pp. 50-57.
17. Y. Jin and Y. Makris, "Hardware Trojan detection using path delay fingerprint," in Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on, 2008, pp. 51-57.
18. R. JS, D. M. Ancajas, K. Chakraborty, and S. Roy, "Runtime detection of a bandwidth denial attack from a rogue network-on-chip," in Proceedings of the 9th International Symposium on Networks-on-Chip, 2015, p. 8.
19. R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware trojans," Computer, vol. 43, pp. 39-46, 2010.
20. L.-W. Kim and J. D. Villasenor, "A Trojan-resistant system-on-chip bus architecture," in Military Communications Conference, 2009. MILCOM 2009. IEEE, 2009, pp. 1-6.
21. G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in Proceedings of the 44th annual design automation conference, 2007, pp. 9-14.
22. Y. Wang and G. E. Suh, "Efficient timing channel protection for on-chip networks," in Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on, 2012, pp. 142-151.
23. H. M. Wassel, Y. Gao, J. K. Oberg, T. Huffmire, R. Kastner, F. T. Chong, et al., "SurfNoC: a low latency and provably non-interfering approach to secure networks-on-chip," in ACM SIGARCH Computer Architecture News, 2013, pp. 583-594.
24. R. Torrance and D. James, "The State-of-the-Art in IC Reverse Engineering," in CHES, 2009, pp. 363-381.
25. S. Narasimhan, W. Yueh, X. Wang, S. Mukhopadhyay, and S. Bhunia, "Improving IC security against Trojan attacks through integration of security monitors," IEEE Design & Test of Computers, vol. 29, pp. 37-46, 2012.
26. J. Porquet, A. Greiner, and C. Schwarz, "NoC-MPU: A secure architecture for flexible co-hosting on shared memory MPSoCs," in Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011, 2011, pp. 1-4.
27. S. R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, et al., "An 80-tile sub-100-w teraflops processor in 65-nm cmos," IEEE Journal of Solid-State Circuits, vol. 43, pp. 29-41, 2008.
28. X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting malicious inclusions in secure hardware: Challenges and solutions," in Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on, 2008, pp. 15-19.
29. Q. Yu and P. Ampadu, "Dual-layer adaptive error control for network-on-chip links," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 20, pp. 1304-1317, 2012.
30. J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Logic encryption: A fault analysis perspective," in Proceedings of the Conference on Design, Automation and Test in Europe, 2012, pp. 953-958.
31. J. Rajendran, O. Sinanoglu, and R. Karri, "Regaining trust in VLSI design: Design-for-trust techniques," Proceedings of the IEEE, vol. 102, pp. 1266-1282, 2014.
32. N. Matveeva, Y. Sheynin, and E. Suvorova, "QoS support in embedded networks and NoC," in Open Innovations Association (FRUCT16), 2014 16th Conference of, 2014, pp. 51-59.
33. L. Lin, W. Burleson, and C. Paar, "MOLES: malicious off-chip leakage enabled by side-channels," in Proceedings of the 2009 International Conference on Computer-Aided Design, 2009, pp. 117-122.
34. S. Saponara, T. Bacchillone, E. Petri, L. Fanucci, R. Locatelli, and M. Coppola, "Design of an NoC interface macrocell with hardware support of advanced networking functionalities," IEEE transactions on computers, vol. 63, pp. 609-621, 2014.
35. J. Sepulveda, G. Gogniat, R. Pires, W. J. Chau, and M. Strum, "Hybrid-on-chip communication architecture for dynamic MP-SoC protection," in Integrated Circuits and Systems Design (SBCCI), 2012 25th Symposium on, 2012, pp. 1-6.
36. H. Salmani, M. Tehranipoor, and J. Plusquellic, "New design strategy for improving hardware Trojan detection and reducing Trojan activation time," in Hardware-Oriented Security and Trust, 2009. HOST'09. IEEE International Workshop on, 2009, pp. 66-73.
37. S. T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou, "Designing and Implementing Malicious Hardware," LEET, vol. 8, pp. 1-8, 2008.

