SECURE NETWORK-ON-CHIP AGAINST BLACK HOLE AND TAMPERING ATTACKS

by

Luka Daoud

A dissertation

submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy in Electrical and Computer Engineering

Boise State University

March 2020

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the dissertation submitted by

Luka Daoud

Dissertation Title: Secure Network-on-Chip Against Black Hole and Tampering Attacks

Date of Final Oral Examination: 4 March 2020

The following individuals read and discussed the dissertation submitted by student Luka Daoud, and they evaluated the presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Nader Rafla, Ph.D., PE	Chair, Supervisory Committee
Gaby Dagher, Ph.D.	Member, Supervisory Committee
Hani Mehrpouyan, Ph.D.	Member, Supervisory Committee
Koushik Chakraborty, Ph.D.	External Examiner

The final reading approval of the dissertation was granted by Nader Rafla, Ph.D., PE, Chair of the Supervisory Committee. The thesis was approved by the Graduate College.

ACKNOWLEDGMENTS

My PhD journey has been full of exciting and motivating moments accompanied with ups and downs. Therefore, I am using this opportunity to express my gratitude to everyone who supported me throughout the courses and research of this dissertation. I am thankful for their aspiring guidance, invaluably constructive critique and friendly advice during the project work.

I would like to extend my sincere gratitude to my adviser, Dr. Nader Rafla, for his invaluable guidance, support, and patience. I am grateful for his continuous encouragement to boost my confidence during stressful times. I would also like to thank Dr. Gaby Dagher for his valuable feedback on my research and concrete discussion of worthy research problems.

I acknowledge my research committee members: Drs. Rafla, Dagher, and Mehrpouyan for their valuable reviews and rigorous feedback that added to the shape of this dissertation. I would like to thank Dr. Benjamin Johnson for his help in using the Cadence tool. It was a valuable guide for synthesizing my proposed architectures of this work.

Furthermore, I gratefully acknowledge the Department of Electrical and Computer Engineering for their financial support and giving me the opportunity to instruct and teach the Digital Systems lab for undergraduate students.

I would like to extend my gratitude to the research lab and my colleagues: Fady Hussein, Shelton Jacinto, Danyal Mohammadi, and Kamran Latif, with whom I contributed in several research projects.

I would also like to thank the International Student Services (ISS) for giving me the chance to interact with many students of various nationalities. I had the opportunity to be exposed and connected to multiple cultures, and virtually lived in their home.

Finally and most importantly, I thank my family back home for their unconditional support, their unwavering confidence in my abilities and their encouragement in all of my endeavors. I am genuinely grateful to all the amazing people who believed in me and supported me on this exciting stage. Without you, this journey would not have been possible. This work is dedicated to all of you.

AUTOBIOGRAPHICAL SKETCH

Luka Daoud is a PhD Candidate at Electrical and Computer Engineering at Boise State University. He has joined the PhD program in August 2014. He received his M.Sc. degree in Electronics and Communications Engineering from Egypt-Japan University of Science and Technology (E-JUST), Alexandria, Egypt, in 2012 and B.Sc. degree in Electrical Engineering from Fayoum University, Fayoum, Egypt. His main research focuses on hardware security, Network-on-Chip, reconfigurable high performance computing, embedded systems, and High-level synthesis (HLS) design. Currently, he is developing trusted and secure architectures for Network-on-Chip.

ABSTRACT

The Network-on-Chip (NoC) has become the communication heart of Multiprocessors-System-on-Chip (MPSoC). Therefore, it has been subject to a plethora of security threats to degrade the system performance or steal sensitive information. Due to the globalization of the modern semiconductor industry, many different parties take part in the hardware design of the system. As a result, the NoC could be infected with a malicious circuit, known as a Hardware Trojan (HT), to leave a back door for security breach purposes. HTs are smartly designed to be too small to be uncovered by offline circuit-level testing, so the system requires an online monitoring to detect and prevent the HT in runtime.

This dissertation focuses on HTs inside the router of a NoC designed by a third party. It explores two HT-based threat models for the MPSoC, where the NoC experiences packet-loss and packet-tampering once the HT in the infected router is activated and is in the attacking state. Extensive experiments for each proposed architecture were conducted using a cycle-accurate simulator to demonstrate its effectiveness on the performance of the NoC-based system.

The first threat model is the Black Hole Router (BHR) attack, where it silently discards the packets that are passing through without further announcement. The effect of the BHR is presented and analyzed to show the potency of the attack on a NoC-based system. A countermeasure protocol is proposed to detect the BHR at runtime and counteract the deliberate packet-dropping attack with a 26.9% area overhead, an average 21.31% performance overhead and a 22% energy consumption overhead. The protocol is extended to provide

an efficient and power-gated scheme to enhance the NoC throughput and reduce the energy consumption by using end-to-end (e2e) approach. The power-gated e2e technique locates the BHR and avoids it with a 1% performance overhead and a 2% energy consumption overhead.

The second threat model is a packet-integrity attack, where the HT tampers with the packet to apply a denial-of-service attack, steal sensitive information, gain unauthorized access, or misroute the packet to an unintended node. An authentic and secure NoC platform is proposed to detect and countermeasure the packet-tampering attack to maintain data-integrity and authenticity while keeping its secrecy with a 24.21% area overhead. The proposed NoC architecture is not only able to detect the attack, but also locates the infected router and isolates it from the network.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

3PIP – Third Party Intellectual Property

ACAP – Adaptive Compute Acceleration Platform

ACK – Acknowledgement

AE – Authenticated Encrypted

AES – Advanced Encryption Standard

AM – Authentication Module

AMD – Algebraic Manipulation Detection

ASR – Authentic and Secure Router

BFT – Butterfly Fat Tree

BHR – Black Hole Router

CDG – Channel Dependency Graph

CMP – Chip multiprocessors

CPU – Central Processing Unit

CRC – Cyclic Redundancy Check

DES – Data Encryption Standard

DOR – Dimension Order Routing

DoS – Denial of Service

DSP – Digital Signal Processor

DyAD – Dynamically Adaptive and Deterministic

e2e – End-to-End

EAC – Ejection Address Checker

ECC – Error Correction Code

EER – Energy-Efficient and Secure Router

FSM – Finite State Machine

GHR – Gray Hole Router

h2h – Hop-to-Hop

HDL – Hardware Description Language

HMAC – Hash-based Message Authentication Code

HT – Hardware Trojan

IC – Integrated Chip

IoT – Internet-of-Things

IP – Intellectual Property

IR – Injection Rate

MAC – Message Authentication Code

MPSoC – Multiprocessors System-on-Chip

NI – Network Interface

NoC – Network-on-Chip

OS – Operating System

OSI – Open Systems Interconnection

PCI – Peripheral Component Interface

PE – Processing Element

PKI – Public Key Infrastructure

PRNG – Pseudo Random Number Generator

QoS – Quality of Services

RL – Routing Logic

RLAN – Runtime Latency Auditor

RTL – Register Transfer Level

S&F – Store-and-Forward

s2s – Switch-to-Switch

SA – Switch Allocator

SDMM – Secure Detection Management Module

SECDED – Single-Error Correction with Double Error Detection

Secure-ACK – Secure Acknowledgement

SMM – Secure Management Module

SoC – System-on-Chip

SP – Scouting Packet

SPIN – Scalable, Programmable, Integrated Network

SSM – Secure Signature Module

SSR – Secure Signature Router

TTL – Time to Live

VCA – Virtual Channel Allocator

VCT – Virtual Cut-Through

VLSI – Very Large Scale Integration

VT – Virtual Channel

CHAPTER 1

INTRODUCTION

1.1 Overview

Rapid advances in technology have boosted unprecedented growth in the semiconductor industry which is impacted by Moore's prediction that the number of transistors in an integrated chip (IC) would double every 18 to 24 months; well-known as Moore's Law [1]. Due to advances in deep sub-micron technology, semiconductor manufacturing is moving towards 7 nm processing technology [2] and by the next decade the physical dimension of CMOS transistors are expected to cross even below 5 nm [3,4]. This leads to increasing the transistor density and allows cramming more components into a single IC creating massive circuits into a single chip.

Following the positive influence of Moore's law, the complexity of the circuits into a chip exponentially increases. In order for design engineers to keep pace with such sophisticated levels of integration, they have come up with a new design methodology, known as System-on-Chip (SoC). The current Very Large Scale Integration (VLSI) technology is able to support high chip density by integrating many computing resources with specific Intellectual Property (IP) cores to build a complete system on a single chip. IP cores are the building blocks of various system on chip designs for embedded system applications. Examples of cores are the Central Processing Unit (CPU), the Digital Signal Processor (DSP), the memory controllers, or the peripheral devices such as Ethernet or the Peripheral

Component Interface (PCI) buss controllers. A SoC may contain hardware cores of digital, analog, or mixed-signal in addition to software cores.

However, with the breakdown of Dennard scaling [5], further clock frequency increases are constrained by practical limits on power dissipation [6]. Because of shrinking the transistor channel, the power dissipation in IC increases with the operating clock frequency, leading to thermal runaway. Therefore, design engineers have moved their focus from improving a single-core performance, by means of higher clock speed and the use of wider and more intricate micro-architectures, to increasing the amounts of parallelism in a system to achieve high performance and efficiency goals [7]. Chip multiprocessors (CMPs) [8] are now the only way to construct a high performance platform by filling up a processor die with multiple, relatively simpler cores instead of just a larger one.

Future designs are expected to integrate hundreds of processing elements (PEs) into a single chip. The processing cores must include an interconnect architecture among themselves and interfaces to peripheral devices. The interconnection architecture consists of physical interfaces and communication mechanisms to allow proper communication among SoC components. Therefore, On-chip communication has a significant impact on the chip-level performance and energy efficiency [9–11]. Usually, the interconnect architecture is based on dedicated wires or shared busses. For a limited number of PEs, dedicated wire architecture is effective. When the number of PEs in the system increases, the number of wires around the PE also increases and thus dedicated wires have poor reusability and flexibility. On the other hand, a shared bus is a set of wires common to multiple cores that is more flexible and totally reusable. The drawback of shared buses is that they allow only one communication transaction at a time. As a result, the communication between other PEs that share the same bus are stalled. The bus communication bandwidth in the system and its scalability is limited to a few PEs. Therefore, it faces the major problem of scalability.

This had prompted the research to attain a new scalable interconnect architecture known as Network-on-Chip (NoC) [12].

The NoC architecture has been proposed as a high performance, scalable and power efficient alternative to the bus-based architecture. The NoC-based interconnection systems are scalable for integrating hundreds of PEs without significant degradation of the performance. It separates the computation system (PEs) from the communication system (NoC). This made NoC ideally suitable for contemporary and future integrated systems [13].

NoC has been adopted as a new interconnect paradigm for IP cores in SoC to overcome the shortcomings provided by the traditional interconnection architectures, wires, and busses. For example, the Intel Teraflop Research chip [14, 15] is an integrated architecture that comprises 80 tiles, each containing a pair of simple floating point execution units arranged in an 8×10 2D array with a NoC-based interconnection. In addition, the Intel Single-Chip Cloud Computer project [16] incorporates 48 Itanium cores arranged in 24 tiles (each tile has 2 cores) connected via a 2D array network of 6×4 routers. Other examples are given by IBM, Larrabee, and Tilera which demonstrate 32, 64, and 100 cores in a single die with NoC backbone interconnection [17, 18]. More recently, Xilinx announced a new programmable device, known as Versal, which is the first adaptive compute acceleration platform (ACAP). The Versal chip adopts processing elements and acceleration engines connected by a programmable NoC [19].

Unlike buses, NoC makes possible sharing wiring resources between several communication flows and at a higher bandwidth, which allows multiple concurrent communications [20]. Figure 1.1 shows the time line for the evolution of on-chip interconnect architectures [21]. In the contemporary era, the NoC has replaced the traditional bus-based interconnection architecture for scalability, high performance, low power, and low design cost. Although there are advantages provided by NoC for integrating complex systems, modern

SoC are exposed to a plethora of security vulnerabilities.

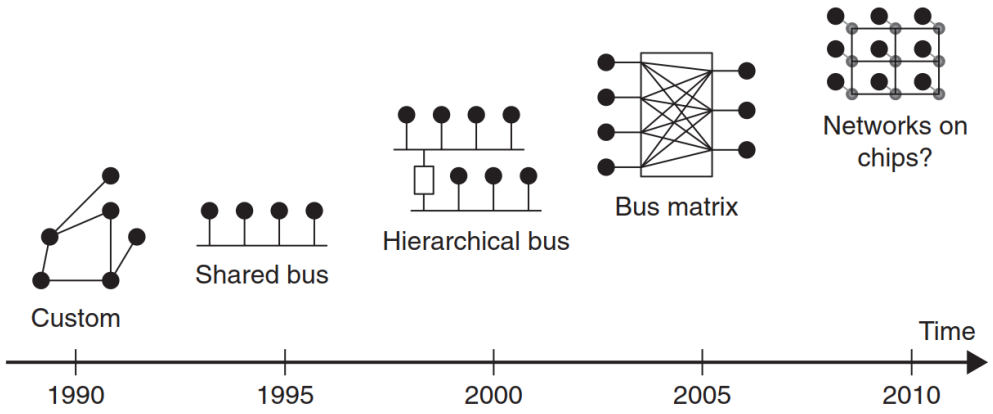


Figure 1.1: Evolution of on-chip communication architectures [21].

The SoC organizational revolution has resulted in a globalized and collaborative supply chain with wide use of Third Party Intellectual Properties (3PIPs) seeking a reduction in time-to-market and the overall design cost. A designed system may be obtained from a various pool of design houses, with a wide array of in-built functionality. This will have far reaching implications towards the security and trustworthiness of the entire chip [22, 23]. 3PIP providers preserve their technological innovations, and limited design information is disclosed which makes the security assurance and verification of 3PIPs a challenge. As a result, many existing techniques based on internal signal inspection are useless for detecting malicious hardware embedded in 3PIPs (e.g.[24]).

The malicious hardware modification of the original design in the chip is known as a Hardware Trojan (HT). HTs can be embedded during any stage of the IC design flow and the manufacturing process of the IC design life-cycle [25]. The aim of such HT attacks is to leak information, degrade the system, manipulate the data, or others [26–29]. HTs are becoming more complex and powerful, such that they are very hard to detect, particularly

complex systems in a single IC which makes it even harder to notice HTs during the test process. Therefore, it became very difficult for SoC designers to pinpoint the physical alterations in 3PIPs. In particular some techniques are applied to camouflage malicious circuits. Recently, a number of cases have been reported where hardware Trojans have compromised the underlying system without being detected. One example is a failure of a Syrian radar system in detecting air strikes due to a HT infection [30]. Another example is demonstrated in [31] where a backdoor was inserted into a military chip in order to access the FPGA configuration. Researchers have also demonstrated many hardware Trojan designs and attack models [32].

In Multiprocessors System-on-Chip (MPSoC), the NoC is the heart of the communication core between processors, which made data transferred from a source to a destination exposed in the on-chip network. This made the NoC a target to security attacks. Additionally, NoC is composed of several based-routing modules and its complexity has increased to guarantee quality of services (QoS) [33]. As a result, the NoC is vulnerable to Hardware Trojan attacks.

1.2 Research Motivation

NoCs are expected to dominate the computing platforms in the near future and take part in the success of future SoCs for embedded applications. NoC platforms will keep pace with technology capabilities and diminish the design productivity gap [34].

The main advantage of NoC is being flexible and dynamically sharing resources, where multiple PEs connected by NoC are dynamically allocated [35–38] and performing various applications simultaneously on a MPSoC. The NoC is the core of the data communication between IPs that could represent a system vulnerability. It plays an important role and can

contribute to the overall system security. Besides flexibility of the NoC, the data and information being transferred between IPs through the NoC are uncovered. This may expose the NoC to security threats, such as data and information snooping, crypto-keys robbing, data altering, hijacking IPs and allowing illegitimate access, shared memory access and data seeping, network bandwidth degradation, and applying Denial of Service (DoS) to degrade the whole system.

It is believed that the Internet-of-Things (IoT) [39] is pervasive in our lives, where several devices interact together through the environment. Additionally, a multiprocessors-based system connected with the environment could run sensitive and critical applications. The possible interference of MPSoC with the environment makes the system vulnerable to attacks which cannot be ruled out. In this case, malicious applications could be downloaded at runtime and infect the processing cores thorough the NoC and run threat attacks on the system.

Much research on the security threats to the NoC have been adopted, but several attacks have not been explored enough and still remain to be counteracted. A search of the literature revealed few studies on the effect of a hardware Trojan in NoC performance and security. However, this work expanded the research on the impact of a HT in an on-chip network to provide a trusted interconnection platform for the IP-cores in the system.

Much research focuses on the impact of hardware Trojans in the NoC while advocating for secure communication among the IP-cores in the system. The main goal of this work is to provide a resilient and secure NoC architecture that mitigates the effects of the HTs and the malicious nodes, such as applying DoS by consciously dropping the data packets from the NoC, thus creating a black hole in the NoC. The Black Hole Router (BHR) is a malicious router that receives the packets from the adjacent nodes and silently discards them and, therefore, the victim packets do not reach their determined recipients. On top

of mitigating HT effects, this work also proposes an authentic and secure on-chip interconnection platform to guarantee various security services of data confidentiality, integrity, and authentication.

1.3 Dissertation Statement

The objective of this research is to answer the following question: **Can we design a secure NoC in the presence of a malicious router?**

More specifically, if a NoC is infected with a malicious element during the design flow, we need to detect and prevent associated internal attacks at runtime.

The goal of this research is to design a secure NoC model that is resilient to denial of service attack while maintaining data-confidentiality, integrity, and authenticity in the NoC even if it is infected with malicious Trojans.

Overall, this dissertation covers the black hole attack in the NoC, where, a HT is hidden in the NoC to silently discard packets from the network without forwarding them to their recipients. It also covers protection to the data packet in the NoC and validating their delivery to the intended destination. This dissertation, additionally, uncovers HT attacks that are inserted in the network router to deliberately tamper with the data packets, to snoop on the transferred data, obtain unauthorized access, or misroute them to apply a denial of service attack.

1.4 Dissertation Contributions

The major contributions of this dissertation are presented in Chapters 4, 5, and 6. The achievements of this research are demonstrated in two main research directions concerning HT attacks and design challenges. The first research direction presents a novel technique

to locate, detect a BHR, and prevent its attack. This work is adopted to provide an energy-friendly technique to detect a BHR with less power and performance overhead. The second research direction presents a novel method to keep the secrecy of moving packets and detect packet-tampering attacks in an infected NoC while maintaining integrity, and authenticity. It also presents a novel technique to detect misrouting and a hidden HT.

1.4.1 Detection and Prevention Protocol for BHR Attack

This work proposes a NoC architecture that mitigates the Black Hole Router attack that drops packets when it is activated.

The contributions made through this work are:

- A new threat model originating from a compromised NoC. This model can be a potent security threat since third party NoCs become common in low cost design.
- A detailed design of a Black Hole Router showing the potency of the associated attack.
- A detailed design of the proposed mechanism to counteract such attacks and guarantee packet delivery even in existence of a Black Hole Router.
- Extensive experiments using a cycle-accurate simulator along with discussion on the effect of the proposed secure protocol on the NoC performance and on its overhead.

1.4.2 Energy-Efficient Detection of BHR

This work proposes a power-efficient NoC model that reduces the power consumption while mitigating the Black Hole Router attack.

The contributions made through this work are:

- A detailed design of a power-gated NoC architecture that detects a BHR with minimum power consumption.
- Extensive experiments using a cycle-accurate simulator along with discussion on the effect of the proposed secure protocol on the NoC performance and its overhead.

1.4.3 Authentic and Secure NoC Platform

This work proposes a secure NoC model for detecting a packet-tampering attack to obtain privilege on the moving packets in the NoC. It demonstrates a novel technique to not only detect the malicious attack but also localize the infected router in the NoC while providing maintaining data-confidentiality, integrity, and authenticity.

The contributions made through this work are:

- Authentic-NoC Model: an on-chip network platform that provides data integrity and authenticity while maintaining the secrecy of the data in the NoC.
- A detailed design of the NoC model to discover the packet tampering attack, packet routing violation, and locate the source of the attack in the NoC.
- Extensive experiments using a cycle accurate simulator along with discussion on the effect of the proposed secure platform on the NoC performance and its overhead.

1.5 Dissertation Organization

Chapter 2 provides the background of the Network-on-Chip and security attacks in embedded NoC-based systems.

Chapter 3 reviews the aspect of security in Multiprocessors System-on-Chip (MPSoC) and provides a literature survey of the security attacks in the NoC.

Chapter 4 presents a study of the Black Hole Router attack on the NoC. It demonstrates the power and area overhead, and explores the effect of their locations and distribution in the network as well.

Chapter 5 presents the contribution to the first proposed research direction. It demonstrates the detection and prevention mechanism for a Black Hole Router that deliberately drops packets from the NoC when activated. In addition, a power-gated technique is proposed to identify BHR and protect the NoC from the attack by an energy-efficient mechanism.

Chapter 6 demonstrates a novel authentic technique to secure the NoC platform to provide packet integrity and authenticity while maintaining its confidentiality. This chapter proposes a technique to localize the source of the attack on the victimized packets.

Finally, Chapter 7 concludes the main research of this dissertation and discusses the potential future research.

CHAPTER 2

BACKGROUND

This chapter discusses the new paradigm of the interconnection of IPs into a single chip (SoC). It is known as a Network-on-Chip (NoC), where the data (message) is divided into packets and then transferred from one node to another through a routing protocol forming a simple network. In this chapter, an overview of the NoC architecture and routing techniques are presented. In addition, this chapter also covers hardware security issues in SoCs, including security threats, hardware Trojans (HTs), and their countermeasure techniques.

2.1 Network-on-Chip (NoC) Platform

2.1.1 Architecture

The Network-on-Chip is an interconnection architecture that consists of three main components: the Network Interface (NI), the Router, and the Link [40]. Figure 2.1 shows a general architecture of the NoC. The NI is an interface module that decouples the computation part (executed by IPs) from the communication part (executed by routers). The Router is a smart engine that directs data packets from a source IP to a target IP through intermediate nodes. The Link is a physical connection, wires, that attaches different routers.

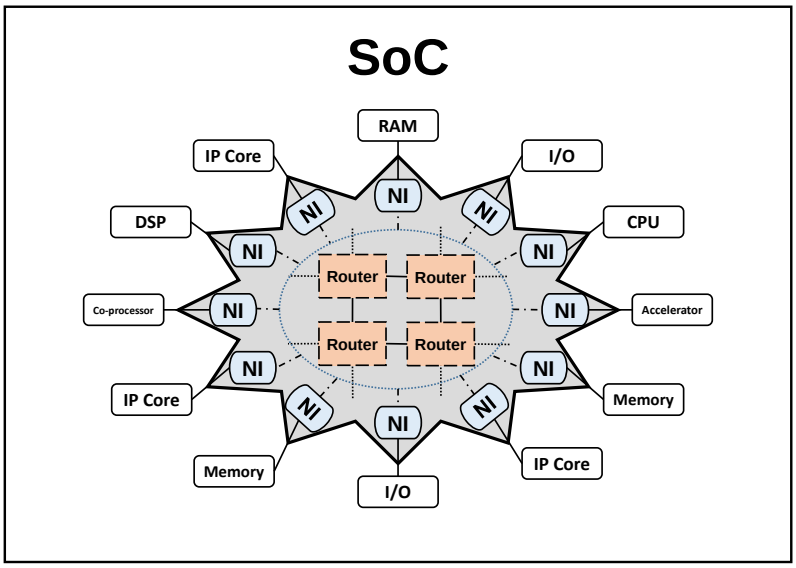


Figure 2.1: General architecture of the Network-on-Chip.

Network Interface (NI)

Each IP core is connected to a respective NI that serves as a logical interface between the communication network and the IP core. The NI is responsible for packetizing or depacketizing the message that is being sent or received, via the network, by an IP core. The NI is generally divided into two main sub-modules: a front-end and a back-end sub-module [41–43]. The front-end module handles the requests of IP cores, while avoiding the knowledge about the network. However, the back-end sub-module is connected directly to the network to handle basic communication services, such as data packetization, network protocols, and reordering packets.

Router

The Router is the main component in the NoC [13]. It is the communication backbone in the NoC-based interconnect architecture. It receives incoming data packets and analyzes their

routing information to determine the best path, based on the routing function. A typical router consists of two main components, the Datapath, where the packets go through from an input port to an output port, and the Controlpath to guide the flow of the data packets in the router. Figure 2.2 shows a classic virtual channel router. A router is built according to the Open Systems Interconnection (OSI) model [44,45] of the NoC. Each layer has its own specific functions to perform [46].

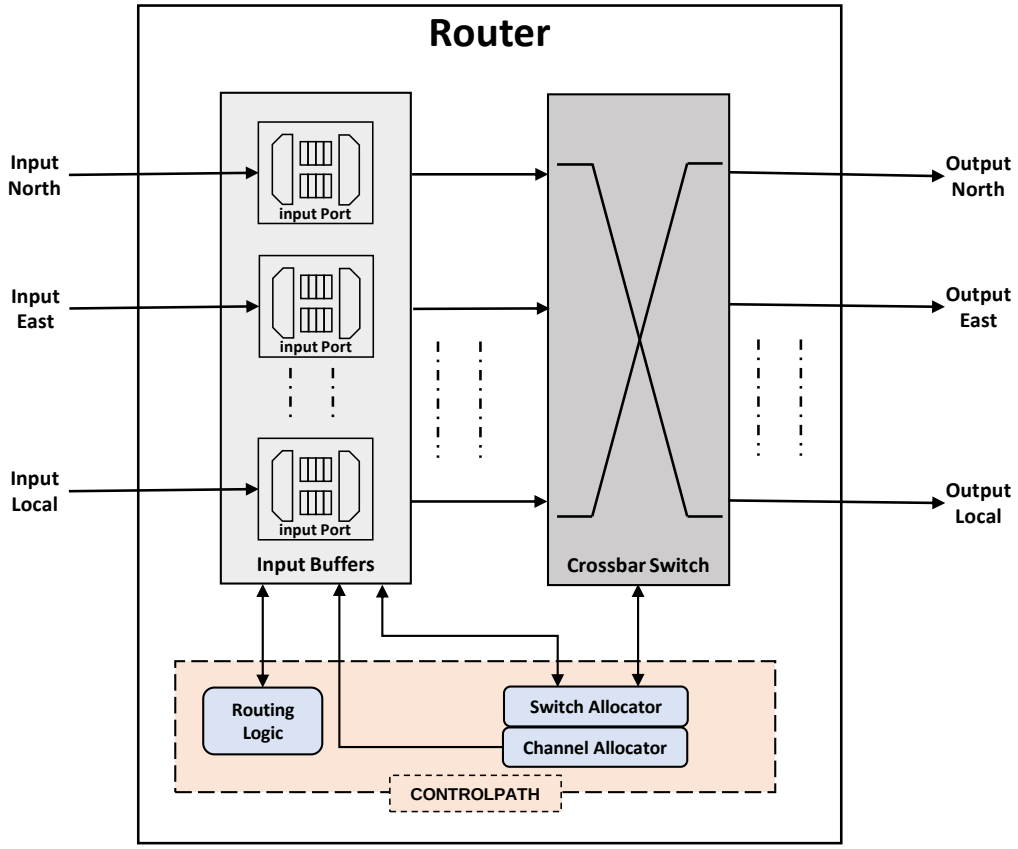


Figure 2.2: Block diagram of a typical router architecture.

The basic router architecture has input ports, output ports, and a local port that is attached to its corresponding IP core. The ports are connected through a switching matrix. The datapath consists of input buffers, and a crossbar switch. It handles the storage

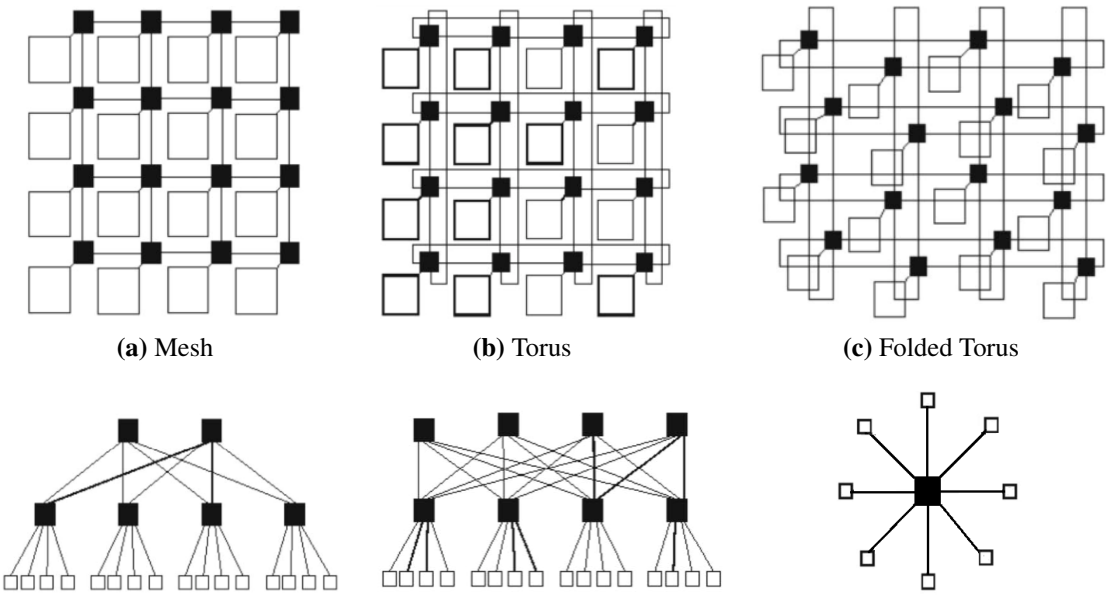
elements and the movements of packets of data or signals. On the other hand, the control path is composed of the *Routing Logic* (RL), the *Virtual Channel Allocator* (VCA) and the *Switch Allocator* (SA) blocks. These modules calculate the routing path, and manage arbitration and resource allocation when a conflict of multiple packets occurs for limited physical resources.

Link

Links are the physical wires that connect the routers in a network, which packets are being transmitted through. Generally, the communication between routers are full-duplex (two physical channels) and the link connection could be serial or parallel. The number of wires per channel is uniform throughout the network and is known as the channel bit width. The channel link is composed of two sets: a data-path for data packets and a control-path for communication and handshaking. The length of the link plays a crucial role in latency, which may require long wires to be partitioned into smaller segments and then a pipeline-mechanism is applied [47]. A link connects different routers in the network and the number of router's ports depends on the chosen network topology.

2.1.2 Topology

The NoC topology determines the physical layout and the connections between network components. It determines how switches and nodes are connected through the communication channels in the platform. The NoC topology determines the number of hops (or routers) that a packet must traverse, and the interconnect lengths between hops. Furthermore, it defines the total number of alternative paths between nodes, which affects the network bandwidth. The topology choice depends on the cost-performance trade-off of the applications. Some of the most popular NoC topologies are illustrated in Figure 2.3.



(a) Mesh **(b)** Torus **(c)** Folded Torus

(d) BFT **(e)** SPIN **(f)** STAR

Figure 2.3: An example of several topologies of the NoC [48].

Mesh

The Mesh topology [34] is a two-dimensional tiled array (grid) network. It consists of m columns and n rows. Figure 2.3a shows an example of a Mesh NoC topology. Their routers are situated at the intersections of two Links and each router is attached with an IP core. Middle routers have five input and output ports corresponding to the four neighboring directions and the local IP core. On the other hand, edge and corner routers have four and three input and output ports, respectively, corresponding to the communication links and IP cores. Due to the realities of semiconductor manufacturing and its layout efficiency, the mesh NoC is the most popular and favored topology among research groups [20].

In [48, 49] many topologies for NoC have been proposed, however, it has been noticed that mesh topology enjoys several advantages over the others. It has regularity in architecture and can be easily implemented inside a chip. The links are short and have equal length.

The routers used within the topology are of same type, except the corner and edge routers. It can be fabricated on a single metal layer. It is also very simple to provide an addressing scheme. Therefore, the 2D Mesh NoC has been addressed in the work of this dissertation.

Torus

The Torus topology [20, 50] is similar to a regular mesh NoC. However, the edge routers are wrapped around to the other side ones, such that each router in the network is connected to four neighbor routers, as shown in Figure 2.3b. The path diversity of a torus is better than the mesh network, and it has more minimal routes. On the other hand, due to the long wraparound channels, the packet transmission delay increases and consequently the network performance is degraded. This can be avoided by folding the torus network as shown in Figure 2.3c

Butterfly Fat Tree (BFT)

A tree topology consists of nodes (vertices) and leaves. In the Butterfly Fat Tree (BFT) topology [48, 51], routers are placed at the nodes and IP cores are placed at the leaves. Figure 2.3d shows an example of a two level BFT. Each router has two parent ports and four child ports.

SPIN

A Scalable, Programmable, Integrated Network (SPIN) [12, 52, 53] is another popular topology. It is similar to fat tree architecture, as shown in Figure 2.3e. Unlike the BFT, where the number of routers at each level reduces by two, the number of parent ports in the SPIN architecture is equal to the number of child ports for every router. There are as many parents (routers) as leaves (IP cores) such that the network is non-blocking, i.e., it is always

possible to establish a connection between any idle IP cores without any effect on existing connections [43, 54].

STAR

A STAR network [55] consists of a central router in the middle of the star, and IP cores or sub-networks at the spikes of the star, as shown in Figure 2.3f. The main issue of the STAR architecture is that the capacity of the central router is quite large, since all the traffic between the spikes goes through it. That causes a remarkable possibility of congestion in the central router of the star.

Others

There are some other proposed topologies such as Octagon [48, 56], Polygon (Hexagon) [55], Spidergon [55], Butterfly, Ring and others [13]. When routers are arranged as an octagon shape, this architecture is called the Octagon topology. The simplest polygon network is a circular network where packets travel in a loop from one router to another. In a Ring topology, the routers are in a ring shape.

Different topologies have a fundamental impact on the network performance, cost, and routing protocols. These criteria allow the design engineers to choose the optimum topology for the target system. In this research, the Mesh network has been used for its simplicity of data routing and architecture which allows the division of the chip into processing regions. So, different protocols may be used in local regions. The work in this dissertation can be applied to other NoC topologies as well.

2.1.3 Switching Techniques

Routers and links, along with the network topology, are employed for providing communication infrastructure for the IP cores. The Data (message) is being moved from a source to a destination in the NoC via routers and links. Based on the router structure and network topology, messages follow a flow-control known as a switching technique. The Switching technique controls the allocation of network buffers and links. It assigns messages to the buffers and the links in the NoC.

When a message is injected into the network, it is first divided into packets, which sometimes are further split into fixed-length flits (flow control units) depending on the switching technique that is used. A message is the logical unit of communication above the network and a packet, however, is the physical unit that makes sense to the network. A Packet is a fixed length data block that contains all the control and routing information in its header. Unlike packets, just the first flit (header flit) contains the routing information; its main task is reserving a path for other flits (data flits) following it, and the last flit (tail flit) releases all the resources reserved by header flit, so all flits of a packet must follow the same route. Figure 2.4 shows an example of packet segmentation. A message is partitioned into packets. A packet is composed of Header and Payload. Each packet carries its routing and sequence information in the header, and the payload contains the data message. Packets are further divided into flits (Header, Body, Tail).

The Performance of the NoC also depends on the switching technique that defines the way and the time of connections between input and output ports inside a switch (router). There exist various switching techniques, but the most popular ones are Circuit Switching, Packet Switching, Virtual Cut-Through Switching, and Wormhole Switching.

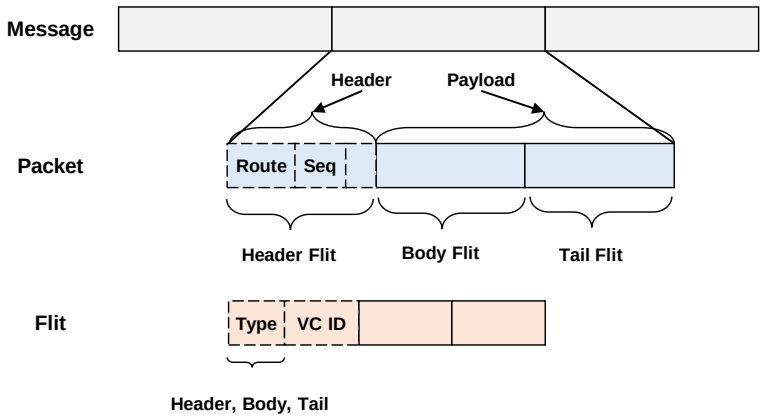


Figure 2.4: Structure of messages, packets, and flits.

Circuit Switching

In circuit switching [54], the physical path connecting the source with the destination is reserved before the start of message transmission and kept busy for the entire duration of the transferring process. The setup procedure is done by injecting the header flit into the network. Since the header flit contains routing information, it moves towards the destination through intermediate routers and reserves the communication links. Once the header flit reaches its destination, the complete path has been assigned to this data transmission and an acknowledgement is sent back to the source. The reserved path is then released when the tail flit has arrived.

This technique reserves network bandwidth for the entire duration of the data transfer while it ties up resources and may cause unnecessary delays. Circuit switching is useful when the messages are infrequent and long. On the other hand, it may block other messages, since the entire physical path is allocated to that transmission.

Packet Switching

In the Packet Switching technique, also called Store-and-Forward (S&F) [54], the message is partitioned into fixed-length packets. Every packet, unlike the circuit switching technique where the path is constructed prior to sending the data, is routed individually to the destination. Each packet has routing and control information embedded into the header, which is used by the intermediate routers to decide the packet's destination. This technique is useful when messages are frequent and short. It also fully utilizes the communication link when there are data to be transferred. However, a packet is completely buffered at each intermediate router before it is forwarded to the next one, require large-sized buffers. This storage requirement at the individual routers can become extensive if the packet size becomes large and multiple packets must be buffered.

Virtual Cut-Through (VCT) Switching

In the packet switching technique, the packet is stored in a buffer, waiting for the whole packet to be completely received before making any routing decisions. However, the size of the packet may be bigger than the width of the physical channel, so transferring the packet will take multiple cycles. Instead of waiting for the entire packet to be received, a router forwards the packet header and the following data once the routing decision has been made and the output buffer is free. In fact, the packet does not have to be buffered at the output of the router and can cut through to the input buffer of the next router before the complete packet has been received at the current router. This switching technique is called Virtual cut-through (VCT) [54].

The VCT and S&F techniques, unlike circuit switching, do not reserve the whole path from the source to the destination, since every packet contains routing information.

However, a physical link between two contiguous routers is reserved during the packet transmission. Additionally, in VCT and S&F, each packet is routed independently and thus they may take different paths.

Wormhole Switching

In wormhole switching [54], packets are further divided into smaller pieces called flits, as shown previously in Figure 2.4. The size of a flit is usually equal to the width of the physical link. The main reason for partitioning the packet into flits is to decrease the buffer size at routers and thus achieve much faster routers. Data is transferred from the source to the destination through the network at the flit level in a pipeline platform. The header flit contains routing information so it reserves resources in routers along the path, which other flits follow, while the tail one releases the resources for subsequent communications. While wormhole switching requires a smaller buffer size, problems due to deadlock and livelock may arise [43]. In case of a blocking, wormhole switching stops every flit in its current router, so it occupies several routers in the constructed path. This blocks the resources and stall flits' movement, causing a deadlock problem [57]. In order to solve this issue, some control logic splits the physical channel into several Virtual Channels (VTs). Each has its own buffer, but they share the same physical link [58].

2.1.4 Routing Algorithms

A routing algorithm determines the path taken by the data packets from the source to reach its destination. Routing methodology has been efficiently used for increasing the network performance and decreasing packet latency. Several routing techniques exist that are generally categorized according to their key characteristics.

Every routing algorithm has a different impact on the network performance. Routing schemes may accompany some problems, such as deadlock, livelock, and starvation:

Deadlock: deadlock is a situation when packets stall and are not able to move. It happens when two or more packets are waiting for each other to be routed forward. For example, if two packets occupy some resources and both are waiting for each other to release the resource, a cyclic dependency between the channels exists, causing a deadlock problem. Deadlock is the most difficult problem to solve.

Livelock: Livelock usually happens in adaptive routing algorithms. It happens when a packet is circulating forever around its destination and the required channel to the destination is always reserved to other packets and not being assigned to it. In this case, the packet will pick any alternative available channel and move to the next switch instead of waiting.

Starvation: Starvation happens when a packet requests a resource that is always granted to other packets. As a result, the packet stops in the traffic and will not obtain the requested resource.

According to the way of defining the routing path, routing techniques can be either deterministic (oblivious), i.e., the routing path is the same between a source and a destination, or adaptive (dynamic), where the routing path from a source to a destination is dynamically changed based on network load, traffic conditions, or other information about the network. Hybrid schemes are also possible where routing algorithms can switch from one technique to another according to the network status. Deterministic routing requires less resources, but it underutilizes the network resources [12]. While adaptive algorithms provide higher throughput and lower latency, they are more complex in implementation, requesting extra hardware, and are deadlock or livelock prone in comparison with oblivious ones. Additionally, packets between the same source-destination pair may take different routes. As a result, they may arrive out of order, and thus a large buffer space is needed for

reordering them [59].

There are various routing techniques in the literature. This work will present a set of approaches being applied or proposed in modern and future multiprocessor interconnects.

Oblivious Routing algorithms

Oblivious routing algorithms have no information about the conditions of the network, such as traffic amounts or congestion.

- **Dimension Order Routing**

  In Dimension Order Routing (DOR), the network topology should be decomposed into several orthogonal dimensions, e.g. hyper cubes, meshes and torus [48]. The routing algorithm is deadlock-free for n-dimensional hyper cubes and meshes since their Channel Dependency Graph (CDG) is acyclic [57]. The data packets move through one dimension until they reach the target coordinate of this dimension, then they switch to the other dimension.

  XY routing: XY routing [60] is a type of DOR routing which routes packets first in a horizontal (X) direction to the target column and then in a vertical (Y) direction to the destination router. This routing algorithm is deadlock and livelock free [60], and well-fit to mesh and torus topologies.

  Pseudo-adaptive routing [60] and Surrounding XY Routing [61] are two schemes derived from XY routing. Pseudo-adaptive routing works in a deterministic mode first, but it switches to an adaptive mode based on the network congestion. Similarly, Surrounding XY Routing works just like basic XY routing as long as the network is not blocked, but it goes around routers that are blocked or inactive.

- **Turn Model**

  Turn model algorithms [62] assign a turn or turns of the packet routing to avoid deadlock between packets travelling in different directions. West-First, North-Last, and Negative-First routings are well-known examples of turn model routing algorithms.

- **Deterministic routing algorithms**

  Deterministic routing algorithms [54] always generate the same routing path for a given pair of source and destination. They usually choose the shortest path. In deterministic routing, the header packets move forward, reserving a new channel at each routing operation, choosing the closer path to the destination. A shortest path routing is one of the simplest deterministic routing algorithms. Packets are always routed along the shortest possible path. Distance vector routing and link state routing are notable examples of shortest path routing algorithms.

  Source Routing: In the source routing technique, the sender makes all the decisions of the routing path for a packet and embeds the whole routing path into the packets' header, such as a vector routing, an Arbitration look ahead scheme (ALOAS) [63], and contention-free routing. They are different version schemes that work the same as source routing.

  Destination-tag routing: A destination-tag routing is also known as a floating vector routing [64]. The address of the receiver – destination tag – is stored into the header of the packet and every intermediate router makes its routing decisions independently.

  Topology adaptive routing: A Topology adaptive routing [65] algorithm works like a basic deterministic algorithm but it has one feature which makes it suitable to dynamic networks. The systems administrator can update the routing tables of the

routers if necessary. A corresponding algorithm is also known as an online oblivious routing [66].

- **Stochastic Routing algorithms**

  They are typically simple and fault-tolerant. However, they are quite slow and use plenty of network resources. Stochastic routing algorithms define the packet's time to live (TTL), which is the time that the packet can wander around in the network before it is dropped off from the network. Flooding Algorithms [67] is the most common stochastic routing algorithm, where routers send a copy of an incoming packet to all possible directions like a flood. The number of copies and the routing directions depend on the type of the flooding algorithm. A probabilistic flood, a directed flood, and a random walk [67] are three different techniques of flooding algorithms.

Adaptive Routing Algorithms

Adaptive routing algorithms do not provide a fixed routing path between a source and a destination. The current network conditions affect the routing decision, which makes the routing more flexible. Therefore, packets could follow different ways instead of waiting for a busy link. Based on the current and destination nodes, an adaptive routing algorithm sets the available output channels and then determines the most appropriate one. Nevertheless, adaptive routing algorithms increase routing flexibility, but the hardware becomes more complex and slower.

- **Minimal Adaptive Routing**

  This algorithm [64] always routes packets along the shortest path. It uses a route which is the least congested.

- **Fully Adaptive Routing**

  A fully adaptive routing algorithm [64] always uses a non-congested route, regardless of it being the shortest path between the sender and the receiver.

  Congestion Look Ahead: In this routing algorithm, information about congestion from other routers are collected and analyzed, then packets will be directed to bypass the congestion [68].

- **Other Adaptive Routing Algorithms**

  Other adaptive routing algorithms can be applied in the NoC. For example, the Odd-Even turn model for adaptive routing [69]. Also, the dynamically adaptive and deterministic (DyAD) NoC system [70]. Q-routing [71] which makes a routing decision based on the network traffic statistics. The Hot-Potato routing algorithm [72] is also an adaptive routing technique, where packets are moving all the time and forward from one router to another without stopping before reaching their destination.

2.2 Security Attacks in a System-on-Chip

Embedded systems are basically built into every electronic device today, ranging from commercial applications such as smart phones and computers, to more critical applications like banking and military systems. The security of such critical applications is highly demanded to protect personal and sensitive information from threats and attacks. If a banking system leaks passwords and secret information or even fails, a tremendous collapse in the wealth of a country could take place. Moreover, if a military system has been cracked, disasters could happen.

Security attacks can be in the form of hardware or software threats. Hardware attacks have emerged as a crucial issue to the security of systems. They can cause various adverse effects on the system, for example changing internal memory values, leaking sensitive information such as secret keys, or degrading the system or even completely damaging the device. Unlike software attacks where they can be always removed, hardware attacks may linger within the system and completely change the functionality of the system. Software attacks that change the actual software may be reversed by overwriting the software with the original software again. Different hardware threats include fault injection, IC counterfeiting, 3rd party intellectual property (IP) piracy, and hardware Trojans (HTs).

2.2.1 Attack Taxonomy

In order to add specific security features to a system, additional costs in the design are applied in terms of design flow modification, and extra hardware or software modules. Consequently, the system performance and power consumption are affected [73]. As a result, design engineers need to deduce the security threats in the system and the requirements to counteract such attacks. This would reduce the implementation cost of the secure system. Therefore, this chapter provides an overview of typical security attacks in system on chip, and their classifications in terms of the way that an attack occurs and its target. Security threats are attacks that exploit the software and physical vulnerabilities through invasive or non-invasive techniques [74]. Attacks could cause severe issues to the privacy of information, the integrity of data, and control of the system.

- **Software Attacks**

  In software attacks, the system architecture is exposed to malicious software agents such as a virus, a Trojan horse, or a worm. These attacks address hidden bugs in the software code, such as those which exploit buffer overflow or similar techniques

[75]. As embedded systems software increases in complexity and functionalities, they become more vulnerable to software attacks in embedded devices, automotive electronics, and several widespread applications. Since hackers can cheaply perform a malicious function through a simple infrastructure, software attacks exemplify the major threat to face in the challenge to a secured system. Additionally, the possibility of modifying functionalities, updating software, or downloading new software applications increases the vulnerability to external attackers and malicious threats to the security of the system. Moreover, systems are connected to the internet such that an attacker no longer needs a physical connection to access the system [76], which reflects an increase in the potential for security threats.

- **Physical Attacks**

  In physical attacks, a physical connection to the system is established in order to directly access sensitive information or interface with it. Attackers should know some details about the system and exploit the characteristic implementation or some of its properties to breach the security of the device. Physical attacks are classified into invasive and non-invasive [77].

  Invasive Attacks: Invasive attacks [77] require depackaging the chip to directly access the internal component. For instance, snooping information carried on the data bus or on the internal wires. Depackaging the chip exposes the system structure, then data wires are attached to micro-probes for observing internal parameters and detecting values on buses, memories and interfaces [73]. Invasive attacks require relatively expensive infrastructure, thus they are much harder to use. However, they can be exploited to gather information about the device, such as information about the system layout and the main components, that can be used to perform other types

of non-invasive attacks.

Non-invasive Attacks: A non-invasive attack only exploits externally available information that are inadvertent emissions which accompany data processing and computations, such as running time, power consumption, and electromagnetic. Unlike invasive attacks, the device is not opened or damaged during the attack. Several types of non-invasive attacks exploit different sources of information gained from the physical implementation of a system, such as power consumption [78], timing information [79, 80], electromagnetic leaks [81], scan-based channel attacks [82, 83] and others, including counterfeit ICs and third party IP piracy [84].

2.2.2 Hardware Trojan Attacks and Countermeasures

Since the work presented in this dissertation considers the security threats of HTs in the NoC, we focus on malicious hardware attacks.

Hardware Trojans

The semiconductor industry has started a globalized business model for the IC design flow. The IC design goes through several stages of design, fabrication, assembly, and test. Because of the global business model, IC design may experience outsourcing. Figure 2.5 depicts the IC design flow, and manifests vulnerabilities connected with outsourcing. Companies consider cheaper ways to sell their high quality product; therefore, they look for the cheapest foundries that will manufacture their product. Since the IC design has been outsourced and handed to a 3rd party, hardware security threats have been increased [85–90]. This has led to undesirable changes and malicious additions to the IC design. These malicious hardware modifications on the original chip are known as a hardware Trojan. HTs can be embedded at any stage of the IC design flow and the manufacturing

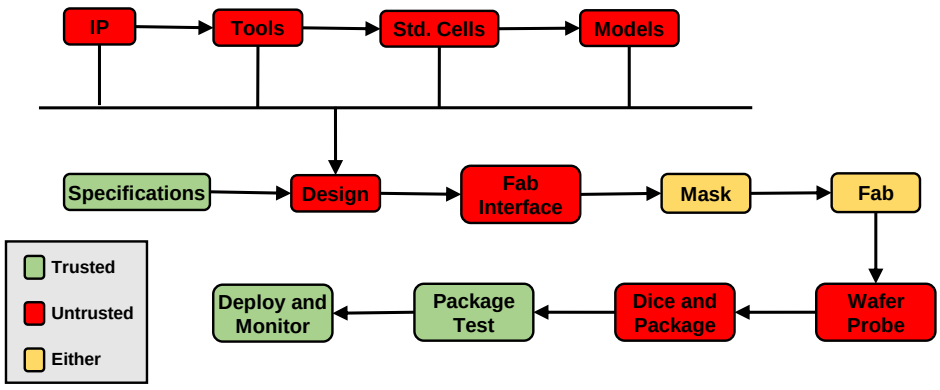


Figure 2.5: Outlined vulnerabilities of the IC design flow [85].

process. They can be embedded at the Register Transfer Level (RTL) in the design's Hardware Description Language (HDL), or straight into the gate level netlist, leading to logical attacks on the system [28]. Also, design layouts can be modified at the fabrication stage to include a HT for changing the internal circuit parametric properties such as delay [25]. HTs must have a malicious goal. They are designed to attack the system in a variety of forms such as a Denial of Service (DoS), information leakage, data manipulation, and system degradation [26–29]. HT designs are becoming more complex and implemented in a way such that they are hard to detect. They become able to bypass robust post-silicon tests due to the nature of the HTs being small enough to consume negligible amount of power compared to the whole system or being triggered by a rare event.

Figure 2.6 shows a hardware Trojan mechanism (Payload is the action or the damage that the HT will execute once it is triggered). HTs can be activated via several types of trigger circuits and then an attack occurs through a payload circuit. HTs can be triggered by analog conditions such as temperature, delay, or voltage variation. Also, digital conditions can trigger HTs such as Boolean logic or sequential events [91].

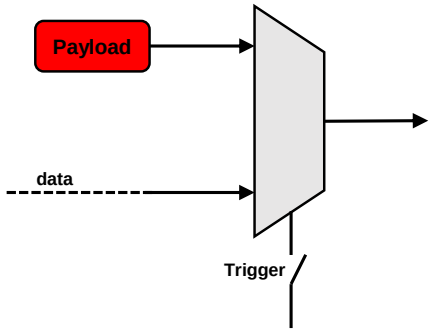


Figure 2.6: An example of a Hardware Trojan scheme.

Nature and Potency of HT Attack

There are three major types of damage or malicious purpose a HT may have. First, it may try to change or control the system's functionality. Second, it may leak sensitive information. Normally this is done through side channels, such as the power network. Finally, it may try to reduce circuit reliability or the lifetime of the system. These type of Trojans include parametric HTs or those that activate applications or functional units to drain resources from the system and reduce the system life.

HT Activation Mechanism

A robust Trojan in hardware is designed to not be in a continuous active state in order to pass the offline test, but it can hold other states such as idle and ready states, waiting for a signal to start the attack. Once it is activated, it applies its malicious payload. A HT can move from one state to another. Activating the HT is the criteria that makes it alive to execute its disruptive job. The activation process can be externally activated (e.g. by an antenna or a sensor that cooperate with the external world), or internally activated (e.g. always active or condition-based).

In the externally triggering mechanism, the target module requires an external input to activate the Trojan. It could be user input (e.g. switches, keyboards, keywords and phrases stream to the input) or component output (e.g. data coming through an external interface such as the RS-232 or an antenna). Additionally, a HT could be activated by the environmental conditions based on the output of a senor that monitors temperature, voltage, or by any type of external interfaces (such as electromagnetic interference, humidity, altitude, or temperature) [25]. For example, a HT can be triggered when the chip temperature exceeds a certain threshold [88]. On the other hand, in the internally triggering mechanism, the activation process is usually based on an internal logic state, a particular pattern of bits, or an internal counter. This will result in a time bomb [92]. Since a HT cannot be always active, it becomes really difficult to detect a Trojan through the testing process.

Hardware Trojan Detection and Challenges

Several Trojan detection techniques have been developed over the past decade. Trojan detection approaches can be classified as either a side-channel analysis or a Trojan activation [25]. In Side Channel Analysis [93, 94], the device goes through side channel examination and analysis of some correlated signals including electromagnetic radiation, delay, and power characteristics. The extracted side channel data is compared to what the "Golden Model" displays to detect the abnormalities in the design. In Trojan activation strategies, testing engineers run the device under several circumstances and input data anticipating that the Trojan will become active and abnormal behaviors will take place. In some cases, this activation detection method is combined with power analysis, where the Trojan has been activated and the malicious circuit will consume more dynamic power.

HT Detection Difficulties:

There are several HTs detection mechanisms, such as an offline detection method that attempts to trigger the Trojan circuit and hence the HT is revealed. Therefore, the switching probability of the trigger circuit is raised at the test time to increase the visibility of HTs [91, 95]. However, for complex circuits, this increases the difficulty of capturing HTs and infected ICs may still bypass offline detection methods. Detection of a HT in a SoC and such suspicious alterations is extremely difficult for several reasons:

- A SoC integrates a large number of soft, firm, and hard IP cores, besides the high complexity of today's IP blocks. Therefore, looking for a Trojan circuit in a system is like looking for a needle in a haystack.

- The nanoscale of IC makes the detection of a malicious circuit by physical inspection and destructive reverse engineering very difficult and costly.

- Trojan circuits are designed such that their activation processes occur under very rare conditions (e.g. sensing power or temperature, or sequence of input bits), which makes them unlikely to be activated and detected using random or functional stimuli.

- An improvement in lithography led to a decrease in the physical feature sizes of transistors which made the process and environmental variations have greater impact on the integrity of the circuit parametrics. Therefore, it is impractical to use simple analysis of these parametrics to detect Trojans.

- Since transistor size is constantly decreasing, it will be difficult for side channel analysis to distinguish the HT impact on the circuit, during the offline test, such as delay and power consumption, due to the process variations of the system. Additionally,

the idea of a “Golden Model” is impractical since the process variations still exist and the HTs can easily be masked in the deviation.

- I/O ports and bit widths are increasing in embedded systems. This increases the testing time for the system verification, which makes the testing process costly in terms of money and time. It is very difficult to detect HTs by logical I/O testing, since HTs may depend on a large amount of combinational logic input conditions that triggers the Trojan circuit.

- For reducing the detection probability, a typical attack does not include HTs in every integrated circuit, however HTs are selectively injected into a portion of the chip population. Therefore, if detecting HTs in some ICs is negative, it does not guarantee that the remaining ICs will be Trojan free.

HTs maliciously can be injected into an on-chip network for information leakage, unauthorized memory accesses, and denial of service (DoS) attacks [96], such as incorrect path routing, deadlock, or livelock [97]. Because of the high complexity of the NoC and the challenges in HT detection, it is very difficult to detect Trojan circuits in a NoC. Thus, run-time solutions can provide a practical defense for the system to mitigate the Trojan effect. Hence, instead of detecting malicious circuits in a system during the post-silicon test, the NoC should be designed to not only protect itself from the HT payload that delivers the attack, but also to detect and to isolate any malicious node in the network system.

CHAPTER 3

LITERATURE REVIEW

This chapter provides a literature review on security threats in the NoC-based systems to-date. It presents the security services and summarizes the security threats addressing the Network-on-Chip. Different types of attacks and countermeasures are presented, along with their effect on the overall system.

3.1 Overview

A Multiprocessors System-on-Chip (MPSoC) is widely used in embedded systems to increase their performance. It is composed of multiple processing elements (PEs), a memory hierarchy, and I/O devices. All these components are connected to each other through a NoC. Figure 3.1 shows a graphical representation of a MPSoC composed of several PEs, memories, and I/O devices.

Since the NoC is the central interconnection platform of the processing cores inside the chip that manages all communications among the Intellectual Properties (IPs), it has become a target of security attacks. Most of the NoCs have been developed without having compromising security issues in the design. So far, most proposed solutions try to secure the IP cores and not the intercommunication itself inside a MPSoC. So, if the interconnection between the IPs is not protected, the whole system will be vulnerable to

security threats. Therefore, including a mechanism to achieve security services in a NoC while maintaining low power and high speed communication is a challenge.

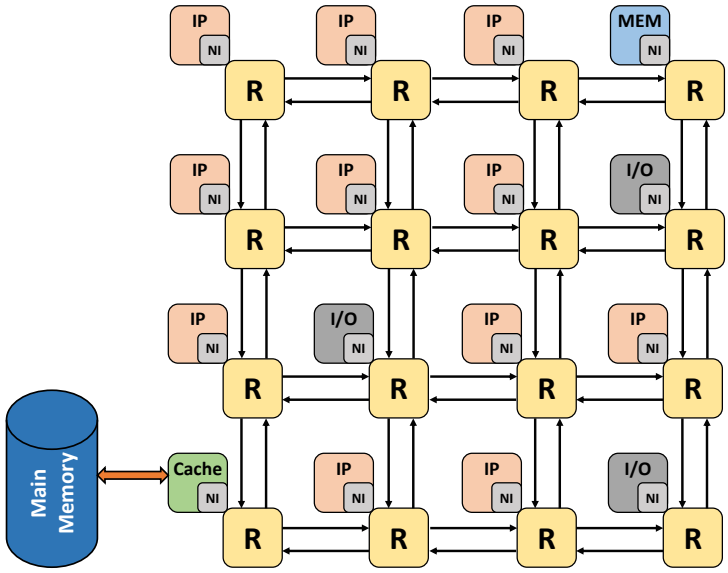


Figure 3.1: An example of a NoC-based Multiprocessors System-on-Chip composed of multiple PEs, memories, and I/O devices connected through the NoC.

3.1.1 Security Services

The security service is a service provided by the communication structure to supply the system with a specific kind of protection to secure data transfer. In MPSoC systems, it can be classified as having primary and secondary attributes [98]. Figure 3.2 illustrates the classification of the security services attributes.

Primary attributes:

- **Data confidentiality:** provides the protection of data from unauthorized disclosure, e.g., sensitive data stored in the memory or packets flowing in the NoC.

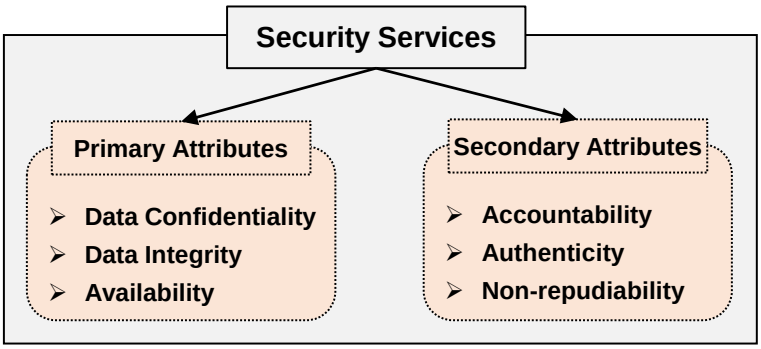


Figure 3.2: Security services attributes.

- **Data integrity:** maintains the data unchanged from unauthorized alteration, e.g., packets moving from the source to the destination should be delivered without changes.
- **Availability:** maintains the obtainability of resources to satisfy the performance requirements, e.g., packets or flits can access buffers if they are available and have the right to do this.

Secondary attributes:

- **Accountability:** availability and integrity of the identity of the IP that performs an operation, i.e., an IP that attempts to access some resources has to provide information that identifies itself to the target.
- **Authenticity:** integrity of a message content and origin, and possibly of some other information, such as the time of emission.
- **Non-repudiability:** availability and integrity of the identity of the sender of a message (nonrepudiation of the origin), or of the receiver (nonrepudiation of the reception). It avoids that any element denies an action, i.e., an IP cannot tamper with its credentials to gain access to protected system resources.

The combination of these security attributes defines a set of security constraints creating a “Security Policy”. Security attributes are provided by security services to ensure the quality of service which guarantees that the MPSoC system has a certain level of performance and data protection.

3.1.2 Security Threats Addressing the NoC

Attackers have different objectives and various attack scenarios to proceed with also. If someone has succeeded with an attack on the on-chip network between the PEs of the MPSoC system, they will be able to steal information, alter data, or degrade the system. Several goals of security attacks and scenarios in the NoC are summarized in Table 3.1.

Table 3.1: Attack Scenarios and Countermeasure Techniques [96].

<table>
<tr><th>Attack Goal</th><th colspan="2">Attack Method</th><th>Countermeasure Techniques</th></tr>
<tr><td>• System Degradation</td><td>Denial of Service:
• Bandwidth depletion
• Incorrect path
• Deadlock
• Livelock
• Black Hole
Power Deprivation</td><td rowspan="3">Hardware Trojan (HT)</td><td rowspan="3">• Firewall
• Security Zones
• Encryption</td></tr>
<tr><td>• Secret Info. Extraction</td><td>Unauthorized Read</td></tr>
<tr><td>• Hijacking</td><td>Unauthorized Write
Unauthorized Reconfiguration</td></tr>
</table>

System degradation

By attacking the communication network between the PEs, attackers can degrade the whole system and its performance.

Denial-of-Service (DoS): The goal of this attack is to waste the network bandwidth and cause higher latency in the packet transfer, resulting in missing deadlines. The following attack scenarios are more damaging because they aim to obstruct channels in the NoC.

- **Bandwidth depletion:** the goal of this attack is to flood the network with redundant packets that have random destinations to occupy some channels in the NoC [99,100].

- **Incorrect path:** this attack injects packets with erroneous paths in the network with the aim to trap them into a dead end to reserve some channels and make them unavailable for other valid packets.

- **Deadlock:** this attack routes the packets to a certain path to intentionally cause deadlock. This leads to the contention of the channel and consequently of a part or of the entire NoC.

- **Livelock:** this attack directs the packets to paths such that they cannot reach their targets and stay turning infinitely in the NoC. This leads to waste of bandwidth, latency, and power.

- **Black Hole:** this attack deliberately drops the packets from the NoC without further information. This leads to packet loss and severe damage to the system [96,101,102].

Power Deprivation: The aim of this attack is to waste more power in the NoC by any means, such as deviating packets in the network to consume more power to reach their destinations. This attack should be taken into account especially for battery-based systems such as a NoC in mobile phones.

Secret Information Extraction

The aim of this attack is to read data in an unauthorized secure target to steal information, sensible data, and crypto-keys.

Hijacking (System Reconfiguration)

This type of attack is the most harmful threat since all security policies are ineffective if the system can be configured by an attacker. The aim of the attacker is to have the write access to the secure area in order to modify or reconfigure the system. In this case, users will fail to have control of the system.

Besides the aforementioned methods of attacks, a HT may be embodied in the system for most kinds of attacks. In this case, the hardware platform is not secure any more. A HT is considered a robust source of attack. If the hardware itself is insecure, the whole system is not safe and no amount of software can secure it [96].

Countermeasure Techniques

A considerable amount of research on security threats in the NoC and their countermeasures has been conducted. It is very expensive to design a NoC to counteract all of the security attacks. Based on the security threat, a countermeasure technique is provided. For example, a firewall around the memory is created to keep the secrecy of the data which authorizes predefined memory spaces for read or write based on the security policy [103–106]. Encrypting packets in the NoC is another method to maintain the data confidentiality [107, 108]. Creating a security zone [109–113] that contains some IPs in the NoC is a mechanism to secure the application running on it. Other techniques are discussed in the study of the related work.

3.2 Survey of Related Work

There have been several studies on security attacks for NoC-based systems. There are various sources of the attacks categorized below.

- A Software-based attack, where a malicious application is mapped to different processing elements in the MPSoC to steal information or apply DoS.
- A Side channel attack, where sensitive information could be read through a timing-side channel attack [114–118].
- A Malicious-NoC attack, where the NoC itself was embedded with a hardware Trojan, or a compromised attack such that the NoC is malicious and works along with a software program mapped onto one or more PEs in the MPSoC.

Security issues in the on-chip network architecture were first studied by Gebotys in [119, 120]. They proposed a framework for security in a general communication NoC at both the network level (transport layer) and the core level (application layer). The work presented a secure exchange of cryptographic keys within the NoC addressing protection from power or EM attacks of a system which is comprised of secure and non-secure cores. This secure structure supports authentication, encryption, and key exchange. However, this framework has to keep track of a large number of keys.

Most software-based security attacks can be revoked by firmware and software updates. However, what if the hardware platform itself is malicious? In this section, the research challenges of the infected NoC with malicious routers are explored. Attackers can embed HTs in the routing nodes to apply security threats, such as extracting sensitive information or causing system degradation. A Trojan in hardware can be in an idle state and waiting for an activation (trigger) signal to run its malicious payload. This makes the HT potent and

very difficult to reveal during post-silicon tests. Therefore, a malicious node in a NoC can be a malignant one [96] that silently applies its attack without any notice.

3.2.1 Hardware Trojan Attacks and Countermeasures

Ancajas et al. in [121] addressed a HT model in the NoC-router that snoops on data and breaks the system confidentiality. The HT is activated by a data-sequence and then it forwards packets to a compromised processor attached to one of the network nodes. For HT threat mitigation, they proposed the Fort-NoC model, which is a mechanism of three layers to protect the on-chip network from third party IPs. The proposed three layers are end-to-end (e2e) data scrambling, packet certification, and node obfuscation to prevent HT activation. Although each layer reduces the risk of triggering the HT, scrambling data and obfuscation fixedly runs the risk of masking an unintended target. While Fort-NoC presents a modest packet security mechanism, it exhibits weakness in the encryption and authentication method, where a static key is used to encrypt the packets and the header of the packet is unprotected. This leaves a loophole for the attacker to comprehend the packet certification and effectively steal senstive data without being detected.

Similar to [121], Hussain and Guo [122] addressed packet leak due to an inserted HT. They implemented a dynamic packet certification, also known as a packet-tag, to overcome the attack analysis and close such loopholes. They proposed a new technique to generate a dynamic tag and scramble it with the data packet. This technique required two separate keys, one to produce the tag, and the other one for scrambling. This technique was able to detect a packet leak, however, it does not detect the location of the malicious router.

In [123], they designed a new packet authentication scheme, where packet-tags are generated at the source node and progressively authenticated during the way to the destination

to reduce the effect on NoC bandwidth. However, this proposed ongoing authentication adds excessive amount of energy and performance overheads.

Hussain et al. in [124] proposed an end-to-end technique to locate HT in an untrusted third party NoC and reduce energy and performance overhead on the system where packet authentication was verified at the final destination. Once a packet fails its validation at the destination node, a localization method is applied to trace back the routing path to find the HT. However, this method fails to detect a hidden HT in some scenarios, as explained in the contribution section of this dissertation presented in chapter 6.

In [125], Biswas et al. analyzed the vulnerability of the NoC router security attacks and their effects on the MPSoC security. They considered two types of router attacks: unauthorized access, and misrouting attacks. In order to detect and prevent such attacks, three different monitoring-based countermeasures were proposed: the Runtime-monitor, the Restart-monitor, and the Ejection address checker (EAC) with an extra percentage area of the router by 26.6%, 22%, and 3.4%, respectively. In addition, Biswas et al. included a local monitoring module inside each router in a non-secure region. This module is based on a time-out counter at each input port to protect the NoC from several malicious effects resulting from a misrouting attack. Although the proposed monitoring technique detects and protects from an attack, it cannot rectify the malicious routing table. In [126], Biswas et al. extended their work and proposed countermeasures with less complexity and area resources.

Yu and Fery [127] addressed HT issues in NoC links. They used an error control coding approach to detect HT-induced link flaws. They applied a reshuffling technique to the link and isolated the attacked link-wires instead of using cryptographic-based rerouting algorithms through alternative paths in order to reduce the overall hardware cost for security protection and the average latency of the NoC. However, their reshuffling technique to

detect errors and correct packets is limited to the two victim wires of the link. Another limitation is that their technique works if the HT is active for a short period of time. In a case that the HT is active for a longer duration, a link isolation algorithm is used and the two-bit errors of the malicious link can be solved.

In [128], Frey and Yu exploited the state-obfuscation technique, inspired from [129] and the logic encryption [130] methods for general IP authentication, to detect and protect the system from a HT that is embedded in the Finite State Machine (FSM) of the NI at runtime. The proposed technique has two phases: 1) key-bits were added to the FSM and dummy states were created to increase the difficulty of meaningful attacks; 2) illegal states were examined to detect the HTs. Unlike the existing approaches of state-obfuscation methods [129], the suggested one provided more paths from the legal states to the illegal one.

In [131], Frey and Yu proposed a dynamic permutation and a flit integrity check method to thwart the HT attacker from modifying the flit type, changing the packet destination address, or sabotaging the integrity of the packet inside the router. Consequently, they mitigated a bandwidth depletion attack. Each router has an independent permutation configuration such that it will be more difficult to perform a successful HT attack via multiple routers. However, attacking the communication link between routers was not considered. Another limitation of their proposed technique is the area and power overhead, since it consumes 39% more area resources and 13% more dynamic power than the baseline design. In addition, the NoC performance was not evaluated nor was considering real secure and non-secure traffic scenarios.

Boraten and Kodi [132, 133] examined a link HT that performs a packet inspection and fault injection attack to apply DoS. The proposed HT model exploited the vulnerabilities created by the fault-tolerant methods. For instance, an Error Correction Code (ECC) can recover data transferred through the link, whereas multi-bit errors using single-error

correction with double error detection (SECDED), corrects only one bit and detects double errors, which triggers retransmitting the data. The purpose of the faults injection is to trigger the ECC to detect errors and request packet-retransmission, hence it starves the network resources by creating deadlock. Unlike [134] and [135] that considered flood-based DoS attacks caused by rogue software, [132] and [133] considered a link HT within the NoC to cause DoS attacks. In order to mitigate the suggested attack, they proposed a heuristic-based fault detection model.

Boraten and Kodi [136] proposed a packet validation technique to protect a compromised NoC from fault injection, side channel, and HT attacks by including two error detection schemes, algebraic manipulation detection (AMD) [137] and cyclic redundancy check (CRC) codes, to ensure data integrity. In normal operating scenarios, a CRC is well capable of detecting faults in packets for low fault rates. However, for sending sensitive data between cores, the system switches to the AMD mode to protect the sensitive information from fault injections attacks. While the security system has been enhanced, the AMD costs significant area and power overhead due to its complexity of encoding.

Rajesh et. al [138, 139] proposed a runtime latency auditor (RLAN) to detect traffic abnormalities caused by a HT that suppresses allocation requests and de-prioritizes arbiters within the router controller. Such a HT gives rise to flit latency and contention, which consequently applies a bandwidth DoS attack in the NoC. RLAN was integrated in the SoC firmware module that interfaces the PE to the NI in order to identify the latency elongation of packets caused by the bandwidth depletion attack. The key problem with the RLAN technique is that it uses delay to detect a DoS attack, which is difficult because several factors influence packet latency during normal operation.

In [140], Zhang et al. demonstrated the effectiveness of two types of HT-assisted DoS attacks, the sinkhole and the blackhole attacks, that target the NoC-based multiprocessor

system. In their attack model, packets are either dropped or diverted to other malicious nodes. The authors only analyzed the effectiveness of the proposed attacks. However, neither a detection nor an avoiding technique was proposed to countermeasure the security threat.

In [101], the effect of the Black Hole router in the NoC was studied. The associated attack based on the number of infected nodes and their distribution in the NoC was analyzed. In [141], a HT model that applies a DoS attack by misdirecting the packets in the NoC was proposed. The authors presented a detection technique and a secure routing scheme. However, the detection method is based on a deterministic routing algorithm.

The main goal of this dissertation is to detect such malicious nodes at run-time and avoid them through secure routing protocols. Unlike fault-tolerant routing algorithms where the faulty nodes are dead routers and are not involved in packets' routing, malicious nodes are nodes that participate in packet routing and apply their payload to breach the system security or degrade the system. In contrast to malicious nodes, faulty nodes are easy to detect and avoid. Malicious nodes are seemingly part of the network, but they attack the system once they are triggered. A run-time detection technique and avoiding such malicious nodes are needed and they are the research challenges of this dissertation.

This chapter has demonstrated the current research literature on HT attacks in the NoC. The next chapters presents the contribution of this dissertation. The work focuses on two attacks: 1) the black hole router attack, where packets are intentionally discarded from the NoC. 2) the packet-tampering attack, where packets are altered to gain the privilege of the NoC and obtain unauthorized access or to apply a DoS attack.

CHAPTER 4

ANALYSIS OF THE BLACK HOLE ROUTER ATTACK IN THE NOC

In this chapter, a HT model that applies a DoS attack by deliberately discarding the packets from the NoC is proposed. The infected router that drops the packets is also known as a Black Hole Router (BHR). The effect of the BHR attack on the NoC is studied. The power and area overhead of the BHR are analyzed. The influence of the location of the BHRs and their distribution in the network is demonstrated as well.

4.1 Threat Overview

As pointed out in the background overview, Chapter 2, HTs can be embodied at any time of the design process and it is hard to detect them due to their potency in hiding. Such malicious circuits are designed as "wolves in sheep's clothing", where they are concealed and are considered as part of the system, however, they quietly attack the platform. It seems possible that a router (node) in the NoC can be infected with a HT which sinks packets as they pass through it and does not forward them to the next hop, forming a Black Hole Router.

"***A Black Hole Router*** is a node in the NoC where incoming or outgoing packets are silently dropped without further information to the system [101]."

Therefore, any incoming or outgoing packets to or from a BHR are disappeared from the NoC. The BHR is classified as a malignant node [96] and in order to silently drop the received packets, it follows the communication handshaking successfully and then swallows them as soon as they are handed off. Figure 4.1 shows a scenario of a BHR (R3). The handshaking (shown as dotted lines) between R1 and R2 are done successfully and the packet is completely forwarded. Similarly, a packet is sent from R2 and handed over to R3 auspiciously. However, R3 silently drops it and does not transfer it to R4 without further notice. It seems to the source (S), R1, and R2 that the packet is being forwarded successfully and neither R4 nor the destination (D) is aware of that drop.

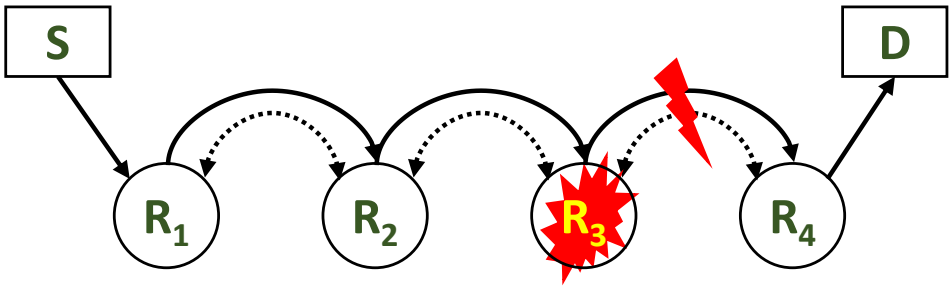


Figure 4.1: A scenario of a Black Hole Router (R3) that discards packets received from R2 [101].

4.2 A Black Hole Router Targeting A Mesh NoC

A HT can be inserted into the control unit of the receiving module of a router, as shown in Figure 4.2, in order to alter the control signals of the buffers of the input ports. Once the HT is activated, it can then attack the received packets.

In this threat model, the HT has two parts: 1) the configuration circuit, which is in charge of activating and deactivating the BHR attack, and 2) the payload circuit, which executes the packet-dropping attack. Each input port needs a HT module, so there are a

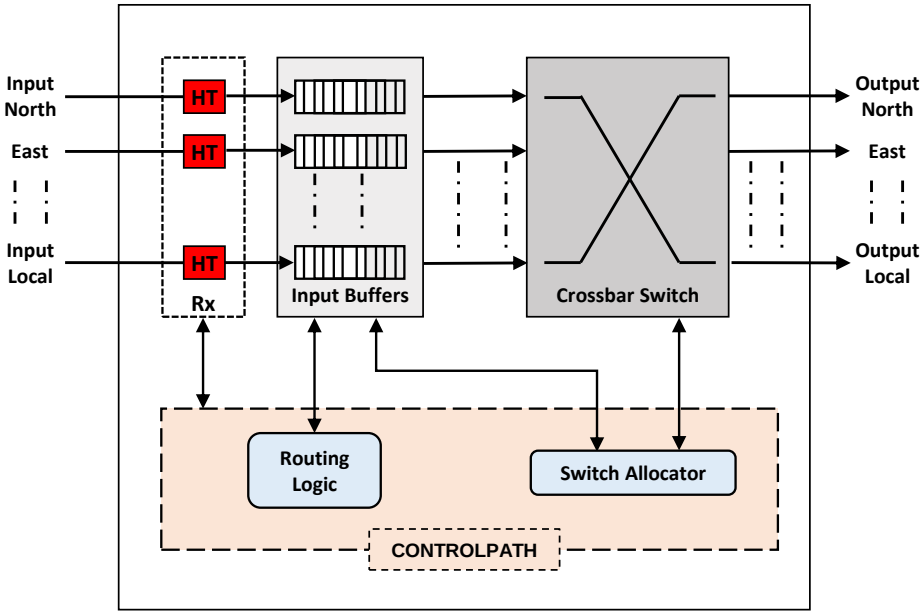


Figure 4.2: An example of a HT inserted in the input ports of a router.

total of five HT circuits that are synchronized to activate and deactivate the BHR attack. Figure 4.3 shows the trigger circuit of the HT and the Black Hole function module. The "Configuration & Trigger" circuit is designed as a Finite State Machine (FSM), where the HT moves from one operating state to another. In order for the HT to move among different states, it first needs to be activated through a trigger signal. A HT can be activated internally or externally.

The HT activation process is the criteria that makes it alive to execute its disruptive job. In this threat model, before the packet-dropping attack takes place, the attacker needs to follow a few preliminary steps to configure and activate the HT. Initially, the HT state is "idle" and it changes to an "active" state when a coded sequence of flits are received at the associated input port. Once the HT is in an active state, a predefined command may change the HT state to the "attacking" state, then it triggers the "Black Hole Attack"

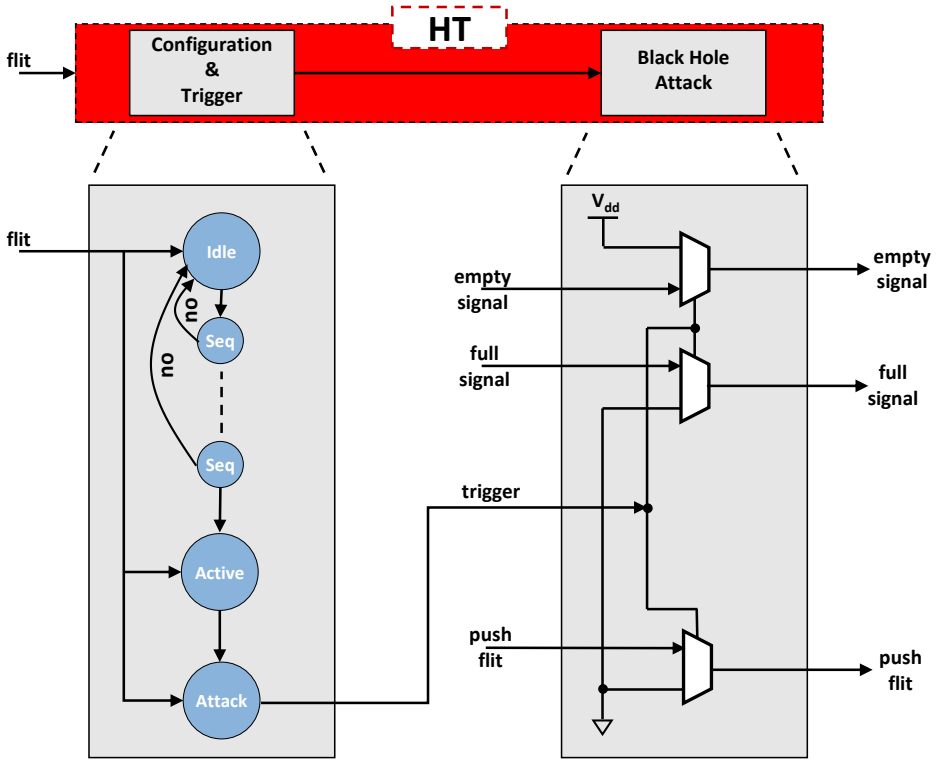


Figure 4.3: The HT structure including the trigger circuit and the Black Hole circuit at an input port.

module. The predefined coded sequence between the attacker and the HT trigger circuit may be generated due to a random traffic. The probability of generating K bits randomly is defined as P_{rnd}.

$$P_{rnd} = (\frac{1}{2})^K$$

Therefore, the probability of generating the coded sequence for five flits of 32 bits each is $\approx 10^{(-48)}$, which is very less likely to be generated randomly. The values and number of the consecutive flits are customized by the attacker design, according to their needs.

Based on the HT status, it can be in one of the following states: idle, active, or attacking. These three states are described in Table 4.1. In the idle state, the HT is inactive and waiting for the activation (trigger) signal. Once it is activated, it applies its malicious payload. The

HT can move from one state to another.

Table 4.1: States of the Hardware Trojan.

State	Description
Idle	The Trojan is inactive and waiting for a trigger signal.
Active	The Trojan is ready and waiting to start its attack.
Attack	The Trojan applies its attack in the system.

4.3 Black Hole Router Attack Evaluation

In this section, the area and power overheads of a BHR in the NoC, along with the effect of its attack on the network are evaluated.

4.3.1 Area and Power Overhead

A robust HT is the one that has an unnoticeable footprint which makes it very difficult to detect. Additionally, the HT is designed to have three states (*Inactive, Waiting, Attacking* as described in Table 4.1.), which make it even harder to be discovered during the offline test. In order to evaluate the area and power overhead of a BHR, a moderate size, five-ports router, with an 8-flit depth FIFO was designed in the C/C++ language, targeting the high-level synthesis (HLS) [142], and the register transfer level (RTL) design was extracted using Vivado HLS [143]. The design was synthesized using 45nm TSMC technology (using the Cadence Design Compiler) for the area and power analysis. The baseline router has an area of 43,180 μm^2 and power of 826.03 μW. The baseline router was modified with the HT-based model, along with the trigger circuit. The malicious router area and power overhead increased by 1.98% and 0.74%, respectively, of the baseline router, which is too small to be revealed during the post-silicon test.

4.3.2 Experimental Setup

In order to evaluate the effect of the BHR attack on the NoC, the proposed threat model was developed in a cycle-accurate simulation environment using SystemC and was integrated with the Noxim simulator [144]. The baseline network size is a 8×8 Mesh-based NoC with an X-Y routing technique. In particular, the simulations with different NoC sizes were performed and their analysis results were very similar to what is provided in this section. A set of certain factors are known which impact on the BHR-based infected network, such as the number of the malicious nodes and their locations in the NoC, the traffic distribution of the packets, and the routing technique. In order to fairly analyze the effect of the BHRs location and their distribution in the NoC, the spatial distribution of the traffic was set to a uniform random traffic. Table 4.2 shows the configuration parameters of the NoC.

Table 4.2: The NoC parameters for the BHR attack evaluation.

NoC parameter	**value**
NoC size	8×8
Packet Size	4 Flits (32 bits/Flit)
Buffer Size	8 Flits
Switching Technique	Wormhole
Routing Algorithm	X-Y routing
Traffic Pattern	Uniform

4.3.3 Experimental Results

In the first experiment, the effect of the location of the BHR in the NoC was analyzed in terms of the number of dropped packets. In this experiment, only one node of the NoC was infected with a BHR at a time. The total number of dropped packets was measured. The simulation was repeated for another infected node and the outcome results were recorded. The simulation was run again to cover each node in the NoC. Figure 4.4 shows the ratio of

the packet loss to the total injected packets in the network. For example, when the node [3, 4] was the only one infected with a malicious circuit, it dropped 12.3% of the total injected packets in the NoC. What can be clearly seen in this figure is that the packet-dropping rate

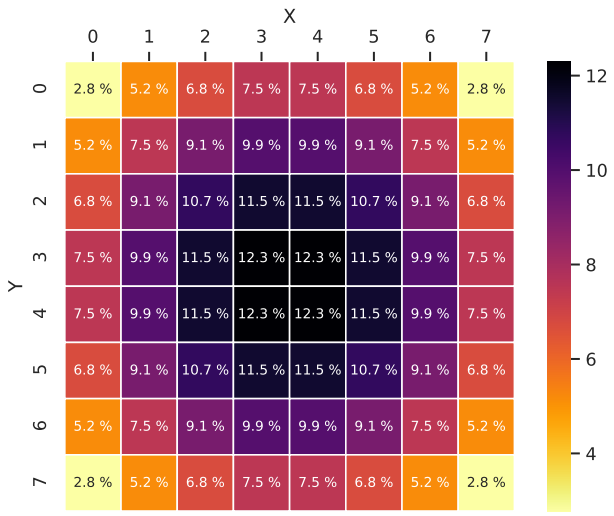
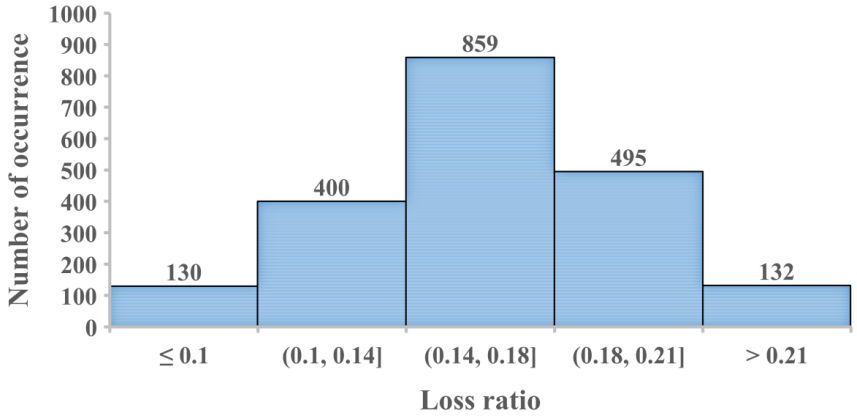


Figure 4.4: Packet loss percentage for each individual node [101].

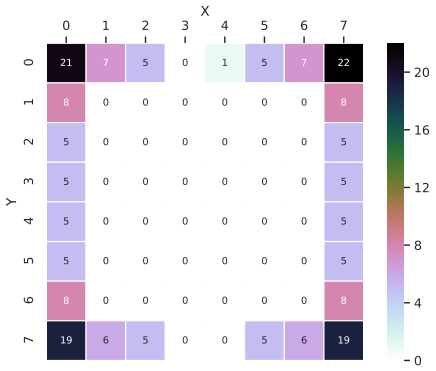
increases when the infected node is closer to the middle of the NoC. Therefore, the location of the BHR in the NoC plays an important role in the attack.

In the second experiment, the NoC was infected with two BHRs at a time. Therefore, there are 2016, i.e. $\binom{64}{2}$, different distributions of two BHRs in the NoC. Figure 4.5 shows the analysis of a NoC attacked by two BHRs. The ratio of the dropped packets varies from 5.2% to 24% for all the combinations of the distribution of the two BHRs in the NoC. Figure 4.5a shows the frequency of the dropping rate periods. It can be noticed that around half of the possible combinations of different BHRs' distributions have 14% to 18% of packet loss rate. Figure 4.5b and Figure 4.5c represent the distributions of two BHRs for the 100 minimum and maximum dropping rate, respectively. The number in the box represents how many times this node was involved in the distribution of the paired BHRs.

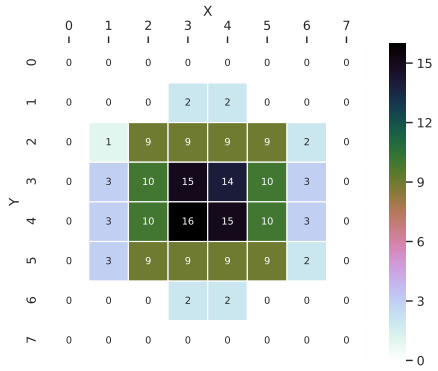
It is obvious that the closer pairs to the center influence the maximum packet drop.



(a) Frequency of the packet loss rate



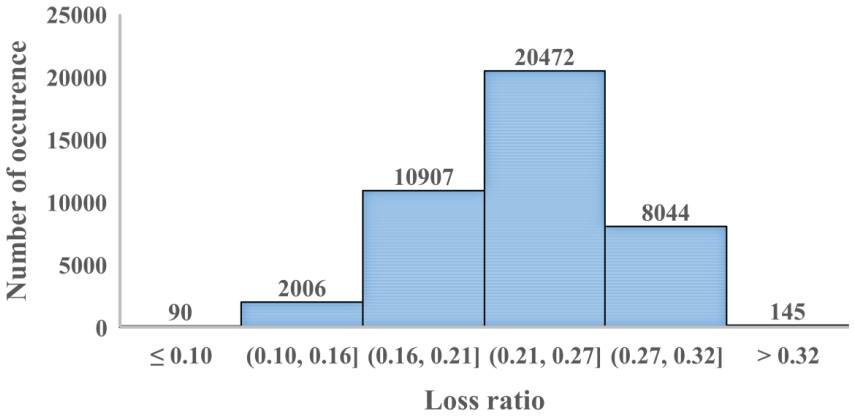
(b) Possibility distributions of two BHRs for minimum loss rate attack



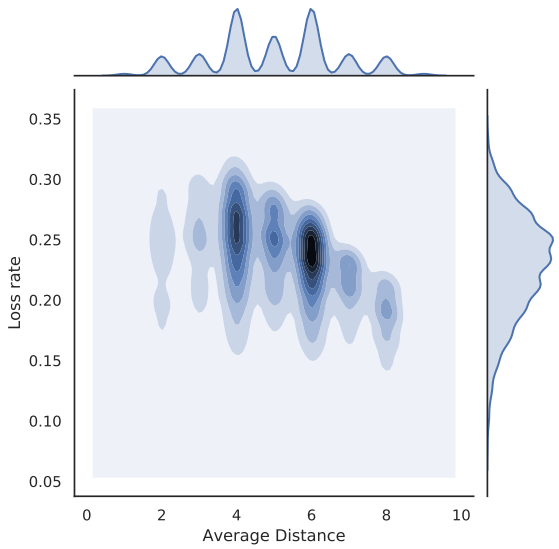
(c) Possibility distributions of two BHRs for maximum loss rate attack

Figure 4.5: An 8×8 NoC under two BHRs attack [101].

In the third experiment, the effect of the average distance between the BHRs was analyzed. In this demonstration, the NoC was infected with three BHRs. The packet dropping rate and the average distance between the BHRs were captured. The average distance between three BHRs is measured as the average number of hops between each pair of them. Figure 4.6a shows the frequency of the packet dropping rate periods for the attacked NoC. The dropped packet rate varies from 7.4% up to 33.8% for all the combinations of the distribution of the three BHRs in the NoC. Half of the possible combinations of BHRs



(a) Frequency of the packet loss rate.



(b) Density of the packet loss rate with the average distance between the BHRs.

Figure 4.6: An 8×8 NoC under three BHRs attack [101].

distribution have 21% to 27% of the packet loss rate. Figure 4.6b demonstrates the density of the packet dropping rate, along with the average distances between each possible three BHRs in the NoC. The graph reveals that a higher density of packet loss rate is located at 21% to 27% when the average distance between the BHRs is medium (for example, the average distance = 6).

4.4 Summary

Thus far, this chapter presented a security threat model that attacks the NoC through a HT that deliberately discards packets from the network, known as a Black Hole Router (BHR). The malicious node has a very small area and power overhead, 1.98% and 0.74% respectively, which makes it very hard to be detected during a post-silicon test. Furthermore, this chapter has shown the potency of the BHR attack on the NoC. It has demonstrated the effect of the BHR in a NoC in terms of the location of the infected nodes, their number, and the average distance between them. Results have shown that the packet dropping rate varies between 5% to 33.8% based on the number of BHRs and their locations in the NoC. Therefore, it is necessary to counteract the BHR attacks in runtime. The countermeasure technique should detect such malicious nodes, and once they are detected, the routing algorithm should be reconfigured to detour around them so as to avoid their effect.

CHAPTER 5

DETECTION AND PREVENTION PROTOCOL FOR THE BLACK HOLE ROUTER ATTACK

This chapter presents a secure interconnection network against the Black Hole Router attack and provides several simulation experiments to study the network performance. The proposed technique not only detects and locates the BHR but also isolates it from the network routing. A secure interconnection platform for MPSoC that guarantees packets delivery is achieved. The designed model is extended to reduce the energy consumption during the BHR detection process.

5.1 Threat Mitigation

Chapter 4 presented the BHR threat model and analyzed its attack on the NoC. Even with the current detection and prevention methods of HT described in Chapter 2, having an effective defense technique in place is required to prevent such attacks at runtime. A two-stage solution is designed and described for the detection and prevention of a Black Hole Router at runtime as follows:

1. Detect the DoS attack caused by the Black Hole Router.

2. Apply a secure routing scheme to detour around such a malicious node.

5.1.1 Black Hole Router Detection

In order to confirm that a packet is not dropped and is moving forward from one hop to another in the NoC, a switch-to-switch (s2s)-based acknowledgement (ACK) is required. Therefore, a single notification, ACK, is sent back to the one hop away (penultimate) router confirming that the packet is moving forward towards the destination.

Figure 5.1 shows a process of forwarding a packet from the source (S) to the destination (D) including the ACKs sent back to the routers (shown as dashed lines). For example, a packet is sent from router (R2) to router (R3). R3 forwards it to R4. R4 should then send an ACK to R2 that the intermediate router (R3) had forwarded the message successfully. In case R3 does not forward the packet, R2 will never receive an ACK that the packet has been forwarded to the next hop. In case R2 does not receive the ACK within a certain time frame, a flag is raised indicating a potential malicious behaviour at R3. The time that R2 should wait for a response from R3 has a threshold value that depends on some factors such as the processing time to prepare for an ACK, NoC traffic pattern and NoC congestion, as evaluated in Section 5.3.2.

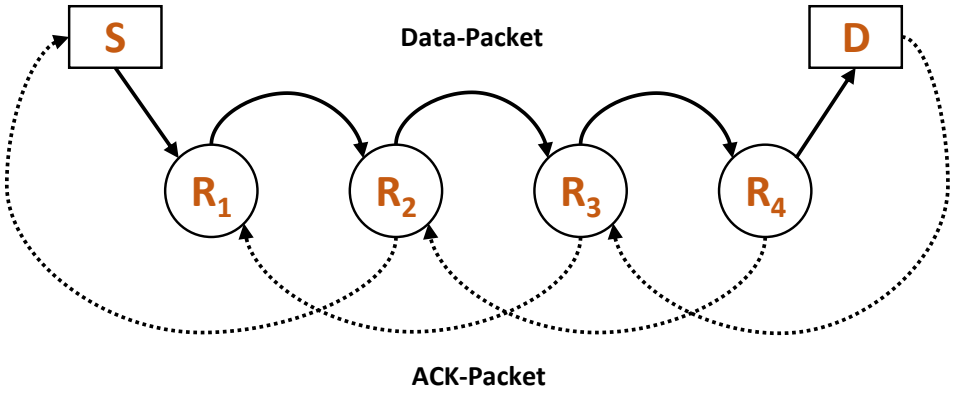


Figure 5.1: A process of forwarding a packet from the source (S) to the destination (D).

Figure 5.2 shows a time sequence diagram for diagnosing the packet-dropping malicious node. In this figure, a packet is being forwarded from a "Penultimate" router,R_P, to the "Current" router, R_C, through an "Intermediate" router, R_I. The detection technique sequence is described as follows: 1) R_P sends a packet to R_I, 2) R_I forwards the packet to the following node and at the same time it sends an ACK to the proper node indicating that the packet has been successfully received, 3) R_C node forwards the packet to the intended destination and at the same time it sends an ACK to the R_P node that the packet has been received. In step (4), an intermediate node is in charge of forwarding the ACK from R_C to R_P, where these two nodes are not directly connected.

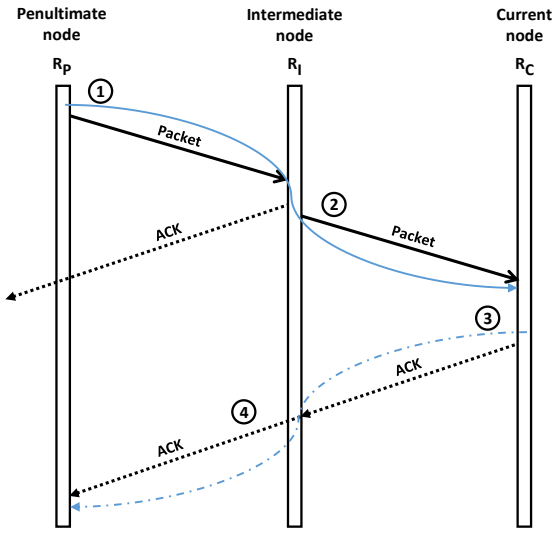


Figure 5.2: Time sequence diagram of the malicious node detection technique.

However, when a malicious node exists in the NoC, some certain security techniques must be applied, since the ACK message of the two penultimate nodes (R_P and R_C) may pass through a malicious node which may forge the ACK, and it would incorrectly sound to R_P that the packet has been forwarded successfully. Therefore, a secure acknowledgment is a must (detailed in Section 5.2). One way to keep its confidentiality is to use a shared

key between R_P and R_C. Then, the secure acknowledgement (Secure-ACK) is signed by R_C using the shared key along with the received packet. In this case, R_I cannot forge the signature unless it has access to the shared key.

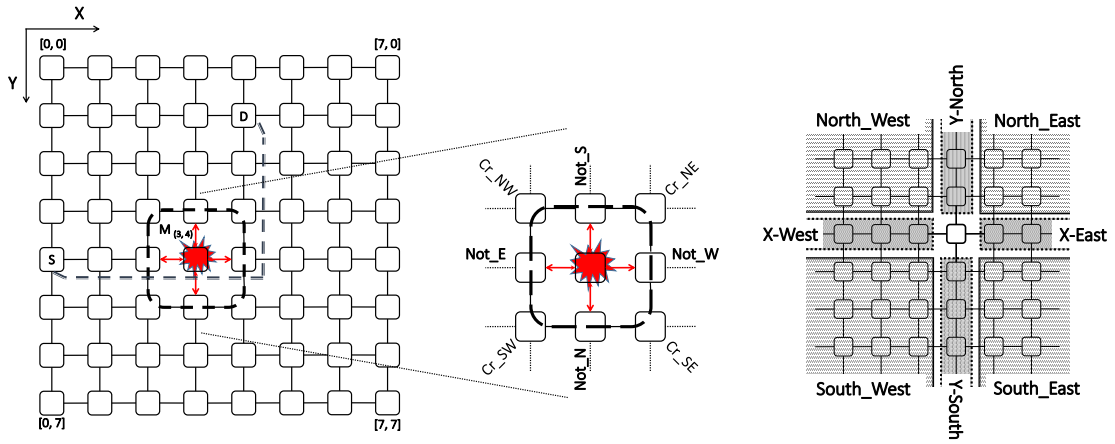
In addition, the Secure-ACK should be prepared in a trusted and secure unit. Therefore, the authenticated ACK is performed in the SoC firmware at the interface of the processing core with the NoC, which is assumed to be designed in-house by a trusted team.

5.1.2 Detouring A Black Hole Router

After detecting a Black Hole Router, the operating system (OS) is informed of an existing BHR along with its location in the NoC in a secure process - this is out of the scope of this research - for further operations such as task migration and future processor allocation.

The surrounding nodes of the Black Hole Router are updated, as shown in Figure 5.3a, creating a shield ring around it, and the NoC is divided into virtual segments. These surrounding nodes alter the packets' routing to avoid the malicious node. Alteration of a packet routing depends on the direction it comes from, the type of the current node, and the destination. The nodes in the NoC have several types. All the nodes of the NoC are of a "NORMAL" type, however, for routing purposes to avoid the infected router, the nodes around a BHR have different types: Not_S, Cr_NE, Not_W, Cr_SE, Not_N, Cr_SW, No_E, and Cr_NW.

There are several routing techniques to detour around a specific node. These techniques are inherited from fault-tolerant routing, such as [62, 145–147]. In order to avoid deadlock, the malicious-tolerant routing algorithm breaks one of the possible cycles of the turn model [62]. The routing algorithm is reconfigured to avoid the BHR based on a secure and routing-aware algorithm [141] that prohibits the East-South and North-West turns, as shown in Figure 5.4, to provide deadlock-free platform.



(a) A shield ring around a Black Hole Router **(b)** Eight segments around a node

Figure 5.3: An 8×8 Mesh NoC with a Black Hole Router.

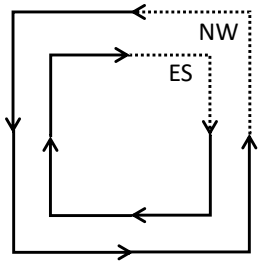


Figure 5.4: The prohibited East-South (ES) and North-West (NW) turns.

Table 5.1 shows the output direction of the secure routing algorithm. The output from the routing module is the direction of the packet, which depends on the type of the node and its location along with the destination region type.

The inputs to the routing module are: 1) Node_Type (Figure 5.3a), 2) Destination Region (Figure 5.3b) and its destination (X_dst, Y_dst), 3) Current Location: (X_local, Y_local), and 4) NoC dimension dimX×dimY. Based on the input to the malicious routing module , the output is one of the directions (North_Direction, East_Direction, South_Direction, West_Direction, or Local_Direction). Appendix A demonstrates a graphical example of a packet detours around a malicious node in 5×5.

Table 5.1: Malicious tolerant routing based on the Node_Type and the packet destination.

<table>
<tr><th></th><th>Y-North</th><th>North_East</th><th>X-East</th><th>South_East</th><th>Y-South</th><th>South_West</th><th>X-West</th><th>North_West</th></tr>
<tr><td>NORMAL</td><td rowspan="8">return North_Direction;</td><td colspan="3" rowspan="3">return East_Direction;</td><td rowspan="8">return South_Direction;</td><td colspan="3">return West_Direction;</td></tr>
<tr><td>Cr_NE</td><td colspan="3">if (X_dst <X_local -1
OR Y_dst <= Y_local)
{ return West_Direction; }
else { return South_direction;}</td></tr>
<tr><td>Not_W</td><td colspan="3">if (Y_local == dimY-1
OR (X_local == 1 && Y_dst <Y_local))
{ return North_Direction; }
else { return South_direction; }</td></tr>
<tr><td>Cr_SE</td><td colspan="3" rowspan="2">if (Y_local == dimY-2 OR Y_dst >= Y_local
OR X_dst >X_local + 2)
{ return East_Direction; }
else{ return South_Direction; }</td><td colspan="3">if (X_local == 1 && Y_dst <Y_local - 1)
{ return North_Direction; }
else{ return West_Direction;}</td></tr>
<tr><td>Cr_NW</td><td colspan="3">if (Y_local == dimy-1
OR (X_local == 1 && Y_dst <Y_local))
{ return North_Direction; }
else { return South_direction; }</td></tr>
<tr><td>Cr_SW</td><td colspan="3">if (Y_dst >= Y_local
OR X_dst >X_local + 1)
{ return East_Direction; }
else { return North_Direction; }</td><td colspan="3" rowspan="3">return West_Direction;</td></tr>
<tr><td>Not_E</td><td colspan="3">if (Y_local == dimY-1
OR Y_dst <Y_local)
{ return North_Direction; }
else{ return South_Direction; }</td></tr>
<tr><td>Not_N</td><td>if (X_local != 0)
{ return West_Direction; }
else{ return East_Direction;}</td><td colspan="3">return East_Direction;</td></tr>
<tr><td>Not_S</td><td>return North_Direction;</td><td colspan="3">if (Y_local == dimY-2
OR X_local == 0 OR Y_dst <= Y_local
OR X_dst >X_local + 1)
{ return East_Direction; }
else { return West_Direction; }</td><td>if (X_local != 0)
{ return West_Direction; }
else { return East_Direction; }</td></tr>
</table>

5.2 Secure NoC Model Against A Black Hole Router

As was pointed out earlier in this chapter in order to detect a BHR, each node that receives a data packet sends a Secure-ACK to the penultimate node to assure that the packet is moving forward to the destination. The Secure-ACK is calculated in a trusted module attached to the router. Figure 5.5 shows the packet format for the ACK. The packet carries the calculated ACK at the current node and necessary routing information and is described as follows:

- Source ID (src ID): to identify the source of the ACK packet.
- Penultimate ID (pen ID): to identify the penultimate node to route the ACK to it.
- Packet type (type): to identify the type of the packet, whether it is data or ACK.

- Packet ID (pkt ID): to identify the packet ID to check the correctness of the ACK.

- ACK-Payload: to carry the calculated ACK at the current node to be compared at the penultimate node.

The ACK-packet is injected at the current node towards the penultimate node. Once it is routed to the proper node, the ACK-Payload is extracted and compared to the calculated ACK at the penultimate node.

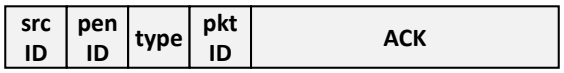


Figure 5.5: ACK packet format.

Figure 5.6 shows the architectural details of a secure Mesh-NoC model. Each node in the NoC is connected to a Secure Signature Module (SSM) that is responsible for generating a signed ACK. On one hand, routers are only in charge of routing packets from a node to another. On the other hand, the SSM is a secure module that has access to the paired keys and computes the Secure-ACKs to detect a BHR. It is implemented in the network interface by a trusted party or in-house. None of the routers in the NoC has access to the keys in the NoC. When an attack is detected, a secure signal is sent to a Secure Detection Management Module (SDMM) through a secure link to check the attack and localize it for further actions, such as isolating the infected node and avoiding mapping tasks to the attached processing element.

In order to design the presented secure NoC model, the baseline router has been modified and a security engine is attached to it to assure data delivery and to countermeasure the Black Hole Router attack. The presented secure router (also called *"Secure Signature Router"* (SSR)) architecture is composed of input buffers, a crossbar switch, and routing

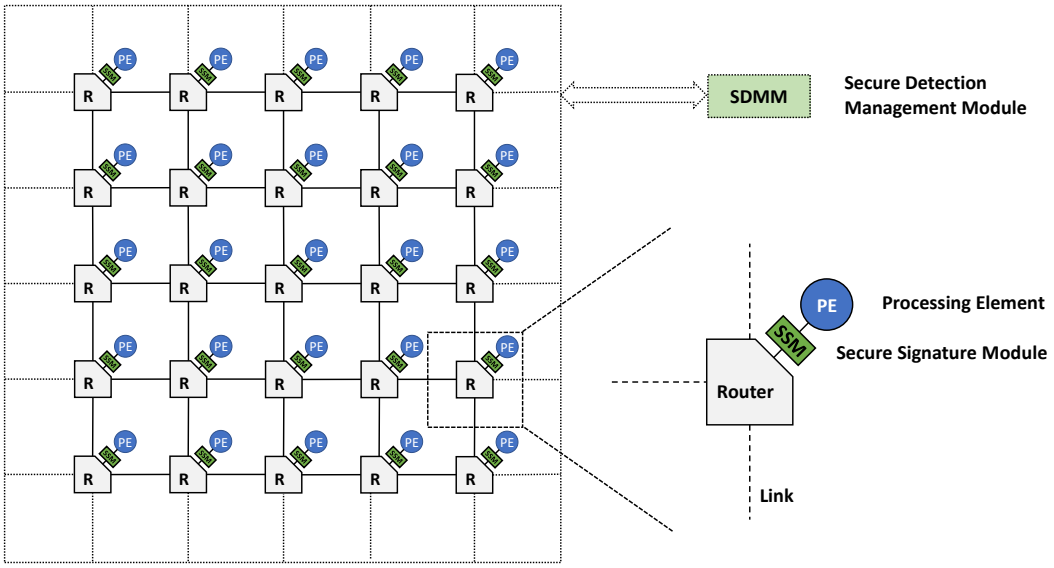


Figure 5.6: Structural overview of the proposed NoC model.

logic. The secure signature module (SSM) is attached to the router, which is assumed to be designed in-house. The SSM signs the received packets and generates the Secure-ACKs.

On one hand, the Data-Routing module is a normal router which contains data routing control unit and input data buffers. On the other hand, the SSM has two main cores: *Signature* and *Confirmation* cores. In order to decouple the data routing from the signed-packets routing, two separate virtual communication channels have been created at each port of the router.

Figure 5.7 shows the architecture of the Secure Signature Router. The received data-packets at a router's port are saved in the input data-buffer. These packets are signed by the SSM using a shared key between the associated routers, then the signed-packet is routed back to the corresponding router for acknowledgment and confirmation. When a signed-packet is received at the appropriate router, a confirmation processes is performed to check if the packet has been successfully forwarded. If the current router is not the destination of the signed-packet, it will route it to the appropriate adjacent one. In this model, the

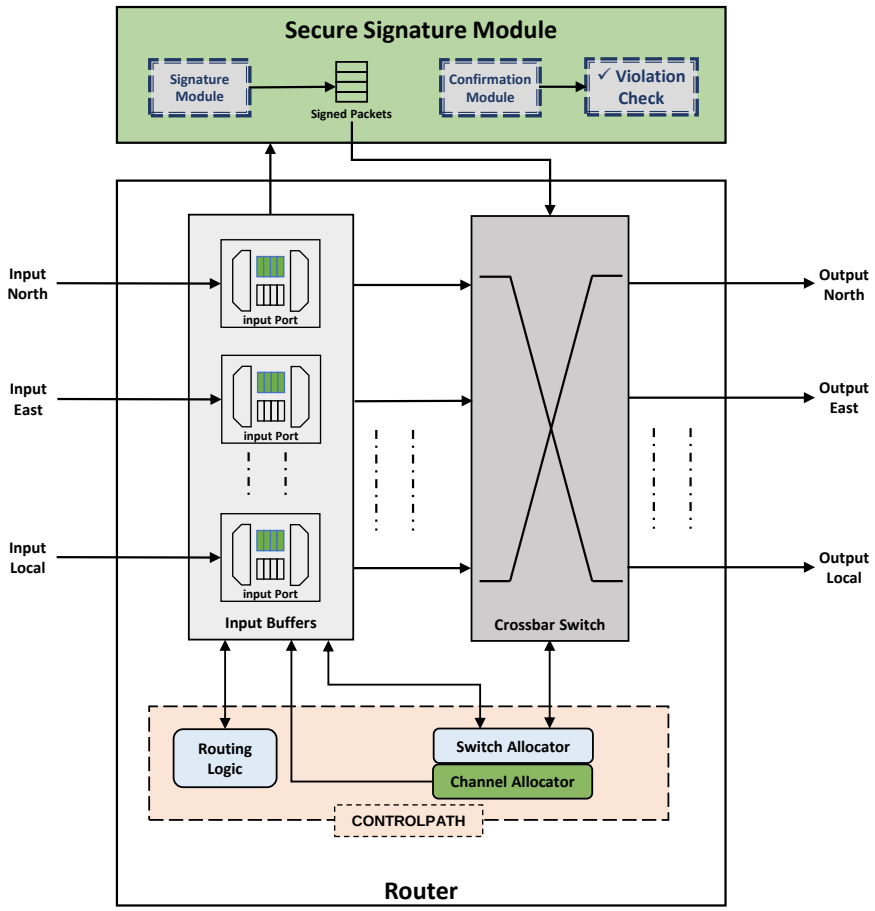


Figure 5.7: Secure Signature Router (SSR) architecture.

data-packet and ACK-packet share the same physical link and their routing is managed by the channel allocator giving higher priority to the ACK-packets.

Figure 5.8 shows the flow control of a received packet inside the router. In this diagram, the type of the packet is checked whether it is data or ACK. In case of a data-packet, it will be pushed into the input data buffer. In the meantime, a secure ACK needs to be calculated at the "Secure ACK Computing Unit" in the SSM module and sent to the penultimate node. In case of receiving an ACK packet, it will be checked by the router for its destination. If the current router is the target destination, it will be checked for its

validity at the "Received ACK Validating Unit", otherwise it will be routed towards its destination. The BHR detection unit checks if an ACK is received within a predefined time frame. If not, a packet-dropping attack might have occurred and the SSM raises a flag to the SDMM to take an action and isolate the BHR.

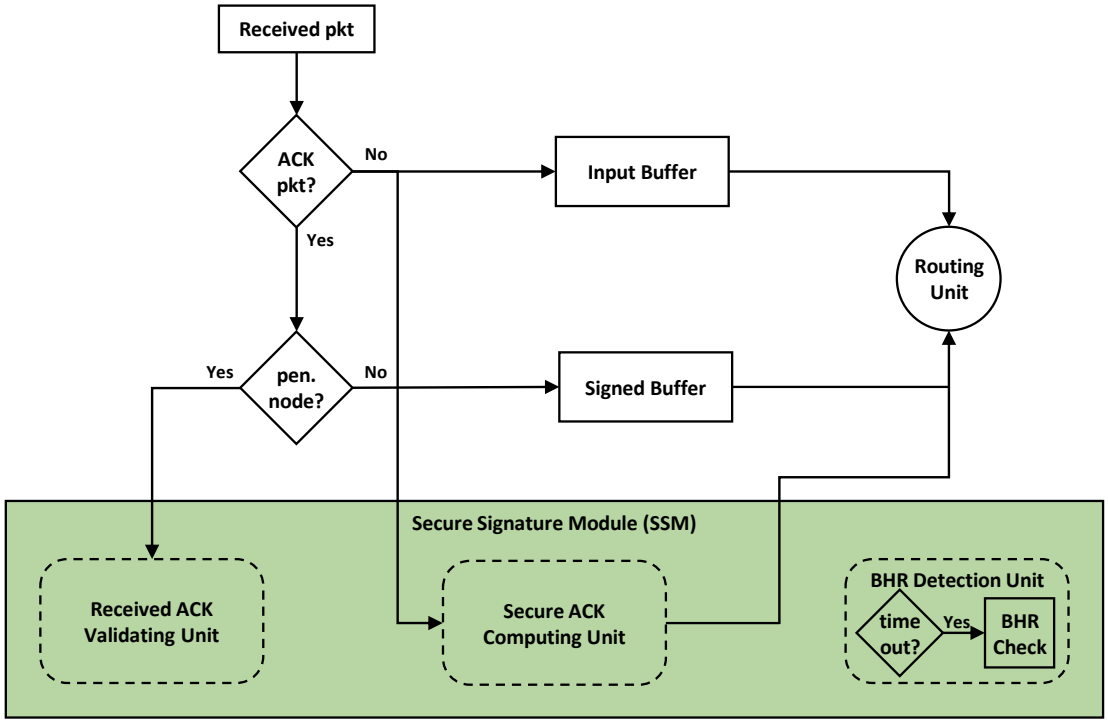


Figure 5.8: Flow control diagram of handling a received packet in the secure router.

Figure 5.9 presents an example of detecting a BHR. In this example, a processing element, P_s, sends a data-packet, *pkt*, to a destination node, P_D. *pkt* is routed successfully through routers R_s, R_1, R_2. At the signature module, *SSM*, attached to R_1, a secure acknowledgement, ACK_1 , is calculated and sent back to the secure module attached to P_s to be validated and clarifies that R_s is HT-free. Similarly, when the *pkt* is forwarded to R_2, ACK_2 is calculated and sent back to *SSM* of R_s to confirm that R_1 is HT-free as well. On the other hand, *pkt* is forwarded to the adjacent router towards the destination, which

is assumed to be infected with a Black Hole Trojan, *BHR*, that drops the packet and does not send back any acknowledgement. Both *SSM* attached to R_1 and R_2 are waiting for an ACK which in fact was not generated. At that moment, the waiting time expires and a Black Hole attack has been detected. Hence, these nodes send control packets to the Secure Detection Management Module (shown in Figure 5.6) indicating the data-packet ID to define the BHR. In order for increasing the arrival chance of these special packets, they are sent (flooded) horizontally (East and West directions) and vertically (North and South directions) to the SDMM. The received packets at the SDMM are analyzed to locate the BHR and then it takes an action to isolate the malicious router from the NoC routing.

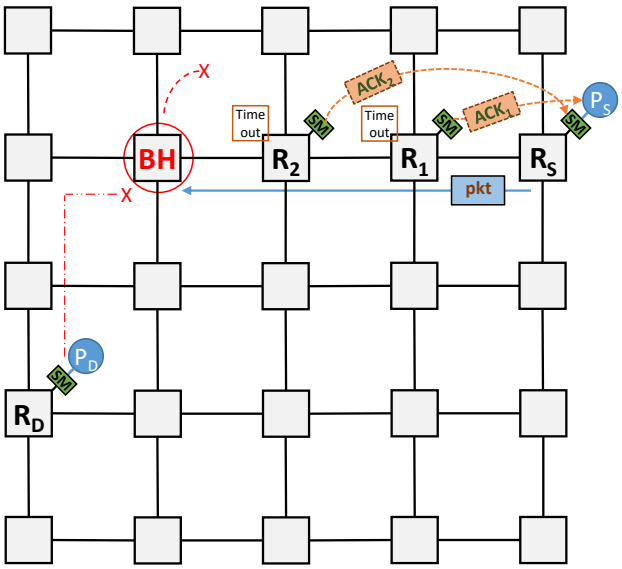


Figure 5.9: An example of the BHR detection technique in the SSR-based NoC model.

Signature Engine

In order to increase the level of security, a Message Authentication Code (MAC) may be used for improving the security of the ACK. A Hash-based Message Authentication Code

(HMAC) [148] is a specific type of MAC that involves a cryptographic hash function, such as SHA-3 [149], and a secret cryptographic key. The HMAC has three rounds of hashing process that takes a longer time to be performed, as proven in [150] that the SHA-3 takes about 70 clock cycles to hash one block, and the HMAC [151] takes 280 cycles to perform one signature block. The HMAC is a power and time consuming technique which may impact the whole system performance, although it offers a high-level of security.

In order to authenticate the ACK packet to detect and countermeasure the BHR, symmetric cryptography such as Advanced Encryption Standard (AES) [152] is a suitable candidate to sign the ACK in the signature module because it is a lightweight encryption technique and provides higher performance compared to the HMAC. The AES is a faster and secure symmetric encryption compared to other encryption standards, such as Data Encryption Standard (DES) [153], and asymmetric key encryption, RSA [154]. RSA is a cartographic technique that provides high security level at the expense of the performance of the system.

However, a light-weight authentication technique is recommended for embedded systems. Therefore, a secure ACK is calculated by bitwise-XOR, the received packet with a one-time pad number shared with the current and penultimate nodes. One way to generate one-time pad random numbers which are shared between two nodes is by designing a Pseudo Random Number Generator (PRNG) and the shared key is the seed of the PRNG circuit. In this work, a PRNG was used for generating dynamic random number to authenticate the ACK-packet. For a higher security level, AES was adopted in the Secure Signature Router to sign the ACK-packet and used for comparison with the light-weight security.

Key Management Scalability

Key management between the nodes is a crucial task to maintain the trust of the NoC operations. The process of key exchange takes place when the system boots up. The secure boot is based on the Public Key Infrastructure (PKI) process to authenticate the NoC and share the keys between the nodes in the NoC. Keys between routers can be safely distributed using the Diffie-Hellman protocol [155], [156]. In the NoC, [157] is used to decentralized the key exchange between the nodes. Key exchanging is out of the scope in this research.

In the secure router architecture, a node needs to send a secure ACK to the penultimate nodes. Figure 5.10 shows the keys between a node and its penultimate nodes. A node (R) communicates with the penultimate nodes via the shared keys: K_1, K_2, K_4, K_5, K_8, K_9, K_{11}, and K_{12} and with the neighbored processing elements through keys: K_3, K_6, K_7, and K_{10}. This indicates that each node in the NoC requires 12, 8, and 5 distinct keys for the middle, side, and corner nodes, respectively.

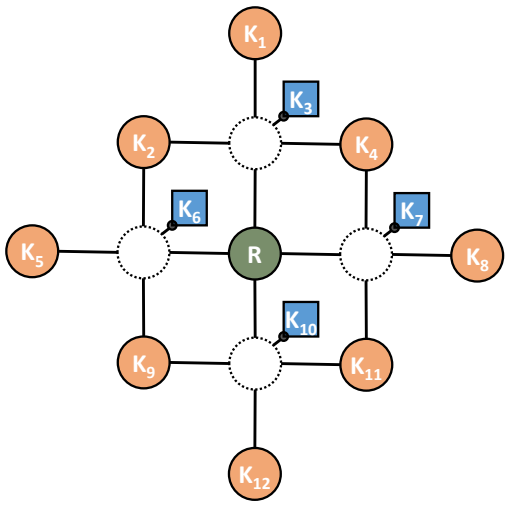


Figure 5.10: Keys shared between a node (R) and its neighboured PEs and the penultimate nodes.

The required total number of keys in the system is proportional to the NoC size. Appendix B studies the number of keys required for the NoC to achieve a secure architecture. The total number of distinguished keys for a $N \times N$ NoC is represented by the following equation:

$$6N^2 - 15N + 2 \qquad \forall \quad N \geq 5$$

Figure 5.11 shows the scalability of seed-keys in the system with the NoC size. Notice that the total number of distinguished keys linearly increases with the NoC size. Each secure network interface of the middle nodes in the NoC carries a maximum of twelve different keys while each node at the corner of the NoC has only five keys, regardless of its size.

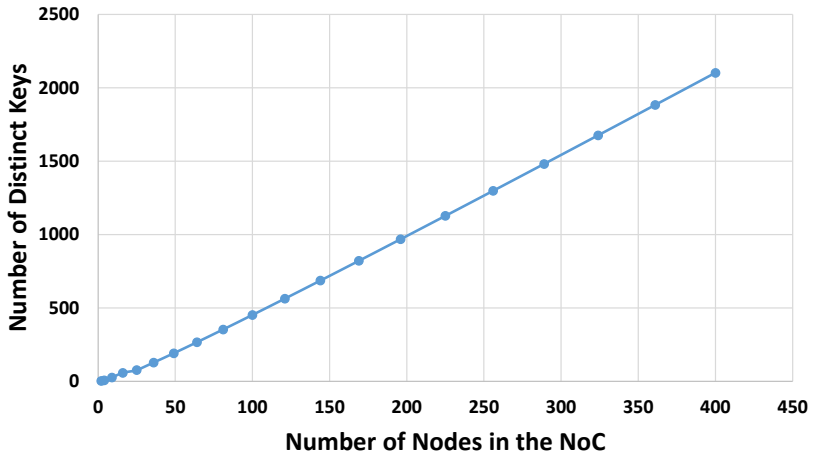


Figure 5.11: Scalability of the distinct seed-keys with the NoC size.

A cryptographic hash function such as SHA-3 [149] is used for securely generating the seed-key for each pair of nodes. The overhead for generating such a key depends on the throughput of the hash function [150]. As explained earlier, the key exchange happens when the system boots up.

Area Overhead

In order to evaluate the area overhead of the secure signature router against the BHR attack, a moderate size, five-ports router, with a 4 packets depth FIFO, each of 4 Flits (32 bits), and a "Store-and-Forward" (S&F) switching technique was designed and synthesized using 45nm TSMC technology (using Cadence Design Compiler). It occupied an area of 73,215 μm^2. The baseline router was modified and the security features against the BHR attack were added. The area overhead was increased by 26.9% of the baseline router.

5.3 Evaluation and Experimental Results

In order to study the NoC behavior and performance overhead of the secure signature router architecture, a cycle-accurate simulation model of a Mesh NoC architecture was developed in SystemC and integrated with the Noxim simulator [144]. The baseline router was designed to have five ports: North, South, East, West, and Local. Each port has an input buffer and full duplex data communication. The Black Hole threat model was designed and inserted into the receiving-unit of the input ports of the victim router with a time-bomb activating circuit. The secure router architecture was developed in SystemC and integrated with the simulation environment for evaluation. The simulation setup was established for two different NoC models:

- A NoC that contained only the baseline routers to be used for collating purposes.
- A NoC that contained the secure router with the same baseline NoC parameters.

The same experiments were run on both setups for fair comparison. Figure 5.12 shows the block diagram of the simulation structure. The path on the left-hand side represents the baseline NoC design with parameterized size, routing technique, buffer depth, ... etc. The right-hand path is for the infected NoC with a HT model, including the secure engine model for BHR detection. The NoC performance for both models were evaluated under various packet-injection rates and several traffic distributions. The configuration of the NoC was manipulated by a Python script to automate the evaluation process and analyze the output results.

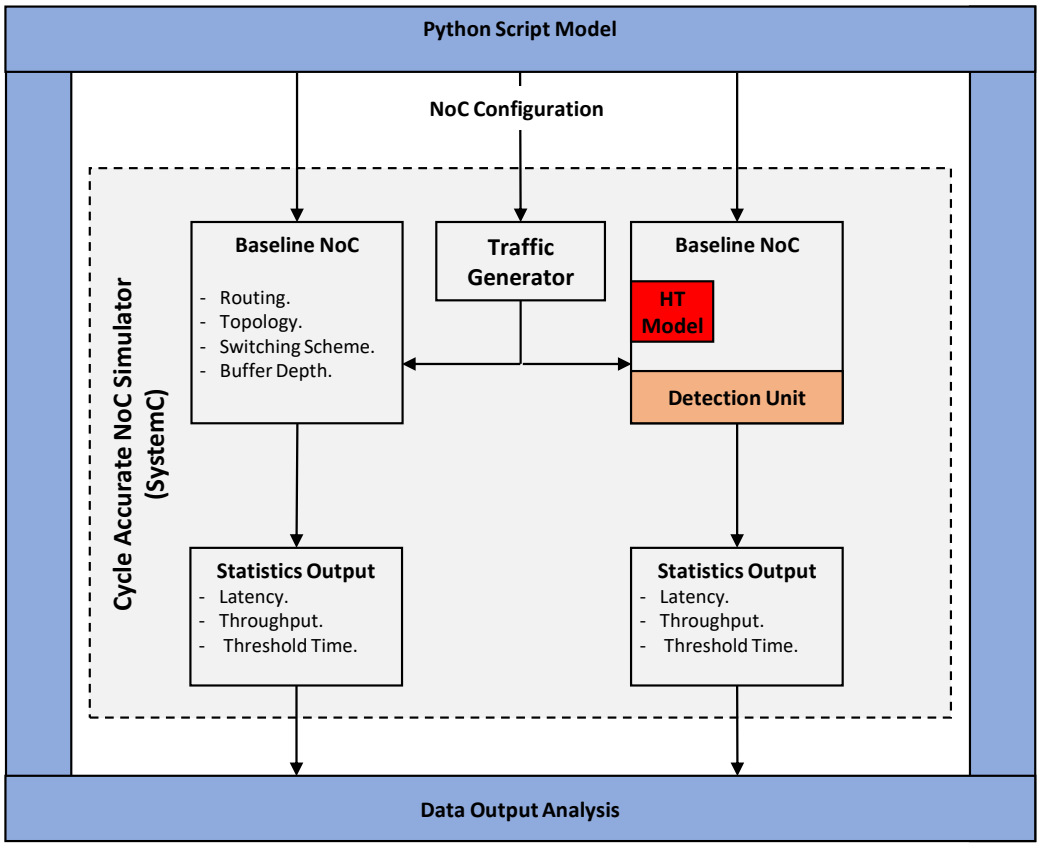


Figure 5.12: A block diagram of the simulation structure.

5.3.1 Experimental Setup

To evaluate the router structure presented in Section 5.2, the NoC parameters and the simulation environment were configured to cover different NoC sizes and buffer depth sizes for various injection rates and several synthetic traffic patterns [64]: Bitreversal, Butterfly, Random, Shuffle, and Transpose. Appendix C describes the traffic patterns of each type. Simulations were run for different buffer depths, NoC sizes, and different injection rates for 100,000 clock cycles. Their analysis results were very similar. Therefore, this section provides the analysis results for an 8×8 NoC size and a buffer depth of 4 packets. The results of different parameters on the NoC performance will be presented in case these parameters play a role in the network behavior. Table 5.2 depicts the configuration parameters of the simulation environment.

Table 5.2: NoC parameters for the experimental simulation.

NoC Size	8×8
Packet Size	4 Flits
Flit Width	32 bits
Buffer Depth	4 Packets
Switching Technique	S&F
Simulation Time	100,000 cc
Warm-up Time	200 cc

5.3.2 Experimental Results

The experimental setup explained in the previous section was used for evaluating the effect of the designed Secure Signature Router (SSR) on the NoC behavior. System throughput and overhead were analyzed for the different NoC parameters listed in table 5.2. The simulation objectives of the NoC model which adopts the signature router are summarized as follows:

1. Identifying the saturation point of the presented SSR-based NoC model and measuring its throughput.

2. Identifying the threshold waiting time for the secure ACK of the SSR router.

3. Measuring the degradation of the NoC performance in the presence of a BHR for the NoC model.

4. Measuring the system scalability.

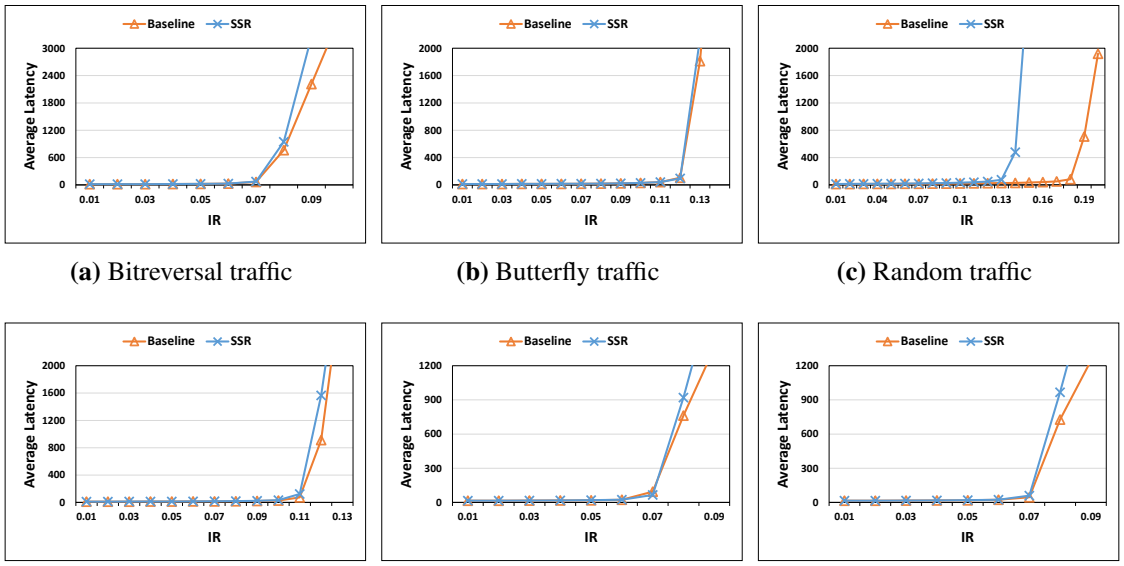
5. Measuring the NoC performance for different security-levels.

Throughput and saturation point

In order to evaluate the overhead of the Secure Signature Router (SSR) on the NoC performance and identify the saturation point of the NoC-based system, the modeled runtime detection technique of the BHR attack was integrated into the baseline NoC. In this experiment, the NoC was not under a BHR attack, and the detection technique is activated without any further action of false positive attacks.

Figure 5.13 shows the average latency (clock cycles) of both the baseline NoC and the SSR-based NoC models. For deterministic traffic distribution (Bitreversal, Butterfly, Shuffle, Transpose_1, and Transpose_2), the SSR-based NoC model almost saturates at a similar point. It saturates at 7%, 12%, 11%, 7%, and 7% for Bitreversal, Butterfly, Shuffle, Transpose_1, and Transpose_2 traffic patterns, respectively. Because of the deterministic traffic patterns, the saturation points of the NoC model with and without a security engine are similar, where the source and destination pairs are fixed. However, for a Random traffic pattern, destinations are randomly selected using a uniform distribution. This allowed different packets to be directed to different nodes from the same source node. Figure 5.13c

shows the average latency of SSR compared to the baseline model, where the SSR-based NoC model saturates at 13% while the baseline model saturates at 19%.

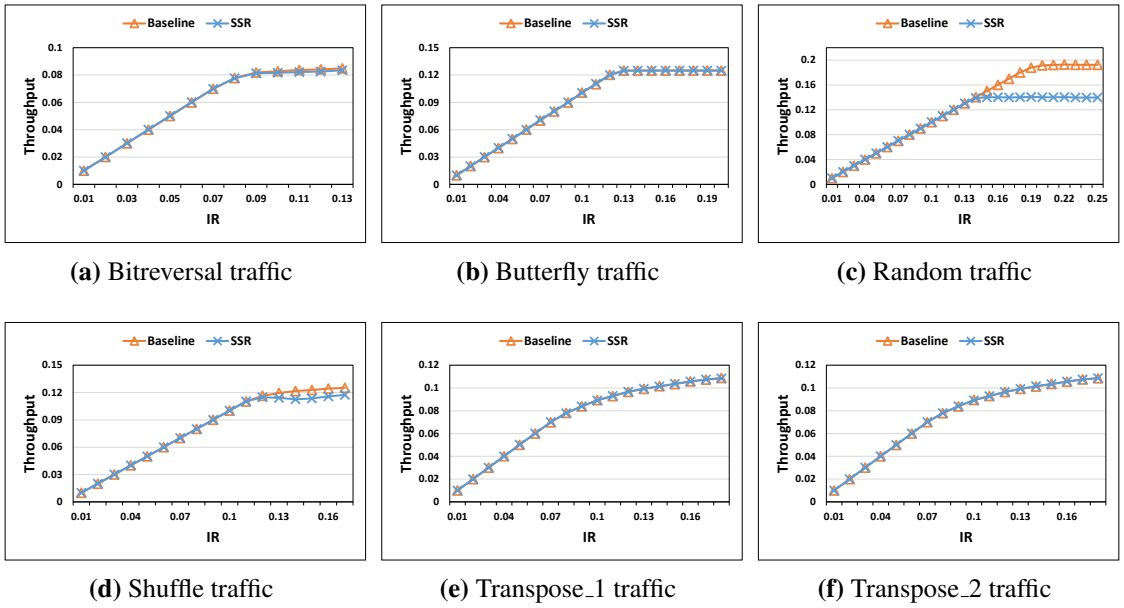


(a) Bitreversal traffic **(b)** Butterfly traffic **(c)** Random traffic

(d) Shuffle traffic **(e)** Transpose_1 traffic **(f)** Transpose_2 traffic

Figure 5.13: Average Latency of the Secure Signature NoC model under several traffic patterns, the injection rate (IR) is measured in Flit/Cycle/Node.

Figure 5.14 shows the throughput of the NoC model for different traffic patterns. Similarly, the SSR-based NoC model has slightly less performance than the baseline one for deterministic traffic patterns. It drops by 1%-2%. This is due to the nature of the distributed traffic where a fixed destination node is allocated to a selected source node. For Random traffic pattern as shown in Figure 5.14c, the SSR-based NoC model experiences 21.31% average performance overhead. It can be observed that the average throughput decreases, yet packet delivery is assured to the notion of graceful degradation of service.



(a) Bitreversal traffic **(b)** Butterfly traffic **(c)** Random traffic

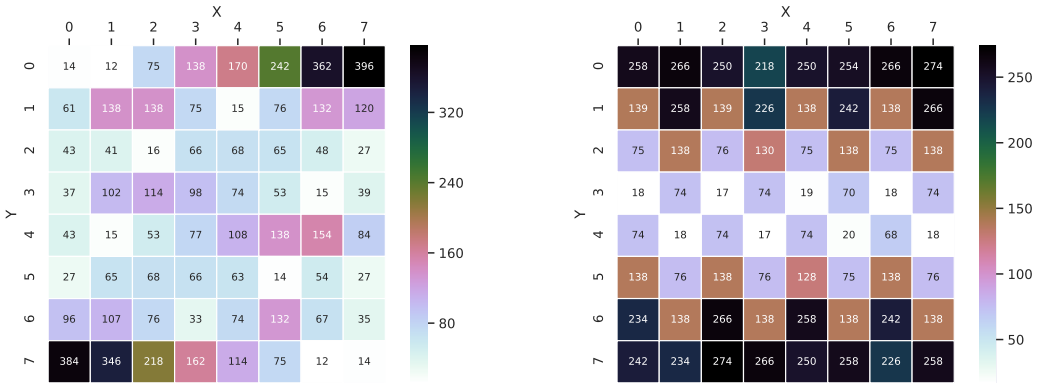
(d) Shuffle traffic **(e)** Transpose_1 traffic **(f)** Transpose_2 traffic

Figure 5.14: Average Throughput of the Secure Signature NoC model under several traffic patterns, the injection rate (IR) is measured in Flit/Cycle/Node.

Threshold time for Secure-ACK

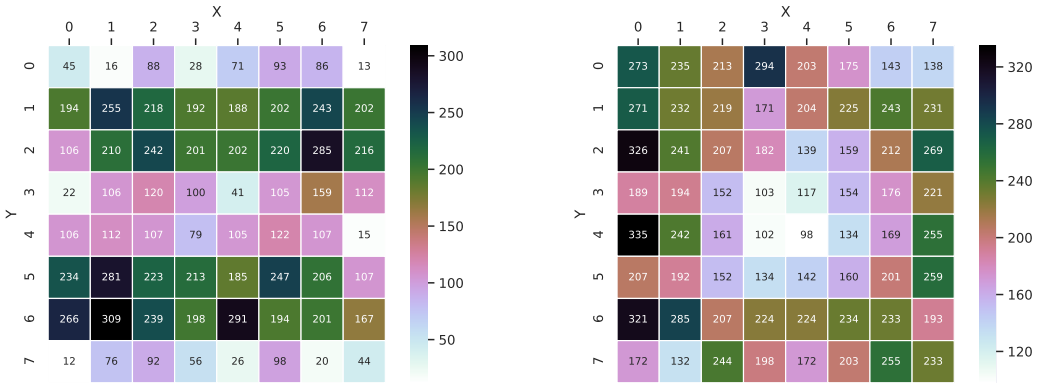
In this experiment, the waiting time for the penultimate router to receive a Secure-ACK from the proper node was studied. The time that a confirmation-packet traverses in two hops depends on several factors, such as injection rate, buffer size, traffic pattern, and the position of the router in the NoC. Therefore, the maximum time that a node waits for receiving a Secure-ACK was measured for each node in the NoC. The injection rate was fixed to the saturation point of the NoC model for different traffic distributions. Figures 5.15, 5.16, and 5.17 show the maximum waiting time in clock cycles at each router for each traffic pattern.

It is noticed that the network traffic plays a main role in determining the maximum waiting time at each router. Additionally, the maximum waiting time is not equally spatially distributed among routers. The reason behind this is that the routers at the mesh network



(a) Bitreversal traffic **(b)** Butterfly traffic

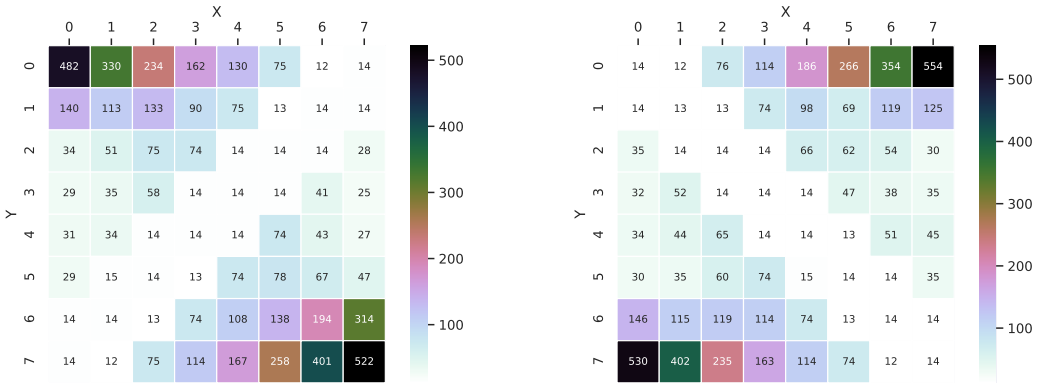
Figure 5.15: The maximum waiting time, in clock cycles, for the Secure-ACK at each node in the NoC for several traffic patterns .



(a) Shuffle traffic **(b)** Random

Figure 5.16: The maximum waiting time, in clock cycles, for the secure-ACK for several traffic patterns.

corners or borders have two or three ports, rather than four for the others, which limit the packet directions. Therefore the threshold waiting time is set differently at each node of the NoC. Choosing a short threshold time for receiving the ACK might lead to poor performance, since the router that has not received an ACK may claim a false attack, requesting the OS for further check. Similarly, choosing a long threshold time may lead to

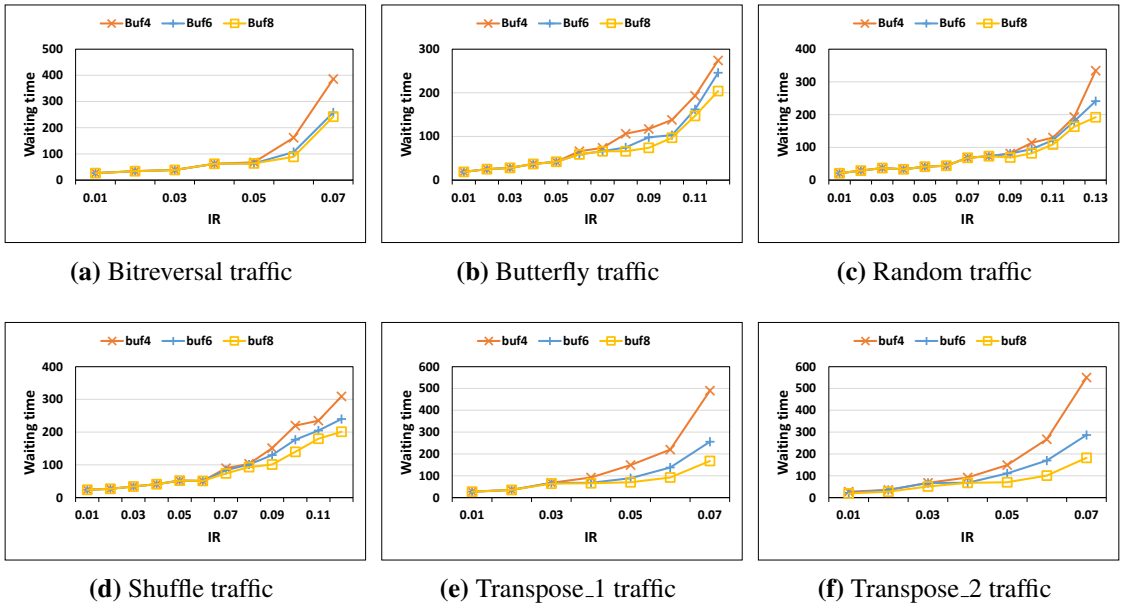


(a) Transpose_1 traffic **(b)** Transpose_2 traffic

Figure 5.17: The maximum waiting time, in clock cycles, for the secure-ACK for several traffic patterns.

poor performance as well, since the router waits for long time to receive the ACK while an attack might be in progress. One way to achieve better performance, a reconfigurable threshold time is needed. In the reset of the experiments, the threshold time was set to 1.25 $\times$ the calculated value at each router.

In order to check the effect of the buffer sizes on the waiting time for the confirmation packet delivery with respect to the injection rate, an experiment was structured to measure the maximum waiting time among the routers for different buffer sizes. The buffer size was set to 4, 6, and 8 packets at each input port for each experiment. Figure 5.18 shows the maximum waiting time for NoC of these buffer sizes. It is noticed that when the input buffer sizes increase, the waiting time for the confirmation ACK decreases because the NoC becomes less congested and packets move faster.



(a) Bitreversal traffic **(b)** Butterfly traffic **(c)** Random traffic

(d) Shuffle traffic **(e)** Transpose_1 traffic **(f)** Transpose_2 traffic

Figure 5.18: The maximum waiting time, in clock cycles, for the secure-ACK for different buffer depth under various traffic patterns.

Performance Degradation

In this experiment, the detouring scheme to avoid an infected node in the network was assessed. Two SSR-based NoC models were designed. One model was HT-free. The other model was infected with a BHR (packet-dropping attack). In both models, the throughput was recorded and evaluated. In this experiment, the infected model has one BHR at a time. The NoC performance was reported. The simulation was repeated for a different infected node and the outcome results were recorded. The simulation was run again to cover each node in the infected NoC. The measured throughput was averaged over the total number of the NoC nodes and compared to those of the HT-free model. The injection rate was fixed to the saturation point, 13% Flits/Node/Cycle. Figure 5.19 demonstrates the normalized NoC throughput of 8×8 NoC size for Random traffic pattern. The throughput is normalized to a Trojan-free NoC. As it is expected, when the NoC experiences HT nodes, the average

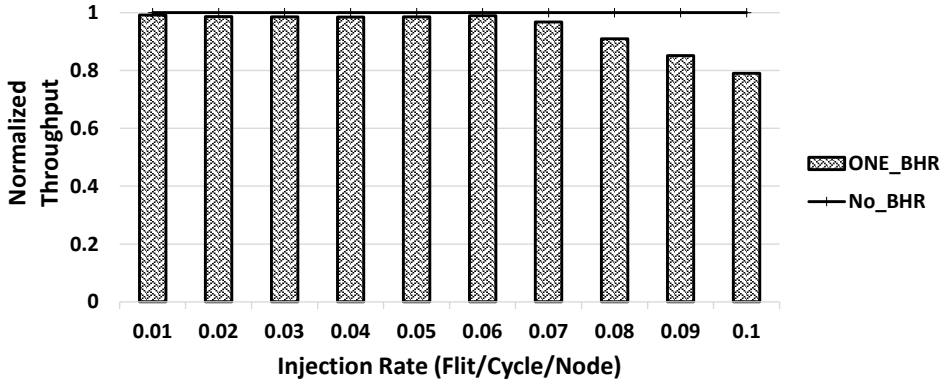


Figure 5.19: Normalized performance of the NoC model under a BHR attack.

throughput is slightly decreased with a low injection rate. The throughput decreases with increasing the injection rate. The performance overhead is due to packets detouring around the infected node experiencing a longer path to reach their destinations. Although the packets experience a longer route to avoid the BHR attack, which increases the packet latency, the NoC is more reliable (packet loss free) and assures packet-delivery to their destinations.

System Scalability

In this experiment, the scalability of the SSR-based NoC model was demonstrated with respect to the network size by measuring the average throughput of the NoC, with different sizes, for several injection rate.

Figure 5.20 shows the scalability of the secure router architecture for different injection rates, 1%, 5%, 10%, and 13%, for a Random traffic pattern. It is noticed that the SSR-based NoC scales efficiently with the NoC size for several injection rates.

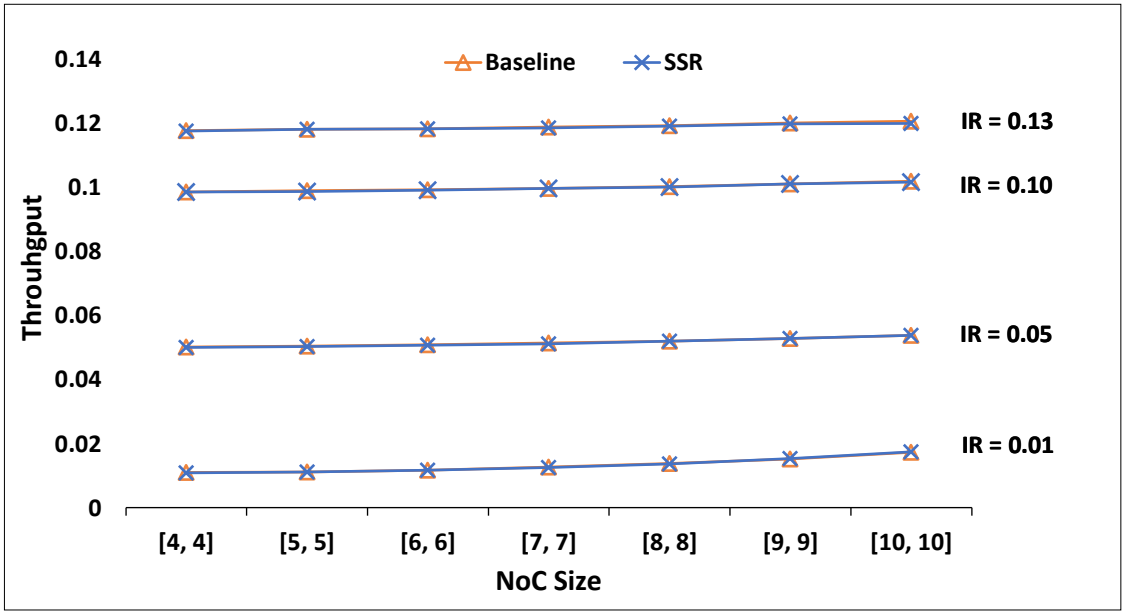
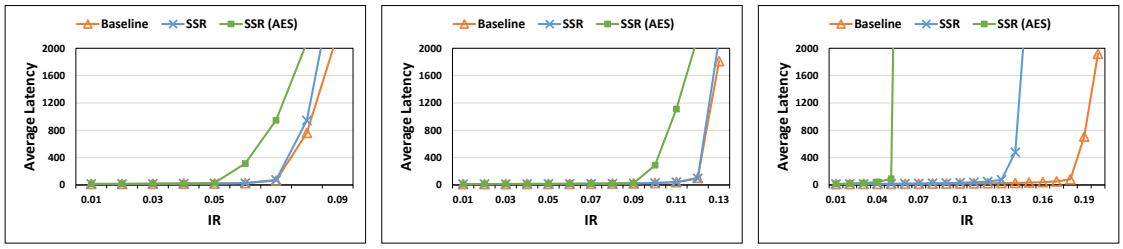


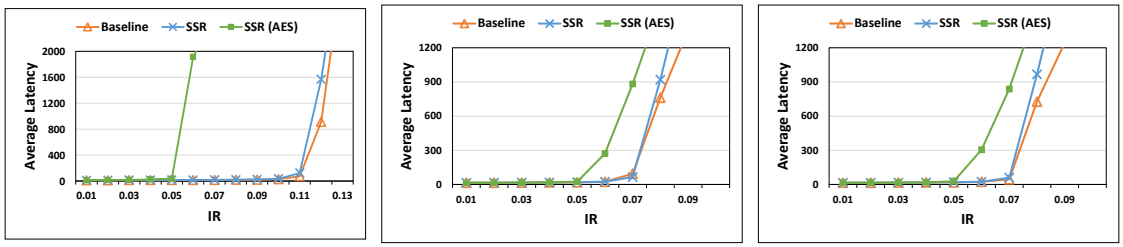
Figure 5.20: Scalability of the SSR-based NoC for different injection rates (IR), Flit/Node/Cycle.

NoC performance and security level

In the SSR-based NoC model, the signing process to generate a secure ACK is calculating bitwise-XOR of the received packet with a one-time pad number. This presents a low level of security. In order to increase the security level, Advanced Encryption Standard (AES)[152] may be used to sign the packet by the shared key at the expense of the processing time. In order to evaluate the performance overhead of the signature process using AES, in this experiment, the signature unit was integrated into the simulation environment, then measured its effect on the network performance without any malicious node. Two different designs for the secure signature router were evaluated. One with a one-time pad number (SSR) and the other with the AES (SSR (AES)). The purpose of this simulation was to measure the throughput and saturation points of the NoC model for the aforementioned techniques of packet signature.

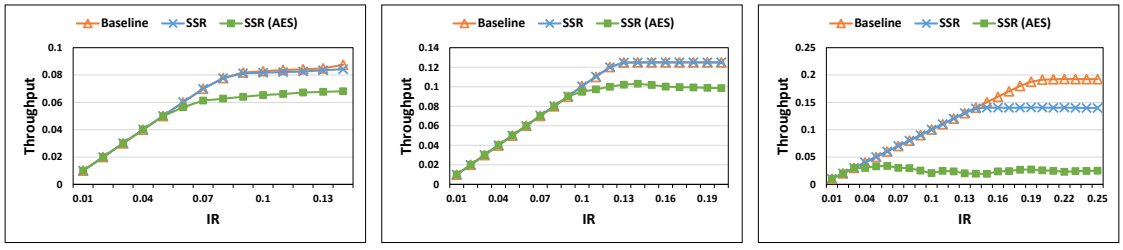


(a) Bitreversal traffic **(b)** Butterfly traffic **(c)** Random traffic

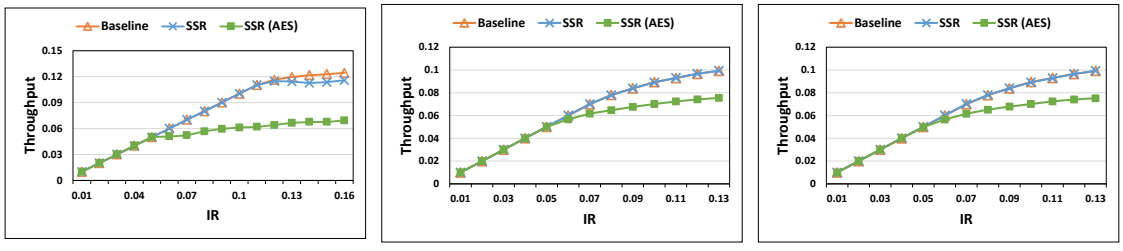


(d) Shuffle traffic **(e)** Transpose_1 traffic **(f)** Transpose_2 traffic

Figure 5.21: Average Latency of the NoC model based on different security levels under several traffic patterns, the injection rate (IR) is measured in Flit/Cycle/Node.



(a) Bitreversal traffic **(b)** Butterfly traffic **(c)** Random traffic



(d) Shuffle traffic **(e)** Transpose_1 traffic **(f)** Transpose_2 traffic

Figure 5.22: Average Throughput of the NoC model based on different security levels under several traffic patterns, the injection rate (IR) is measured in Flit/Cycle/Node.

Figure 5.21 shows the average latency for the two presented techniques, SSR and SSR (AES), compared with the baseline model. As expected, SSR (AES) saturates faster than SSR. This is due to the fact that AES takes a longer time to generate a secure-ACK. Additionally, the generated secure-ACK has the same size as the packet. This loads the network with a heavy number of packets in the system. Consequently, SSR (AES) presents lower performance compared to the SSR, as shown in Figure 5.22.

5.4 Energy-Efficient Detection Technique for the Black Hole Router

5.4.1 Refined Problem

The detection and protection schemes of the BHR attacks presented in the previous section were primarily based on a hop-to-hop (h2h) technique, where each router forwards a secure ACK back to the penultimate one indicating that the packet has been received and was successfully forwarded through the intermediate router. This leads to more power consumption due to moving the ACK between the nodes in the NoC.

In order to study the energy consumption of the h2h detection technique to detect BHR in NoC, several experiments were run for different NoC sizes and traffic distributions for the secure NoC model, including the signature modules to detect the BHR attack. Figure 5.23 shows the relative energy overhead of the h2h BHR detection unit to the baseline router in 8×8 Mesh NoC with uniform random traffic distribution for several injection rates. As can be seen, online h2h BHR detection incurs up to 24% of energy consumption overhead to the baseline NoC model. This indicates that a significant amount of energy is being consumed by the always active h2h detection technique.

If the HT is not active, as is often the case in most of the cases [91] or the system is HT-free, such an on-going h2h confirmation only causes an unnecessarily significant energy

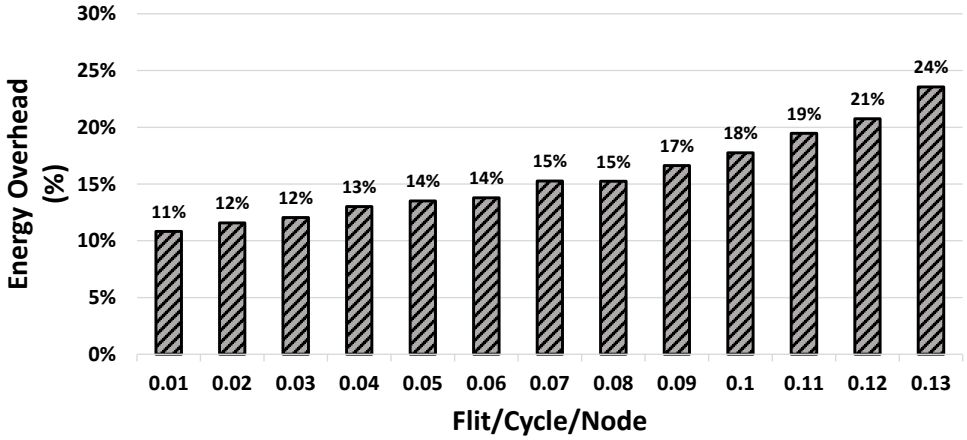


Figure 5.23: Energy consumption overhead of the SSR-based NoC model over the baseline NoC.

and performance overheads. Therefore, in this work, an end-to-end (e2e) power-gated technique that reduces the energy consumption along with BHR detection was designed. The presented e2e flow control does not only guarantee an attested interconnection, but is also able to localize the BHR online in the infected NoC.

5.4.2 Energy-Efficient BHR Detection Technique

In order for a source node to confirm a successful arrival of a message to the final destination, the final recipient node sends a secure-ACK to the source. This technique supplies a packet delivery assurance.

After packetizing the message at the network interface of the source node, it is injected into the NoC. Upon its arrival at the receiver side, the packet is processed by the PE and an ACK is generated and sent back to the source node. Packets usually consist of a number of flits and the acknowledgement can be considered as a single flit to assure a successful delivery of one packet. Once the sender receives a correct ACK, it assures that the packet has been received successfully at the recipient node. In contrary to h2h, the e2e protocol is

applied on a higher network layer, such as the transport layer, where the processing element may generate the ACK once it receives the message.

Figure 5.24 shows the sequence diagram of basic e2e packet delivery assurance protocol. In this model, the message is partitioned into multiple packets, P(1), P(2), .., P(n), before it gets injected into the network. At the destination, packets are received into a buffer in the network interface, then are grouped together to reformat the determined message. In order to confirm message arrival at the recipient node, an ACK-packet is created for each received packet, ACK(1), ACK(2), .., ACK(n). For an optimized model, only one ACK-packet may be sent to represent a successful arrival of the whole message.

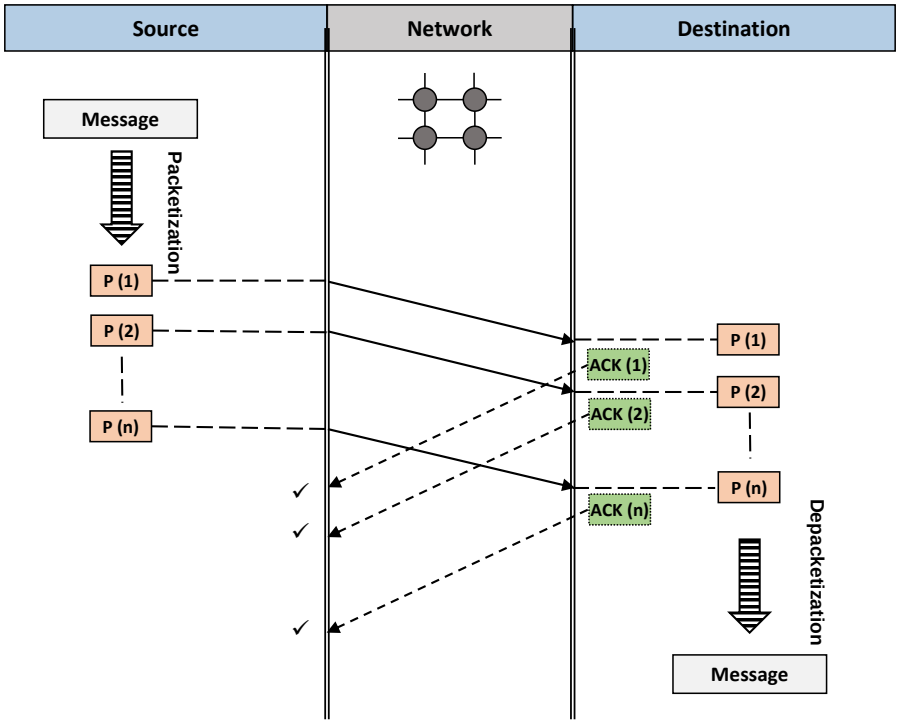


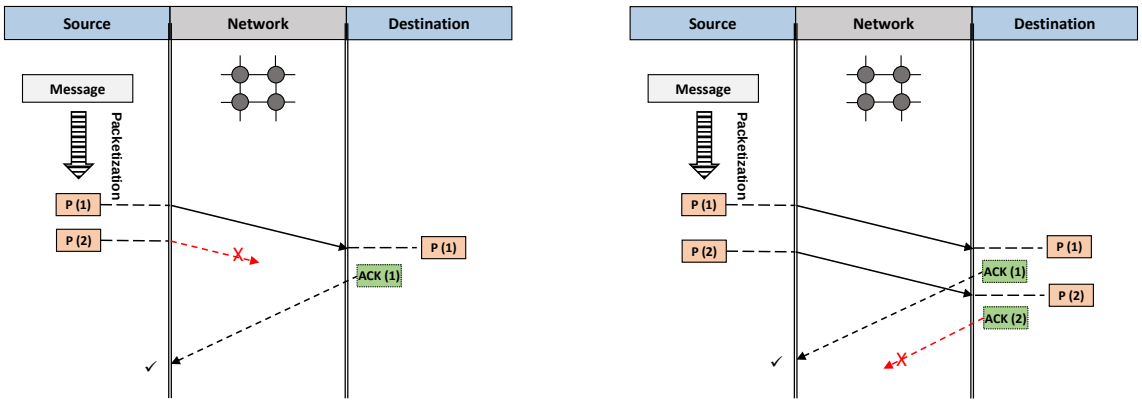
Figure 5.24: A sequence diagram of an end-to-end communication protocol for packet delivery assurance.

However, a packet could be discarded from the NoC due to a BHR attack. In this case, there are two possible attack scenarios: 1) The message is dropped on its way to

the destination node at an intermediate malicious router, i.e., BHR. 2) The ACK itself is dropped on its way to the source node, i.e., the message has been successfully received at the recipient node but the source is not aware of its successful delivery, which leads to re-sending the data again. In either of these behaviors, the BHR attack has occurred by either dropping the message or the ACK. Figure 5.25 shows these two possible scenarios of DoS attack due to BHR.

In Figure 5.25a, packet, P(1), was successfully received at the intended destination. Consequently, an ACK(1) is calculated and sent back to the source node. However, P(2) was dropped on its way to the destination node, as a result, ACK(2) will not be generated. Figure 5.25b presents a scenario where an ACK is dropped on its way to the source node despite a successful arrival of the packet to the final destination node.

In case a data-packet has been lost due to a BHR, the recipient node will never receive a packet and thus it will not generate an ACK or the source node will never receive an ACK although the packet has been received successfully.



(a) Packet drop scenario **(b)** ACK drop scenario

Figure 5.25: Two scenarios of the BHR attack on e2e packet delivery protocol.

Our e2e protocol of the BHR detection is divided into two parts: 1) BHR attack detection, i.e., detecting the drop of a data packet or an ACK packet, and 2) BHR localization,

i.e., localizing the infected router in the NoC to be isolated.

In order to detect the packet dropping attack in the NoC, a threshold time (t_Δ) that specifies the maximum waiting time for receiving an ACK from the receiver is set at the sender side. In case an ACK was not received within (t_Δ), the packet is re-transmitted. If the ACK has not been received, the sender raises a flag indicating that a packet dropping attack has been detected. The threshold time values between a source and each destination node depend on several factors: NoC size, injection rate, and the NoC traffic distribution. Therefore, (t_Δ) has to be determined for each case and application.

In the scenario when the ACK packet itself is dropped from the NoC, the protocol is still able to detect the BHR attack. In this case when (t_Δ) expires, the source node re-sends the packet again one more time, as explained. At the destination node, the packet sequence is compared with the one that was previously received. If they are the same, the receiver detects a BHR attack on the ACK packet.

Figure 5.26 shows the packet format for the energy efficient NoC model. The packet carries valuable information to the e2e model platform and is described as follows:

- Source ID (src ID): to identify the source of the message for routing the ACK packet back to it.
- Destination ID (dst ID): to identify the destination node for routing the data-packet to it.
- Packet type (type): to identify the type of the packet if it is either data, an ACK, or a control packet.
- Message ID (msg ID): to identify the message and its sequence to the source and recipient nodes which is used for generating and verifying proper ACKs.

- Packet ID (pkt ID): to identify the packet in the NoC, where each data-packet has a corresponding ACK-packet.
- Payload: to carry the data to be transferred via the NoC in case it is a data packet, or the acknowledgement for ACK-packet type.

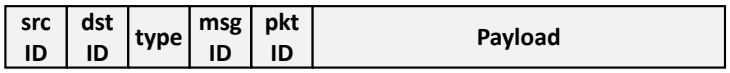
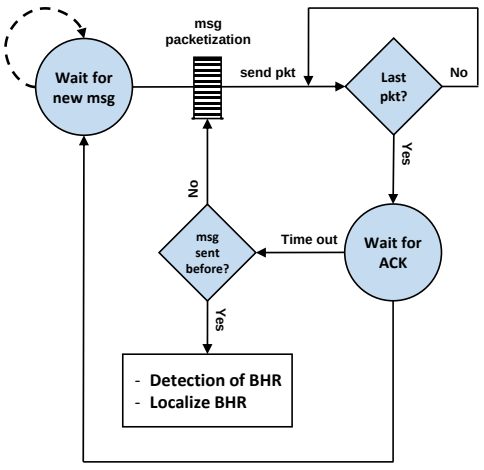
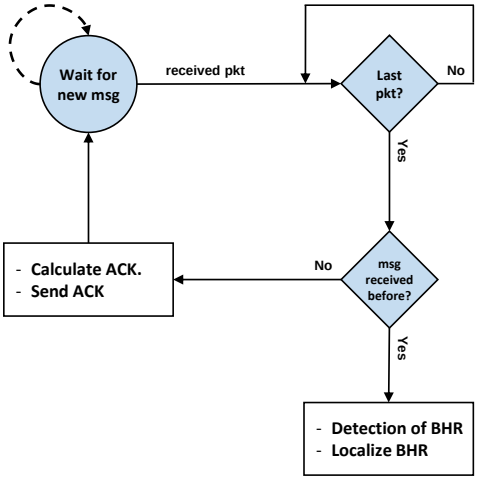


Figure 5.26: Packet format for the Energy-Efficient e2e protocol.

Figure 5.27 shows the state diagram of the transmitter and the receiver protocols at the source and the destination nodes, respectively. In Figure 5.27a, the transmitter waits for a new message. As soon as a message is ready to be sent to the intended node, it is packetized to multiple packets and injected into the NoC. When the last packet is released, the sender module waits for an ACK from the intended destination node to be received within the threshold time (t_Δ). For successful arrival, an ACK should be received in the



(a) State diagram of the transmitter module at the source



(b) State diagram of the receiver module at the destination

Figure 5.27: State diagram of the transmitter and the receiver modules at a node.

defined time. If waiting time expires, the sender checks if the message was sent before. If not, the message is resent. If it was sent before, it declares a Black Hole attack and requests for localization process to identify its location in the NoC. In Figure 5.27b, the receiving module receives a message. Once the last packet is received, it checks if the message was previously received. If not, it calculates the proper ACK and sends it back to the source. If it was received before, it means that a BHR attack exists on the path of the ACK and requests searching for the location of BHR.

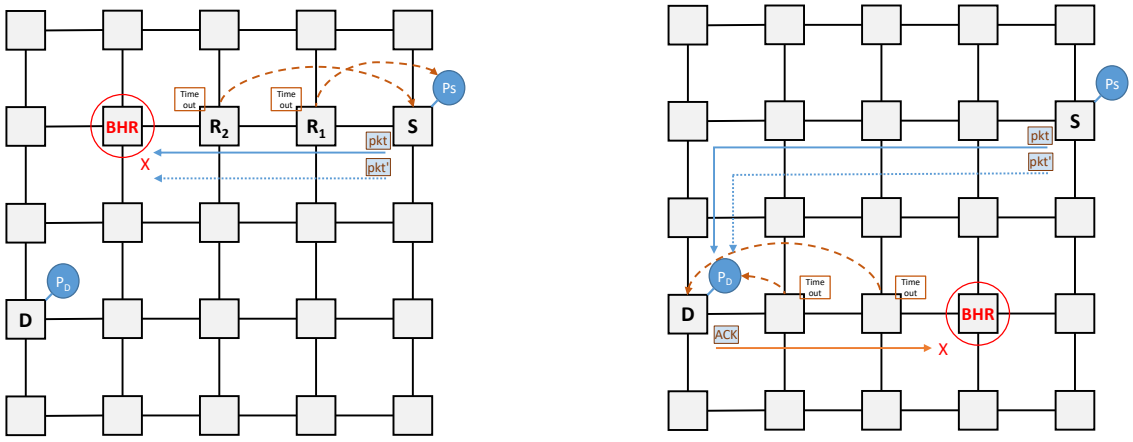
At this moment, the DoS attack has taken place and a BHR attack has been detected. However, this method does not identify the location of the BHR in the NoC because detection alone does not provide enough information to interpret the origin of the attack.

After the BHR attack is detected, the h2h detection technique, explained previously in Section 5.2, is activated only for this path and the victimized message will be re-sent requesting the h2h Secure-ACK. This security scheme has two different detection procedures, e2e and h2h. In this scheme, an energy-efficient BHR attack detection and localization technique is designed to leverage the existing e2e and h2h approaches to selectively activate and deactivate the detection units at each hop in the routing path between the source and the final destination nodes.

Figure 5.28 shows two scenarios of attack and how the BHR localization is performed. The first scenario is shown in Figure 5.28a when packets are dropped on their way to the destination. In this example, P_s sends packet, pkt, to P_D and waits for a secure confirmation that it has successfully arrived. In case that P_s has not received an ACK, it resends the packet, pkt', one more time and waits for an acknowledgement. If it does not receive an ACK, the Black Hole attack has been detected and then it activates the secure signature technique by a special control packet to locate the BHR.

The second scenario of attack is explained in Figure 5.28b where the packet, pkt is

injected by the source, P_s, and has arrived at the destination, P_D, that has generated an acknowledgment packet, ACK, and sent it to P_s. However, it has been dropped due to a Black Hole attack. Since P_s has not received an acknowledgement that the packet has been delivered, it re-sends the packet, pkt', one more time. At the destination side, P_D receives pkt' which was previously received. If P_D receives a packet two times, it means that a Black Hole attack has occurred on the ACK on its way to the source node. Therefore, a special control packet is sent on the way of the ACK to activate the h2h secure signature technique to localize the BHR.



(a) Localization of a BHR on the path of the packet

(b) Localization of a BHR on the path of the ACK

Figure 5.28: Localization of the BHR in two different scenarios of attack.

In these scenarios of detecting the Black Hole attack and localizing its source, a message may be sent multiple times. However, the destination node compares the msg ID and pkt ID to identify duplication of the received packets. On the other hand, the ACK may be received late at the source node after the packet has been re-sent. In this case, when the packet is received at the destination node for the second time, it sounds that the ACK was dropped and activates the signature detection technique to detect and localize the attack

although it was not. At the source side, the ACK will be received one more time, which can be omitted by the source after checking its ID. This will not affect the correctness of the model but it will add a little extra overhead.

Detection and localizing technique of a BHR is not the final solution to a successful packet-delivery. Once the BHR location is spotted, the malicious node is isolated from the NoC and the routing algorithm is reconfigured to detour around such malicious nodes. This presented scheme has three security stages: First, detecting the BHR attack; Second, localizing the BHR; and the Third is isolating it.

5.4.3 Evaluation and Experimental Results

In this section, we evaluate the performance overhead of the Energy-Efficient and Secure Router (EER) and compare it with the Secure Signature Router (SSR). In order to evaluate the EER architecture, the NoC parameters and the simulation environment were configured to cover different NoC sizes and buffer depth sizes for various injection rates and several synthetic traffic patterns [64]. The setup of the simulation environment is the same as previously described in Section 5.3. Simulations were run for different buffer depths and NoC sizes. The analysis results of different NoC and buffer sizes were very similar. Therefore, the analysis results of an 8×8 NoC size with buffer depth of 4 packets are provided in this section.

The main objective of this section is to evaluate the Energy-Efficient router model. Extensive experiments were run to assess the presented secure router and the NoC under different network traffic, buffer sizes, and injection rates. The system throughput and overhead were analyzed for these NoC parameters. The collected results were compared to the NoC model presented in Section 5.2. The simulation objectives are summarized as follows:

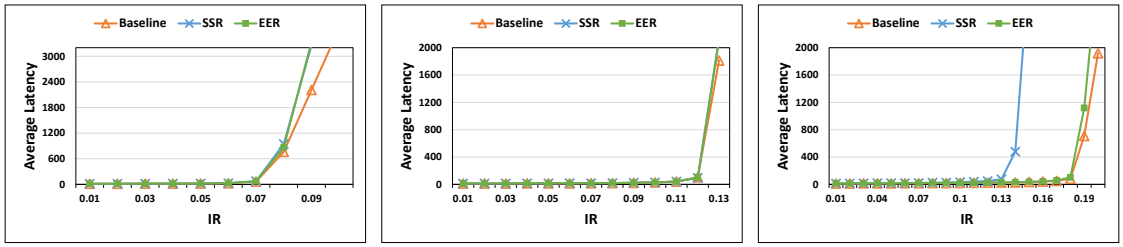
1. Measuring the saturation point of the EER-based NoC model and its throughput.
2. Measuring the energy consumption of the EER-based NoC model.
3. Measuring the degradation of the NoC performance in the presence of a BHR for the EER-based NoC model.

Throughput and saturation point

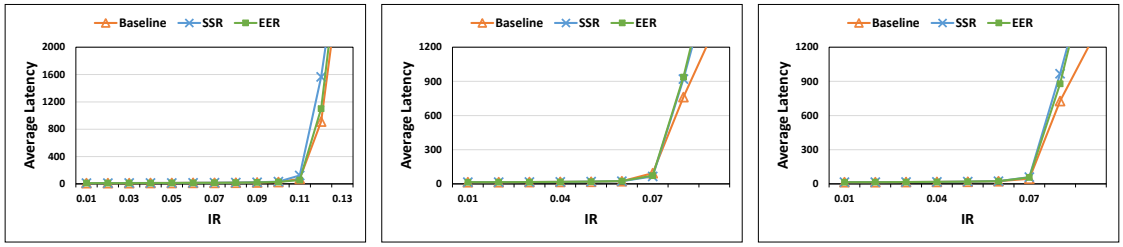
In order to evaluate the overhead of EER-based NoC model performance and identify the saturation point, the e2e runtime detection technique of the BHR attack was integrated into the baseline NoC. The NoC was HT-free and the localization unit is deactivated.

Figure 5.29 shows the average latency of the baseline NoC and both SSR-based and EER-based NoC models for different traffic patterns. The EER-based NoC model has quite similar saturation points compared to the baseline model. It saturates at 7%, 12%, 19%, 11%, 7%, and 7% for Bitreversal, Butterfly, Random, Shuffle, Transpose_1, and Transpose_2 traffic patterns, respectively. The similarity of saturation points between EER-based and baseline models is due to the less number of the ACK injected in the NoC, where the ACK is generated at the final destination node, unlike the SSR-based NoC model, where several ACKs are generated for each hop.

Figure 5.30 shows the throughput of the NoC model for different traffic patterns. Similarly, the EER-based NoC model has less overhead and shows similar performance to the baseline one.

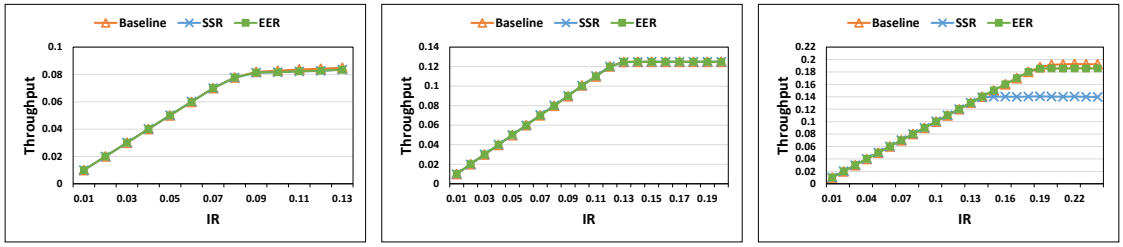


(a) Bitreversal traffic **(b)** Butterfly traffic **(c)** Random traffic

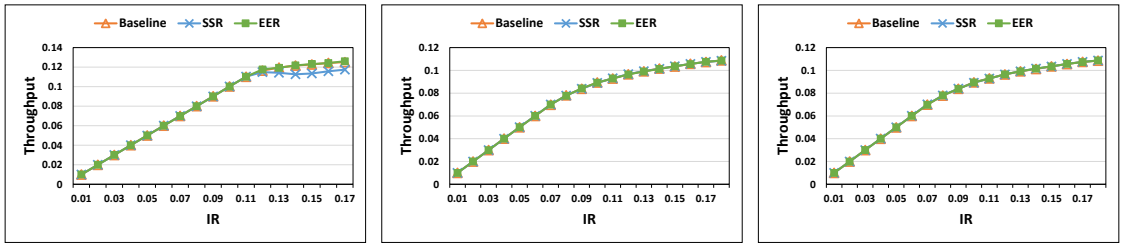


(d) Shuffle traffic **(e)** Transpose_1 traffic **(f)** Transpose_2 traffic

Figure 5.29: Average Latency of the SSR-based NoC model and the EER-based NoC model under several traffic patterns, the injection rate (IR) is measured in Flit/Cycle/Node.



(a) Bitreversal traffic **(b)** Butterfly traffic **(c)** Random traffic



(d) Shuffle traffic **(e)** Transpose_1 traffic **(f)** Transpose_2 traffic

Figure 5.30: Average Throughput of the SSR-based NoC model and the EER-based NoC model under several traffic patterns, the injection rate (IR) is measured in Flit/Cycle/Node.

Energy consumption overhead

In this experiment, the energy consumption of both the SSR-based and the EER-based NoC models were measured under different traffic patterns and for different injection rates. The measured energy overhead was normalized to the baseline NoC model. Figure 5.31 shows the energy overhead of both the SSR-based and the EER-based NoC models to the baseline router for different traffic distributions at the injection rate of the saturation point. As expected, the always-active h2h BHR detection consumes a significant amount of energy compared to the energy-efficient model. The EER-based NoC model adds 1% to 3% of energy overhead, unlike the SSR-based one where the energy overhead ranges from 9% to 24%. The low overhead provided by the EER-based NoC model makes it appealing to embedded system design.

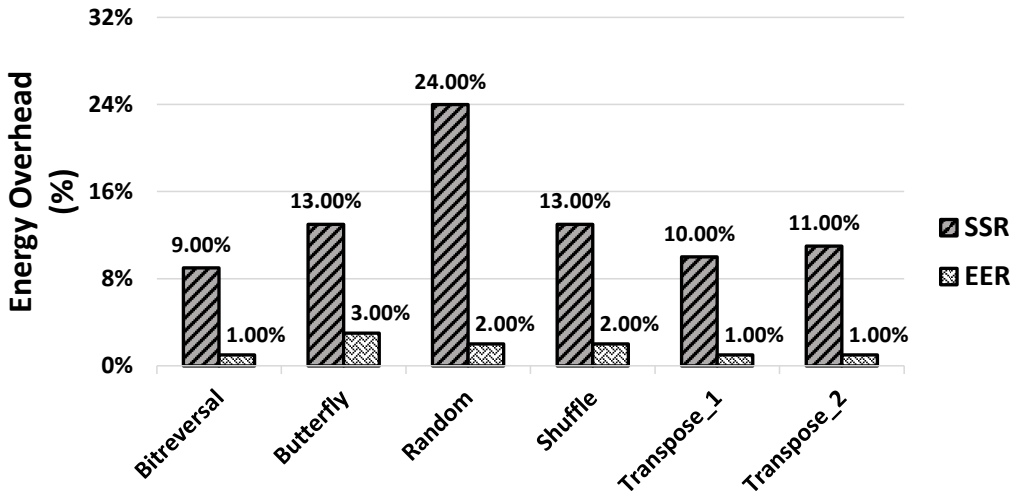


Figure 5.31: Energy consumption overhead of the SSR-based NoC model and the EER-based NoC model for different traffic patterns.

Performance Degradation

In this experiment, the effect of the EER on the NoC performance during detouring around an infected node was evaluated. The NoC was infected with a BHR (packet-dropping attack) and the throughput was recorded and evaluated. Figure 5.32 demonstrates the normalized NoC throughput of an 8×8 NoC size for Random traffic pattern for both the SSR-based and the EER-based NoC models. The presented architecture includes the detouring scheme to detect and avoid the BHR attack.

The throughput is normalized to a Trojan-free NoC. As it is expected, when the NoC experiences a BHR, the average throughput is slightly decreased with a low injection rate. The throughput decreases with increasing the injection rate. The performance overhead is due to packets detouring around the infected node experiencing a longer path to reach their destinations. The EER-based NoC model shows slightly less performance degradation. However, it presents higher throughput compared to the SSR-based one.

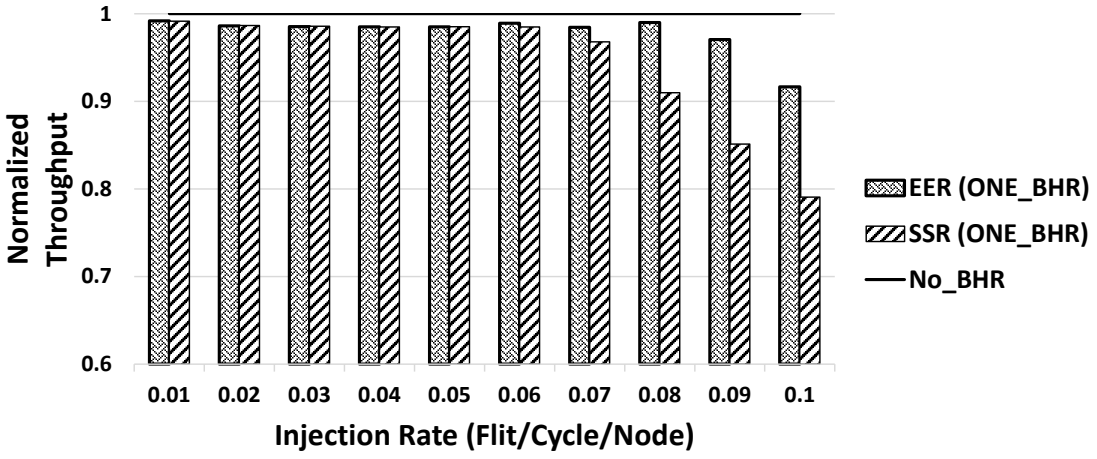


Figure 5.32: Normalized performance of the SSR-based NoC model and the EER-based NoC model under a BHR attack.

5.5 Summary

In this chapter, a novel NoC model to detect a packet-dropping attack in the Network-on-Chip (NoC) due to outsourcing design that may infect the network with a Black Hole Router (BHR) was presented. A secure protocol for the NoC-router with runtime detection and protection of the Black Hole attack was also modeled. The effect of detection of a BHR and avoiding it on the performance of the NoC was studied. The designed router architecture detects the BHR in runtime and detours around infected nodes once they are detected, with an overhead 26.9% in resources utilization. This novel, always-active, BHR detection technique has 21.31% overhead in performance, with 22% overhead in energy consumption, as compared to the baseline NoC model.

The model was extended to provide an Energy-Efficient technique to detect the BHR with negligible overhead in the NoC throughput and power consumption based on an end-to-end (e2e) acknowledgement technique. It presented 2% and 1% overheads in the energy consumption and the throughput, respectively, compared to the baseline NoC model.

CHAPTER 6

AUTHENTIC AND SECURE NOC MODEL AGAINST PACKET TAMPERING ATTACKS

The protection schemes discussed in the previous chapters have focused on protocols to detect the Black Hole Router and isolate it from the NoC by detouring the packet routing around it to prevent the source of the attack and to assure a packet delivery. In this chapter, a new protocol is presented to provide a security service to the data packets moving in the NoC and to detect a packet tampering attack. This new scheme provides a security level to the packets moving in the NoC by encrypting them, maintains their integrity, and supplies packets with authentication. This technique provides an Authentic and Secure Router (ASR) to the NoC model. The ASR-based NoC is not only able to recognize the attack, but also to localize the source of the attack and to isolate it from the NoC routing to prevent any further attacks.

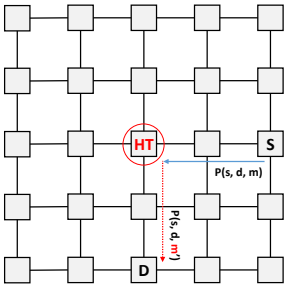
6.1 Threat Overview

In the NoC, packets are routed from a node to another in plaintext, which makes the system vulnerable to the theft of sensitive information by listening to the flowing packets via either invasive or non-invasive methods [77]. An attacker can run a malware application on one of the compromised processors to receive an unauthorized copy of the packets passing through a certain node [121]. In order to countermeasure this attack, a technique to hide

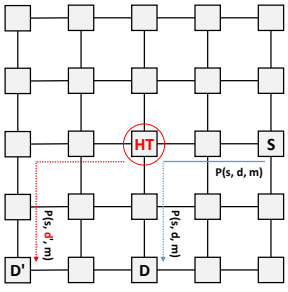
the information carried by the packets is required. This can be performed by using an encryption protocol.

This security threat model may tamper with the packets and direct them to a certain destination to leak some information, gain unauthorized access, or apply a DoS attack. Figure 6.1 presents several types of attacks on the NoC due to tampering with the packets, where $P(s,d,m)$ is a packet P from a source (s) to a destination (d) with a payload, or a message, (m).

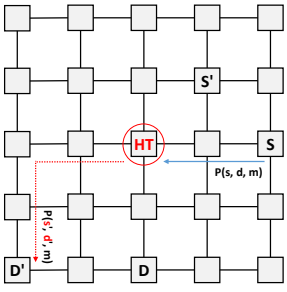
Figure 6.1a shows an example of tampering with the message (m) to apply a DoS, where the destination receives a wrong message (m') and may take an undesired action leading to the degrading of the NoC-based system. A different type of packet tampering attack is presented in Figure 6.1b, The figure shows an example of a packet-leak attack which occurs by copying a packet to a compromised node by changing the destination to (d'). In this attack, packets still progress to their intended destination, but another copy of them are forwarded to the attacker for further analysis. Figure 6.1c shows a third attack scenario for obtaining unauthorized access to a certain node. For a certain security level, some nodes in the NoC are not allowed to be accessed by others for security purposes. For example, in a secure zone, the sources of the packets are validated before accessing the destination node.



(a) A packet tampering to apply DoS



(b) A packet tampering to leak sensitive information



(c) A packet tampering to gain unauthorized access

Figure 6.1: Potentials for packet tampering attacks and their purposes: a) to apply DoS, b) to leak information, or c) to gain unauthorized access.

In order for an attacker to access an unauthorized node, the packet header is altered by a HT to change their source identification (s') to obtain the privilege to access a forbidden node (d').

The purpose of the several scenarios of the packet attacks in the NoC presented in Figure 6.1 is to apply DoS, steal sensitive information, or access unauthorized data. In this chapter, a security layer is added onto the NoC to prevent the possibility of packet attacks. The goal of this work is not only to maintain the data confidentiality, integrity, and authenticity, but also to identify the source of the attack and detach it from the NoC.

6.2 Threat Mitigation

In order to hide sensitive information carried by packets in the NoC, packets need to be encrypted during their path from the source to the destination. Several encryption techniques [107, 108, 119, 120] may be applied to provide confidentiality to the system. However, packets may be diverted to an unauthorized and compromised node to be analyzed by a crypt-analysis technique to cover sensitive information or to steal the cryptographic key. This may require a robust encryption technique to conceal the data and make it harder to reveal. Additionally, encrypted packets can be altered on their way and be directed to a different destination. In this case, the destination receives incorrect information after decrypting the received message.

In order to address such problems, encryption alone is not enough and maintaining the integrity of the data is a must. Therefore, only correct packets received at the recipient node will be accepted and the wrong or unauthorized packets will be rejected. In order to achieve this goal in the presented scheme, Authenticated Encryption (AE), a tag is formed by the source node and is appended to the packet. At the receiver side, the tag is generated

from the received packet and is compared to the received one, i.e., the source-tag. Since the generated tag is correlated with the data, an equality of the destination-tag and source-tag means the data remained intact on its way to the recipient node. Generating the tag from the payload only guarantees data integrity however it does not guarantee authenticity, where an attacker may change the source or destination IDs and generate the corresponding tag. Hence, in this presented AE technique, the source and destination IDs, and the payload are included in generating the tag.

Figure 6.2 shows the Authenticated Encrypted (AE) message format being used, where the tag is composed of the source and destination IDs, the type of the packet, and the digest of the information that needed to be transferred. The type of the packet is included inside the AE message to be prevented from alteration by the HT.

In this AE scheme, the AE message is created at the source node inside the network interface which is assumed to be trusted and built-in-house. At the source side, the message is divided into multiple packets, each packet has a 32-bits payload. For an 8×8 NoC size, each of the source (src) and the destination (dst) is represented by 6-bits length. The message can have different types, either a data packet or a control packet. It could also

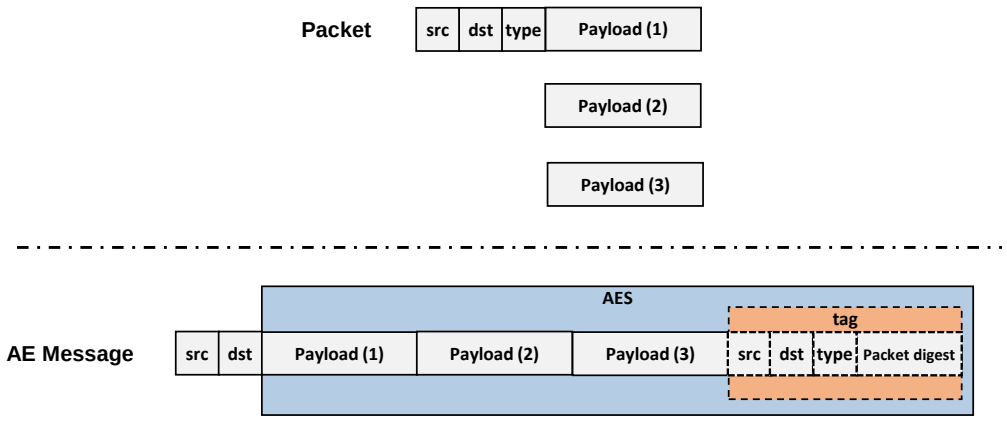


Figure 6.2: Authenticated Encrypted (AE) message format.

be extended to include other types based on the environment of the system. In this case study, “type” is 4-bits length, which could represent up to 16 different types for future extension. “Packet digest” is 16-bits generated from “Payload (1)”, “Payload (2)”, and “Payload (3)”. Generating the “Packet digest” is discussed in the next section. Therefore, the size of the generated “tag” is 32-bits and the total size of the message is 128-bits. After forming the payloads and the generated tag, they are encrypted using the Advanced Encryption Standard (AES) [152], AES-128, by using a shared key between the source and the destination. The source safely injects the AE message into the network to its intended destination. At the destination side, the received AE message is decrypted and the data payloads and the tag are extracted, and then the node recalculates the packet digest and compares it, along with the source and the destination IDs, to the received tag, in order to check the correctness of the received packet.

Figure 6.3 shows a block diagram of the sender entity at the authentic module of a source node. Initially, the sender is waiting for a new message from its attached processing element. Once a new message (msg) has arrived, it is divided into packets. A digest is calculated for each packet and is concatenated with the source ID, the destination ID, and the type of the packet to create the packet-tag. Each node has a secure storage for cryptographic keys saved in the “key Table”. The required key is extracted from the “key Table” using the destination ID address. For authenticating the message, the data-packet along with the packet-tag are signed by the shared key, $Key_{(src,dst)}$, between the source and the destination nodes using AES-128. Therefore, the data is secure to be pushed into the network. The process is repeated for every packet till the end of the message.

The purpose of this message format is to maintain the following goals:

- **Data Confidentiality**: the data packets (Payload (1), Payload (2), and Payload (3)) are encrypted.

- **Data Integrity**: the correctness of the data is maintained when the received tag matches the calculated tag at the recipient node.

- **Authenticity**: only the permitted sources have the right to access the computational or memory element at the destination node. The message authentication is achieved when the calculated tag is correct, since the crypto-key used for the encryption and decryption is uniquely shared between the source and the destination.

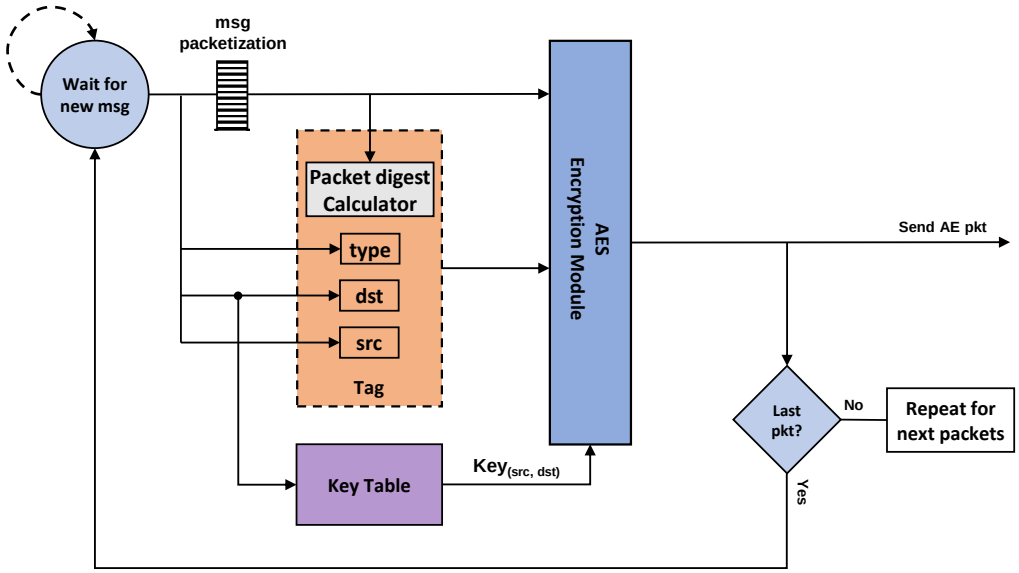


Figure 6.3: A block diagram of the sender node.

6.2.1 Packet Digest

There are many techniques to create a digest of a message, such as calculating the hash of the message using a one way function, such as secure hash algorithm as seen in MD5, SHA-1, SHA-2, or SHA-3 [149]. These techniques are time demanding and require massive computation resources [150].

In the presented AE scheme, a secure hashing is not necessary for two reasons: 1) the tag is calculated and then is signed with the data by the shared key, so an attacker will not have access to the tag to run crypt-analysis to detect the message. 2) altering a single bit of the encrypted message leads to a complete different output message, due to the diffusion property of the standard AES protocol, so a simple packet digest is enough to check the correctness of the message.

The purpose of the packet digest inside the encrypted message is to guarantee packet integrity. It was explained that the data packet and its digest are encrypted before being sent to the destination. This prevents an attacker from altering a packet and generating a corresponding new tag. Therefore, a cyclic redundancy check (CRC), a Hamming code, or an arithmetic checksum are fit candidates to calculate the message digest.

In this work, an arithmetic checksum is used for calculating the packet digest. Figure 6.4 illustrates the packet digest using the arithmetic checksum. In this AE scheme, a packet is composed of three payloads, each having 32-bits of length. Payloads are divided into multiple words of 16-bits length and added together. The 16-bits output summation result (CheckSum) is the digest of the packet.

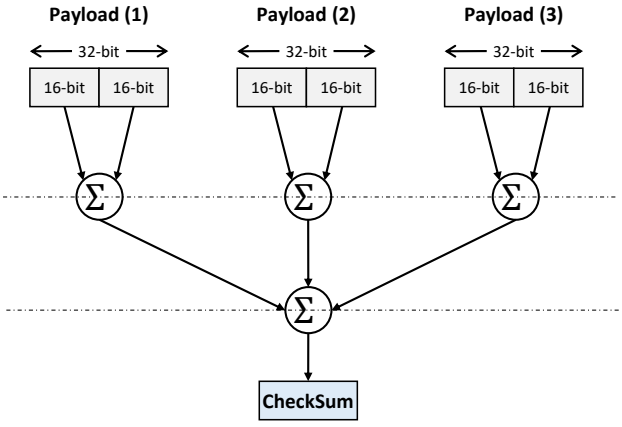


Figure 6.4: Generating the packet digest using the arithmetic checksum.

6.2.2 Attack Detection

By following the presented AE message format, the receiving node can detect any alteration of the received packet by decrypting the message using the shared key with the source. If the packet has been attacked, the decrypted message will not have a matched tag, which is either, src, dst, or the packet digest. Once the tag is not correct, the destination node can claim that a packet attack has occurred. It is must be pointed out that packet alteration due to link failure is out of the scope of this research, since the focus here is on attacks due to a malicious circuit in the router.

Figure 6.5 shows a block diagram of the detection unit at the receiving node. The received packet contains the source and destination IDs, the type of the packet and the authenticated encrypted packet. The source IDs are the address to obtain the shared key between the two nodes from the secured storage, "Key Table", which is used for decrypting the received packet. The output from the "AES Decryption Module" is composed of the payload (plaintext message) and the packet-tag that contains information about the source and destination IDs, packet type, and the packet digest. The payload is passed to the "Packet digest Calculator" to calculate the message digest. Therefore, the calculated message digest along with the received source and destination IDs will be compared with the decrypted packet-tag. If a packet has changed on its way to the receiving node, the calculated tag will not match the decrypted received tag.

The moving packet in the network consists of several parts. Some of them are in plaintext and known to the intermediate routers for the routing process such as, source ID, destination ID, and packet-type. Others are encrypted in the authenticated packet (AE pkt) such that the internal information is hidden to the routers of the NoC. Tampering with a packet can happen in one or more parts of the moving packet. For example, in the plaintext

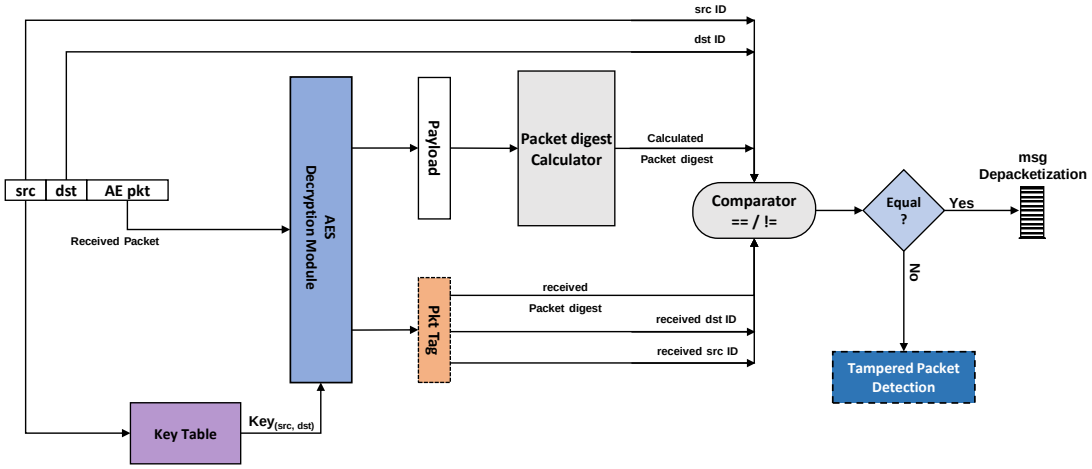


Figure 6.5: Detection of the packet tampering at the receiving node.

information or in the encrypted packet. Flipping one bit of the moving packet leads to an expected mismatch of the received tag with the calculated one.

Table 6.1 shows the consequence of tampering with different parts of the packet. When the source ID or the destination ID is tampered with, the cryptographic key will be different than the one that was used to encrypt the AE packet. As a result, the decrypted packet will be different than the intended packet. This will lead to a different tag as well. Additionally, when the AE packet is tampered with, the decrypted packet will be different, which may affect the correctness of the sent-tag due to the avalanche effect of the AES [152].

Table 6.1: Effect of the packet tampering on the decrypted packet.

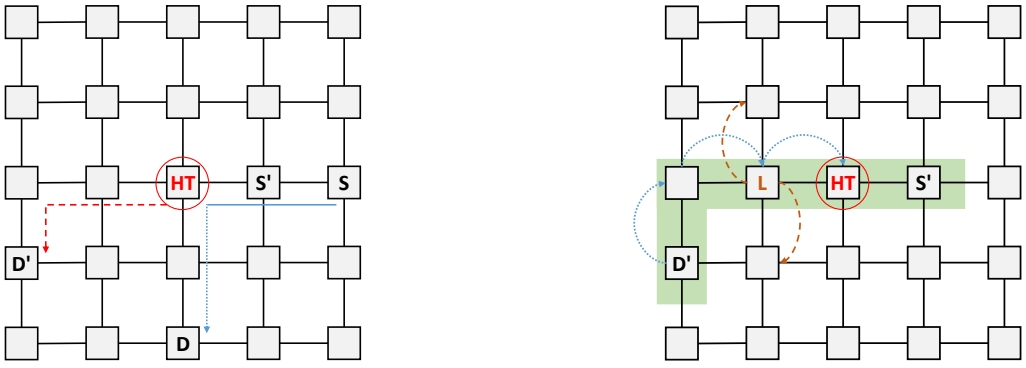
Packet Part	**Effect**
src ID	Wrong Key
dst ID	Wrong Key
AE pkt	Wrong decrypted packet. (src, dst, type, or/and packet-digest)

6.2.3 HT Localization

Once a tampering attack has happened to the message during the transfer process, it will be detected at the Authentication Module (AM), which is attached at the recipient node. However, the source of the attack has not been identified. In this work, a novel protocol to identify the location of the tampering node in the NoC is presented.

In order to accomplish this task, the AM creates a Scouting Packet (SP) to track the path of the received message to identify the source of the attack. The SP traces back, hop by hop, from the recipient node to the source of the message. The source ID is extracted from the plaintext header information, which is the *src ID*. Figure 6.6 shows an example of the detection and searching processes of the HT.

In Figure 6.6a, a packet was supposed to be forwarded from a source node, S, to a destination node, D, however it has been tampered with at a HT node. The source and the destination IDs were changed to S', and D', respectively, for an attacking purpose. Once the tampered packet arrives at D', it will be decrypted by a different key, $Key_{(S',D')}$, which



(a) Detection of HT attack **(b)** HT Localization

Figure 6.6: Searching for the source of the attack.

leads to an unmatched tag. Hence, tampering is detected. Figure 6.6b shows the path for searching the tampering router. At this moment when an attack has been detected at D', the AM at D' instantiates an SP to be routed from D' to S'. At each hop, an encrypted packet is sent from a node to the adjacent one on the searching path. Once it arrives at a node, it replies with a Scouting Packet Reply (SP_REPLY), which is a secure confirmation, back one hop and calculates the SP and forwards it one hop towards S'. The process is repeated till it reaches the intended node, S'. The SPs are routed in a reverse direction of the X-Y routing technique, i.e., the reverse path of the data packet from D' to S' in the Y-X routing direction. The SP and SP_REPLY have a similar format to the original data packet, which is described in Figure 6.2. In their format, the Payload (1) is a random number generated by the node, and the Payload (2) carries information about the packet, such as the packet ID, however, the Payload (3) carries information about the original source, src_{org} (D') and the original destination, dst_{org} (S'), IDs. The SP_REPLY packet has the same packet format except that Payload (1) is a response to the received one from the SP. In this presented model, a bitwise-Inverse function of the received Payload (1) was used as a response to the the SP packet. The type of packets are described in Table 6.2.

Table 6.2: Types of the decrypted packet.

Packet Type	**Description**
DATA_TYPE	The packet carries data information
SP	The packet carries a control packet to detect & localize a HT
SP_REPLY	The packet carries a control packet to confirm the node's authentication

In the case that the SP faces an infected router (HT), as shown in Figure 6.6b, the AM attached to the node, L, sends a secure packet to the HT, however the HT tampers with the packet and a confirmation packet will not be sent back to L. At this time, L keeps

waiting for an ACK till it exceeds a threshold time (T_{th}). The value of T_{th} is defined based on the time that the AM takes to create the SP and SP_REPLY. Hence, the AM runs the same protocol with the other adjacent nodes to check the correctness of the attached router, *L*. Once *L* is HT-free and does not receive a SP_REPLY, its attached AM sends the attack location to the OS through a secure link to inform it to isolate the HT node and reconsider mapping the applications - migrating the applications is out of the scope of this research. Algorithm 1 shows the pseudo code of detecting a routing violation in an infected NoC.

Algorithm 1: Runtime detection and localization of the tampering-HT.

```
Input:    Packet: (src, dst, Payload(1,2), src_org, dst_org, tag)
Given:    Current Node ID (x_curr, y_curr)
Result:   Tampering-HT Location
Packet Decrypt () ;
Packet Correctness Verify () ;
if type == DATA_TYPE then
  | Packet Consume () ;
end
if type == SP then
  | Scouting Packet Reply Create () ;
  | Scouting Packet Create&Forward () ;
end
if type == SP_REPLY then
  | Scouting Packet Correctness Verify () ;
end
```

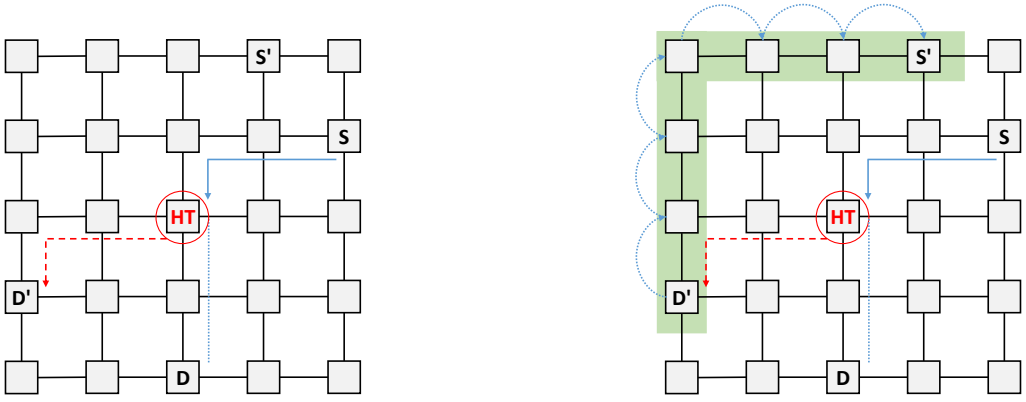
So far, this technique finds the location of a HT in the NoC as long as it is in the searching path. A question might be raised. Is there a case where the HT hides itself from the searching path?

In answering this question, a case that a HT covers itself by a simple technique is assumed. The following section describes such failure cases and how they are mitigated in this secure scheme.

Mitigation of Unsuccessful Localization

As mentioned earlier, a tampering router (HT) may change any part of the packet. It might alter the source ID of the message such that it removes its location from the searching path. Figure 6.7 shows a scenario where a HT hides itself and the detection protocol fails to detect it. As shown in Figure 6.7a, a packet is sent from S to D passing through a HT which alters the packet source and destination IDs to S' and D', respectively. Once the packet arrives at D', the tampering attack will be detected by the AM attached to the receiving node. So, it instantiates the SP, forming the searching path as shown in Figure 6.7b. By following the searching path using the traditional technique, the HT will not be discovered.

Having discussed that a HT can conceal itself, an additional technique is required to discover a such unnoticeable HT, which is addressed next. As previously mentioned, the searching path is a trace back from D' to S' and the data packet was supposed to follow the routing protocol. Therefore, if a packet has changed its direction but it follows the routing restrictions, the searching path will include the tampering node.



(a) Undetected HT technique

(b) HT escapes from the searching path

Figure 6.7: An example of a HT localization failure.

Figure 6.8 shows an example of a routing violation that occurred because of a HT and is detected at node L. In this example, a data packet is moving from S to D. However, the HT router changes its direction and changes the source and the destination IDs to S' and D'. However, the router L receives the tampered message as it was sent from S' to D'. L realizes that it is not supposed to receive such a packet from S' to D' for the X-Y routing protocol. Therefore, L detects a routing violation at the east input port. Hence, it reports that the adjacent east router (HT), that is attached to the input port, had violated the routing protocol.

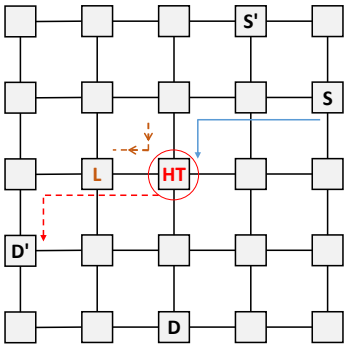


Figure 6.8: An example of routing violation due to a HT.

Routing Violation Detection

In the X-Y routing algorithm, packets are routed all the way horizontally in the x-direction ($EAST$ or $WEST$) till the x-coordinate of the destination is aligned with the x-coordinate of the current router, then packets are directed vertically in the Y-direction ($NORTH$ or $SOUTH$) to their final destination. If the packet is deviated to a wrong direction, the receiving (neighbor) router will detect it. Hence, when a packet is received at each port of a router, the receiving module of the router checks if the routing has been violated. In order

to detect the routing violation, the network is divided into certain regions around a certain node router. Figure 6.9 shows eight regions surrounding a specific node ("Local") in the NoC. In the X-Y routing, when a packet is received at the *EAST* input port of a router, the source should be in the $X-East$ region and the destination should be at any of the regions $Y-North$, $Y-South$, $X-West$, $North\_West$, $South\_West$, or $Local$.

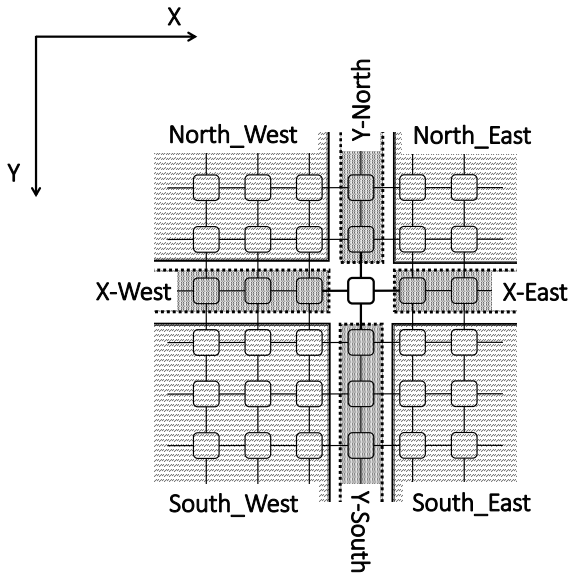
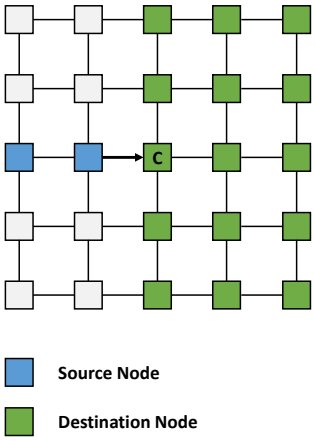


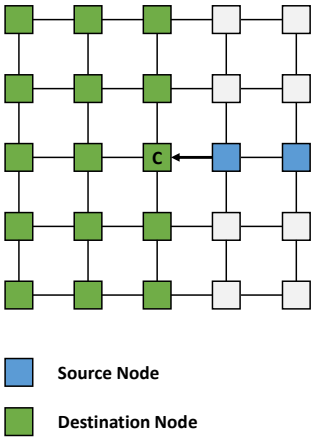
Figure 6.9: Eight types of segments around a certain node.

In X-Y routing, the packet follows some certain turns and directions at a local router based on the source and the destination IDs. When a packet is received at the *WEST* input port, the source should be in the $X-West$ region and the destination should be at any of the regions $Y-North$, $Y-South$, $X-East$, $North\_East$, $South\_East$, or $Local$, as shown in Figure 6.10a. Similarly, when a packet is received at the *EASTT* input port, the source should be in the $X-East$ region and the destination should be at any of the regions $Y-North$, $Y-South$, $X-West$, $North\_West$, $South\_West$, or $Local$, as shown in Figure 6.10b. When a packet is received at the *NORTH* input port, the source should be in the Y_North region and the destination should be in either of these regions Y_South, or $Local$,

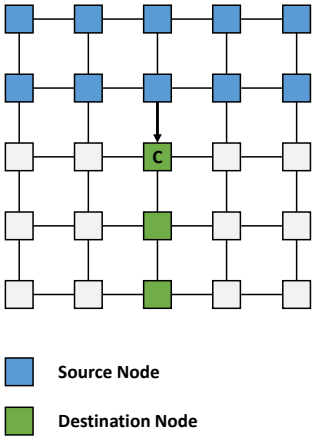
as shown in Figure 6.10c. Similarly, when a packet is received at the *SOUTH* input port, the source should be in the *Y_South* region and the destination should be in the *Y_North*, or *Local*, as shown in Figure 6.10d. Otherwise, a routing violation would have occurred and can be detected.



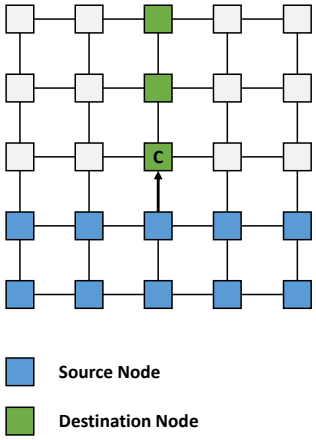
(a) A packet is received at the West input port



(b) A packet is received at the East input port



(c) A packet is received at the North input port



(d) A packet is received at the South input port

Figure 6.10: A graphical explanation of the relation between the source IDs, destination IDs, and the input port of a router to detect the packet misrouting attack.

Algorithm 2 shows the pseudo code of detecting the routing violation in an infected NoC.

Algorithm 2: Run-time detection of routing Violation.

```
Input:    Input direction, in_dir, (E, W, N, S)
          Source Coordinates (x_src, y_src)
          Destination Coordinates (x_dest, y_dest)
Given: Current Router ID (x_curr, y_curr)
Result: Violation Detection (True, False)
if in_dir == E then
    check (y_src == y_curr) ;
    check (x_src > x_curr) ;
    if dst != Local then
        check (x_dst <= x_curr) ;
    end
end
if in_dir == W then
    check (y_src == y_curr) ;
    check (x_src < x_curr) ;
    if dst != Local then
        check (x_dst >= x_curr) ;
    end
end
if in_dir == N then
    check (y_src < y_curr) ;
    if dst != Local then
        check (y_dst > y_curr) ;
        check (x_dst <= x_curr) ;
    end
end
if in_dir == S then
    check (y_src > y_curr) ;
    if dst != Local then
        check (y_dst < y_curr) ;
        check (x_dst == x_curr) ;
    end
end
return False        // No Violation;
```

In the presented NoC model, once the malicious node is detected, the routing technique of its surrounding nodes is updated and, hence, they do not completely follow the X-Y routing protocol. So, their routing violation detection techniques perform differently than the other normal nodes. Appendix D presents the routing violation detection technique for the reconfigurable routing algorithm that allows a packet to detour around the infected node.

Key Management Scalability

In order to maintain the authenticity of the ACK packet, it is encrypted with a shared key between the sender and the receiver. Keys between the processing elements can be safely distributed using Diffie-Hellman protocol [155], [156]. The key exchange process occurs when a new application is allocated to some processing elements. If these processing elements interact with each other, each one should have a separate key with the other nodes. It is not necessary that each node in the NoC has a distinct key with all the other nodes of the NoC. For example, if the total number of nodes in a NoC is sixteen and only four processing elements are mapped to one application and interact with each other, each node of these four processing elements should have only three distinct keys rather than fifteen keys per node. This will reduce the number of shared keys between the nodes in the NoC.

In the worst case, when all the PEs in N×N NoC interact with each other, the total number of keys, Num_{keys}, required in the system is proportional to the NoC size and equal to:

$$Num_{keys} = {}^{N^2}P_2$$

Since these keys are shared between each other, the total number of the distinguished keys for a $N \times N$ NoC is represented by the following equation:

$$Num_{disKeys} = ({}^{N^2}C_2) = \binom{N^2}{2}$$

Figure 6.11 shows the scalability of the keys in the system with the NoC size. It is clear from the figure that the total number of keys exponentially increase with the NoC size. However, for an application mapped to certain nodes in the NoC, the nodes should hold only the shared keys between them rather than for the whole NoC, where these nodes communicate only with each other. This is based on the applications that are mapped on these sets of processing elements.

Without loss of generality, in order for an application to run on certain processors in a multiprocessors system, an allocation process is required to allocate the application on the selected processors [35–38]. In this case, it is not necessary for those processors to have shared keys with others which are not in communication with them. This will reduce the number of the keys per node in the system. The key-exchange process between a certain nodes may be performed during the mapping process of the application using the Diffie-Hellman protocol [155], [156], such as in [113, 157–159].

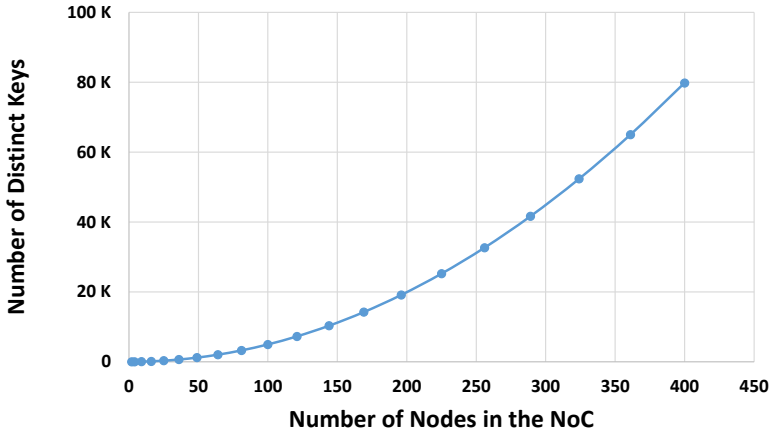


Figure 6.11: Scalability of the distinct keys with the NoC size.

6.2.4 Model of Authenticity and Security

Figure 6.12 shows a diagram of the Authentic and Secure Router (ASR)-based NoC model. Each router in the NoC is connected to an Authentication Module (AM), which is in charge of creating the authenticated and encrypted message before it is injected into the network. The AM also checks the message authentication to detect packet-alteration and spots the source of the tampering. The AM is a secure module that has access to the paired keys and is implemented in the network interface. Authenticated encryption is performed on the NoC firmware level. None of the routers in the NoC has access to the keys in the NoC. Routers are used for forwarding the packets from one node to another. For each router, a routing violation detection unit is designed and is placed in each input port of the router.

When an attack is detected and its location is defined, a secure signal is sent to a Secure Management Module (SMM) to take an action, such as isolating the infected node and

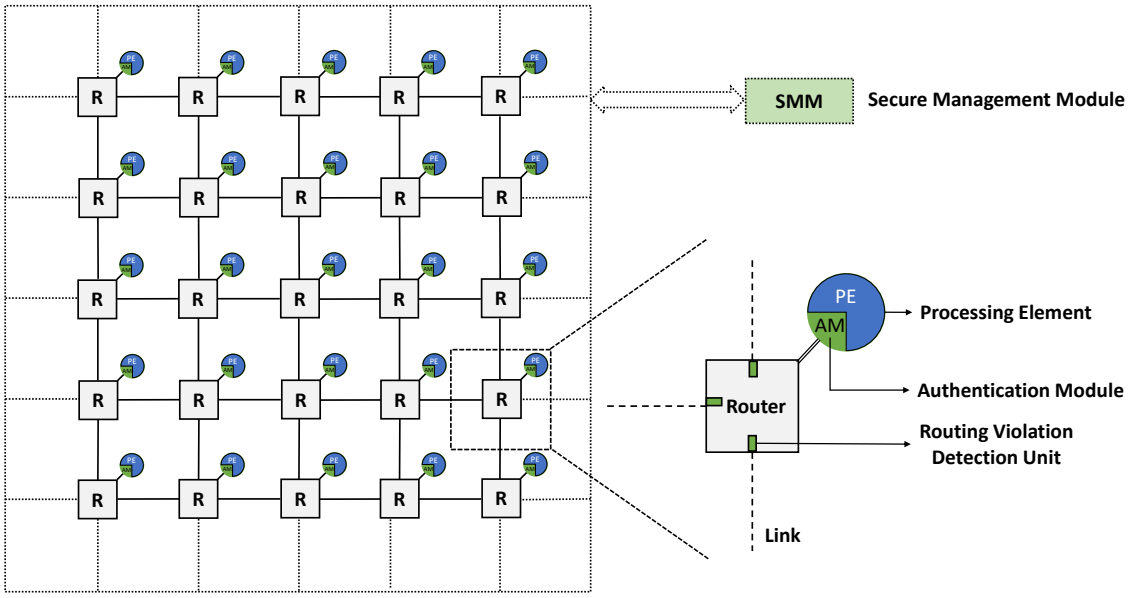


Figure 6.12: Secure NoC model with authentication module attached to the processing element.

avoiding mapping tasks to the attached processing elements. In order to increase the arrival chance of the secure signal to the SMM without tampering, the AM sends the special packet horizontally and vertically to the SMM. The received packets at the SMM are analyzed, to take an action to isolate the malicious router from the NoC routing.

In order to evaluate the area overhead of the presented secure router architecture against the packet-integrity attack, a five-ports router with a 4 packets depth buffer, each of 4 Flits (32 bits), with a "Store-and-Forward" (S&F) switching technique was designed and synthesized using 45nm TSMC technology (using the Cadence Design Compiler). It occupied an area of 73,215 μm^2. The baseline router was modified and the security features against the attack were added. The area overhead was increased by 24.21% of the baseline router.

6.3 Evaluation and Experimental Results

In this section, the performance overhead of the authentic and secure NoC model is evaluated . The NoC parameters and the simulation environment were configured to cover different NoC sizes and buffer depth sizes for various injection rates and several traffic patterns. In the conducted experiments, the packet size was set to three flits, each is 32-bits, and the tag size was set to 32-bits, which includes information about the source and the destination IDs, the packet type, and the packet-digest. The packet was authenticated and encrypted by the AES-128 protocol [152, 160, 161]. Experiments were run for different injection rates for 100,000 clock cycles.

6.3.1 Experimental Results

Extensive experiments were run to assess the authentic and secure NoC model under different network traffics, buffer sizes, injection rates, and etc. The system throughput and the

overhead were analyzed for different NoC parameters. The collected results were compared to the baseline NoC model.

The goal of these experiments is to identify the maximum injection rate where the presented NoC saturates. Additionally, it is crucial to recognize the required time to locate a HT in the NoC, where the source of the tampering attack can be spotted within the studied period of time. The simulation objectives are summarized as follows:

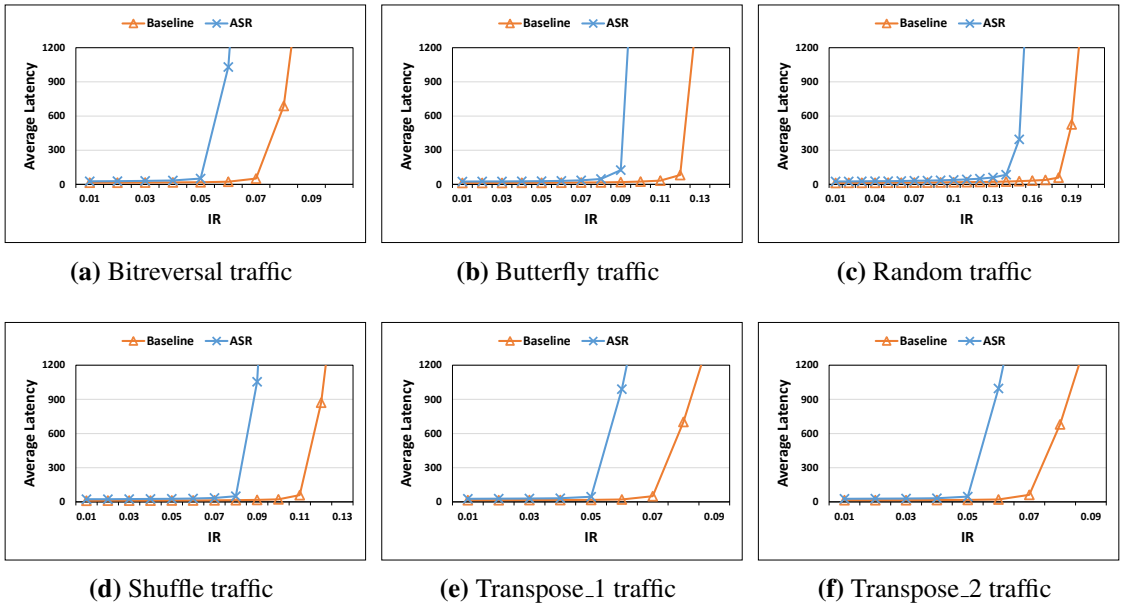
1. Identifying the saturation point of the presented authentic and secure NoC model and measuring the throughput.

2. Evaluating the searching time overhead for the HT localization.

3. Measuring the degradation of the NoC performance in the presence of a HT for the presented NoC model.

NoC Throughput and saturation point

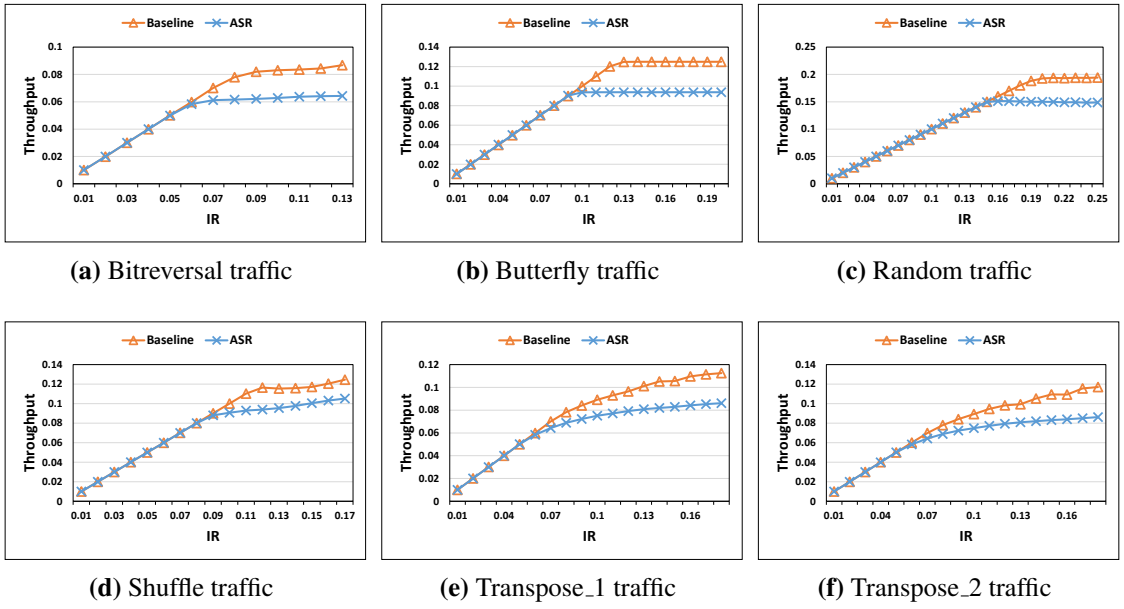
In order to evaluate the overhead of the Authentic and Secure Router (ASR) on the NoC performance and identify the saturation point of the presented NoC model, the NoC was free of HTs and the tampering detection technique was activated. Figure 6.13 shows the average latency (clock cycles) of the baseline NoC model and the presented, ASR-based, NoC model for different traffic patterns. The x-axis represents the injection rate of the data-flits in the NoC. The ASR-based NoC model saturates earlier compared to the baseline one for different traffic patterns as shown: (5% - 7%, Bitreversal), (9% - 12%, Butterfly), (15% - 19%, Random), (8% -11 %, Shuffle), (5% - 7%, Transpose_1), and (5% - 7%, Transpose_2). One reason that ASR-based NoC model saturates at a lower injection rate than the baseline one is that each packet is appended with a tag, which increases the number of flits in the NoC, however, the actual number of data-flits are less than the injected flits.



(a) Bitreversal traffic **(b)** Butterfly traffic **(c)** Random traffic

(d) Shuffle traffic **(e)** Transpose_1 traffic **(f)** Transpose_2 traffic

Figure 6.13: Average Latency of the ASR-based NoC model under several traffic patterns, injection rate (IR) is measured in Flit/Cycle/Node.



(a) Bitreversal traffic **(b)** Butterfly traffic **(c)** Random traffic

(d) Shuffle traffic **(e)** Transpose_1 traffic **(f)** Transpose_2 traffic

Figure 6.14: Average Throughput of the ASR-based NoC model under several traffic patterns, injection rate (IR) is measured in Flit/Cycle/Node.

For example, three data-flits (payloads) requires one tag-flit. This indicates that 75% of the routed flits are data and the rest 25% are tag-flits. Because of this, the throughput of the proposed NoC model decreases, as well, as shown in Figure 6.14. It can be observed that the average throughput slightly decreases, yet the confidentiality along with the packet integrity and authenticity are maintained to the notion of graceful degradation of service.

Localization Time Overhead

In this experiment, the overhead of the searching technique to localize the source of the packet-tampering attack is evaluated. The distance between the node that detects the attack and the source of the attack is the main factor to estimate the localization time. Therefore, in the experiment setup, only one node of the NoC was infected with a packet-tampering attack. For any packet that passed through this infected node (HT), its destination was tampered with and forwarded to a fixed node (D') that was n hops away from the infected node. As previously mentioned, the authentication module connected to the D' initiates the scouting packet to spot the source of the attack. The searching time to localize the infected node was measured and recorded along with the number of hops between the D' and HT nodes. The simulation was repeated for a different D' and a fixed HT position to cover all of the nodes in the NoC. After that, the location of the HT was changed to another node and the previous simulations were run and the localization time was recorded along with number of hops between the D' and HT nodes to cover all the possible cases.

Figure 6.15 demonstrates the overhead of the localization technique to spot the position of the infected router in the NoC for different distances between the infected node (HT) and the destination node (D') that initiated the scouting process. The graph shows the distribution of the localization time overhead at each possible single distance between the detector, D', and the tampering, HT, nodes. It is shown that the localization time overhead

is linearly proportional to the number of hops between the start of the scouting process and the infected node. It takes 90 to 730 clock cycles to localize the tampering router in an 8×8 NoC. Therefore, the HT can be located if it is active for up to 730 clock cycles, excluding the travel time of the tampered packet to the detector node.

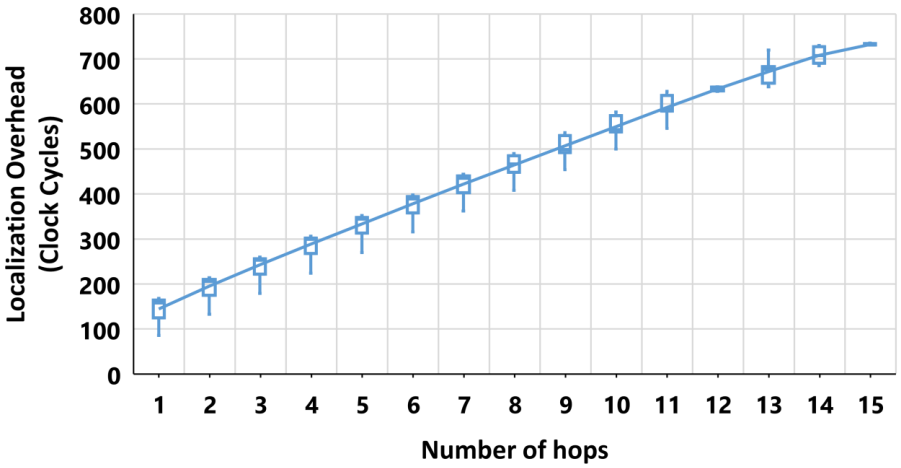


Figure 6.15: HT Localization overhead vs number of hops.

Performance Degradation

In this experiment, the NoC was infected with a packet-tampering router that alters the packet on its way to the recipient node. The authentication module of the ASR-based NoC detects the attack and initiates the searching technique to spot the infected node. The infected node was isolated and packets detoured around it. The routing technique of the nodes that surround the malicious router were updated and reconfigured to avoid routing the packets through it. Additionally, the routing violation detection technique was updated for each surrounding node, since they do not follow X-Y routing technique. The throughput of the infected NoC was measured and normalized to the HT-free ASR-based NoC model.

Figure 6.16 demonstrates the normalized NoC throughput of an 8×8 NoC size for a Random traffic pattern. The graph shows that there has been a slight decrease in the performance of the NoC when it avoids an infected node at higher injection rate, 17% degradation at 10% injection rate. However, the packets maintain their integrity, secrecy, and authenticity.

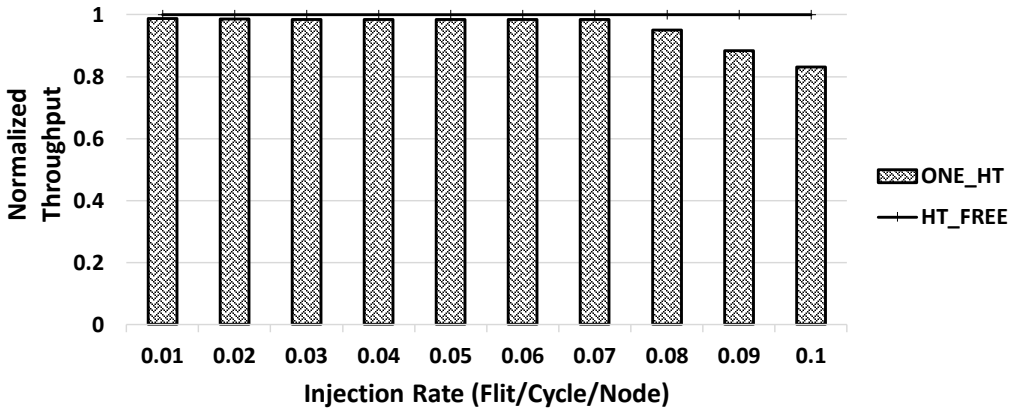


Figure 6.16: Normalized performance of the NoC model under a HT attack.

Discussion

A comparison of the presented NoC model and the packet leak detection of the Fort-NoC proposed in [121] is demonstrated in Table 6.3. The main feature of the Fort-NoC is to detect the packet leak in the NoC due to a HT hidden in the router. In the Fort-NoC model, packets are scrambled by bitwise-XOR with a static key. Though Fort-NoC offers less area overhead, it exhibits weakness in the encryption and authentication method, leaving a backdoor for the attacker to comprehend the packet certification. The packet certificate is fixed for the destination and can be easily detected. In this presented design, AES is used

for encrypting and authenticating the packets, which makes it very hard for a crypt-analysis attack to uncover the data or gain unauthorized access.

Table 6.3: Comparison between the Fort-NoC [121] and the ASR-based NoC model.

	Fort-NoC [121]	**ASR-based Model**
Packet Leak	✓	✓
Security Level	Low (bitwise-XOR)	High (AES)
Misrouting Detection	×	✓
HT Localization	×	✓
Area	9.91%	24.21%

6.4 Summary

In this chapter, an authentic and secure NoC model was presented. It provides data confidentiality, integrity, and authenticity. In this model, each node is attached to an authentic and secure module, Authentication Module (AM), to generate authenticated and encrypted packets using the AES encryption standard, before they are injected into the NoC. At the receiver side, the received packets were verified and validated to check their integrity and authenticity. Once a tampered packet is detected at the receiving node, it generates a scouting packet to localize the source of the tampering router. Experiments were conducted on an 8×8 NoC and showed that the technique maintains the packet integrity and can localize the tampering node within 90 to 730 clock cycles, based on the number of hops a HT is located from the start of the searching process. The presented secure NoC model has a 24.21% area overhead and the injected packets in the NoC carry 75% of the information data compared to the baseline router.

CHAPTER 7

CONCLUSIONS

7.1 Conclusions

The Network-on-Chip (NoC) provides a high performance and scalable interconnection between hundreds of processing elements. It decouples the computation sub-system from the communication one. This has made the NoC widely used in contemporary and future Multiprocessor systems-on-chip (MPSoC). Third Party Intellectual Properties (3PIPs) have been adopted in many design due to time-to-market and to reduction in the overall design cost.

However, the outsourcing of designs and the concern about 3PIPs have raised the security concerns about the possibility of placing Hardware Trojan (HT) circuits in the NoC to compromise data manipulation, steal sensitive information, or degrade the system. It is too difficult to pinpoint physical alterations in complex circuits of 3PIPs within a large state-space, since HTs are usually in an idle state and become active upon launching an attack. This leads HTs to escape testing cases and appear in final products. Therefore, run-time detection solutions are necessary to provide a practical defense for the system to mitigate the HT effect, since post-silicon testing, in most cases, is usually incapable of detecting such malicious circuits.

Chapter 3 provided a literature research and revealed several studies on the different types of security attacks on NoC-based systems and on the applications running on the IP-

cores of the systems. Several studies focused on the impact of a HT in the NoC performance and the system security.

The goal of this work was to provide a resilient and secure NoC model that mitigates the HT effects. The research contributions provided in this dissertation considered two HT-based attacks:

- A packet-dropping attack, which is known as Black Hole Router (BHR) attack, where packets are dropped from the NoC once they pass through a BHR, without further information.

- A packet-tampering attack, where packets experience alteration on their way to the destination due to a hidden HT in the NoC router.

Chapter 4 presented an analysis for the BHR attack and its effect on the NoC, and examined the influence of the number of the infected nodes, along with their distribution, on the potency of the attack. Experiments were designed and were run on a cycle-accurate simulator for an 8×8 NoC which showed that the packet loss rate varied between 5% to 33.8% based on the number of the BHRs and their locations in the NoC. This studied showed that by considering some factors in the BHR design, a violent attack can occur with less effort.

Chapter 5 developed a novel mechanism to detect the BHR and prevent its attack by detouring around the infected node. The BHR detection was based on an acknowledgement (ACK) from nodes two hops away to confirm successful forwarding of the packets. In order to achieve a secure ACK, the packet confirmation, i.e. ACK, was signed by bitwise-XORing the packet with a one time pad number shared between the two paired penultimate routers. The ACK signature handler is assumed to be executed in a secure module in the network interface of the NoC model. The experiments on a simulated 8×8

NoC showed that the detection protocol identifies the BHR attack in runtime and avoids it with an overhead of 26.90% in resource utilization and 21.31% in performance when it is compared to a baseline NoC model for the same parameters.

The mechanism was further developed to provide an energy-efficient technique that detected the BHR with negligible overhead in the NoC throughput and power consumption based on an end-to-end (e2e) technique. The energy-efficient scheme was able to detect data-packets or ACK-packets dropping attacks and could identify the BHR location with minimal power overhead. The experiments, on an 8×8 NoC, showed that this expanded mechanism was more efficient than the traditional always-on hop-to-hop (h2h) confirmation. The energy-efficient mechanism showed 2% and 1% overheads in the energy consumption and the NoC throughput, respectively, compared to the baseline NoC model.

Chapter 6 focused on packet-tampering attacks due to a HT. The novel strategy maintained the packets' integrity and authenticity. Packets are authenticated and encrypted at the source node using the Advanced Standard Encryption (AES-128) [152]. The received packets are verified and validated at the destination node to check their integrity and authenticity. With Regard to locating the source of the tampering attack, in the case of packet-alteration, the destination node generates a special packet, the Scouting Packet (SP), to check the correctness of path of the packet between the source and the destination nodes to locate the HT router. Experiments showed that this technique can locate the tampering router within 90 to 730 clock cycles, based on the hop distance between the beginning of the localization process and the HT.

7.2 Future Work

This section discusses some potential future research directions. The future work concerns deeper analysis of potential attacks and particular mechanisms. There are some ideas that extend the Black Hole Router (BHR) attack and the tampering attacks.

7.2.1 Colluding Routers

The research documented in this dissertation concentrated only on a BHR attack where all packets passing through it are discarded from the NoC. The proposed detection technique is based on generating an acknowledgement (ACK) and sending it back to the penultimate router, thus assuring that the intermediate router has forwarded the packet. In the colluding routers attack, two adjacent routers may collude with each other to drop packets that are passing through them, where the first rogue router sends a packet to the collaborating one, which confirms its arrival while discarding the packet. The first rogue router forwards the ACK to the intended node. In this scenario, it will not be possible for this dissertation's technique to detect such BHR due to the collusion of two adjacent routers. This can be solved by extending the acknowledgement across the path of the packet. More precisely, the ACK should be received from each single router in the path of the packet to the source node. Although, the NoC will be more congested, this technique will identify the colluding routers and detect the source of BHR attack.

7.2.2 Gray Hole Router

Another interesting problem may be raised in the detection of packet-dropping attacks. The NoC may experience a special case of a packet-dropping attack by selectively discarding a subset of packets and passing the others. This is known as a Gray Hole Router (GHR).

In the current energy-efficient detection, a GHR could discard data packets during normal routing, however, it allows packets to be forwarded when the detection technique is activated. This attack will be very hard to detect. Although, in this case, packets eventually arrive to their destinations, the NoC-based system experiences degradation in the NoC bandwidth.Therefore, a GHR needs more investigation in future work.

7.2.3 Replay Attack Detection

In the Authentic and Secure NoC model, packets are encrypted by the AES-128 [152] before being injected in the network. At the receiver side, the packets are decrypted and checked for their correctness. One possible attack which can be applied is a replay attack, where a valid packet is stored by a malicious router for an attacking purpose. In this scenario, the malicious router could retransmit this valid packet after a delay to deplete the NoC bandwidth, degrade the NoC performance, or cause an undesired action from the destination node. In the presented NoC model, since the packet has not been tampered with, it will successfully pass the authentication check. However, it was not intended to be sent or was repeatedly sent, leading to an unpleasant effect. In order to detect such an attack, the packet should be time-stamped before being injected into the NoC. Therefore, at the destination side, the time-stamp of the packet is checked for assurance of delivery within a certain period of time, otherwise, the packet is more likely to have experienced a replay attack. This scheme will be explored in the future design.

7.2.4 Routing Violation for Adaptive Routing

The presented Authentic and Secure NoC model can detect packet-tampering attacks and locate the HT causing them. Packets are checked at each input port of the router for routing violations for a deterministic X-Y routing protocol. Future work could focus on adaptive

routing protocols and how a routing violation can be distinguished from the normal adaptive routing.

Bibliography

[1] G. E. Moore, "Cramming more components onto integrated circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, 1998.

[2] TSMC. (2019) Tsmc ramping up 7nm chip production. [Online]. Available: https://www.digitimes.com/news/a20180622PD204.html

[3] A. Shilov. (2019) Tsmc: 5nm on track for q2 2020 hvm, will ramp faster than 7nm. [Online]. Available: https://www.anandtech.com/show/15016/tsmc-5nm-on-track-for-q2-2020-hvm-will-ramp-faster-than-7nm

[4] A. Shilov. (2019) Samsung's aggressive euv plans: 6nm production in h2, 5nm & 4nm on track. [Online]. Available: https://www.anandtech.com/show/14695/samsungs-aggressive-euv-plans-6nm-production-in-h2-5nm-4nm-on-track

[5] R. H. Dennard *et al.*, "Design of ion-implanted mosfet's with very small physical dimensions," *IEEE Journal of Solid-State Circuits*, vol. 9, no. 5, pp. 256–268, 1974.

[6] H. Esmaeilzadeh *et al.*, "Dark silicon and the end of multicore scaling," in *Proceedings of the 38th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2011, pp. 365–376.

[7] S. Borkar, "Thousand core chips: a technology perspective," in *Proceedings of the 44th annual Design Automation Conference*. ACM, 2007, pp. 746–749.

[8] K. Olukotun, L. Hammond, and J. Laudon, "Chip multiprocessor architecture: techniques to improve throughput and latency," *Synthesis Lectures on Computer Architecture*, vol. 2, no. 1, pp. 1–145, 2007.

[9] P. Gratz *et al.*, "Implementation and evaluation of on-chip network architectures," in *Proceedings of the International Conference on Computer Design (ICCD)*. IEEE, 2006, pp. 477–484.

[10] R. Kumar, V. Zyuban, and D. M. Tullsen, "Interconnections in multi-core architectures: Understanding mechanisms, overheads and scaling," in *Proceedings of the 32nd International Symposium on Computer Architecture (ISCA)*. IEEE, 2005, pp. 408–419.

[11] D. Sanchez, G. Michelogiannakis, and C. Kozyrakis, "An analysis of on-chip interconnection networks for large-scale chip multiprocessors," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 7, no. 1, pp. 1–28, 2010.

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

[13] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Computing Surveys (CSUR)*, vol. 38, no. 1, p. 1, 2006.

[14] S. R. Vangal *et al.*, "An 80-tile sub-100-w teraflops processor in 65-nm cmos," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 29–41, 2008.

[15] Y. Hoskote *et al.*, "A 5-ghz mesh interconnect for a teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, 2007.

[16] J. Howard *et al.*, "A 48-core ia-32 processor in 45 nm cmos using on-die message-passing and dvfs for performance and power scaling," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 173–183, 2011.

[17] A. A. Chien, "A cost and speed model for k-ary n-cube wormhole routers," *Urbana*, vol. 51, p. 61801, 1993.

[18] C. Demerjian, "A look at a 100-core tilera core gx. efficiency," *Microprocessor and Servers*, 2009.

[19] Xilinx Inc., *Versal Architecture and Product Data Sheet: Overview*, December, 2019.

[20] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proceedings of the Design Automation Conference*. IEEE, 2001, pp. 684–689.

[21] S. Pasricha and N. Dutt, *On-chip communication architectures: system on chip interconnect*. Morgan Kaufmann, 2010.

[22] M. Abramovici and P. Bradley, "Integrated circuit security: new threats and solutions," in *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*. ACM, 2009, p. 55.

[23] G. Linden and D. Somaya, "System-on-a-chip integration in the semiconductor industry: industry structure and firm strategies," *Industrial and Corporate Change*, vol. 12, no. 3, pp. 545–576, 2003.

[24] A. Waksman, M. Suozzo, and S. Sethumadhavan, "Fanci: identification of stealthy malicious logic using boolean functional analysis," in *Proceedings of the ACM SIGSAC Conference on Computer & Communications Security*. ACM, 2013, pp. 697–708.

[25] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE design & Test of Computers*, vol. 27, no. 1, 2010.

[26] J.-P. Diguet *et al.*, "Noc-centric security of reconfigurable soc," in *Proceedings of the first International Symposium on Networks-on-Chip (NOCS)*. IEEE, 2007, pp. 223–232.

[27] H. M. Wassel *et al.*, "Networks on chip with provable security properties," *IEEE Micro*, vol. 34, no. 3, pp. 57–68, 2014.

[28] Y. Jin, N. Kupp, and Y. Makris, "Experiences in hardware trojan design and implementation," in *Proceedings of the International Workshop on Hardware-Oriented Security and Trust (HOST)*. IEEE, 2009, pp. 50–57.

[29] X. Zhang and M. Tehranipoor, "Case study: Detecting hardware trojans in third-party digital ip cores," in *Proceedings of the International Workshop on Hardware-Oriented Security and Trust (HOST)*. IEEE, 2011, pp. 67–70.

[30] S. Adee, "The hunt for the kill switch," *IEEE Spectrum*, vol. 45, no. 5, pp. 34–39, 2008.

[31] S. Skorobogatov and C. Woods, "Breakthrough silicon scanning discovers backdoor in military chip," in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2012, pp. 23–40.

[32] B. Shakya *et al.*, "Benchmarking of hardware trojans and maliciously affected circuits," *Journal of Hardware and Systems Security*, pp. 1–18, 2017.

[33] E. Bolotin *et al.*, "Qnoc: Qos architecture and design process for network on chip," *Journal of Systems Architecture*, vol. 50, no. 2, pp. 105–128, 2004.

[34] A. Jantsch, H. Tenhunen *et al.*, *Networks on chip*. Springer, 2003, vol. 396.

[35] L. B. Daoud, M. E.-S. Ragab, and V. Goulart, "Faster processor allocation algorithms for mesh-connected cmps," in *Digital System Design (DSD), 2011 14th Euromicro Conference on*. IEEE, 2011, pp. 805–808.

[36] L. Daoud, M. E.-S. Ragab, and V. Goulart, "Processor allocation algorithm based on Frame Combing with Memorization for 2D mesh CMPs," in *Proceedins of the Third Latin American Symposium on Circuits and Systems (LASCAS)*. IEEE, 2012, pp. 1–4.

[37] L. Daoud, "Efficient processor allocators for mesh-connected chip-multiprocessors," Master's thesis, Egypt-Japan University of Science and Technology (E-JUST), 02 2012.

[38] L. Daoud and V. Goulart, "High performance bitwise or based submesh allocation for 2d mesh-connected cmps," in *Proceedings of the Euromicro Conference on Digital System Design (DSD)*. IEEE, 2013, pp. 73–77.

[39] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO white paper*, vol. 1, no. 2011, pp. 1–11, 2011.

[40] D. Atienza *et al.*, "Network-on-chip design and synthesis outlook," *Integration*, vol. 41, no. 3, pp. 340–359, 2008.

[41] A. Radulescu *et al.*, "An efficient on-chip ni offering guaranteed services, shared-memory abstraction, and flexible network configuration," *IEEE Transactions on computer-aided design of integrated circuits and systems*, vol. 24, no. 1, pp. 4–17, 2005.

[42] I. Loi, F. Angiolini, and L. Benini, "Synthesis of low-overhead configurable source routing tables for network interfaces," in *Proceedings of the Conference on Design, Automation and Test in Europe*. European Design and Automation Association, 2009, pp. 262–267.

[43] G. De Micheli and L. Benini, *Networks on chips: technology and tools*. Academic Press, 2006.

[44] H. Zimmermann, "Osi reference model–the iso model of architecture for open systems interconnection," *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 425–432, 1980.

[45] M. support. (2017, Aug.) The osi model's seven layers defined and functions explained. [Online]. Available: https://support.microsoft.com/en-us/help/103884/the-osi-model-s-seven-layers-defined-and-functions-explained

[46] M. Dehyadgari *et al.*, "A new protocol stack model for network on chip," in *Proceedings of the Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*. IEEE, 2006, pp. 440–441.

[47] D. Bertozzi and L. Benini, "Xpipes: A network-on-chip architecture for gigascale systems-on-chip," *IEEE circuits and Systems Magazine*, vol. 4, no. 2, pp. 18–31, 2004.

[48] P. P. Pande *et al.*, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 1025–1040, 2005.

[49] S. Kumar *et al.*, "A network on chip architecture and design methodology," in *Proceedings of the Computer Society Annual Symposium on VLSI*. IEEE, 2002, pp. 117–124.

[50] W. J. Dally and C. L. Seitz, "The torus routing chip," *Distributed Computing*, vol. 1, no. 4, pp. 187–196, 1986.

[51] P. P. Pande *et al.*, "Design of a switch for network on chip applications," in *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*, vol. 5. IEEE, 2003, pp. 217–220.

[52] A. Adriahantenaina *et al.*, "Spin: a scalable, packet switched, on-chip micro-network," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*. IEEE, 2003, pp. 70–73.

[53] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Proceedings of the Design, Automation and Test in Europe Conference*. ACM, 2000, pp. 250–256.

[54] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection networks: an engineering approach*. Morgan Kaufmann, 2003.

[55] V. Rantala *et al.*, *Network on chip routing algorithms*. Turku Centre for Computer Science, 2006.

[56] E. Nilsson *et al.*, "Load distribution with the proximity congestion awareness in a network on chip," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*. IEEE Computer Society, 2003, pp. 1126–1127.

[57] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers*, vol. C-36, no. 5, pp. 547–553, 1988.

[58] W. J. Dally, "Virtual-channel flow control," *IEEE Transactions on Parallel and Distributed systems*, vol. 3, no. 2, pp. 194–205, 1992.

[59] J. Hu and R. Marculescu, "Exploiting the routing flexibility for energy/performance aware mapping of regular noc architectures," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*. IEEE, 2003, pp. 688–693.

[60] M. Dehyadgari *et al.*, "Evaluation of pseudo adaptive xy routing using an object oriented model for noc," in *Proceedings of the 17th International Conference on Microelectronics (ICM)*. IEEE, 2005, pp. 204–208.

[61] C. Bobda *et al.*, "Dynoc: A dynamic infrastructure for communication in dynamically reconfugurable devices," in *Proceedings of the International Conference on Field Programmable Logic and Applications*. IEEE, 2005, pp. 153–158.

[62] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," *ACM SIGARCH Computer Architecture News*, vol. 20, no. 2, pp. 278–287, 1992.

[63] K. Kim *et al.*, "An arbitration look-ahead scheme for reducing end-to-end latency in networks on chip," in *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2005, pp. 2357–2360.

[64] W. J. Dally and B. P. Towles, *Principles and practices of interconnection networks*. Elsevier, 2004.

[65] T. Bartic *et al.*, "Topology adaptive network-on-chip design and implementation," *IEE Proceedings-Computers and Digital Techniques*, vol. 152, no. 4, pp. 467–472, 2005.

[66] N. Bansal *et al.*, "Online oblivious routing," in *Proceedings of the 15th annual ACM symposium on Parallel Algorithms and Architectures*. ACM, 2003, pp. 44–49.

[67] M. Pirretti *et al.*, "Fault tolerant algorithms for network-on-chip interconnect," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*. IEEE, 2004, pp. 46–51.

[68] J. Kim *et al.*, "A low latency router supporting adaptivity for on-chip interconnects," in *Proceedings of the 42nd annual Design Automation Conference*. ACM, 2005, pp. 559–564.

[69] G.-M. Chiu, "The odd-even turn model for adaptive routing," *IEEE Transactions on parallel and distributed systems*, vol. 11, no. 7, pp. 729–738, 2000.

[70] J. Hu and R. Marculescu, "Dyad: smart routing for networks-on-chip," in *Proceedings of the 41st annual Design Automation Conference*. ACM, 2004, pp. 260–263.

[71] M. Majer *et al.*, "Packet routing in dynamically changing networks on chip," in *Proceedings of the 19th Parallel and Distributed Processing Symposium*. IEEE, 2005, pp. 8–pp.

[72] U. Feige and P. Raghavan, "Exact analysis of hot-potato routing," in *Proceedings the 33rd Annual Symposium on Foundations of Computer Science*. IEEE, 1992, pp. 553–562.

[73] S. Ravi *et al.*, "Security in embedded systems: Design challenges," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 3, no. 3, pp. 461–491, 2004.

[74] S. Ravi, A. Raghunathan, and S. Chakradhar, "Tamper resistance mechanisms for secure embedded systems," in *Proceedins of the 17th International Conference on VLSI Design*. IEEE, 2004, pp. 605–611.

[75] E. Chien and P. Ször, "Blended attacks exploits, vulnerabilities and buffer-overflow techniques in computer viruses," *Virus*, vol. 1, 2002.

[76] P. Kocher *et al.*, "Security as a new dimension in embedded system design," in *Proceedings of the 41st annual Design Automation Conference*. ACM, 2004, pp. 753–760.

[77] S. Bhunia *et al.*, "Protection against hardware trojan attacks: Towards a comprehensive solution," *IEEE Design & Test*, vol. 30, no. 3, pp. 6–17, 2013.

[78] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in cryptology—CRYPTO'99*. Springer, 1999, pp. 789–789.

[79] P. Kocher, "Timing attacks on implementations of die-hellman, rsa, dss and other systems," *Advances in Cryptology-Crypto*, vol. 1109, pp. 104–113, 1996.

[80] J.-F. Dhem *et al.*, "A practical implementation of the timing attack," in *International Conference on Smart Card Research and Advanced Applications*. Springer, 1998, pp. 167–182.

[81] J.-J. Quisquater and D. Samyde, "Electromagnetic analysis (ema): Measures and counter-measures for smart cards," *Smart Card Programming and Security*, pp. 200–210, 2001.

[82] E. B. Eichelberger and T. W. Williams, "A logic design structure for lsi testability," in *Proceedings of the 14th design automation conference*. IEEE, 1977, pp. 462–468.

[83] B. Yang, K. Wu, and R. Karri, "Scan based side channel attack on dedicated hardware implementations of data encryption standard," in *Proceedings of the International Test Conference (ITC)*. IEEE, 2004, pp. 339–344.

[84] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing ic piracy using reconfigurable logic barriers," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 66–75, 2010.

[85] R. S. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware trojan: Threats and emerging solutions," in *Proceedings of the International Workshop on High Level Design Validation and Test (HLDVT)*. IEEE, 2009, pp. 166–171.

[86] S. T. King *et al.*, "Designing and implementing malicious hardware." *Proceedings of the 1st USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, vol. 8, pp. 1–8, 2008.

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

[88] R. Karri *et al.*, "Trustworthy hardware: Identifying and classifying hardware trojans," *Computer*, vol. 43, no. 10, pp. 39–46, 2010.

[89] H. Liu, H. Luo, and L. Wang, "Design of hardware trojan horse based on counter," in *Proceedings of the International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE)*. IEEE, 2011, pp. 1007–1009.

[90] X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting malicious inclusions in secure hardware: Challenges and solutions," in *Proceedings of the International Workshop on Hardware-Oriented Security and Trust (HOST)*. IEEE, 2008, pp. 15–19.

[91] S. Bhunia *et al.*, "Hardware trojan attacks: threat analysis and countermeasures," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229–1247, 2014.

[92] F. Wolff *et al.*, "Towards trojan-free trusted ics: Problem analysis and detection scheme," in *Proceedings of the conference on Design, automation and test in Europe*. ACM, 2008, pp. 1362–1365.

[93] S. Narasimhan *et al.*, "Improving ic security against trojan attacks through integration of security monitors," *IEEE Design & Test of Computers*, vol. 29, no. 5, pp. 37–46, 2012.

[94] Y. Jin and Y. Makris, "Hardware trojan detection using path delay fingerprint," in *Proceedings of the International Workshop on Hardware-Oriented Security and Trust (HOST)*. IEEE, 2008, pp. 51–57.

[95] J. Rajendran, O. Sinanoglu, and R. Karri, "Regaining trust in vlsi design: Design-for-trust techniques," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1266–1282, 2014.

[96] L. Daoud, "Secure network-on-chip architectures for mpsoc: Overview and challenges," in *Proceedings of the 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, Aug 2018, pp. 542–543.

[97] S. Evain and J.-P. Diguet, "From noc security analysis to design solutions," in *Proceedings of the Workshop on Signal Processing Systems Design and Implementation*. IEEE, 2005, pp. 166–171.

[98] A. Avizienis *et al.*, "Basic concepts and taxonomy of dependable and secure computing," *IEEE transactions on dependable and secure computing*, vol. 1, no. 1, pp. 11–33, 2004.

[99] L. Fiorin *et al.*, "Secure memory accesses on networks-on-chip," *IEEE Transactions on Computers*, vol. 57, no. 9, pp. 1216–1229, 2008.

[100] S. Baron, M. S. Wangham, and C. A. Zeferino, "Security mechanisms to improve the availability of a network-on-chip," in *Proceedings of the 20th International Conference on Electronics, Circuits, and Systems (ICECS)*. IEEE, 2013, pp. 609–612.

[101] L. Daoud and N. Rafla, "Analysis of black hole router attack in network-on-chip," in *Proceedings of the 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2019, pp. 69–72.

[102] L. Daoud and N. Rafla, "Detection and prevention protocol for black hole attack in network-on-chip," in *Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip*. ACM, 2019, p. 22.

[103] J. Coburn *et al.*, "Seca: security-enhanced communication architecture," in *Proceedings of the 2005 international conference on Compilers, architectures and synthesis for embedded systems*. ACM, 2005, pp. 78–89.

[104] L. Fiorin *et al.*, "A data protection unit for noc-based architectures," in *Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis*. ACM, 2007, pp. 167–172.

[105] P. Cotret *et al.*, "Distributed security for communications and memories in a multiprocessor architecture," in *Proceedings of the International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW)*. IEEE, 2011, pp. 326–329.

[106] M. D. Grammatikakis *et al.*, "Security in mpsocs: a noc firewall and an evaluation framework," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 8, pp. 1344–1357, 2015.

[107] K. Sajeesh and H. K. Kapoor, "An authenticated encryption based security framework for noc architectures," in *Proceedigns of the International Symposium on Electronic System Design (ISED)*. IEEE, 2011, pp. 134–139.

[108] H. K. Kapoor *et al.*, "A security framework for noc using authenticated encryption and session keys," *Circuits, Systems, and Signal Processing*, vol. 32, no. 6, pp. 2605–2622, 2013.

[109] J. Sepulveda *et al.*, "Dynamic noc-based architecture for mpsoc security implementation," in *Proceedings of the 24th symposium on Integrated circuits and systems design*. ACM, 2011, pp. 197–202.

[110] J. Sepulveda *et al.*, "Hierarchical noc-based security for mp-soc dynamic protection," in *Proceedings of the 3rd Latin American Symposium on Circuits and Systems (LASCAS)*. IEEE, 2012, pp. 1–4.

[111] R. Fernandes *et al.*, "A security aware routing approach for noc-based mpsocs," in *Integrated Circuits and Systems Design (SBCCI), 2016 29th Symposium on*. IEEE, 2016, pp. 1–6.

[112] J. Sepulveda *et al.*, "Elastic security zones for noc-based 3d-mpsocs," in *Proceedings of the 21st International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2014, pp. 506–509.

[113] J. Sepúlveda, D. Flórez, and G. Gogniat, "Reconfigurable security architecture for disrupted protection zones in noc-based mpsocs," in *Proceedings of the 10th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*. IEEE, 2015, pp. 1–8.

[114] Z. Wang and R. B. Lee, "Covert and side channels due to processor architecture," in *Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC)*. IEEE, 2006, pp. 473–482.

[115] R. Stefan and K. Goossens, "Enhancing the security of time-division-multiplexing networks-on-chip through the use of multipath routing," in *Proceedings of the 4th International Workshop on Network on Chip Architectures*. ACM, 2011, pp. 57–62.

[116] C. Reinbrecht *et al.*, "Side channel attack on noc-based mpsocs are practical: Noc prime+ probe attack," in *Proceedings of the 29th Symposium on Integrated Circuits and Systems Design (SBCCI)*. IEEE, 2016, pp. 1–6.

[117] L. S. Indrusiak, J. Harbin, and M. J. Sepulveda, "Side-channel attack resilience through route randomisation in secure real-time networks-on-chip," *arXiv preprint arXiv:1607.03450*, 2016.

[118] C. Reinbrecht *et al.*, "Timing attack on noc-based systems: Prime+ probe attack and noc-based protection," *Microprocessors and Microsystems*, 2017.

[119] C. H. Gebotys and R. J. Gebotys, "A framework for security on noc technologies," in *Proceedings of the Annual IEEE Computer Society Symposium on VLSI*. IEEE, 2003, pp. 113–117.

[120] C. H. Gebotys and Y. Zhang, "Security wrappers and power analysis for soc technologies," in *Proceedings of the 1st IEEE/ACM/IFIP International Conference on Hardware/software Codesign and System Synthesis*. ACM, 2003, pp. 162–167.

[121] 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*. ACM, 2014, pp. 1–6.

[122] M. Hussain and H. Guo, "Packet leak detection on hardware-trojan infected nocs for mpsoc systems," in *Proceedings of the International Conference on Cryptography, Security and Privacy*. ACM, 2017, pp. 85–90.

[123] M. Hussain, and H. Guo, "A bandwidth-aware authentication scheme for packet-integrity attack detection on trojan infected noc," in *Proceedings of the International Conference on Very Large Scale Integration (VLSI-SoC)*. IEEE, 2018, pp. 201–206.

[124] M. Hussain *et al.*, "Eetd: An energy efficient design for runtime hardware trojan detection in untrusted network-on-chip," in *Proceedings of the Annual IEEE Computer Society Symposium on VLSI (ISVLSI)*. IEEE, 2018, pp. 345–350.

[125] A. K. Biswas, S. Nandy, and R. Narayan, "Router attack toward noc-enabled mpsoc and monitoring countermeasures against such threat," *Circuits, Systems, and Signal Processing*, vol. 34, no. 10, pp. 3241–3290, 2015.

[126] A. K. Biswas, S. K. Nandy, and R. Narayan, "Network-on-chip router attacks and their prevention in mp-socs with multiple trusted execution environments," in *Proceedings of the International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. IEEE, 2015, pp. 1–6.

[127] Q. Yu and J. Frey, "Exploiting error control approaches for hardware trojans on network-on-chip links," in *Proceedings of the International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. IEEE, 2013, pp. 266–271.

[128] J. Frey and Q. Yu, "Exploiting state obfuscation to detect hardware trojans in noc network interfaces," in *Proceedings of the 58th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2015, pp. 1–4.

[129] 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, no. 10, pp. 1493–1502, 2009.

[130] J. Rajendran *et al.*, "Logic encryption: A fault analysis perspective," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2012, pp. 953–958.

[131] J. Frey and Q. Yu, "A hardened network-on-chip design using runtime hardware trojan mitigation methods," *Integration*, vol. 56, pp. 15–31, 2017.

[132] T. Boraten and A. K. Kodi, "Mitigation of denial of service attack with hardware trojans in noc architectures," in *Proceddings of the International Parallel and Distributed Processing Symposium*. IEEE, 2016, pp. 1091–1100.

[133] T. Boraten and A. Kodi, "Mitigation of hardware trojan based denial-of-service attack for secure nocs," *Journal of Parallel and Distributed Computing*, vol. 111, pp. 24–38, 2018.

[134] H. M. Wassel *et al.*, "Surfnoc: a low latency and provably non-interfering approach to secure networks-on-chip," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3, pp. 583–594, 2013.

[135] D. Fang *et al.*, "Robustness analysis of mesh-based network-on-chip architecture under flooding-based denial of service attacks," in *Proceedings of the 8th International Conference on Networking, Architecture and Storage (NAS)*. IEEE, 2013, pp. 178–186.

[136] T. Boraten and A. K. Kodi, "Packet security with path sensitization for nocs," in *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*. EDA Consortium, 2016, pp. 1136–1139.

[137] R. Cramer *et al.*, "Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors," in *Proceedins of the annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2008, pp. 471–488.

[138] R. JS *et al.*, "Runtime detection of a bandwidth denial attack from a rogue network-on-chip," in *Proceedings of the 9th International Symposium on Networks-on-Chip*. ACM, 2015, p. 8.

[139] R. JayashankaraShridevi *et al.*, "Security measures against a rogue network-on-chip," *Journal of Hardware and Systems Security*, pp. 1–15, 2017.

[140] L. Zhang *et al.*, "Effectiveness of ht-assisted sinkhole and blackhole denial of service attacks targeting mesh networks-on-chip," *Journal of Systems Architecture*, vol. 89, pp. 84–94, 2018.

[141] Luka Daoud and Nader Rafla, "Routing aware and runtime detection for infected network-on-chip routers," in *Proceedings of the 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2018, pp. 775–778.

[142] L. Daoud, D. Zydek, and H. Selvaraj, "A survey of high level synthesis languages, tools, and compilers for reconfigurable high performance computing," in *Advances in Systems Science*. Springer, 2014, pp. 483–492.

[143] Xilinx Inc., *Vivado Design Suite User Guide: High-Level Synthesis*, December, 2018.

[144] V. Catania *et al.*, "Cycle-accurate network on chip simulation with noxim," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 27, no. 1, p. 4, 2016.

[145] K.-H. Chen and G.-M. Chiu, "Fault-tolerant routing algorithm for meshes without using virtual channels," *J. Inf. Sci. Eng.*, vol. 14, no. 4, pp. 765–783, 1998.

[146] Z. Zhang, A. Greiner, and S. Taktak, "A reconfigurable routing algorithm for a fault-tolerant 2d-mesh network-on-chip," in *Proceedings of the 45th annual Design Automation Conference*. ACM, 2008, pp. 441–446.

[147] Y. Fukushima, M. Fukushi, and S. Horiguchi, "Fault-tolerant routing algorithm for network on chip without virtual channels," in *Proceedings of the 24th International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*. IEEE, 2009, pp. 313–321.

[148] N. FIPS, "198: The keyed-hash message authentication code (hmac)," *National Institute of Standards and Technology, Federal Information Processing Standards*, p. 29, 2002.

[149] N. FIPS, "Sha-3 standard: Permutation-based hash and extendable-output functions, draft fips 202," *National Institute of Standards and Technology, Federal Information Processing Standards*, 2014.

[150] H. S. Jacinto, L. Daoud, and N. Rafla, "High level synthesis using vivado hls for optimizations of sha-3," in *Proceedings of the 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2017, pp. 563–566.

[151] M. K. Latif *et al.*, "Optimization of a quantum-secure sponge-based hash message authentication protocol," in *Proceedings of the 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2018, pp. 984–987.

[152] N. F. Pub, "197: Advanced encryption standard (aes)," *Federal information processing standards publication*, vol. 197, no. 441, p. 0311, 2001.

[153] P. FIPS, "46-3: Data encryption standard (des)," *National Institute of Standards and Technology*, vol. 25, no. 10, pp. 1–22, 1999.

[154] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[155] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[156] E. Rescorla, "Diffie-hellman key agreement method," *RFC 2631*, 1999.

[157] J. Sepulveda *et al.*, "Efficient security zones implementation through hierarchical group key management at noc-based mpsocs," *Microprocessors and Microsystems*, vol. 50, pp. 164–174, 2017.

[158] J. Sepúlveda, D. Flórez, and G. Gogniat, "Efficient and flexible noc-based group communication for secure mpsocs," in *Proceedings of the International Conference on ReConFigurable Computing and FPGAs (ReConFig)*. IEEE, 2015, pp. 1–6.

[159] Sepúlveda, Johanna and Flórez, Daniel and Gogniat, Guy, "Reconfigurable group-wise security architecture for noc-based mpsocs protection," in *Proceedings of the 28th Symposium on Integrated Circuits and Systems Design (SBCCI)*. IEEE, 2015, pp. 1–6.

[160] L. Daoud, F. Hussein, and N. Rafla, "Optimization of advanced encryption standard (aes) using vivado high level synthesis (hls)," in *Proceedings of the 34th International Conference on Computers and Their Applications*, ser. EPiC Series in Computing, vol. 58. EasyChair, 2019, pp. 36–44. [Online]. Available: https://easychair.org/publications/paper/b2ZJ

[161] L. Daoud, F. Hussein, and N. Rafla, "High-level synthesis optimization of aes-128/192/256 encryption algorithms," *International Journal of Computers and Their Applications*, vol. 29, pp. 129–136, 2019.

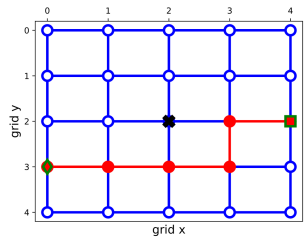
APPENDICES

APPENDIX A

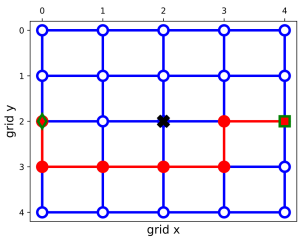
A GRAPHICAL EXAMPLE OF THE RECONFIGURABLE ROUTING TECHNIQUE

In this appendix, a graphical example is shown to demonstrate the reconfigurable routing technique for detouring around a specific node.

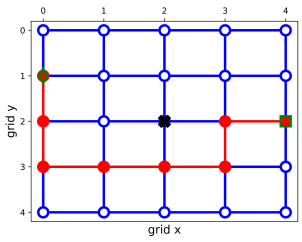
The following figures show the detouring path around a malicious node (2,2) of a packet injected from the source node (4,2) to multiple destinations for a 5×5 NoC.



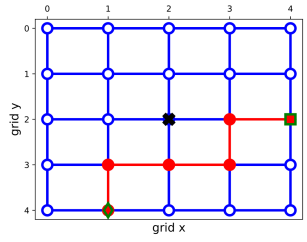
(a) A packet path to node (0,3)



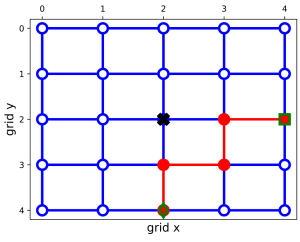
(b) A packet path to node (0,2)



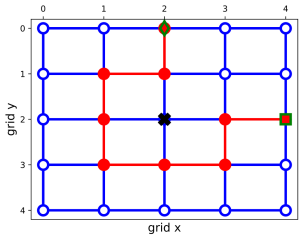
(c) A packet path to node (0,1)



(d) A packet path to node (1,4)

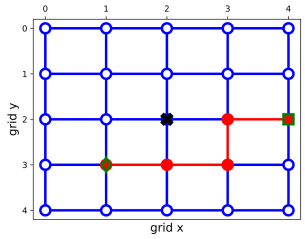


(e) A packet path to node (2,4)

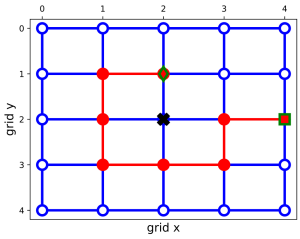


(f) A packet path to node (2,0)

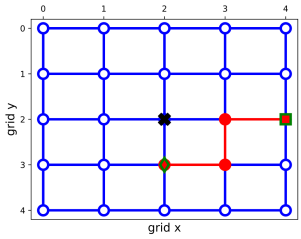
Figure A.1: A graphical path of a packet injected from node (4,2) with detours around node (2,2).



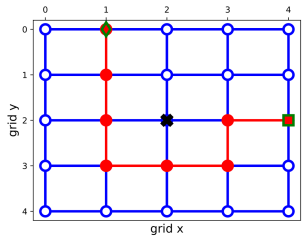
(a) A packet path to node (1,3)



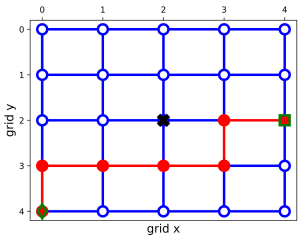
(b) A packet path to node (2,1)



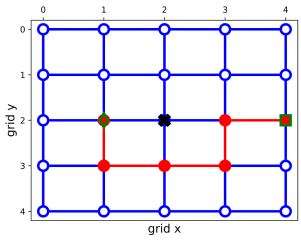
(c) A packet path to node (2,3)



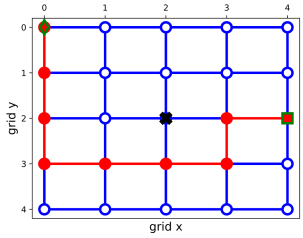
(d) A packet path to node (1,0)



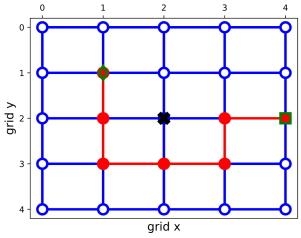
(e) A packet path to node (0,4)



(f) A packet path to node (1,2)



(g) A packet path to node (0,0)



(h) A packet path to node (1,1)

Figure A.2: A graphical path of a packet injected from node (4,2) with detours around node (2,2).

APPENDIX B

KEY EXCHANGE SCALABILITY

In order to study the number of keys required for the NoC to achieve a secure architecture, a 5×5 NoC is used as an example, shown in Figure B.1, where nodes are grouped into six groups based on the number of keys required for each node.

In Figure B.1, the nodes are numbered into groups: G1, G2, G3, G4, G5, and G6 which are named Corner, Side Corner, Side Middle, Inner Corner, Inner Middle, and Middle, respectively. These groups are described as following:

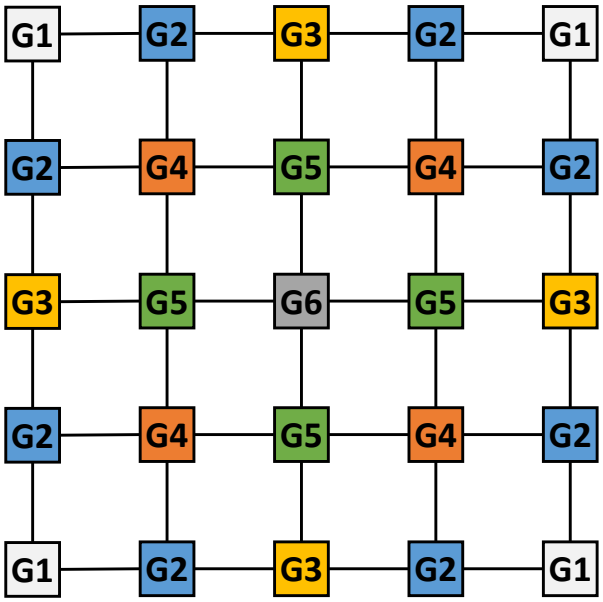


Figure B.1: A 5×5 NoC with different group of keys.

- G1, (Corner): This group contains the corner nodes of the NoC. Each node in this group contains five different keys. Three distinct keys are shared with the penultimate routers of the G3 and G4 groups. Two distinct keys are shared with the processor elements of G2 group.

- G2, (Side Corner): This group contains the nodes that are adjacent to the corner nodes of the NoC. Each node in this group contains seven different keys. Four distinct keys are shared with the penultimate routers of other G2 and G5 groups. Three distinct keys are shared with the processor elements of the G1, G3, and G4 groups.

- G3, (Side Middle): This group contains side nodes in the NoC. Each node in this group contains eight different keys. Five distinct keys are shared with the penultimate routers of the G1, G4, and G6 groups. Three distinct keys are shared with the processor elements of the G2, and G5 groups.

- G4, (Inner Corner): This group contains the inner corner nodes of the NoC. Each node in this group contains ten different keys. six distinct keys are shared with the penultimate routers of the G1, G3, and G4 groups. Four distinct keys are shared with the processor elements of the G2, and G5 groups.

- G5, (Inner Middle): This group contains the inner middle nodes of the NoC. Each node in this group contains eleven different keys. Seven distinct keys are shared with the penultimate routers of the G2 and other G5 groups. Four distinct keys are shared with the processor elements of the G3, G4, and G6 groups.

- G6, (Middle): This group contains the middle nodes of the NoC. Each node in this group contains twelve different keys. Eight distinct keys are shared with the

penultimate routers of the G3 and G4 groups. Four distinct keys are shared with the processor elements of the G5 group.

B.1 Derivation of the Total Number of Keys:

For the larger NoC size, the extra nodes will belong to G6 for middle nodes, G5 for inner middle nodes , and G3 for side middle nodes. The number of nodes in groups G1, G2, and G4 is fixed and equal to 4, 8, and 4, respectively. Therefore, the number of keys in these groups is derived as:

$$5 \times G1 + 7 \times G2 + 10 \times G4 = 5 \times 4 + 7 \times 8 + 10 \times 4 = 116$$

For a $N \times N$ NoC, the number of nodes in groups G3, G5, and G6 are equal to $4 \times$(N-4), $4 \times$(N-4), and $(N-4)^2$, respectively. Therefore, the number of keys in these groups is derived as:

$$4(N-4) \times [G3 + G5] + (N-4)^2 \times G6 = 4(N-4) \times [8 + 11] + (N-4)^2 \times 12$$

$$= 76(N-4) + 12(N-4)^2$$

Therefore, the total number of keys, Num_{keys}, equals: 116 + $76(N-4)$ + $12(N-4)^2$

$$Num_{keys} = 12N^2 - 30N + 4, \qquad \forall N \geq 5$$

Since these keys are shared between a pair of nodes, the total number of distinguished keys, $Num_{disKeys}$, is half of Num_{keys}.

$$Num_{disKeys} = (\frac{Num_{keys}}{2}) = 6N^2 - 15N + 2$$

.

Similarly, the total number of distinguished keys for NoC sizes equal to 2×2, 3×3, and 4×4 was derived and is equal to 6, 26, and 58, respectively.

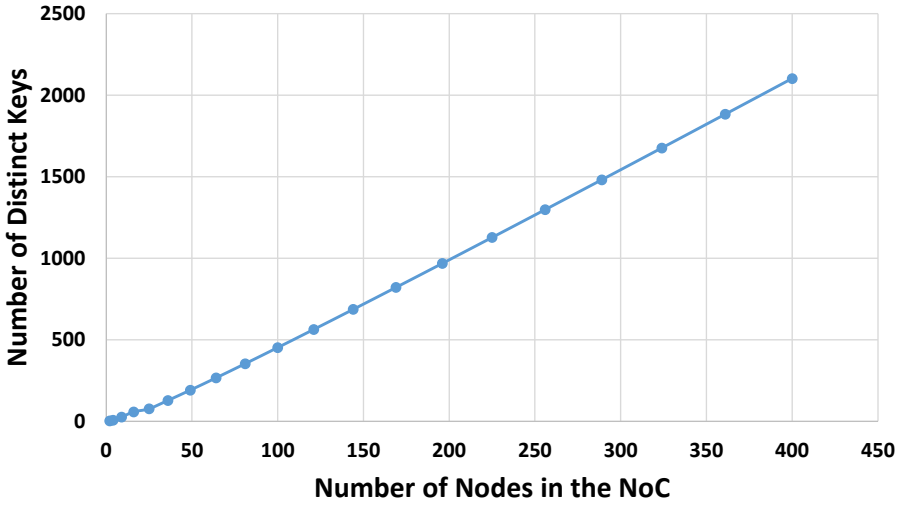


Figure B.2: Scalability of the distinct seed-keys with the NoC size.

APPENDIX C

NOC TRAFFIC PATTERNS

This appendix is dedicated to providing details for the NoC traffic patterns [64].

C.1 Synthetic Traffic Patterns

The traffic pattern defines the destination node (D) based on the source (S) node, the network dimensions (dim_x and dim_y), and number of nodes (N) in the NoC. s_i (d_i) denotes the i^{th} bit of the source (destination) address, whereas S_x (D_x) denotes the x^{th} radix-k digit of the source (destination) address. The bit length of an address is b = $log_2 N$. The traffic patterns are described as follows:

- **Random:** The destination is chosen randomly following a uniform random distribution.
- **Transpose1:** The destination is defined as follows:

$$tmp_x = dim_x \text{ - } 1 \text{ - } Sy,$$
$$tmp_y = dim_y \text{ - } 1 \text{ - } Sx$$

$$D_x = \begin{cases} 0, & \text{if } tmp_x \leq 0 \\ dim_x - 1, & \text{if } tmp_x > dim_x - 1 \\ tmp_x, & \text{otherwise} \end{cases}$$

$$D_y = \begin{cases} 0, & \text{if } tmp_y \leq 0 \\ dim_y - 1, & \text{if } tmp_y > dim_y - 1 \\ tmp_y, & \text{otherwise} \end{cases}$$

- **Transpose2:** The destination is defined as follows:

  $D_x = S_y,$

  $D_y = S_x$

- **Bitreversal:** The destination is defined as follows for a source $S = [s_{b-1}...s_1 s_0]$:

  $D = [s_0...s_{b-2}s_{b-1}]$, i.e., $d_i = s_{b-i-1}$.

- **Butterfly:**

$$\begin{cases} d_0 = s_{b-1} \\ d_{b-1} = s_0 \\ d_i = s_i, \quad for\ 0 < i < b-1 \end{cases}$$

- **Shuffle:**

$$\begin{cases} d_0 = s_{b-1} \\ d_i = s_{i-1}, \quad for\ 0 < i < b \end{cases}$$

APPENDIX D

ROUTING VIOLATION DETECTION FOR RECONFIGURABLE ROUTING TECHNIQUE

In this appendix, the routing violation detection for the reconfigurable routing proposed in section 6.2.3 is studied for the nodes that are surrounding the infected node. In the presented Authentic and Secure NoC model, the routing technique is reconfigured to detour around a HT node. Therefore, the surrounding nodes have a different routing method, which their routing violation check is different.

Figure D.1 shows an 8×8 NoC indicating the nodes of the different routing violation checks. In this figure, the routing violation rule of the "Normal" nodes is the same as the presented one for the X-Y routing. However, the "Surrounding" nodes have different routing and, as a result, they have a different violation check algorithm. Though the "Involved" nodes obey the X-Y routing, they are still involved in the routing violation check of packets that detoured around the HT node. As an example, a packet is injected from the source node (0,3) to the destination (6,3). It will follow the path [(0,3), (1,3), (2,3), (2,4), (3,4), (4,4), (5,4), (6,4),(6,3)]. The packet is detoured at node (2,3). At node (5,4), it will seem that the packet violated the routing, because it should not receive a packet from source (0,3) unless it is received at the North input port and its destination is in the Y-South direction.

There are thirteen different types of nodes that are involved in the new routing violation

check presented in Figure D.1. They are given a distinct symbol (A, B, C, D, E, F, G, H, I, J, K, L, M). The routing violation checker for these nodes is slightly different than the X-Y routing and is presented as follows in Sections D.0.1 to D.0.13.

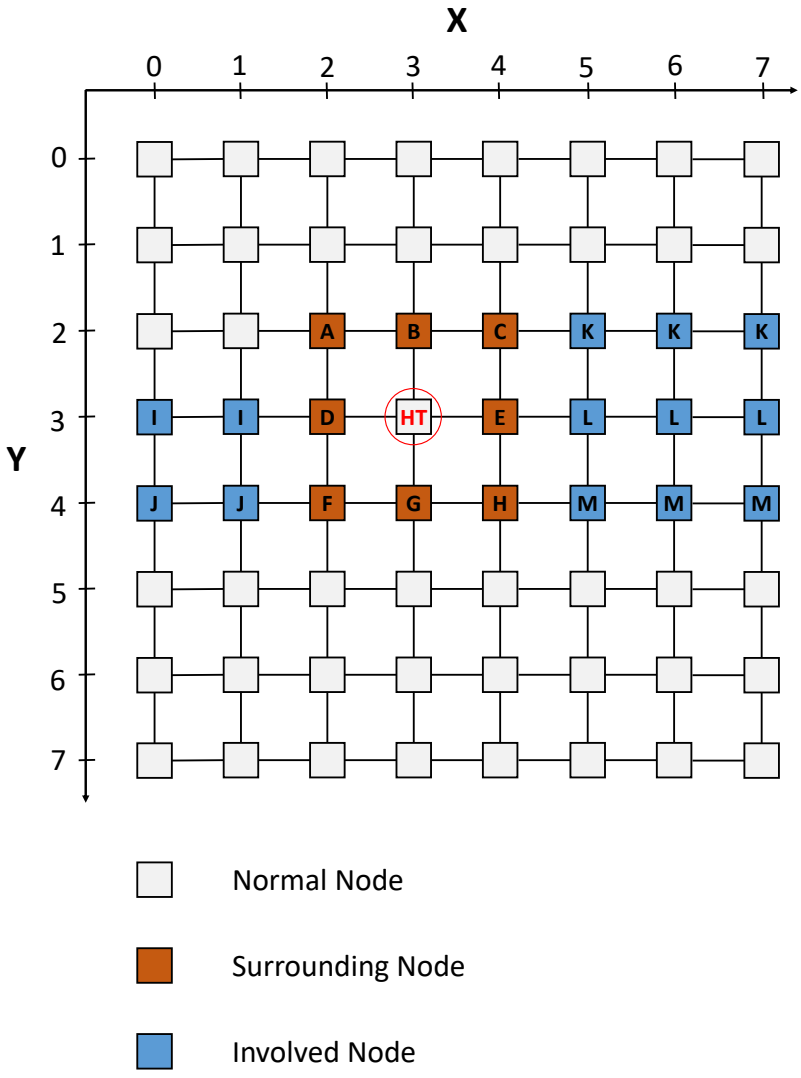


Figure D.1: An example of an 8×8 NoC indicating nodes involved in a X-Y routing violation.

D.0.1 Routing Violation Check at Node "A"

Node "A" applies the X-Y routing violation check for packets received at the North and West input ports. However, packets received at the East input port will follow a different

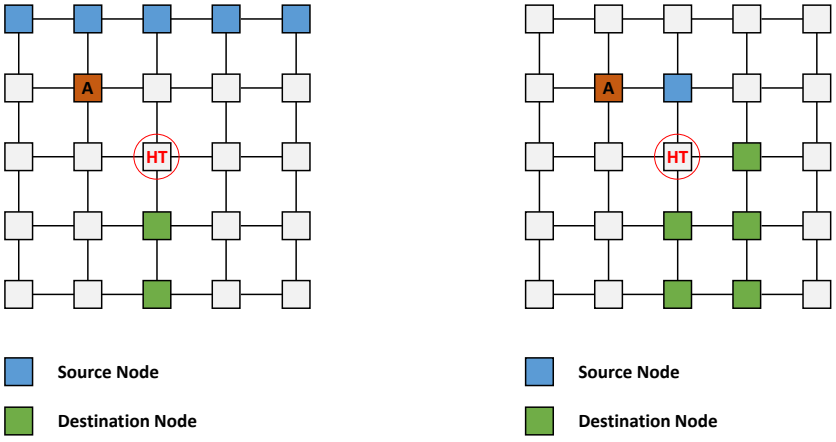


Figure D.2: Source nodes and their corresponding destination nodes for the East input port of node "A".

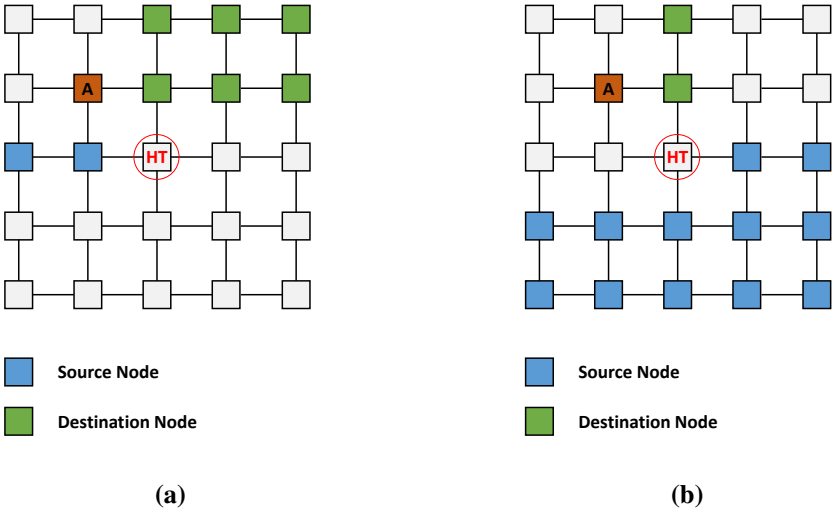


Figure D.3: Source nodes and their corresponding destination nodes for the South input port of node "A".

check, as shown in Figure D.2, where at the East input port, the algorithm should check the source and destination of the received packets and they should match the demonstrated ones in the figure. Similarly, for packets received at the South input port of the node “A”, their source and destination should match what is presented in Figure D.3.

D.0.2 Routing Violation Check at Node “B”

Node “B” applies the X-Y routing violation check for packets received at the North, East, and South input ports. However, packets received at the West input port will follow a different check, as shown in Figure D.4, where at the West input port, the algorithm should check the source and destination of the received packets and they should match the demonstrated ones in the figure.

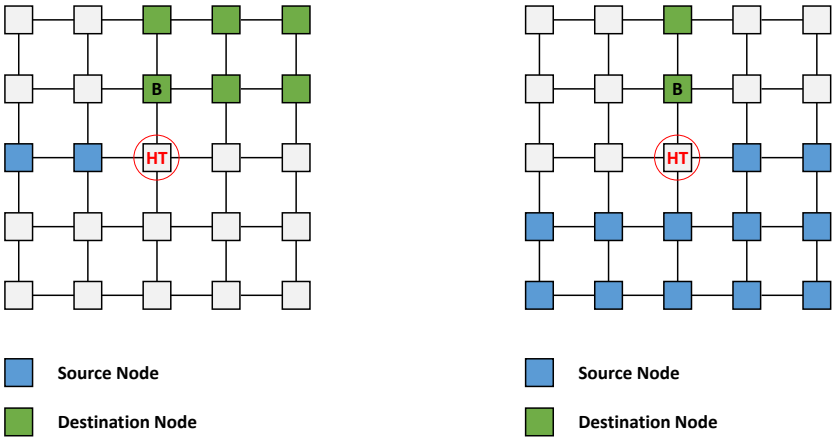


Figure D.4: Source nodes and their corresponding destination nodes for the West input port of node “B”.

D.0.3 Routing Violation Check at Node “C”

Node “C” applies the X-Y routing violation check for packets received at the North, East, and South input ports. However, packets received at the West input port will follow a different check, as shown in Figure D.5, where at the West input port, the algorithm should check the source and destination of the received packet and they should match the demonstrated ones in the Figure.

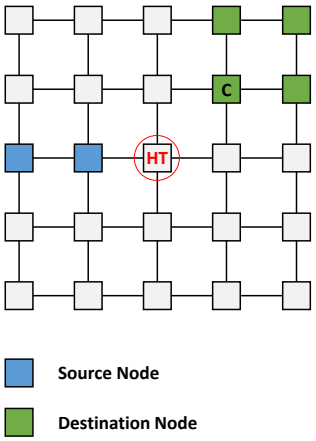
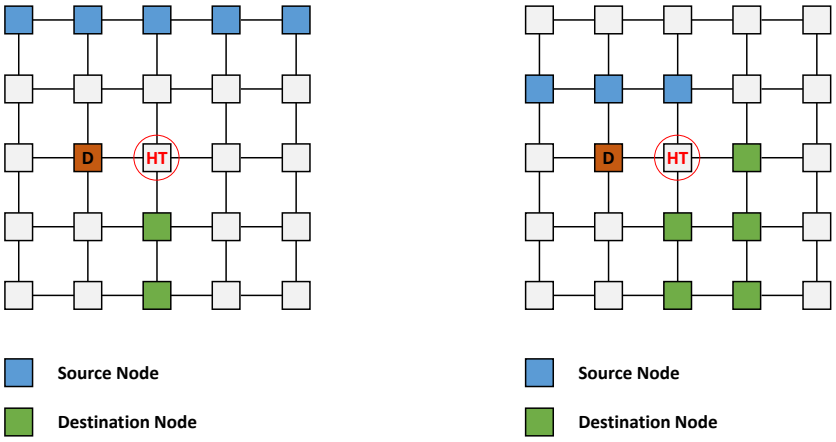


Figure D.5: Source nodes and their corresponding destination nodes for the West input port of node “C”.

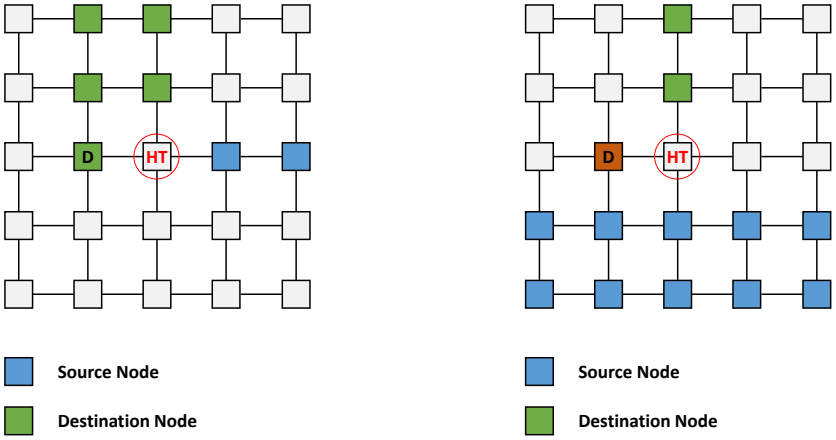
D.0.4 Routing Violation Check at Node “D”

Node “D” applies the X-Y routing violation check for packets received at the East and West input ports. However, packets received at the North input port will follow a different check, as shown in Figure D.6. Similarly, for packets received at the South input port of the node “D”, their source and destination should match what is presented in Figure D.7.



(a) (b)

Figure D.6: Source nodes and their corresponding destination nodes for the North input port of node "D".



(a) (b)

Figure D.7: Source nodes and their corresponding destination nodes for the South input port of node "D".

D.0.5 Routing Violation Check at Node "E"

Node "E" applies the X-Y routing violation check for packets received at the East and West input ports. However, packets received at the North input port will follow a different check,

as shown in Figure D.8. Similarly, for packets received at the South input port of the node "E", their source and destination should match what is presented in Figure D.9.

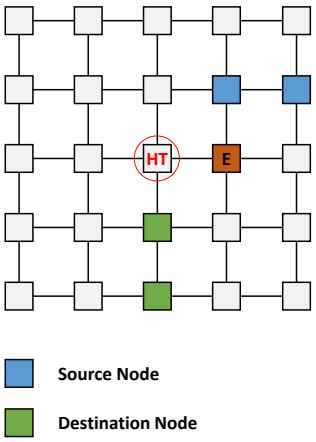


Figure D.8: Source nodes and their corresponding destination nodes for the North input port of node "E".

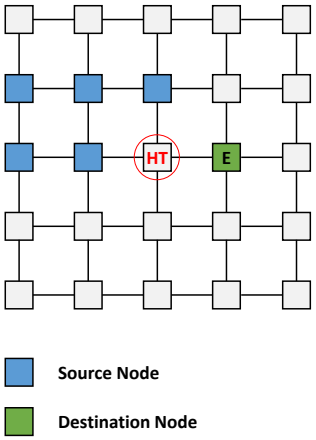


Figure D.9: Source nodes and their corresponding destination nodes for south input port of node "E".

D.0.6 Routing Violation Check at Node "F"

Node "F" applies the X-Y routing violation check for packets received at the South and West input ports. However, packets received at the North or East input port will follow a

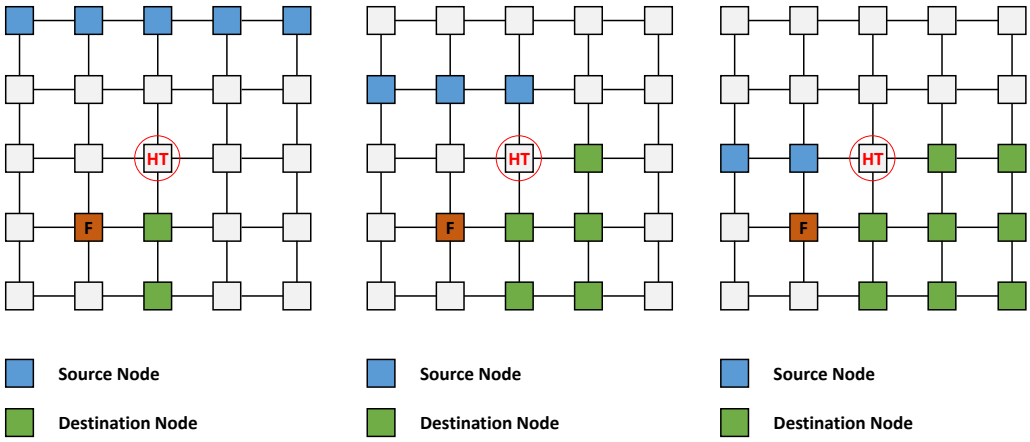


Figure D.10: Source nodes and their corresponding destination nodes for the North input port of node "F".

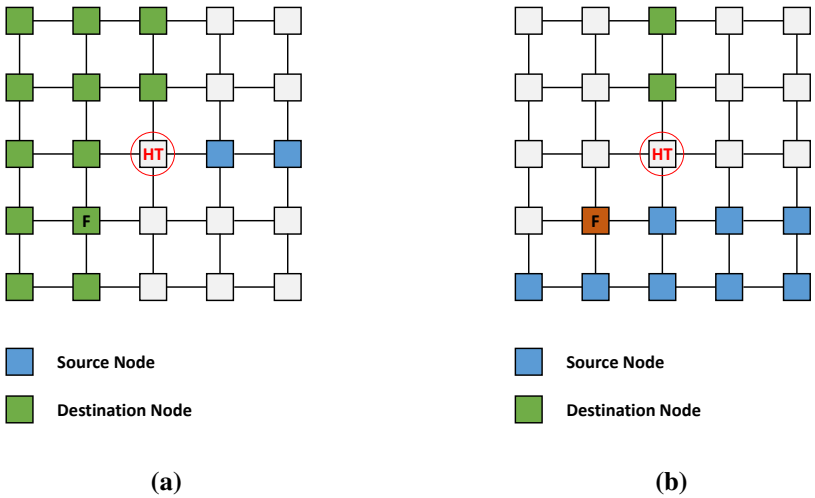


Figure D.11: Source nodes and their corresponding destination nodes for East input port of node "F".

different check, as shown in Figure D.10. Similarly, for packets received at the East input port of the node “F”, their source and destination should match what is presented in Figure D.11.

D.0.7 Routing Violation Check at Node “G”

Node “G” applies the X-Y routing violation check for packets received at the North and South input ports. However, packets received at the East or input port will follow a different check, as shown in Figure D.12. Similarly, for packets received at the West input port of node “G”, their source and destination should match what is presented in Figure D.13.

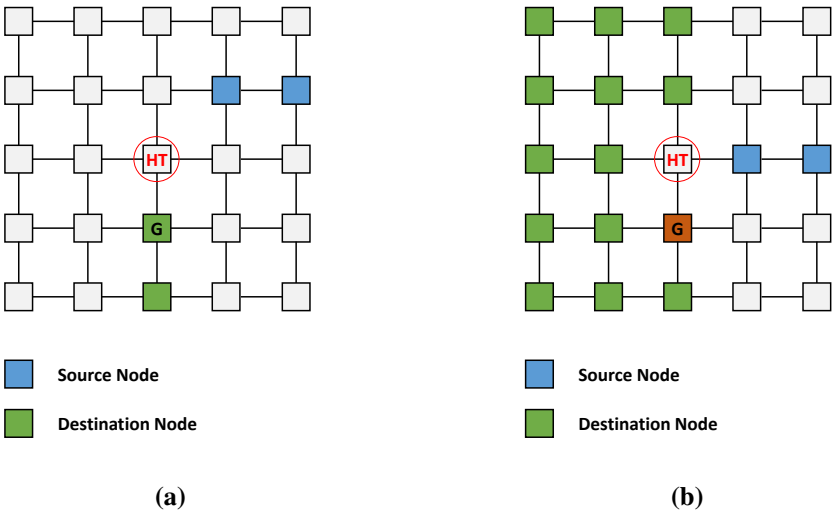


Figure D.12: Source nodes and their corresponding destination nodes for North input port of node “G”.

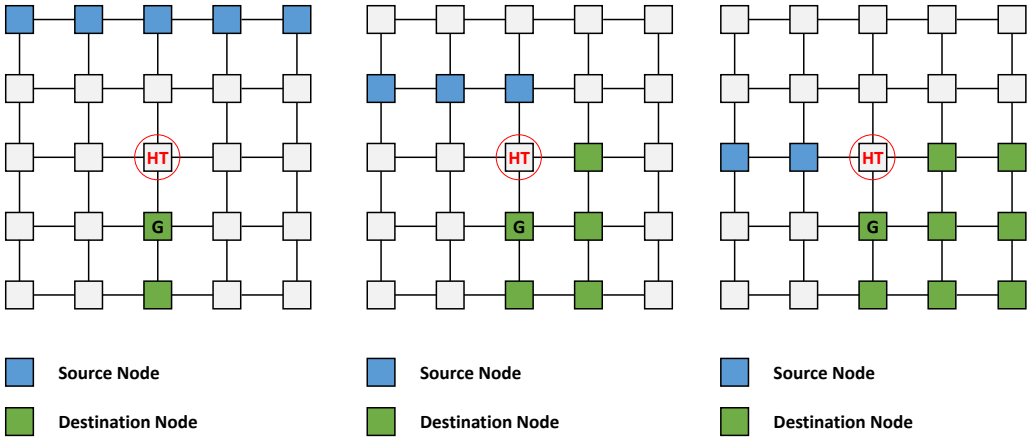


Figure D.13: Source nodes and their corresponding destination nodes for the West input port of node "G".

D.0.8 Routing Violation Check at Node "H"

Node "H" applies the X-Y routing violation check for packets received at the East and South input ports. However, packets received at the North or West input port will follow a

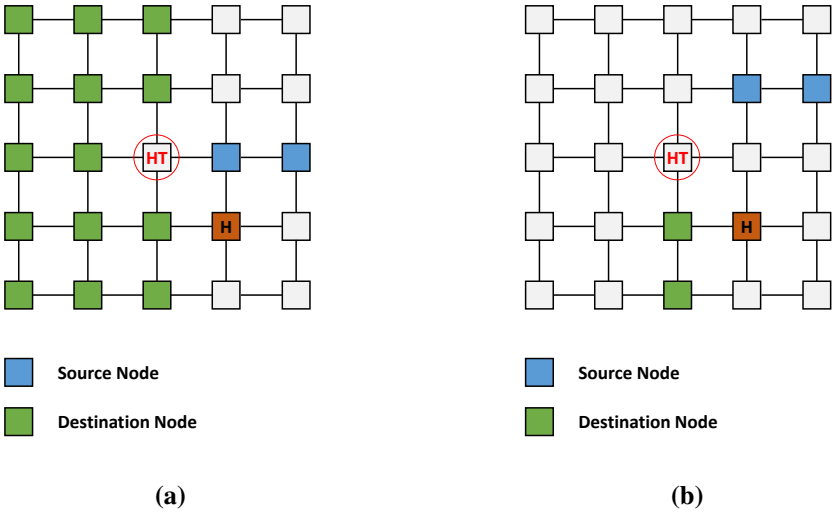
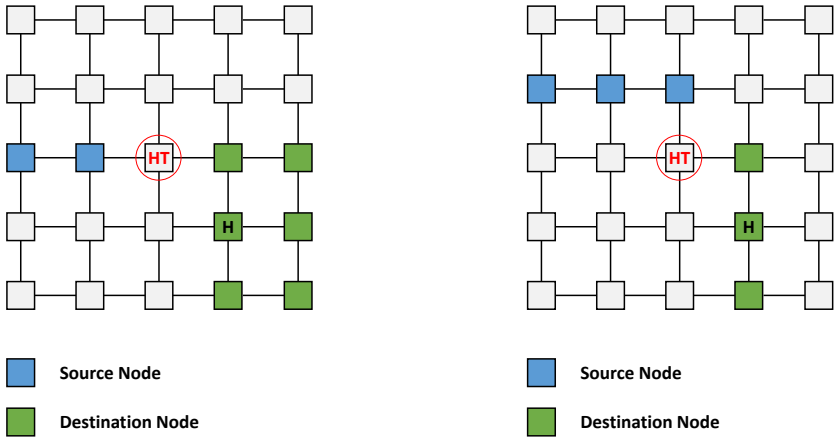


Figure D.14: Source nodes and their corresponding destination nodes for the North input port of node "H".



(a) **(b)**

Figure D.15: Source nodes and their corresponding destination nodes for the West input port of node “H”.

different check, as shown in Figure D.14. Similarly, for packets received at the West input port of the node “H”, their source and destination should match what is presented in Figure D.15.

D.0.9 Routing Violation Check at Node “I”

Node “I” applies the X-Y routing violation check for packets received at the North, East, and West input ports. However, packets received at the South input port will follow a different check, as shown in Figure D.16.

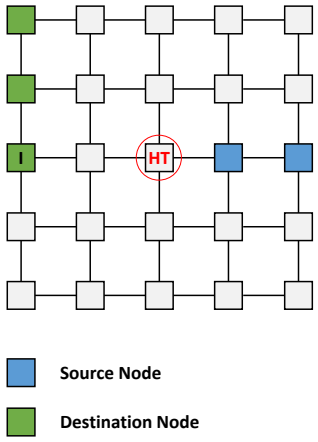


Figure D.16: Source nodes and their corresponding destination nodes for the South input port of node "I".

D.0.10 Routing Violation Check at Node "J"

Node "J" applies the X-Y routing violation check for packets received at the North, South, and West input ports. However, packets received at the East input port will follow a different check, as shown in Figure D.17.

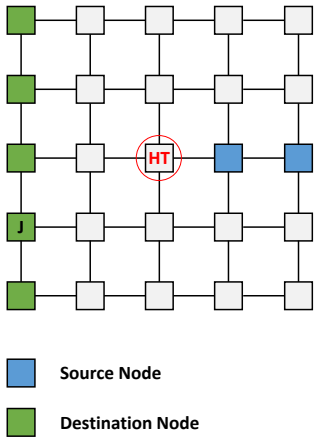


Figure D.17: Source nodes and their corresponding destination nodes for the East input port of node "J".

D.0.11 Routing Violation Check at Node “K”

Node “K” applies the X-Y routing violation check for packets received at the North, East, and South input ports. However, packets received at the West input port will follow a different check, as shown in Figure D.18.

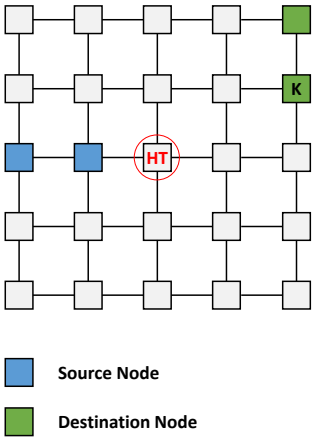


Figure D.18: Source nodes and their corresponding destination nodes for the West input port of node “K”.

D.0.12 Routing Violation Check at Node “L”

Node “L” applies the X-Y routing violation check for packets received at the North, East, and West input ports. However, packets received at the South input port will follow a different check, as shown in Figure D.19.

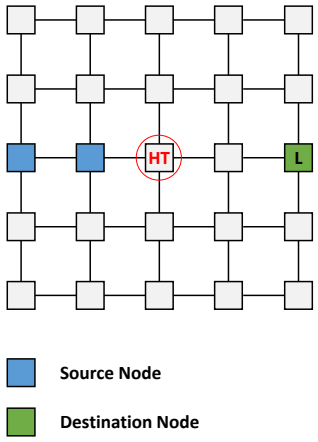


Figure D.19: Source nodes and their corresponding destination nodes for the South input port of node “L”.

D.0.13 Routing Violation Check at Node “M”

Node “M” applies the X-Y routing violation check for packets received at the North, East, and South input ports. However, packets received at the west input port will follow a different check, as shown in Figure D.20.

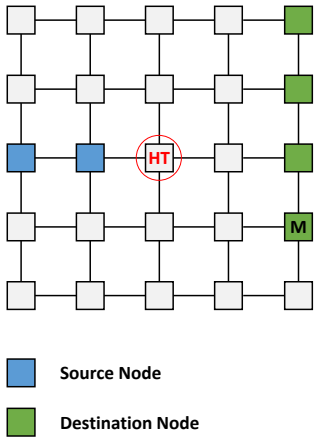


Figure D.20: Source nodes and their corresponding destination nodes for the West input port of node “M”.