Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Informática
Programa de Pós-Graduação em Ciência da Computação

A SECURITY-AWARE ROUTING APPROACH FOR NETWORKS-ON-CHIP

RAMON COSTI FERNANDES

Dissertation submitted to the Pontifícia Universidade Católica do Rio Grande do Sul in partial fullfillment of the requirements for the degree of Master in Computer Science.

Advisor: Prof. Ph.D. César Augusto Missio Marcon
Co-Advisor: Prof. Ph.D. Martha Johanna Sepúlveda Flórez

Porto Alegre
2017

Ficha Catalográfica

F363s Fernandes, Ramon Costi

A security-aware routing approach for networks-on-chip / Ramon Costi Fernandes . – 2017.
100 f.
Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. César Augusto Missio Marcon.
Co-orientadora: Profa. Dra. Martha Johanna Sepúlveda Flórez.

1. Redes Intrachip. 2. NoCs. 3. Roteamento Intrachip. 4. Segurança em NoC. I. Marcon, César Augusto Missio. II. Flórez, Martha Johanna Sepúlveda. III. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS com os dados fornecidos pelo(a) autor(a).

Ramon Costi Fernandes

A SECURITY-AWARE ROUTING APPROACH FOR NETWORKS-ON-CHIP

This Dissertation has been submitted in partial fulfillment of the requirements for the degree of Master of Computer Science, of the Graduate Program in Computer Science, School of Computer Science of the Pontifícia Universidade Católica do Rio Grande do Sul.

Sanctioned on March 23, 2017.

COMMITTEE MEMBERS:

Prof. Dr. Alexandre de Moraes Amory (PPGCC/PUCRS)

Prof. Dr. Eduardo Augusto Bezerra (UFSC)

Prof. Dr. César Augusto Missio Marcon (PPGCC/PUCRS - Advisor)

Profa. Dra. Martha Johanna Sepúlveda Flórez (TUM – Co-Advisor)

Uma Abordagem de Roteamento Seguro para Redes Intrachip

RESUMO

A próxima geração de sistemas multiprocessados intra-chip, do inglês *MultiProcessor Systems-on-Chip* (MPSoC), comportará centenas de elementos de processamento num único chip, com a promessa de alta vazão de comunicação, baixa latência e, preferencialmente, baixo consumo de energia. Devido à elevada demanda de comunicação paralela de aplicações para MPSoCs, a rede intra-chip, do inglês *Network-on-Chip* (NoC), tem sido amplamente adotada como um meio de comunicação confiável e escalável para MPSoCs.

O espaço de projeto para NoCs deve ser explorado para atender à demanda das aplicações atuais. Dentre os parâmetros que definem uma NoC, o algoritmo de roteamento tem sido utilizado para prover serviços como tolerância à falhas, liberdade de *deadlocks* e de *livelocks*, assim como *Quality of Service* (QoS). Conforme a adoção e complexidade de *Systems-on-Chip* (SoC) aumenta para sistemas embarcados, a preocupação com a proteção de dados também torna-se um requisito para o projeto de MPSoCs.

Atualmente, MPSoCs podem ser atacados explorando vulnerabilidades em hardware ou software, sendo o último responsável por 80% dos incidentes de segurança em sistemas embarcados. A proteção contra vulnerabilidades de software pode acontecer em: (i) *Nível de Aplicação*, utilizando técnicas como a criptografia, para evitar a transmissão de dados desprotegidos entre os elementos de um MPSoC, conhecidos como módulos de propriedade intelectual, do inglês *Intellectual Property* (IP); ou (ii) *Nível de Comunicação*, inspecionando ou filtrando elementos na arquitetura de interconexão através de monitores de comunicação ou firewalls, respectivamente. Portanto, um algoritmo de roteamento, ciente dos requisitos de segurança do sistema, deve oferecer proteção ao utilizar rotas confiáveis na NoC, evitando elementos potencialmente maliciosos em rotas porventura inseguras.

A principal contribuição deste trabalho é uma técnica de proteção para NoCs que atua em nível de comunicação, adaptando os algoritmos *Segment-based Routing* (SBR) e *Region-based Routing* (RBR) para que estes considerem aspectos de segurança do sistema, estes caracterizados por zonas de segurança definidas na NoC de acordo com o mapeamento de aplicações nos IPs. A avaliação da técnica de roteamento considera aspectos como a escalabilidade das tabelas de roteamento, a quantidade de rotas seguras definidas entre os IPs, e o impacto desta técnica de roteamento em aplicações do *benchmark NASA Numerical Aerodynamic Simulation* (NAS) *Parallel Bencharm* (NPB).

Palavras-Chave: Redes Intrachip, NoCs, Roteamento Intrachip, Segurança em NoC.

A Security-Aware Routing Approach for Networks-on-Chip

ABSTRACT

The next generation of MultiProcessor Systems-on-Chip (MPSoC) will encompass hundreds of integrated processing elements into a single chip, with the promise of high-throughput, low latency and, preferably, low energy utilization. Due to the high communication parallelism required by several applications targeting MPSoC architectures, the Network-on-Chip (NoC) has been widely adopted as a reliable and scalable interconnection mechanism.

The NoC design space should be explored to meet the demanding requirements of current applications. Among the parameters that define a NoC configuration, the routing algorithm has been employed to provide services such as fault tolerance, deadlock and livelock freedom, as well as Quality of Service (QoS). As the adoption and complexity of System-on-Chip (SoC) increases for embedded systems, the concern for data protection appears as a new design requirement.

Currently, MPSoCs can be attacked by exploiting either hardware or software vulnerabilities, with the later responsible for 80% of the security incidents in embedded systems. Protection against software vulnerabilities can occur at (i) *Application Level*, by using techniques such as data encryption to avoid plain data transmissions between Intellectual Property (IP) modules; or (ii) *Communication Level*, inspecting or filtering elements at the interconnect fabric with communication monitors or firewalls, respectively. As such, a routing algorithm aware of security requirements could also offer protection utilizing trusted communication paths in the NoC, avoiding potential malicious elements in otherwise unsafe communication paths.

The main contribution of this work is a NoC protection technique at communication level by adapting *Segment-based Routing* (SBR) and *Region-based Routing* (RBR) algorithms to consider system security requirements, characterized by security zones which are defined on the NoC according to the mapping of applications on IP modules. Evaluation of the proposed routing technique considers aspects such as the scalability of routing tables, the number of secure communication paths, and the impact of this technique on applications of the NASA Numerical Aerodynamic Simulation (NAS) Parallel Benchmark (NPB).

Keywords: Networks-on-Chip, NoCs, Intrachip Routing, NoC Security.

List of Figures

List of Tables

List of Acronyms

AES – Advanced Encryption Standard

AMD – Algebraic Manipulation Detection

CABA – Cycle Accurate and Bit Accurate

CAFES – Communication Analysis for Embedded Systems

CRC – Cyclic Redundancy Check

DoS – Denial-of-Service

FS – Full-System

FSM – Finite State Machine

GALS – Globally Asynchronous Locally Synchronous

IDS – Intrusion Detection System

IP – Intellectual Property

ISA – Instruction Set Architectures

LUT – LookUp Tables

MPSoC – MultiProcessor System-on-Chip

NI – Network Interface

NAS – Numerical Aerodynamic Simulation

NoC – Network-on-Chip

NPB – NASA Parallel Benchmark

OS – Operating System

PE – Processing Element

QoS – Quality of Service

RSMM – Reconfiguration Security Manager Module

RBR – Region-based Routing

RTL – Register-Transfer Level

SBR – Segment-based Routing

SE – System-call Emulation

SHOC – Secure-enhanced Hybrid on-chip Communication

SZA – Security Zone Awareness

SoC – System-on-Chip

SSP – Square Spiral Protection

TLM – Transaction Level Model

TSVs – Through-Silicon-Vias

Contents

1. INTRODUCTION

The next generation of MultiProcessor System-on-Chip (MPSoC) will encompass hundreds of integrated processing cores into a single chip, with the promise of high- throughput, low latency and, preferably, low energy utilization [33]. However, as the number of processors increases, on-chip communication becomes a critical factor for achieving performance demands [37].

Due to the high communication parallelism required by several applications targeting MPSoC architectures, there is a growing need for a reliable and scalable communication infrastructure. The Network-on-Chip (NoC) paradigm [8], provides these requirements and has been widely adopted as a viable interconnection mechanism for MPSoCs [60][65].

A NoC consists of routers and links. Intellectual Property (IP) modules interface with routers through Network Interfaces (NI). The routing units constitute the underlying communication fabric of the system, where multiple interconnected routers form network topologies that satisfy specific system requirements. NIs connect IPs to the communication fabric, abstracting data translation used by the NoC to the IP modules. Lastly, IPs may be any computational resource in the system, such as a Processing Element (PE), memories, or peripherals, for example.

Communication in NoCs employs a packet-based paradigm akin to computer networks [2]. Each IP is assigned to a unique address in the NoC, which is used by other IPs for message exchanging. Data from a source IP is encapsulated following the NI protocol and injected into the NoC. Routers forward these packets according to routing algorithms and switching techniques to their destination. NIs at the destination then decapsulate packets, delivering the received data to the destination IP.

However, System-on-Chip (SoC) environments are constrained by power dissipation and area consumption, and therefore incapable of employing all of the advanced mechanisms present in computer networks. The NoC design space should be explored to meet the demanding requirements of current applications. Among the parameters that define a configuration of a NoC, the routing algorithm has been employed to provide services such as fault tolerance, *deadlock* and *livelock* freedom, as well as Quality of Service (QoS) [18]. To address such requirements, routing algorithms and switching methods must employ efficient techniques that satisfy SoC constraints [62].

As the adoption and complexity of SoCs increases for embedded systems, the concern for data protection appears as a new design requirement [7]. An MPSoC system may be used in scenarios where availability is a critical factor and downtimes must be minimized. These systems may also handle sensitive information and as such, it is necessary to protect this data from unauthorized access.

Therefore, these security requirements have to be addressed while, at the same time, they are constrained by the computational limitations of SoCs. Similarly to the routing issues, security solutions used in general purpose computing systems are usually unsuitable for MPSoCs due to their demanding computational complexity, requiring approaches that efficiently address such issues [52][53].

Currently, MPSoCs can be attacked by exploiting either hardware or software vulnerabilities, with the latter responsible for 80% of the security incidents in embedded systems [15]. Software attacks in MPSoCs usually start with an abnormal communication, e.g., flooding or an unauthorized element accessing a restrict system resource [54] [17]. Protection against software vulnerabilities can occur at (i) *Application Level*, by using techniques such as data encryption and source authentication, to avoid plain data transmission between IP modules and prevent unauthorized access to system resources, respectively [3]; or (ii) *Communication Level*, by using monitoring elements at the interconnect fabric to detect abnormal communication behavior in the system [17], or firewall modules capable of traffic filtering sensitive content in the system [22].

Protection at communication level can prevent situations where the software layers in the system may become compromised. Due to the intercommunication fabric's system-wide visibility and critical role in enabling system operation, it can be exploited to detect and prevent a broad range of software-based security attacks [15]. NoCs have been shown to aid in the overall MPSoC protection. The works of [17], [54], [28], among others, implement security services at the network level, detecting abnormal communication events of elements and applications in the system. Therefore, a routing algorithm aware of security requirements could also offer protection utilizing trusted communication paths in the system, avoiding potential malicious elements in otherwise unsafe communication paths.

This work proposes a NoC protection technique based on the routing algorithm implementation. By specifying security zones [50] in an MPSoC, the routing algorithm prioritizes communication among paths deemed safe in the system, according to application requirements. This work assumes that the model of *security zones* coupled with a routing algorithm aware of security requirements should enhanced protection from software-based attacks.

1.1 Motivation

Protection of sensitive data and system security is an ever-increasing concern in today's pervasive computing environments. As multicore processors find increasing adoption in domains such as aerospace, medical devices, and the automotive industry, failures could be catastrophic or expose sensitive information [64].

An example of vulnerabilities targeting MPSoCs includes access to shared memories, often employed in MPSoC systems. As the memory has global visibility of the hardware elements in the system, malicious software can access sensitive memory regions for stealing or modifying information, or for disrupting system operation [25]. A hardware-based protection approach offers higher reliability by providing access control outside of the software context, usually protecting the system at communication level by inspecting memory access requests.

Other vulnerabilities attempt to saturate the NoC used in MPSoCs by generating a series of useless communication requests [17], also known as Denial-of-Service (DoS) attacks. These attacks can saturate the network infrastructure and disrupt normal system operation, which can lead to failures of time sensitive processes, as they cannot meet operational deadlines. In some cases, attacks can exploit technological characteristics of the NoC infrastructure in ways that compromise data integrity among communicating elements in the system [56]. Providing traffic isolation and communication level monitoring can identify or prevent such malicious behaviors in the system, increasing protection.

More advanced techniques can monitor the communication behavior of processes in the system, capturing information that can analyze or narrow down the possible values of cryptographic keys of encryption methods [59]. The adoption of security aware routing paths, traffic filtering, and communication level monitoring, can significantly increase system protection by making such attacks infeasible.

In all of the previously mentioned cases, abnormal communication usually precedes software attacks. As such, a protection mechanism that operates at network level can mitigate such vulnerabilities. By establishing safe routing paths among communicating elements in the system, it is possible to enhance the overall system security.

1.2 Objectives

This work proposes to address vulnerabilities in NoC-based systems through a security aware routing algorithm that is capable of considering *security zones* in an MPSoC. Characterization of these zones requires taking into consideration the MPSoC configuration and its applications.

The approach adopted in this work should offer the following benefits: (i) *Enhanced system security*, by routing information in the system through safe communication paths; (ii) *Hardware-assisted security*, by offloading system protection to the interconnect fabric.

The following specific objectives must be satisfied to achieve this general objective:

i. The definition of a routing model that is capable of considering system security requirements and define secure routes in the system, while preserving key characteristics such as *deadlock* and *livelock* freedom;

ii. The creation of an abstract model for the secure routing algorithm, allowing the proposed model to be verified for its correctness;

iii. Validation and evaluation of this new routing algorithm for different security configuration scenarios; and

iv. The adaptation of an existing NoC simulation platform to employ this new routing model, evaluating its behavior for simulated scenarios.

By completing these objectives, the main contributions achieved by this work are:

i. The modeling of security constraints for routing in NoCs;

ii. A study regarding the impact of security zones for securely routing information in NoCs; and

iii. A study on the scalability, the creation of secure routes, and the performance impact of the proposed routing technique for different NoC and workload configurations.

1.3 Organization

The remainder of this document is organized into nine Sections. Section 2 discusses NoC general concepts and its fundamentals. In Section 3, security concerns pertaining NoCs are discussed, further motivating the study of security measures in MPSoCs. Related works regarding studies and attempts to address NoC security are presented in Section 4. Section 5 presents the threat model that constitutes the premises for the proposed solution. Section 6 discusses the model and algorithms of this work to address NoC related security concerns. The evaluation criteria are discussed in Section 7. Results are shown in Section 8. Finally, Section 9 presents the conclusions and future work ideas of this work.

2. NETWORK-ON-CHIP

[8] mentions the term NoC as a network-based SoC interconnection. As future SoCs would be increasingly complex, requiring a modular, component-based approach to both hardware and software design, the NoC has emerged as a communication architecture for complex multiprocessor systems, offering better performance, throughput, scalability, and a high degree of data transmission parallelism when compared to shared bus systems [65]. Further, NoCs favor a Globally Asynchronous Locally Synchronous (GALS) model that simplifies clock skew issues in complex SoC design while propitiating the management of independent clock domains for improved power consumption [1].

Conceptually, NoCs are similar to general-purpose networks by employing a *micro network stack* (Figure 2.1) that encompasses different levels of abstraction. The *Physical layer* is responsible for the physical aspects of communication, such as wiring and the embedded logic responsible for signal processing. *Architecture and Control layer* employs the concept of data links and routing algorithms that establish point-to-point or end-to-end connections among the communicating elements, and encapsulate data into packets for exchange among different NoC elements. In the *Software layer*, system services and applications execute on top of the lower level interfaces. This paradigm decouples abstraction layers, increasing modularity and subsequent reuse of previously designed IP modules.

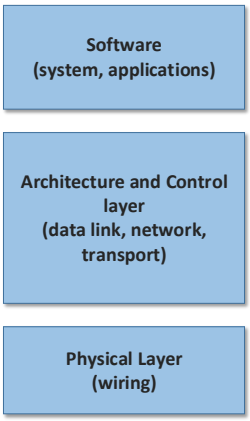


Figure 2.1: Network stack representation. Upper layers represent increasing design abstraction levels. (based on [8])

The architectural design of a NoC relies on NIs that encapsulate data into packets containing protocol information required for communication in the NoC. IPs interface with NIs and each NI is then interconnected to a routing unit that implements a routing algorithm and switching techniques responsible for packet exchange with different NoC elements. Routing units interconnect among each other forming the network, which is configured to satisfy MPSoC requirements. Figure 2.2 illustrates a 2D mesh NoC with 16 routing units and IPs.

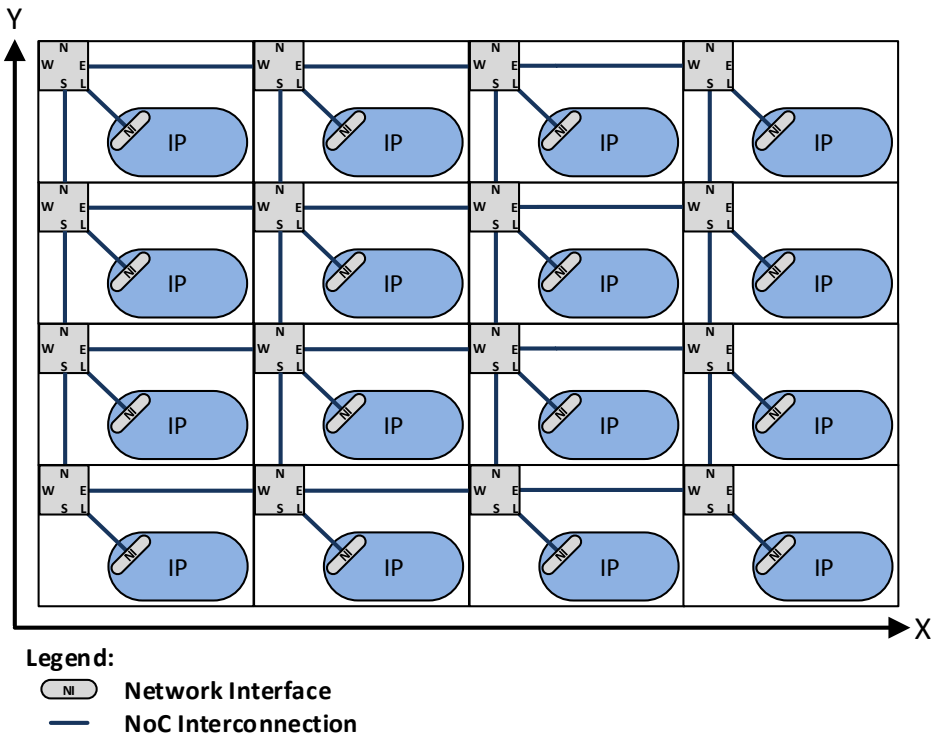


Figure 2.2: 2D Mesh NoC topology with 16 routers and IPs, forming a regular tile-based architecture.

There is not a commonly agreed definition of what is the minimum network configuration that classifies NoCs [49]. However, their survey identifies the following basic properties on existent NoC implementations: (i) separates the communication from the computation; (ii) allows the adoption of distributed / decentralized controllers for communication; (iii) allows an arbitrary number of IP blocks; (iv) may be implemented by different topologies and configurations to satisfy communication requirements (scalability); (v) typically does not utilize long, global wires spanning the whole chip; (vi) allows the customization of several structures, e.g., link width, buffer depth, topology; (vii) facilitates the implementation of an architectural model that uses multiple voltage and frequency domains; (viii) allows in-order data delivery, either naturally or via a layered protocol; (ix) offers varying guarantees for transfers, e.g., QoS; and (x) enables easier implementation of system testing.

The following Sections address implementation characteristics in NoCs, such as switching techniques and routing algorithms in mesh topologies, which are of special relevance to this work.

2.1 Switching Techniques

The switching method defines how data is transmitted among the routing units in the NoC. It is necessary to allocate resources of the communication fabric, establishing logical channels to exchange data among source-destination pairs.

According to [1], routers implement switching techniques that can be classified based on network characteristics. There are two groups: (i) *Circuit Switching*, where a path from source to destination is reserved before transmitting information. It has the advantage of contentionless data transmission once the path is configured. However, any other communication that requires a link in an already reserved path is denied; (ii) *Packet Switching*, where a message is divided into packets and each packet hops through the routing units contained in the path from source to destination. In the works of [23] and [42], the most important modes used in *Packet Switching* are:

- *Store-and-Forward*: packets are stored completely in each router's buffer before forwarding to the next hop. If a router in the path does not have sufficient buffer space, contention occurs. Each router needs a buffer capable of storing at least one whole packet;
- *Virtual cut-through*: packets are forwarded to the next hop once it is guaranteed that the full packet can be stored. Therefore, unlike *Store-and-Forward*, there is no need to wait for the storage of the whole packet before forwarding it to the next hop. Still, this method also requires buffering capacity for at least one whole packet;
- *Wormhole*: each packet is divided into small units called flits. Only the header flit contains the routing information, with subsequent flits containing either the packet size and body or the packet body followed by a tail. The header flit reserves a path between hops and establishes a channel where one or many body flits follow. The major advantage of *wormhole switching* is that there is no need of buffering capacity for an entire packet.

While switching techniques describe how to perform data transferring among elements in the NoC, the routing algorithm implements the source-to-target path selection [10], as presented in Section 2.2. However, before discussing routing algorithms, it is important to identify two key issues in switching techniques that routing algorithms need to address, namely *deadlock* and *livelock*.

2.1.1 Deadlock

In the case of *wormhole switching*, a packet may occupy several intermediate routers at the same time creating a scenario where packets circularly block each other, preventing further communication and therefore creating a *deadlock* [65]. It is important to note that *deadlocks* are not tied to *wormhole switching* and may occur in other switching schemes as well.

Avoiding *deadlocks* requires breaking circular dependencies among packets. Figure 2.3 illustrates a *deadlock* situation where *Packets* 0 to 3 are waiting for buffer availability to reach their destinations. In mesh topologies, it is clear to identify the possible turns. A two-dimensional mesh contains eight turns, as shown in Figure 2.4. Avoiding *deadlocks* requires restricting possible turns in a way that prevents creating scenarios with circular dependencies. However, choosing which turns to restrict requires careful evaluation, as simply adding restrictions does not guarantee *deadlock* freedom.

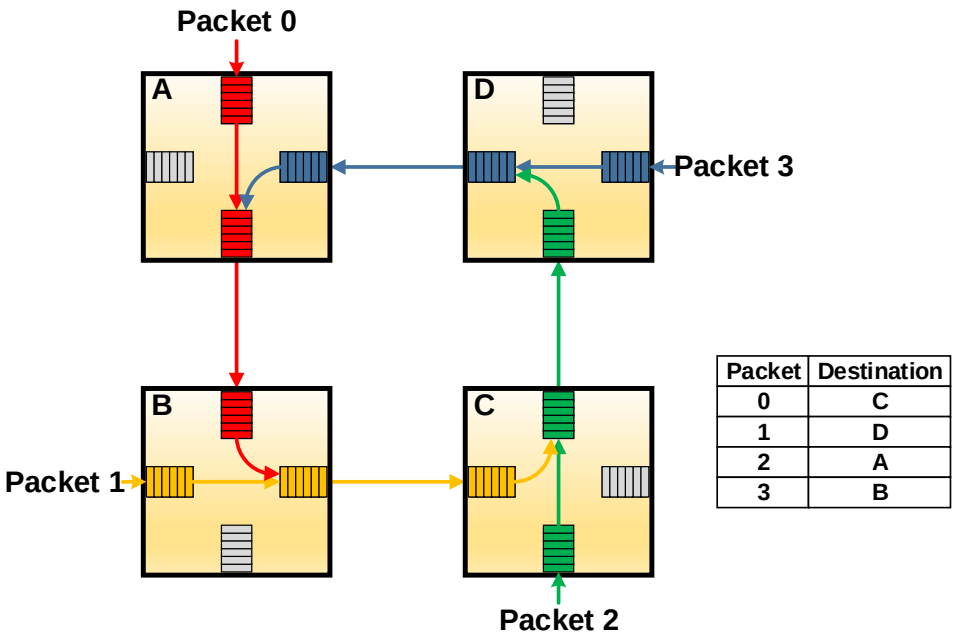


Figure 2.3: Routing *deadlock* caused by a circular dependency. Packets are unable to reach their destination due to network resources occupancy by other packets.

One approach is *dimension-ordered* routing, where packets route in one dimension first, such as the *X*-axis in a mesh topology. Upon arrival at the destination column, packets travel in the *Y*-axis, reaching their destination. Regarding the turns listed in Figure 2.4, *dimension-ordered* routing eliminates *SW*, *SE*, *NW* and *NE* turns (half among the eight possibilities).

Although the approach used in *dimension-ordered* routing solves the *deadlock* problem, it is clear that it also greatly restricts routing possibilities. Other routing algorithms employ a different set of restrictions to address the *deadlock* issue [6] in a less restrictive

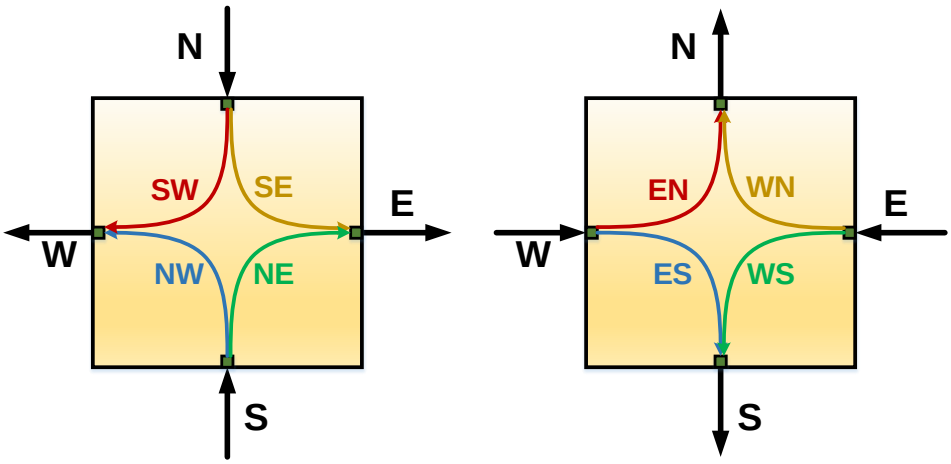


Figure 2.4: The eight possible turns in a two-dimensional mesh-based topology, namely: *SW*: South-West; *SE*: South-East; *NW*: North-West; *NE*: North-East; *EN*: East-North; *WN*: West-North; *ES*: East-South; and *WS*: West-South.

manner. *West-first* routing only prohibits *SW* and *NW* turns. In the *North-Last* algorithm, *NE* and *NW* turns do not occur. *Negative-First* routing eliminates *ES* and *NW* turns. Another algorithm called *Odd-Even* applies turn restrictions differently at each column in mesh topologies, where even columns prohibit *ES* and *EN* turns, and odd columns do not allow *SW* and *NW* turns.

Another model proposed in [41] prevents *deadlocks* by partitioning the NoC into segments. Each segment then has a local bidirectional turn restriction independent from other segments. This model adds a *locality independence* property that allows customization of turn restriction placement according to the different requirements criteria of the system. This work uses proposal discussed in [41] to achieve a *deadlock* freedom implementation, as it is further discussed in Section 6.2.1.

2.1.2 Livelock

A *livelock* is a situation where packets keep circulating the network without ever reaching their destination [18, Chapter 3]. These can only occur when packets are allowed to follow non-minimal paths, such as in adaptive routing algorithms.

According to [18], *livelocks* can be avoided with relative ease. Enforcing minimal paths is possibly the simplest way to guarantee *livelock* freedom. However, non-minimal paths enables routing adaptiveness, which may be of interest in fault tolerance approaches for efficient use of the available network resources.

Deflection routing [65] is one example that illustrates a *livelock* scenario. This algorithm is based on the idea of forwarding a packet through a router's output port at each cycle, assuming that each router has an equal number of input and output ports, so packets can

always find at least one output exit. In *deflection routing*, when contention would occur, the packet picks any available output port to move to the next router, so that it never waits longer than a clock cycle at any given router. Note that this method does not guarantee a minimum routing path. In the worst case, a packet will travel back and forth in the NoC, without reaching its intended destination. However, it has already been shown that deflection routing is *livelock-free* in a probabilistic way so that when a sufficiently long period is considered, the probability of not reaching the destination converges to zero for all packets [18, Chapter 3].

2.2 Routing Algorithms

Routing algorithms define the path taken by a packet between the source and the destination. The propagation nature of packets in the NoC varies with two modes: (i) *Unicast*, where a source-destination communication is a one-to-one data exchange. It means that for a single source to communicate with multiple destinations, various communication events are necessary; or (ii) *Multicast*, where a source can communicate to a set of destinations in a single communication event. According to the survey presented in [1], on-chip communication favors *Unicast* routing strategies due to the presence of point-to-point communication links among routers in the NoC.

Regarding the place where decisions are made in *Unicast* routing, it is possible to characterize it in four groups: (i) *Centralized Routing*, where a central controller determines the packet flow; (ii) *Source Routing*, where the whole path in decided at the source router; (iii) *Distributed Routing*, where each router receives a packet and decides the direction to send it; or (iv) *Multiphase Routing*, which is a hybrid approach that combines *Source Routing* and *Distributed Routing*.

The NoC routing algorithms are most commonly implemented by checking LookUp Tables (LUTs) that contain routing entries, or by executing an algorithm, which can be represented by a Finite State Machine (FSM) [18, Chapter 4]. LUTs are represented by data structures that hold pre-calculated output values for an input information. They can be implemented either logically in software or by dedicated hardware components. According to where routing decisions are made, a single, centralized LUT may exist in the system, or each router may contain its table. The advantage of LUTs is that the routing algorithm can be changed as required, replacing the LUT entries. However, this approach can be inefficient concerning area consumption required to store the LUT as the NoC grows, reducing scalability. Compression techniques are used to significantly reduce LUT size, such as the one mentioned in [26] and [40], which is a basis for the proposed work. Regarding FSMs, these can be implemented in either software or hardware, executing the routing algorithm for each packet as needed.

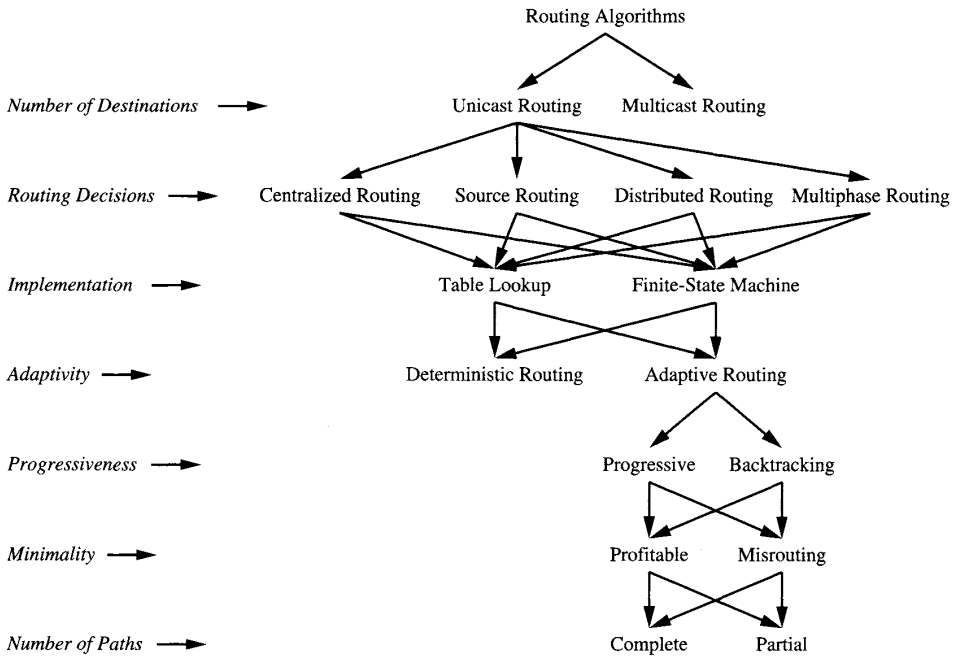


Figure 2.5: Taxonomy of routing algorithms. (captured from [18, Chapter 4])

Another characteristic of routing algorithms, which is implementation independent, regards *deterministic* or *adaptive* behavior [18, Chapter 4]. On one hand, in *deterministic* routing packets always follow the same path between a source and a destination pair. *Adaptive* algorithms, on the other hand, depend on network conditions, such as traffic and/or channel status. This characteristic of *Adaptive* algorithms provides adaptability to network loads or faulty regions in the network. There are other features of *adaptive* routing identified in [18] regarding strategies used for link reservation (*progressiveness*), path deviation (*minimality*) and path coverage (*number of paths*), as Figure 2.5 illustrates.

The following subsections provide a general overview of existing *deterministic* and *adaptive* routing algorithms for either *FSM* or *Lookup Table* implementations.

2.2.1 Deterministic Routing Algorithms

Some NoC topologies, such as meshes, can be decomposed into several orthogonal dimensions. Computing distances in a source-destination pair in these cases becomes a matter of measuring the offset in each dimension. Progressive algorithms aim to reduce this offset at each routing step, such as *dimension-ordered* routing [18, Chapter 4.2], mentioned in Section 2.1.2. This algorithm is also extensible for *n*-dimensional meshes as well, being usually called *XY* or *XYZ* for two-dimensional and three-dimensional mesh topologies, respectively.

Another approach is *source routing*, where the source precomputes the path in a source-destination pair and appends the path information to the packet header [10]. This model requires simple hardware implementation as it is only necessary to notify the output port through where the packet needs to be sent. However, the whole path pre-computation is required, and additional communication overhead is introduced, as the entire routing path must be present in the packet's header.

2.2.2 Adaptive Routing Algorithms

The simplest approach in adaptive routing is *deflection routing*, discussed in Section 2.1.2. This algorithm provides full adaptability by not imposing turn restrictions or conditional behavior that depends on network conditions.

Other variations in *adaptive routing* include *West-First*, which is partially adaptive. Turns to the west are prohibited, so packets traveling west of the source router are routed in a deterministic way. Eastbound packets, however, are routed with full adaptivity [63]. Similarly, the *North-Last* algorithm does not allow turns from north to east or from north to west, so that a packet will only head north (if necessary) once it has traveled with full adaptivity in the other dimensions. In *Negative-First* routing, turns from north to west or from east to south are prohibited, so that packets can travel a miss-routing path to the southwest of its destination. *Odd-even routing* is another algorithm that provides adaptivity and offers a different approach for turn restriction placement. Unlike the previous algorithms, in *odd-even* all turns are permitted globally, but each column in a mesh topology has different turn constraints in a manner that prevents the routing path from ever forming a cycle [18, Chapter 4.3.2].

Routing table based algorithms are also a possibility for adaptive routing, providing multiple routing entries to reach a destination. [26] proposed the *Region-Based Routing* (RBR), which has the benefit of reducing the necessary table space to store routing entries. By grouping destinations into regions in the NoC, it is possible to use a single routing entry to reach a set of destinations. This work uses this algorithm for packet routing, which is further discussed in Section 6.2.3.

3. NOC SECURITY

A security-aware design of communication architectures is becoming a necessity in the context of embedded MPSoC security. Even though complex communication infrastructures, such as NoCs, may lead to weaknesses in the system that can be critical and should be carefully studied and evaluated [24], NoCs can be a contributing element in secure SoC design by providing the means to monitor system behavior, enabling the detection of attacks.

Weaknesses during design time inevitably remain in the system implementation and are often exploited by attackers in the form of either software or hardware attacks. Software attacks that exploit vulnerabilities in application code or weaknesses in the system design are the most common type of attacks [45]. Given that multiple existing devices already employ MPSoC architectures, it is imperative that security is considered at design time rather than be employed as a reactive measure.

As [15] mentions, embedded systems are typically designed by assembling various IP modules on top of a standard communication architecture. Due to the intercommunication fabric's system-wide visibility and critical role in enabling system operation, it can be exploited to detect and prevent a broad range of software-based security attacks. The communication architecture is capable of determining system events, e.g., (i) which components or programs are accessing a given memory region; (ii) if a component obeys system-level rules; (iii) if the present configuration for a peripheral device is valid for accessing a component; and (iv) system intrusions due to changes in the communication behavior of an application.

The following Sections discuss general principles employed in security solutions, and the identification of vulnerabilities and associated techniques to protect NoC-based MPSoC systems.

3.1 Security Services

According to [5], security encompasses three *primary* attributes. In the context of MPSoC systems, these principle correspond to:

- *Confidentiality*: The absence of unauthorized disclosure of information; e.g., applications or IP blocks that do not possess access rights cannot read sensitive data stored in an MPSoC;
- *Integrity*: The absence of improper alterations, e.g., the content of packets traversing the communication fabric must remain unchanged by unauthorized parties in the path from source to destination; and

- *Availability*: The readiness for correct service, e.g., it is necessary to guarantee that sufficient resources remain available to satisfy the performance requirements in the system, such as *real-time* processes. QoS techniques can also address this property.

Furthermore, *secondary* attributes can be defined, refining or specializing the *primary* attributes, namely:

- *Accountability*: Availability and integrity of a claimed identity. The originator of a request to access a secured component must present credentials that prove its identity, e.g., an IP block that attempts to access resources located on a target IP has to provide information that identifies itself for the target;
- *Authenticity*: Integrity of a message content and origin, possibly including other message attributes related to the origin, e.g., an application executing in a PE requests read/write operations to a memory address in the MPSoC. The system needs to verify if the source application has access permissions at the specified memory address; and
- *Nonrepudiability*: Availability and integrity of the identity of elements in the system. It avoids that any element denies an action, e.g., an IP block cannot forge its credentials to gain access permissions to protected system resources.

The combination of these security attributes define a set of security constraints which systems must adhere by, characterizing a *Security Policy*. The enforcement of these constraints is performed by *Auditing* the system behavior, e.g., logging configuration and communication events in the MPSoC with different granularity levels, such as IP blocks, applications, and router configurations [47].

Security attributes are provided by *security services*. These ensure the system protection by managing the system resources [34]. Implementation of *security services* can be either physical (physical barriers) or logic (such as cryptography, digital signatures, or hash functions, for example). According to [15], the inclusion of *security services* in SoCs is a challenging process, as these can easily exceed the computational capacity of embedded systems.

Such *security services* can be integrated into the communication infrastructure of the MPSoC. By exploring the NoC and NI configuration parameters, system security can be enhanced. The following sections present threats to MPSoC systems, which a secure NoC can protect.

3.2 Memory Access Vulnerabilities

Perhaps one of the most discussed vulnerabilities in literature regards memory access in MPSoCs [24][28][29][15][25]. As Figure 3.1 shows, a typical MPSoC architecture

includes PEs and shared memories accessible through the intercommunication architecture. Without an access control mechanism, any process running on some PE may access any address range in the memories, for either capturing sensitive information or disrupting system operation. Even if a protection implementation exists at Operating System (OS) level, there are no guarantees that malicious applications could not bypass OS protection, leaving the system vulnerable.

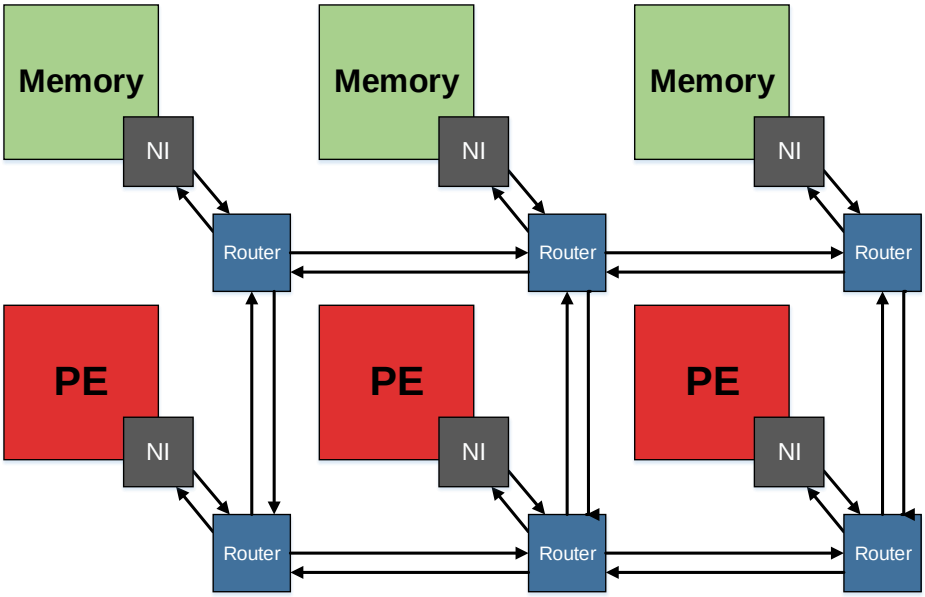


Figure 3.1: Typical NoC-based MPSoC architecture with memories and PEs.

To address this vulnerability, [24] proposes an architecture with *probes* embedded into the NI of PEs that monitor communication events and filter address ranges in memory accesses, specified in the content of packets, providing *Accountability* and *Authorization*. Attempts to access protected memory blocks from an unauthorized traffic initiator are blocked and notified to the system, signaling the presence of a compromised core. Other works such as [28] or [31] adopt a similar technique while using *firewall* modules, decoupled from the NI and containing memory access rules.

The verification modules placement depends on implementation, and could occur either: (i) *at the traffic initiator*, which prevents unauthorized packets from entering the NoC, thus alleviating resource usage but increasing overhead, as each verification module is individually reconfigured for all memory related accesses; or (ii) *at the target*, which filters incoming memory access requests while offering relatively simpler reconfiguration, since only the protected component's verification module needs to be reconfigured, but enabling packets with invalid access requisitions to consume NoC resources, facilitating attacks as discussed in Section 3.3.

3.3 Communication Vulnerabilities

Another common attack aims to disrupt system operation through the form of *DoS* attacks; also known as *Bandwidth Reduction* or *Draining Attacks*, this often occurs by injecting useless packets that saturate the communication architecture.

According to [17], *DoS* attacks can be further categorized in four kinds of attack scenarios: (i) ***Replay***, which corresponds to repeated requests that wastes bandwidth, increasing communication latency; (ii) ***Incorrect path***, which introduces packets with false paths that could trap communication channels and prevent valid packets from traversing the NoC; (iii) ***Deadlock***, which creates packets with paths that intentionally disrespect deadlock-free rules of the routing techniques; and (iv) ***Livelock***, which injects packets that can never reach their destination and stay indefinitely traversing the network, wasting bandwidth, latency and power. While the latter three categories should not affect distributed routing algorithms (but could disrupt source-routing implementations), the first category targets any of the routing techniques previously discussed in this work.

Some *DoS* attacks may also be indistinguishable from normal communication events. Detecting this unusual traffic requires data collection and analysis similar to Intrusion Detection Systems (IDS) used in computer networks [32].

Variations of *DoS* attacks target specific technological characteristics of MPSoCs. Through-Silicon-Vias (TSVs) used for 3D MPSoC interconnection, as described in [56], are prone to coupling effects by adjacent TSVs that can disrupt communication. A malicious application can generate traffic at specific TSVs to (i) alter sensitive data in adjacent TSVs; (ii) induce degradation of a victim TSV under the effect of an aggressor TSV by increasing interfacial cracks or by promoting electromigration effects; and (iii) modify its performance by delaying or speeding up signal transitions.

Techniques to mitigate such attacks employ an *Integrity* service that includes traffic scheduling among interference prone TSVs to avoid these effects. Another strategy would be routing algorithms aware of sensitive communications that promote traffic isolation while providing *Availability*. The adoption of routing algorithms with safe routing paths and aware of security requirements, as this work proposes, could be extended to address this attack scenario.

3.4 Time-based Attacks

Time-based attacks employ more sophisticated techniques to exploit the communication patterns at MPSoC. Relying on access patterns of traffic flows, attackers can infer sensible information regarding applications executing in the system.

This aspect of MPSoC security is studied in [59]. As many-core architectures are based on memory hierarchies, several levels of cache can be integrated.

Some cores may integrate a processor with its associated caches. In the event of a cache miss, the coherency mechanism has to access another distant core that contains the next element in the memory hierarchy. The attack scenario, shown in Figure 3.2, consists of two processes executing in parallel. A sensible process (*S*) is performing a cryptographic function, such as the *Advanced Encryption Standard* (AES), which employs a secret key. In the event of a cache miss during *S* execution, the cache coherency mechanism has to access the memory located at *D*. Meanwhile, an attacker *A* is constantly injecting packets into the NoC to the same memory *D*. Due to communication resource sharing between *S* and *A* to access *D*, *A* suffers throughput degradation, which leaks information about the nature of the memory access of the crypto-process in *S*. As [44] extensively details, time-based attacks such as this can reveal considerable information for cryptanalysis by reducing the search space of secret keys, weakening encryption strength, thus violating *Confidentiality* and possibly *Integrity*. [59] suggests the use of random arbitration and adaptive (non-deterministic) routing as protection for this sort of vulnerability.

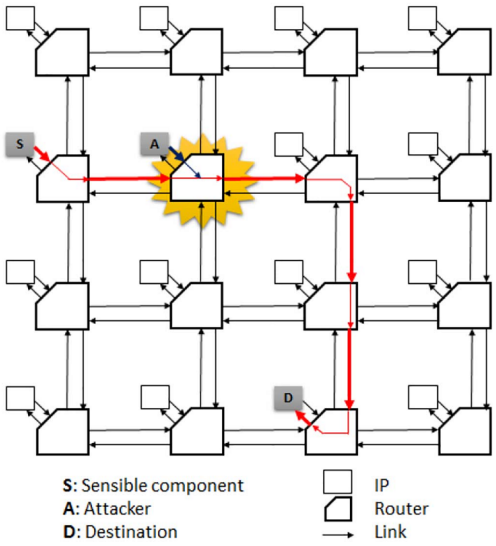


Figure 3.2: Sensible process communicating over the NoC with an attacker in the routing path. (captured from [59])

Time based-attacks require knowledge of the underlying communication architecture to measure throughput variations and later inference of information on sensitive processes. Thus, random arbitration introduces variations in access times of the routing path, masquerading memory access behavior of sensitive processes. Another approach is to adopt non-deterministic routing strategies on the NoC. Adaptive algorithms such, as *west-first*, offer low-cost and *deadlock* freedom routing. As the routing path is unpredictable in most situations, the inference of memory access patterns of sensitive processes by an attacker becomes infeasible, protecting the system from time-based attacks.

4. RELATED WORK

This section presents the works regarding security in NoC-based MPSoCs. It is important to note that the items discussed in this section do not represent all existing works that takes the subject of NoC security into account.

4.1 Security Effectiveness and a Hardware Firewall for MPSoCs

The work of [28], [29] proposes a hardware based firewall module placed between the NoC's routing units and IP modules. This firewall is capable of application level filtering through the Process Identifier (PID) of each task, monitoring memory requests invoked by any initiator connected to the NoC. A lookup table contains the access rights for each task, preventing malicious communication from entering the NoC. Rule checking considers the memory's physical address, as it takes place after virtual-to-physical address translation. The main purpose of memory protection is preventing a process from accessing shared memory regions not mapped for it that could contain sensitive data. This approach protects only the initiator side, assuming that memory protection takes place at processor requests instead of memory accesses.

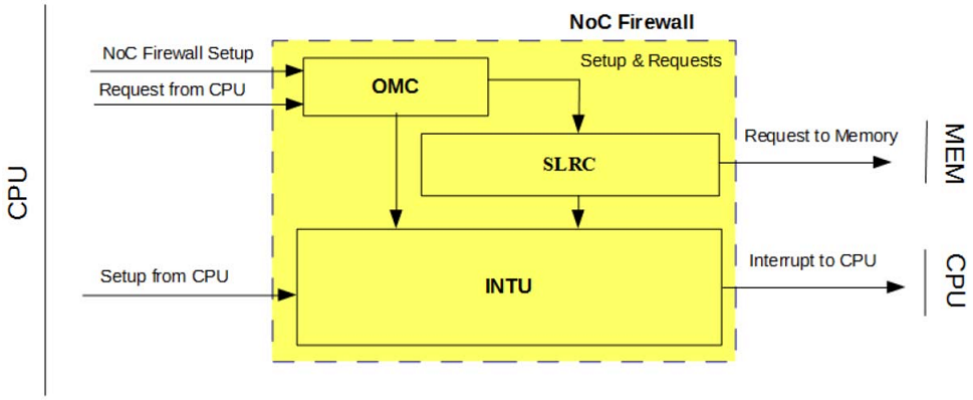


Figure 4.1: Top-level illustration of the Firewall module. (captured from [28])

The architecture, as Figure 4.1 illustrates, consists of the following submodules:

- The *Operating Mode Controller* (**OMC**), which accepts, decodes and dispatches NoC firewall commands.
- The *Segment-Level Rule-Checking* module (**SLRC**) that processes incoming memory access requests and configuration commands. This module implements the deny rules used for filtering application requests and internal monitors for recording NoC related activity.

- The *Interrupt Unit* (**INTU**), which accepts, in parallel, interrupt requests from the OMC and SLRC blocks, and reports interrupts to the CPU this firewall module is associated.

A Cycle Accurate and Bit Accurate (CABA) SystemC model, as well as a higher-level Transaction Level Model (TLM) in the Gem5 system simulator, validates their system. Results evaluate the effect on system latency and power consumption at network layer by measuring the effects of having the firewall modules enabled/disabled for malicious traffic scenarios. Unfortunately, no results are illustrating the performance impact of the firewall modules regarding power and latency overhead, as the authors did not consider this evaluation.

4.2 Elastic Security Zones for NoC-Based 3D-MPSoCs

In [57], the authors propose *Adaptive Ciphered Elastic 3D-NoC*, targeted at 3D-MPSoCs (3D-ACeNoC), for protecting against software attacks. The system architecture consists of dynamically reconfigurable hardware firewalls that adapt to the set of tasks mapped into the MPSoC. This characteristic creates security zones within the NoC topology. Components inside the same security zone are trusted and considered secure; hence, the transactions among them do not require security checking nor content protection.

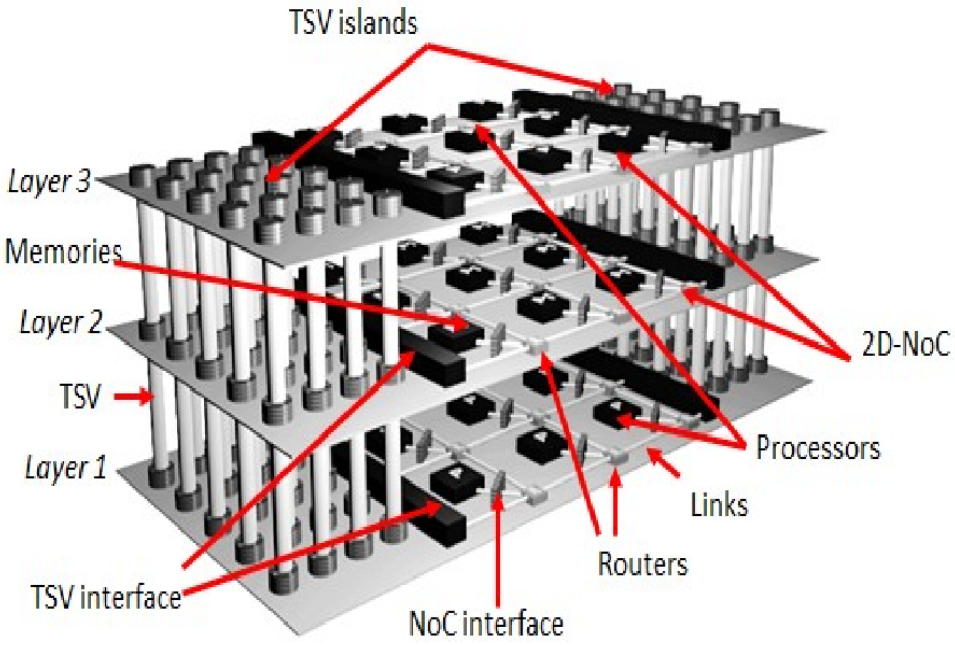


Figure 4.2: Example of a 3D-NoC based 3D-MPSoC (captured from [57])

Their architecture consists of three key components:

1. *Security Checkers*: responsible for blocking or allowing transactions according to security policies. These devices implement access control (authorization for the use of

a resource), authentication (validates the initiator IP integrity) and confidentiality (data ciphering). There are four types of security checkers, depending on their location in a 3D topology like the one shown in Figure 4.2:

- *Computation-Communication*, linking IPs to routers;
- *Memory-Router*, linking memory dies to routers;
- *Router-Router*, linking consecutive routers; and
- *Routers-TSVs*, interconnecting routers to vertical connections.

Either these devices can verify or not incoming/outgoing data according to the configured security zone, as traffic originated by trusty components do not require verification.

2. *Reconfiguration and Security Manager Modules* (RSMM): is a software task with super privileges that configures the *Security Checkers* according to a security policy. This module is located on a processor die of the 3D-MPSoC. It also manages the security keys of each zone used for ciphering data. This module is aware of the application mapping within the 3D-MPSoC to define the security zones. The RSMM module generates packets whose final destination is the target firewalls, to perform table updating for the Security Checkers. All traffic originated from this module is secure and have the highest communication priority.

3. *Monitors*: embodied at the NoC routers, these modules are constantly auditing the communication behavior of the MPSoC. They are responsible for verifying the completion of the communication among the different master/slave pairs of the 3D-MPSoC. The RSMM module manages the elasticity of security zones (contract/expand) by enabling/disabling the *router-router* security checkers. A security zone expands when the security policy of a neighbor IP component is compliant with the security policy of the security zone, and contracts otherwise.

Validation uses a SystemC-TLM model that performs latency, power and thermal evaluations. Synthetic and real application traffics allow the implementation verification by filtering malicious data flows. The synthetic traffic uses Poisson and Long Range Dependence characteristics, while the real application traffic uses the SPLASH-2 benchmark [61]. Their results indicate a higher performance with lower energy requirements while compared to other works with similar characteristics, such as 3D-SZNoCs [55] – an implementation by the same authors without security zones – and 3D-SENoC [54], which implements security zones but without elasticity.

4.3 3D-LeukoNoC: A Dynamic NoC Protection

Targeting specific 3D NoC vulnerabilities, [56] proposes a secure architecture capable of traffic isolation and QoS. Interconnections among different planes in 3D NoCs often employ the TSV bus technology. TSVs are very short, fine-pitched links that create high-bandwidth communication channels, and are usually grouped into islands for thermal, reliability and fabrication considerations.

It is known that TSVs greatly suffer from coupling effects, which turns this technology vulnerable to integrity and operational problems. These vulnerabilities can be exploited by malicious software that generates traffic into adjacent TSV links, promoting electromigration effects that may alter legitimate traffic or alter signal transitions. Therefore, unlike security mechanisms used in 2D NoCs, 3D topologies require specific countermeasures.

The 3D-LeukoNoC uses the concept of Keep-out-Zones to impose an isolation area around TSVs that carry sensitive data. The Keep-out-Zone size depends on technological and dimensional parameters. Besides, even when multiple TSVs may be carrying sensitive information, isolation of these channels help prevent communication degradation caused by the coupling effect.

The security infrastructure encompasses three key components: (i) *Recognizers* ($\boldsymbol{R_G}$), which filter data by inspecting packet fields with known security rules and can be enabled or disabled accordingly; (ii) *QoS routers* ($\boldsymbol{QoS_R}$), which are QoS-enabled routers with reconfigurable arbitration parameters; and (iii) *Lymphocyte-B* (L_B), a reconfiguration and security manager module that reconfigures the R_G and QoS_R components. The secure NoC architecture operates by blocking malicious traffic through the $\boldsymbol{R_G}$ and manages inter/intra-layer communications through $\boldsymbol{L_B}$ and $\boldsymbol{QoS_R}$ at run-time.

Security policies and performance requirements exist for each application in the 3D-MPSoC. Packets in the NoC carry identifiable information for each application so that 3D-LeukoNoC can enforce its policies and deliver the necessary performance.

Experiments used a SystemC-TLM model capable of integrating latency, power, thermal and coupling characteristics of the 3D NoC. Results compared 3D-LeukoNoC to other secure NoC architectures, some that solely employ 2D security countermeasures while others attempt to extend 2D techniques to 3D NoCs. However, as argued by the authors, 2D security techniques are shown to be insufficient, emphasizing the proposed architecture.

Although the authors have chosen traffic contention mechanisms to mitigate the threat model, it should be possible to achieve similar results by adopting routing techniques that take the security aspect of communication into consideration, as this work proposes.

4.4 Automatic ILP-based Firewall Insertion for Secure Application-specific Networks-on-Chip

In [31], the authors propose NoC based hardware firewalls for application-specific NoCs. Their research focuses on firewall placement in general-purpose NoCs, which are often integrated at the NI of targets holding sensitive data. Since certain non-reprogrammable traffic initiator MPSoC components cannot be compromised, traffic from these trusted modules would unnecessarily pass through firewalls, introducing significant communication overhead (an increase of 3% to 9% of total bandwidth), as the packet header from these components requires additional fields for the firewalls to perform their functions.

Firewalls used in their architecture provide three security features: (i) *Authenticity*, which guarantees that sensitive data in a target can only be accessed by authorized initiators; (ii) *Accountability*, which guarantees that the packet is generated from the specified initiator; and (iii) *Availability*, which guarantees that critical flows are not blocked even in attacking scenarios.

To guarantee *Authenticity*, the firewalls implement three incremental security levels: (a) $L = 1$: firewalls check if the packet initiator has access to the target; (b) $L = 2$: firewalls also verify address ranges accessed at the target; and (c) $L = 3$: firewalls also check the initiator's role. The firewalls provide *Accountability* using three incremental security levels: (a) $L = 1$: firewalls crosscheck the initiator ID with the packet ID; (b) $L = 2$: firewalls also check the packet's routing path; and (c) $L = 3$: firewalls also check the packet's unique sequential number. Finally, the firewalls implement a bandwidth counter that registers each passing traffic flow within a time interval to prevent bandwidth overusing, guaranteeing *Availability*.

For determining the required firewall security levels, each traffic initiator i has a trust level t_i while each target j possess a security level s_j. If $t_i \geq s_j$ then traffic does not need to be checked. However, if $t_i < s_j$ then a firewall of at least level s_j is required.

The remainder of their work discusses in detail the techniques used to predict optimal firewall placement, through integer linear programming, and experimental results. While the former is best explained in [31, Section III], the later indicates an overhead reduction of 63% to standard solutions for an industrial chip specification that contains 25 PEs and 140 traffic flows, and 54% overhead reduction for a smartphone chip specification that contains 25 PEs and 93 traffic flows.

4.5 A Non-intrusive and Reconfigurable Access Control to Secure NoCs

The work shown in [22] proposes a secure NoC architecture using firewall modules that provide three security properties: (i) *Accountability*, where each element in the system has to provide credentials that prove its identity; (ii) *Authenticity*, where only authenticated sources may access restrict resources in the system; and (iii) *Nonrepudiability*, where there is the guarantee that a traffic origin cannot be forged. The presumed NoC architecture uses packets for exchanging data, which contain a header with routing related information, as Figure 4.3 illustrates, and an independent circuit dedicated for firewall configuration.

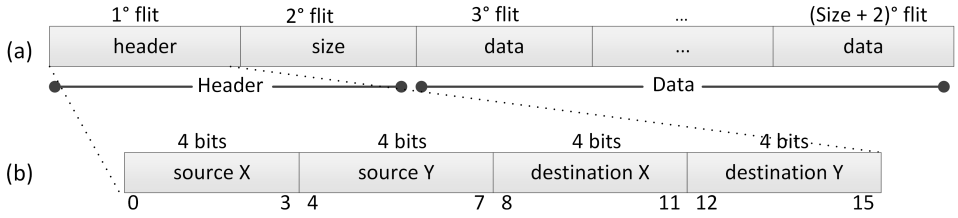


Figure 4.3: NoC packet composition for a 16-bit flit.

Firewalls can provide *Authenticity* by inspecting packet headers, as these contain the traffic initiator's source address. This address uniquely identifies elements in the system, thus providing the *Accountability* property. If the initiator is authorized to communicate with a destination, then traffic enters the NI of the target IP. Otherwise, the packet is rejected upon arriving at its intended destination. *Nonrepudiability* occurs by the firewall modules inspecting the source field information of the packet's header upon injection by the initiator IP. If the source address is different from the associated router's, then the packet is discarded before entering the NoC. Otherwise, it is normally injected.

The placement of the firewall modules occurs between the NI and router ports, which does not require altering existing NoC components besides the interconnections. This approach, illustrated in Figure 4.4, simplifies implementation as the NoC may incorporate security mechanisms through the firewall modules without significant architectural redesign.

Their model is validated using the Hermes NoC [42], a 2D mesh NoC described in VHDL with SystemC-based packet injectors. Results include area overhead and a synthetic traffic scenario that mixes legitimate, malicious and forbidden dataflows. Synthesis shows the firewall modules to be scalable to the NoC size with 12% to 16% of area overhead. Meanwhile, NoC communication occurs normally while detecting non-secure traffic flows.

The firewall placement strategy is similar to the one adopted in this work, as firewall modules reside between routers and NIs in a manner that does not require complex architectural changes.

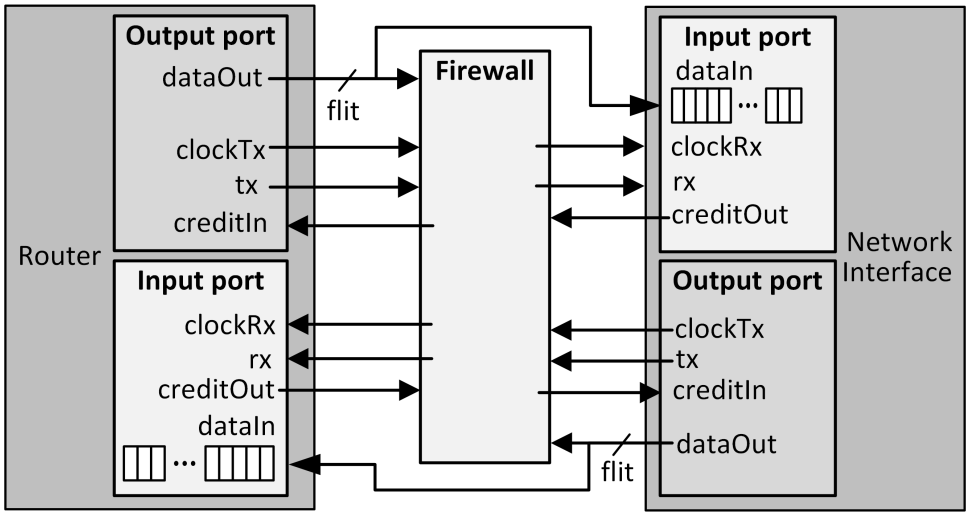


Figure 4.4: NoC port interface and firewall interconnection.

4.6 IP Protection of Mesh NoCs Using Square Spiral Routing

In contrast to other works that address data and system vulnerabilities in NoC-based MPSoCs, [36] proposes a mechanism for preventing the illegal distribution of IP cores.

While usual IP protection mechanisms employ the generation of a uniquely identifiable signature (known as watermarking) through FSMs, the authors argue that this approach is inefficient due to two reasons: (i) FSMs are deterministic by nature and a centralized implementation requires visibility of the entire network, while traffic in NoCs can be defined as stochastic with distributed routing decisions, making it difficult to use FSMs as a control for the entire network; and (ii) since each router in a NoC has an FSM, it would be possible to apply watermarking to each router. However, the overhead of implementing individually strong watermarking per router would be much larger than a solution that works at the network level. It is estimated that a 64-bit wide watermark increases router area by 6.1%, on average [16]. Using this approach on multiple routers would multiply the area overhead.

The proposed technique explores characteristics of routing algorithms to generate a system-wide watermark. Special authentication packets travel the NoC using Square Spiral Protection (SSP) routing, collecting encrypted path information from all visited IP cores. As the packet travels the NoC, each IP's network interface decodes its contents, appends the encrypted path information, and forwards the packet to the next router, completing a predefined spiral path, as shown in Figure 4.5.

As NoCs usually employ routing algorithms that aim to prevent *deadlocks* and performance improvements, certain routing paths may rarely occur. These unusual paths can then generate a signature, which defines a unique traveling path. Since the number of distinct spiral paths exponentially increases according to NoC dimensions, when a buyer *j* makes a purchase request, NoC designers randomly select SSP paths as an authorship

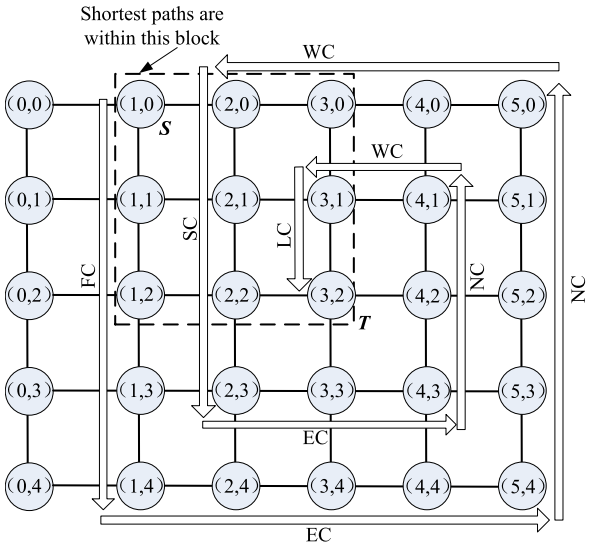


Figure 4.5: Spiral routing for IP core verification. (captured from [36])

watermark R_i for a NoC IP i. A watermark authority then randomly selects a secret key K_j to generate a unique signature $R_i^{K_j}$ that is used with 128-bit AES encryption. If a NoC IP i is then illegally distributed there is undeniable proof that buyer j is the IP's source.

The remainder of their work details attack scenarios and the improbability of signature collision, while emphasizing the validity of their approach to enforce IP authenticity in MPSoCs. Similar to the proposed work, the authors have identified routing techniques as a way to address security related problems in NoC-based architectures.

4.7 Packet security with path sensitization for NoCs

The authors of [12] propose **Packet-Security** (**P-Sec**), which is a packet validation scheme for use in compromised NoCs. Their work ensures communication integrity through fault tolerance, employing encoding to prevent malicious tampering of packet contents during transmission.

The threat model considers the existence of a hardware Trojan, as shown in Figure 4.6, which is capable of creating a side channel attack. In their work, they limit such attacks to fault injection at a NoC's links.

Their proposed solution consists of inserting Algebraic Manipulation Detection (AMD) between IP cores, providing end-to-end integrity and confidentiality of transmitted packets. Unlike traditional error detection techniques like Cyclic Redundancy Check (CRC), strong AMD codes cannot be masked into other valid code words for any other value generated by a side channel attack. Therefore, for applications working with sensitive data, AMD encoding

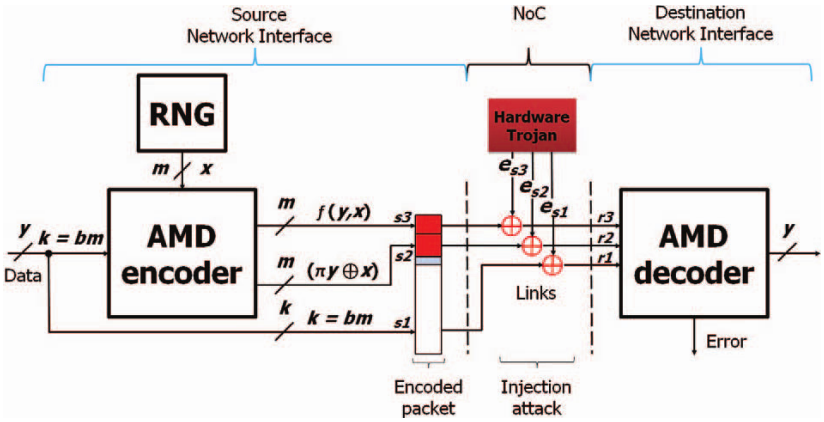


Figure 4.6: Threat model of a hardware Trojan. (captured from [12])

should be used, while non-critical applications could be encoded with simpler approaches (like CRC) for baseline protection.

While the proposed technique greatly enhances system security, results show that area overhead (300%), power overhead (200%), and latency overhead (200%) are significant due to the complexity of AMD encoding. However, since the encoding strength can be chosen according to application requirements, the cost of **P-Sec** can be minimized for non-critical situations.

4.8 Towards Risk Aware NoCs for Data Protection in MPSoCs

In [51], the authors propose the *Global Risk-Aware NoC* (*GRaNoC*), a NoC architecture able to monitor and evaluate the risk of the communication paths inside a NoC. This architecture can protect sensitive data by using low-risk paths, where the risk is defined by the probability of a malicious application of spying, denying communication or corrupting data at NoC hops. Like this work, [51] also employs the routing algorithm to enhance security.

Risk measurement considers notifications generated by firewall, modules which monitor the violation of system security rules. Usually, *GRaNoC* employs XY routing to define the paths for sensitive data, which is a well-known routing algorithm due to area and power efficiency. When a *risk threshold* exceeds a design time constraint, new routes for the sensitive path are set employing source-based routing, where the source IP defines a secure communication path.

GRaNoC is modeled in SystemC for scalability, performance, and security evaluation, using the SPLASH-2 benchmark for traffic generation. Scalability measures the impact of computing secure routing paths when security rules are violated. Four approaches are considered for route computation and compared to an exhaustive approach: (i) *Deterministic*, containing predefined alternative paths stored at design time; (ii) *Hop-based*, defining

a low risk path by backtracking the routing path from the destination back to the source; (iii) *Weighted-2D*, which is similar to the *Hop-based* approach but considers the risk of more than the immediate neighboring router; and (iv) *Bounded*, which works like *Weighted-2D* but restricts routes to generate a minimal path. The *deterministic* approach yields the lowest performance overhead since there is no route computation involved. Considering the dynamic approaches, *Hop-based*, *Weighted-2D* and *Bounded* offer similar performance and greatly reduce the overhead of an exhaustive approach by up to 75%.

Performance considers the NoC latency overhead when employing the security solution. While using the secure routes for communication have a low impact in *Deterministic* mode, the dynamic techniques impose a significant overhead due to the communication of neighboring routers for risk assessment. Each time a secure route is computed, the source queries routers in the communication path. Only after all routers report back to the source, the route is computed and then data is transmitted. This process can nearly double the NoC latency with small injection rates of 20%, and grows exponentially with higher communication volumes.

Lastly, security evaluation measures the efficacy of the secure technique for blocking unauthorized access to system elements and preventing *DoS* attacks from disrupting NoC communication. In this regard, the dynamic routing approaches yield higher efficacy than *Deterministic*, with the *Bounded* approach blocking all attacks used during evaluation.

4.9 Overview of Related Works

This section summarizes the previously discussed works on security for NoC-based MPSoCs. Table 4.1 presents an overview of the characteristics commonly found in all studied related works, and how these characteristics relate to the proposed work.

The classification criteria for *Services* relates to the key properties in information security, defined in Section 3.1, namely: **1** - Accountability; **2** - Authenticity; **3** - Integrity; **4** - Availability; **5** - Confidentiality; **6** - Auditing; and **7** - Nonrepudiability. Most of the studied works employ *Accountability* and *Authenticity* in MPSoCs. To prevent exploiting these two security measures, *Nonrepudiability* usually accompanies the security solution to avoid credential forgery.

The *Implementation* characteristic differentiates how the security solution was implemented in the system, which can be: **H** - Hardware-based; and **S** - Software-based; All studied works employ at least a hardware-based component for the security solution. The software components, when present, are usually used in conjunction with the hardware elements, handling configuration and/or monitoring of the software layers of the MPSoC.

Granularity relates to the nature of the content filtered by the security services, with each subsequent level complementing the former, namely: **C** - Communication level filtering; **M** - Memory access filtering; and **T** - Task role filtering. Due to the nature of the studied works, *Communication level filtering* is always present with some works adopting detailed verification (e.g., address ranges in memory accesses and identification of initiator software tasks).

Finally, the *Technique* relates to the techniques employed in the security service, which may be: **F** - Firewalls for implementing security rules; **AT** - Access tables that contain rule checking parameters; **EC** - Encoding of sensitive content; and **R** - Routing techniques targeting security; Although firewalls and access tables may be similar, the former filters communication flows from accessing system resources, while the later evaluates requests to system resources, considering the initiator's role.

Table 4.1: Overview of related works.

Title	Services	Implementation	Granularity	Technique	Reference
Security Effectiveness and a Hardware Firewall for MPSoCs	**1**, **2**	**H**	**C**, **M**	**AT**	[28]
Elastic Security Zones for NoC-Based 3D-MPSoCs	**1**, **2**, **5**, **6**, **7**	**H**, **S**	**C**	**F**, **AT**	[57]
3D-LeukoNoC: A Dynamic NoC Protection	**3**, **4**	**H**, **S**	**C**	**F**, **AT**	[56]
Automatic ILP-based Firewall Insertion for Secure Application-specific Networks-on-Chip	**1**, **2**, **4**, **7**	**H**, **S**	**C**, **M**, **T**	**F**, **AT**	[31]
A Non-intrusive and Reconfigurable Access Control to Secure NoCs	**1**, **2**, **7**	**H**	**C**	**F**	[22]
IP Protection of Mesh NoCs Using Square Spiral Routing	**1**, **6**, **7**	**H**	**C**	**R**	[36]
Packet security with path sensitization for NoCs	**1**, **3**	**H**	**C**	**EC**	[12]
Towards Risk Aware NoCs for Data Protection in MPSoCs	**1**, **3**, **4**	**H**	**C**, **M**	**F**, **AT**	[51]
This work	**3**, **4**, **5**	**H**, **S**	**C**	**R**	-

In this work, **3**, **4** and **5** are obtained by using safe routing paths, enhancing protection from *DoS* and timing attacks, which are further discussed in Section 5. The security solution is implemented by **H** (routing tables) and **S** (routing algorithm) elements, which operate at communication (**C**) level, based purely on a routing (**R**) technique.

5. THREAT MODEL

This work is based on two premises: (i) that sensitive applications communicate over the NoC in MPSoC systems, and are therefore vulnerable to interference by malicious applications; and (ii) that protection from software attacks is achieved by communication level protection, e.g., by defining safe communications paths for sensitive applications. Therefore, a routing algorithm that is capable of defining secure routing paths for sensitive applications executing in MPSoCs should enhance the overall system protection.

Considering the existence of an MPSoC composed by a set of IPs interconnected by a shared NoC, the MPSoC is capable of executing parallel applications that can use either a shared memory or message exchanging for communication. Each application can be split into smaller pieces of code, called tasks, and mapped into different and several IPs to enhance system performance.

However, splitting an application into distributed tasks in an MPSoC forces the communication of data through the NoC. Applications that handle sensitive system information are, therefore, forced to communicate sensitive data through the interconnect fabric. The communication path between a pair of IPs that execute a critical application is called a *sensitive path*. Figure 5.1 illustrates an MPSoC with nine IPs in a 3x3 2D Mesh NoC, highlighting a sensitive communication path between the IPs *S* (source) and *D* (destination).

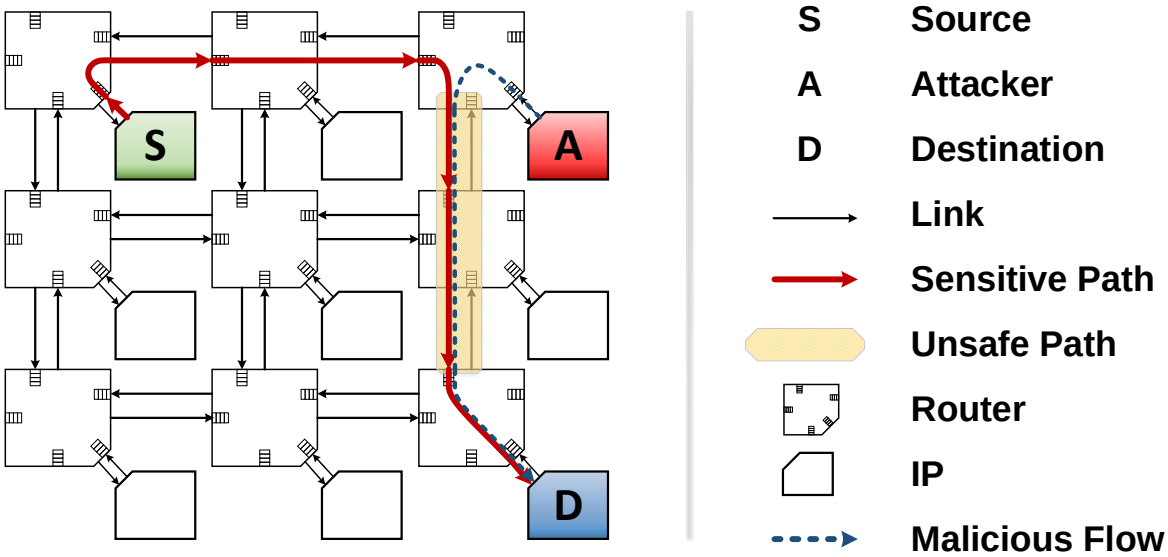


Figure 5.1: A source process **S** communicates with a destination **D** over an insecure path due to a *DoS* or Timing Attack from an attacker **A** to **D**.

An attacker *A*, aware of the system architecture and the mapping strategy in the MPSoC, performs a software attack by infecting an IP through a malicious code. The infection can occur by downloading a malware directly off the Internet into the chip or by modifying the external memory used by the MPSoC to store applications. The NoC is considered se-

cure, which means the attacker cannot tamper its resources (e.g., routers) to compromise system security. Application mapping also homogeneously assigns tasks into IPs so that distinct applications do not share an IP block's resource.

It is desirable to have the infected IP located inside the sensitive path to increase the efficiency of the attack. Knowledge of the routing algorithm by the attacker is a requirement if the malicious application performs timing attacks to extract sensitive information from the MPSoC, but not mandatory while performing *DoS* attacks, whose goal is to disrupt system operation by overloading resources.

The attacks considered in this work are timing and *DoS*, previously described in Sections 3.4 and 3.3, respectively. Both attacks create malicious traffic flows originating from an infected IP to compromise the MPSoC security. Timing attacks exploit the communication collision between the sensitive and malicious traffics to infer secret information off the traffic pattern. *DoS* attacks, however, simply flood the NoC resources to disrupt the sensitive path with useless communication.

6. SECURITY AWARE ROUTING

The routing algorithm approach proposed by this work consists of two concepts: (i) *Security Zones*; and (ii) the *Routing Algorithm*. The following sections describe these concepts as well as their applicability in this work.

6.1 Security Zones

A security zone is a physical space (continuous or disrupted, Figure 6.1) that wraps IPs that execute critical applications. IPs that belong to a security zone are considered trusted among them. The task mapping of critical applications inside the MPSoC defines the shape of the security zone. Certain IP blocks might not be assigned to any security zone, e.g., idle system resources, or shared memories that contain information from multiple applications.

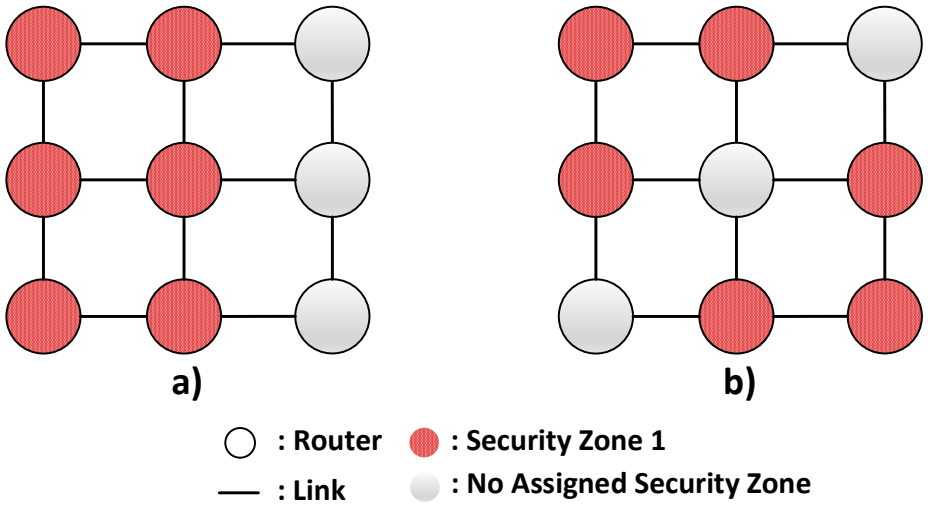


Figure 6.1: Possible security zones configurations: **a)** A continuous security zone; **b)** A disrupted/noncontinuous security zone.

A set of IP cores (P) that executes a critical application defines a security zone SZ, such that the elements $p_i, p_j \in P$ are considered secure and trusted. A transaction from p_i to p_j, where $i, j \in [0, N-1)$ with N representing the number of IP blocks in the system, is called sensitive and must be performed inside SZ. However, typical NoC routing algorithms may force the route of the sensitive path outside the SZ. Three communication scenarios regarding security zones are shown in Figure 6.2 among source (S) and destination (D) pairs:

1. ***Full intra-zone communication (FIZ)***: S and D are in the same SZ. The sensitive path is **completely** inside the SZ, e.g., the path from *IP1* to *IP2*;

2. ***Partial intra-zone communication (PIZ)***: *S* and *D* are in the same *SZ*. However, the sensitive path is **partially** inside the *SZ*. ***PIZ*** occurs in disrupted security zones, or for irregular *SZ* shapes, when the routing algorithm forces out the sensitive traffic, e.g., the path from *IP3* to *IP4*;

3. ***Inter-zone communication (IZ)***: *S* and *D* are in different *SZ*, e.g., the path from *IP5* to *IP6*.

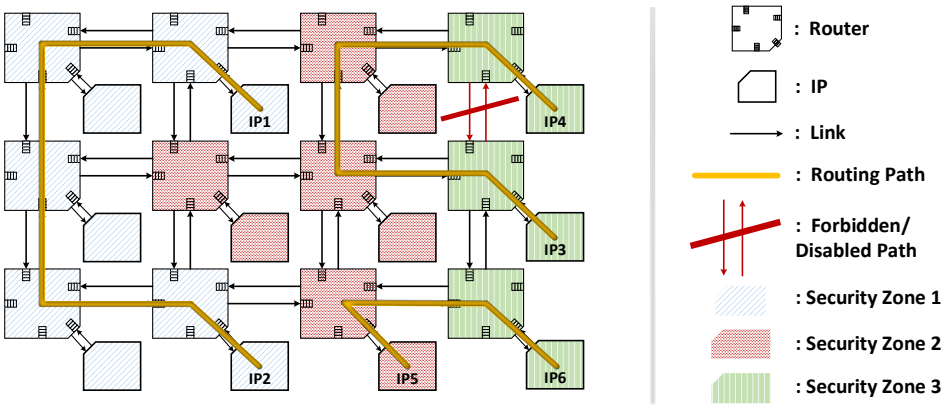


Figure 6.2: The three communication scenarios with security zones: ***FIZ*** (*IP1* to *IP2*); ***PIZ*** (*IP3* to *IP4*); ***IZ*** (*IP5* to *IP6*). This scenario defines a synthetic mapping of three applications on a 4x3 2D Mesh NoC.

FIZ communication is the most secure situation, as sensitive communications are contained in secure elements of the system. Whenever possible, traffic flows should adhere to this model. ***PIZ*** occurs when the security zone is noncontinuous, or when the routing algorithm forbids a communication path, usually to avoid a route that would lead to a routing deadlock. Lastly, ***IZ*** should occur due to communications: (i) among distinct applications in the system; or (ii) between an application and IPs that does not belong to any security zone (e.g., a shared memory).

6.2 Routing Algorithm

Searching for secure paths on MPSoCs demands high flexibility. As shown in Figure 6.2, establishing secure paths inside the *SZ* requires in some cases non-minimal paths, e.g., the path from *IP1* to *IP2*. It heavily depends on the shape of the security zone.

Several routing algorithms have been studied before in the areas of high performance and fault-tolerant MPSoCs. *Segment-based Routing* (*SBR*) [41] and *Region-based Routing* (*RBR*) [26] have been used in conjunction to find non-minimal paths efficiently. *SBR* is responsible for deadlock prevention while *RBR* computes the routing entries.

6.2.1 Segment-based Routing

SBR logically partitions the NoC into subnets and subnets into segments. Since each segment is independent, this allows the placement of bidirectional turn restrictions locally within each segment.

Segments are characterized by a list of interconnected routers and links. Figure 6.3 illustrates four segments labeled **S1** to **S4** computed for an arbitrary NoC N, where the first segment contains the routers $\{A, B, C, D\}$ and the links $\{1, 2, 3, 4\}$. Each router and link $\in N$ belong to one and only one segment.

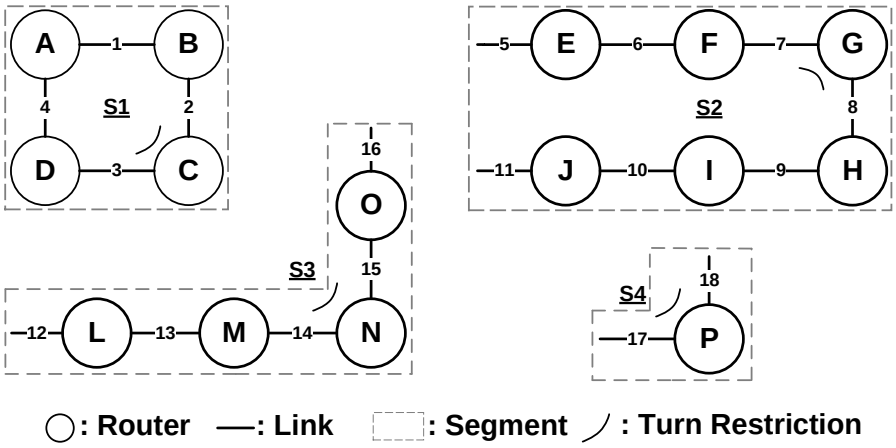


Figure 6.3: Four possible *SBR* segment configurations with routers and links.

Furthermore, segments are grouped into subnets. A subnet is a set of segments that connects to the rest of the network (other subnets) through only one link. Figure 6.4 shows two configurations for the same set of routers. In Figure 6.4a the segmentation process results in segments S1 to S3 and a single subnet. By removing one of the links from the NoC topology (*link 5*), the result is two segments (S1 and S2) and two subnets (Subnet1 and Subnet2) interconnected by *link 2*, as shown in Figure 6.4b.

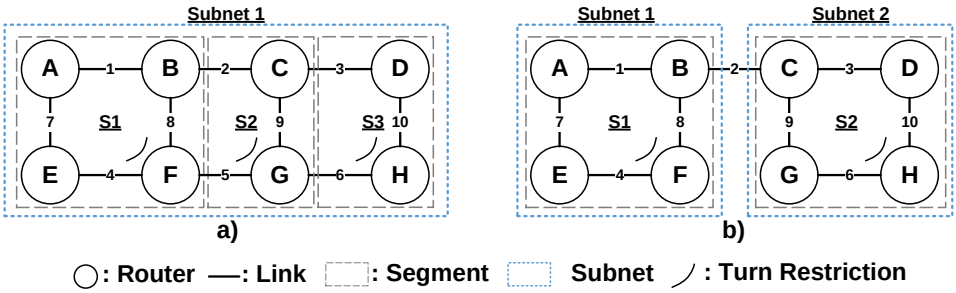


Figure 6.4: Segment connectivity defines subnets in *SBR*: **a)** segments S1 to S3 form a single subnet; **b)** link 5 is removed, so only link 2 connects S1 to S2, forming two subnets.

SBR computation is composed of two phases: (i) segment computation; and (ii) routing restriction placement. At segment computation, *SBR* partitions the NoC into segments

comprising routers and links. Each segment is characterized by a turn restriction to avoid routing deadlocks. Figure 6.5 illustrates the three possible types of segments computed by *SBR*: (i) *starting*, which starts and ends at the same router, forming a loop; (ii) *regular*, which starts at a link, contains at least one router, and ends at another link; and (iii) *unitary*, which contains a single link that does not allow traffic.

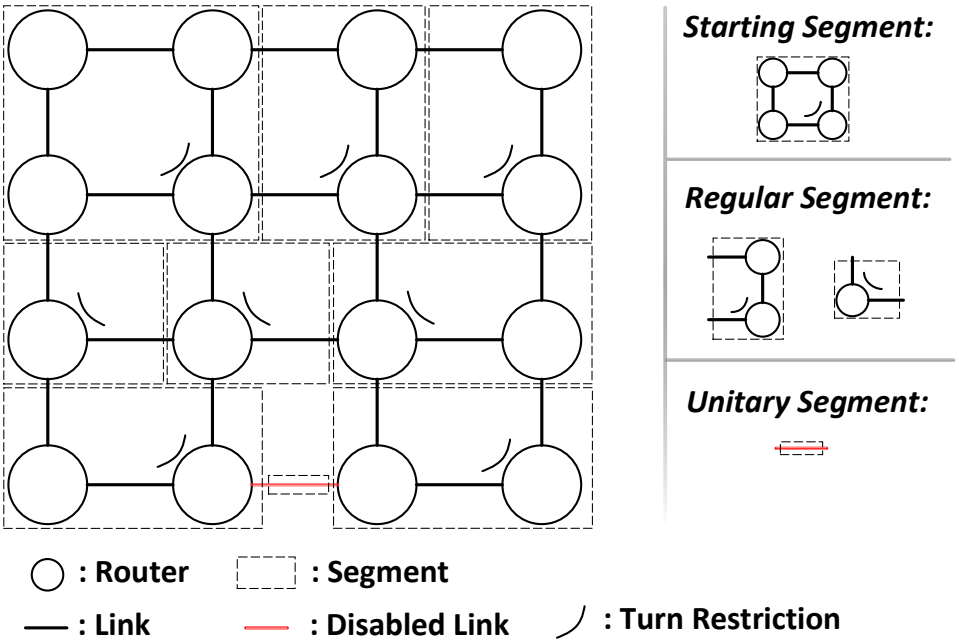


Figure 6.5: Segments and turn restrictions computed by the *SBR* algorithm in a 4x4 2D Mesh NoC (based on [41]).

SBR aims to create segments that minimize the number of elements per segment to reduce the occurrence of ***unitary*** segments, as these segments restrict certain links in the topology. During the execution of *SBR*, routers and links can be in the following states:

- *not visited*: initially, all routers and links are in this state;
- *visited*: a router or a link becomes *visited* once it belongs to an already computed routing segment;
- *tvisited*: during the process of computing segments, routers and links may be temporarily marked as *tvisited*;
- *starting*: a router is marked as *starting* if it is the first router chosen to compute a segment within a subnet;
- *terminal*: a router is marked as *terminal* if no segment is found through at least one of its links.

Algorithms 6.1 and 6.2 demonstrate how *SBR* defines segments and subnets in the NoC topology. Table 6.1 contains a description of each function used by the these al-

gorithms. Once the entire NoC is partitioned into segments, bidirectional routing restrictions are placed at each segment according to its type.

```
input  : nocTopology, startingRouter
output: segmentList
begin
    segmentList ⟵ ∅
     subnet ⟵ 0
     segIndex ⟵ 0
     router ⟵ nocTopology[startingRouter]
     router.starting ⟵ True
     router.subnet ⟵ subnet
     router.visited ⟵ True
     router.starting ⟵ True
     segList[] ⟵ ∅
     segment ⟵ segList[segIndex]

    while router ≠ null do
        if find (nocTopology, router,
          segment, subnet) then
            segIndex ⟵ segIndex + 1
             segment ⟵ segList[segIndex]
        else
            router.terminal ⟵ True
             router ⟵ nextVisited()
        end
        if router == null then
            router ⟵ nextNotVisited()
        end
        if router ≠ null then
            subnet ⟵ subnet + 1
             router.starting ⟵ True
        end
    end
end
```

Algorithm 6.1: SBR

```
input  : nocTopology, router, segment,
         subnet
begin
    router.tvisited ⟵ True
     segment.append(router)
     links = suitableLinks(router)

    if links == ∅ then
        router.tvisited ⟵ False
         segment.remove(router)
         return False
    end
    foreach link ln ∈ links do
        ln.tvisited = True
         segment.append(ln)
         nrouter = aTop(router, ln)
         if (nrouter.visited and
         nrouter.subnet = subnet) or
         find(nrouter, segment, subnet))
         then
            ln.visited ⟵ True
             router.visited ⟵ True
             ln.tvisited ⟵ False
             router.tvisited ⟵ False
             return True
        else
            ln.tvisited ⟵ False
             segment.remove(ln)
        end
    end
    segment.remove(router)
     router.tvisited ⟵ False
     return False
end
```

Algorithm 6.2: *Find* procedure used by SBR to compute segments.

The estimated computational cost of *SBR* depends on its implementation and its phases. Considering a 2D mesh NoC, the algorithms previously presented results in each link being visited only once yielding $O(l)$ cost, with l as the number of links in the NoC topology. Adding routing restrictions only requires evaluating the segments computed in the first phase, so that the computational cost of this phase is $O(s)$, where s is the total number of segments.

Table 6.1: *SBR* functions used for segment computation.

find(topology, router, segment, subnet)	Defines a segment for the specified topology, starting at a given router and subnet (Algorithm 6.2).
nextVisited()	Finds a router marked as visited, belonging to the current subnet, and with at least one link not marked as visited.
nextNotVisited()	Finds a router not marked as visited, not marked as terminal, and attached to a terminal router.
aTop(router,link)	Returns a new router that is connected to the specified router through the specified link.
suitableLinks(router)	Returns a list links for the specified router that are not marked as *visited* or *tvisited*.

Once the turn restrictions are in place, a routing algorithm, such as *RBR*, computes the routing paths between the source and destination pairs.

6.2.2 SBR Security Zone Awareness

SBR capabilities can be used to calculate the segments and turn restrictions based on the security zones of the system. Figure 6.6 shows two cases of segment computation for the same six routers of a NoC topology. Depending on the NoC segments computed by *SBR*, the communication path between *S* and *D* can be either ***PIZ***, as shown in Figure 6.6a; or ***FIZ***, as illustrated in Figure 6.6b.

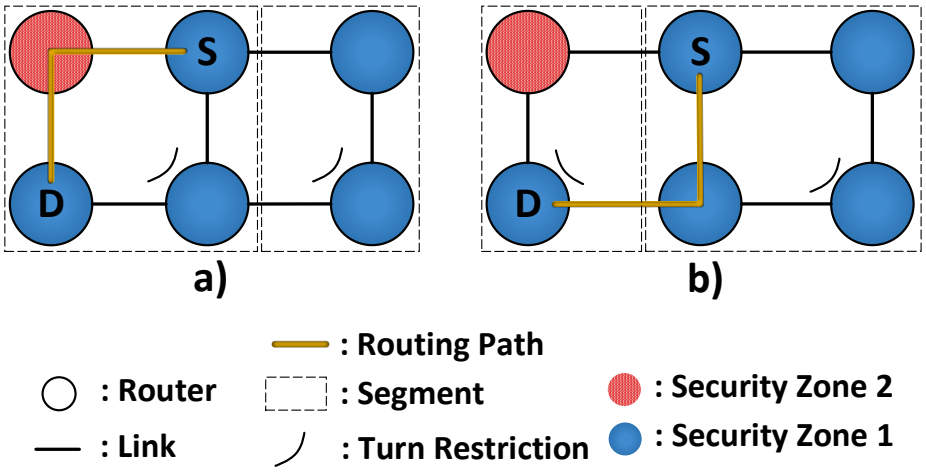


Figure 6.6: Two *SBR* segment computations: **a)** the path between S and D goes through an insecure element due to a routing restriction; **b)** the set of routing restrictions in the segments enables a secure path between S and D.

This work proposes *SBR* Security Zone Awareness (*SBR-SZA*), which uses the *SBR* algorithm to place turn restrictions that favor the occurrence of ***FIZ*** scenarios. The segments are tailored to a security zone so that *SBR-SZA* creates the smallest possible segments that contain elements from the same security zone.

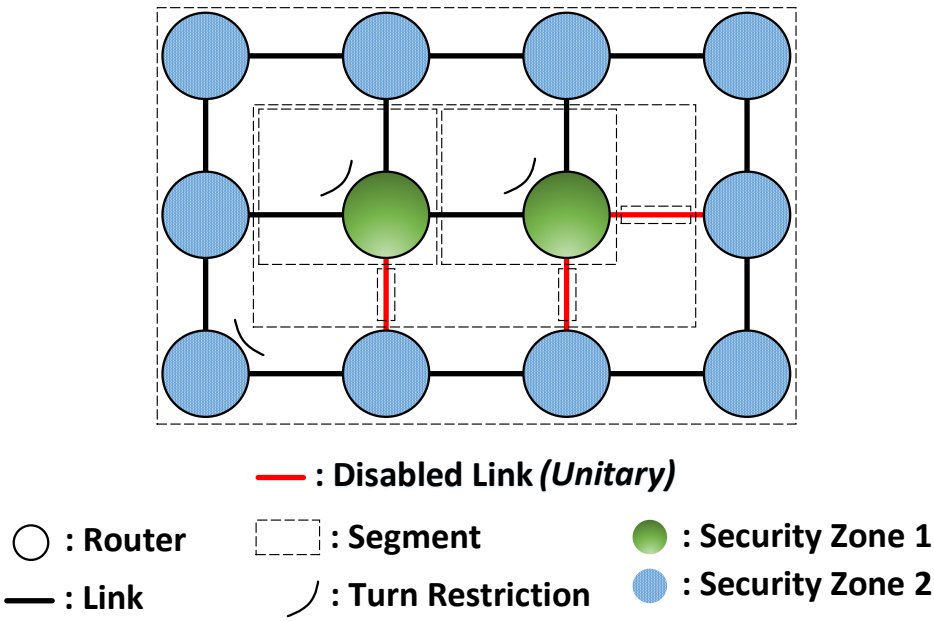


Figure 6.7: Segmentation example with *SBR-SZA*. While ***PIZ*** scenarios do not occur, the segmentation results in *unitary* segments.

While *SBR-SZA* should favor the occurrence of ***FIZ*** scenarios, a performance impact is a possibility since the segmentation can lead to a greater occurrence of *unitary* segments, as Figure 6.7 illustrates.

The computational cost of *SBR-SZA* is the same of traditional *SBR*. The difference is that instead of computing as small as possible segments, *SBR-SZA* attempts to close segments by visiting routers from the same security zone of the initial router used in the *find* procedure, shown in Algorithm 6.2.

6.2.3 Region-based Routing

RBR takes the turn restrictions computed by *SBR* to find paths between all sources and destinations in the NoC. As a result, the routing entries for each router are generated. The main advantage of *RBR* is that a single routing entry can represent a path to more than one destination. Additionally, destinations can be packed using a technique called *interval routing* [18, Chapter 4], in which only the delimiting coordinates of routers in a rectangular region are necessary to describe a routing entry. These characteristics of *RBR* can reduce the size of routing tables significantly.

There are three steps in the computation of *RBR*, shown in Figure 6.8. The first step is *routing computation* from each NoC router to every other router. Path selection is performed according to the designer goals and system requirements, e.g., security. The computed paths are stored in the routing table of the router. Each entry is represented by 3-tuple (N, d, S), where N is the packet input port, d the destination address, and S is the router output port used to reach d.

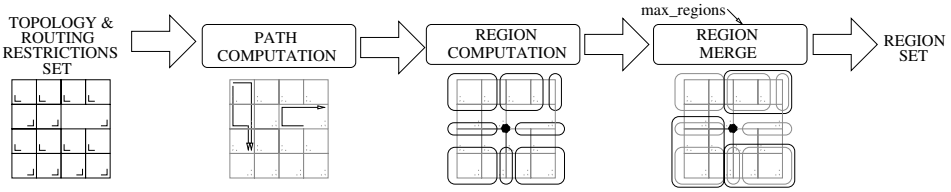
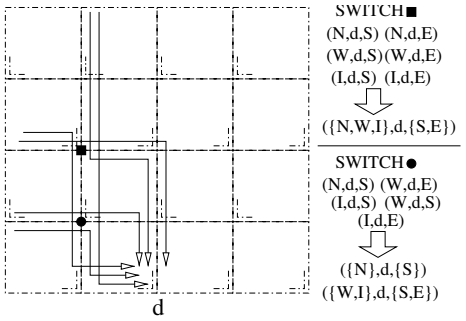
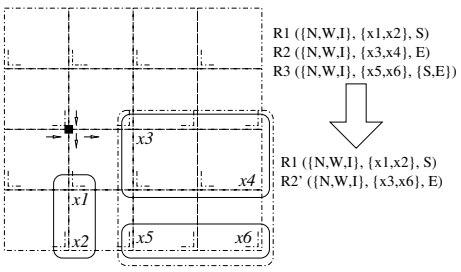


Figure 6.8: The three computation steps of *RBR* algorithm. A NoC topology with its configurations is the sole input.

The second step is *region computation*, where multiple entries are joined based on the input and output port values. Figure 6.9a shows the paths computed by *RBR* algorithm to a destination d from two different source routers. The entries (N, d, S), (W, d, S), and (I, d, S), of router ■, which have the same set of output ports for the same destination can be grouped. As a result, a single routing entry $(\{N, W, I\}, d, S)$ is stored. Analogously, the 3-tuples with the E output port can be packed into $(\{N, W, I\}, d, E)$. Further packing of these entries can be done based on their sets of input ports and destination, leading to a single entry $(\{N, W, I\}, d, \{S, E\})$. This result represents adaptive routing, as more than one output port exists to reach the same destination. Lastly, routing entries can be grouped by destination if they possess the same set of input and output ports, forming regions as shown in Figure 6.9b (R1 to R3).



(a) Paths computed in *RBR* from two sources to a destination ***d***.



(b) Regions defined for a router. Overlapping regions R2 and R3 can be merged into a single entry, albeit sacrificing adaptability.

Figure 6.9: An example of paths and regions computed by RBR algorithm. (captured from [26])

The third step, *region merge*, merges overlapping regions to reduce the quantity of routing entries. Figure 6.9b illustrates two regions (R2 and R3). Even though the set of destinations are distinct from these regions, each region is adjacent to one another, and R2's

output ports is a subset of R3's. By sacrificing adaptability, these regions can be merged into a single region (R2'), reducing the number of routing entries.

Algorithm 6.3 demonstrates *RBR*'s main procedure for computing regions. Table 6.2 contains a description of functions used by *RBR*. Once all regions are computed, the routing tables of NoC routers are generated.

```
input  : nocTopology
output: routingRegions
begin
    foreach src ∈ nocTopology.routers do
        foreach dst ∈ nocTopology.routers do
            if src == dst then
                skip
            else
                foreach inputPort ∈ src.ports do
                    outputPort = getPath(nocTopology, src, dst)
                     if outputPort ≠ null then
                        src.entries.add(inputPort, dst, outputPort)
                    end
                end
            end
        end
        packOutputPorts(src.entries)
         packInputPorts(src.entries)
         packDestinations(src.entries)
         mergeRegions(src.entries)
    end
end
```

Algorithm 6.3: RBR

Regarding the computational cost of *RBR*, checking the paths from all sources to all destinations is $O(n^2)$, where n represents the number of routers in the NoC. Due to this inherent costly characteristic of *RBR*, it is necessary to use an efficient pathfinding algorithm, such as *Dijkstra* shortest path or heuristic graph searching algorithms, further discussed in Section 6.2.5.

6.2.4 Modeling of Security Zones

This work models routers and links of the NoC as a graph. Vertices correspond to routers and their associated IPs, while edges correspond to links that interconnect routers. Each vertex can belong to a security zone, and each edge has a positive weight that is set according to the pathfinding algorithm iteration.

Table 6.2: *RBR* functions used for region computation.

getPath(topology, src, dst)	This procedure is responsible for computing the path between src and dst for the specified topology.
packOutputPorts(entries)	Groups the specified routing entries by output port. Two entries can be merged if they have the same input port and destination.
packInputPorts(entries)	Groups the specified routing entries by input port. Two entries can be merged if they reach the same destination through the same set of output ports.
packDestinations(entries)	Groups the specified routing entries by destination. Two entries can be merged if they have the same set of input and output ports.
mergeRegions(entries)	Merges the specified routing entries, defining regions in the NoC. Entries can be merged if they share the same subset of input and output ports.

When computing the path from a source router to a destination, the weight of edges adjusts to favor paths to other routers from the same security zone as the source. In Figure 6.10, there are three paths from *S* to *D*. The topmost path traverses an insecure element in the context of *S*. Therefore, it has a higher edge cost than edges that lead to a router of the same security zone of *S*. The middle path traverses two secure routers, reaching *D*. Meanwhile, the bottommost path has a turn restriction that forbids traffic from *S* to *D*.

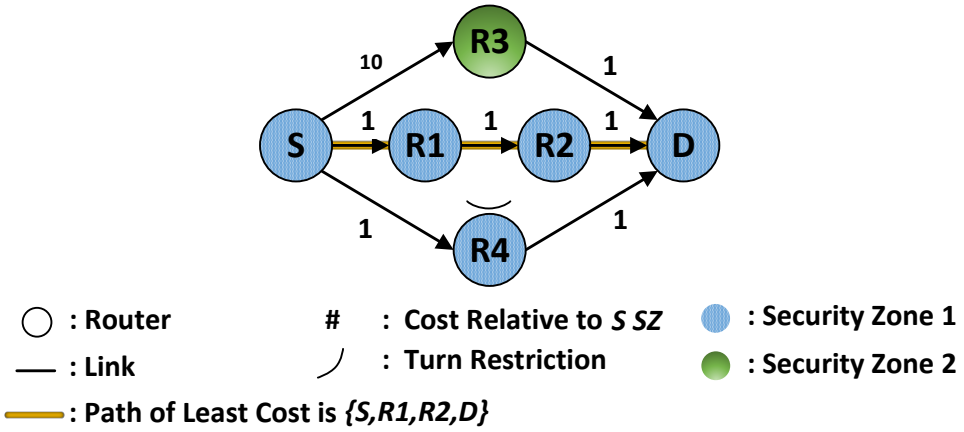


Figure 6.10: Modeling of the paths from a source *S* to a destination *D*. Traversing a different security zone leads to a higher path cost.

Determining the edge cost is about system requirements. It could represent the cost to protect a NoC package by a cryptographic module while traversing to an insecure zone, the cost of passing a firewall, or the risk of a given path. There is a broad spectrum of possibilities for defining the edge cost in this model.

6.2.5 Path-Finding Algorithm

The choice of the path-finding algorithm used for computing the source and destination pairs for the routing algorithm is independent of *RBR*. The only constraints are: (i) it has to consider the turn restrictions computed by *SBR*; and (ii) it must take into consideration the edge cost between hops in the NoC, according to the model defined in Section 6.2.4. An approach that satisfies these requirements is the *greedy best-first search* family of graph searching algorithms, which consider the entire path cost between a source and a destination to compute paths.

The most widely known form of best-first search is called **A* search** [48, Chapter 3.5]. It expands vertices from a source towards a destination to quickly obtain a solution. The cost to reach a destination from a vertex v is given by $f(v) = g(v) + h(v)$, where $g(v)$ is the cost to reach a vertex, and $h(v)$ is the cost to get from a vertex to a destination. $h(v)$ defines a heuristic function that when admissible, i.e., it does not overestimate the cost to reach a destination, greatly reduces the search space and therefore the time complexity of the algorithm.

Regarding two-dimensional mesh NoCs, computing the heuristic function can be done in $O(1)$ time complexity by using Manhattan Distance [48, Chapter 3.6], shown in Equation 6.1, where d corresponds to the destination. Figure 6.11 illustrates the heuristic function applied to a two-dimensional mesh NoC.

$$h(v) = |v.X - d.X| + |v.Y - d.Y|. \tag{6.1}$$

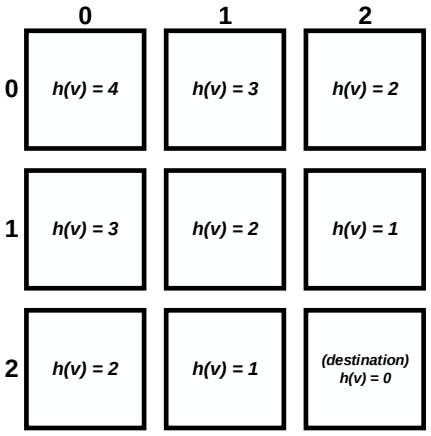


Figure 6.11: Manhattan Distance computed for a 3x3 mesh NoC from all routers to a destination in coordinates (2,2).

In *A* search*, path finding between a source and a destination only expands vertices that lead to shortest paths, as shown in Figure 6.12. Vertices that deviate from the optimal solution define a frontier that will only be expanded once vertices that yield a shorter path

are evaluated. Since Figure 6.12 does not illustrate turn restrictions and considers that each hop costs exactly 1, multiple optimal paths exist.

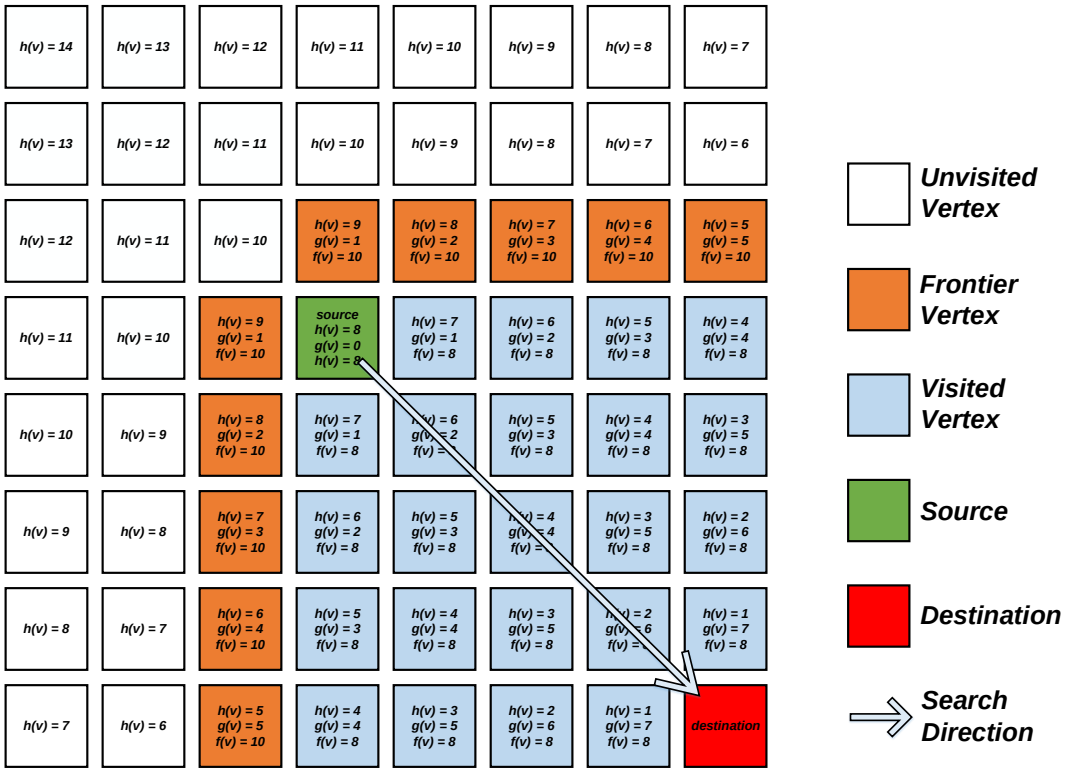


Figure 6.12: *A* search* in an 8x8 mesh NoC. Only paths that lead to the destination are expanded.

When considering the context of security zones and turn restrictions employed on this work, the set of optimal paths can be more restricted. In these scenarios, *A* search* can approach linear time complexity as the expansion of graph vertices is bound by the length of the optimal solution. Algorithms 6.4 and 6.5 demonstrate an implementation of *A* search* considering a two-dimensional NoC and the modeling of security zones discussed in Section 6.2.4.

```
input : topology, source, destination
begin
    foreach router ∈ topology do
        router.hn ⟵ |(router.X − destination.X)| + |(router.Y − destination.Y)|
        if router == source then
            router.gn ⟵ 0
            router.fn ⟵ router.hn
        else
            router.gn ⟵ ∞
            router.fn ⟵ ∞
        end
    end
    currentRouter ⟵ source
    priorityQueue[] ⟵ ∅
    while currentRouter ≠ NULL do
        if currentRouter == destination then
            path = extractPath(destination)
            outputPort = Port(Path[0], Path[1])
            return outputPort
        end
        updateNeighbors(topology, source, currentRouter, priorityQueue)
        currentRouter.visited = true
        currentRouter ⟵ priorityQueue[0]
    end
    return NULL
end
```

Algorithm 6.4: *A** Path-Finding

```
input : nocTopology, source, currentRouter, priorityQueue
begin
    outputLinks ⟵ currentRouter.links
    foreach link ∈ outputLinks do
        neighbor ⟵ topology.neighbor(currentRouter, link)
        if neighbor ≠ NULL then
            if neighbor.visited then
                continue
            end
            if source.zone == neighbor.zone then
                linkWeight ⟵ 1
            else
                linkWeight ⟵ IZ_EdgeCost
            end
            cost ⟵ source.gn + linkWeight
            if cost > neighbor.gn then
                continue
            else
                neighbor.previous ⟵ currentRouter
                neighbor.gn ⟵ cost
                neighbor.fn ⟵ neighbor.gn + neighbor.hn
                priorityQueue.append(neighbor, neighbor.fn)
            end
        end
    end
end
```

Algorithm 6.5: Neighbor Update function.

7. EXPERIMENTAL SETUP AND EVALUATION CRITERIA

This work adopts an abstract model of a parameterizable NoC specified in the Java programming language to evaluate the proposed secure routing technique. This model supports the configuration of the NoC topology *at design time* using a graphical tool (Figure 7.1), with configurable parameters such as the security zone of each router and the operational status of individual links (enabled or disabled). The inter-zone traversal cost for computing the routing paths among source and destination pairs can also be configured to satisfy system design specifications.

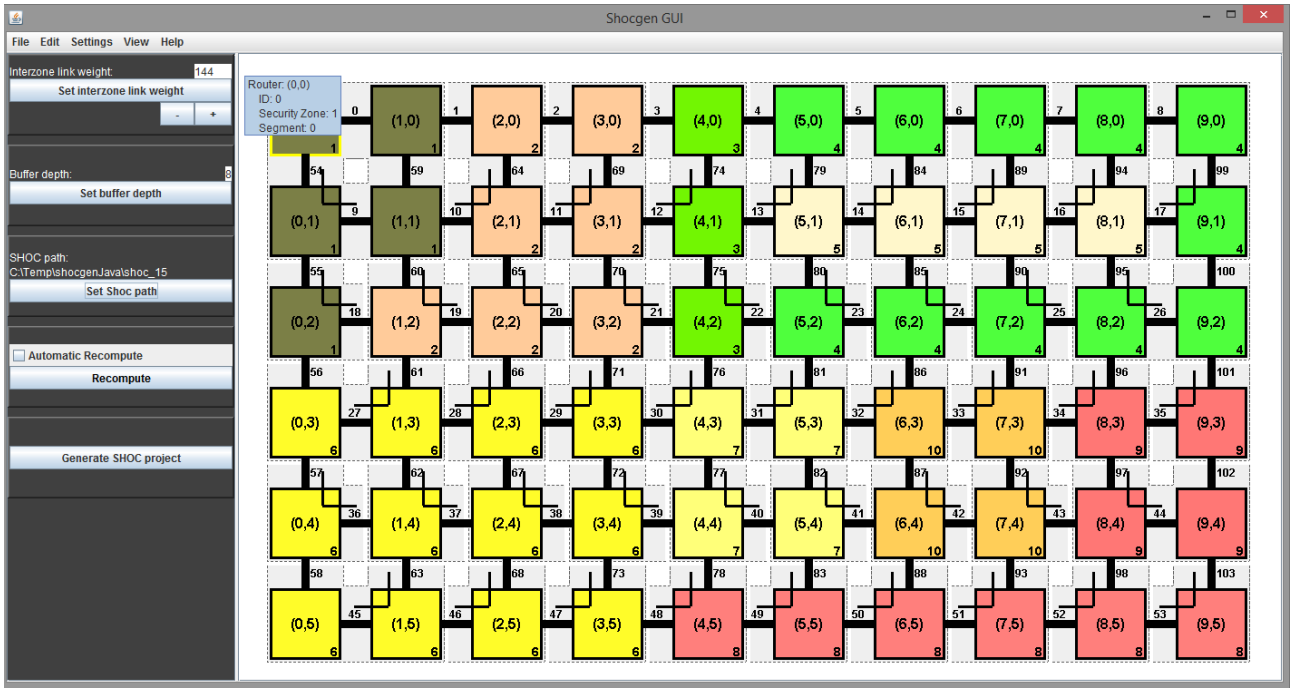


Figure 7.1: Abstract NoC configuration tool.

The NoC configuration tool implements both *SBR* and *RBR* algorithms according to their specifications discussed in Section 6.2. Once the NoC configuration is complete, the tool generates the routing entries for each NoC router according to the specified design time constraints. Additionally, it is also possible to output the configured NoC to a simulation platform, such as the ones discussed in Section 7.1.

7.1 Network Simulation Platforms

This work adopts two platforms for network simulation: (i) a SystemC parameterizable NoC with TLM precision called Secure-enhanced Hybrid on-chip Communication (SHOC) [58][54]; and (ii) a SystemC parameterizable NoC with Register-Transfer Level (RTL) precision called Hermes [42].

The SHOC platform enables the customization of the following parameters: (i) NoC dimension; (ii) number of ports per router, which enables customized topologies; (iii) placement of port buffers (at input, at output, or both); (iv) arbitration; and (v) routing algorithm. Communication among NoC elements occurs by traffic generators that simulate the behavior of IP blocks.

A limitation of SHOC's TLM simulation is the precision of packet communication among NoC elements. Links are modeled as having variable bit widths, so that transferring a whole packet among routers takes a single clock cycle, regardless of its size. This results in a packet switching model similar to *Store-and-Forward*, where packets are forwarded to the next hop once there is enough buffer space to store an entire packet. Buffers are also modeled in the same manner, having variable bit width (enough to store a whole packet) per buffer slot.

Despite these limitations, the TLM offered by the platform is sufficient to validate the proposed secure routing technique. SHOC is capable of modeling contention due to resource sharing, or employ parameterizable communication delays; e.g., to simulate a data protection mechanism to protect packet contents. Moreover, SHOC offers faster simulation speeds than flit-accurate or cycle-accurate models.

The Hermes NoC contains a similar set of configurable parameters as the ones present in SHOC, encompassing: (i) NoC dimension; (ii) arbitration; (iii) flit width; (iv) buffer depth; and (v) routing mechanism. Routers contain five communication ports. Ports *East*, *West*, *North*, and *South* connect one router to another, forming a two-dimensional mesh topology. The *Local* port is used exclusively for connecting to an IP block.

Differently from SHOC TLM, the Hermes NoC used in this work is described in SystemC-RTL, which is a higher precision communication model, using *Wormhole* switching where a minimum transmission unit (*flit*) is transmitted among NoC elements per clock cycle. Packets are split into *flits* and sent from one router to another. Links are modeled at design time with fixed bit widths, specified by the *flit* size. Buffers are located at each one of the five router input ports and have fixed bit widths as well, according to the *flit* size. The routing mechanism of Hermes also employs a clock accurate multistage state machine for arbitration and switching.

These characteristics of Hermes NoC enables greater simulation precision for measuring network latency. Modeling of contention and resource sharing is accurate with an RTL implementation, enabling to measure the impact of the proposed secure routing technique on network latency.

7.2 System Simulation Platform

To evaluate the impact of the proposed routing technique on real application workloads, this work employs *Gem5* [9]. This simulator utilizes a modular discrete event model for simulating system architectures, encompassing processors, memories, peripherals, and intercommunication interfaces (e.g., buses and NoCs). Gem5 supports most commercial CPU's Instruction Set Architectures (ISA) available (ARM, ALPHA, MIPS, Power, SPARC and x86), with ARM, ALPHA and x86 enabling to boot an unmodified Linux kernel.

Gem5 employs a wide variety of simulation modes. CPUs can implement *AtomicSimple*, a very simplified, event driven simulation model that responds to input requests, or *O3*, which is a very complex model with pipelines and out-of-order instruction execution. The system in Gem5 can be modeled in two ways. *System-call Emulation* (*SE*) emulates most system-level services, which avoids the need to model devices or an OS. Alternatively, *Full-System* (*FS*) mode models a complete system, including the OS and devices. Lastly, Gem5 enables two memory modes. *Classic* mode employs a fast and reconfigurable memory system, while *Ruby* enables the modeling of a variety of cache coherence memory systems.

Out of the three CPU ISAs that support Linux kernel simulation, ARM is the only actively developed version which supports (*FS*) mode [27]. Alpha architecture is mostly deprecated and therefore unsupported, while x86 only supports *SE* mode. To perform an accurate simulation on application performance, this work adopts an *O3* ARM-based architecture using *FS* simulation and a *Classic* memory mode.

The architecture used for simulations consists on differently sized NoCs with a single CPU connected to each router local port, as shown in Figure 7.2. Each IP is characterized by an ARMv7a processor with 64 KiB of L1 cache for data, 32 KiB L1 cache for instructions, and 2 MiB L2 cache, running Linux Kernel 3.3.0-rc3gem5+ to execute user applications. However, Gem5's ARM ISA does not provide a NoC implementation. NoCs in Gem5 are only available for ALPHA and x86 ISAs, using the *Ruby* memory system which is unavailable for ARM ISA. Due to this limitation, this work uses a NoC model that operates over Gem5 with flit precision, as presented in [13]. This NoC model employs packet-based congestion by means of a normal probability distribution, where the values for mean and standard deviation were obtained from previous packet-based NoC works [14] [20].

The NoC model is based on Hermes NoC characteristics, described in Section 7.1, which consist on: (i) 5 clock cycles for packet switching; (ii) 1 clock cycle to forward flits; and (iii) wormhole switching. Routing a packet *P* between a source *S* and a destination *D* pair, therefore, takes into consideration a Hermes NoC implementation along with a congestion

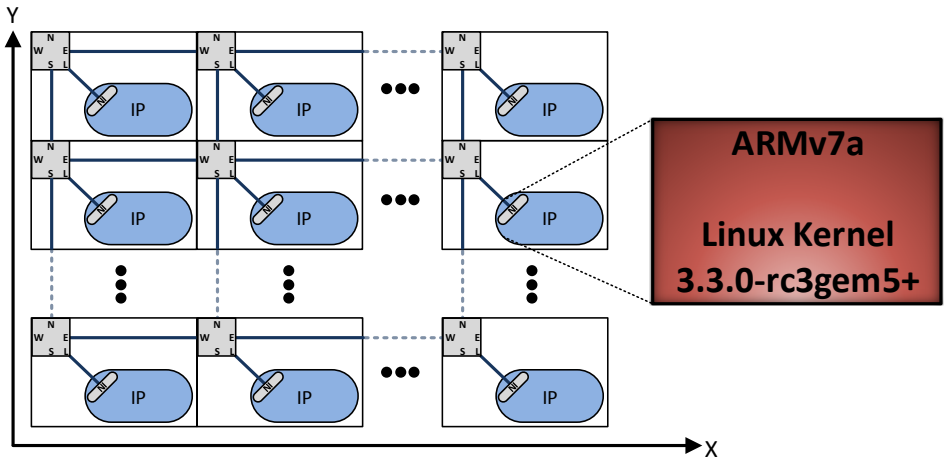


Figure 7.2: General system architecture used for Gem5 simulation. ARMv7a CPU running a Linux Kernel are connected to each router in the NoC topology.

probability model C, such that the communication delay L is given by Equation 7.1:

$$L(S, D, P) = H + PathLen(S, D) + Size(P) + C \tag{7.1}$$

where *PathLen* corresponds to the routing path length between S and D, and H to the router arbitration latency, which is fixed at 5 for the Hermes NoC.

The NoC schedules packet delivery according to the delay computed by Equation 7.1 and the packet injection time T_i. Since Gem5 is a discrete event simulator, a packet delivery time T_d is the sum of T_i and $D(S, D, P)$, as shown in Equation 7.2.

$$T_d(S, D, P) = T_i + L(S, D, P) \tag{7.2}$$

This flexibility for configuring NoC constraints allows the routing model shown in this work to be implemented for Gem5 simulations. The delay among source and destination pairs posses delays which depend on the paths computed by *RBR*, according to the mapping of applications in the system. Section 7.3 details the set of applications used for evaluating this work.

7.3 Evaluation Scenarios

To evaluate the proposed secure routing technique for real workload scenarios, this work uses the *NASA Numerical Aerodynamic Simulation* (*NAS*) *Parallel Benchmark* (*NPB*) [43]. Each application in *NPB* has varying communication and computation requirements. To explore a varied set of communication requirements, six applications out of the eleven provided by the benchmark were used, namely:

- **EP - Embarrassingly Parallel**: Measures floating point performance. A series of random numbers is generated and spread among multiple processes for computation. The end result is then verified by each process, without significant interprocessor communication;
- **MG - MultiGrid kernel**: Solves the 3D discrete Poisson differential equation. Tests both short and long range structured communication;
- **CG - Conjugate Gradient**: Computes the eigenvalues of a large, sparse, symmetric and positive definite matrix. Complementary to **MG**, this application tests irregular long range communication;
- **FT - 3D Fast Fourier Transform**: Solves 3D partial differential equations using Fast Fourier Transformations. This application requires considerable communication for matrix operations. Tests long distance communication performance;
- **IS - Integer Sort**: Performs a sorting operation. This application tests both computation and communication performance; and
- **LU - Lower-Upper Gauss-Seidel solver**: Computes fluid dynamics calculations by solving nonlinear partial differential equations. This application is very sensitive to communication performance, as it sends a very large amount of small messages.

This set of *NPB* applications can be split into multiple tasks for parallel processing. To generate scenarios with various security zones and varying amounts of communication patterns, we consider NoC configurations with enough IPs to support all six *NPB* applications executing simultaneously. By varying the degree of parallelism on each application, four base scenarios were created. Table 7.1 illustrates the NoC configurations and degrees of parallelism applied to all *NPB* applications.

Table 7.1: NoC and *NPB* configurations.

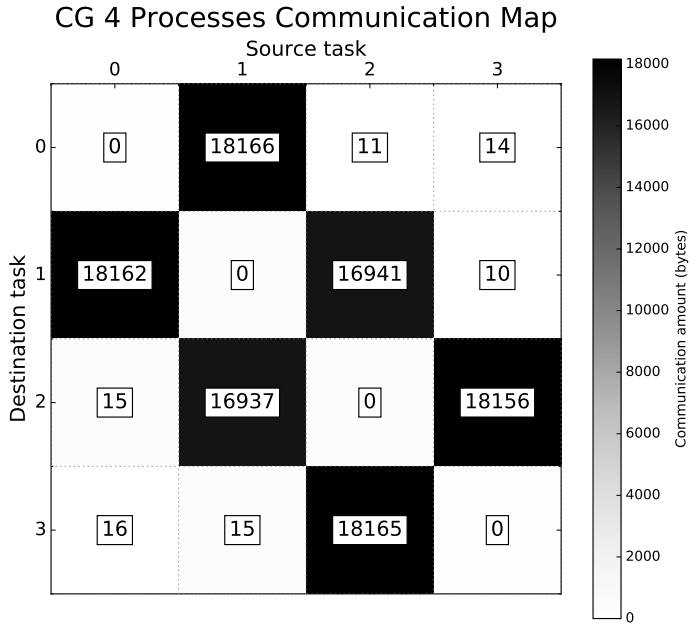
	Processes per application						**Total processes**
NoC dimension	**EP**	**MG**	**CG**	**FT**	**IS**	**LU**	
5x5	4	4	4	4	4	4	24
7x7	8	8	8	8	8	8	48
10x10	16	16	16	16	16	16	96
14x14	32	32	32	32	32	32	192

Since all benchmark applications execute independently and the NoC employed by Gem5 models contention probabilistically, it is possible to execute each application as a standalone simulation. The communication delays for an application depends solely on the NoC model and task mapping, the latter defining the length of the communication paths between source and destination pairs in the NoC. To generate NoCs scenarios with security zones and *NPB* applications, some preliminary configuration steps are necessary:

1. A communication map of all applications is extracted. Each of the 24 application configurations, shown in Table 7.1, are executed in Gem5. The mapping of tasks at this step is irrelevant, since only the communication dependency among tasks is necessary;

2. The Gem5 NoC is capable of generating a report that details the communication volume among NoC elements. This report is used to generate a communication map of all tasks of an application, as shown in Figure 7.3.

3. Once the communication map is obtained, applications are mapped on the NoC configurations described in Table 7.1.



CG tasks		
Source	Destination	Bytes sent
CG 0	CG 0	0
CG 0	CG 1	18166
CG 0	CG 2	11
CG 0	CG 3	14
CG 1	CG 0	18162
CG 1	CG 1	0
CG 1	CG 2	16941
CG 1	CG 3	10
CG 2	CG 0	15
CG 2	CG 1	16937
CG 2	CG 2	0
CG 2	CG 3	18156
CG 3	CG 0	16
CG 3	CG 1	15
CG 3	CG 2	18165
CG 3	CG 3	0

Figure 7.3: Communication Map for CG application with four processes. The map illustrates the amount of bytes exchanged among all tasks of the application.

A mapping algorithm then defines the distribution of tasks on each NoC configuration according to their communication dependencies. The resulting mappings also correspond to the security zones. The mapping strategies and application mappings adopted in this work is discussed in Section 7.4.

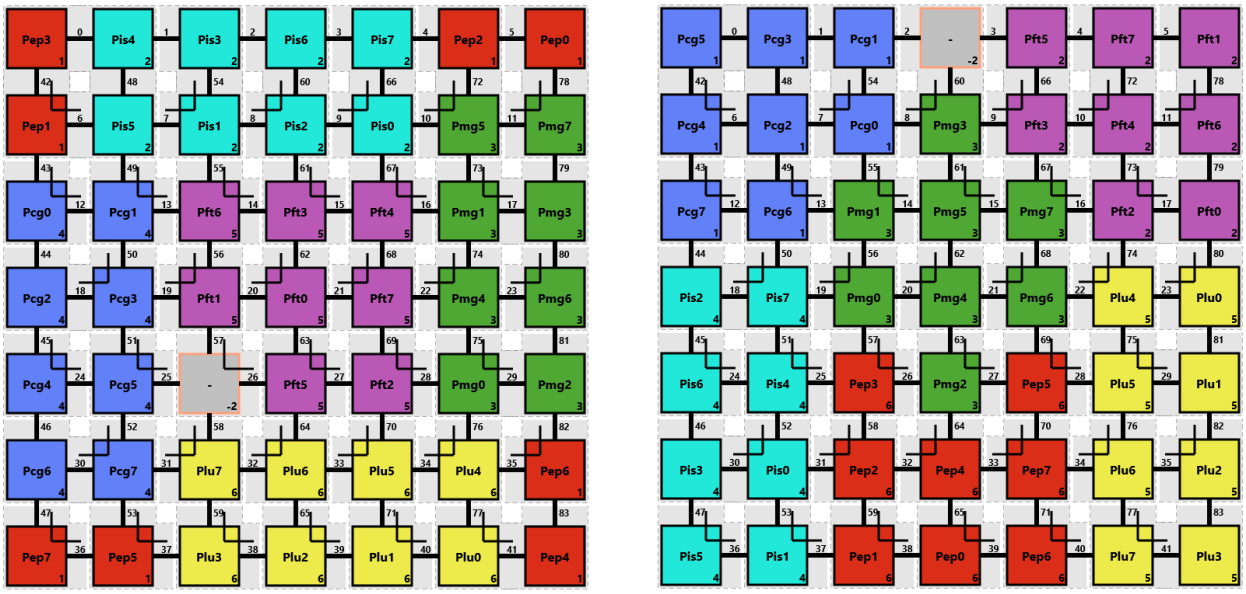
7.4 Task Mapping

Once the communication model is obtained, the set of applications is mapped using a framework called Communication Analysis for Embedded Systems (CAFES) [38]. This framework is capable of mapping application tasks into IP blocks connected to a NoC.

CAFES enables applications to be described with models that consider aspects such as computation and communication.

The *NPB* applications executed in Gem5 produce a communication map among all tasks, as shown in Section 7.3. This map is used in the communication model in CAFES for defining task mapping on the various NoC configurations. Tasks are mapped according to their communication requirements, placing highly communicating tasks closer to one another in order to increase application performance.

However, the default mapping approach employed in CAFES does not take into consideration the security aspects explored in this work, as tasks are grouped solely on their communication requirements. This mapping characteristic may lead to security zone fragmentation, which is an undesirable situation as it increases ***PIZ*** routes. Figure 7.4a illustrates how a security zone (encompassing application **EP**) is fragmented after mapping with CAFES. This fragmentation occurs because tasks have different communication requirements. Highly communicating tasks, such as **LU**, communicate orders of magnitude more than lowly communicating tasks, e.g., **EP**. Figure 7.5a illustrates this difference.

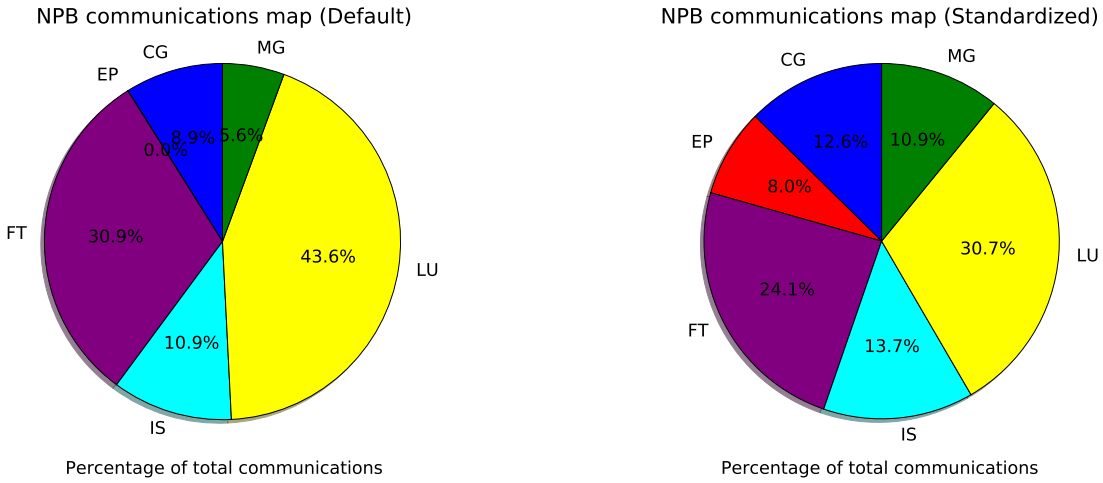


(a) Default Mapping.

(b) Standardized Mapping.

Figure 7.4: A 7x7 NoC with *NPB* applications split into 8 tasks. Application **EP** is fragmented using Default mapping, creating ***PIZ*** routes. The Standardized approach overcomes this undesired effect.

To better illustrate the communication dependency model regarding each application, we apply Standardization (also known as Z-score normalization [39]) on the communication model dataset. The communication model's features are rescaled according to Equation 7.3, where μ corresponds to the communication amount average, and σ to the standard



(a) Default communication model.

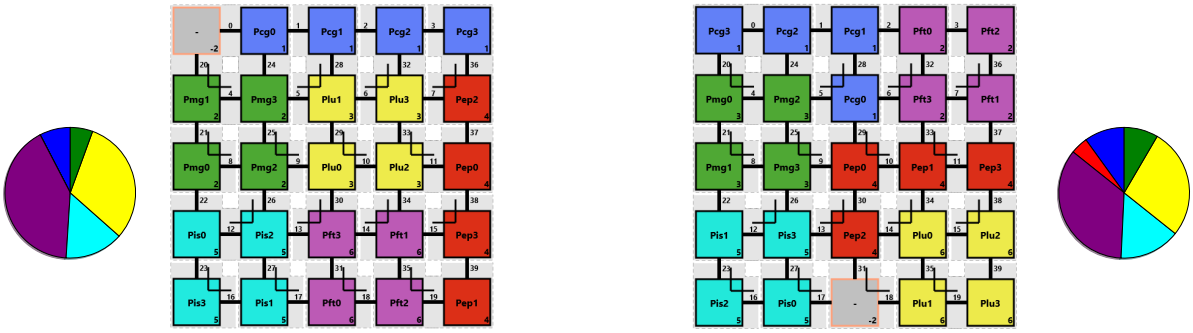
(b) Standardized communication model.

Figure 7.5: A 7x7 NoC communication distribution. Standardization masks the absolute differences between datasets and better illustrates the communication dependency model.

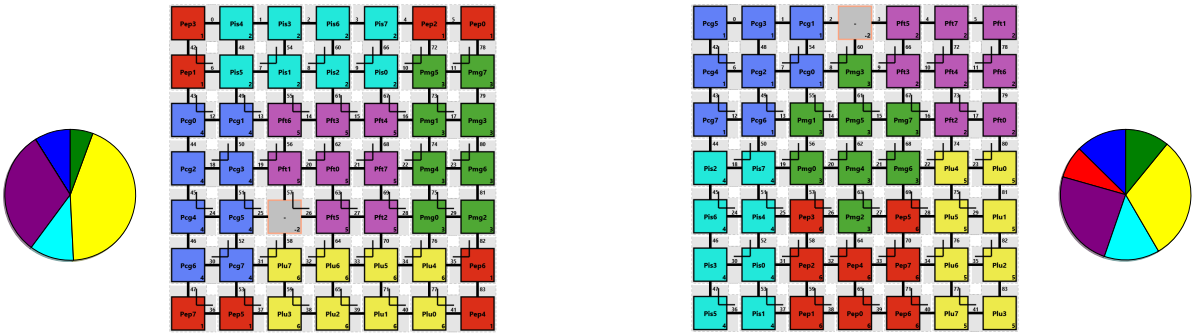
deviation. Standardizing the communication model masks the absolute differences between each *NPB* application, resulting in the distribution shown in Figure 7.5b and mapping shown in Figure 7.4b.

$$z = \frac{x - \mu}{\sigma} \tag{7.3}$$

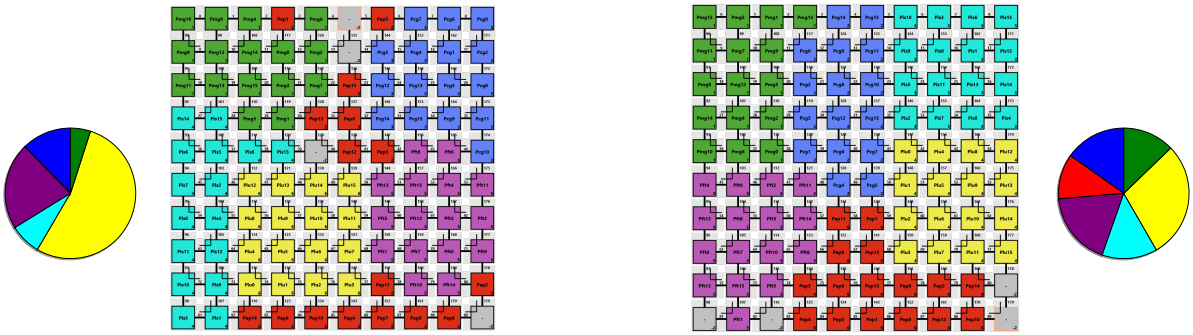
Considering the Default and Standardized mapping approaches and the four NoC configurations shown on Table 7.1, this results in eight scenarios for the *NPB* applications. Figure 7.6 illustrates all mapping configurations that will be used for evaluating this work, along with the communication amounts of each application.



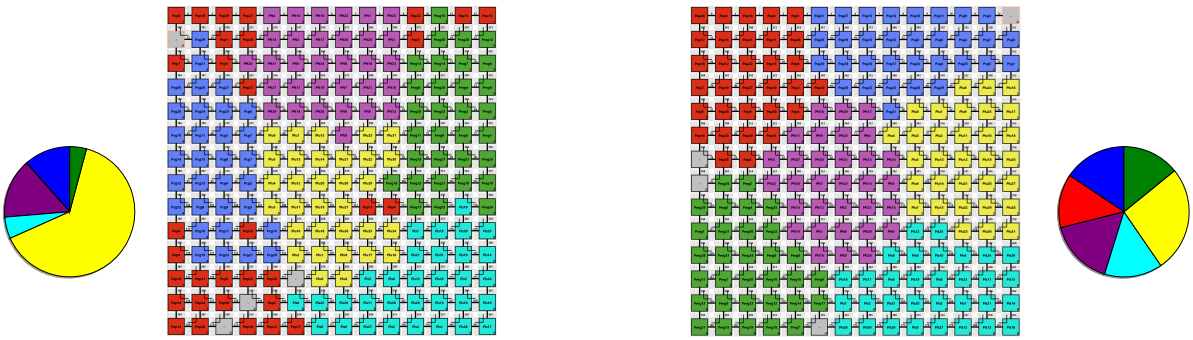
(a) 5x5 NoC Default Mapping. (b) 5x5 NoC Standardized Mapping.



(c) 7x7 NoC Default Mapping. (d) 7x7 NoC Standardized Mapping.



(e) 10x10 NoC Default Mapping. (f) 10x10 NoC Standardized Mapping.



(g) 14x14 NoC Default Mapping. (h) 14x14 NoC Standardized Mapping.

Figure 7.6: *NPB* application mapping on all four NoC configurations with two mapping strategies: Default and Standardized. Pie charts illustrate the communication amount for each application.

7.5 Evaluation Overview

Each evaluation of the scenarios shown in Section 7.4 consists of a set of preliminary steps:

- *Initial router for segment computation*: since segments in *SBR* are sequentially computed, varying the initial router (seed) used for segment computation can yield different segmentation configurations as it was previously shown in Figure 6.6. By checking different seeds, it is possible to verify which configurations offer a higher occurrence of ***FIZ*** routes;
- *Segment computation*: after setting the seed for segment computation, algorithm *SBR* is executed to place bidirectional turn restrictions to achieve deadlock-free routing. This step generates two configurations, using traditional *SBR* and *SBR-SZA*;
- *Routing computation*: once the turn restrictions are in place, *RBR* computes the routes from all source and destination pairs, generating the routing entries for each NoC router;
- *Evaluation of obtained configuration*: once *SBR* and *RBR* compute the NoC configuration, an assessment of the obtained configuration is performed.

Table 7.2: Summary of evaluated scenarios.

Scenario	NoC dimension	Mapping	SBR seeds	Segmentation modes	Configurations per scenario
NPB 4 processes	5x5	Default	25	*SBR* / *SBR-SZA*	50
NPB 8 processes	7x7	Default	49	*SBR* / *SBR-SZA*	98
NPB 16 processes	10x10	Default	100	*SBR* / *SBR-SZA*	200
NPB 32 processes	14x14	Default	196	*SBR* / *SBR-SZA*	392
NPB 4 processes	5x5	Standardized	25	*SBR* / *SBR-SZA*	50
NPB 8 processes	7x7	Standardized	49	*SBR* / *SBR-SZA*	98
NPB 16 processes	10x10	Standardized	100	*SBR* / *SBR-SZA*	200
NPB 32 processes	14x14	Standardized	196	*SBR* / *SBR-SZA*	392
				Total scenarios:	1480

Table 7.2 contains a summary of all tested configurations for each scenario. The amount of configuration scenarios is obtained by combining all *SBR* seeds with all *Segmentation Modes*. Three aspects of the proposed secure routing approach are evaluated: (i) the scalability of the *RBR* routing tables; (ii) the occurrence of ***FIZ*** routing scenarios; and (iii) the impact on *NPB* applications execution time given the proposed routing mechanism. Section 8 shows a discussion on the evaluation methodology and results of these aspects.

8. RESULTS AND EVALUATION

This section presents the results obtained from the evaluation procedure described in Section 7. The results shown in the following sections consist on an evaluation of routing table scalability, ***FIZ*** routing occurrence, and *NPB* applications execution time.

8.1 Routing Table Scalability Evaluation

Routing table scalability evaluation consists of measuring how many routing entries are generated at each router in the NoC. It is desirable to have fewer entries per router to reduce the necessary routing table memory. Since this work considers 2D Mesh-based NoCs, *Width* and *Height* are the number of routers that correspond to the 2D NoC width and height, respectively. Moreover, the 2D Mesh NoC adopted by this work consists of routers with the same amount of input and output ports P

By taking into consideration bi-dimensional NoC dimensions, it is possible to calculate the size of each routing entry. Equations 8.1 and 8.2 demonstrate how to calculate, in bits, the size of each routing entry of a router.

$$EntrySize = \underbrace{2 \times \lceil \log_2 P \rceil}_{\substack{\text{Input and} \\ \text{output ports}}} + \underbrace{Coordinates}_{\text{Destination region}} \tag{8.1}$$

$$Coordinates = X_{min} + Y_{min} + X_{max} + Y_{max}, where \begin{cases} x_{min} = \lceil \log_2 Width \rceil \\ y_{min} = \lceil \log_2 Height \rceil \\ x_{max} = \lceil \log_2 Width \rceil \\ y_{max} = \lceil \log_2 Height \rceil \end{cases} \tag{8.2}$$

Due to the modeling of security zones with non-minimal distance routing paths, it is expected that additional *RBR* routing entries are necessary to accommodate routes within security zones. This behavior is shown in Figures 8.1 and 8.2, which illustrates the average size of routing entries per router in each scenario.

Additionally, Figures 8.1 and 8.2 demonstrate that varying the seed used for *SBR* or *SBR-SZA* computation slightly impacts the amount of routing entries necessary per router. Since the placement of bidirectional turn restrictions can limit the available routing paths to reach a group of destinations, this results in routes sharing the same paths, which *RBR* packs into fewer routing entries.

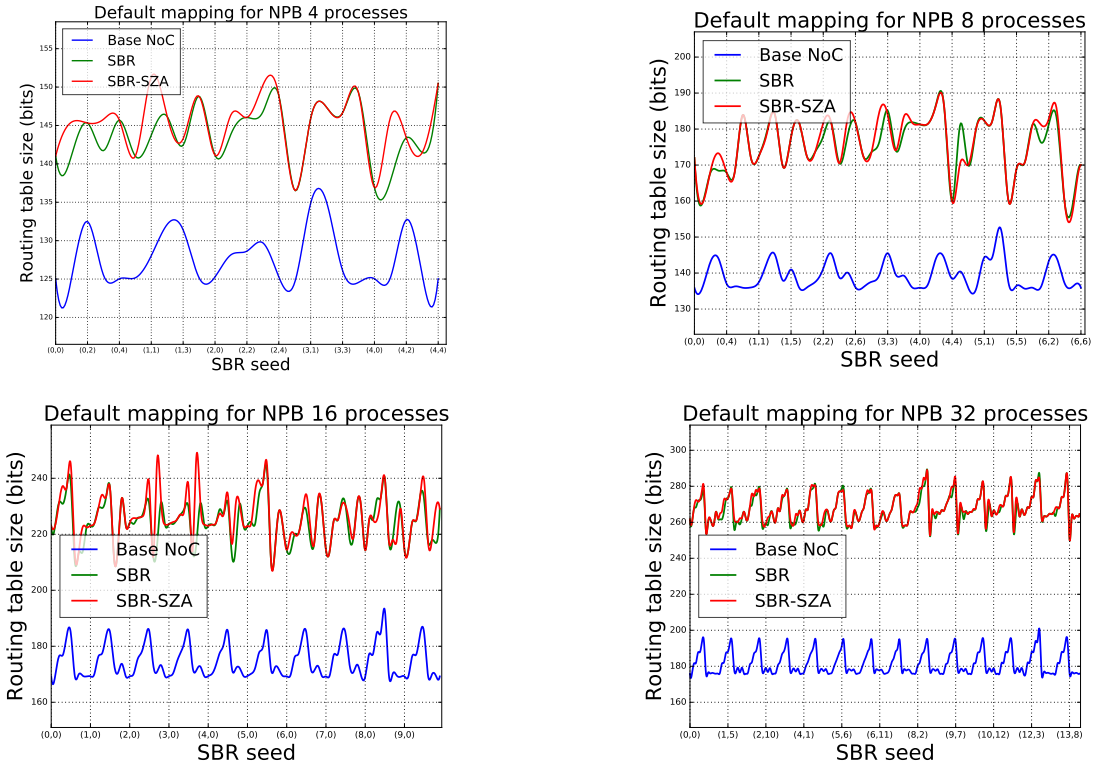


Figure 8.1: Average size of routing entries, in bits, per router per *SBR* seed using Default Mapping. Both segmentation modes are used (*SBR* and *SBR-SZA*) and compared with a Base-NoC without security zones.

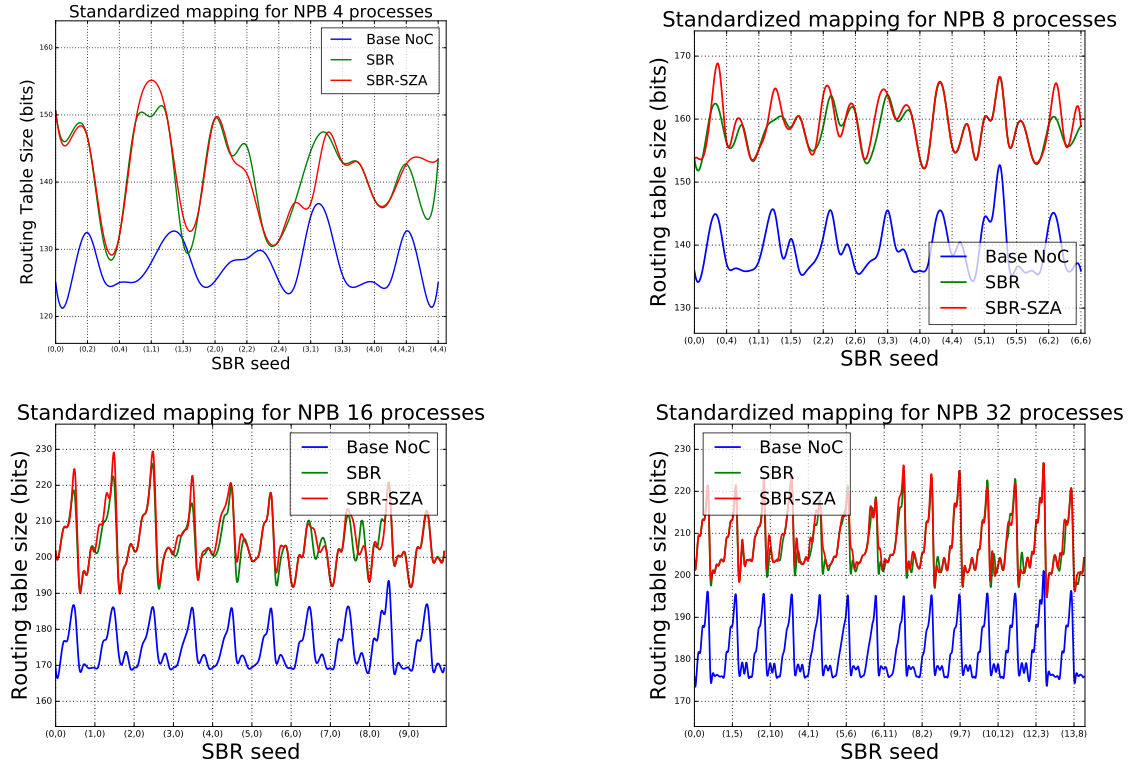
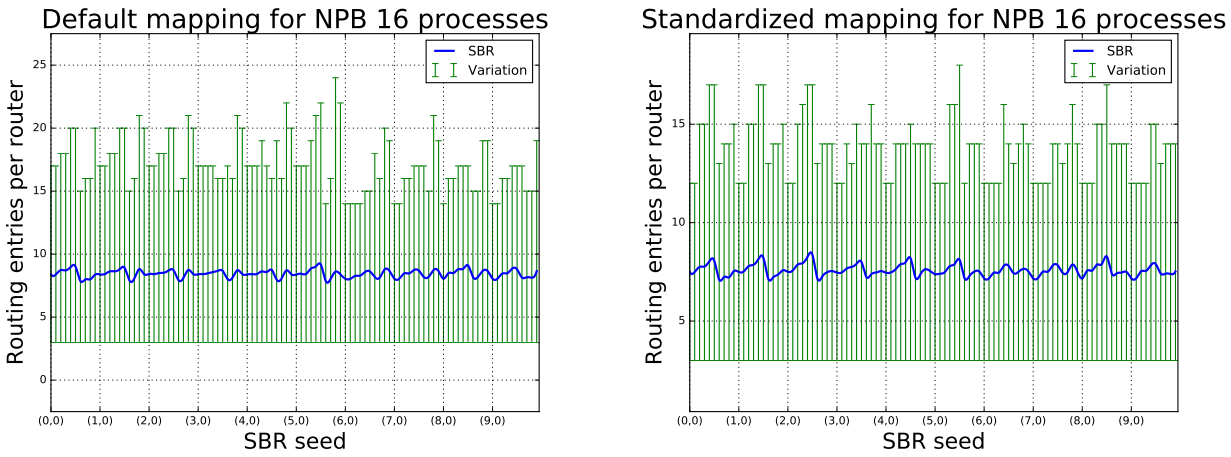


Figure 8.2: Average size of routing entries, in bits, per router per *SBR* Seed using Standardized Mapping. Both segmentation modes are used (*SBR* and *SBR-SZA*) and compared with a Base-NoC without security zones.

The amount of routing entries per router in a NoC configuration can vary greatly. Routers on the edges of the NoC have fewer output ports and, therefore, fewer routing paths to reach destinations. In these cases, *RBR* can pack more routing entries due to the common set of destinations and output ports. Routers located at the center of the NoC, however, have more routing options, which can result in more entries that cannot be grouped by *RBR*. Figure 8.3a shows for the 10x10 NoC scenario the average amount of routing entries when changing the seed used for segmentation. The amount of routing entries can be as few as 3 entries in a router, up to 24 entries, depending on the *SBR* seed.



(a) Default Mapping.

(b) Standardized Mapping.

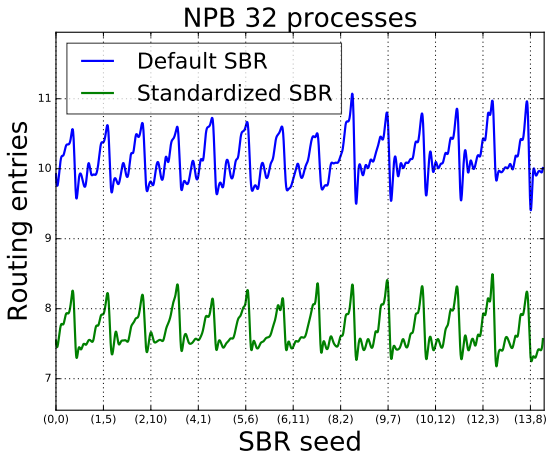
Figure 8.3: Variation of routing entries per router in *NPB* scenario with 16 processes per application in a 10x10 NoC.

Even though the variance in routing entries shown in Figure 8.3 is large, the average amount of entries is lower than the worst case, which shows that most routers do not contain many routing entries. The average distribution of routing entries per router also does not drastically change with the segmentation process. Regarding the 10x10 NoC with Default mapping, the coefficient of variation on the amount of routing entries for different *SBR* seeds is only 3.59%, indicating that the segmentation process has a low impact on the amount of routing entries. Considering a *Base-NoC* with no security zones, there is an average increase of 23.84% and 24.47% of routing entries per router for the 10x10 NoC scenario in Default mapping with *SBR* and *SBR-SZA* segmentation modes, respectively. The results for all NoC configurations is shown in Table 8.1, which illustrates that the segmentation process has little impact on routing table scalability. Meanwhile, the mapping approach is of greater significance, with Standardized mapping offering reduced routing table overhead in all cases, as is also shown in Figure 8.4.

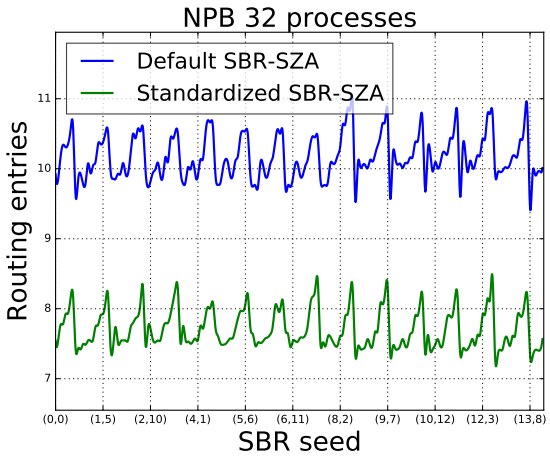
Figures 8.5, 8.6, 8.7 and 8.8 show a heatmap of the amount of routing entries per router in each scenario with Default mapping. Routers near the border of the NoC contain fewer entries than inner routers since these have fewer output ports, resulting in a greater packing of routing entries by *RBR* due to a common set of destinations and output ports.

Table 8.1: Summary of routing table evaluation results.

Scenario	Segmentation	Increase in routing entries per router	Variation in routing entries per SBR seed
5x5 Default	**SBR**	12.12%	3.07%
	SBR-SZA	13.05%	3.02%
7x7 Default	**SBR**	22.71%	4.93%
	SBR-SZA	22.93%	5.12%
10x10 Default	**SBR**	23.84%	3.59%
	SBR-SZA	24.47%	3.99%
14x14 Default	**SBR**	34.22%	3.05%
	SBR-SZA	34.42%	3.00%
5x5 Standardized	**SBR**	10.68%	4.93%
	SBR-SZA	10.91%	4.84%
7x7 Standardized	**SBR**	12.88%	2.36%
	SBR-SZA	13.35%	2.78%
10x10 Standardized	**SBR**	15.13%	3.85%
	SBR-SZA	15.54%	4.21%
14x14 Standardized	**SBR**	13.29%	3.48%
	SBR-SZA	13.51%	3.50%



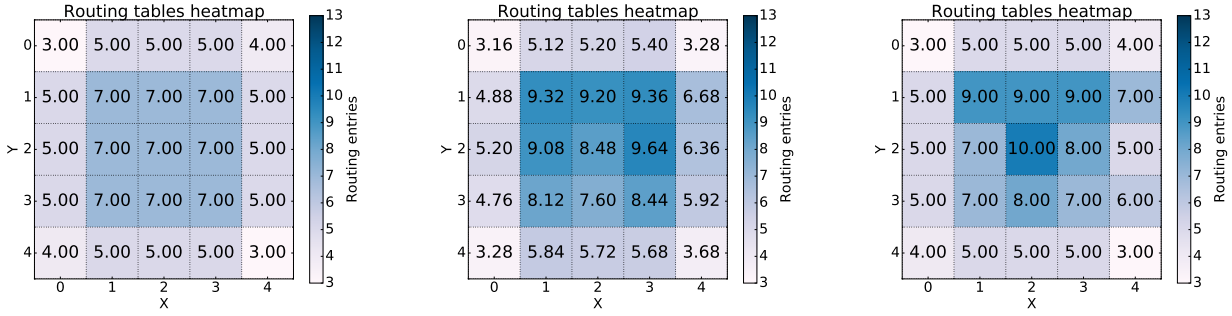
(a) Using *SBR* segmentation mode.



(b) Using *SBR-SZA* segmentation mode.

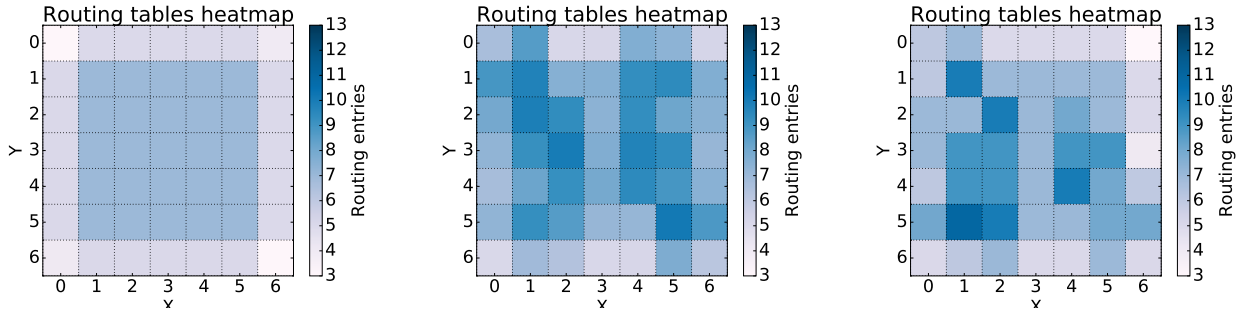
Figure 8.4: Average routing table size comparison for 14x14 NoC scenario. Standardized mapping offers reduced area overhead with both segmentation modes.

Scenarios without security zones contain an homogeneous distribution of routing entries, independent of the NoC size (Figures 8.5a, 8.6a, 8.7a and 8.8a). However, when security zones are present, the routing entries per router increases, and the distribution of these entries depends on the shape of the security zones. The equivalent comparison on the distribution of routing entries is shown in Figures 8.9, 8.10, 8.11 and 8.12 for Standardized mapping. In all cases, the overhead of adding security zones is small, not greatly impacting the scalability of routing tables for the tested scenarios.



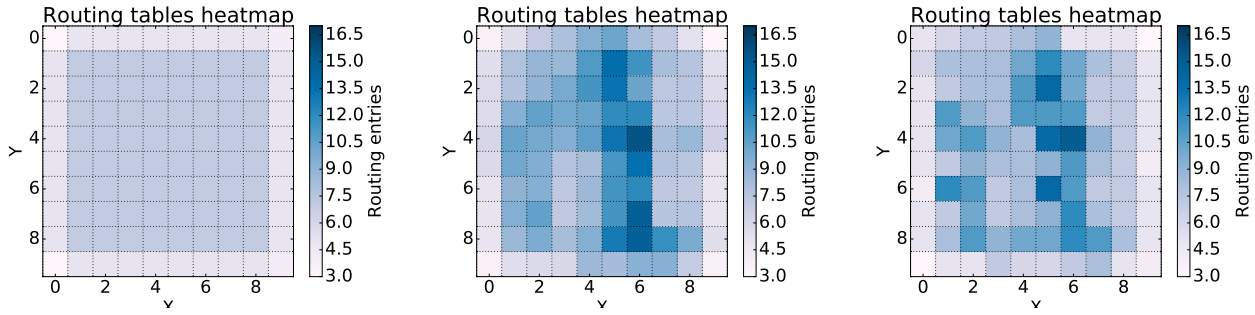
(a) Optimal for 5x5 NoC. (b) Average over all *SBR* seeds. (c) Using optimal *SBR* seed.

Figure 8.5: Routing entries per router heatmap in *NPB* 4 Processes 5x5 NoC scenario with Default mapping.



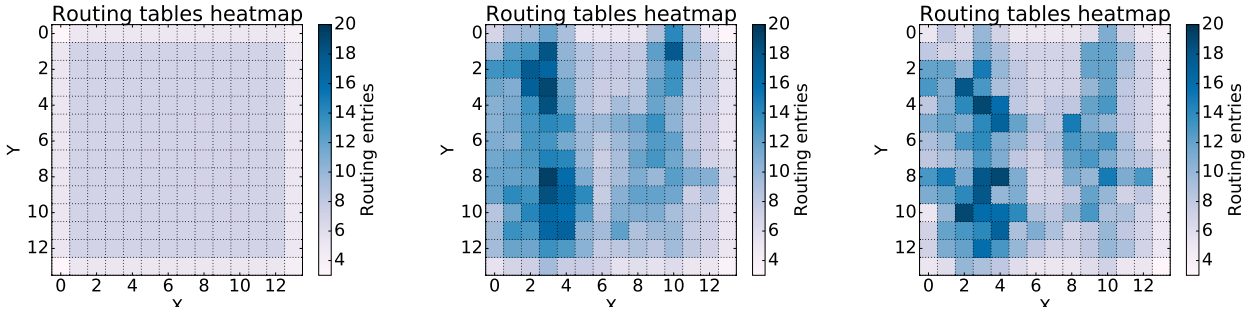
(a) Optimal for 7x7 NoC. (b) Average over all *SBR* seeds. (c) Using optimal *SBR* seed.

Figure 8.6: Routing entries per router heatmap in *NPB* 8 Processes 7x7 NoC scenario with Default mapping.



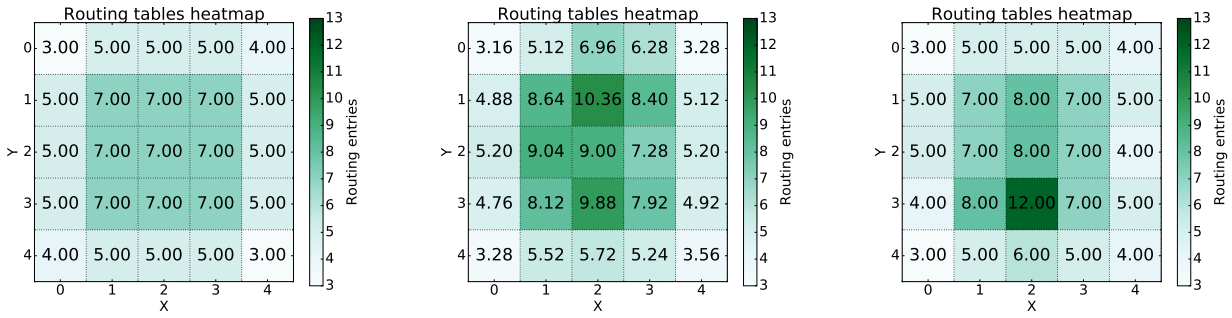
(a) Optimal for 10x10 NoC. (b) Average over all *SBR* seeds. (c) Using optimal *SBR* seed.

Figure 8.7: Routing entries per router heatmap in *NPB* 16 Processes 10x10 NoC scenario with Default mapping.



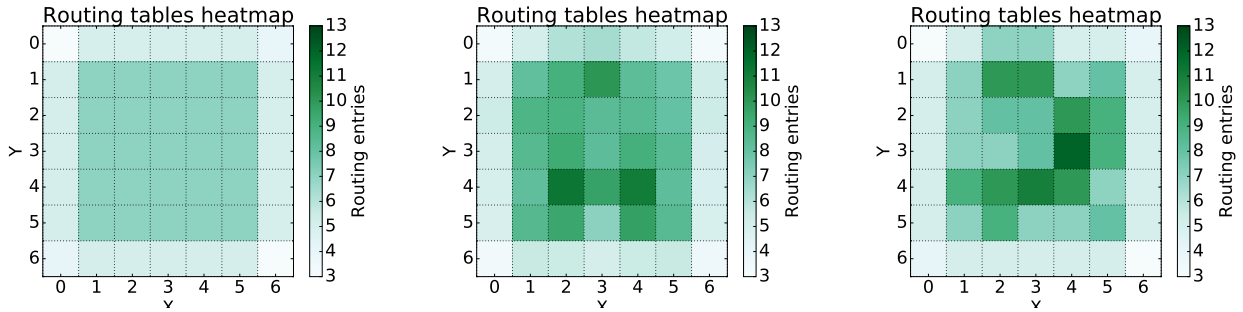
(a) Optimal for 14x14 NoC. (b) Average over all *SBR* seeds. (c) Using optimal *SBR* seed.

Figure 8.8: Routing entries per router heatmap in *NPB* 32 Processes 14x14 NoC scenario with Default mapping.



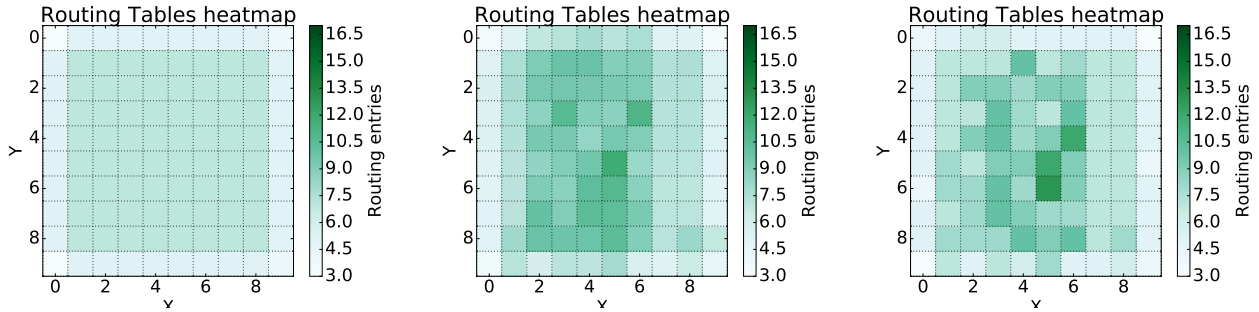
(a) Optimal for 5x5 NoC. (b) Average over all *SBR* seeds. (c) Using optimal *SBR* seed.

Figure 8.9: Routing entries per router heatmap in *NPB* 4 Processes 5x5 NoC scenario with Standardized mapping.



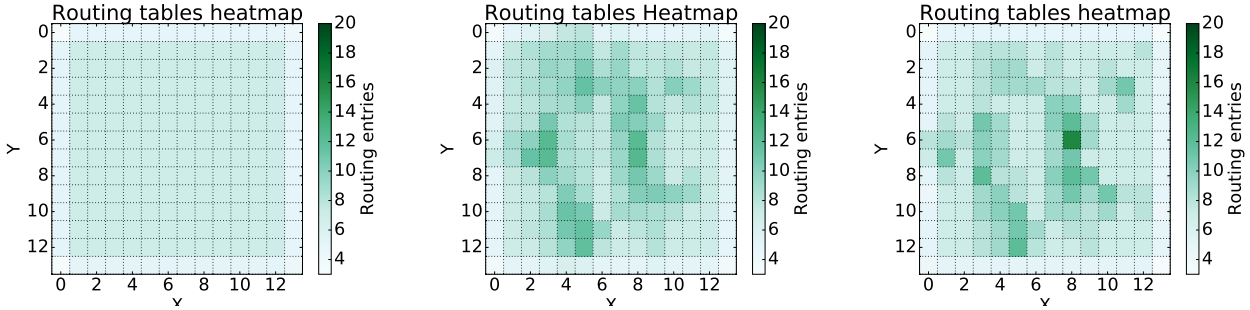
(a) Optimal for 7x7 NoC. (b) Average over all *SBR* seeds. (c) Using optimal *SBR* seed.

Figure 8.10: Routing entries per router heatmap in *NPB* 8 Processes 7x7 NoC scenario with Standardized mapping.



(a) Optimal for 10x10 NoC. (b) Average over all *SBR* seeds. (c) Using optimal *SBR* seed.

Figure 8.11: Routing entries per router heatmap in *NPB* 16 Processes 10x10 NoC scenario with Standardized mapping.



(a) Optimal for 14x14 NoC. (b) Average over all *SBR* seeds. (c) Using optimal *SBR* seed.

Figure 8.12: Routing entries per router heatmap in *NPB* 32 Processes 14x14 NoC scenario with Standardized mapping.

8.2 FIZ Occurrences Evaluation

Evaluation of the occurrences of ***FIZ*** routing paths is performed by checking the routes from all sources to all destination pairs in the NoC. ***FIZ*** and ***PIZ*** scenarios are inversely proportional. A configuration with zero occurrences of ***PIZ*** routes means that all communication occurs as ***FIZ***. Therefore, configurations with fewer occurrences of ***PIZ*** scenarios are preferred.

To determine if a routing path is ***FIZ*** or ***PIZ***, consider Equation 8.3. The routing path from a source router S to a destination D consists of a set of routers (*Path*). If for every router $R \in Path$ the condition $SZ_R = SZ_S$ is satisfied (where SZ corresponds to the security zone of a router) then the routing path is ***FIZ***. Otherwise, if there exists $R \in Path$ such that $SZ_R \neq SZ_S$, then the routing path is ***PIZ***.

$$RoutingPath = \begin{cases} \textbf{\textit{FIZ}} & \iff \forall R \in Path | SZ_R = SZ_S \\ \textbf{\textit{PIZ}} & \iff \exists R \in Path | SZ_R \neq SZ_S \end{cases} \tag{8.3}$$

Figures 8.13 and 8.14 illustrate the distribution of ***PIZ*** communications in each scenario for different seeds used by the segmentation algorithms. The 5x5 NoC is the only scenario that obtains ***FIZ*** routing with either mapping methods. In Default mapping, any segmentation approach results in only ***FIZ***, which is a characteristic specific to that mapping layout. Standardized mapping, on the other hand, contains ***PIZ*** routes depending on the seed used in *SBR*.

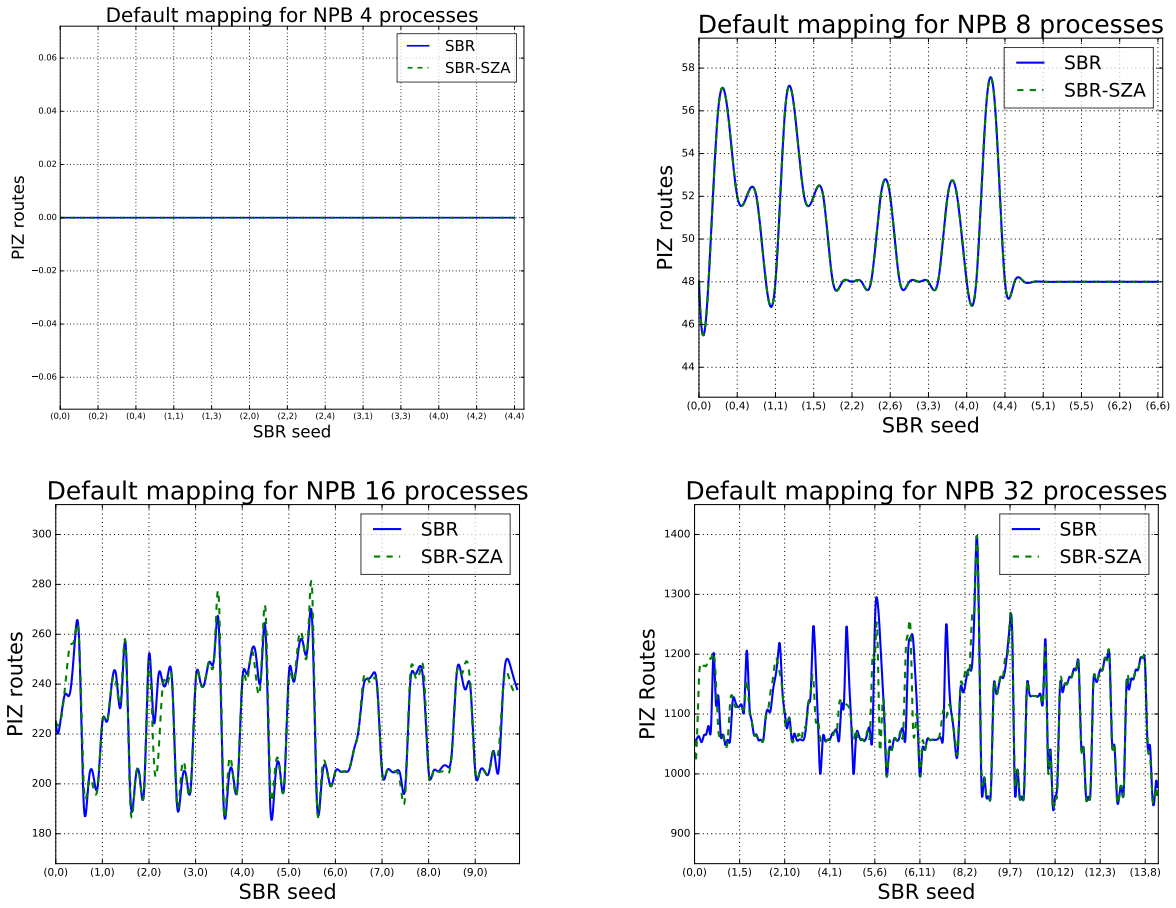


Figure 8.13: Distribution of ***PIZ*** routers per *SBR* seed using Default mapping. Both segmentation modes are used (*SBR* and *SBR-SZA*). When zero ***PIZ*** routers occur, it characterizes a configuration with only ***FIZ*** communication.

A summary on ***PIZ*** routes occurrences for all evaluated configurations is shown in Table 8.2. Employing *SBR-SZA* does not yield more secure paths than normal *SBR*, indicating that this technique did not improve the amount of secure routes for the tested scenarios. However, the choice of the segmentation seed plays am important role in defining secure routes; for instance, the 14x14 NoC scenario possesses a 7.12% and 21.62% variation in the occurrence of ***PIZ*** routes per *SBR* seed with Default and Standardized mapping, respectively. It is important to note that the 5x5 NoC with Standardized mapping has a very high coefficient of variation due to the existence of many segmentation modes with no ***PIZ*** routes, as the chart in Figure 8.14 shows.

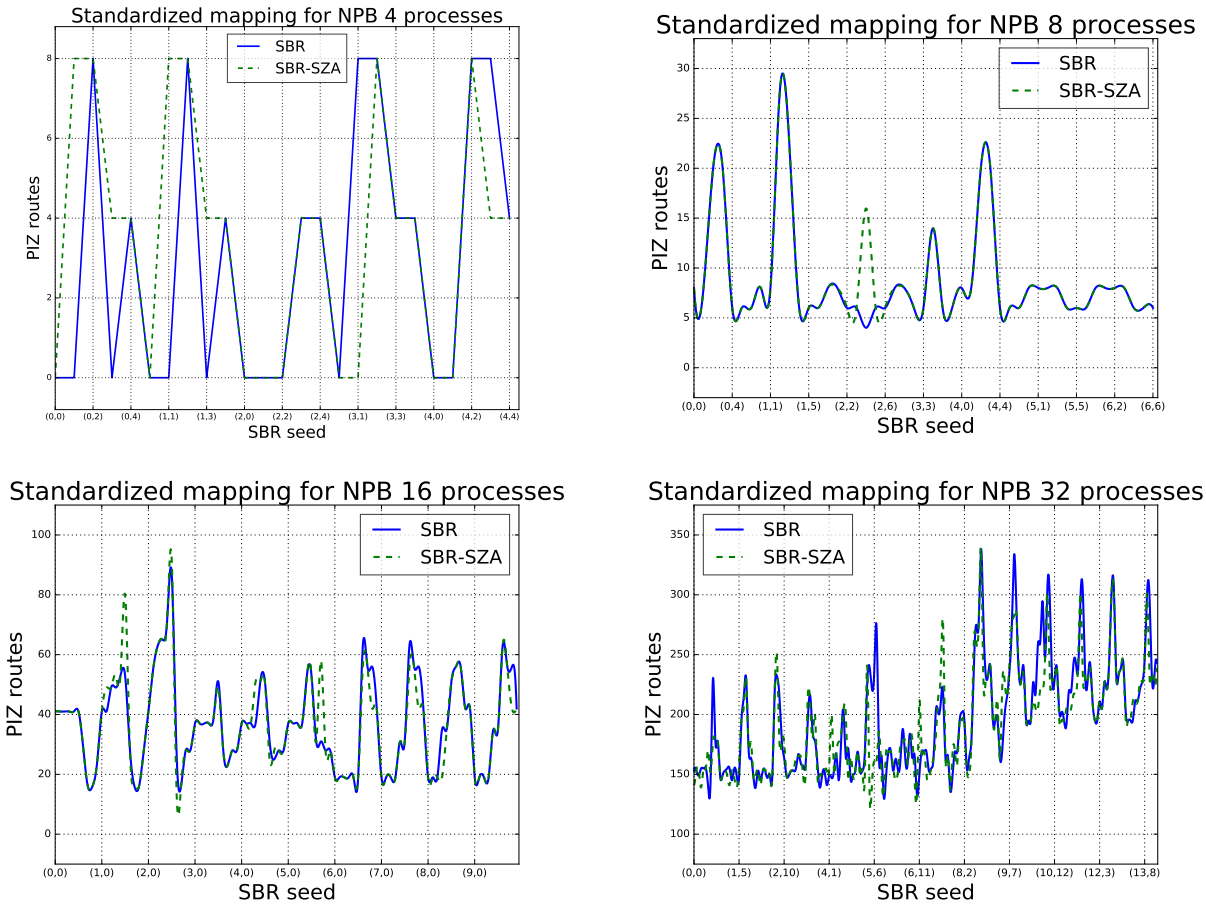


Figure 8.14: Distribution of ***PIZ*** routers per *SBR* seed using Standardized mapping. Both segmentation modes are used (*SBR* and *SBR-SZA*). When zero ***PIZ*** routers occur, it characterizes a configuration with only ***FIZ*** communication.

Table 8.2: Summary of ***FIZ*** Evaluation Results.

Scenario	Segmentation	Minimum PIZ Routes	Variation in PIZ Scenarios per SBR Seed
5x5 Default	**SBR**	0	0%
	SBR-SZA	0	0%
7x7 Default	**SBR**	48	5.64%
	SBR-SZA	48	5.64%
10x10 Default	**SBR**	182	9.72%
	SBR-SZA	182	9.94%
14x14 Default	**SBR**	946	7.12%
	SBR-SZA	946	6.85%
5x5 Standardized	**SBR**	0	107.09%
	SBR-SZA	0	86.96%
7x7 Standardized	**SBR**	4	58.39%
	SBR-SZA	6	57.39%
10x10 Standardized	**SBR**	6	40.29%
	SBR-SZA	6	42.14%
14x14 Standardized	**SBR**	121	23.33%
	SBR-SZA	121	21.62%

Regarding the mapping techniques, Standardized mapping provides the lowest occurrence of ***PIZ*** routing scenarios. In the 10x10 NoC there is a reduction of 96.71% of ***PIZ*** routes. Significant reductions also occur for the 7x7 and 14x14 NoC configurations, with the 5x5 NoC being the only exception due to the existence of only ***FIZ*** routes in both mappings. This significant reduction demonstrates that the mapping technique can improve system security by lowering the amount of insecure communication paths. Figure 8.15 illustrates the occurrence of ***PIZ*** routes on the tested NoC configurations with both mapping modes when varying the *SBR* seed, demonstrating the lower occurrence of ***PIZ*** routes in Standardized mapping. With the exception of the 5x5 NoC, all other scenarios benefit from the Standardized Mapping technique.

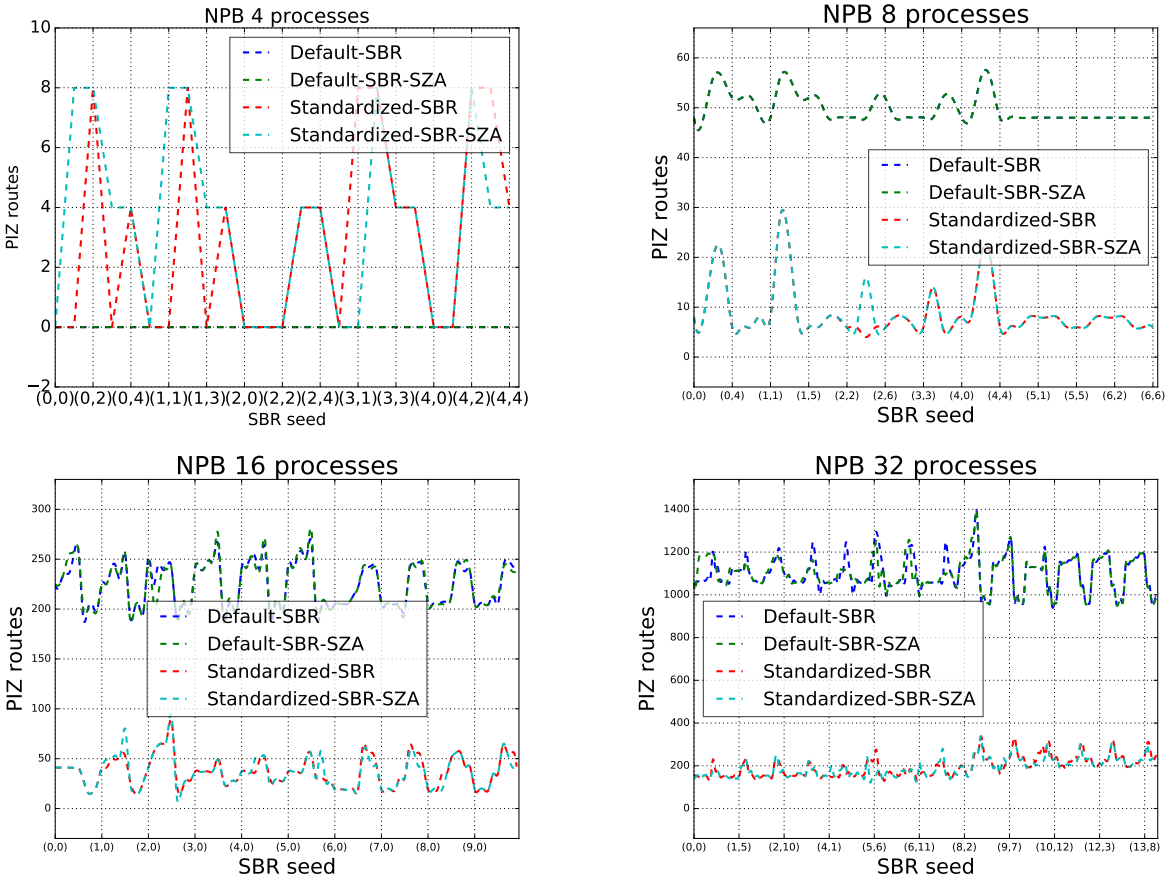


Figure 8.15: Comparison of Default and Standardized Mappings on the occurrence of ***PIZ*** routes.

8.3 NPB Applications Evaluation

It is possible to evaluate the impact of the secure routing technique on system workloads by measuring application execution times. All of the *NPB* applications, described in Section 7.3, generate a simulation report that measures their final execution time. An

evaluation of their performance, therefore, consists of executing the benchmark applications while using different network latency configurations on the NoC employed by Gem5.

There are two main parameters that define the network latency: the *Routing Path* and the *Communication Delay*. The *Routing Path* depends on two variables, namely: (i) the NoC Size, which corresponds to the four configurations shown in Table 7.1; and (ii) Task Mapping, corresponding to the eight mappings illustrated in Figure 7.6. Using these parameters it is possible to compute the routing paths among tasks distributed in the NoC with *SBR* and *RBR*. Since the occurrences of ***PIZ*** scenarios did not change with *SBR-SZA* compared to *SBR*, as it was shown in Section 8.2, the evaluations consider only *SBR* segmentation. Additionally, in all configurations the NoC was segmented using the optimal *SBR* seed for minimal ***PIZ*** routes.

The *Communication Delay* depends on the NoC model. This work assumes a Hermes NoC model, described in Section 7.1, where routing packets from one router to another occurs in five stages, encompassing: (i) input request; (ii) buffering; (iii) route computation; (iv) arbitration; and (v) packet switching to the next router. Each stage requires a clock cycle to perform, therefore each hop of a packet takes five clock cycles to complete, disregarding contention.

Assuming that packets which are traversing an insecure zone employ encryption as the protection mechanism, the *Communication Delay* also takes into consideration symmetric block-cipher techniques that simulate the protection of NoC packets. Table 8.3 shows the encryption cost of seven block cipher implementations. Out of the seven techniques, three were chosen, namely *AES-128*, *DES*, and *mCrypton*, characterizing high, medium, and low cost block cyphers, respectively.

We also consider a *Base* scenario that disregards security. The *Base* configuration employs *SBR* and *RBR* routing with no security zones and no encryption in any cases, generating minimal routing paths for all communication pairs. This scenario enables the evaluation of the overhead introduced by the secure routing implementation in applications execution time.

Table 8.3: Summary of block-cipher modules (based on [11]).

	Key size	Block size	Cycles per block	Throughput at 100KHz (Kbps)
Present-80 [11]	80	64	32	200
AES-128 [19]	128	128	1032	12.4
HIGHT [30]	128	64	34	188.2
mCrypton [35]	96	64	13	492.3
Camellia [4]	128	128	20	640
DES [46]	56	64	144	44.4
DESXL [46]	184	64	144	44.4

The network latency is obtained by considering the communication path length $|Path(S,D)|$ between each source S and destination D. The hop cost H is determined according to how packets are forwarded through routers, which is presumed to take four clock cycles for switching, and another clock cycle to forward, which we call F. If a packet requires protection, then the cost of the encryption process (in clock cycles) $E_{cost}(cipher)$ is considered, depending on the chosen *cipher* mode. Equations 8.4 and 8.5 demonstrate how to estimate the network delay between a source and a destination pair.

$$Delay(S,D) = |Path(S,D)| \times H + (|Path(S,D) + 1) \times F + E_{cost}(cipher) \tag{8.4}$$

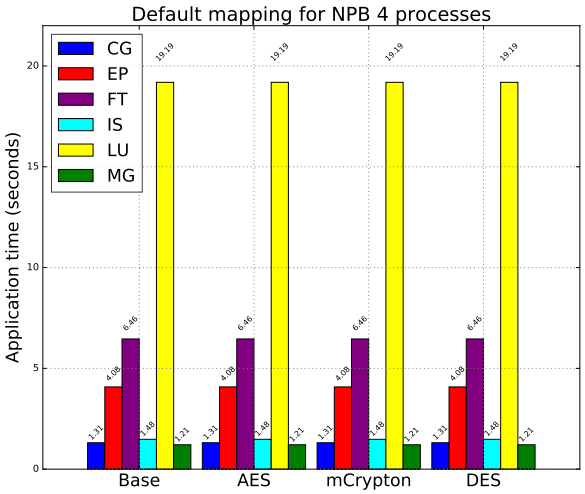
$$E_{cost}(cipher) = \begin{cases} \textbf{\textit{FIZ route}}, & 0 \\ \textbf{\textit{PIZ route}}, & \text{Block cost of Table 8.3} \end{cases} \tag{8.5}$$

Figures 8.16 and 8.17 illustrate the benchmarks execution time with Default and Standardized mapping, respectively. In most cases application performance remains the same with either mapping modes, and the secure routing implementation does not incur an overhead when compared to the *Base* scenario without security zones. Variations in execution time when employing an encryption technique are due to the occurrence of ***PIZ*** routes for communication flows of an application. When ***PIZ*** routes are present, any communication incurs a latency overhead due to the cost of the block cypher employed to protect NoC packets. Still, some applications with low amounts of communication might not be affected by encryption, as most of the application time is used for computation instead of communication.

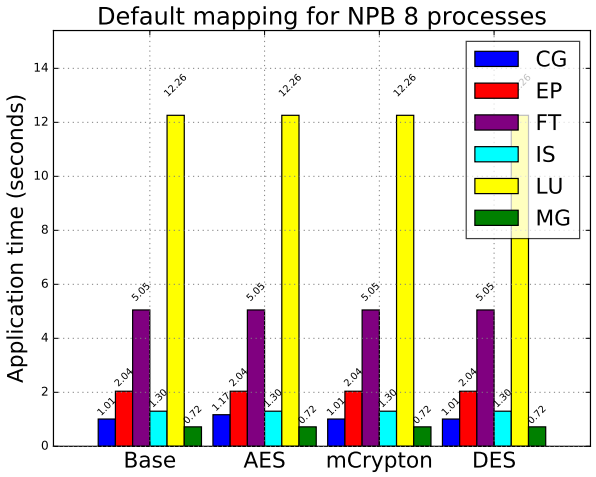
The 5x5 NoC configurations with Default and Standardized mapping do not posses ***PIZ*** routes for any one of the *NPB* applications and, as a result, execution times do not vary when employing packet protection. Applications with low communication requirements, such as **EP**, are also not affected by the overhead of encrypting NoC packets, even thought this application concentrates the majority of ***PIZ*** routes in all configurations.

However, **CG**, **FT**, **IS** and **MG** can suffer significant variations in execution time. Using the Default mapping, **CG** with 32 processes and *AES-128* encryption has an increase of over 400% in execution time. For the same application, *mCrypton* and *DES* incur a much lower overhead, at 1.3% and 31.29%, respectively.

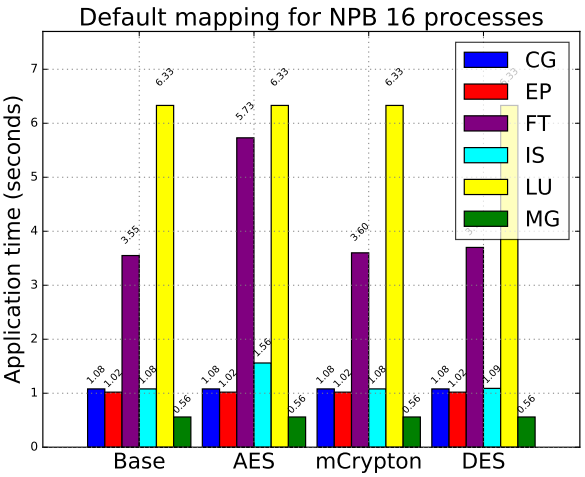
An interesting situation is shown in **LU** with 32 processes. Out of the six *NPB* applications used in this work, **LU** concentrates the highest communication amount, exchanging several small packets among its tasks. As such, task mapping plays a crucial role on this application's performance. The mapping criteria employed by CAFES consists into distributing applications on a NoC topology such that a pair of tasks with high communication amounts are as close as possible. In Default mapping, this property is properly accomplished by



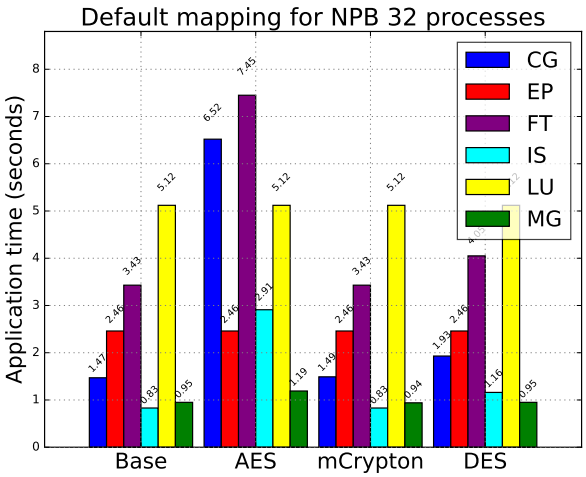
(a) 5x5 NoC with applications split into 4 processes.



(b) 7x7 NoC with applications split into 8 processes.



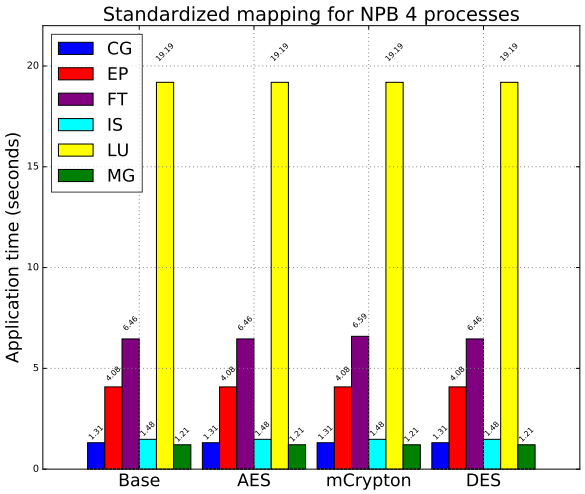
(c) 10x10 NoC with applications split into 16 processes.



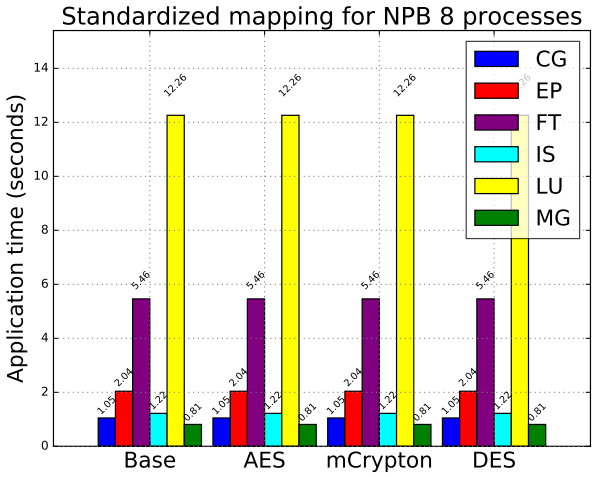
(d) 14x14 NoC with applications split into 32 processes.

Figure 8.16: *NPB* application time using *AES-128*, *mCrypton*, and *DES* encryption modes in Default mapping.

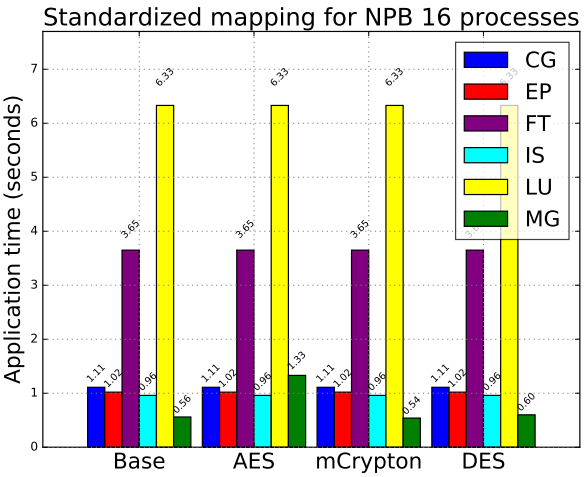
CAFES since the absolute communication volume of tasks is the criteria used for application mapping. Therefore, **LU** with 32 processes improves 20% its execution time versus **LU** with 16 processes due to higher parallelization with Default mapping. However, the Standardized technique can lead to situations that mask the difference in communication amounts among pairs of tasks, since the absolute values are leveled off. As a result, the distribution of tasks inside a security zone can be such that two highly communicating tasks are mapped far apart, degrading communication performance. A non-optimal mapping strategy can greatly impact applications, which is the case in **LU** with 32 processes using Standardized mapping, where its execution time is over three times longer than in Default mapping. Also, due to the high communication volume of **LU**, the associated cost of an encryption technique is negli-



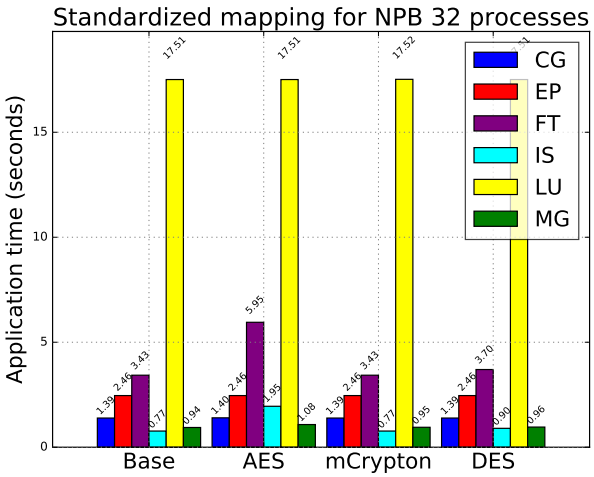
(a) 5x5 NoC with applications split into 4 processes.



(b) 7x7 NoC with applications split into 8 processes.



(c) 10x10 NoC with applications split into 16 processes.



(d) 14x14 NoC with applications split into 32 processes.

Figure 8.17: *NPB* application time using *AES-128*, *mCrypton*, and *DES* encryption modes in Standardized mapping.

gible, not affecting its execution time even though ***PIZ*** routes are present in the 14x14 NoC scenarios.

The results shown in Table 8.4 demonstrate that the mapping technique has greater impact in application performance than the secure routing implementation. In all cases that do not contain ***PIZ*** routes, a scenario that employs the secure routing algorithm does not increase application execution time when compared to the *Base* scenario. However, differences are present for the same application among different mappings. **CG** with 8 processes is 3.69% slower in Standardized mapping versus Default. Meanwhile, **FT** with 8 processes is 7.51% faster in Standardized mapping. However, for other applications, like **LU** with 4, 8, and 16 processes, there is not difference in execution time among either mapping technique.

Table 8.4: *NPB* Simulation Results.

Benchmark	Default Mapping					Standardized Mapping				
	PIZ Routes	Execution Time (seconds)				PIZ Routes	Execution Time (seconds)			
		Base	AES	mCrypton	DES		Base	AES	mCrypton	DES
CG 4	0	1.31	1.31	1.31	1.31	0	1.31	1.31	1.31	1.31
CG 8	0	1.01	1.01	1.01	1.01	4	1.05	1.05	1.05	1.05
CG 16	0	1.08	1.08	1.08	1.08	0	1.11	1.11	1.11	1.11
CG 32	43	1.47	6.52	1.49	1.93	30	1.39	1.4	1.39	1.39
EP 4	0	4.08	4.08	4.08	4.08	0	4.08	4.08	4.08	4.08
EP 8	48	2.04	2.04	2.04	2.04	0	2.04	2.04	2.04	2.04
EP 16	169	1.02	1.02	1.02	1.02	0	1.02	1.02	1.02	1.02
EP 32	780	2.46	2.46	2.46	2.46	36	2.46	2.46	2.46	2.46
FT 4	0	6.46	6.46	6.59	6.46	0	6.46	6.46	6.46	6.46
FT 8	0	5.46	5.46	5.46	5.46	0	5.05	5.05	5.05	5.05
FT 16	3	3.55	5.73	3.6	3.7	0	3.65	3.65	3.65	3.65
FT 32	4	3.43	5.95	3.43	3.7	14	3.43	7.45	3.43	4.05
IS 4	0	1.48	1.48	1.48	1.48	0	1.48	1.48	1.48	1.48
IS 8	0	1.3	1.3	1.3	1.3	0	1.22	1.22	1.22	1.22
IS 16	10	1.08	1.56	1.08	1.09	0	0.96	0.96	0.96	0.96
IS 32	12	0.83	2.91	0.83	1.16	10	0.77	1.95	0.77	0.9
LU 4	0	19.19	19.19	19.19	19.19	0	19.19	19.19	19.19	19.19
LU 8	0	12.26	12.26	12.26	12.26	0	12.26	12.26	12.26	12.26
LU 16	0	6.33	6.33	6.33	6.33	0	6.33	6.33	6.33	6.33
LU 32	91	5.12	5.12	5.12	5.12	19	17.51	17.51	17.52	17.51
MG 4	0	1.21	1.21	1.21	1.21	0	1.21	1.21	1.21	1.21
MG 8	0	0.72	0.72	0.72	0.72	0	0.81	0.81	0.81	0.81
MG 16	0	0.56	0.56	0.56	0.56	6	0.54	1.33	0.54	0.6
MG 32	20	0.95	1.19	0.94	0.95	12	0.94	1.08	0.95	0.96

9. CONCLUSIONS AND FUTURE WORK

Due to the growing adoption of complex SoCs in embedded systems, the exploration of security features at the design phase for such systems can offer increased protection from vulnerabilities. Embedding security at the interconnect fabric of SoCs has been shown to enhance system security with low performance impact [17], [54], [28]. As such, this work's main contribution is the utilization of the routing algorithm used in NoCs to define secure routing paths for applications targeting MPSoCs. To achieve this purpose, this work presented a routing technique that is capable of considering security zones defined on the NoC, evaluating aspects such as the overhead in the routing tables of the NoC routers, the amount of secure routing paths achieved by the routing technique, and the impact of this method in real application workloads. The proposal of this work has already been published and accepted at the *Symposium on Integrated Circuits and System Design* (SBCCI) in 2016 [21].

Results demonstrate that the adopted routing strategy is capable of considering system security requirements for defining the communication paths among elements in a NoC. The proposed model has been validated using the abstract NoC configuration tool presented in Section 7; *SBR* is correctly inserting bidirectional turn restrictions in order to obtain *deadlock-free* routing, while *RBR* is correctly defining regions in the NoC and populating the routing tables of the routers.

Evaluation has shown that an implementation of *RBR* with security zones brings an overhead in the size of routing tables, depending on the shape and the number of security zones. Nevertheless, this impact is small, corresponding to, at most, a 34% increase in the size of routing tables when compared to a NoC without security zones. Even so, this overhead has the benefit of reducing ***PIZ*** communication scenarios, contributing to system security. The proposed segmentation process using *SBR-SZA* has not improved the amount of secure routing paths for the tested scenarios, demonstrating that traditional *SBR* is sufficient. Still, the segmentation process has a significant impact for reducing ***PIZ*** routes in the NoC, as it was shown in Table 8.2.

As it was previously discussed in Section 6.2, the shape of the security zone plays a crucial role for defining secure routing paths. Therefore, the mapping technique can greatly impact the evaluated results. While there is a 34% increase in the size of routing tables for the 14x14 NoC with Default mapping, the Standardized technique was able to reduce this overhead to 13%; a significant decrease due to the greater packing of application tasks, reducing the necessary amount of secure routes. The same benefits of Standardized mapping applies to the amount of ***PIZ*** routes, previously at 946 for the 14x14 NoC with Default mapping, and now at 121 routes.

The *NPB* results have demonstrated that the utilization of non-minimal routing paths has a very low performance impact on that set of applications. Most of the overhead comes from applications that contain ***PIZ*** routes while employing a costly data protection mechanism, such as *AES-128*. For low cost encryption techniques, like *mCrypton*, the overhead becomes negligible, not presenting any noticeable changes in application performance. Still, applications will behave differently based on their communication requirements. An application like **EP** with several ***PIZ*** routes, for example, suffers no performance degradation even with *AES-128* due to its low communication requirements. However, for applications with high communication demands like **FT**, a small amount of ***PIZ*** routes is enough to incur a 74% overhead in its execution time in a 14x14 NoC with *AES-128* in Default mapping. While the *NPB* applications do not represent an exhaustive evaluation on the impact of the secure routing technique shown in this work, the varied communication demands of its applications should provide a reasonable understanding regarding the behavior of this routing technique.

We would also like to extend the exploration of this work to consider runtime reconfiguration. During normal operation of a MPSoC, tasks can be loaded and unloaded from IPs, generating different mappings and, therefore, other security zones configurations. The computational complexity of *RBR* can become a determinant factor when a security zone changes, as it is necessary to recompute all secure routes for all routers in the NoC. Depending on the NoC size, this process could negatively impact performance, specially if it frequently occurs. As such, it is important to use low overhead path-finding algorithms and to exploit the inherent parallelism of MPSoCs to minimize *RBR* recomputation.

Another approach to generate secure routes could be *Circuit Switching*, where the routing path among a source and a destination is reserved during data transmission. By locking the communication resources during the transmission of sensitive content among IPs, it would be possible to mitigate timing or *DoS* attacks from interfering with the communication; still, malicious applications could drain the communication resources by constantly requesting NoC routes, degrading system performance. Much like in this work, unless it is possible to isolate regions in the MPSoC in order to lock malicious applications out of sensitive zones, *DoS* attacks are still a vector for disrupting normal system operation.

When we consider the possibility of traffic isolation and reducing the overhead of runtime reconfiguration, mapping applications into security zones that do not generate ***PIZ*** routes presents interesting possibilities. As the results demonstrate, the Standardized mapping technique is able to greatly reduce the occurrence of ***PIZ*** routes due to the packing of application tasks into non-fragmented security zones. A similar approach could use predefined security clusters for mapping applications. Figure 9.1 illustrates a NoC with nine clusters, each with eight IPs. The mapping algorithm then distributes tasks into these clusters, creating the security zones. A cluster can only contain a single zone, but a zone can spam multiple clusters. As Figure 9.1 shows, *Zone 3* spams two clusters. Specific links connect each cluster and are enabled or disabled according to system security requirements. A

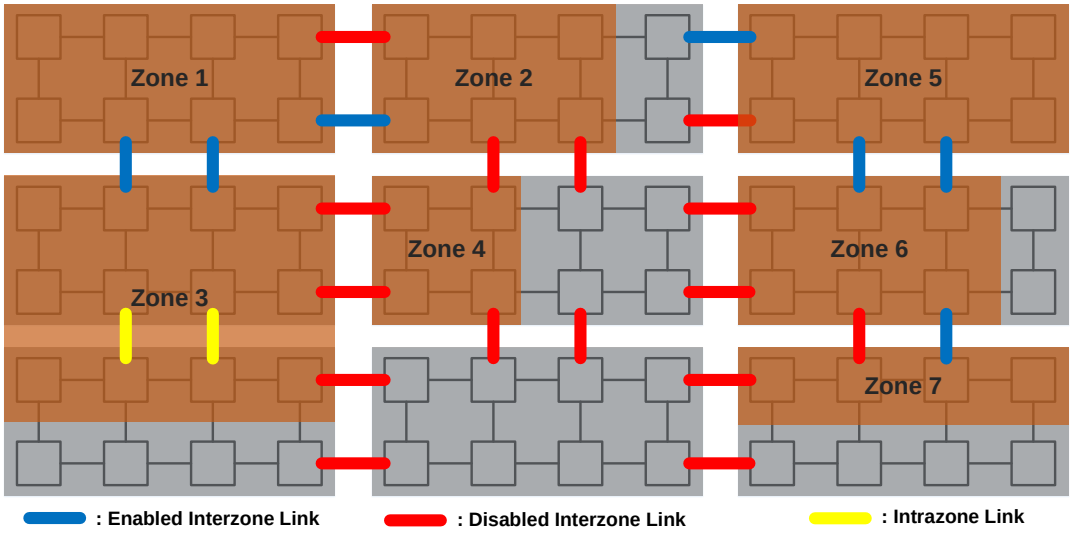


Figure 9.1: Clustered security zones. A cluster might contain a single security zone, and each security zone might span to more than one cluster.

zone can also communicate with another zone if needed, e.g., Zones 1, 2 and 5, or execute completely isolated, e.g., Zone 4. Clusters contain routing paths defined at design time that never change during runtime, therefore never requiring route recomputation. The obvious downside of this approach is intra-cluster fragmentation since not all IPs from a cluster might be used.

References

[1] Agarwal, A.; Shankar, R. “Survey of network on chip (NoC) architectures & contributions”, *Journal of Engineering, Computing and Architecture*, vol. 3–1, Jan 2009, pp. 15.

[2] Ahmed, A. B.; Abdallah, A. B.; Kuroda, K. “Architecture and design of efficient 3D network-on-chip (3d NoC) for custom multicore SoC”. In: International Conference on Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010, pp. 67–73.

[3] Ancajas, D. M.; Chakraborty, K.; Roy, S. “Fort-NoCs: Mitigating the threat of a compromised NoC”. In: Design Automation Conference (DAC), 2014, pp. 1–6.

[4] Aoki, K.; Ichikawa, T.; et al.. “Camellia: A 128-bit block cipher suitable for multiple platforms - design and analysis”. In: Workshop on Selected Areas in Cryptography (SAC), 2001, pp. 39–56.

[5] Avizienis, A.; Laprie, J. C.; et al.. “Basic concepts and taxonomy of dependable and secure computing”, *IEEE Transactions on Dependable and Secure Computing*, vol. 1–1, Mar 2004, pp. 11–33.

[6] Bahrebar, P.; Stroobandt, D. “The hamiltonian-based odd-even turn model for maximally adaptive routing in 2d mesh networks-on-chip”, *Computers and Electrical Engineering*, vol. 45–C, Jul 2015, pp. 386–401.

[7] Baron, S.; Wangham, M. S.; Zeferino, C. A. “Security mechanisms to improve the availability of a network-on-chip”. In: International Conference on Electronics, Circuits, and Systems (ICECS), 2013, pp. 609–612.

[8] Benini, L.; Micheli, G. D. “Networks on chips: a new SoC paradigm”, *Computer*, vol. 35–1, Aug 2002, pp. 70–78.

[9] Binkert, N.; Beckmann, B.; et al.. “The gem5 simulator”, *ACM SIGARCH Computer Architecture News*, vol. 39–2, Aug 2011, pp. 1–7.

[10] Bjerregaard, T.; Mahadevan, S. “A survey of research and practices of network-on-chip”, *ACM Computing Surveys (CSUR)*, vol. 38–1, Jun 2006, pp. 51.

[11] Bogdanov, A.; Knudsen, L.; et al.. “PRESENT: An ultra-lightweight block cipher”. In: International Workshop on Cryptographic Hardware and Embedded Systems (CHES), 2007, pp. 450–466.

[12] Boraten, T.; Kodi, A. K. “Packet security with path sensitization for NoCs”. In: Design, Automation and Test in Europe (DATE), 2016, pp. 1136–1139.

[13] Cataldo, R. “Design and exploration of 3d MPSoCs with on-chip cache support”, Master’s Thesis, Pontifícia Universidade Católica do Rio Grande do Sul, 2015, 129p.

[14] Cataldo, R.; Fernandes, R.; et al.. “The impact of routing arbitration mechanisms on 3d NoC latency”. In: International Conference on Electronics, Circuits and Systems (ICECS), 2014, pp. 890–894.

[15] Coburn, J.; Ravi, S.; et al.. “SECA: Security-enhanced communication architecture”. In: International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES), 2005, pp. 78–89.

[16] Cui, A.; Chang, C. H.; et al.. “A robust FSM watermarking scheme for IP protection of sequential circuit design”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30–5, May 2011, pp. 678–690.

[17] Diguet, J. P.; Evain, S.; et al.. “NOC-centric security of reconfigurable SoC”. In: International Symposium on Networks-on-Chip (NOCS), 2007, pp. 223–232.

[18] Duato, J.; Yalamanchili, S.; Lionel, N. “Interconnection Networks: An Engineering Approach”. Morgan Kaufmann Publishers Inc., 2002, 600p.

[19] Feldhofer, M.; Dominikus, S.; Wolkerstorfer, J. “Strong authentication for RFID systems using the AES algorithm”. In: Cryptographic Hardware and Embedded Systems (CHES), 2014, pp. 357–370.

[20] Fernandes, R.; Brahm, L.; et al.. “OcNoC: Efficient one-cycle router implementation for 3d mesh network-on-chip”. In: International Conference on VLSI Design (VLSID), 2015, pp. 105–110.

[21] Fernandes, R.; Marcon, C.; et al.. “A security aware routing approach for NoC-based MPSoCs”. In: Symposium on Integrated Circuits and Systems Design (SBCCI), 2016, pp. 6.

[22] Fernandes, R.; Oliveira, B.; et al.. “A non-intrusive and reconfigurable access control to secure NoCs”. In: International Conference on Electronics, Circuits, and Systems (ICECS), 2015, pp. 316–319.

[23] Fernandez-Alonso, E.; Castells-Rufas, D.; et al.. “Survey of NoC and programming models proposals for MPSoC”, *International Journal of Computer Science Issues (IJCSI)*, vol. 9–2, Mar 2012, pp. 22–32.

[24] Fiorin, L.; Palermo, G.; Silvano, C. “A security monitoring service for NoCs”. In: IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2008, pp. 197–202.

[25] Fiorin, L.; Palermo, G.; et al.. “Secure memory accesses on networks-on-chip”, *IEEE Transactions on Computers*, vol. 57–9, Sep 2008, pp. 1216–1229.

[26] Flich, J.; Mejia, A.; et al.. “Region-based routing: An efficient routing mechanism to tackle unreliable hardware in network on chips”. In: International Symposium on Networks-on-Chip (NOCS), 2007, pp. 183–194.

[27] Gem5. “Architecture support - gem5”. Source: http://www.gem5.org/Architecture_Support, Jun 2016.

[28] Grammatikakis, M. D.; Papadimitriou, K.; et al.. “Security effectiveness and a hardware firewall for mpsocs”. In: International Conference on High Performance Computing and Communications, International Symposium on Cyberspace Safety and Security, International Conference on Embedded Software and Syst (HPCC,CSS,ICESS), 2014, pp. 1032–1039.

[29] Grammatikakis, M. D.; Papadimitriou, K.; et al.. “Security in MPSoCs: A NoC firewall and an evaluation framework”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34–8, Jun 2015, pp. 1344–1357.

[30] Hong, D.; Sung, J.; et al.. “HIGHT: A new block cipher suitable for low-resource device”. In: International Conference on Cryptographic Hardware and Embedded Systems (CHES), 2006, pp. 46–59.

[31] Hu, Y.; Müller-Gritschneder, D.; et al.. “Automatic ILP-based firewall insertion for secure application-specific networks-on-chip”. In: International Workshop on Interconnection Network Architectures: On-Chip, Multi-Chip (INA-OCMC), 2015, pp. 9–12.

[32] Ilgun, K.; Kemmerer, R. A.; Porras, P. A. “State transition analysis: a rule-based intrusion detection approach”, *IEEE Transactions on Software Engineering*, vol. 21–3, Mar 1995, pp. 181–199.

[33] ITRS. “International technology roadmap for semiconductors”. Source: http://www.itrs.net/reports.html, Nov 2015.

[34] Kocher, P.; Lee, R.; et al.. “Security as a new dimension in embedded system design”. In: Design Automation Conference (DAC), 2004, pp. 753–760.

[35] Lim, C. H.; Korkishko, T. “mCrypton – a lightweight block cipher for security of low-cost RFID tags and sensors”. In: International Conference on Information Security Applications (WISA), 2006, pp. 243–258.

[36] Liu, Q.; Ji, W.; et al.. “IP protection of mesh NoCs using square spiral routing”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24–4, Apr 2016, pp. 1560–1573.

[37] Luo-Feng, G.; Gao-ming, D.; et al.. “Design and performance evaluation of a 2D-mesh network on chip prototype using FPGA”. In: IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), 2008, pp. 1264–1267.

[38] Marcon, C.; Palma, J.; et al.. “Modeling the traffic effect for the application cores mapping problem onto NoCs”, *Vlsi-Soc: From Systems To Silicon*, vol. 240–1, Oct 2007, pp. 179–194.

[39] Md.TariqueAhmad, S.; Haque, S.; Shoeb Khan, P. “Privacy preserving in data mining by normalization”, *International Journal of Computer Applications*, vol. 96–6, Jun 2014, pp. 14–18.

[40] Mejia, A.; Flich, J.; Duato, J. “On the potentials of segment-based routing for NoCs”. In: International Conference on Parallel Processing (ICPP), 2008, pp. 594–603.

[41] Mejia, A.; Flich, J.; et al.. “Segment-based routing: an efficient fault-tolerant routing algorithm for meshes and tori”. In: International Parallel Distributed Processing Symposium (IPDPS), 2006, pp. 10.

[42] Moraes, F.; Calazans, N.; et al.. “HERMES: An infrastructure for low area overhead packet-switching networks on chip”, *Integration, the VLSI Journal*, vol. 38–1, Oct 2004, pp. 69–93.

[43] NASA. “NAS parallel benchmarks”. Source: http://www.nas.nasa.gov/publications/npb.html, Jun 2016.

[44] Page, D. “Defending against cache-based side-channel attacks”, *Information Security Technical Report*, vol. 8–1, Mar 2003, pp. 30–44.

[45] Patel, K.; Parameswaran, S.; Ragel, R. G. “Architectural frameworks for security and reliability of MPSoCs”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19–9, Sep 2011, pp. 1641–1654.

[46] Poschmann, A.; Leander, G.; et al.. “A family of light-weight block ciphers based on DES suited for RFID applications”. In: Fast Software Encryption (FSE), 2006, pp. 16.

[47] Ramachandran, J. “Designing Security Architecture Solutions”. John Wiley & Sons, Inc., 2002, 452p.

[48] Russell, S. J.; Norvig, P. “Artificial Intelligence: A Modern Approach”. Pearson Education, 2003, 2 ed., 1132p.

[49] Salminen, E.; Kulmala, A.; Hamalainen, T. D. “On network-on-chip comparison”. In: Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD), 2007, pp. 503–510.

[50] Sepúlveda, J.; Flórez, D.; Gogniat, G. “Reconfigurable group-wise security architecture for NoC-based MPSoCs protection”. In: Symposium on Integrated Circuits and Systems Design (SBCCI), 2015, pp. 6.

[51] Sepúlveda, J.; Flórez, D.; et al.. “Towards risk aware NoCs for data protection in MPSoCs”. In: International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2016, pp. 1–8.

[52] Sepúlveda, J.; Gogniat, G.; et al.. “Hierarchical NoC-based security for MP-SoC dynamic protection”. In: Latin American Symposium on Circuits and Systems (LASCAS), 2012, pp. 1–4.

[53] Sepúlveda, J.; Gogniat, G.; et al.. “Hybrid-on-chip communication architecture for dynamic MP-SoC protection”. In: Symposium on Integrated Circuits and Systems Design (SBCCI), 2012, pp. 1–6.

[54] Sepúlveda, J.; Gogniat, G.; et al.. “Security-enhanced 3D communication structure for dynamic 3D-MPSoCs protection”. In: Symposium on Integrated Circuits and Systems Design (SBCCI), 2013, pp. 1–6.

[55] Sepúlveda, J.; Gogniat, G.; et al.. “Security-enhanced 3d communication structure for dynamic 3D-MPSoCs protection”. In: Symposium on Integrated Circuits and Systems Design (SBCCI), 2013, pp. 1–6.

[56] Sepúlveda, J.; Gogniat, G.; et al.. “3D-LeukoNoC: A dynamic NoC protection”. In: International Conference on ReConFigurable Computing and FPGAs (ReConFig14), 2014, pp. 1–6.

[57] Sepúlveda, J.; Gogniat, G.; et al.. “Elastic security zones for NoC-based 3D-MPSoCs”. In: International Conference on Electronics, Circuits and Systems (ICECS), 2014, pp. 506–509.

[58] Sepúlveda, J.; Lozada, G.; Strum, M. “A tlm-based network-on-chip performance evaluation framework”. In: Iberoamerican network of Microelectronics (IBERCHIP), 2007, pp. 4.

[59] Sepúlveda, M. J.; Diguet, J. P.; et al.. “NoC-based protection for SoC time-driven attacks”, *IEEE Embedded Systems Letters*, vol. 7–1, Mar 2015, pp. 7–10.

[60] Sgroi, M.; Sheets, M.; et al.. “Addressing the system-on-a-chip interconnect woes through communication-based design”. In: Design Automation Conference (DAC), 2001, pp. 667–672.

[61] SPLASH-2. “SPLASH-2”. Source: http://www.capsl.udel.edu/splash/, Aug 2015.

[62] Svensson, L.; Monteiro, J. “Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation”. Springer Berlin Heidelberg, 2009, 366p.

[63] Tsai, W.-C.; Chu, K.-C.; et al.. “Non-minimal, turn-model based NoC routing”, *Microprocessors & Microsystems*, vol. 37–8, Nov 2013, pp. 899–914.

[64] Wassel, H. M. G.; Gao, Y.; et al.. “Networks on chip with provable security properties”, *IEEE Micro*, vol. 34–3, May 2014, pp. 57–68.

[65] Ye, T. T.; Benini, L.; De Micheli, G. “Packetization and routing analysis of on-chip multiprocessor networks”, *Journal of Systems Architecture: the EUROMICRO Journal*, vol. 50–2-3, Feb 2004, pp. 81–104.