UNIVERSITY OF CALIFORNIA,
IRVINE

Three-Dimensional NoC Reliability Evaluation Automated Tool (TREAT)

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Electrical and Computer Engineering

by

Ashkan Eghbal

Dissertation Committee:
Professor Nader Bagherzadeh, Chair
Professor Jean-Luc Gaudiot
Professor Pai H. Chou

2016

DEDICATION

I dedicate my dissertation work to my parents and many friends. A special feeling of gratitude to my loving parents, for their endless love, support, and encouragement. I also dedicate this dissertation to my many friends who have supported me throughout the process.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my adviser, Prof. Nader Bagherzadeh for offering the opportunity for a PhD in addition to his guiding and supporting me over the years. He has set an example of excellence as a researcher, mentor, and role model.

I would like to thank my thesis committee members, Prof. Jean-Luc Gaudiot and Prof. Pai H. Chou, for their insightful comments; their discussions, ideas, and feedbacks have been absolutely invaluable.

I would like to thank my labmate, Pooria M. Yaghini and my collaborator Siavash S. Yazdi, who contributed to this research. I am very grateful to both of you.

I would like to thank IEEE, Elsevier, and ACM publishers who have given me the permission to incorporate my previous published work with them into my dissertation.

CURRICULUM VITAE

Ashkan Eghbal

EDUCATION:

- **Doctor of Philosophy in Computer Engineering** — **2016**
  University of California, Irvine — *Irvine, California*

- **Master of Science in Computer Engineering** — **2010**
  Amirkabir University of Technology of Tehran — *Tehran, Iran*

- **Bachelor of Science in Computer Engineering** — **2007**
  Azad University of Iran, Central Branch — *Tehran, Iran*

RESEARCH EXPERIENCE:

- **Graduate Research Assistant** — **2010–2016**
  University of California, Irvine — *Irvine, California*

- **Graduate Research Assistant** — **2008-2010**
  Amirkabir University of Technology of Tehran — *Tehran, Iran*

TEACHING EXPERIENCE:

- **Lecturer** — **Long Beach, California**
  Microprocessors and Controllers III — Spring 2016
  Data Structure and Algorithms
  Digital Circuit Design I — Fall 2015
  Computer Organization and Assembly Programming
  Dept. of CECS, California State University of Long Beach

- **Teaching Associate** — **Irvine, California**
  Introduction to Digital Design Laboratory — Summer 2015
  Dept. of EECS, University of California, Irvine

- **Teacher Assistant** — **Irvine, California**
  Introduction to Digital Design Laboratory — Fall 2013, Summer 2014
  Computer system and C programming — Spring 2012/2015, Summer 2013

Computer Organization — Winter 2012/2013/2014, Fall 2014/2015
Dept. of EECS, University of California, Irvine

• **Peer Mentor** — **Irvine, California**
International Students of University of California, Irvine — Fall 2011/2012/2013

SELECTED HONORS AND AWARDS:

• **Author of featured paper** — **December 2015**
IEEE Transaction on Computers (ToC)

• **Awarded NSF travel grant** — **May 2015**
International Symposium on Computer Architecture (ISCA)

• **DAC Young Student Support Program Award** — **May 2015**
52^{nd} Design Automation Conference (DAC), 2015

• **AGS Travel grant Award (UCI)** — **March 2015**
To present my accepted paper in VTS 2015
33^{th} VLSI Test Symposium (VTS), 2015 IEEE

• **Second place of IEEE Orange County and** — **Oct 2013**
Western Digital Student Design Contest
11^{th} International System-on-Chip Conference

• **First place of IEEE Orange County and** — **Oct 2012**
Western Digital Student Design Contest
10^{th} International System-on-Chip Conference

• **First place of IEEE Orange County and** — **Nov 2011**
Western Digital Student Design Contest
9^{th} International System-on-Chip Conference

• **Graduate Research Student Fellowship** — **2010–2011**
EECS Dept. of University of California Irvine

PROFESSIONAL MEMBERSHIPS:

Institute of Electrical and Electronics Engineers (IEEE) — **Since 2013**

PUBLICATIONS:

Journal papers:

[J5] P.M. Yaghini, **A.Eghbal**, S.S.Yazdi, and N. Bagherzadeh, M.M. Green, "Capacitive and Inductive TSV-to-TSV Resilient Approaches for 3D ICs, " IEEE Transactions on Computers, vol.65, no.3, pp.693-705, 2016.

[J4] **A.Eghbal**, P.M. Yaghini, M. Khayambashi, and N. Bagherzadeh, "Analytical Fault Tolerance Assessment and Metrics for TSV-based 3D Network-on-Chip, " IEEE Transactions on Computers, vol.64, no.12, pp.3591-3604, Dec 2015.

[J3] P.M. Yaghini, **A.Eghbal**, M. Khayambashi, and N. Bagherzadeh, "Coupling Mitigation in 3D Multiple-Stacked Devices, " IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.23, no.12, pp.2931-2944, 2015.

[J2] P.M. Yaghini, **A.Eghbal**, and N. Bagherzadeh, "On the Design of Hybrid Routing Mechanism for Mesh-based Network-on-Chip, " Integration, the VLSI Journal - Elsevier, Vol50, pp. 183-192, 2015.

[J1] M. Khayambashi, P.M. Yaghini, **A.Eghbal**, and N. Bagherzadeh, "Analytical Reliability Analysis of 3D NoC Under TSV Failure, " ACM Journal on Emerging Technologies in Computing Systems (JETC), vol. 11, no.4, pp. 43:1-43:16, 2015.

Conferences:

[C5] F. Verbeek, P.M. Yaghini, **A.Eghbal**, N. Bagherzadeh, "Deadlock Verification of Cache Coherence Protocols and Communication Fabrics," accepted in DATE 2016.

[C4] P.M. Yaghini, **A.Eghbal**, S.S.Yazdi, and N. Bagherzadeh, "Accurate System-level TSV-to-TSV Capacitive Coupling Fault Model for 3D-NoC," Proceedings of the 9th International Symposium on Networks-on-Chip, pp. 3:1–3:8, 2015.

[C3] **A.Eghbal**, P.M. Yaghini, and N. Bagherzadeh, "Capacitive Coupling Mitigation for TSV-based 3D ICs, "The 33th International IEEE VLSI Test Symposium (VTS), pp.1-6, April 2015.

[C2] **A.Eghbal**, P.M. Yaghini, S.S.Yazdi, and N. Bagherzadeh, "TSV-to-TSV Inductive Coupling-Aware Coding, "IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, pp. 92-97, 2014.

[C1] P.M. Yaghini, **A. Eghbal**, and N. Bagherzadeh, "A GALS Router for Asynchronous Network-on-Chip, " The 2nd ACM International Workshop on Many-core Embedded systems, pp. 52-56, 2014.

ABSTRACT OF THE DISSERTATION

Three-Dimensional NoC Reliability Evaluation Automated Tool (TREAT)

By

Ashkan Eghbal

Doctor of Philosophy in Electrical and Computer Engineering

University of California, Irvine, 2016

Professor Nader Bagherzadeh, Chair

Technology scaling and higher operational frequencies are no longer sustainable at the same pace as before. The processor industry is rapidly moving from a single core with high-frequency designs to many-core with lower frequency chips; Network-on-Chip (NoC) has been proposed as a scalable and efficient on-chip interconnection among cores. In addition, employing Three-Dimensional (3D) integration instead of Two-Dimensional (2D) integration is the other trend to keep the traditional expected performance improvements. The combination of 3D integration and NoC technologies provides a new horizon for on-chip interconnect design. In more detail, the reduction of the length and number of global interconnects; by applying Through-Silicon Via (TSV) is the major advantage of 3D NoCs.

However, shrinking transistor sizes, smaller interconnect features, and 3D packaging issues, lead to higher error rates and unexpected timing variations. Although many researches have focused on reliability issues for 3D NoC architectures, To develop a general technique to advance both the intuitive understanding and the quantitative measurement of how potential physical faults influence the behavior of 3D NoC is lacking. The goal of my dissertation is to develop a Three-Dimensional NoC Reliabil-

ity Evaluation Automated Tool (TREAT), for the first time, as an automated analysis tool to analyze effects of static and dynamic faults in 3D NoC architectures. It is capable of evaluating the vulnerability of different architectural components in the presence of faults by using the fault injection method. This approach allows injecting faults into the 3D NoC platform dynamically by monitoring the status of links and components to decide where and when inject faults accurately. TREAT provides the strength of different components in terms of reliability-based metrics such as Mean Time Between Failure (MTBF) and header/data/trailer flit failure rate for different level of granularity. The output reports of TREAT are critical in devising fault-tolerant techniques with low overhead cost. TREAT can be used at the early stage of the design process in order to prevent costly redesigns after assessing dependability for the target architecture.

Comparing to existing fault injector tools, TREAT is specifically developed for 3D NoC platforms and it is not a general fault injector tool. Such a tool is needed since the characteristics and behavior of a 3D NoC component is different from other computational platforms; 3D NoCs are susceptible to different fault sources that are not fully addressed by existing tools. Furthermore, one of the most important advantages of TREAT is supporting dynamic fault injection by monitoring the status of the NoC platform. This is critical since based on the reported experiments in this dissertation, random TSV coupling fault injection may result in 26%-99% inaccuracy of reliability evaluation process. The fault injector interface is responsible for injecting fault accurately where and when they should in order to enhance the reliability evaluation. None of the existing tools offer these capabilities as a single package.

Chapter 1

Introduction

Since the technology scaling does not treat wire delay and gate delay equally, many-cores' interconnection design became challenging. Consequently, a novel communication method is required to take the benefits of many-core designs as they are highly demanded for both embedded systems and super-computers. Additionally, the reliability characteristic is of crucial importance in today's High Performance and Energy Efficient Computing (HPEEC) applications since the malfunctioning of a component results in lower performance and waste of energy, [112]. Therefore, similar to any new technology the reliability of the proposed communication methodologies is a crucial issue.

With this in mind, a reliability analysis tool is needed to uncover the sensitive components and to evaluate the efficiency of the proposed robust designs. Although many general analysis tools have been presented especially for investigating the reliability properties of the processors, these tools are not accurate enough to reveal fault tolerance characteristics of the next generation on-chip communication links.

The purpose for this research is to develop a specific reliability analysis tool, to

address the unexpected effects of physical faults in on-chip interconnections statically and dynamically. In the rest of this chapter, the necessity and the basic concepts of on-chip links with their reliability issues are briefly discussed. Additionally, the available reliability analysis tools are also introduced and compared with the one which is proposed in this dissertation. Then the contributions of this research are listed and discussed and finally, the outline of the dissertation is provided.

1.1 The Future Design Trends

The three well known factors in traditional high performance Processing Element (PE) designs are: Instruction Level Parallelism (ILP), gates per clock, and process technology. However, the first two factors have already reached their limits [40, 67]. The problem of process technology scaling is more related to wire issues and energy consumption concerns rather than the device design challenges. Applying higher frequencies results in more energy consumption, clocking synchronization concerns, and signal/power integrity issues. Moreover, the unexpected heat generation in PEs is already known as a major concern though they have never been super-fast yet. In addition, technology and supply voltage scaling slows down due to physical limitation in nano fabrication feature size. The leakage energy consumption, lithography complexity, and process variation issues are some critical examples of these physical constraints. On the other side, there are plentiful of computational applications in the market that demand more complex processing calculations such as smart phones, gaming consoles, or weather forecast applications. With these limitations there are significant concerns in moving forward to follow the *Moore's* law for next generation of IC designs. Therefore, different solutions have been suggested by researchers to exploit the technology capacity and keep the traditional performance improvements

such as: proposing many-core designs, Three-Dimensional (3D) integration, and new generation of interconnections [67, 33, 89]. Each of these solutions is discussed separately in the following subsections.

1.1.1 Moving from Single Core to Many-core

With each passing year, the number of problems where experiments are impossible, dangerous, or inordinately costly is growing. To provide vastly more accurate predictive models and the analysis of massive quantities of data are some examples of these problems [2]. In order to meet performance demands, parallel computing is ample motivation for proposing many-core architectures as predicted by ITS reports which is shown in Figure 1.1. Based on the represented trend in Figure 1.1 the number of cores will be increased by a factor of 10 by the next decade.

However, many-core popularity is not only limited to super-computer applications. Modern embedded systems also require massive processing power due to computationally intensive embedded applications, such as self-driving automobiles' control unit, real-time speech recognition, video conferencing, software-defined radio, and cryptography. With these considerations it is concluded that, many-core design is the only de facto solution for HPEEC applications [42], which has been already found its way into mass production. Two major benefits of replacing a high frequency single core chip by the many-core designs with a reasonable frequency are providing better level of resource sharing and occupying less Printed Circuit Board (PCB) space. The other advantage of many-core design is increasing the raw performance by enhancing the clock per instruction factor rather than frequency, which consequently results in slower growth in energy consumption. Therefore, many-core architecture manifests a good tradeoff between technology scaling and limited energy budget requirement.

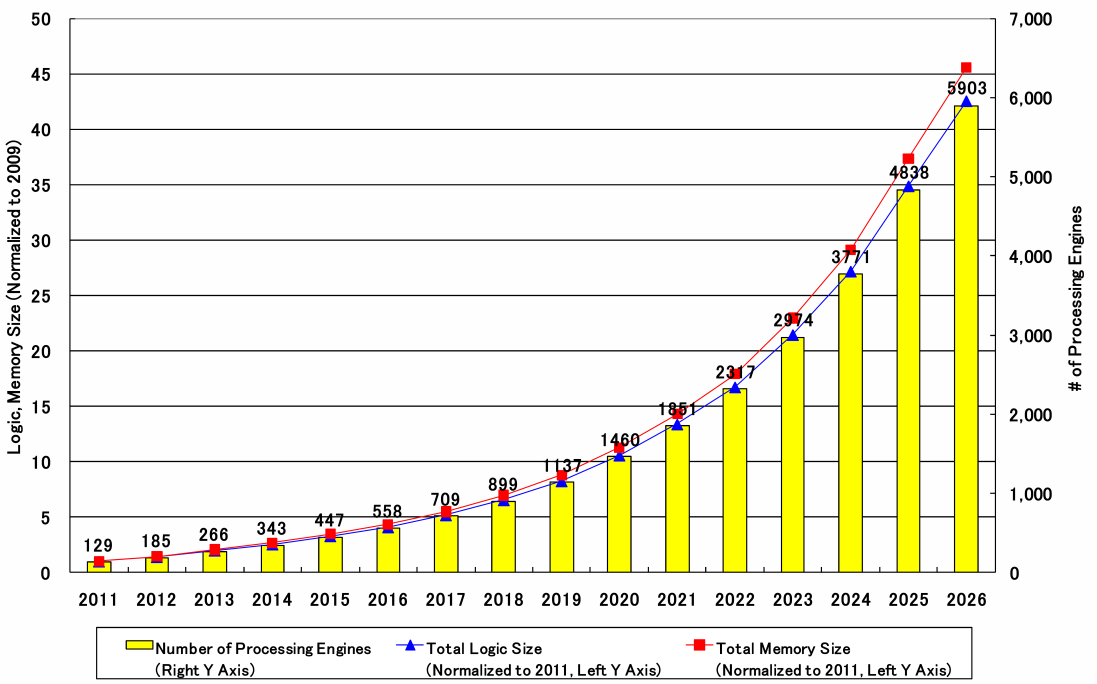


Figure 1.1: SoC consumer portable design complexity trends [44]

1.1.2 Moving from 2D to 3D Integration

The other solution which has been recently become popular is to add one more dimension to the 2D state of the art architectures (resulting in 3D designs). This method provides numerous opportunities to connect PEs and memory cells on one package with larger bandwidth. The 3D memory, known as Hyper Memory Cube (HMC), is an industrial example of applying this technique [47, 105]. Figure 1.2 shows the graphical structure of 3D memory in which memory banks are stacked on top of each other and connected through vertical interconnection known as Through Silicon-Via (TSVs). One of the promises of HMC is that it eliminates the so-called "memory wall," the problem of the disparity between CPU clock speed/bandwidth and the memory systems [24]. The 32-layer V-NAND flash memory is the other example of this approach which has been started for mass production by Samsung Company. In addition to industrial examples, the 3D cache architecture has become a popular topic of study among researchers [128, 37, 124].

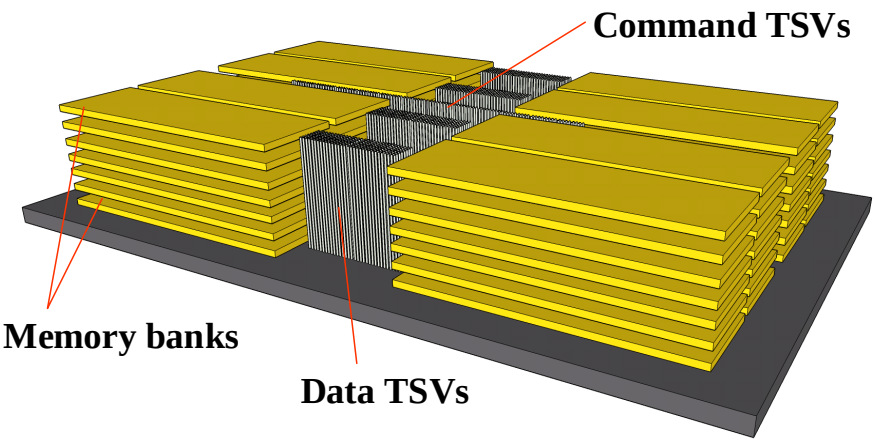


Figure 1.2: TSV-based 3D multiple-stacked memory

In general, 3D ICs are proposed as a viable solution for integrating more PEs on a chip, without the need for extra footprint area [126]. According to the characteristics of the 3D architectures, data transmission is done both horizontally and vertically at the same time. Moreover, 3D designs provide shorter interconnect links in which cores are usually located centimeters apart on a chip are now could be placed millimeters apart from each other vertically. Therefore, as 3D ICs compared to the 2D ones are expected to support higher performance, lower latency, smaller form factor, and less energy consumption.

1.1.3 Moving from Bus-based to Network-based Interconnects

The chip data transmission needs more attention as the complexity of systems increases, in a way that devising a new communication infrastructure is necessary. In other words, the traditional, bus-based core communication infrastructures, are considered as one of the performance bottleneck in many-core systems. The reason is that the wire delays have become more critical in smaller feature size, as technology

scaling results in larger capacitance for global wires. It has been reported that under 45nm technology size, Metal-1 and intermediate-level metal wires (local wires) are expected to have the same line-widths and thicknesses, demonstrating similar process variation effects. The effect of process variations on local metal layers are expected to have smaller effect on overall circuit delay because of their relatively shorter lengths as compared to global wires [125]. Therefore, global wires are not scalable in terms of delay, energy consumption, and bandwidth. In addition, global clocking is another challenge in traditional bus-based interconnections. The clock tree method has been suggested to resolve this issue, but it consumes large portion of energy and area budget. In addition, due to clock skew effect even larger portion of the total cycle time [67] is needed. It should be also added that a bus-based system has a limited capability of concurrent communication. The intrinsic resistance and capacitance of the bus also increase by connecting more number of cores. Moreover, buses broadcast the data to transfer them among the cores in which the entire bus wires should be switching from on to off state periodically; resulting in large waste of energy. Because of these important considerations, the Network-on-chip (NoC) architecture has been proposed as a practical alternative for the traditional system-on-chip (SoC) approach, supporting better modularity, scalability and higher bandwidth [29, 92, 67]. In fact, an NoC consists of multiple point-to-point data links interconnected by routers to implement the connectivity of on-chip Intellectual Properties (IPs).

1.1.4 3D NoC

NoCs have become significant over the past 10 years in order to fully utilize the integration capacity with billion-transistor on a chip. However, because of the aforementioned benefits of 3D IC designs, as discussed in subsection 1.1.2, the 2D NoC designs must be expanded to support 3D schemes [89]. The combination of 3D inte-

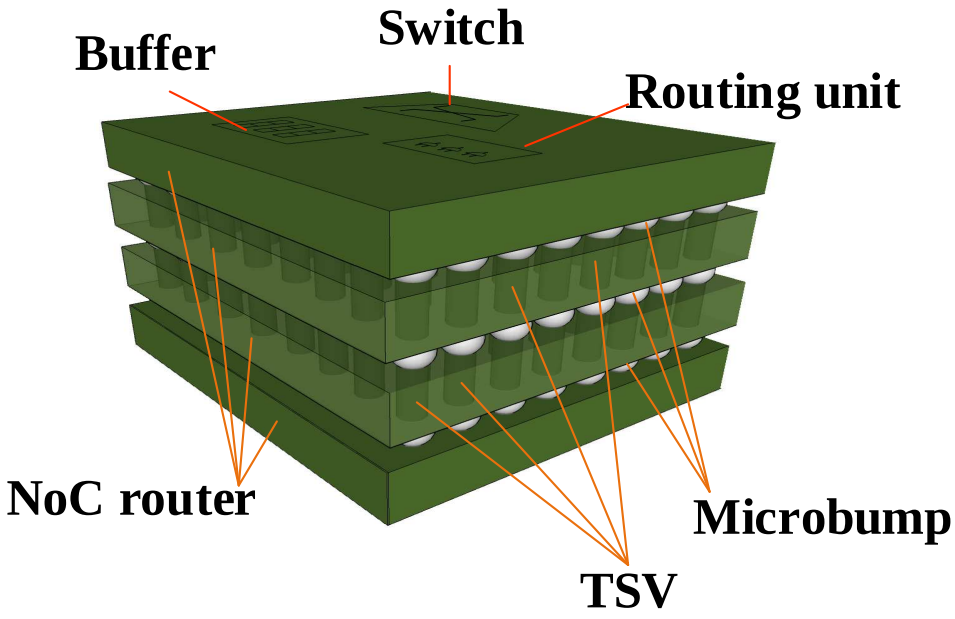


Figure 1.3: Side view of 3D NoC

gration and NoC technologies provides a new horizon for on-chip interconnects design, which combines the benefits of short vertical interconnects of 3D ICs and the scalability of NoCs. Compared with 2D NoCs, 3D NoCs greatly reduce the network diameter and overall communication distance; they improve communication performance and reduce energy consumption [89].

The TSV is currently the most popular technology to connect 2D NoC routers vertically. They are actually large vertical wires which are surrounded by an insulation and depletion layers. Additionally, TSVs are very short and fine-pitched links, which provide high-bandwidth communication channels. They are also highly compatible with the standard CMOS process which makes them capable of integrating into existing circuits without extra design cost.

In more details, 2D NoC routers are connected by TSVs (inter-die interconnects) through connectors with small diameters called Microbumps [19]. Microbumps are actually the metal connectors on both topside and backside of the chip which are

integrated into the copper filled TSV process for electrical connection between chips. Figure 1.3 shows 3D structure of stacked 2D NoC routers vertically connected by Microbumps and TSVs. The buffer, crossbar switch, and routing unit are the fundamental components of a 2D NoC router, which are individually explained in Section 3.2. In fact, TSVs implement a serial data transmission method in which each TSV is capable of single bit data transmission at a time, sequentially. This is why each NoC router is connected to several TSVs and Microbumps as shown in Figure 1.3. It should be noted that, due to large sizes of TSVs, all the routers cannot be connected to the routers of the other layers. Therefore, vertically partially connected 3D NoCs are more practical and have been the focus of majority of researchers instead of vertically fully connected 3D NoCs.

On the other side, 3D NoC architectures demand many trade-offs in order to meet some specific performance requirements, which results in various design issues. One of these major design concerns is the reliability of 3D NoC which is the goal of this research. The reliability issue in 3D NoC is briefly described in subsection 1.2 and further discussed in Chapter 4.

1.2 Reliability of 3D NoC

To design a robust computing circuit has been considered for many years ago. Recently it has received a great deal of attention as the CMOS technology approaches the nanoscale level. Shrinking transistor sizes, smaller interconnect features, 3D packaging issues, and higher frequencies of CMOS circuits lead to higher error rates and more thermal and timing violations [70]. Many approaches have been proposed to enhance the robustness of NoCs, mostly in the areas of routing algorithms [88, 82], communication infrastructure [12, 76, 77, 98], or micro-architecture designs [57, 74, 29].

Many fault-tolerant routing algorithms have been proposed for both 2D [23] and 3D NoC [79] designs. The idea of bypassing faulty data paths within failed routers has been suggested as a lightweight fault-tolerant method [58]. A fault-aware IP-core mapping to NoC routers is proposed in [69]. They address the problem of transient link failures by means of temporally and spatially redundant transmission of messages. A stochastic communication paradigm is proposed in [12] to provide a fault-tolerant communication. This approach is based on probabilistic broadcast where packets are forwarded randomly to the neighboring nodes. However, none of them protect faulty datapath inside routers. In [9], the authors deploy some error correcting schemes to achieve combined energy minimization and reliability optimization design goals. A fault-tolerant router architecture is proposed in [29] to guarantee the functionality of NoC in the presence of faults. They detect the most tenuous components of router against different sources of faults first and then tolerate them to save power and area overhead of employing fault-tolerant techniques. An adaptive fault-tolerant NoC router with the power consumption consideration is proposed in [74]. They improve error correction capability of their design based on the frequency of fault occurrence. A novel bidirectional fault-tolerant NoC architecture capable of mitigating both static and dynamic channel failures is also proposed [108].

However, in all of these approaches the dependability of NoC design is improved, regardless of imposing redundancies (hardware, information, and time) or based on reliability analysis reports with general tools. Also, there is no measurement capability to compare the effectiveness of the proposed reliable 3D NoC architectures. In fact, experimental and analytical techniques are two popular methods to explore the dependability of systems. Categorizing the effects of potential faults on the performance of a system is a necessity for both techniques. An analytical model for reliability evaluation of 2D NoC has been presented in [21], but it is not reusable for 3D die-stacking designs in which new sources of faults are presented.

Many major reliability issues are presented by the emergence of 3D structures such as thermal concerns and TSV issues as a result of 3D structures. The unwanted heat generation by PEs in 3D NoC should be removed effectively otherwise it may result in other problems. Three major effects of heat generation are increasing the device temperature, generating thermal stress, more thermomigration as a result of cooling down methods. In addition, TSVs may also cause unwanted noise sources as a result of inductive and capacitive coupling effects. These effects are different from the traditional crosstalk issues due to differences between the physical characteristics of TSVs and 2D wires.They are still active areas of research. Additionally, the generated noise signals by TSVs, may also have undesirable effect on the performance of 2D components such as transistors or 2D wires. A minimum distance between TSVs and these components is required to prevent undesirable effects of TSVs impacting the functionality of the 3D NoC architectures. Furthermore, there are still additional issues for developing 3D architecture EDA tools [14]. Accuracy of simulated environment is a concern for experimental methods in order to analyze dependability of a system. On the other hand, measurements are expensive and time consuming while time-to-market cycle is of great importance.

Consequently, developing general analytical techniques to advance both the intuitive understanding and the quantitative measurement of how potential physical faults influence the behavior of 3D NoC are still lacking although many researches have focused on reliability issues for 3D NoC architectures.

1.3 Contribution of this Thesis

The goal of this dissertation is to present an accurate reliability analysis tool for 3D NoC platform. The presented reliability analysis tool in this dissertation is called

Three-Dimensional NoC Reliability Evaluation Automated Tool (TREAT). TREAT is able to analyze effects of static and dynamic faults in 3D NoC architectures. This tool can be applied investigating the capability of proposed fault-tolerant 3D NoC approaches at the early stage of the design, which saves both time and cost of design. The major stages of my dissertation are:

- Discovering the source of physical faults in 3D NoC and categorizing them based on their possible effect on the functionality and performance of the 3D NoC. In more details, all possible potential sources of physical faults in 3D NoC environment have been studied. The impacts of all potential physical faults on 3D NoC components are also addressed to be used for reliability metrics in TREAT. All main components of a 3D NoC router architecture and sensitive entities with their possible responses to physical faults are categorized in this dissertation [26].

- Designing and developing an HDL environment of 3D NoC router as a platform of running the experiments, called Three-Dimensional HDL Emulator NoC (THENoC). As a side project an optimized on-chip routing algorithm is also proposed which takes advantages of both source routing and distributed routing algorithms for 2D NoC and scalable to 3D NoC designs. This method eliminates the heavy routing process of intermediate routers for each packet from source toward its destination by imposing a light overhead independent from the network size [119].

- Modeling the logic-level of all possible physical faults in 3D NoC router and defining the property of injected faults dynamically by profiling the status of different components. These logic-level faults are modeled and applied as fault libraries in TREAT. By comparing the value of selected signals and their assertion time in 3D NoC components after running both faulty and faultless experiments the vulnerability of each component is reported. The effect of psychical fault models which are discussed in Chapter 4 are used as evaluation metrics in TREAT.

- Automating the fault injection and reliability evaluation processes and verifying its functionality with THENoC. It is capable of implementing static fault injection based on the provided information by the user or to accomplish dynamic fault injection process based on the analyzed feedback from the simulator. For static fault injection process this experiment is repeated as many times as the user defines for various fault activation time and periods, while a single experiment is needed for dynamic fault modelings.

1.4 Similar Tools

Fault injection is one of the popular method of reliability analysis which is addressed in Chapter 2. The impacts of faults have been evaluated by several research groups for various design objectives. Different methods are proposed with the goal of fault injection based on the analysis such as software fault injection, simulation fault injection and physical level fault injection.

Physical-based fault injection involves augmenting the system under analysis with specially designed test hardware to allow for the injection of faults into the system and examine their effects. Although the experimental process in this method is fast, but the experimental set up is time consuming. In addition, this method needs the actual hardware in order to run the fault injection process, resulting in a costly method. Software-based fault injection includes the modification of the running software on the system under analysis in order to modify the system state according to the programmers modeling view of the system. This is popular for cases that have communicative or cooperative functions so that there is enough interaction to make fault injection useful. Comparing to physical-based fault models they do not need the actual model of the system, but they have some limitation in terms of fault injection coverage and

observability. Finally, simulation-based fault injection method requires an accurate model of the system under analysis. The accuracy of the model and simulation time, are major concerns for these approaches, but they have full control of fault injection and fault modeling. This approach is not expensive either, since it does not need the actual hardware of the design. Because TREAT is a simulation-based fault injector tool, various simulation-based fault injector tools are introduced to show they are not appropriate for 3D NoC reliability analysis objective.

- **MEFISTO-C**: is a VHDL-based fault injection tool developed at Chalmers University of Technology, Sweden that conducts fault injection experiments using VHDL simulation models. The tool is an improved version of the MEFISTO which was developed jointly by LAAS-CNRS and Chalmers. Also a similar tool called MEFISTO-L has been developed at LAAS-CNRS. MEFISTO-C uses the vantage optimum VHDL simulator and injects faults via simulator commands in variables and signals defined in the VHDL model. It offers the user a variety of predefined fault models as well as other features to set-up and automatically conducts fault injection campaigns on a network of UNIX workstations. This tool generates modified components descriptions called mutants. The mutants generation is based on a static information and the model has to be recompiled for each experiment [34].

- **VHDL-based Evaluation of Reliability by Injection Faults Efficiently** (VERIFY): is developed at University of Erlangen-Nurnberg, Germany. VERIFY uses an extension of VHDL for describing faults correlated to a component, enabling hardware manufacturers, which provide the design libraries, to express their knowledge of the fault behavior of their components. Multi-threaded fault injection which utilizes checkpoints and comparison with a golden run is used for faster simulation of faulty runs. The proposed extension to the VHDL language requires modification of the VHDL language itself. VERIFY uses an integrated fault model, the

dependability evaluation is very close to that of the actual hardware [101].

- **HEARTLESS**: is a hierarchical register-transfer-level fault-simulator for permanent and transient faults that was developed, by CE Group-BTU Cottbus in Germany, to simulate the fault behavior of complex sequential designs such as processor cores. Furthermore, it serves for the validation of online test units of embedded processors. It can support permanent stuck-at faults, transient bit flip and delay faults. HEARTLESS was developed in ANSI C++. The whole design or parts (macros) can be selected for fault simulation based on fault list generation. Fault list is reduced according to special rules derived from logic level structures and signal traces. HEARTLESS can be enhanced by utilizing macros described in a C-function [93].

- **GSTF**: is another VHDL-based fault injection tool developed by Fault Tolerance Systems Group at the Polytechnic University of Valencia, Spain. This tool is presented as an automatic and modeling dependent fault injection tool for use on an IBM-PC or compatible system to inject faults into VHDL models (at gate, register and chip level). The tool has been incorporated into a commercial VHDL simulator (V-System by Model Technology) and can implement the main injection techniques: Simulator commands, saboteurs and mutants. Both transient and permanent faults, of a wide range of types, can be injected into medium-complexity models. The tool can inject a wide range of fault models, surpassing the classical models of stuck-at and bit-flip and it is able to analyze the results obtained from the injection campaigns, in order to study the error syndrome of the system model and/or validate its fault-tolerance mechanisms [6].

- **Fault Tolerance Injection** (FTI): is developed at Universidad Carlos III de Madrid in Spain, for fault-tolerant digital ICs in the RT abstraction level. The main objective of FTI is to generate a fault-tolerant VHDL design description. Designer provides an original VHDL description and some guidelines about the type

of fault-tolerant techniques to be used and their location in the design. FTI tool processes original VHDL descriptions by automatic insertion of hardware and information redundancy. Therefore, a unified format to deal with descriptions is needed. There are several intermediate formats that represent, by means of a database, the VHDL description in a formal way that could be accessed and processed with some procedural interface. Fault-tolerant components to be included into VHDL original descriptions have already been described and stored in a special library called FT library. These components come from previous researches about FT and designer just use them. FTI use an intermediate format for VHDL descriptions (FTL/TAURI) and it will work only with synthesizable descriptions IEEE 1076 [32].

Many other fault injector tools have been presented for various design objectives by research groups in addition to the famous ones which are listed in this section. However, none of the existing tools are efficiently reusable for 3D NoC platforms, since 3D NoCs are susceptible to different fault sources that are not fully addressed by existing tools like 3D design issues. Furthermore, components and behavior of a 3D NoC is different from other computational platforms as discussed in [26]. ARROW [11] is the only fault injector tool for NoC platform, which only supports 2D NoC design and it does not provide any reliability analysis. TREAT is specifically developed for 3D NoC platforms and it is not a general fault injector tool. Additionally, one of the most advantages of TREAT is its capability of intelligent fault injection by monitoring the status of the NoC platform (dynamic faults), while most of existing fault injector tools inject faults randomly. This is critical since based on our experiments; random fault injection may result in 26%-99% inaccuracy of reliability evaluation process. The presented tool is capable of evaluating the Verilog model of 3D NoC architecture or fault-tolerant techniques. TREAT includes the following features which none of the mentioned tools offer them all in one package:

- Automatic fault-injection
- Rich set of fault models
- Designed specifically for NoC
- Single and multi-bit fault activation
- Dynamic and static fault models
- 3D IC-specific fault models
- NoC specific reliability analysis and reporting

1.5 Dissertation Outline

The rest of the thesis is organized as follows: Chapter 2 discovers the basic terminologies in fault-tolerant and fault evaluation communities which are used in other chapters. The basic components for a general fault injector tool are also discussed in this chapter. The 3D NoC architecture is explained in detail in Chapter 3, which is needed to recognize the major causes of fault occurrence and their effects on a 3D NoC design. The presented 3D NoC architecture in Chapter 3 is also applied in this dissertation to as a platform to run the experiments. The reliability challenges in 3D NoC interconnection are uncovered in Chapter 4, with their logic-level fault models. The effects of each of presented physical fault on the functionality of 3D NoC design are also discussed in this chapter. The framework of TREAT is introduced in Chapter 5. The experimental results of both static and dynamic fault injection processes are also presented in Chapter 6 to verify the capability of TREAT. Finally, Chapter 8 concludes the dissertation and discusses the potential directions for future work of this research.

Chapter 2

Dependability Threats and Means

To guarantee the correctness of computing and communication services of many-core systems, has always been a major challenge for both designers and end users. Designers need to develop more complex architectures which are capable of providing the expected services even in a faulty environment with limited energy consumption and area overhead. On the user's side, adding the dependability and fault tolerance features to electronic devices convey higher market prices. As an example, the price of the DDR memory slots with ECC (Error Correction Code) functionality is almost double the price of elemental components. In general, dependability is defined as the ability to deliver service that can justifiably be trusted.

A systematic exposition of the concepts of dependability consists of three parts: the threats to, the attributes of, and the means by which dependability is attained. There are many well-known attributes for dependability such as: reliability, availability, safety, security, survivability, maintainability, and other well known attributes. As discussed in Chapter 1, the goal of this dissertation is to provide a reliability analysis tool for 3D NoC environments which is considered one of the major attributes of the

system dependability. The dependability threats and means terminologies in addition to the concept of fault injection which are all needed for discussions of the following chapters are addressed in this chapter.

2.1 Dependability Threats

A system may not always perform as it is expected to, because of environmental conditions or internal faults. Higher temperature or voltage, neutrons and alpha particles, interconnect noises, electromagnetic interferences, and electrostatic discharge are some examples of undesirable environmental conditions for on-chip circuits. Each of these environmental conditions may result in an internal fault and consequently causes other unexpected behavior in the system. In general, causes and consequences of deviations from the promised functionality of a system are known as dependability threats including: fault, error, and failure. Each of these terminologies is individually discussed in the following subsections [8, 3]:

2.1.1 Fault

Fault is a physical defect, imperfection, or flaw that occurs within some hardware or software component. A delay in an electronic circuit or missing the priority of operators in a math equation in a software program are some examples of faults.

A fault is classified into two major categories which are called active and dormant ones. A fault is active when it ends in an error, otherwise it is dormant. An active fault is either an internal or external fault. An internal fault was actually a dormant fault which has been triggered by the computation process or environmental conditions.

An active fault may again become dormant, awaiting a new trigger. Most of internal faults switch repeatedly between their dormant and active states. As an example assume there is a memory cell with a predefined operational temperature. If the environmental temperature has changed for an unexpected reason the memory cell will be infected by the thermal fault. This is an example of external fault in which a dormant fault is converted to an active one. Also as an example of internal fault assume a single bit is infected by a fault before it is accessed. In this case the infected single data bit represents as a dormant fault until it is approached. Once this data bit is read, it becomes an active fault [38]. In case of write access, the infected fault bit is called to be overwritten before causing an error.

In other words, faults can also be categorized based on their temporal persistence known as: transient, permanent, or intermittent. A transient fault has an active status for a limited time and never gets activated again till the end of system's simulation. However, a permanent fault exists in the system till the end of system's life time. In addition, many of the faults switch between their dormant and active states which are called intermittent faults. But their activation time and intervals' length are not necessarily the same.

2.1.2 Error

An error is defined as a deviation from accuracy or correctness which is the manifestation of a fault. To miss capturing the serial data bits by a memory cell because of delay fault or to generate an unexpected value as a result of missing the priority of operators are both error examples.

Similar to faults, an error can be categorized into two types which are: detected or latent errors. With applying error detection techniques; an error is called a latent

one, as long as it has not been recognized, otherwise it remains as latent error in the system [86]. As an example an error in a single memory bit of the cache entry caused by positive charge and kinetic energy of alpha particles, known as Single Event Upset (SEU) fault, can be either latent or detected. If the ECC decoder unit of the cache entry has recognized the result of SEU fault, it is detected; otherwise it keeps its latent status. In addition, a latent error may disappear or be overwritten before it is detected. But, most often many of the errors are propagated and consequently generate more errors or pass through the system which result in single or more failures.

2.1.3 Failure

A failure is the non-performance of some action that is due or expected. Such as packet loss in data transmission because of missing correct data capturing or missing a software event as a result of regenerating unexpected values are examples of failures. A component failure causes a fault for the system which it belongs to or the other components which are interacting with.

Figure 2.1 summarizes the relation between faults, errors, and failures. This figure also includes an example for each transition from a component's fault to failure which results in another component's fault and consequently a system failure. To illustrate this example assume there is a many-core system in which instructions are supposed to be prefetched from the memory, while the memory and cores are connected to each other through a network. If a fault in arbitration component of routing unit is activated; it will generate an unfair packet transmission through the system which consequently results in a packet latency failure. In a normal condition the maximum delay for each packet including instructions is predictable if the arbitration component works correctly, otherwise the packet will be captured with an unknown latency.

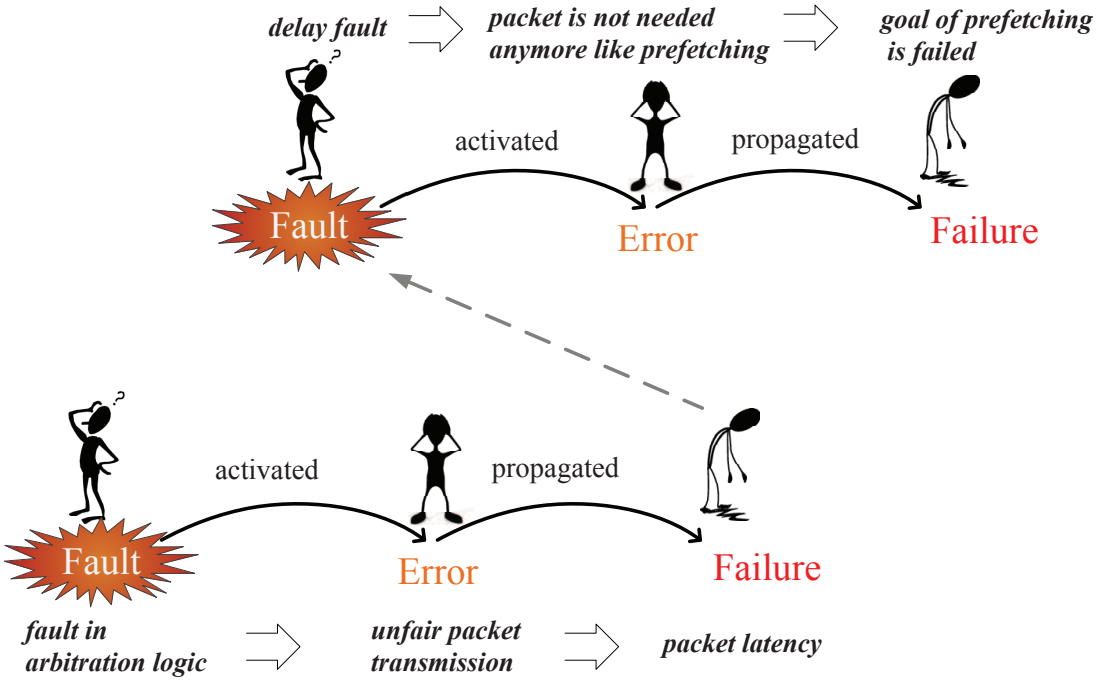


Figure 2.1: Fault, error, and failure

This failure can be considered as a kind of delay latency in the receiver side which is supposed to perform the prefetching operation. The activation of the delay fault results in an error in which the packet is not needed anymore. This error results in a system failure in the prefetching process. This example is a typical case especially in modern GPUs that are composed of many-core and memory units in which the prefetching technique has been proposed in order to increase the performance.

In addition to dependability threats, the means of dependability are also discussed in the next subsection to classify the types of methods discussed with the goal of resolving the unexpected behavior of faulty electronic circuits.

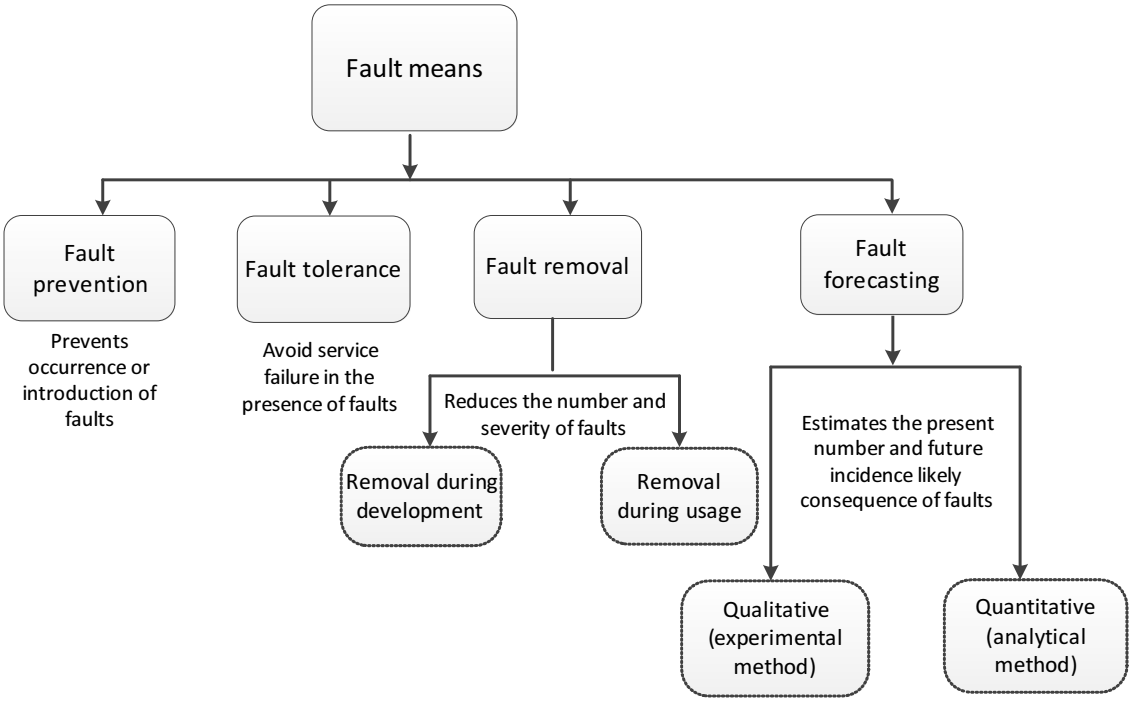


Figure 2.2: Fault means

2.2 Dependability Means

Dependability means can be grouped into four major classes to attain the expected level of dependability attributes. These categories are: fault prevention, fault tolerance, fault removal, and fault forecasting. Figure 2.2 summarizes these approaches which are discussed individually in the following subsections; through they are not independent from each other.

2.2.1 Fault Prevention

It includes all the methods which prevent the occurrence or introduction of faults. In more details, any hardware (such as hardware shielding and radiation hardening) or software activities (such as information hiding and modularized programming) at design time with the goal of protecting the system from different sorts of fault

occurrence is known as fault prevention. However, if the applied method is not resilient enough to prevent the fault occurrence, they cannot resolve the issue.

2.2.2 Fault Tolerance

These methods are proposed to protect the system in the presence of active faults to deliver its correct service. In higher level of abstraction fault-tolerant approaches intend to remove the errors from the system (known as error processing techniques) or to prevent the transition of dormant faults to be active ones (by applying fault handling methods). These two methods are discussed separately in the following paragraphs:

Error Processing

The error processing technique can be implemented by either of applying error recovery or error compensation approaches. For the error recovery approach, the erroneous state is replaced by an error-free state, while for the error compensation technique the integrated redundancies guarantee the delivery of error-free service. The error detection method is a critical step in error recovery approaches, for recognizing the erroneous state, resulting in the notion of self-checking component. The error recovery process is implemented in two different formats which are: backward recovery or forward recovery. For the backward recovery there is a transition from the current state to the state prior to error occurrence. As expected this method cannot resolve the permanent faults. On the other hand, for the forward recovery there is a transition from the current erroneous state to a new state from which the system can operate correctly.

In error compensation approach, the self-checking component is still needed. In this

method the execution of non-failed component is delivered to present the correct service, if the error is detected by the self-checking component. In more details, when error compensation is performed in a system, composed of self-checking components partitioned into classes executing the same tasks, there is simple switching within a class from a failed component to non-failed one. Additionally, error recovery approach imposes larger time overhead but less hardware or information redundancy as compared to the error compensation method.

Fault Handling

Fault handling is the other well-known technique proposed for a fault-tolerant design by preventing the located faults to be activated again. Fault handing is accomplished in four stages:

[C1] to locate and recognize the cause(s) of errors.

[C2] to prevent faults to be activated again which can be done by isolating the faulty identified components from the system.

[C3] to reconfigurable the system if it is not able to deliver the system without the faulty isolated component.

[C4] to reinitialize the system by checking and updating the new configuration of the system.

To select either of error processing and fault handling techniques and their implementation depends on the system requirement, overhead budget, and underlying fault assumption. It is critical that the fault-tolerant mechanisms are resilient enough against the faults that can affect them [86, 3]. Additionally, for any given fault-tolerant method, the fault hypothesis and the fault coverage are two major properties which should be considered by the designer. The fault hypothesis represents the types of faults which the proposed technique should tolerate and the fault coverage shows the

probability that the fault hypothesis is respected when a fault actually occurs in the system.

2.2.3 Fault Removal

This method can be applied both during the development phase or operational system's lifetime. During the development phase fault removal is done in three stages including: verification, diagnosis, and correction. In the verification stage it is checked to investigate if the system fulfills the given properties, which are described as verification conditions. If it does not, the other two steps follow: diagnosing the fault(s) that prevented the verification conditions, and then performing the required alteration. An important aspect of this technique during development stage is the verification of fault tolerance mechanisms. Verifying the capability of the proposed fault-tolerant methods is critical. Fault removal during the operational phase of a system's life is either corrective or preventive maintenance. The corrective methods remove the reported faults ended in error(s), while protective approaches recognize and remove faults before they cause an error [3].

2.2.4 Fault Forecasting

Fault forecasting is a method to evaluate the system behavior against the fault occurrence or activation. It can be accomplished in two different techniques: quantitative (formal) evaluation and qualitative (experimental) one. For quantitative method, the whole system is modeled by probabilistic equations. In this approach the malfunctioning probability of the modeled components are reported as measures of dependability. This method is fast but complex especially as the size of system grows, but the system is simulated and its behavior is evaluated in the presence of faults by running

large number of experiments. This method is not as fast as formal methods, but it has more controllability over more parts of the system. Fault injection technique is a typical example of this method. Fault injection can also take place in the verification stage of the fault removal method.

All of the four dependability means are somehow related to each other and in some cases they have some overlaps. However, a typical question is that which of them guarantees to deliver a dependable system. In most cases, the expected dependability is not achieved by using any of them, separately. Different fault-tolerant techniques have been introduced by researchers, but it would be more cost effective to employ them if the resiliency of the target device against different sources of fault is available. This knowledge can be achieved by applying fault forecasting methods in order to address the vulnerable components of the system. With this strategy the expected dependability is met while redundancies cost is kept low. In practice, fault removal and fault forecasting methods are followed by each other. In other words, after rejecting a system by the fault forecasting, several fault removal tests are applied. These new tests provide actions that can help the designer to improve the system. This loop is repeated until reaching the desirable design. The main goal of this dissertation is to present a fault forecasting technique for 3D NoC environment. To reach this goal fault injection technique is used which is briefly in the following Section.

2.3 Fault Injection

Fault injection is a process in which responses of the system are analyzed to verify the behavior of the designed system in the presence of predefined faults. This is a popular approach in reliability evaluation of computing and communication systems such as processors, micro-controllers, and interconnections [31, 27, 116]. In this method faults

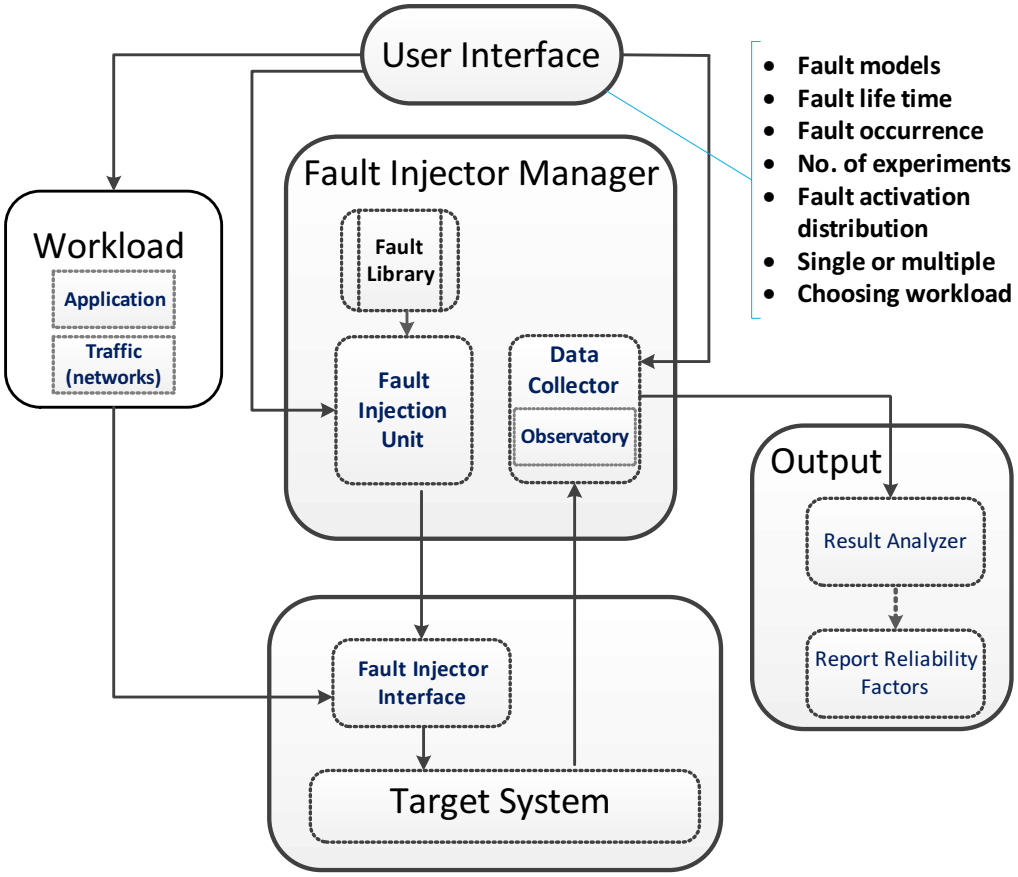


Figure 2.3: A typical fault injector components

are injected into the system states and points, that were previously determined by an initial system analysis. There are two major types of fault injection: hardware-based and simulation-based [8]. In the hardware-based method the actual component is examined by triggering some faults through the system and determining their effects. This method is expensive and usually is considered at the final stage of mass production. Moreover, the model of system is evaluated in the simulation-based approach to study the behavior of it against the fault models. This method is much cheaper than hardware-based evaluation and it is considered at the beginning stages of design, but the simulation-based method takes longer time than the hardware-based one. In addition, the other concern of simulation-based method is the accuracy of the model of the target system, which can be verified by running different kinds of applications.

Fault injection is often the only choice when the complexity of a target system precludes analytical solutions for reliability analysis. It facilitates fault removal and fault forecasting methods, resulting in the following benefits:

- Understanding the effects of real faults on the system either internal or external ones. A fault-tolerant designer is able to focus on the most vulnerable components rather devising a general tolerant approach. This capability results in reliable coarse-grained components designs and consequently more cost-efficient fault-tolerant systems.
- Efficiency assessment of fault-tolerant mechanisms embedded in the target system. This method is also able to compare the resiliency of proposed fault-tolerant method in the presence of different sorts of faults.
- Forecasting the faulty behavior of the target system, in particular developing a measurement of the coverage provided by the fault-tolerant mechanisms.
- Estimating the failure coverage and latency of fault-tolerant mechanism.

Several efforts have been made to develop techniques for injecting faults into a system prototype or model as introduced in Chapter 1. Figure 2.3, represents the typical modules in a fault injection tool with their interactions which is composed of five main components: user interface, workload, fault injector manager, target system, and output.

The user interact with the fault injector tool by the help of the **user interface** unit. Some examples of definitions which should be defined by the user are shown in Figure 2.3, such as type, number, and activation time of faults.

The **workload** component is composed of benchmarks which are supposed to be run on the target system while examining its behavior against injected fault models. The

workload component may include either of synthetic or real applications or even both of them. In some cases an interface between this component and target system is needed to translate the generated raw data by the applications for the target system.

The **fault injector manager** is the heart of fault injector tool which is responsible for activating the fault models which are predefined by the tool designer and accessible by the fault injection unit. In other words, fault libraries are some fault models which will be injected to different components of the target system based on the user preference. The fault injection sub-component is responsible to activate each of the injected faults into the system. The injected faults are the dormant faults and once they are triggered by the fault injector tool, they will be treated as active faults. The observatory list is another sub-component of the fault injector manager in which a list of comparison signals are stored. The fault injector tools compare the values of the observatory list signals after running each faulty run by golden run. A faulty run is the execution of the workload once at least one of the injected faults is activated, while the golden run refers to the execution of the workload over the target system when all of the injected faults are in their dormant state.

The fourth major component is the **target system**, which is actually the model of system that should be studied. The fault injector interface verifies the injected fault models are syntax error free in the language in which target system is modeled. In case of evaluating the proposed dependability means such as fault-tolerant or fault removal methods, this component should be integrated with the proposed technique.

Finally, the result of golden run and fault run comparison is reported by the **output** component, which are considered as an evaluation metrics to compare the reliability of the target system in presence of different sorts of faults.

Chapter 3

The 3D NoC Architecture

As discussed in Chapter 1, NoC is the dominant communication infrastructure for many-core systems with thousands of cores. The NoC offers higher flexibility and modularity, supporting simpler interconnection models with higher bandwidth as compared to traditional SoC approaches. In addition, the advent of 3D integration results in considerably shorter interconnects wires in the vertical direction. The 3D integration supports new opportunities by providing feasible and cost effective approaches for developing heterogeneous cores to realize future computing systems. Additionally, the 3D structure supports heterogeneous stacking because different types of components can be fabricated separately, and silicon layers can be implemented with different technologies. Furthermore, vertical integration technologies are particularly compatible with the traditional CMOS IC design process; they can be easily integrated into the state of art designs. These capabilities are inherent to 3D NoCs, and have been introduced as a promising scalable and energy-efficient communication method for modern ICs with high performance [25, 90, 18, 78].

In general, a 3D NoC is composed of three major components including: links, routers,

and Network Interface (NI). The **links** physically connect all the cores to perform the communication process. There are two different types of links in 3D NoC which are: inter-die connections and intra-die connections. The former one connects routers in different layers, while the latter one relates to communication links among NoC routers located in the same layer. The communication links inside the NoC router also belong to intra-die interconnections category. The **router** is responsible to run the communication protocols in order to accomplish the data transmission process among the connected cores. In other words, the router receives data packets from the shared links and based on their destination address, forwards them to the local connected core or to the corresponding neighbor router. In the latter case, the router is called an intermediate router. The routers' protocols should support deadlock and livelock free routing algorithms, fair data transmission by applying arbitration unit, and performing an efficient switching methods. Finally, the **NI** is an interface between IP cores and network, since the processed data format of each IP does not necessarily match the data transmission format and network protocols. Before discussing the architecture of the 3D NoC which is developed as a part of this dissertation, some basic definitions are discussed first in the following subsections.

3.1 Basic Definitions

This section covers brief definitions for some of the well-known terminologies in the NoC environment, including: flit and phit, flow control, routing algorithm, and topology [20]. These terminologies are refereed to in the rest of this dissertation.

3.1.1 Flit and Physical Unit (phit)

An injected message into the network by the local connected IP, is first segmented into packets by the NI component, which are then divided into fixed-length flits. Flits are the unit of data transmission flow control in the most popular NoC routers. In other words, a flit is defined at the link level representing atomic units that form packets. There are typically three types of flits: header flits, data flits, and tail flits. Header flits contain the routing information for the whole packet and show the presence of a new packet. Data flits include the actual data bits which are supposed to be transferred among the cores. Finally, tail flits show the end of packets. Routers release the dedicated resources once the tail flit reception is confirmed by the local IP or neighbor router that the packets are forwarded to.

But, for some designs a smaller unit of data transmission is applied which is called a phit. In general, a flit may be composed of several phits in applications with highly serialized links [20]. But, in many research articles, a flit size is the same as a phit size, which is the smallest unit of data transmission. In these cases, the characteristics of different components in an NoC architecture such as the buffer size and intra-die links' bandwidth, are reported in terms of flits. In addition, flits are also used as the performance evaluation metric units such as the number of transmitted flits per second or the power consumption for each flit transmission. In the rest of this dissertation the unit of data transmission is assumed to be a flit (or it is assumed a flit is composed of a single phit).

3.1.2 Flow Control

Flow control is the mechanism which decides network resource allocation, such as channel bandwidth and buffer capacity for each packet. A good flow control proto-

col minimizes the average of packet transmission latency. In other words, a typical flow control protocol improves the network throughput rather than the throughput of individual routers. There is a tradeoff between the complexity of flow control implementation and its efficiency. One of the critical factors affecting the efficiency of flow control is to choose a suitable switching technique. Circuit switching, store and forward, cut-through, and wormhole are the popular switching approaches for NoC designs.

In circuit switching technique, the whole path is reserved from the source toward destination for the entire duration of transmission. In this method all the intermediate reserved links will be released once the message has arrived at the destination router and the corresponding acknowledgment is confirmed by the source router. The advantage of this method is that the intermediate routers do not need any buffer unit. On the other hand, this method has a set-up phase which is not efficient especially for short messages. In addition, it violates the benefits of bandwidth sharing; resulting in poor bandwidth utilization.

The other approach is packet-based flow control which has been proposed due to the inefficiency of circuit switching method for NoC applications. The store and forward and cut-through switching methods are some examples of packet-based flow control technique. In packet-based method the message is divided into smaller units called packets; packets are partitioned into smaller units of transmission (flits). In this technique the links are not reserved and each packet can choose a different path to reach the destination. So all the flits of a packet use the same path toward destination like a train, consecutively. Unlike to circuit switching method, the buffer component is needed to store in-flight packets. The only difference between store and forward technique and cut-through switch method is that in the former the whole packet should be received by an intermediate router before being forwarded to the Local

IP core or the appropriate neighboring NoC router. This condition is not necessary for transmission of packets with cut-through switching method. Finally, in wormhole switching, which is the most popular one, the flits are allowed to be forwarded before the entire packet is received by the intermediate routers similar to the cut-through method. But, the required buffer size in this method is less than what is needed for the cut-through approach. In other words, the buffer size in the wormhole switching is always less than a packet size unlike the cut-through method [80].

3.1.3 Routing Algorithm

The routing algorithm includes the process of choosing the appropriate path for a message through the network toward its destination. A good routing algorithm should balance the traffic evenly all over the network in order to prevent congestion through the network and consequently throughput degradation. Also, it should be implemented with as low as possible complexity to not stretch the critical path delay or impose area overhead. The routing algorithms are classified according to different criteria. Depending on how a path is defined, routing algorithms can be classified as deterministic or adaptive methods. The complexity of deterministic algorithm is lower than the adaptive one. In this method, a predefined algorithm is run once the header flit of a packet is received at each node. This method does not consider the status of the network in choosing the path for incoming flits. The adaptive algorithm may choose different paths for individual flits of a packet. This decision is done by keeping track of the network status. The complexity of adaptive routing algorithm implementation increases as more information of the network is needed for routing decision. Furthermore, an adaptive routing algorithm can be considered as a fault-tolerant method in case there is a fault on a packet's path to the destination.

There are two major types of routing algorithms: source routing and distributed routing. In the source routing the whole path for a packet transmission through all the intermediate routers are selected in source router. This routing information should be added to the packet's header flit. On the other hand, for distributed routing algorithm choosing the path for packet transmission is done through the intermediate routers. In the source routing method there is no need for routing decision process through the intermediate nodes, resulting in lower latency and power consumption. The disadvantage of this method is that the size of header flit increases for larger network sizes. The best method to take advantageous of both of these methods is to devise a mixed algorithm which has been done as a part of this research [119] and is discussed thoroughly in Section 3.3.

3.1.4 Topology

Topology is another important factor in designing an NoC, which defines how the routers are connected to each other. This connectivity affects the number of hops for each packet transmission from source toward destination and also the interconnect links; indirectly influencing the power consumption of the whole network. Many different topologies have been proposed inheriting from the traditional semantic network topics, which are evaluated by different metrics such as degree, hop count, maximum channel load, and path diversity [80]. The degree parameter refers to the number of connected links to each router. In some topologies all the routers have the same degrees such as the ring topology, while in some other cases different routers may have different degrees such as the mesh topology. The hop count includes the number of intermediate routers that a packet passes on its way from the source to the destination. The maximum hop count is usually considered as a factor of router and links delay to estimate the maximum latency of flit transmission through the network. But,

it may not be an accurate metric in case the network congestion occurs. Maximum channel load factor predicts the maximum bandwidth that a router can support in terms of bits per seconds (bps). The path diversity is the other metric which defines the number of shortest path between a given source and destination pair. The path diversity metrics shows the capability of the topology in supporting load-balancing and fault-tolerant mechanisms.

In general, a network topology is divided into two major categories including: a direct topology and an indirect topology. For direct topology, each IP core is connected to a router. For this type of topology, each router is a source, destination, or an intermediate one. Ring, mesh, tori, and torus are some of the well-known examples for direct topologies. In an indirect topology, routers and IP cores are separated from each other. All the IP cores are source and destination nodes, while routers are all the intermediate nodes. The butterflies, Clos networks, and fat free topologies are some examples of the indirect ones; the direct topologies are more common in on-chip designs because of their ease of fabrication process.

3.2 The 3D NoC Router Architecture

The structure of the 3D NoC architecture is discussed in this subsection. This architecture is designed and developed in Verilog HDL environment, called THENoC, as an experimental platform for this dissertation. It supports wormhole switching, mesh and torus topologies, and employs XYZ routing algorithm in which each router is composed of 7 bidirectional ports. But, it is implemented completely modular and consequentially capable of supporting other specifications with nominal modifications. THENoC is actually a synchronous simulator with the capability of profiling all the communication links during the packet transmission among connected local ports.

The asynchronous design of the similar architecture for 2D NoC has been also developed, which is embedded with two synchronous to asynchronous and asynchronous to synchronous wrappers [118]. The synchronous and asynchronous 2D NoC architectures are compared in terms of their power consumption [114] and reliability [115, 120] characteristics.

All the modules of THENoC are also developed synthesizable, for the goal of reporting the real power consumption and critical path delay. THENoC is also able to report different performance metrics to compare the efficiency of any proposed energy-efficient, fault-tolerant, or high performance techniques. It is composed of five main components including the links, an NI, an input buffer, a routing management unit, and a crossbar switch [119] which are discussed individually in the following subsections.

3.2.1 Links

A 3D IC is a chip with multiple tiers of thinned-active 2D ICs that are stacked, bonded, and electrically connected together. As discussed earlier two major types of links in 3D NoC are: inter-die and intra-die links. The inter-die connections are also known as 2D wires, since they connect routers of the same layer together. The inter-die connections inside the router are depicted in detail in following subsections.

On the other hand, there are various vertical interconnect technologies. Wire-bonding and flip-chip stacking have made their ways into mainstream semiconductor manufacturing in recent years, but they are not considered anymore for new generation of 3D integrations [39]. TSVs are currently the most popular vertical-electrical connection forms through silicon or oxide layers [39, 49] since they provide shorter signal paths with superior electrical characteristics in terms of reduced resistive, inductive and ca-

pacitive components. Additionally, the TSV technology provides more I/O terminal ports comparing to tradition system in package integration methods. In the wire-bonding for example the I/O ports should be located on the edges of the chip. With this capability of supporting I/O ports in arbitrary locations of the TSV-based 3D chips, they support more number of interconnections with higher levels of flexibility of circuit layout [127].

A TSV is typically a cylinder composed of the uniform conducting material more often the copper. The TSV is surrounded by silicon dioxide as an insulator to prevent leakage and resistive coupling through the silicon substrate. The TSV interconnections are fabricated in a silicon substrate and their distribution mainly depends on partitioning and placement approaches. However, irregular TSV placement results in a critical fabrication issue which is known as chip warpage. According to the prediction of ITRS, the maximum height of TSV is about $200\mu m$ or less which is the same as thickness of the Si chip. The diameter of vias in TSV is now $20\mu m$ and may reach $5\mu m$ in the future [109]. A copper TSV in standard Si-bulk technology normally will have via diameter of $2\mu m$-$8\mu m$ by 2018, $5\mu m$ by $5\mu m$ contact pads, $4\mu m$-$16\mu m$ via pitch, $0.5\mu m$ oxide thickness (t_{ox}), and $20\mu m$-$50\mu m$ layer thickness including substrate and metallization [45].

A TSV-based 3D NoC structure is illustrated in Figure 3.1, in which TSVs are connected to each other with Microbumps as discussed in Chapter 1. This figure is a sample 3D design in which each layer has different types of processing elements, while other designs can be different. It also shows a top view of a TSV which is typically composed of three different layers: depletion, insulator, and conductor (TSV body).

It is critical to note that the TSV fabrication process is independent of the CMOS fabrication technology and TSVs do not scale down with the same pace. TSV diameters and pitches are two to three orders larger than transistor gate lengths. Additionally,

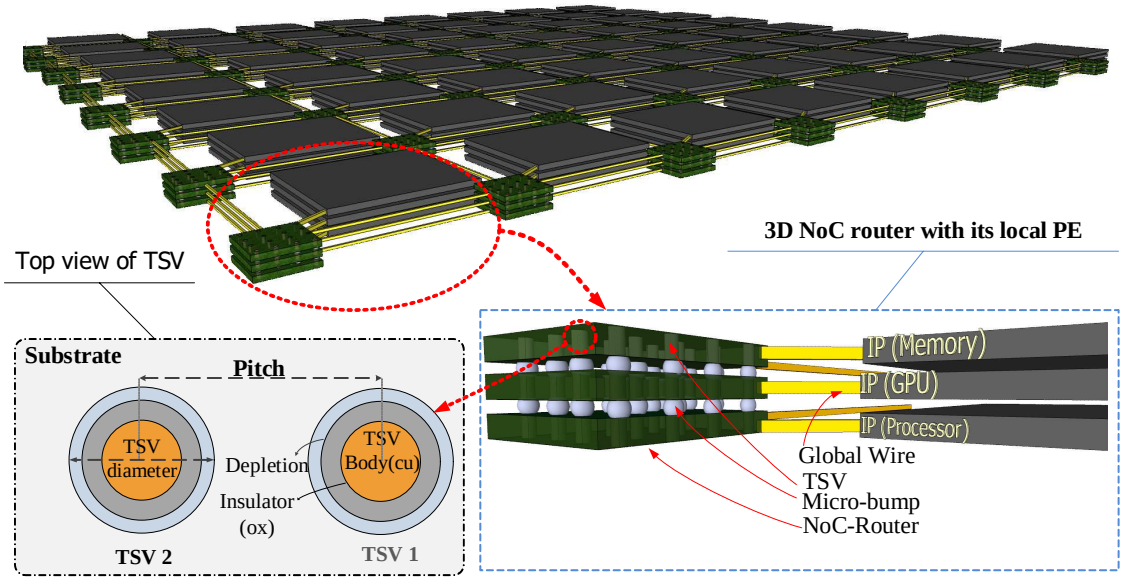


Figure 3.1: TSV-based 3D NoC and TSV top view

TSVs have different characteristics other than 2D wires, resulting in new challenging designs which are discussed in Chapter 4.

3.2.2 NI

The NI is responsible for interpreting generated data by the connected PE to each of the NoC routers though the local ports. The functionality of this component has been verified by injecting both realistic and synthetic data traffics. One of the major tasks of the NI is to prepare the transmission units of data, which are flits in this dissertation, by defining some special protocols. These protocols are defined based on the 3D NoC router simulator implementation parameters and flow control attributes such as buffer size, flit size, and switching method. Figure 3.2 illustrates the functionality of NI component through an example in which the packet is composed of 32 bits and the size of each flit is 8 bits plus the ID bits. The ID bits are used by the router components to recognize the type of flits while transferring through the network. The NI also generates both header and tail flits, attaching them to the

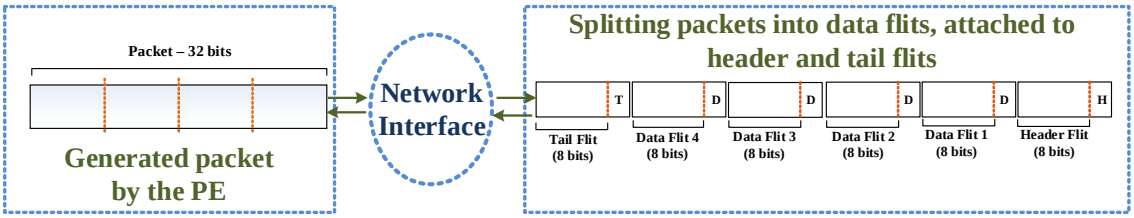


Figure 3.2: Example of NI functionality

front and end of each packet, respectively. In this dissertation, the NI component of THENoC simulator splits all the messages into 64-flit packets, while each flit is composed of 32 bits. The generated header flit in this component also supports the XYZ routing algorithm which is the default routing method in THENoC. The reverse process is done again in the NI component of each router after capturing the incoming packets through the local port. In this step, the incoming data flits are rearranged into a readable message format for the connected PE to the local port.

The major components of router architecture for THENoC including: the input buffer, routing management, and crossbar switch components, are discussed in the following subsections. Figure 3.3 illustrates these components and their connections (inter-die links).

3.2.3 Buffer

The buffer component is supposed to store the header, data, or tail flits on the intermediate routers in packet-based switching methods as discussed in Section 3.1.2. THENoC employs the input buffers rather than output buffer as they are more energy-efficient [103]. Furthermore, the size of buffers are less than the packet size in THENoC to support wormhole switching in which the data flits follow the header flit as a chain of flits. In THENoC the input buffers are supposed to store the incoming flits until they are granted with the appropriate output port. These output ports

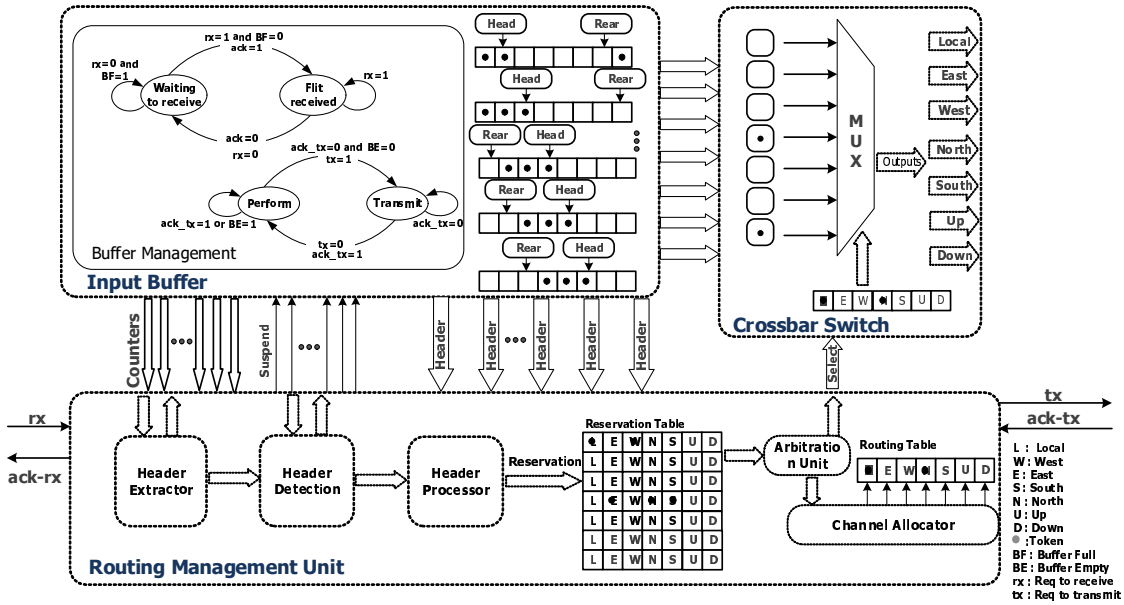


Figure 3.3: 3D NoC router architecture

are reserved until the whole flits of the packets reach the destination. Once the tail flits leave any intermediate router the corresponding input buffer port is deallocated and is available to service other requests. The buffer management is composed of two state machines which are designed to handle the read/write process from/to the buffer unit. In other words, this subcomponent is in charge of tracking the empty and full spaces inside the buffer in order to preclude replacing the old flits with the new ones, before they are transmitted. The buffer management grants the incoming flits if there is a free space in the buffer unit, otherwise the requested flit waits in the intermediate router's buffer until to get the grant.

In a normal buffer implementation, there is a chance of suspending the buffer component's grants though there is a free space inside the corresponding buffer component. But the input buffer component of THENoC is implemented as a circular queue to address this issue. With this implementation, it is guaranteed that no port request of a flit is stalled if the buffer has an empty space. For this implementation two pointers are needed to keep track of occupied locations in the buffer component as shown

in Figure 3.3. In THENoC implementation, the head pointer refers to the head of queue, including the occupied cells of the buffer, and rear pointer refers to the empty location right before the last occupied cell of the same queue. In this approach one cell of the buffer is always reserved which cannot be used for storage usage.

3.2.4 Routing Management Unit

The routing management unit component is the central unit of the router architecture which is composed of four subcomponents including: a header detector, a header extractor, a header processor, and an arbitration unit in addition to a look-up table. All of these units and their interactions are represented in Figure 3.3. An XYZ routing algorithm is implemented in THENoC which can be replaced by other routing algorithms with modifying the routing unit module of THENoC.

The header detection unit distinguishes the header flit from data and tail flits. In THENoC, the buffer component has a counter which counts the total number of received flits for each packet and it resets once the tail flit is read from the input buffer and transferred to the appropriate output port. The header detector component monitors this counter to recognize the header flit as shown in Figure 3.3. The header extractor splits the header flits into different fields and prepares it for header processor component which is responsible to run the routing algorithm. Each of the input channels can reserve one of the output ports when the routing process is accomplished. An arbitration unit locks the dedicated output channel until the end of packet transmission. It also guarantees a fair output channel allocation among input ports. In more details, in case of receiving a request from the new header flit for an already allocated output port channel, the new header flit is stalled and its request is tagged in a priority queue. This priority queue is checked before the upcoming output

port channel allocation to the header flit requests. This priority queue in Figure 3.3 is shown as the Reservation Table.

The routing unit grants the requested input port and updates the selector signals of the output multiplexers once the routing process is accomplished successfully. Such grant and activation signals are disabled as the tail flit of a packet is transmitted to its desired output port.

3.2.5 Crossbar Switch

The crossbar switch is responsible to connect the active input buffer to their appropriate output ports. As shown in Figure 3.3, the crossbar switch is composed of multiplexers for each of the output ports. The crossbar switch component actually establishes the connection of input buffer to selected output port by reading the routing control signals which are provided by the routing management component. As discussed earlier THENoC implements the wormhole switching, in which all of the flits which follow the header flit, are directed toward destination in a pipeline manner. In other words, the data and tail flits are blocked if their header is impeded on its way towards destination. In this case these flits (data and tail flits) are kept in the intermediate buffers until the header flit is granted the channel resources to start moving toward destination. Once a packet transmission has been confirmed by each intermediate or destination routers, the corresponding switch is unlocked to serve other input channels.

As an additional aspect of this dissertation, an efficient routing algorithm has been proposed which takes the advantages of both source and distributed routing methods, called TagNoC [119]. The TagNoC algorithm and design are discussed in the following section.

3.3 Proposed TagNoC Routing Method

TagNoC is actually a hybrid routing algorithm which takes the advantages of both source routing and distributed routing approaches. In other words, the goal of the proposed TagNoC method is to reduce the intrinsic overhead of conventional source routing and per hop routing decision process in distributed routing methods [119]. Additionally, TagNoC is actually a micro-architectural level method which can be integrated to any deterministic routing algorithms. It inherits the routing-specific characteristics of the target-oriented routing algorithm and supports all turn model routing algorithms such as XY, NF, WF and NL. Since XY routing algorithm is the reference model for most of the NoC related research, in this project XY-based TagNoC coding and corresponding architecture are presented. But, the presented architecture can be simply modified to support any other turn-based routing model. In this approach a header flit is composed of the destination router's coordination in addition to, two extra bits which are called Tag bits. The header flit including the Tag bits values are generated in the NI of the source router. In XY-based TagNoC method, the header flits is first directed horizontally and then vertically toward its destination similar to XY routing. With this methodology there is only one turn for the header flit on its way toward the destination. With this knowledge the source router is able to predict the turn point of the header flit. In this method for each header flit at the turn point, there are three options: turn up, down, or choosing the local port, if the source and destination have the same vertical coordination. These three options are encoded for each header flit and presented by Tag bits as a part of header flit as shown in Table 3.1.

With this technique in all of the routers the destination address is compared with the corresponding coordination of the intermediate routers by employing XNOR gates. The architecture of TagNoC routing approach is presented in Figure 3.4, in which the

Table 3.1: Horizontal direction

Z_x	**Tag[1]**	**Tag[0]**	**Rotation**
0	×	×	no turn
1	0	1	turn up
1	1	0	turn down
1	1	1	local

Table 3.2: Vertical direction

Z_y	**Tag[1]**	**Tag[0]**	**Rotation**
0	×	×	180°
1	×	×	0°

output of XNOR gates for both horizontal and vertical dimensions are shown by Z_x and Z_y, respectively. If Z_x is equal to 1 it means the header flit does not have any further horizontal movement toward destination or it has reached its turn point. The Tag bits are evaluated at this point to choose the next movement of the header flit as shown in the fourth column of the Table 3.1. After a header flit has turned the Tag bits are not evaluated anymore as shown in Table 3.2, since the header flit has only two options at this point which can be decided by the Z_y signal.

The proposed TagNoC method is compared with other routing approaches in terms of scalability, performance, and power consumption. These four routing algorithms are: baseline (XY technique), None Encoded Address (NEA), Encoded Address (EA), and Optimized Encoded Address (OEA) routing algorithms [119]. For the **NEA** method, the header flit contains the coordinates of all intermediate routers from source toward its destination. The elimination of routing decision through the intermediate routers results in less latency and power consumption, at the cost of increasing header flit size. But, NEA is not scalable and imposes a large data overhead. In the **EA** technique, only two bits are needed to identify the output port for each of the intermediate routers in a 5-port 2D NoC assuming the packet from any of the incoming ports is not directed to the same port. To encode four of output ports dynamically, the output ports are flagged by a turning degree value. Based on the proposed definition, the local port is always labeled by 0°. The other output ports are labeled as 90°, 180°, and 270° values in a counter clock wise direction starting from the incoming port as illustrated in Figure 3.5. In Figure 3.5a. it is assumed that incoming port is the

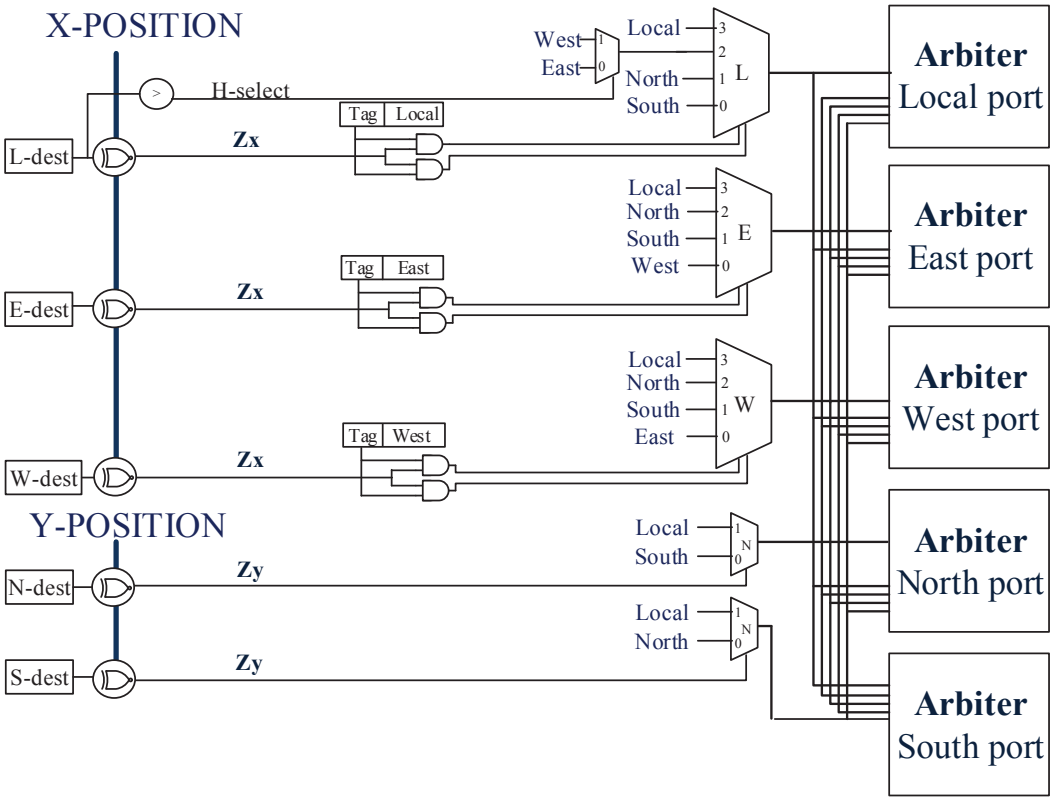
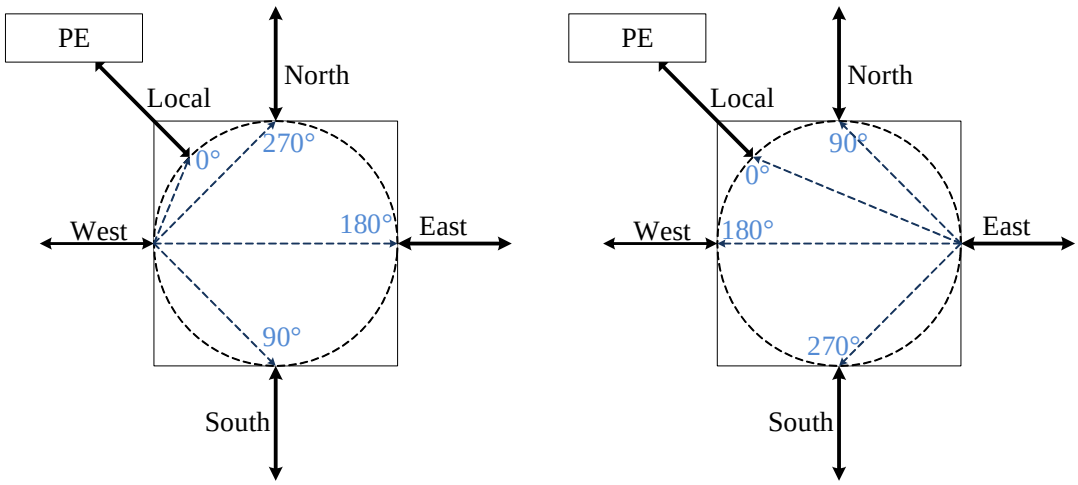


Figure 3.4: TagNoC architecture for a 5-port NoC router

west port, while for the incoming header flit from the east port (see Figure 3.5b), the north, west, and south output ports will be recognized by 90°, 180°, and 270° labels, respectively. With this dynamic labeling output ports, relative to the incoming port, there is no need to use an extra bit for storing the appropriate output port for the EA technique.

The idea of **OEA** approach is to use the same encoding technique as EA method until the header flit changes its dimension toward the destination. Then the header flit has only two options, choosing the local port or traversing in the same direction. For the OEA routing techniques one bit per hop is employed to navigate the header toward destination, after the header turns to move vertically. This optimization is because of the property of the dimension order routing algorithms in which only one



(a) Incoming port is west (b) Incoming port is east

Figure 3.5: Port forwarding directions

turn is permitted. The comparison of these four approaches with the TagNoC method is discussed in the following subsections.

3.3.1 Scalability

Routing algorithm plays a major role in the degree of scalability for an NoC design. The number of enforced bits in the header flit by a routing algorithm is the limiting factor. The size of header flit influences the flit size which directly impacts the bandwidth of the network. If the size of a flit is large, the limiting factor in wire routing or bandwidth efficiency parameters is not satisfied.

Table 3.3: Header size comparison of proposed methods

Approach	**Header size**	**Header bits**
Baseline	$(\lceil \log_2 X \rceil + \lceil \log_2 Y \rceil)$	8 bits
NEA	$(\lceil \log_2 X \rceil + \lceil \log_2 Y \rceil) \times ND$	144 bits
EA	$2 \times ND$	36 bits
OEA	$2 \times (X-1) + (Y-1)$	27 bits
TagNoC	$(\lceil \log_2 X \rceil + \lceil \log_2 Y \rceil) + 2$	10 bits

Another critical factor for the NoC is communication reliability. Any fault occurrence

on the header flit may result in a misrouting [29] of the whole packet. The probability of fault occurrence on header flit grows as the header flit size increases, with the same Bit Error Rate (BER)[1] for both data and control flits. The extra routing information bits in the header flit which limits the scalability and decrease the reliability of NoC is the major problem of source routing methods.

The minimum number of bits in a header flit for all discussed approaches are presented in Table 3.3 where X and Y represent the horizontal and vertical size of the network, respectively. The third column of the Table 3.3 demonstrates the header size for each of discussed routing approaches in a 10×10 NoC mesh. *ND* term represents network diameter which is $(X + Y - 2)$ for a mesh topology.

For the EA method, two bits are needed for each intermediate node from the source to destination. Comparing to NEA, EA method is more scalable but still is a function of network dimensions. In comparison with EA, the header flit in the OEA method requires even fewer bits. The reason is that in OEA method, two bits are needed for each horizontal movement while after the turn; only one bit is enough for each vertical movements. In TagNoC, regardless of NoC dimensions, only two extra Tag bits are used so the number of header flit bits for TagNoC technique is considerably smaller than the rest of proposed approaches, while it is almost with the same order of baseline XY routing algorithm. The presented function in Table 3.3 are visually compared for different range of network sizes in Figure 3.6. In this figure, it is demonstrated that the size of header flit in TagNoC grows logarithmically, similar to the baseline distributed XY routing with negligible constant amount of overhead.

The performance, power consumption, area overhead, and the frequency of TagNoC

[1] The Packet Error Rate (PER) is the number of received data packets with errors divided by the total number of received packets. A packet is faulty, if at least one bit is erroneous. This assessment depends on the Bit Error Rate (BER) as a standard measure of the performance of a channel in the presence of fault.

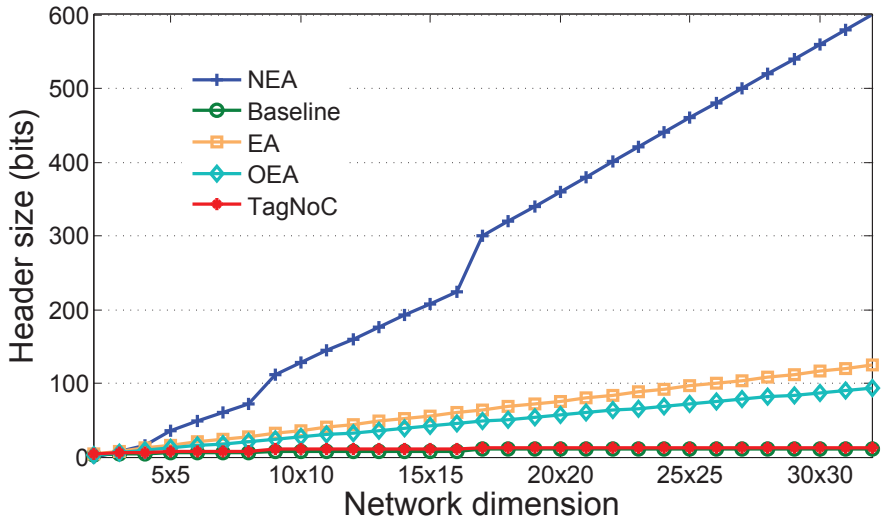


Figure 3.6: Routing overhead bits in the header flit vs. number of nodes

method is compared with baseline, EA, and OEA approaches in the following paragraphs. The NEA method is not considered for these evaluation metrics because of its large design complexity and overhead.

3.3.2 Performance

In this section, the experimental framework is introduced first and then experimental results are reported to compare the efficiency of all proposed methods.

Experimental Setup

A 16-node, 4×4 mesh network with 16-bit interconnection links is implemented in Verilog as an NoC simulator for this experiment. The accuracy of the NoC model is verified by tracing the traverse of packets through the network with the help of SystemVerilog language for different traffic patterns. Both synthetic and application-level traffics analyses are considered in order to investigate the efficiency of the proposed hybrid routing approach under different performance constraints.

Routers contain five 128-bit buffers per input port and operate with the frequency of 1 GHz. Selected signals for the simulation are profiled after the simulator is warmed up and the majority of routers are involved in packet transmissions as source or intermediate nodes in order to report accurate results. Header flit arrival time and the moment which the whole packet is received by its destination local port are captured in order to measure packet transmission delay. Throughput is defined as the rate at which packets are delivered by the network for a particular traffic pattern [22]. Each network is designed to operate at its maximum frequency with the assumption that processor tiles operate on their own frequency domain asynchronously with respect to the interconnection network, similar to recent industry many-core chips [41].

Zero-load network latency and both synthetic and application-level traffic patterns are used in order to analyze impacts of the proposed router architecture on the NoC. This factor is widely used as a performance metric for traditional interconnection networks which gives a lower bound on the average latency of a packet through an empty network in which a packet never contends with other packets to obtain resources [22]. It does not depend on type of the generated traffic patterns as there is no contention among routers to access the interconnection resources.

The zero-load latency in this experiment, is implemented by the injection rate of 0.1%. The impact of proposed architectures on zero-load latency for different number of nodes under the uniform random traffic pattern is plotted as shown in Figure 3.7 for baseline, EA, and TagNoC methods. All EA, OEA, and TagNoC methods require the same number of clock cycles with different clock cycle time since they have different routing unit architectures. Figure 3.7 shows that as the network size increases, the latency for all methods increases. The performance of source routing methods scales much better than the baseline architecture for a fixed network size. In a 100-node, 10×10 mesh network the zero-load latency of TagNoC technique is 26% less than

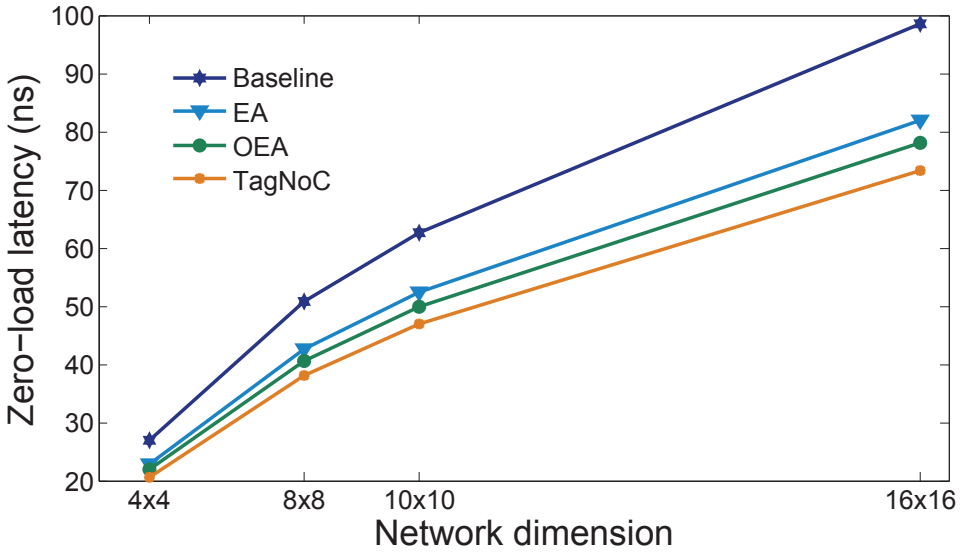
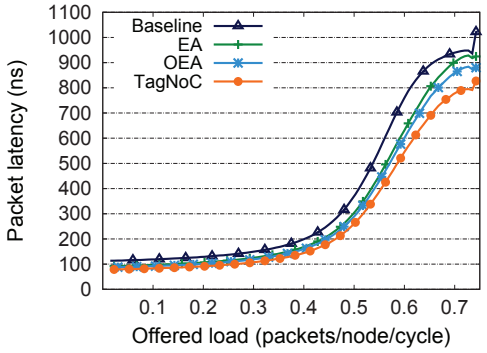


Figure 3.7: Zero-load latency versus different network size

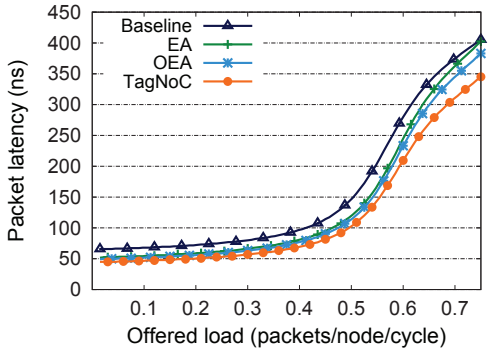
baseline method.

Synthetic Performance

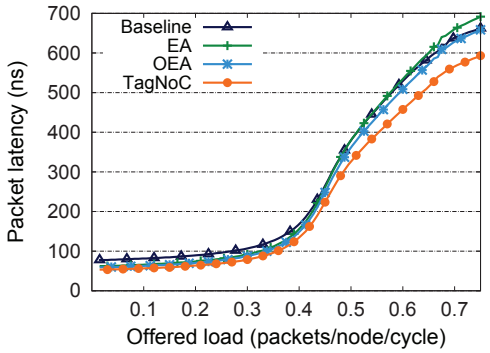
Spatial distribution of messages in interconnection networks are considered using traditional synthetic traffic pattern to evaluate the latency of baseline and proposed routing approaches. These synthetic traffic patterns include Bit_complement, Bit_rotate, Bit_reverse, Neighbor, Shuffle, Tornado, Transpose, and Uniform [5]. Packets with variable data flit sizes are injected to routers in this experiment, for each of traffic patterns as the NoC designs are expected to support multiple applications. Figure 3.8 illustrates the average packet latency versus offered load for all the approaches under running synthetic traffics as a function of aggregate offered load for mesh network of 16 nodes. Latency and offered load are reported in units of nanoseconds and packet/cycle/node, respectively. According to the experimental results, the performance of the EA, OEA, and TagNoC routing algorithms are better than the baseline method for the majority of synthetic traffics.



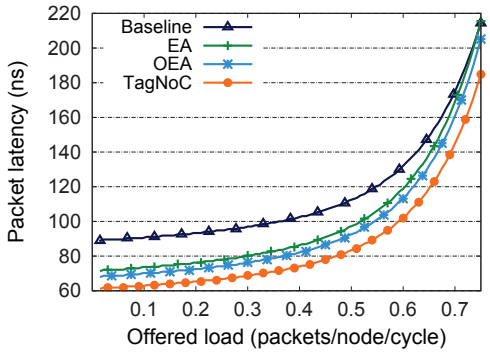
(a) Bit complement



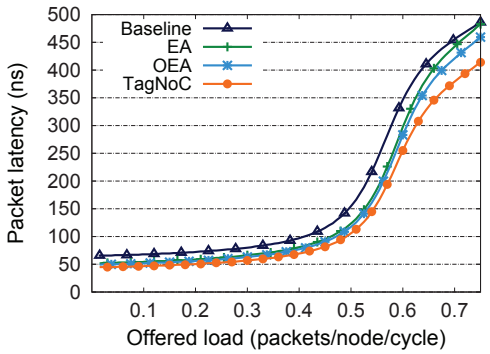
(b) Bit rotate



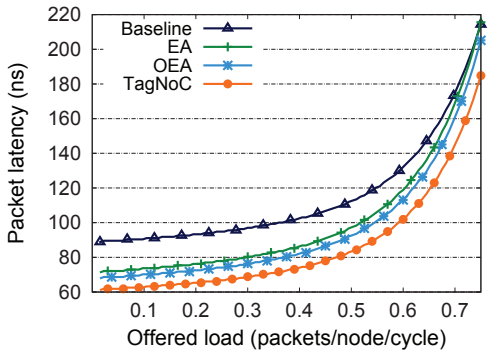
(c) Bit reverse



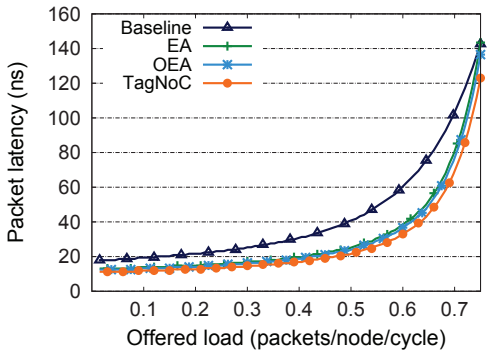
(d) Neighbor



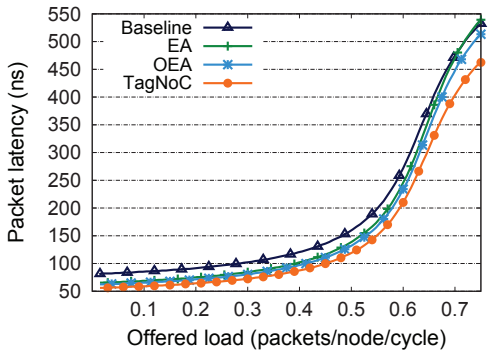
(e) Shuffle



(f) Tornado



(g) Transpose



(h) Uniform

Figure 3.8: Average packet latency under synthetic traffic

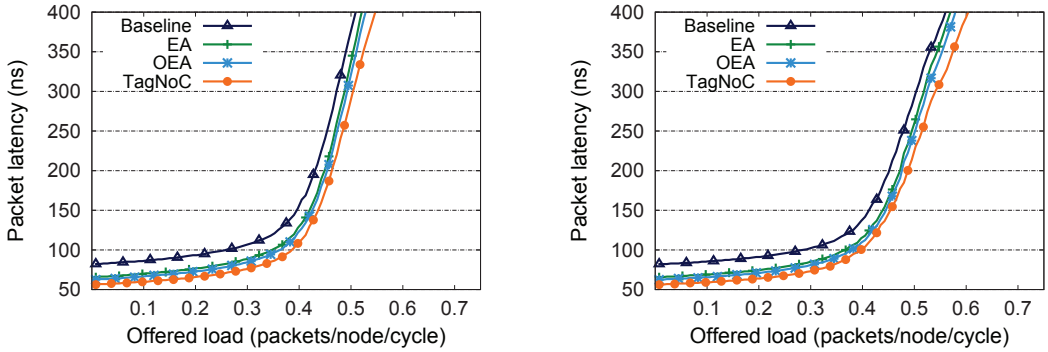
Table 3.4: System configuration parameters

Parameter	Value
Cores	16
Topology	4x4 mesh
Processor	SPARC
L1 Cache I/D	64KB, 2-way, 3-cycle access
L2 Cache	Shared, 6-cycle bank access
Cache Coherence Protocol	MESI
Memory Access Latency	220 cycles
Packet Size	22 bytes
Flit Size	16 bits
Buffer Depth	8, 16-bit entries/port
Switching Scheme	Wormhole
Routing Algorithm	Dimension Ordered Routing

The only exception is the Bit_reverse traffic for higher injection rate in which baseline has better performance than EA method. The turning point of injection rate for the synthetic traffic patterns is between 0.35-0.55 packets/node/cycle (see Figure 3.8). With higher injection rates than the turning point $\approx$ 0.45 packets/node/cycle the average latency of all approaches under running synthetic traffics rise sharply until the network is saturated, while in TagNoC approach the network is saturated at higher injection rate, which provides a better performance for all synthetic traffics.

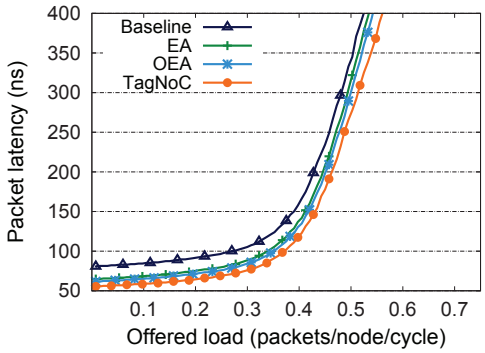
Application Performance

Application traces are obtained from the GEMS simulator [72] using the SPLASH-2 application benchmark suites [113]. Full simulation system parameters are reported in Table 3.4. The average packet latency versus offered load for the baseline, the TagNoC, and the source routing approaches under running SPLASH-2 application traffics as a function of aggregate offered load for mesh network of 16 nodes are plotted in Figure 3.9.

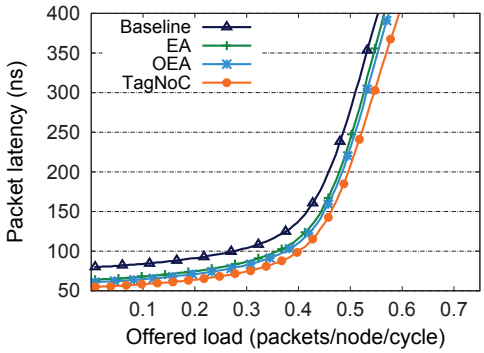


(a) LU

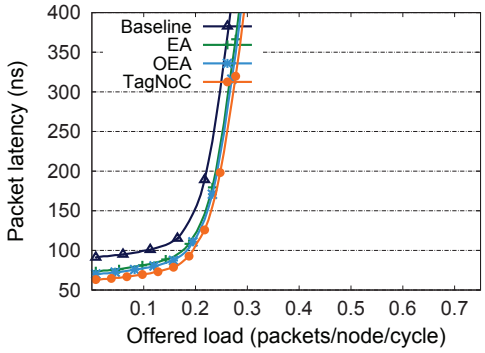
(b) FFT



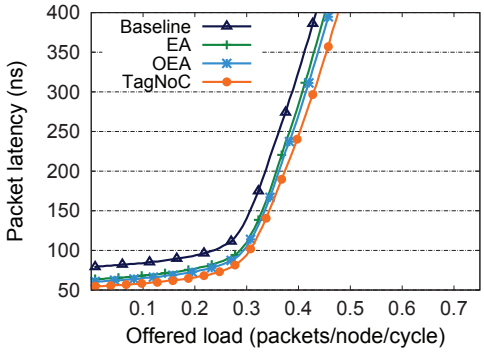
(c) Water nsquared



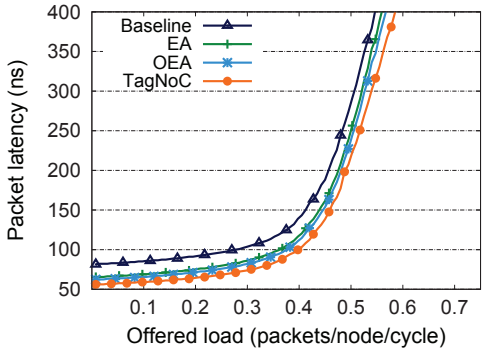
(d) Ocean



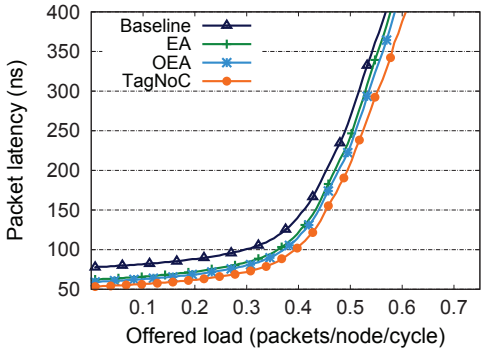
(e) Raytraces



(f) Barnes



(g) Water spatial



(h) Radix

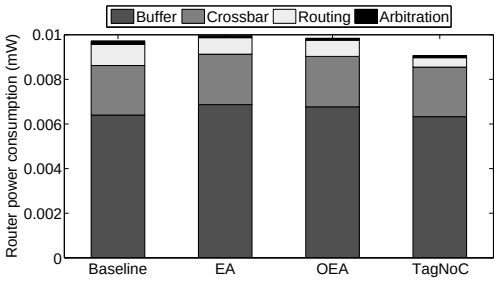
Figure 3.9: Average packet latency under SPLASH-2 application traffic

The reported average packet latency for SPLASH-2 traffics demonstrate similar results of performance reports as reported for synthetic traffics, although the turning point of diagrams for SPLASH-2 application traffics is between the offered load of 0.2-0.4 packets/node/cycle. TagNoC architecture has a better performance for all application traffics (see Figure 3.9). This is because packets are buffered for shorter time through intermediate routers and can be quickly routed; the NoC is able to handle higher injection rates. The other conclusion of this fact is supporting shorter clock cycle time in TagNoC approach as compared to other methods. With this feature the NoC is able to support cores with higher operational frequencies.

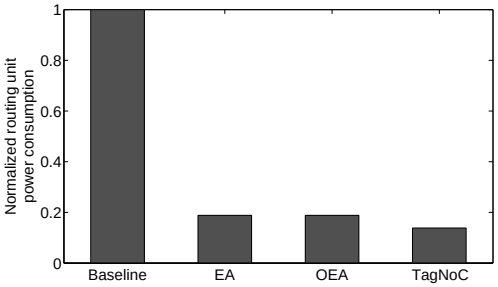
3.3.3 Power Consumption

The power consumption and area overhead all discussed architectures are obtained by a combination of cycle-accurate RTL router simulation, Verilog synthesis, and performing post-synthesis gate-level simulation. The Verilog synthesis is done by Synopsys Design Compiler with TSMC 40 nm standard cell library and post-synthesis gate-level simulation is done by Synopsys PrimeTime in order to extract experimental results.

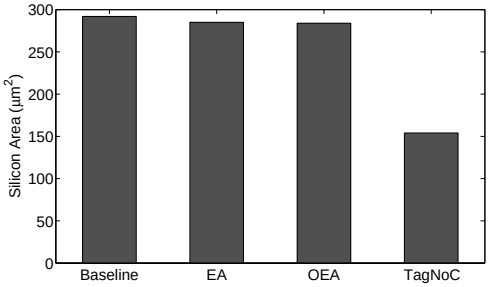
Figure 3.10a shows dynamic power consumption under a 1 GB/s/node *fft* traffic load for a 16-node network with the injection rate of 0.5 packets/node/cycle. NoC with TagNoC router architecture is reported as the lowest power consumer in contrast to other approaches since it eliminates extra comparison circuits. However, total dynamic power consumption of the baseline, the TagNoC, and other source routing techniques do not increase since they all employ the same buffer, crossbar, and arbiter components. Dynamic power consumption of the routing management unit of different approaches are compared separately in order to highlight improvements of



(a) Network dynamic power consumption under 1GB/s/node *fft* traffic



(b) Routing unit power consumption (normalized with respect to the Baseline method)



(c) Routing logic area results (μm^2.)

Figure 3.10: Power consumption and area occupation comparison

TagNoC technique (see Figure 3.10b). The extra power consumption of NI to generate the extra routing information bits is reported in Table 3.5 in terms of micro-watt (μW), which are negligible due to their tiny logic, as expected. It should be noted this extra power consumption is imposed once per packet only during packet injection into the router.

As it is shown in Figure 3.10b EA/OEA methods as source routing algorithms consume considerably less power than the baseline distributed routing. This is due to the

Table 3.5: NI power overhead for extra routing information computation

Power consumption (μW)	**TagNoC**	**EA**	**OEA**
Dynamic power	0.69	9.066	10.86
Leakage power	18.2	210.94	186.9
Internal power	2.42	23.2	20.43
Total	21.37	243.18	218.23

Table 3.6: Router clock periods

Architecture	Clock period
Baseline	0.7466ns
EA	0.8196ns
OEA	0.7801ns
TagNoC	0.7328ns

elimination of routing logic in EA/OEA approaches. But, at the router level it can be observed that EA/OEA methods consume slightly higher power than the baseline. The reason behind this observation is that in EA/OEA methods the number of header bits imposed by the routing algorithm is more than the baseline method which leads to more power consumption. It is shown that TagNoC architecture consumes power approximately 83% and 27% less than the baseline router and EA/OEA architectures.

3.3.4 Frequency Results

Table 3.6 summarizes the minimum clock cycle time for baseline and other methods, extracted using Synopsys Design Compiler. Due to shifting the header bits, EA and OEA routing circuits need higher clock cycle time in comparison with baseline method as shown in Table 3.6. TagNoC is reported as the fastest technique by 2%, 12%, and 6% less clock period comparing to baseline, EA, and OEA techniques.

3.3.5 Area Analysis

Area values of routing logic for all the baseline, EA, OEA, and TagNoC architectures are illustrated in Figure 3.10c. Since they do not impact the buffer, arbiter, and crossbar components, there is no difference between them. Based on the experimental results the area of input buffer, crossbar switch, and arbiter for all the architectures are reported $9471\mu m^2$, $520\mu m^2$, and $1065\mu m^2$, respectively. As it is depicted in

Figure 3.10c, TagNoC routing logic occupies 53% less than the distributed router embedded with baseline routing algorithm. In total, TagNoC router architecture needs $138\mu m^2$ (2%) less silicon area than baseline router due to elimination of conventional routing unit. Overall, the total TagNoC router saves area by elimination of routing logic. The reason that EA and OEA methods seem to occupy approximately the same amount of logic as baseline is because of applying a shift register in their architecture design. They use this shift register to drop some parts of header flits and push the rest to the front once the header flit passes each intermediate router.

Chapter 4

Reliability Challenges in 3D NoC

The need for reliability assessment of 3D NoC architecture has become critical with shrinking CMOS technologies [64]. The anticipated fabrication geometry in 2018 scales down to $8nm$ with projected $0.6v$ supply voltage [45], which results in more transistors on the same die area and tighter noise margin. With theses features higher error rates are expected and consequently the probability of a chip failure is rising. In addition, applying power and performance aware techniques such as power gating, power density, prefetching methods, or even non-ideal threshold and supply voltage scaling exacerbate the failure probability in 3D NoC architectures.

As discussed in Chapter 3, TSV-based 3D design is the most promising solution for 3D integrations of future on-chip interconnections as compared to other radical technologies, but the impact of sub-micron TSVs on future 3D NoCs is still an active area of research [55].

Consequently, the mass production of 3D ICs for consumer electronic products are not reliable enough without a systematic and thorough reliability assessment. Quantitative (analytical) and qualitative (experimental) techniques are well-known fault

forecasting methods, as discussed in Chapter 2. Both of these methods need a comprehensive study on the sources of faults and their effects, which is the focus of this chapter. These features should be addressed in order to develop the authentic fault models and fault metrics as a part of the reliability evaluation of a given 3D NoC architecture. The goal of this chapter is to highlight the reliability challenges of TSV-based 3D NoCs design. More specifically, the potential physical-level faults in 3D NoC, their corresponding logic-level models, and their effects on the functionality of 3D NoC components are covered in this chapter.

4.1 Physical-level Potential Faults in 3D NoC

Based on discussion of Chapter 2, either of internal or external causes results in active faults. At the physical-level, a failure mechanism is the mechanical or chemical action that actually causes the manufactured circuit to perform different than what is expected. Figure 4.1 summarizes the potential physical faults, affecting the performance of 3D NoC design. As shown in this figure, TSV issues, thermal concerns, and SEE impacts are the main physical sources of faults in future 3D NoC designs [109, 100, 85]. It is unlikely that existing technology becomes ubiquitous in near future until there are solutions for these issues [45]. These physical fault sources influence behavior of 3D NoC during packet transmission. Thermal concern has effects on both 2D planar and vertical links, while SEE impacts more transistor of 2D planar designs. The presented categorization in Figure 4.1 enhances the accuracy and efficiency of both fault injection and signal observation steps as major operations of a fault injector tool. It is also useful for providing more realistic formal equations in analytical techniques. Each of these three major physical faults in 3D NoC environment, which are individually divided into smaller subcategories, are discussed in the

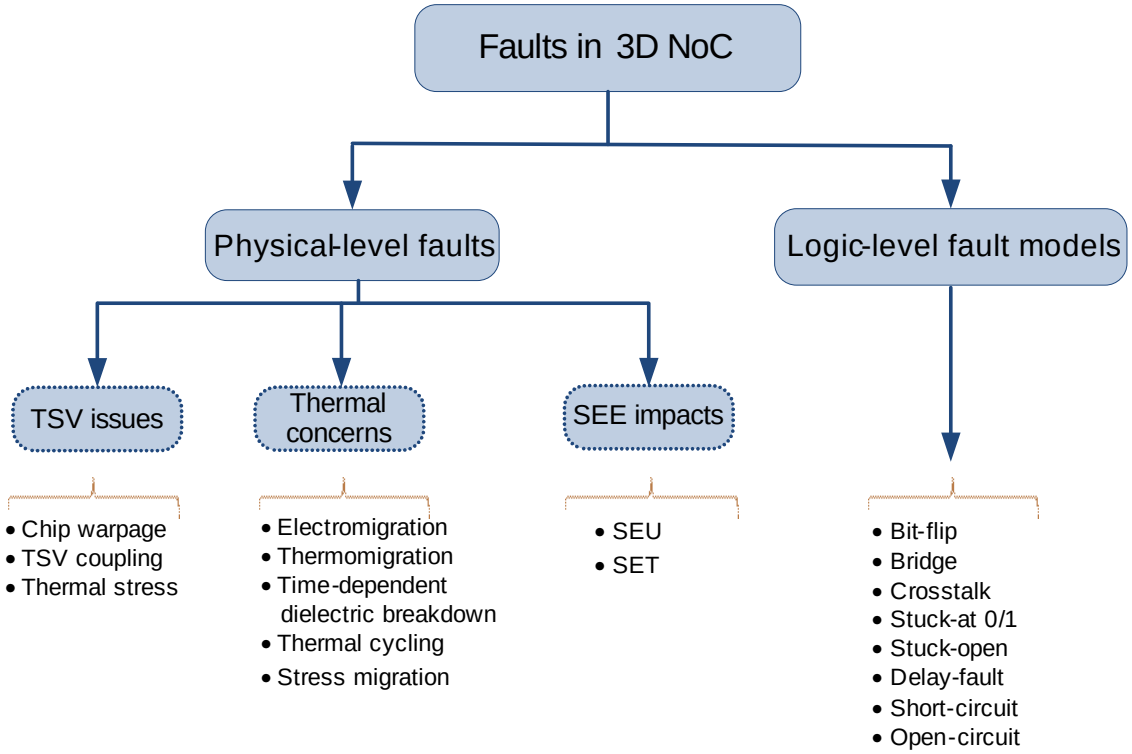


Figure 4.1: Potential physical-level and logic-level faults in 3D NoC

following subsections.

4.1.1 TSV Issues

3D NoCs are expected to offer various benefits such as higher bandwidth, smaller form factor, shorter wire length, lower power, and better performance than the traditional 2D NoCs. These advantages may not be completely achieved because of chip warpage, TSV coupling, and thermal stress as major TSV failure causes [109, 45].

Chip warpage: When TSVs are used for vertical interconnects, it may lead to chip damage, since the TSVs are arranged in a non-uniform manner. The other reason is that the thermal expansion of Si and Cu are different, resulting in chip compression stress. Typically, TSVs are placed on the peripheral or the center of a

chip. TSV-related defects might occur in the fabrication process of TSV placement, in the bonding of the TSVs to the next layer like wafer warpage [109]. The wafer warpage is considered a fabrication defect as a result of annealing process; it is more interesting for test engineers rather than researchers with fault evaluation or fault-tolerant design expertise.

TSV coupling: TSV (intra-die) links have larger capacitance values than 2D wires (inter-die links), which consequently increase the latency of 3D signal paths. Although buffer insertion method has been proposed with the goal of delay reduction, but it imposes extra area and power consumption overheads. The unexpected delay of data transmission through TSVs, depends on fabrication technology, physical parameters and design factors of TSVs. This is because the occupied die area by TSVs is quite significant which in turn compromises the wire length benefit of 3D ICs. On the other hand, small TSVs causes larger capacitance values depending on the liner thickness and doping concentration of the substrate. Meaning that, small TSVs have less area overhead, but they cause serious delay overhead.

Furthermore, this will be more critical considering the CMOS technology integration results in tighter timing requirements for circuits though TSV scaling and CMOS technology scaling are independent. So, the TSV coupling is one of the major issues for the future 3D NoC designs. In more details, TSV coupling may result in delay because of mutual coupling effects between adjacent TSV pairs [45, 62]. The term TSV coupling refers to capacitive and inductive couplings among neighbor TSVs. Electric field results in capacitance coupling and magnetic field is a source of inductive coupling. Inductive coupling among neighbor TSVs is more critical in higher frequency data transmissions [4], while capacitive coupling is more challenging in lower range of frequencies [123]. TREAT, discussed in more detail in Chapter 5, is able to simulate

the effect of capacitive TSV-to-TSV coupling fault models as dynamic fault injection.

Thermal stress: The mismatch between the Coefficient of Thermal Expansion (CTE) of a TSV fill material and the silicon induces a residual thermal stress in the region surrounding the TSV [73, 71]. The thermal stresses can drive interfacial delamination between the TSV and the Si matrix, damaging the on-chip wiring structures [96, 91]. It can affect the carrier mobility due to the piezoresistivity and degrading the performance of the MOSFET devices. Thermal stress can also degrade the saturation current of the transistor down to 30% [94]. This parameter limits the maximum permitted number of TSVs on 3D NoC by increasing the Keep-Out Zone (KOZ) parameter [63, 66].

4.1.2 Thermal Concerns

Thermal concerns and effective heat removal methods are highly demanded in 3D NoC designs since on-chip circuits are stacked on top of each other and packed densely. Temperature cycling and thermal shock accelerate fatigue failures depending on the environmental temperature range [51]. Thermomigration is mass transport driven by a temperature gradient, which unbalances a homogeneous alloy under a temperature gradient. Thermomigration is not a concern in Al and Cu interconnect technology; but it is now recognized as a serious reliability problem for flip chip solder joints in 3D packaging.

Transistors, contacts, multi-layered Cu, Al interconnects, and solders joints are the sources of heat in 3D designs. The majority of power consumption is attributed to transistors, because the resistivity of silicon is much higher than metals and there are more than a billion transistors on advanced chips. Furthermore, the heat generated

by solder joints is much lower than the transistor and interconnects metallization; it affects locally the area surrounding (underfill) solder bumps. The polymer-based underfill has a low glass transition temperature, so the thermal concern may change the viscosity of the underfill. The flow of the underfill reduces its role in the protection of the bump as well as the chip. Thermal concern is considered as waste-heat that increases the conductor temperature, resulting in more joule heating. Thermal concern increases the device temperature and affects atomic diffusion [109]. Four main thermal issues in 3D NoCs are reported in the following subsections:

Electromigration: Electromigration is a failure mechanism where electrons flowing through metal (Al, Cu) lines collide physically with the metal atoms, causing the metal atoms to migrate and form voids in the metal lines which leads to increased metal line resistance and disconnection. Electromigration is a key failure mechanism that determines the long-term reliability of metal lines. It strongly depends on the material of the metal. Copper has more resistance against electromigration as compared to Aluminum. It also depends on the running temperature of the system [102].

Time-Dependent Dielectric Breakdown (TDDB): The importance of leakage power has increased dramatically as technology scales down. Leakage (off-state) current has a direct relation with temperature while it has reverse relation with on-state current of a transistor [68]. Additionally, leakage current of MOSFET gates depends on the quality and thickness of oxide gate. When the leakage current of gate oxide reaches its limitation, the breakdown may happen which results in the failure of device. Furthermore, thin gate oxides and their silicon/silicon-dioxide interface are affected by various physical mechanisms like Hot Carrier Injection (HCI), Negative Bias Temperature Instability (NBTI), and TDDB. TDDB has been considered among researchers as a significant failure mode for deep sub-micron technologies. TDDB is

known as one of the main issues for high temperature in thin oxides [68, 75, 65]. It is much more critical in 3D designs because of thermal concerns and thermomigration effect among the layers.

Stress migration: Stress migration is a failure mechanism in which the applied stress to metal lines causes the metal atoms to be crept. Stress is generated in the metal lines (Al, Cu) used in the IC due to temperature differences between the heat treatment process in the manufacturing process and the operating environment temperature. This stress can cause composition deformation in metal lines, resulting in short-circuits between metal lines, or vacancies in the metal lines causing creep and converge in a single location which consequently form a void [102]. In more details, stress migration occurs due to the interaction between the metal line stress and the metal atom creep speed. Whereas the metal atom creep speed increases at high temperatures, the stress acting on the metal lines decreases at high temperatures, so there is known to be temperatures peaks at which stress migration occurs.

Thermal cycling: Thermal cycling is the process of cycling through two temperature extremes. It causes cyclic strains and develops cracks in a similar way to natural usage and weakens the joint structure by cyclic fatigue [15]. The effect of thermal cycling gets worse in 3D design, since the temperature of different layers are not the same and there is always a thermal flow between layers. Thermal cycling may also affect other thermal issues such as electromigration, stress migration, and thermal stress.

4.1.3 Single Event Effect (SEE) Impacts

SEEs induced by heavy ions, protons, and neutrons become an increasing limitation of the reliability of electronic components, circuits, and systems. SEE has been the main concern in space applications, with potentially serious consequences for the spacecraft, including loss of information, functional failure, or loss of control. It can be destructive or transient, according to the amount of energy deposited by the charged particle and the location of strike on the device. ICs malfunctions due to radiation effects from high energy alpha particles at ground level are a major concern because of continued technology size scaling [85]. The electron carriers are collected by the electric field and cause the charge collection to expand resulting in a sudden current pulse. The diffusion current dominates until all the excess carriers have been collected, recombined, or diffused away from PN junction area [48, 35]. To investigate the effects of physical SEE faults on electronic systems at system level is considered by researchers to establish and verify the target system's robustness with a reasonable level of accuracy in shorter time. In general SEE is divided into two main categories: Single Event Upset (SEU) and Single Event Transient (SET) [7].

SEU: It is a change of state caused by ions or electromagnetic radiation striking a sensitive node in a micro-electronic device. These phenomena can affect the behavior of sequential circuits such as memory cells, register files, pipeline flip-flops, and cache memories. The sensitivity of both PMOS and NMOS transistors are high when they are off. The upset rate of can be estimated by having the knowledge of the mechanism by which radiation particles cause the anomaly. Traditionally, two different major space radiation sources ending in SEUs were: high energy protons and cosmic rays, specifically, the heavy ion component of either solar or galactic origins. The heavy ions cause direct ionization within a device while protons can make a large contribution

to the overall upset rate. However, within the Earths atmosphere and at the ground level, neutrons are the most frequent cause of SEUs [111].

SET: It is a temporary variation in the output voltage or current of a combinational circuit due to the passage of a heavy ion through a sensitive device results in an SET. In analog devices, SETs called Analog Single Event Transients (ASETs), are mainly transient pulses in operational amplifiers, comparators or reference voltage circuits. When a charged particle hits a logic cell sensitive node in the combinational logic, it generates a transient pulse in a gate that may propagate in a path and eventually be latched in a storage cell [52].

As the goal of TREAT is to present a run-time reliability evaluation tool for 3D NoC, the chip warpage is not considered as a fault library in the TREAT. However, it can be easily added as a simple predefined permanent fault library in case it is needed. In addition, to study the effects of thermal stress on TSVs and thermal concerns needs a full system simulation to profile the chemical interaction of materials with different CTEs. In other words, the physical parameters of TSVs are critical in accuracy of any proposed dynamic thermal concern and thermal stress fault models for 3D structures. So the current version of TREAT is not able to model dynamic thermal stress and thermal concerns, but they are supported as static fault libraries in the current version of TREAT. As discussed earlier due to structural implementation of TREAT any fault library model can be added later as a plug-in to the TREAT.

4.2 Logic-level Fault Models

To model the presented faults in Section 4.1.1 at the physical-level, one has to develop an accurate reliability evaluation of 3D NoC, but it is time consuming as compared to

logic-level or higher level modeling. To model the physical fault effects by the logic-level fault model is preferable in order to keep the complexity low and simulation time short, while meeting the required accuracy [29, 36]. The logic-level fault models which are capable of modeling each of the presented physical faults in 3D NoC environment are also shown in Figure 4.1. Evaluating the reliability of 3D interconnection architecture at the logic-level enforces and facilitates the process of fault injection as a fault forecasting method.

Logic-level fault models represent the effect of physical faults on the behavior of the modeled system. The results of early studies with logic-level fault models provide the basis for fault simulation, test generation, and other testing analysis applications. A higher level fault model allows derivation of the inputs that test the chip without any knowledge of characteristics of physical-level failure causes. The accuracy of logic-level fault modeling will be increased by covering all the necessary areas of testing analysis. In addition, the physical-level is not adapted for the simulation of complex designs such as 3D NoCs because of their long simulation time. This approach decreases the complexity by employing a single logic-level fault model for different technologies, and representing the effects of physical faults which are not completely understood.

Logic-level model of 3D NoC for the qualification test must be close enough to the physical-level to include all the unwanted impacts of existing physical faults on the system. Understanding causes and effects of physical faults prevents extreme computation time augmentation in logic-level modeling by providing an intelligent fault modeling technique. Table 4.1 summarizes the causes and correlated logic-level fault models for each of the introduced physical faults. A brief description of all the represented logic-level fault models in Table 4.1 are presented in the following paragraphs:

- **Bit-flip:** It is used to emulate the effects of SEU and other transient disturbances.

Table 4.1: Physical faults and their corresponding logic-level fault models

Physical Fault Name	**Cause**	**Logic Fault Model**
Chip warpage	• Crack in Micro-bumps [109]	• Open-circuit
TSV coupling	• Increase path delay [59, 62] • May result in a wrong logic function switch and cause unintentional flip in the target signal	• Delay-fault • Bit-flip
Thermal stress	• Affects carrier mobility of transistor which degrades the performance [91]	• Delay-fault
Electromigration	• Increase wire resistance [102] • Disconnection	• Delay-fault • Open-circuit
TDDB	• Dielectric breakdown of the gate dielectric film influences on transistor behavior	• Stuck-at-0-1
Stress migration	• It deforms composition in metal lines or makes vacancies (atom holes) in the metal lines, forming a void by creeping and converging in a single location [75]	• Open-circuit • Short-circuit
Thermal cycling	• Interfacial crack [94]	• Open-circuit • Delay-fault
SEU	• Electrical noise induced by high energy ionizing particles on sequential circuits	• Bit-flip
SET	• Electrical noise induced by high energy ionizing particles on combinational circuits	• Stuck-open

A bit-flip in any resource will only manifest itself once this resource is read to perform an operation [13]. Activating several of them at the same time results in modeling Multiple Event Upset (MEU) physical faults.

• **Stuck-at:** If there are short circuits to the ground or to the supply voltage, a

stuck-at-0 or stuck-at-1 fault has occurred, respectively. These sorts of faults can be modeled by forcing a signal to logical value of 0 or 1 [13, 110].

- **Bridging:** It models a situation in which two signals are connected to each other when they should not. This may result in a wired-OR or wired-AND logic function [83].
- **Stuck-open:** It models an open circuit in the drain or source interconnects of parallel transistors in combinational logic gates which can occur during the operation [16].
- **Delay-fault:** It represents physical faults where the signal eventually carries the correct value, but more slowly (or rarely, more quickly) than normal [84].
- **Open-circuit:** The open-circuit fault model simulates increasing the resistance of wires to an extremely high value, ceasing the current flow [104].
- **Short-circuit:** Short circuit models any unwanted wiring connections in the system [104].

4.3 Fault Effects on 3D NoC

The potential faulty locations are precisely recognized by exploring 3D NoC architecture in addition to the physical fault sources in Section 4.1. 3D NoC is composed of different components and links, including 2D planar structures bound by inter-die links as discussed in Chapter 3. The vulnerable components of 2D planar NoC and TSV links are itemized in Figure 4.2. Categorizing sensitive components of 3D NoC results in more accurate and faster reliability analysis evaluation. Therefore, a reliability analysis tool can utilized this information to model the related fault models in specified location rather than considering the whole design with random compo-

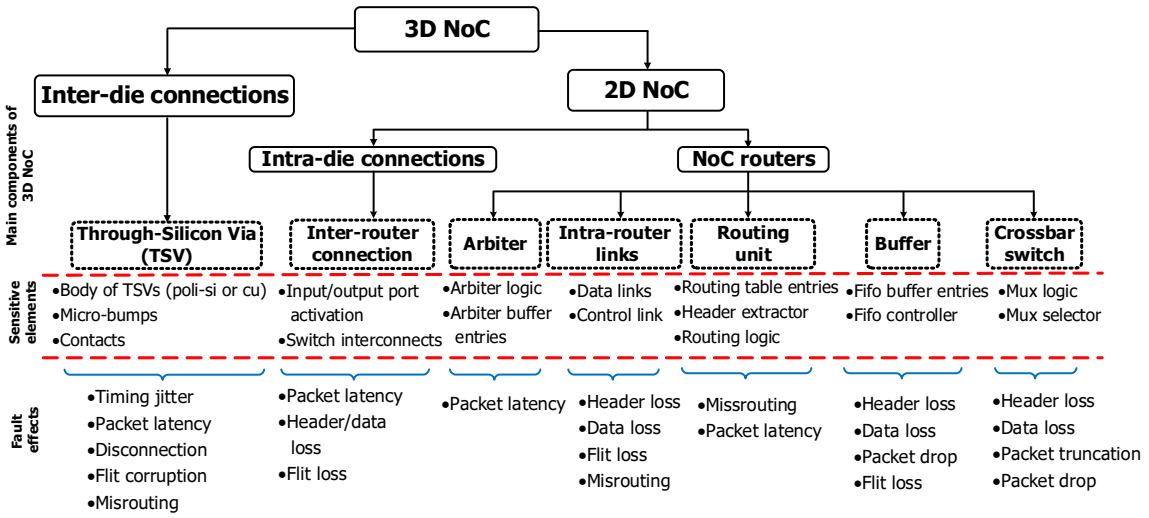


Figure 4.2: Classification of faults and their effects in 3D NoC components

nent selection. In addition, different sources of fault have different effects in 3D NoC components which are itemized in Figure 4.2. These fault effects can be utilized for any 3D-NoC specific reliability evaluation metrics and some of them are applied in TREAT. These effects include header/data flit loss, packet drop, packet truncation, packet latency, misrouting, timing jitter, flit corruption, and disconnection. Reporting the percentage of each of these metrics is a an important approach to compare the reliability of any fault-tolerant design. A brief description of them is listed next.

- **Header/data flit loss:** It represents any header/data flit alteration of an incoming packet. It would happen if there is any problem with the logic circuit of crossbar switches, FIFO controller, or internal connections among components inside a router. Header flit loss which has a lower probability than data flit loss is more critical. A packet might never reach its expected destination, if a header loss occurs.

- **Packet drop:** The incoming packet will be skipped or forwarded to an invalid output port if a fault occurs in input buffer pointers/counters or output switch.

- **Packet truncation:** It happens when erroneously a data flit is recognized as a tail flit, where one or more data flits from the end of a packet will be lost.

- **Packet latency:** Packet latency is the result of the router arbitration malfunction because of a fault inside the arbiter logic, internal connections, or TSV connections. TSV issues and some of the thermal related faults may change the conductivity of TSVs or internal links by changing their physical structures.
- **Misrouting:** Misrouting is the consequence of fault occurrence on either header flit while transferring among NoC architecture components or on comparator modules of routing unit component. It can be resolved if it happens because of any transient faults. Permanent faults in a routing unit component result in extra number of packet transmissions between adjacent routers and consequently causing congestion in some parts of the network.
- **Timing jitter:** The short-term variations of a digital signals significant instants from their ideal positions in time is called timing jitter. It is a significant, and usually undesired, factor in the design of almost all communications links. The delay caused by temperature variation and TSV coupling, leads to a timing jitter, which is more significant for TSV because of its size.
- **Flit corruption:** Flit corruption in a packet happens when a fault occurs in the data path of a router. It may occur in intra-router links, crossbar switch components, or TSV link.
- **Disconnection:** Different types of faults may cause the vertical link components (TSV body, contact, micro-bump) to fail. Electromigration affects the body and micro-bumps, causing the path to be implied as an open link or disconnected.

TREAT uses some of these effects as reliability metrics that are discussed in more details in Chapter 5 in which the framework of the TREAT is presented.

Chapter 5

TREAT Framework

TREAT is a fault-injection based reliability analysis framework for 3D NoC environment. It uses logic-level fault models to evaluate the effects of physical faults which were discussed in Chapter 4. These fault models are added to TREAT as fault model libraries and they are inserted into the target model components based on user's request. TREAT is capable of activating static and dynamic fault models through two separate simulations set-ups. The current version of TREAT is able to model SEU, SET, TDDB, Thermal cycling, Thermal stress, and Electromigration physical faults through a static fault injection process. It is also able to estimate the effects of TSV coupling physical fault in 3D NoC designs through a dynamic fault injection process. It has been developed in a structural way to support future plug-in fault models. It traces the given Verilog HDL description of a 3D NoC design regardless of its description types (behavioral, structural, or data flow) to accomplish the fault injection process. This chapter describes the major components of TREAT in addition to the static and dynamic fault modeling implementation in TREAT.

5.1 TREAT Components

Similar to a typical fault injector tool, TREAT is composed of five major modules including: workload, user interface, fault injector manager, target system, and statistics report components. These modules with their subcompoents are illustrated in Figure 5.1. This figure represents the modified version of Figure 2.3, in which the basic components of a typical fault injector tool are discussed.

The **workload** component is necessary to evaluate the reliability of different components by running various traffic patterns. More variety of workload applications is desired to guarantee all routers are active during reliability evaluation process with different network congestion spots. TREAT actually employs the NI component of THENoC which interprets both synthetic and real data traffics and converts them into appropriate transmission packets as discussed in Chapter 3. But, it is also able to generate synthetic traffic applications such as random with uniform distribution and permutation traffic patterns. For permutation traffic such as bit complement, bit rotate, bit reverse, neighbor, transpose, tornado, and shuffle, there is a unique destination for all source nodes [5].

The **user interface** module in TREAT interprets the given user defined information to accomplish the fault injection process. A sample set of information from the user is listed in Figure 5.1, which is mostly needed for the goal of static fault injection routine. More specifically, TREAT traces through all of these predefined values and stores them in a hash data structure, which is accessed periodically during fault injection process.

The **fault injector manager** is able to monitor the 3D NoC (target system) to implement dynamic fault injection and it also interacts with user interface to accomplish static fault model experiment. In more details, this component is composed of fault

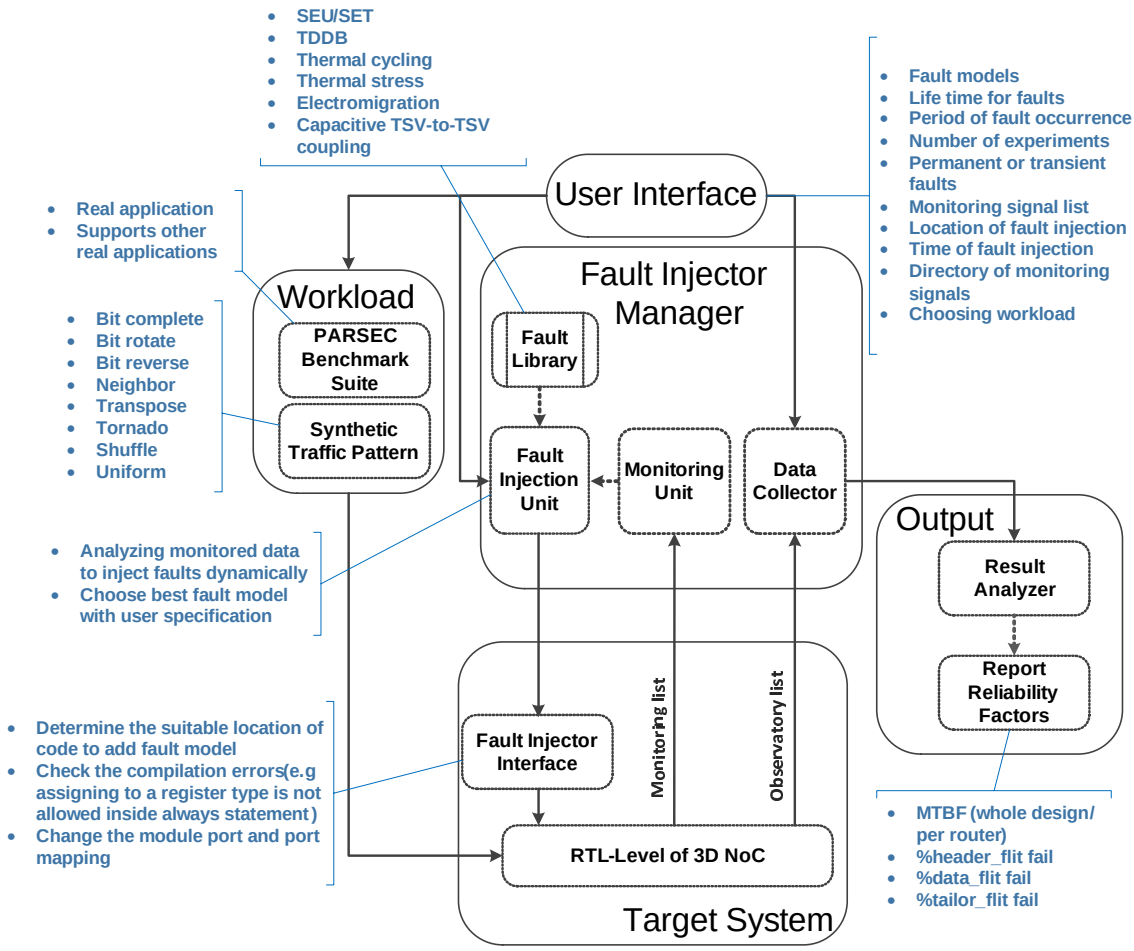


Figure 5.1: TREAT components

library, fault injection unit, monitoring unit, and data collector subcomponents. The fault library contains all the predefined fault models for TREAT. The supported fault models in TREAT are shown in Figure 5.1. Some of the physical faults such as SEE faults depends on environmental conditions while some others like capacitive TSV-to-TSV coupling are data dependent as discussed in Section 5.3. Due to structural implementation of TREAT, it is able to support the state of the art and future fault models. The fault injection unit is responsible to update the target files for the fault injection process by accessing the fault library, user information, and the feedback of the target system. The feedback of the 3D NoC facilitates the process of dynamic fault injection. The monitoring unit is capable of profiling the predefined monitoring

list signals. It also analyzes the profiled signals and sends the results toward the fault injection unit subcomponent. Data collector is the other subcomponent of fault injector manager which profiles the observatory list signals with the goal of reporting reliability evaluation statistics.

Target system is the RTL model of the system which is going to be evaluated. THENoC is employed in this experiment, to verify the functionality of the TREAT. The fault injector interface as the major subcomponent of target system is responsible for injecting faults. The locations and time of fault injections are chosen randomly for static fault models in TREAT. More specifically, it declares all faulty signals in the top module of the target system in order to activate user defined fault models. It also automatically distributes all faulty signals over the smallest modules of 3D NoC design based on the user preference. To reach this goal, it updates all the needed modifications through the port definition and port mapping sections of top module down to the smallest ones. The result of this process supports the HDL syntax in which the target system is implemented, without changing the functionality. Since THENoC simulator is implemented in Verilog, the presented fault injector interface supports Verilog syntax for the goal of injection process. Unlike to many existing fault injector tools, TREAT does not need to merge all the HDL implemented module files into single file. It only needs the knowledge of design hierarchy in which, the top module and smaller components are specified. In addition, in case of having a vector faulty signal, TREAT is able to choose the index numbers randomly or based on the user preference. For dynamic fault injection experiments, TREAT is able to inject the faults run-time based on the feedback from the system, ending in more accurate reliability evaluation.

And finally, the **statistics report** module of TREAT is responsible to provide the reliability evaluation metrics. In case of static fault modeling, TREAT automatically

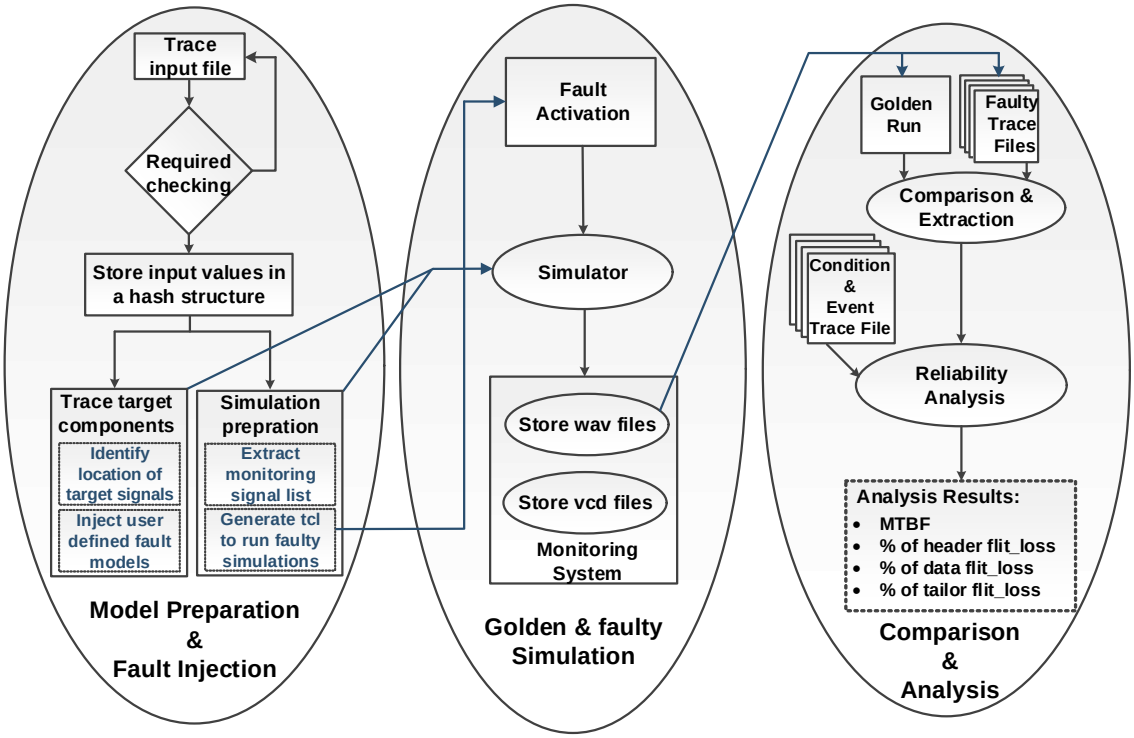


Figure 5.2: TREAT phases

runs the same simulation for various injection time and fault activation periods and stores the values of user defined signals in separate files, called faulty runs. TREAT also runs the simulation once without any fault activation and stores the values of user defined monitoring signals as a golden run. The result analyzer subcomponent compares the results of faulty and golden (faultless) experiments to present the reliability evaluation of the given 3D NoC design. This component reports the effect of psychical fault models which are discussed in [26] as evaluation metrics in TREAT. In dynamic fault injection process, there is no need to repeat the faulty experiments for several times since faults are accurately injected wherever and whenever they should happen by profiling the run time system. In this cases the effects of them on the functionality of different components of 3D NoC are reported as reliability evaluation metrics. Both static and dynamic fault injection processes in TREAT are explained in the following sections.

5.2 Static Fault Models in TREAT

Static fault models are employed to disrupt the values of user defined signals with predefined timing characteristics. Fault model injection time and life time are two major of these timing characteristics. For TREAT, static fault models are activated with a random uniform distribution function. TREAT guarantees to inject faults when the target system is stable. The life time of the static fault models are also defined by a random function with exponential distribution. The average value of this exponential distribution is requested from the user. TREAT runs the experiment for each of the static fault models for the predefined number of times specified by the user. In each iteration, the random function chooses different injection time and life time for the chosen fault model. The accuracy of reliability evaluation increases as more the number of simulations are repeated. However, there is a trade-off between the accuracy of the static fault modeling and simulation time.

The reliability evaluation of static fault models are accomplished in three phases (see Figure 5.2) including: (1) Model Preparation and Fault Injection, (2) Golden and Faulty Simulation, and (3) Comparison and Analysis, which are further discussed in the following subsections.

5.2.1 Model Preparation and Fault Injection

In this phase, the user input values are traced and stored in a hash data structure. This hash data structure is used several times during the fault injection process. TREAT warns if there is a conflict in the user defined input values. For example if the total number of requested faults in a component is less than the requested number of fault models for the same component, TREAT prompts a warning message. Or if

different fault models are defined to affect the same signal, TREAT notifies the user.

TREAT uses the input values to generate tcl commands for *cadence incisive* simulator tool as a part of automatic fault injection process. These commands can be simply replaced in case of applying any other simulator tools. It also provides the corresponding tcl commands to profile the user defined monitoring signals. In addition to simulation preparation files, the fault models are also inserted into the given Verilog models of the 3D NoC. TREAT recognizes the faulty signals and then inserts fault models differently based on how they are assigned in the target Verilog models. In other words, it uses different fault injection methods for concurrent and sequential structures to support Verilog syntax rules.

TREAT keeps track of different sequential block statements to assure the same requested signal is infected during the fault injection process. Also if there are multiple assignments for a given signal under different conditions inside sequential statement, all of them are infected with the same fault. For example if the given signal *tx* is assigned with different values inside different case statements they are all infected with the same fault if signal *tx*, is selected by user. With this consideration the effect of faulty signal is evaluated regardless of the 3D NoC operation, resulting in more accurate reliability evaluation reports.

Furthermore, TREAT is able to choose the victim indices of faulty vector signal or it uses the predefined indices by the user. In case of random index selection, TREAT is smart enough not to choose the same index numbers randomly. For example, if the vector *data* with the size of 32 bit is chosen as a faulty vector, TREAT will choose randomly 4 different index numbers by keeping track of all the previously generated random numbers.

5.2.2 Golden and Faulty Simulation

In this phase a script file is called to run the whole simulation process for both faulty and golden runs. This script actually employs the generated tcl commands in previous phase of fault injection. TREAT profiles the user defined monitoring signals in two different database formats: Simulation History Manager (SHM) and Value Change Dump (VCD) formats databases. In both of these data base formats, all the signals' events with their event time are captured. The size of these files depends on the simulation time and also the number of events on them. TREAT uses tcl commands to store only the signals which are defined by the user to save both simulation time and memory.

The VCD file is actually an ASCII file which includes header information, variable definitions, and the value changes for specified variables in a given design. The value changes for a variable are given in scalar or vector format, based on the nature of the variable. TREAT is embedded with an open source VCD parser which is able to interpret the value, time, and hierarchy of specified signals. The alternative database is the SHM one, which is a cadence proprietary waveform database, to store the selected signals. The current version of TREAT uses SHM database to report reliability evaluation reports. However, as discussed it also provides the VCD files and their interpreted versions for the same monitoring signals if needed.

5.2.3 Comparison and Analysis

The last phase of TREAT provides the reliability analysis reports of the given 3D NoC design. In this phase, the SHM database of the faulty runs are compared with the golden run. This comparison is accomplished by running another script in which the values of monitoring signals are compared automatically. To recognize whether the

signal values mismatch are because of timing delays or not, depends on the method of 3D NoC implementation. In other words, in some cases the signal value mismatch can be resolved by applying a fixed time shifting operation. If the given model of 3D NoC is able to provide a unique ID for each flits, TREAT is able to distinct the packet latency failure from the other types. But the current version of TREAT reports an inconsistency of two simulation experiment, if the signal values are not the same or they have same values in different timing slots.

TREAT is able to report analysis results such as: MTBF, %header flit loss, %data flit loss, and %tailor flit loss. More specifically, it can report these values in different granularity for each signals, module, or the whole system. It is also able to differentiate these values for each experiment or fault models. Providing the MTTF values is also possible by TREAT, but MTTF values are defined for none repairable systems (permanent fault modeling). Because of the importance of transient faults in future many-core systems, only transient fault model implementation are reported in this experiment; MTTF values are not reported as reliability analysis results.

As discussed in Chapter 4 reporting other reliability metrics such as packet latency needs to keep track of all the flits and depends on the test bench of the 3D NoC model. THENoC is able to report these values but current version of TREAT does not provide other reliability metrics independently.

5.3 Dynamic Fault Models in TREAT

In addition to different static fault models, one of the advantages of TREAT compared to other fault injector tools is the capability of supporting dynamic fault models. With this methodology, the accurate location and time are determined for fault in-

jection process based on the dynamic monitoring of the system. In other words, for dynamic fault models only a single experiment is needed. This feature compresses the simulation time dramatically with high level of reliability evaluation accuracy. Capacitive TSV-to-TSV coupling is the dynamic fault model supported by TREAT, which is discussed in this section.

TSV-to-TSV coupling is a major source of crosstalk between adjacent TSVs. [30, 53]. Looking at the cross-sectional view of a TSV bundle, it is assumed for the sake of analysis that mutual coupling is due only to neighboring TSVs that are directly adjacent, not diagonally adjacent. This is because the distance of diagonal TSVs is larger than adjacent TSVs, which is also assumed by other research groups [95, 17, 60, 107]. The presented Capacitive TSV-to-TSV Coupling (CTTC) categorizations is based on this assumption. On the other hand, the number of neighboring TSVs can vary depending on whether they are located in the middle or on the border of TSV meshes. Figure 5.3 illustrates all the possible locations of a victim TSV (each in a different color) relative to active adjacent (shown in gray). The colored and gray TSVs in Figure 5.3 represent victim and aggressors, respectively.

In this Section an analysis of the current flow in TSVs is first presented which is the fundamental concept for proposed dynamic fault model. Then the circuit-level model of TSV used for this experiment in addition to classification of CTTC effects are explored. This classification is based on the maximum capacitive coupling by a victim TSV, while the value of the resulting crosstalk depends on the TSV current flow direction. Finally, the CTTC effects on timing requirements of the circuits as well as dynamic fault modeling process in TREAT are discussed.

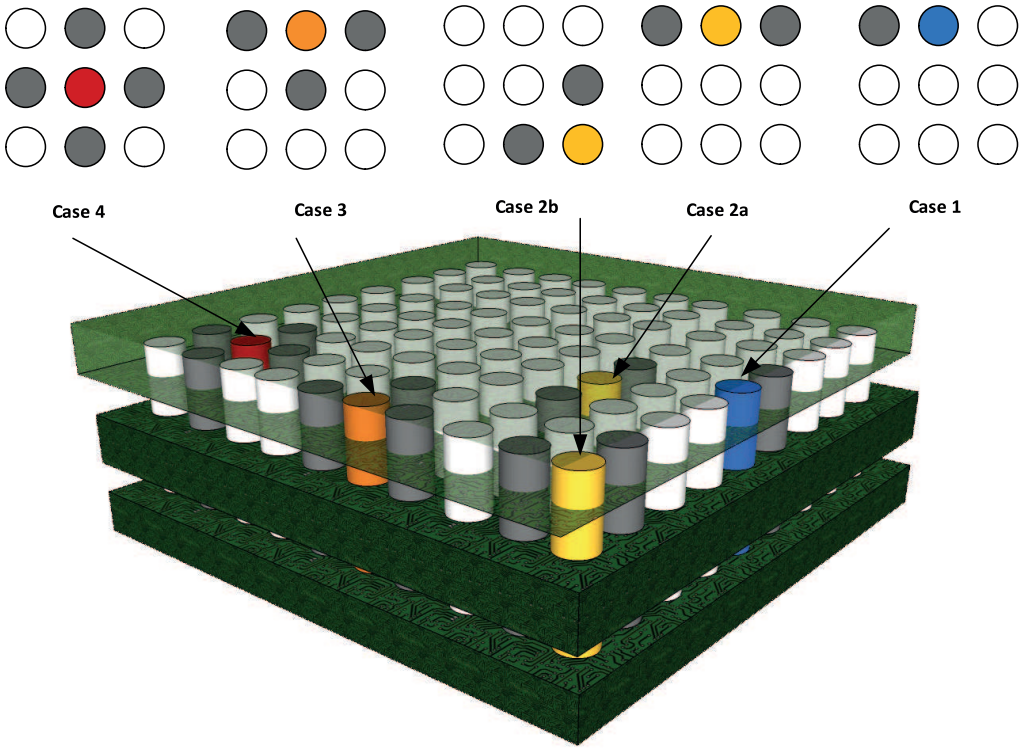


Figure 5.3: Different TSV patterns leading to various TSV coupling classes

5.3.1 Current Direction in TSV

In order to characterize the effects of capacitive and inductive coupling between TSVs used in a CMOS digital circuit, it is first necessary to characterize the direction of current in a given TSV, based on the direction of the data signal and the type of data transition present. The six possible cases are illustrated in Figure 5.4. According to these cases a TSV has three possible current flow directions including: Upward ($\otimes$), Downward ($\odot$), and No-current ($\bigcirc$). For the cases where the data is transmitted from an upper to a lower layer, Figure 5.4a shows that the TSV current is conducted downward if its voltage makes a high-to-low transition; Figure 5.4b shows that the TSV current is conducted upward if its voltage makes a low-to-high transition. For the cases where the data is transmitted from an upper to a lower layer, the currents are in the opposite direction of those indicated in Figure 5.4a and Figure 5.4b,

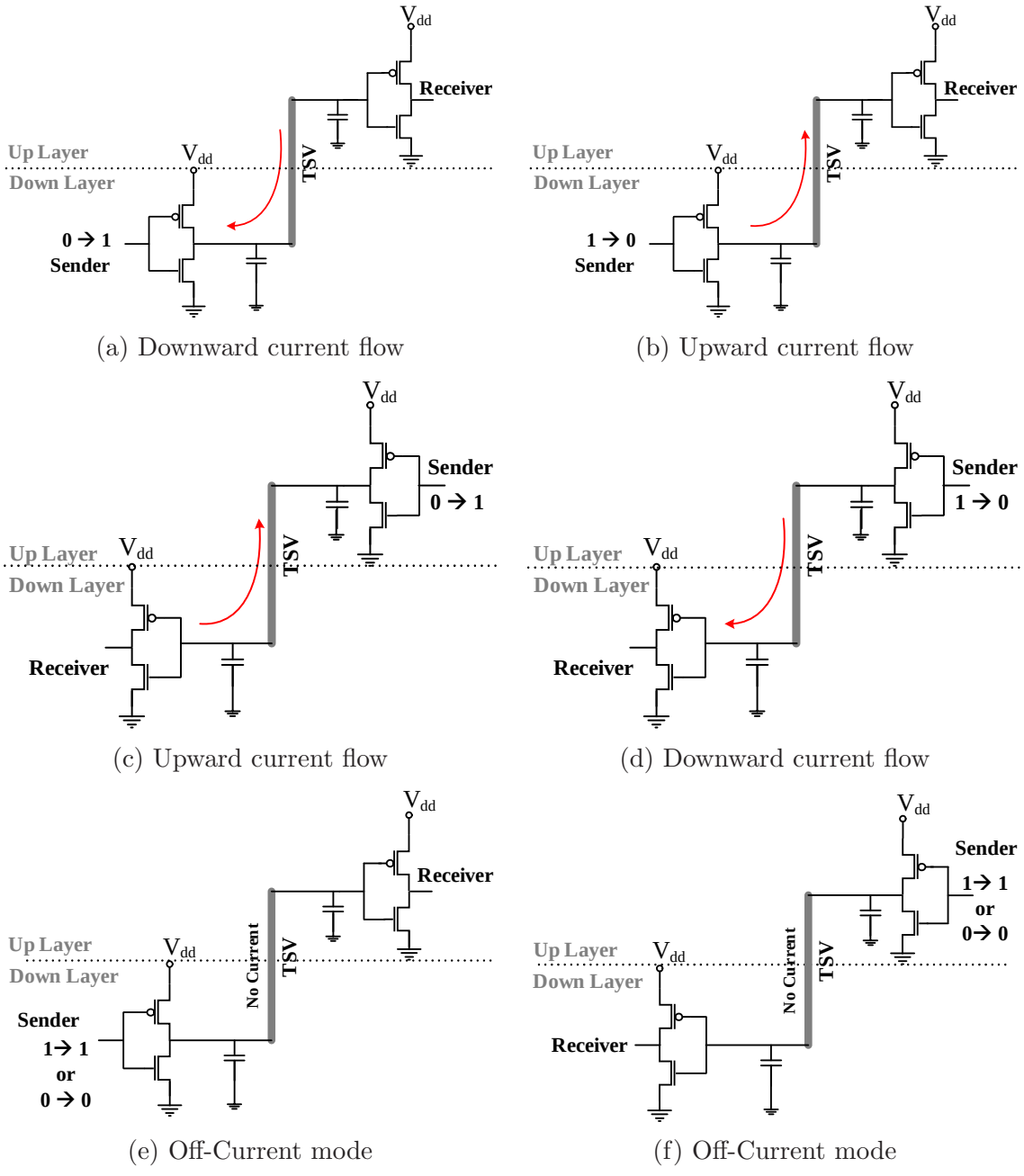


Figure 5.4: Current flow direction in TSV.

as shown in Figure 5.4c and Figure 5.4d, respectively. If there is no output data transition on the TSV, then no current will conduct, as shown in Figure 5.4e and Figure 5.4f. This is because there is no potential difference between two sides of a TSV. In the rest of this dissertation a TSV without an electric current is called an inactive TSV. Furthermore, in the following subsections, all possible TSV current flow configurations resulting in different inductive and capacitive parasitic noises are used to classify CTTC.

5.3.2 CTTC Circuit-level Modeling

A framework consisting of multiple TSVs at circuit-level using Synopsys HSPICE is implemented in this experiment, to study the sources of CTTC effect. Developing TSV simulation framework allows extracting the realistic accurate CTTC effect for different parameters. The coupled TSV structure is modeled as a lumped RLC circuit. The lumped RLC circuit circuit-level model of TSVs presented in [62, 56, 81] is employed for this experiment.

Figure 5.5 illustrates the circuit-level model of TSV applied in this experiment, in which R_{TSV}, L_{TSV}, R_{si}, C_{si}, and C_{TSV} represent TSV resistance, TSV inductance, substrate resistance, substrate capacitance, and oxide capacitance, respectively. The value of the circuit elements are modeled based on physical and operational parameters of TSVs such as oxide thickness, silicon substrate height, TSV radius, and TSV pitch and by material properties like dielectric constant and resistivity. The actual values of these physical parameters are extracted from ITRS reports [45]. The thermal impact is also considered in the TSV model using equations in [61]. Furthermore, since the parasitic capacitive effect of a diagonal neighboring TSV is less than 1/5 of an adjacent TSV neighbor [28], only the effect of adjacent TSV neighbors are exam-

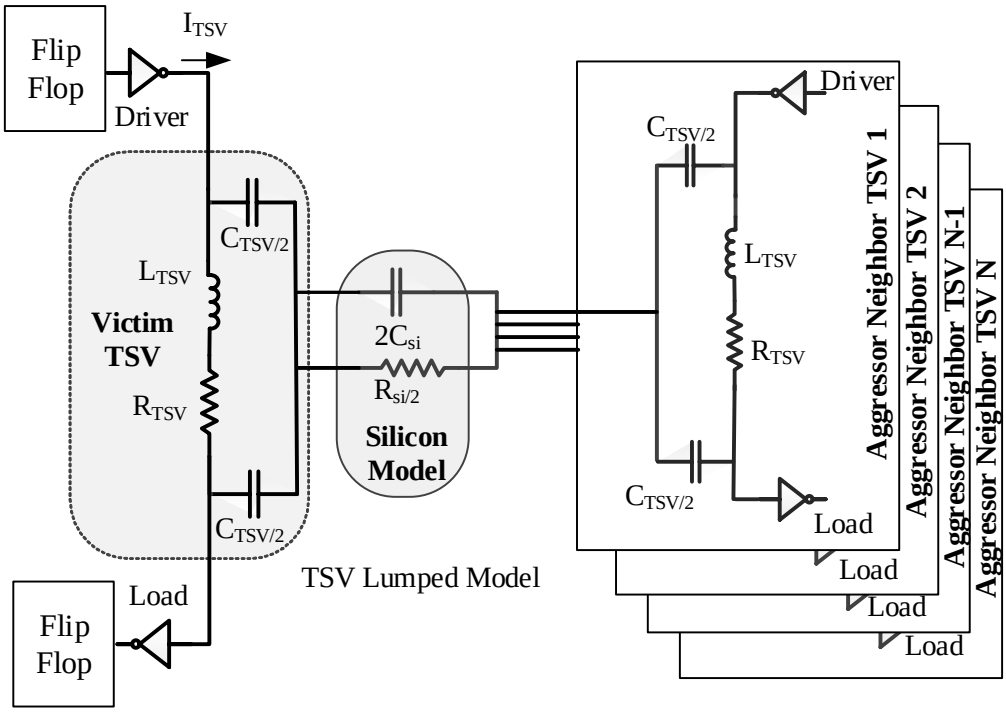


Figure 5.5: Circuit-level of victim TSV and its neighbors

ined in this experiment. However, our fault model can be expanded to support more TSVs as well.

In this analysis, a TSV is connected to the output of an inverter (driver) on one side and to the input of another inverter on the other side (load). These inverters are needed to record the propagation delay and its dependency to parasitic capacitivenoises. Two flip-flops, one before the driver inverter and one after the load inverter are inserted to capture the parasitic capacitive effects on timing requirements of the circuit. The input data pattern with output data pattern to catch the parasitic capacitive effects are compared.

Predictive Technology Model (PTM) [87] FinFET transistor models are employed to implement inverters and flip-flops. Then a comprehensive set of simulations is performed on the developed TSV framework. The impact of operational frequency, temperature, technology, TSV radius, and TSV oxide thickness are investigated which

is discussed in 5.3.4. SPICE model of TSVs are employed to examine the CTTC effect among a victim and its aggressor TSVs.

The range of all of these parameters are also chosen according to ITRS [46] and industrial interconnect reports. The minimum TSV depth normally is about 40-100μm which is projected to reach 30-40μm by 2018. A copper TSV in standard Si-bulk technology is expected to have minimum via diameter of 2-4μm, 1:20 minimum aspect ratio, 4-7μm minimum via pitch, 0.5μm oxide thickness (t_{ox}), and there can be up to 2-8 die per stack [46]. TSV diameters and pitches are two to three orders of magnitude larger than transistor channel gate lengths. Furthermore, in order to reach a high yield rate, manufacturers typically impose a minimum TSV density in order to maintain the planarity of the wafer during chemical and mechanical polishing [106]. For example, Tezzaron requires that there be at least one TSV in every $250\mu m \times 250\mu m$ area [54].

The characterization for a range of operational frequency and different TSV parameters is elaborated in Figure 5.6. In all of these graphs the vertical axis is the additional delay, relative to the clock period, caused by the parasitic capacitive coupling, which is given by:

$$TV = \frac{APD - NPD}{T_{clk}} = f_{clk}\,(APD - NPD) \tag{5.1}$$

where APD refers to Actual Path Delay (when there is CTTC), NPD refers to Nominal Path Delay (when there is no CTTC), T_{clk} is clock period, and f_{clk} is the clock frequency.

In Figure 5.6a, increasing clock frequency does not have tangible effect on the CTTC severity, but TV is increasing linearly for larger operational frequencies. This is because tighter timing requirements are needed in higher operational frequencies. For

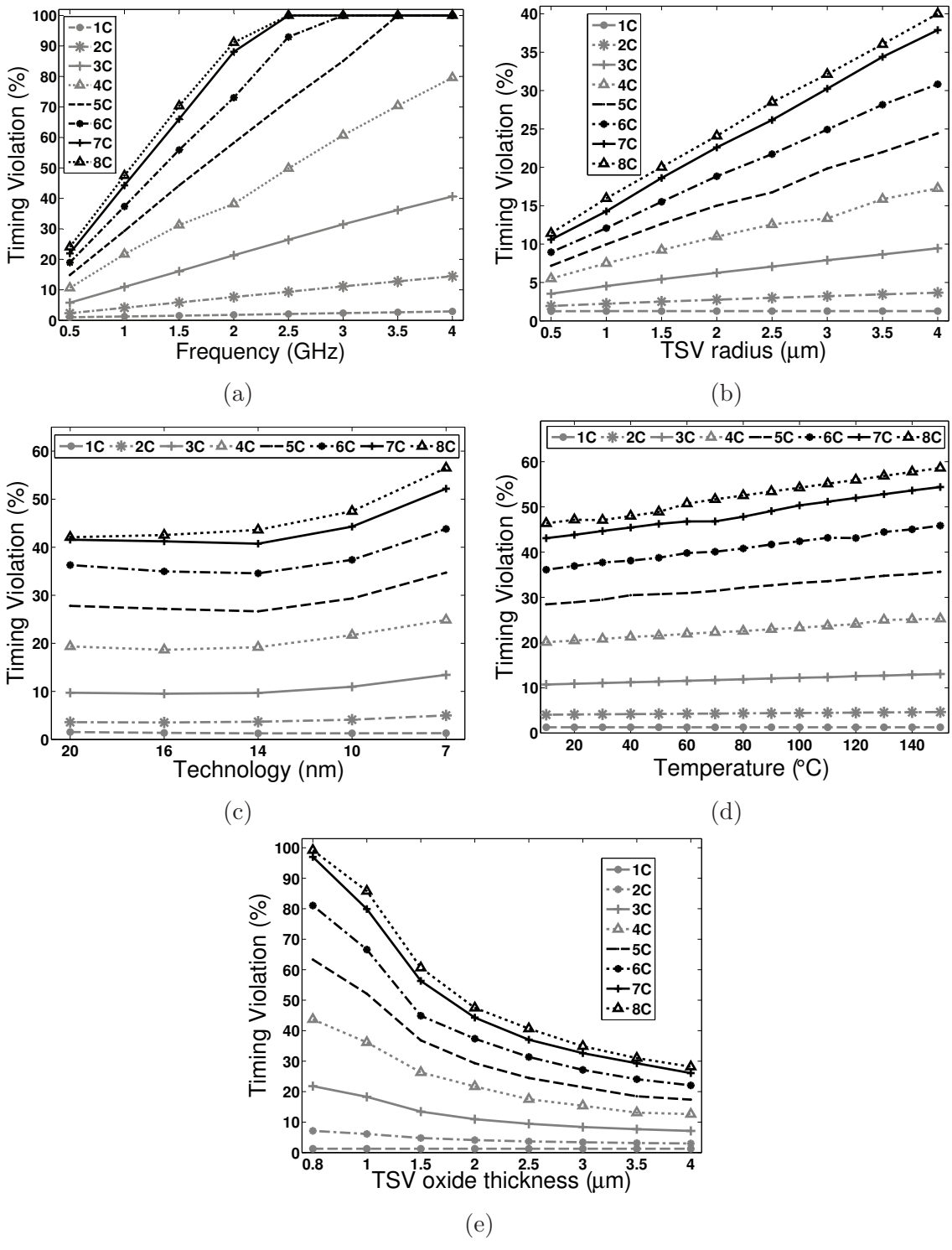


Figure 5.6: Characterizing capacitive coupling against various parameters

a TSV pair with a constant pitch, as the radii of the TSVs increase, more capacitive coupling is observed; this results in larger TV values as shown in Figure 5.6b. As the technology scales down the input capacitance of the flip-flop seen by the TSV decreases, resulting in larger coupled voltage on the TSV (shown in Figure 5.6c). The permittivity of the silicon rises as a weak linear function of temperature [61], which results in an increase in capacitive coupling, also increasing coupling and TV, as depicted in Figure 5.6d. Finally, as shown in Figure 5.6e, thicker oxide provides better isolation and reduces the value of capacitive coupling, resulting in less capacitive coupling and consequently less TV.

5.3.3 CTTC Effect on TV

Having analyzed and modeled the CTTC, the effect on realistic data traffic using PARSEC benchmark has been evaluated. This realistic data benchmark is run on the circuit-level model of 64 TSVs for various configurations. The TV values on the receiver side of each TSV is captured. A configuration is a set of physical parameters including TSV radius, length, pitch, oxide thickness, and process technology, operating frequency, and temperature. The configuration values are selected in a way to cover different CTTC effects.

Figure 5.7 shows the probability of TV for PARSEC benchmark workloads for three different TSV configurations (presented in Table 5.2). Each group of three bars in this figure from left to right refer to configurations A, B, and C in Table 5.2.

In addition, the TV probabilities are reported for different percentage of maximum synthesis frequencies. The results show that, the percentage of TV for lower synthesis frequency drops down for all workloads, however, an average of 40% TV at 100% synthesis frequency still confirms the importance of CTTC analysis for TSV-based

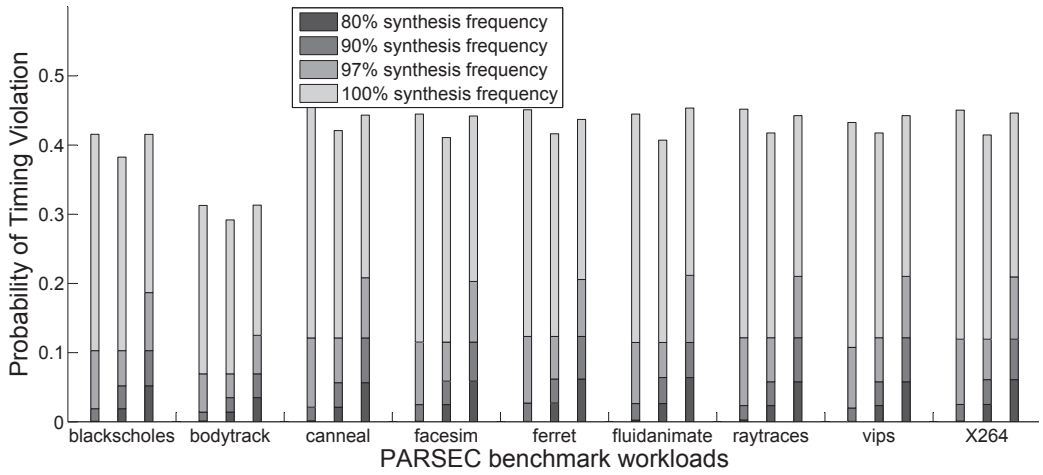


Figure 5.7: Probability of TV for PARSEC benchmark workloads

3D-NoC architectures. Furthermore, the result of Figure 5.7 shows how CTTC limits the maximum operational frequency of 3D NoCs. The reported values in Figure 5.7 are based on running PARSEC benchmarks under different conditions which are listed in Table 5.2. The reason is that the propagation delay of TSVs are extended as an effect of CTTC. In other words, the circuit is able to handle more percentage of TV with lower operational frequency.

5.3.4 CTTC Classification

A capacitive coupling classification is presented here based on the severity of CTTC parasitic values which depends on TSVs' current flow. On the other hand, the current flow direction of a TSV is data-dependent as discussed in Subsection 5.3.1; consequently CTTC depends on the data pattern. The previous ($Data_{i-1}$) and current ($Data_i$) data bit values of a TSV's driver should be evaluated in order to identify the current flow of each TSV. Assuming the effect of adjacent neighbor TSVs is higher than diagonal ones, the capacitive coupling between an adjacent pair of TSVs is represented in this experiment, by 0C (if they both have the same current direction), 1C (if one of them is inactive and the other is active), 2C (if they are both active

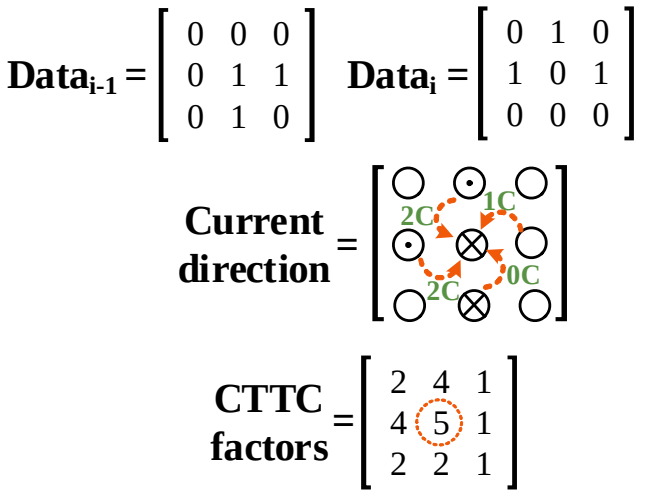


Figure 5.8: Current and CTTC matrices in a 3×3 mesh of TSVs

with reverse current direction). The total capacitive coupling voltage on the victim TSV is equal to the sum of voltages coupled by each aggressor on the victim TSV. Therefore, the total CTTC can be quantified as:

$$CTTC_{tot} = \sum_{i=1}^{N} |d_{vic} - d_{agg_i}| \tag{5.2}$$

where $CTTC_{tot}$ is the total capacitive coupling factor on a given TSV (which is used in categorizing capacitive coupling types in Table 5.1), N represents the number of adjacent aggressors for a victim TSV, and d is the current direction for the corresponding TSV. The current direction d in Equation 5.2 represents an upward current with +1, a downward current with -1 and no-current with 0.

Figure 5.8 shows an example of current matrix generation process for the given data bits values in a mesh of 3×3 TSVs. It also shows the generated CTTC factors for all TSVs, caused by their adjacent neighbors. $CTTC_{tot}$ factor on the victim TSV (situated in the middle one) for given data bit values in Figure 5.8 is equal to 5C, which is the sum of generated capacitive coupling values by each adjacent TSV neighbors.

There are $3^5 = 243$ possible TSV arrangements for 5 TSVs with three possible current flows, resulting in different capacitive coupling values from 0C to 8C. Many of these 243 arrangements generate the same capacitive coupling values, which are quantified in terms of their frequency and probability of occurrence in Table 5.1. A sample pattern resulting in each capacitive coupling is also shown in this table. The maximum $CTTC_{tot}$ in this representation is equal to 8C, if the middle TSV has reverse current flow comparing to all of its neighbors, as shown in the most right column in Table 5.1.

In addition, these capacitive coupling values can disrupt the timing requirement of a 3D IC based on the operational and TSV physical parameters. This issue, called Timing Violation (TV) in this experiment, is defined as additional delay to TSV lines caused by parasitic coupling capacitance. The effect of CTTC for each of the capacitive coupling types presented in Table 5.1 are characterized at the circuit-level model.

5.3.5 CTTC Fault Model

Circuit-level simulation takes much longer than system-level simulation. This approach considers the effects of circuit-level representation of CTTC for a system-level platform to reduce simulation time while maintaining accuracy. This operation is performed at runtime by monitoring the data signals that are being transmitted through

Types	0C	1C	2C	3C	4C	5C	6C	7C	8C
Sample pattern	⊙ ⊙⊙⊙ ⊙	⊙ ⊙⊙⊙ ○	⊙ ⊙⊙⊙ ⊗	⊙ ⊙⊙○ ⊗	⊙ ⊙⊙⊗ ⊗	○ ⊙⊙⊗ ⊗	⊗ ⊙⊙⊗ ⊗	⊗ ○⊙⊗ ⊗	⊗ ⊗⊙⊗ ⊗
Occurrence frequency	3	16	44	64	54	32	20	8	2
Occurrence probability	0.01	0.07	0.18	0.26	0.22	0.13	0.08	0.03	0.01

Table 5.1: CTTC categorization with occurrence probability of each type

Table 5.2: Different configuration of TSV arrays

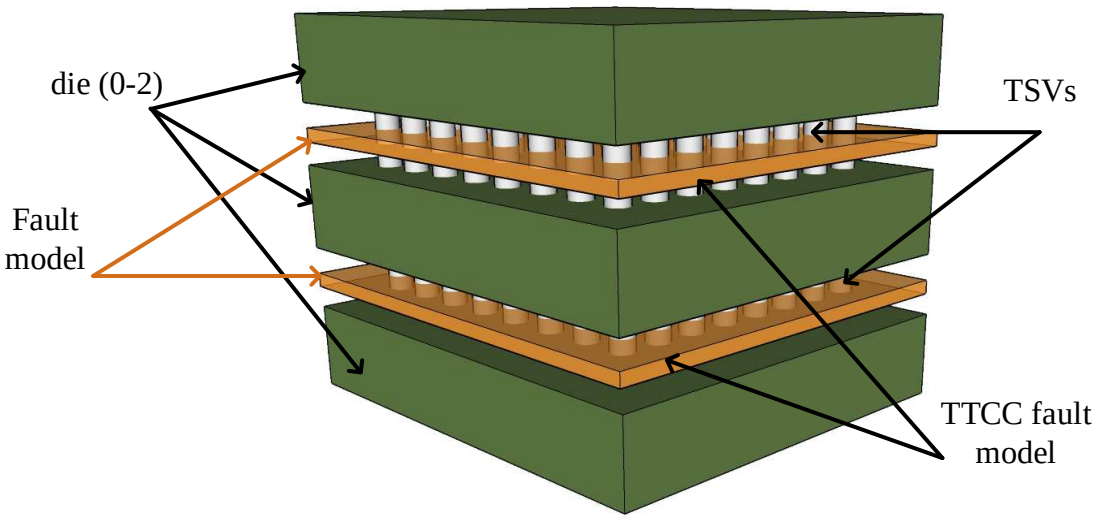
Configuration	A	B	C
radius μm	3	3	2
length nm	15	20	25
pitch μm	9	9	6
T_{ox} μm	3	2.5	1
Technology nm	20	16	10
Max Freq. GHz	1	1	1
Temperature $^{\circ C}$	50	75	100

TSVs in order to identify where and when potential faulty TSVs occurred. Next these faulty candidate TSVs are triggered with the observed effect at the circuit-level of CTTC fault.

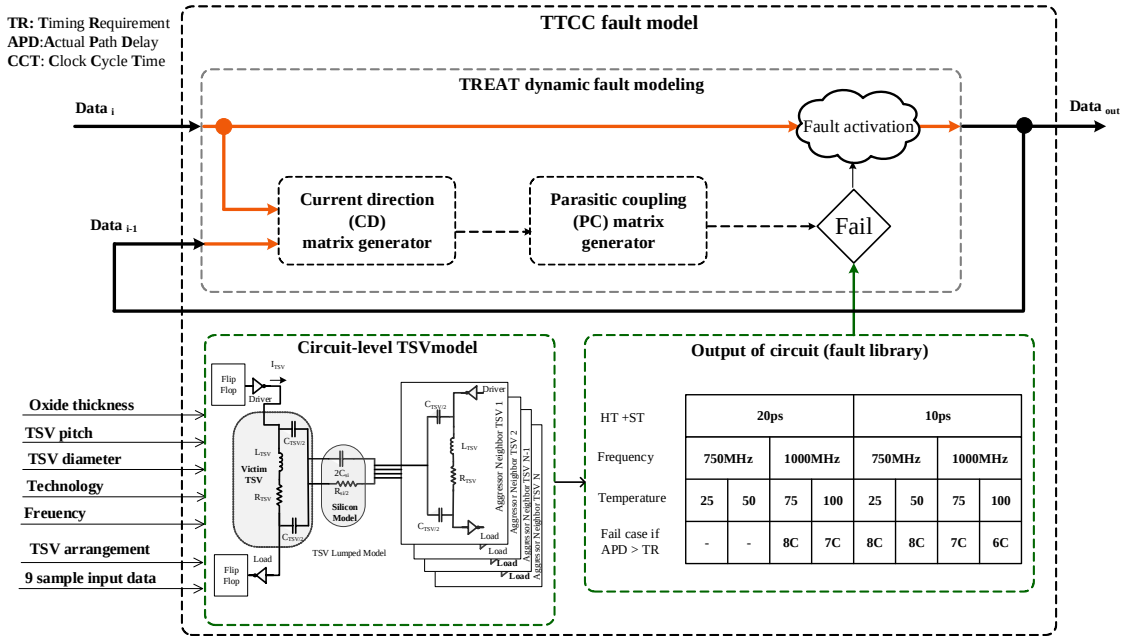
Figure 5.9 shows the 3D-NoC framework and the proposed CTTC fault model. The fault model is envisioned to be employed as an intermediate component among TSVs connecting routers in different dies, as shown in Figure 5.9a. This fault model does not affect the functionality of 3D-NoC; it only decides the time and location of fault activation through the TSVs based on data input patterns and the provided fault library at the circuit level. Figure 5.9b depicts the functionality of the proposed fault model in details. The input parameters of the circuit-level model are TSV arrangement configuration (number of rows and columns) connected to 3D-NoC, operating frequency, process technology, silicon oxide thickness, TSV-to-TSV pitch, TSV length, and TSV diameter. The output of this model is a table, which shows the corresponding parasitic capacitive factors violating the timing requirements for each configuration. An example of the fault library table is shown in Figure 5.9b. This output is used as a fault library in TREAT, in which the parasitic capacitive factors are extracted by comparing the transmitted ($Data_{i-1}$) and ready to transmit ($Data_i$) data bits through TSVs. With this configuration the CTTC fault model decides intelligently and accurately when and where a CTTC fault should be activated. The steps of this

methodology are as follows:

- **Step.0 Configuration and Setup:** Prior to instantiating and utilizing the devised CTTC fault model, first it needs to be configured and setup. In this phase, input parameters of the model such as the TSV length, TSV diameter, TSV pitch,



(a) Fault model location in IC simulator



(b) Fault model

Figure 5.9: Fault model usage demonstration in 3D-NoC

oxide thickness of TSVs, process technology, frequency, and temperature are specified. Frequency and temperature parameters are defined as a range with a specific granularity to support dynamic changes at run-time. The other inputs of the circuit-level model is all the possible data inputs resulting in 9 parasitic capacitive factors from 0C to 8C.

- **Step.1 Capturing transferred data:** In this step, the data bits transferring through TSV links (Up/Down port) are captured as the input of fault model at run-time. These captured data bits are adjusted if the fault activation condition is met.

- **Step.2 Data analysis to determine the TSV current directions:** At this step, the current directions of all TSVs are identified. The previous ($Data_{i-1}$) and current ($Data_i$) data-bit values of a TSV's driver are profiled and compared in order to recognize the current direction of each TSV. This process is done with the same approach as discussed in [30]. The output of this stage is stored as Current Direction (CD) matrix.

- **Step.3 Determine the induced capacitive coupling case for each TSV:** Looking at the current direction of each TSV and its adjacent neighbors, in this step an appropriate capacitive case is assigned per TSV. Now the fault library generated at pre-runtime is used to look up the timing delay associated for the corresponding case and is recorded as Parasitic Coupling (PC) matrix.

- **Step.4 Map circuit-level faults to system level delay fault/failure for each TSV:** Considering the timing requirement parameter of the destination receiver flip-flop (input buffer of receiver router), at this point, using PC matrix, the decision for faulty TSVs is made and an appropriate fault is applied to the detected faulty TSVs. If the reported timing delay associated with the capacitive case violates the timing requirement of the receiver logic, then the data bit is assumed faulty. The

fault type, depending on the specified delay tolerance for the circuit, can be either a *"delay fault"* or a *"glitch fault"*. However, if the specified timing requirement of flip-flop tolerates the undesirable propagation delay because of CTTC, then this fault will not have any impact and it is ignored in the model. Now the data is ready to be forwarded towards destination. Also a copy of last transmitted data (flit) is maintained for subsequent current direction determination.

Chapter 6

TREAT Evaluation

A case study is presented in this chapter to show the capability of TREAT in reliability evaluation of a 3D NoC platform. A cycle-accurate simulation, on the 64-node 3D mesh NoC is performed in this experiment. The parameterizable THENoC (see Chapter 3) simulator is applied as the target system in this experiment. More details about the simulation parameters such as the network and packet properties are summarized in Table 6.1.

The fault injection characteristics and experimental results for both static and dynamic faults are reported, separately. As discussed in Chapter 5 these simulations are executed through separate processes. In case of static fault injection, for each of the injected faults the simulation is repeated for user defined number of times, and they are all compared with the result of golden run. But, in dynamic fault injection a single experiment for each fault is enough to analyze the effect of capacitive coupling on functionality of TSVs in terms of TV (defined in Chapter 5).

Table 6.1: System configuration parameters

Parameter	Value
Topology	3D fully connected Mesh
Network Size	(4x4x4) 64 Routers
Flit Size	32 bits
Buffer Depth	8, 16-bit entries per port
Switching Scheme	Wormhole
Routing Algorithm	Dimension ordered routing
Simulator	THENoC [117]
Frequency	1GHz
Simulation time	10000 ns

6.1 Static Fault Injection Parameters and Experimental Results

As discussed in Chapter 5, TREAT interprets the user defined fault injection characteristics in order to launch the static fault modeling evaluation through THENoC simulator. The fault injection properties first discussed in this section and then the reliability evaluation metrics reported by TREAT are presented.

6.1.1 Static Fault Injection Characteristics

Transient faults are known as the major cause of failure in systems and they occur more frequently than permanent faults [99, 27, 31]. This experiment evaluates the effects of transient faults although TREAT is able to model permanent fault by updating the life time with the total running time value. This experiment is actually based on injection of 2048 transient faults in various points of THENoC. Repeating each experiment for 5 times results in 10240 transient fault injection process.

In this experiment, THENoC frequency is defined as 1GHz, and fault life time for each fault model is defined by an exponential distribution with average of 5 ns. The

Fault model	Components					
	Buffer	Header extractor	Switch	Routing unit	Tailor detector	Arbiter
Bit-flip	50%	–	–	28.5%	–	25%
Stuck-at-0/1	12.5%	50%	87.5%	28.5%	50%	25%
Delay-fault	25%	50%	12.5%	28.5%	50%	25%
Open-circuit	12.5%	–	–	14.25%	–	25%
Total fault distribution though components	25%	6.8%	25%	25%	6.8%	12.5%

Table 6.2: Fault injection distribution

fault injection process is executed when THENoC is stable. In this experiment, THENoC is simulated for 10000 ns and the fault models are injected randomly with uniform distribution in the simulation time gap between 1000 ns to 9000 ns. The fault models are distributed through major components of THENoC simulator without affecting the functionality of 3D NoC platform unless they are activated. The fault locations and their quantity is chosen based on complexity of each component and major physical fault causes in each component (as discussed in Chapter 4).

Table 6.2 summarizes the percentage and location of injected faults through different components of THENoC for this experiment. As expected more portion of fault models are injected through the buffer, routing unit, and switch components of THENoC. Also no Bit-flip fault is injected through the header extractor, switch, and tailor detector components since they are all composed of combinational circuits.

Furthermore, this experiment considers single fault effects although TREAT is also able to support multiple fault modeling with minimal modification of the function that is responsible to generate simulation commands. In other words, the fault models are activated and their effects are observed, separately.

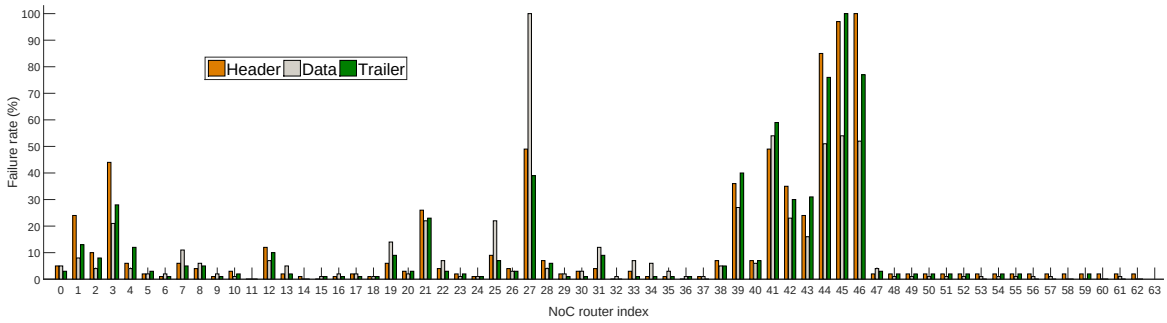


Figure 6.1: Header, data, tailor flit failure rate.

6.1.2 Experimental Results

One of the capabilities of TREAT is to report the percentage of header, data, and tailor flit loss for different level of granularities such as per router, per component, and even per signal. These values for each router are presented in Figure 6.1. The values in this figure are normalized to the maximum failed router. Different failure rate for header, data, and tailor flits are reported for each router under running FFT traffic through THENoC, in this experiment. The reason is that the source-destination pairs of routers are not uniformly selected in FFT traffic. For example, in this experiment, no router send packet to any of the routers with index number higher than 48.

TREAT is also able to trace the input fault mapping table to categorize failure percentage of all flit types caused by each injected of fault models. With this fault mapping table, it is able to report the effect of each category of injected faults on the 3D NoC design and its components. These results are presented in Figure 6.2, in which *SA* is Stuck-at, *OC* is Open-circuit, *BF* is Bit-flip, and *DE* is Delay-fault. As shown in this figure, the importance of Bit-flip and Stuck-at logic-level fault models are higher than Open-circuit and Delay-fault in terms of header, data, tailor flit loss rate. Although the reported experimental results depend on the running traffic pattern, but with this information a fault-tolerant designer is able to choose more

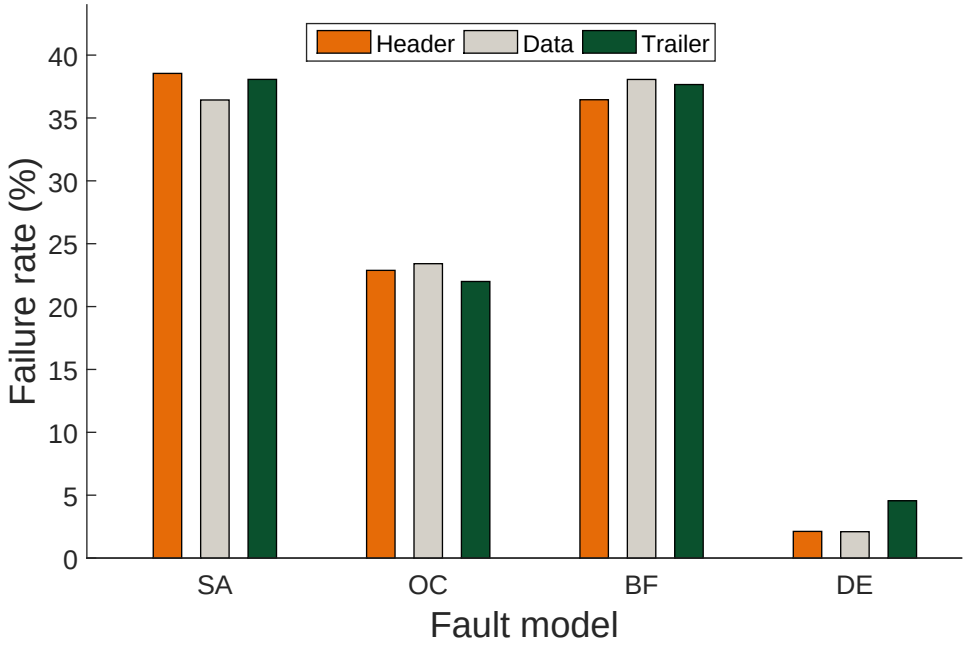


Figure 6.2: Failure rate contribution of each fault model.

efficient robust methods against major faults in some specific parts of the 3D NoC architecture.

The other capability of TREAT is to provide the MTBF for different level of granularities in 3D NoC design. Figure 6.3 reports the MTBF for major components of THENoC which are selected for the fault injection process. According to the experimental results, the MTBF of buffer component is considerably less than the other components, which shows the necessity of fault-tolerant design for this component.

6.2 Dynamic Fault Injection Parameters and Experimental Results

TREAT is also able to inject CTTC faults as dynamic fault modeling in 3D NoC environment [121]. In this section first, the accuracy of the proposed fault model is

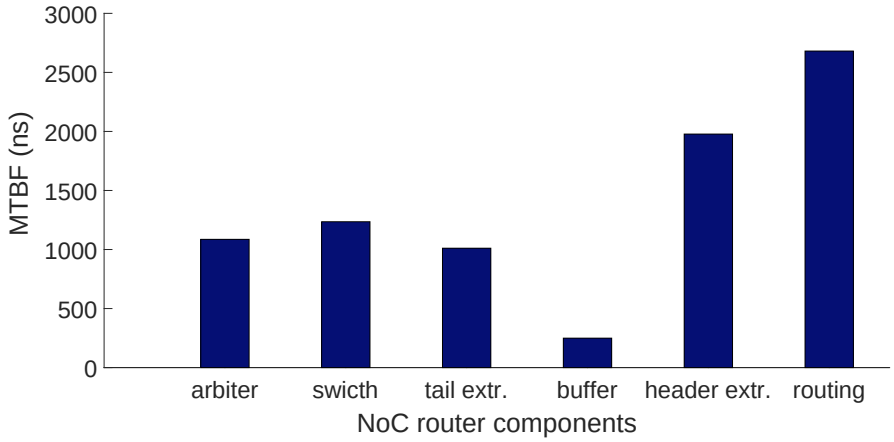


Figure 6.3: MTBF for each component of NoC router

evaluated. Then, the rate of TV occurrences by running PARSEC benchmark traces in THENoC through vertical links is reported. Finally, the density of CTTC effects over TV parameter in a matrix of TSVs for a specific synthetic traffic is illustrated as the other output of TREAT. These reports are presented to show the capabilities of TREAT for dynamic fault injection evaluation. They are all major parameters for 3D-NoC designers to assess their design's sensitivity against CTTC under various TSV physical parameters and operating conditions accurately.

The proposed CTTC fault model is implemented as a dynamic fault model inside TREAT which is called during the data transmission through TSVs at runtime to locate and trigger the parasitic capacitive for the specific configuration of TSVs with the given physical parameters, as elaborated in Chapter 5.

Prior to running the simulation, each configuration is run in HSPICE and the result is passed as a static configuration library to the TREAT. By extracting the circuit-level results from HSPICE (fail cases) and analyzing current direction matrix, the fault model detects the CTTC fault and triggers the corresponding TSV on-the-fly. The fault model operation is detailed in Chapter 5 and is illustrated in Figure 5.9b.

6.2.1 Fault model accuracy

In order to demonstrate the accuracy of the proposed fault model, a comparison is made with a crosstalk fault model [1]. These faults are activated with the uniform random distribution. The comparison shows that the accuracy of conventional fault model for TSVs is substantially insufficient.

First, TREAT is used to profile the number of fault occurrences for a TSV bundle using the proposed CTTC fault model under running random traffic. Then, using conventional fault model, the 3D-NoC simulation is run for 10000 iterations while in each experiment, faults are injected randomly across the TSV bundles. These simulations are repeated with different configurations, where a failing case is the result of parasitic capacitive with a minimum value of 8C, 7C, 6C, 5C, 4C, and 3C capacitive coupling values. In random fault injection, the reported probability of each case in Table 5.1 is employed as a parameter for random function generation process. For example, if 6C parasitic coupling results in failure based on the circuit-level simulation for a given configuration, the failure probability of a TSV in random simulation will be equal to the sum of occurrence probability of 8C, 7C, or 6C cases which is 10/81 (as reported in Table 5.1). For a fair comparison, the occurrence probability of capacitive coupling higher than 4C and 6C for TSVs on the corner and boundaries in a TSV bundle are also extracted.

Finally the results of these two set of experiments are compared to measure the reliability analysis inaccuracy of conventional random fault injection in modeling the CTTC effects in 3D NoC environment.

Figure 6.4 shows the inaccuracy introduced by random fault model distribution for different TSV configurations (leading to different failing cases). The inaccuracy of 0% means the random distribution matches the model's probability of occurrence,

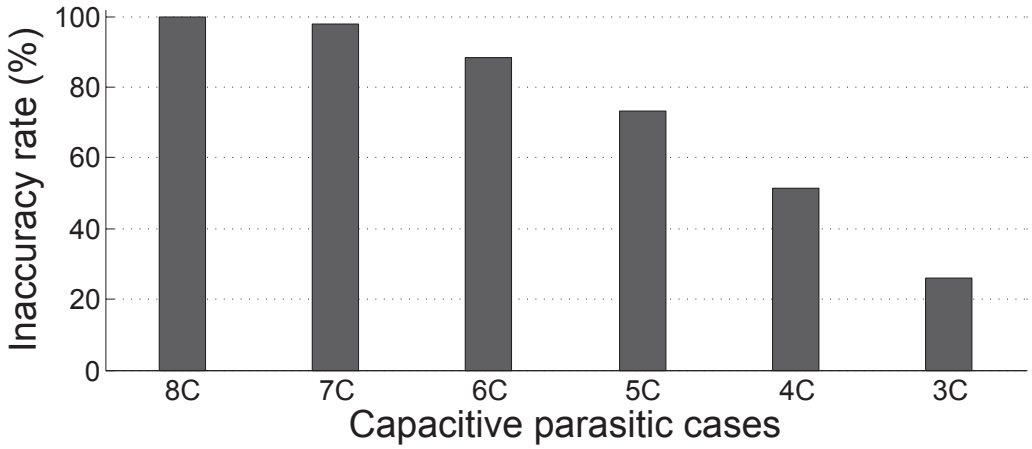


Figure 6.4: Inaccuracy introduced by random fault models

while 100% inaccuracy implies that the random distribution does not predict the corresponding fault type occurrence. It is observed that randomly distributed faults across the 8×8 TSV bundle introduces almost 99% inaccuracy for a given TSV specification leading to 8C capacitive coupling case. In other words, if the circuit-level simulations shows the 8C capacitive coupling case results in failure the random fault modeling is almost inaccurate for all the experiments. However, inaccuracy percentages decrease as the factor of failing cases of parasitic capacitive reduces. The reason is that the occurrence probability of parasitic capacitive with smaller values are higher than the ones with larger factors. Consequently, the percentage of inaccuracy decreases for smaller parasitic capacitive causing failure. However, even with these considerations the 26% inaccuracy for 3C parasitic capacitive value is still reported.

6.2.2 CTTC Fault Characterization

In order to demonstrate the effects of network traffic on CTTC fault, the PARSEC benchmark traces were collected using GEM5, and then injected them into THENoC which its specifications is summarized in Table 6.1. Figure 6.5 shows the ratio of TV occurrence over total number of data transmission in vertical links for three different

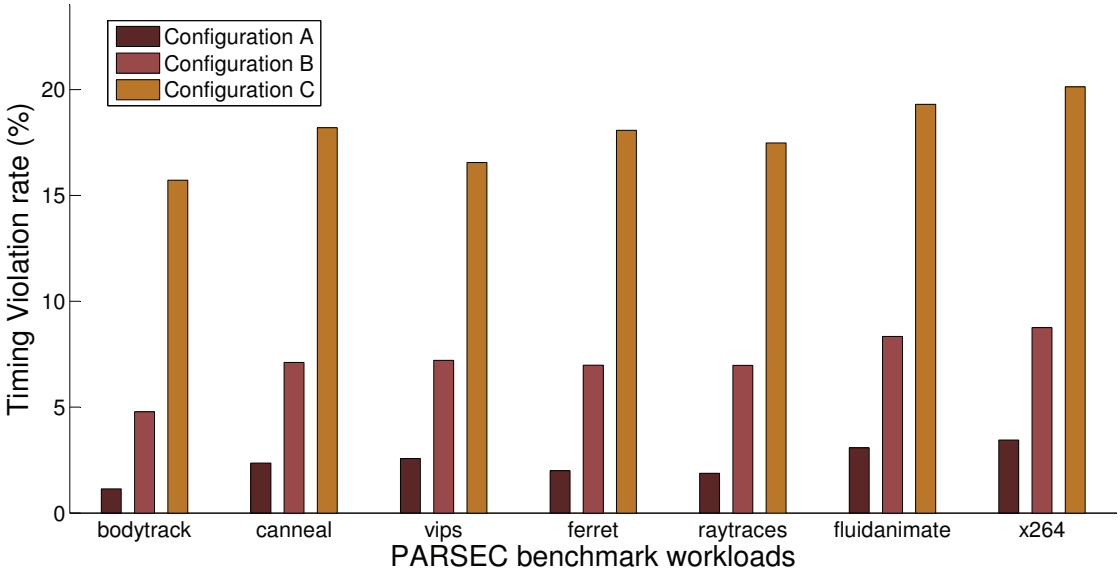


Figure 6.5: Fault model usage demonstration in 3D-NoC

configuration at 90% synthesis frequency. The reported experimental results in Figure 6.5 are based on running PARSEC benchmark for different conditions which are discussed in Table 5.2. For *Configuration C*, an average of 18% TV is observed because of running at higher temperature and low TSV oxide thickness both exacerbate the CTTC effects. As the configuration parameter values get relaxed, the TV due to CTTC also decreases accordingly.

Finally, CTTC fault density map for the $4 \times 4 \times 4$ simulated network with *Canneal* workload traffic is depicted in Figure 6.6. This figure illustrates 4×4 NoC routers of a specific layer sending data to their lower/upper layer routers (*Down port*/*Up port*). The values are normalized to the maximum number of TV in entire 3D-NoC. The layer 0 down port and layer 3 up port are not shown since they do not exist. With this map designers have proper estimation which TSVs for a specific application are more resilient and which of them are not. It can be seen that the data transmission from *layer 2* downward *layer 1* cause a large number of CTTC faults.

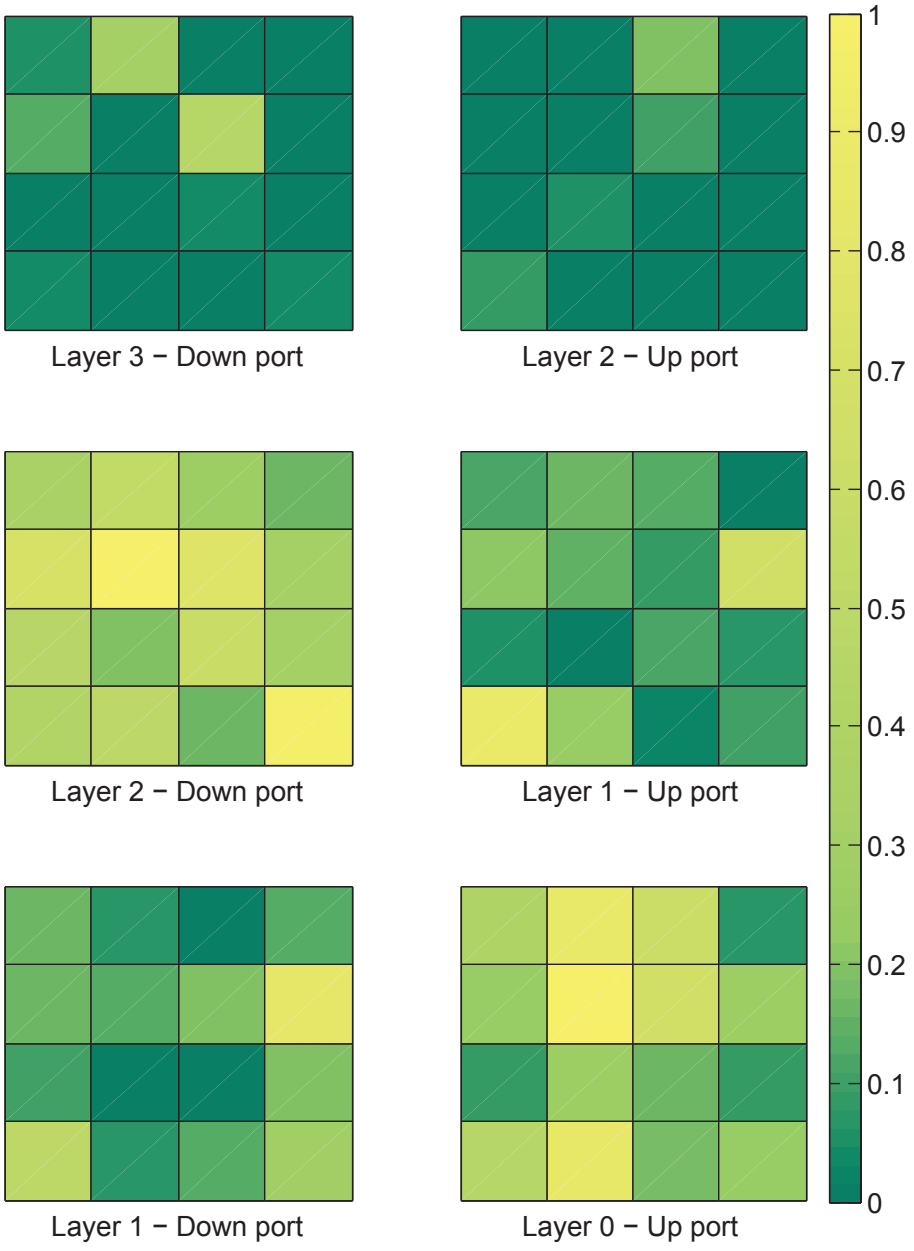


Figure 6.6: Fault density map

Chapter 7

Fault Mitigation Codings for TSV-based 3D ICs

As a side project of this dissertation two separate methods have been proposed to alleviate the unwanted effects of both capacitive and inductive TSV-to-TSV coupling issue in 3D ICs. These methods are called: Inductive TSV-to-TSV Coupling Mitigation (ITCM) [122] and TSV-to-TSV Capacitive Coupling Mitigation Algorithm (TCMA). A designer can choose either approach based on the operational frequency of the target system. Comparing to [60], the proposed method in this chapter does not include any information redundancy and the simulation results are based on circuit-level simulations.

7.1 Proposed TCMA Technique

The probability of 7C and 8C capacitive coupling emergence are reduced by applying by TCMA method in which the flits are adjusted before transmission. Mitigation is

chosen in this experiment since eliminating all 7C and 8C capacitive coupling imposes a complex architecture which is not scalable for any size of TSV meshes [60]. In this section, the proposed baseline TCMA is discussed first which can be used in smaller interconnections. Then the limitation of the baseline approach for larger interconnections is highlighted. Finally, the enhanced TCMA is presented which supports larger mesh of TSVs.

7.1.1 Baseline TCMA

The TTCC is data-dependent as described in Section 5.3. The basic idea of the baseline TCMA is to encode, if necessary, the consecutive data bits transmitting over the TSVs in order to mitigate the frequency of 7C and 8C capacitive coupling. This method does not limit any pattern of data transmission bits by encoding them before transmission and decoding them in receiver side, if needed. The inversion operation is chosen as a simple but light and efficient practical coding method in TCMA in order to keep the overhead low, while mitigating TTCC noise. In a mesh of TSVs, a single bit per row is needed in TCMA to determine whether the inversion process is needed or not at the receiver side. TCMA stores the last transmitted data bit of each TSV and compares it with the available data bit which has not been transmitted yet. The current direction matrix of all TSVs is generated by comparing these successive data bits as described in Section 5.3. Then the capacitive coupling for each of TSVs are calculated based on the current direction of its neighbor TSVs. Each row of 2D array of TSVs including 8C or 7C parasitic capacitance values is nominated for the data encoding process. By encoding the ready to transmit data bits, 8C capacitive coupling will be 2C and 7C capacitive coupling will be 1C or 3C in this method.

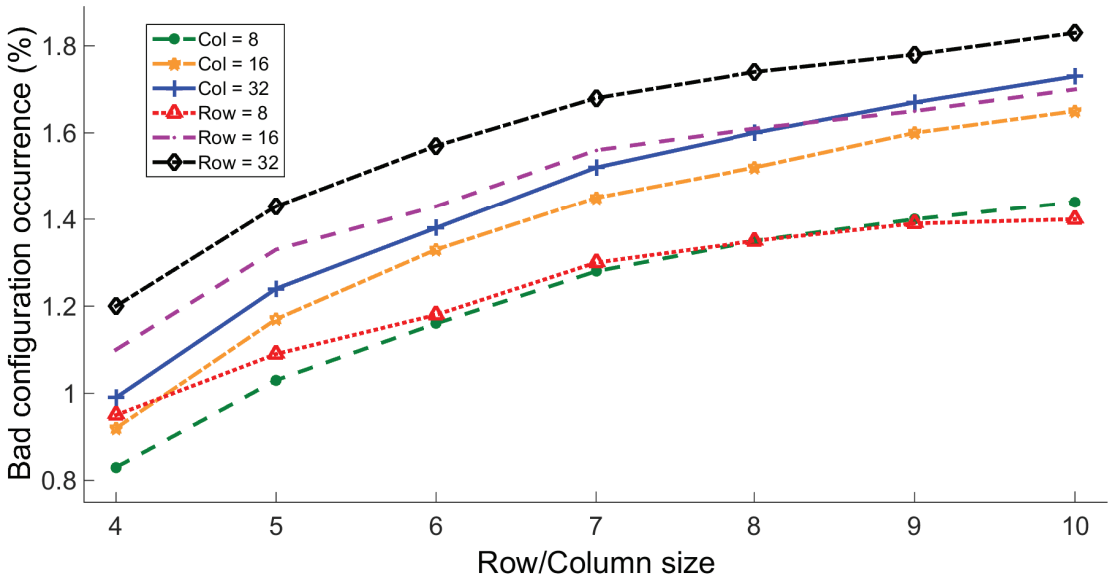


Figure 7.1: Probability of bad configuration occurrence

7.1.2 Enhanced TCMA

Although the baseline TCMA reduces the quantity of 8C and 7C parasitic capacitance values, but it may have some undesirable side effects by converting a row of data bits. For some special data patterns, converting a single row of data bits may generate unexpected 8C or 7C parasitic capacitance values, which happens in a mesh of TSV with more than 3 rows or 6 columns. These special cases are called bad configuration in the rest of this chapter.

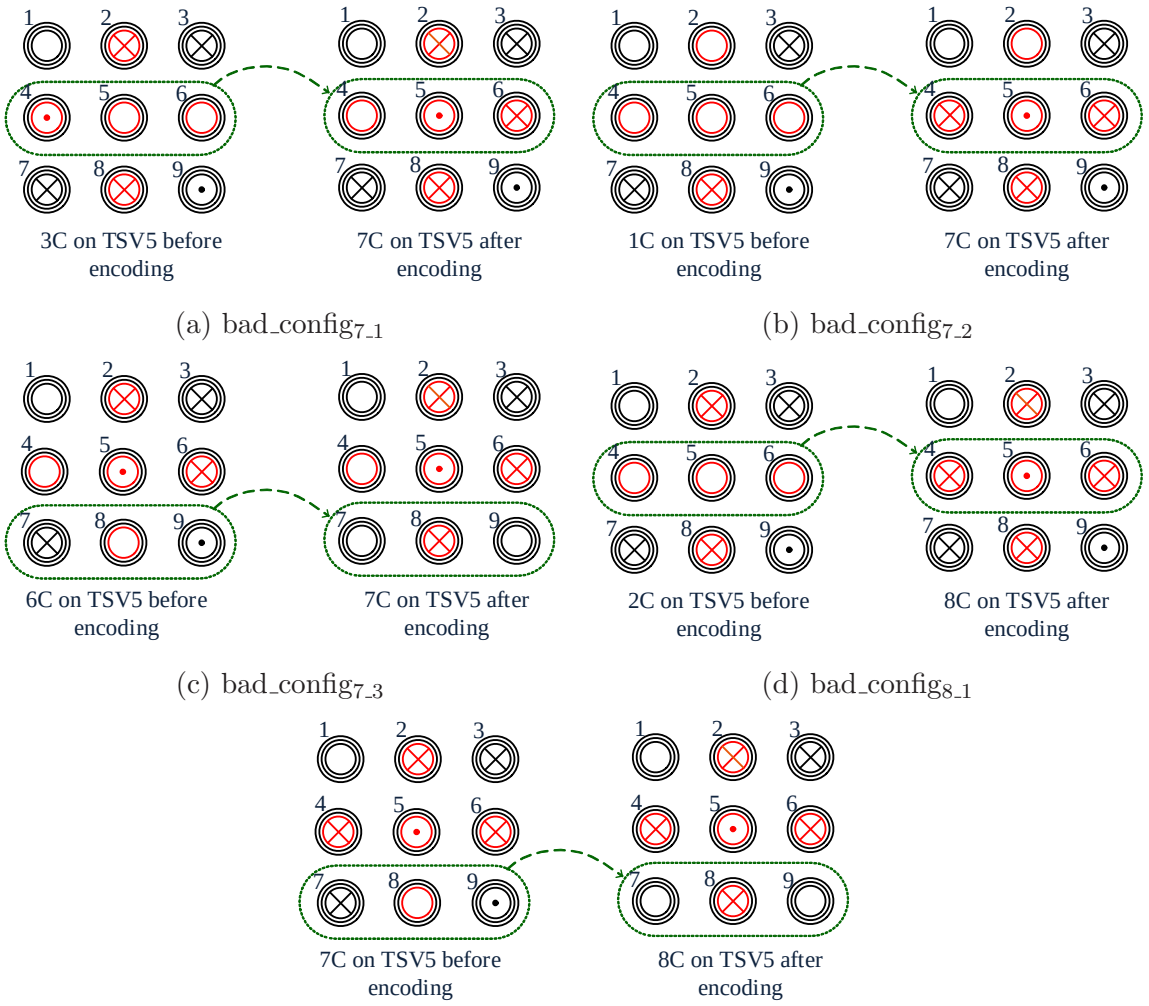
A bad configuration is a subset of TSV mesh which potentially generates unexpected 8C or 7C parasitic capacitance values by converting a single row of data bits. In more details, the row encoding affects the other data bits of the same row or the data bits in predecessor or successor rows in 2D matrix of TSVs. Since the probability of bad configuration occurrence is low, specially for smaller matrices of TSVs, the baseline coding is still efficient for smaller data buses (less than 64 bits) which are considered in 3D Network-on-Chip (3D NoC applications). Figure 7.1 shows the probability of bad configuration occurrences in different mesh size of TSVs.

Table 7.1: Current flow of tsvs before and after encoding

Sent data	**Ready to send data**	CF_{bi}	CF_{ai}
0	0	○	⊙
0	1	⊙	○
1	0	⊗	○
1	1	○	⊗

This experiment is done by running the Monte Carlo simulations for 10000 iterations for different row/column dimensions. According to experimental results the reported percentage of bad configuration for all of the experimented dimensions is less than 2%. The baseline coding is not scalable for larger data buses (more than 64 bits) which are applied in 3D memory applications according to the increasing trend in Figure 7.1. The enhanced version of TCMA is devised for these sorts of application to make sure the encoding process of a selected data bit of TSVs does not worsen the total capacitive coupling. First, the bad configuration concept is explored in detail and then the proposed solution is presented.

Table 7.1 summarizes the TSV current flow direction before and after encoding its ready to send data bit. CF_{bi} shows the current flow of TSV before inverting the ready to send data, while CF_{ai} represents the current flow of TSV after inversion. Based on this table an inactive TSV current flow (○) may convert to active TSV (either ⊙ or ⊗), while an active TSV (either of the ⊙ or ⊗) is converted into an inactive one (○) after inverting the ready to send data bits. Based on the experimental result analysis, a bad configuration occurs in five cases, while two of them are potential to generate unwanted 8C capacitive coupling and the other three may generate unwanted 7C capacitive coupling. They are called bad_config_{8_1}, bad_config_{8_2}, bad_config_{7_1}, bad_config_{7_2}, and bad_config_{7_3}. Figure 7.2 illustrates these five cases in top view of 2D array of TSVs in a 3x3 mesh of TSVs. The candidate row for inversion is recognized by dashed lines in this figure. It also shows the capacitive coupling value of middle TSV in the recognized row by dashed lines before and af-



(a) bad_config$_{7_1}$ (b) bad_config$_{7_2}$

(c) bad_config$_{7_3}$ (d) bad_config$_{8_1}$

(e) bad_config$_{8_2}$

Figure 7.2: Potential configurations to generate 7C and 8C capacitive coupling

ter encoding process. Each of these bad configurations affects the result of baseline coding with some conditions which are discussed in the following.

In the baseline method and in case of encoding, the 3C capacitive coupling, if any, is converted into 7C (see Figure 7.2a) by encoding the second row of 2D array of TSVs with following four conditions:

- There are exactly two inactive TSV next to each other in potential row for encoding process as in TSV5 and TSV6.

- TSV2 and TSV8 are active with the same current direction.
- The current direction of TSV6 after encoding should be the same as the current direction of TSV2 and TSV8.
- The current direction of TSV5 should be reverse of the current direction in TSV2, TSV6, and TSV8 after encoding.

The 1C capacitive coupling is converted into 7C (see Figure 7.2b) by encoding the second row of 2D array of TSVs with following four conditions:

- There are at least three inactive TSV next to each other in potential row for encoding process as in TSV4, TSV5, and TSV6.
- Either of TSV2 or TSV8 is inactive and the other should be active.
- The current direction of TSV4 and TSV6 after encoding should be the same as the current direction of either TSV2 or TSV8 which was active.
- The current direction of TSV5 after encoding should be reverse of the current direction of TSV4, TSV6, and either TSV2 or TSV8 which was active.

The 6C capacitive coupling is converted into 7C (see Figure 7.2c) by encoding the third row of the 2D array of TSVs with following four conditions:

- In capacitive matrix there is a 6C capacitive coupling in predecessor row which is selected for encoding in a way that TSV5 has reverse current direction of TSV2 and either of TSV4 or TSV6.
- TSV8 which is in the nominated row for encoding is inactive.
- One of TSV4 or TSV6 is inactive and the other should should be active with reverse current direction of TSV5.

- The current direction of TSV8 after encoding should be same as current direction of TSV2 and either of TSV4 or TSV6 which was active.

The 2C capacitive coupling is converted into 8C (see Figure 7.2d) by encoding the second row of 2D array of TSVs with following four conditions:

- There are at least three inactive TSVs beside each other in potential row for encoding process like TSV4, TSV5, and TSV6.
- TSV2 and TSV8 are active with same current direction.
- The current direction of TSV4 and TSV6 after encoding should be the same as the current direction of TSV2 and TSV8.
- The current direction of TSV5 should be reverse of the current direction in TSV2, TSV4, TSV6, and TSV8 after encoding.

The 7C capacitive coupling is converted into 8C (see Figure 7.2e) by encoding the third row of 2D array TSV, if the following conditions are satisfied:

- In capacitive matrix there is a 7C capacitive coupling in predecessor row which is selected for encoding. The inactive TSV should be also in the selected row for encoding.
- TSV8 has the reverse current direction of TSV5 after encoding.

The probability of bad configuration presence in a mesh of TSVs is very low since all the discussed conditions should be satisfied simultaneously. However, the goal of the enhanced TCMA, which is summarized in Algorithm 1 is to guarantee the encoding process will not worsen the total number of 7C and 8C capacitive coupling in a 2D array of TSVs. In the enhanced version of TCMA the encoding process will be done

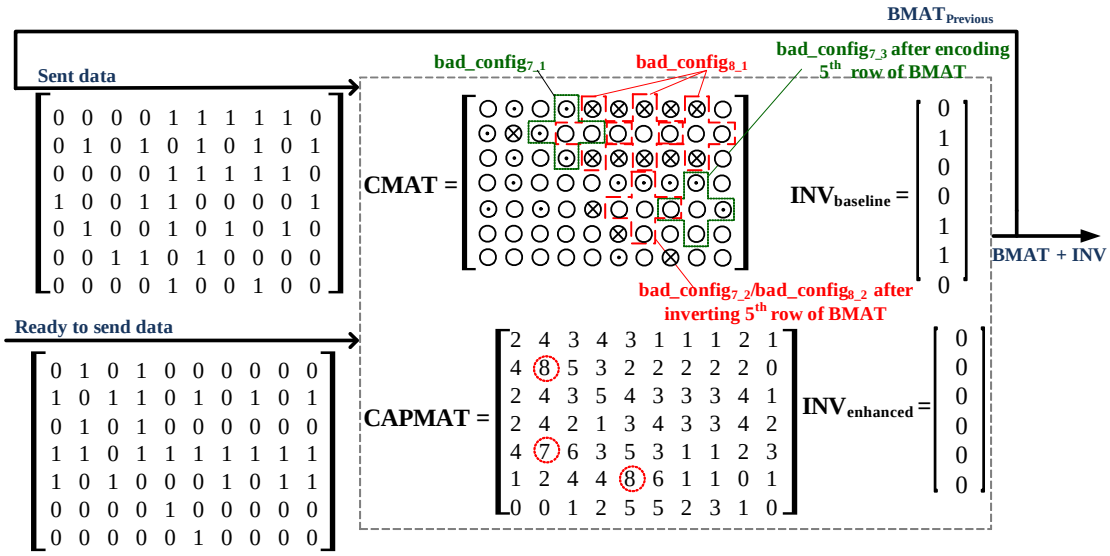
Algorithm 1 Enhanced TCMA coding algorithm

AMAT ← Sent data bits
BMAT ← To be sent data bits
CMAT ← Current direction of each TSV generated by AMAT & BMAT
CAPMAT ← Capacitive parasitic noise of each TSV generated by CMAT
INV ← Redundant vector for inversion process decision at receiver side
for each $R \in Rows$ **do**
    for each $C \in Columns$ **do**
        if CAPMAT[R][C] == 8 or CAPMAT[R][C] == 7 **then**
            $78C\_counter++$
        end if
        if (there is a bad configuration bad_config_{7_1} or bad_config_{7_2} or bad_config_{7_3}) **then**
            $bad\_config_7\_counter++$
        end if
        if (there is a bad_config_{8_1} or bad_config_{8_2}) **then**
            $bad\_config_8\_counter++$
        end if
    end for
    if $(78C\_counter > bad\_config_7\_counter + bad\_config_8\_counter)$ **then**
        Encode the BMAT[R]
        INV[R]=1
    end if
end for

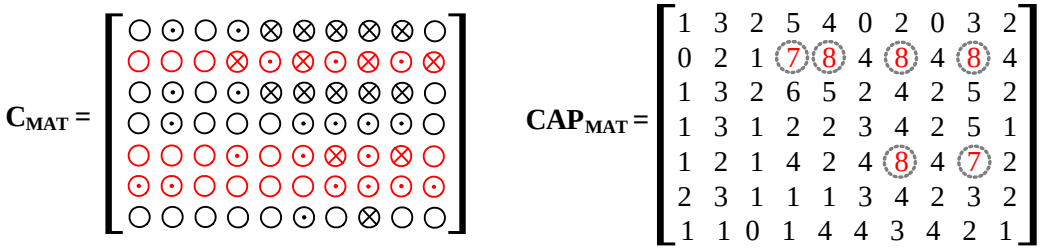
if the total number of 7C and 8C capacitive coupling in capacitive matrix is higher than the total number of bad configuration in each row.

7.1.3 TCMA Elaboration and Evaluation

Figure 7.3a illustrates an example of the baseline and enhanced algorithm for 7×10 given AMAT and BMAT matrices. These matrices and the ones which are used in following sentences are defined in Algorithm 1. This dimension has been chosen to show the advantages of the enhanced approach over the baseline technique for higher bandwidth data buses. First, CMAT and then CAPMAT matrices are generated form the sent (AMAT) and not sent yet (BMAT) data lines. The current flow of each TSV is presented with the same method as discussed in Section 5.3. Then, CAPMAT is generated from CMAT by counting the total mutual capacitive parasitic difference between each TSV and its adjacent neighbors. The INV matrix is evaluated in the receiver side to extract the original data values if they are encoded. $\text{INV}_{baseline}$ of



(a) Example of baseline and enhanced TCMA



(b) CMAT and CAPMAT after baseline TCMA

Figure 7.3: An example that shows baseline algorithm issues

this example shows that the second, fifth, and sixth rows of the BMAT matrix have been encoded since there are 8C or 7C parasitic capacitance values in these rows of CAPMAT matrix. Since the number of 7C and 8C capacitive coupling are not higher than the number of bad configuration in enhanced method, the $INV_{enhanced}$ shows none of the rows the BMAT has been encoded.

Figure 7.3b represents the updated CMAT and CAPMAT matrices in the baseline approach after encoding the second, fifth, and sixth rows of BMAT matrix in which the total number of 7C and 8C capacitive coupling increases from 3 to 6. This example illustrates all 5 possible bad configurations. The $bad\_config_{7\_1}$ and $bad\_config_{8\_1}$ are depicted in second row of CMAT in Figure 7.3a, resulting in three 8C and one 7C

after encoding second row of BMAT. The bad_config$_{7_2}$ of fifth row is highlighted in CMAT matrix of Figure 7.3a. After encoding fifth row of BMAT, the undesirable 7C will be generated in CAPMAT[5][7], which is also bad_config$_{8_2}$. Furthermore, encoding fifth row of BMAT generates a bad_config$_{7_3}$ in CAMPMAT[5][9].

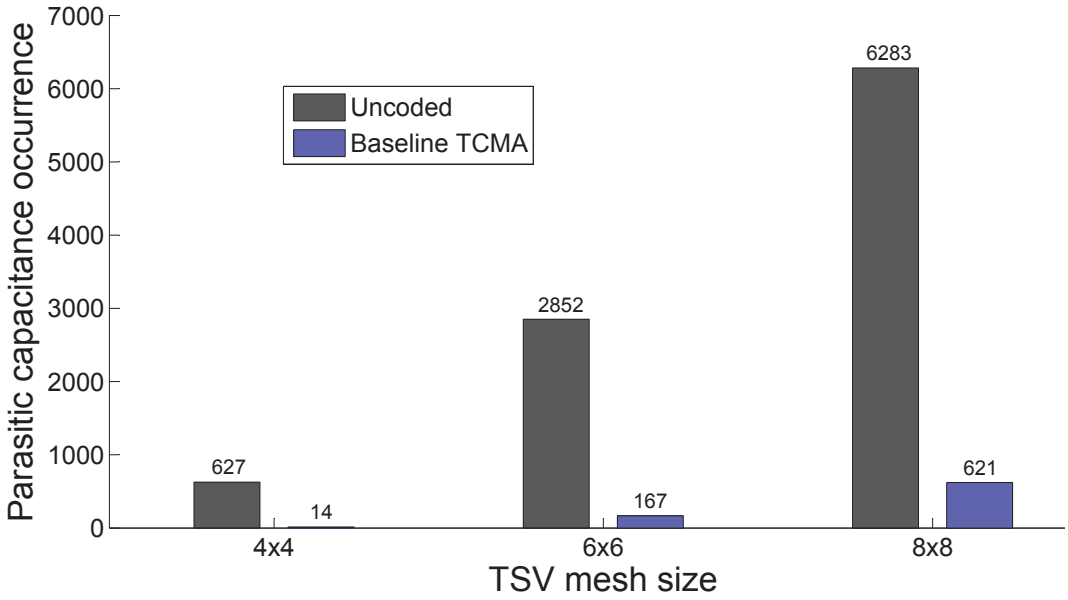
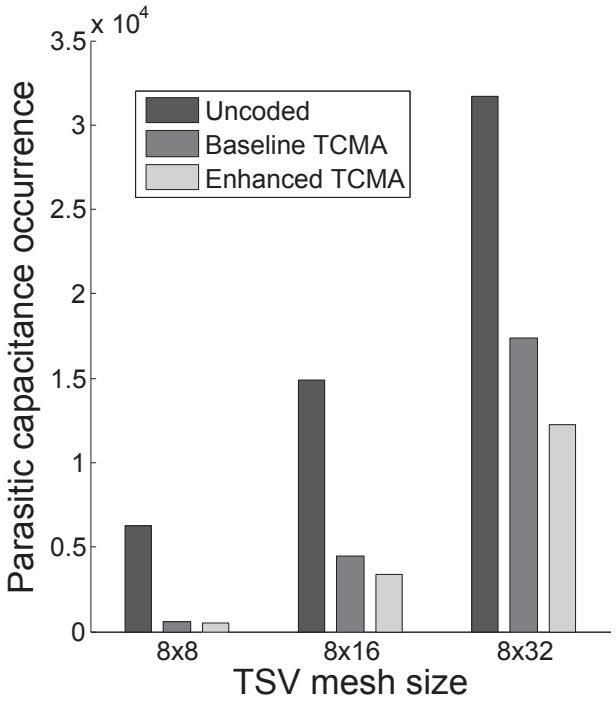
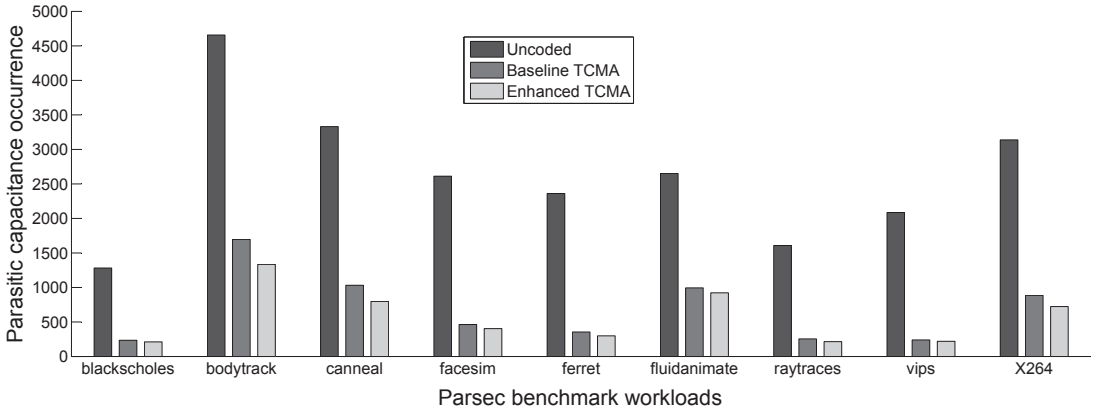


Figure 7.4: Number of 7C/8C for random data bit patterns in small mesh of TSVs

Since the encoding decision is supposed to be done row by row in one direction (from top to bottom in this example) or reverse, the unwanted generated 7C and 6C in fifth row of CAPMAT are potential to generate 8C and 7C, respectively by encoding the sixth row of BMAT. Due to the presence of 8C in sixth row of CAPMAT, it is selected for encoding process and both of bad_config$_{7_3}$ and bad_config$_{8_2}$ generate undesirable 8C and 7C in fifth row of CAPMAT which is shown in Figure 7.3b. However, the enhanced algorithm prevents all of these bad effects by predicting them.



(a) Random data bit patterns in larger mesh of TSVs



(b) PARSEC application data bit patterns in 8×32 mesh of TSVs

Figure 7.5: 7C and 8C capacitive coupling for random and PARSEC applications

To evaluate the advantages of the baseline TCMA for smaller mesh size, Monte Carlo simulations for 10000 iterations on different sizes of TSV mesh are examined. The total number of 7C and 8C capacitive coupling before and after applying the baseline TCMA for different mesh size of TSVs is shown in Figure 7.4. It is depicted that the mitigation rate of 7C and 8C capacitive coupling after applying the baseline

TCMA are almost 98%, 94%, and 90% for 4×4, 6×6, and 8×8 mesh of TSVs. The information redundancy of the baseline TCMA method for these sizes of mesh of TSVs are 25%, 16%, and 12%, respectively. However, the mitigation rate of the baseline TCMA is increased for large mesh of TSVs, as expected. This is because of the probability of bad configuration occurrence rises by increasing the sizes of TSV meshes. The Monte Carlo simulations for 10000 iterations for larger mesh of TSVs are also examined for both baseline and enhanced TCMA to show the advantages of enhanced TCMA. Although the mitigation rate of total number of 7C and 8C capacitive coupling values is increasing by using larger mesh of TSVs, enhanced TCMA prevents encoding process if the result is worsen. This is shown in Figure 7.5a, in which the mitigation rate of 7C and 8C capacitive coupling occurrence by applying enhanced TCMA are always higher than baseline approach.

PARSEC benchmark [10] as a realistic data traffic for large size of mesh of TSVs are also applied to check the performance of the baseline and enhanced TCMA. Memory traces of PARSEC applications have been employed in this experiment, which are extracted by the PIN tool [43], a dynamic binary instrumentation framework for the IA-32 and x86-64 instruction-set architectures. The total number of 7C and 8C parasitic capacitance values for memory traces of PARSEC application workloads through the TSVs are reported for a 8×32 mesh of TSVs in Figure 7.5b. The migration rate of TCMA for Blackscholes, Facesim, Vips, and Raytraces are between 80% to 90% and for the rest of them is almost 70%. Although the differences between the mitigation rates of baseline and enhanced TCMA are not very much, but the result of enhanced method is always better than baseline as it is expected. In other words, it is always guaranteed that by applying the enhanced TCMA the total number of 7C and 8C capacitive coupling will never be worse off because of the bad configuration presence.

In order to evaluate the proposed coding methods, the baseline and enhanced TCMA en-

Table 7.2: Hardware synthesize results

Mesh size	Baseline		Enhanced	
	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)
8×8	918	2340	1096	3000
8×16	1818	4520	2173	5900
16×8	2094	5260	2165	5880
8×32	3321	8840	4331	11700
32×8	4086	11000	4323	11700

coders are implemented in Verilog and synthesized by Synopsys Design Compiler using 28nm TSMC library (1.05V, 25 °C). Table 7.2 reports the synthesis results as representing power consumption and occupied area. The latency of the enhanced method is reported by the critical path including: registers latching the adjusted output data bits toward the feedback input for subsequent CMAT computation. In other words, it does not depend on the dimension of TSV arrays. According to the logic synthesis, the latency of the baseline and enhanced TCMA are reported as 69.5ps and 74.9ps for all the given TSV dimensions in Table 7.2. The feasibility of both proposed coding algorithms are confirmed by considering the obtained coupled capacitive coupling mitigation and its tangible footprint and power consumption. Decoder units are not implemented in this experiment, since they are only composed of a comparator and a mix of inverter gates. They are much lighter than encoder components in terms of area, power consumption, and latency.

7.2 Inductive TSV-to-TSV Coupling Analysis and Mitigation

Similar to the presented analysis of CTTC effect, the effect of Inductive TSV-to-TSV Coupling (ITTC) is also reported in this section first and then the proposed ITCM

technique is presented which is able to mitigate the undesirable effect fo ITTC effects.

7.2.1 ITTC Characteristics

Although the capacitive coupling is a well known source of noise in TSV-based 3D ICs, inductive coupling has also some undesirable effects like overshooting and supply/ground bounces. This is because inductive coupling generates high amplitude noise pulses with short durations comparing to capacitive coupling.

In order to evaluate the effect of ITTC, a 3×3 array of TSVs is modeled in HSPICE, in which the middle TSV is considered to be the victim.

In simulations, the top end of each TSV is connected to the output of an inverter, which drives the input of another inverter connected to the bottom end of the TSV. Similar to CTTC evaluation, PTM library are used to implement inverters and circuit-level model of TSVs are both simulated by Synopsys HSPICE tool.

The worst-case induced voltage on the victim TSV is reported as regression lines for different TSV radii (Figure 7.6a), process technologies (Figure 7.6b), and TSV lengths (Figure 7.6c) over different frequencies. As TSVs become longer the magnetic flux linking the two TSVs increases proportionally. Thus the coupling voltage between victim and aggressor TSVs and hence the total inductive coupled voltage increase as the length of the TSVs increase(see Figure 7.6c).

Although the linkage flux between two TSVs is a strong function of the length, its dependence on the TSV radius is very weak. Changing the radius of cylindrical TSVs affects mutual inductance by changing the magnetic field because of TSV aggressors and the exposed surface to the linkage flux. As long as the current distribution in a TSV remains almost symmetric, assuming proximity effect and other high order

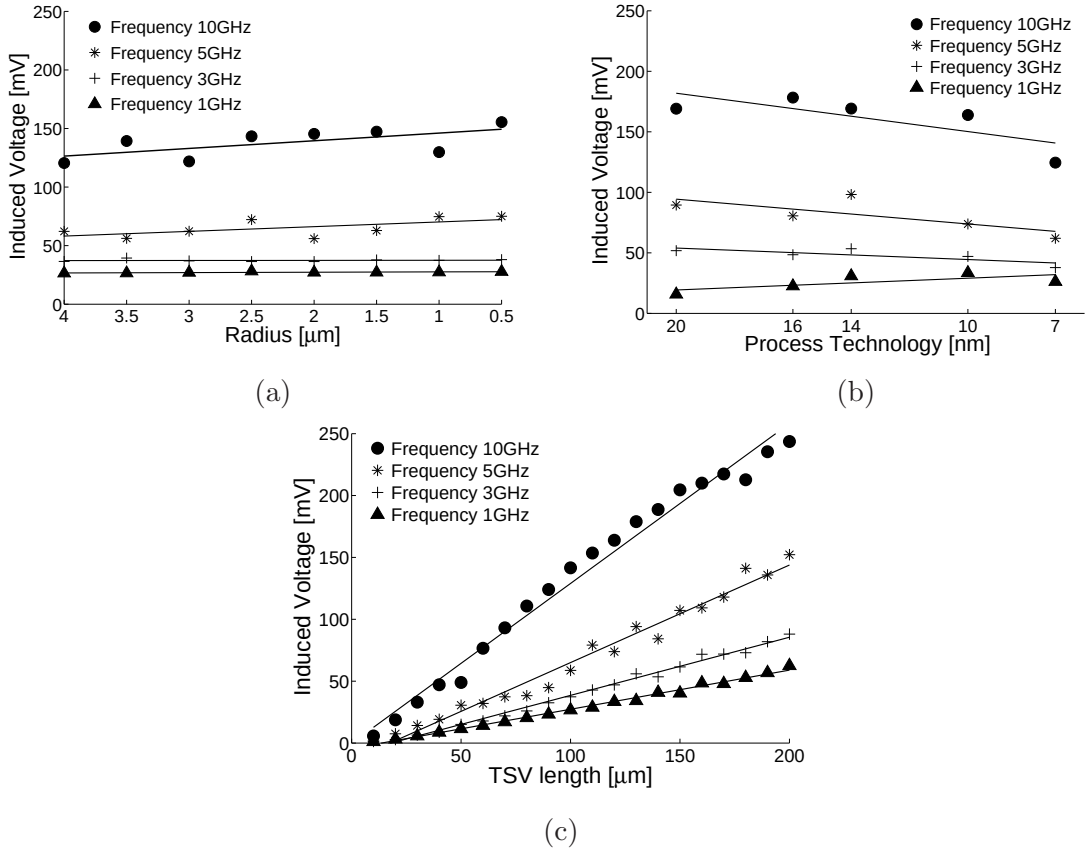


Figure 7.6: Characterizing inductive coupling against various parameters

effects are trivial, and the magnetic field created by an aggressor is almost constant. In other words, the TSV radius changes effect on the magnetic field of TSVs is not critical. Since the length of the TSV is at least an order of magnitude longer than its radius, the second effect is small, but the linkage flux and the mutual coupling decreases slightly as the radius increases (see Figure 7.6a).

As shown in Figure 7.6b, induced voltage is a weak function of the process technology. As processes scale down, the gate capacitance becomes smaller while the rise time and fall time of the voltages become shorter. Overall the charging and discharging current of gate capacitance remains almost constant. The same current that charges (or discharges) the gate capacitance passes through TSV and causes the inductive coupling to neighboring TSVs. Inductively coupled voltage remains almost constant

Types[1]	0L	|1|L	|2|L	|3|L	|4|L
Sample pattern	⊙ ⊗⊕⊗ ⊙	⊙ ⊙⊕⊗ ○	⊙ ⊙⊕⊙ ⊗	⊙ ⊙⊕○ ⊙	⊙ ⊙⊕⊙ ⊙
Occurrence frequency	19	32	20	8	2
Occurrence probability	0.23	0.39	0.25	0.1	0.03

[1] The absolute values are exactly same as α values in Equation 7.1

Table 7.3: ITTC categorization with occurrence probability of each type

for different technologies. As technology advances and supply voltage shrinks, the coupled voltage becomes a greater portion of the V_{dd} and increases the probability of error. Consequently, the length of TSVs has a major impact on inductive coupling, resulting in unexpected noises in 3D NoC, as the number of layers increases.

If the inductive coupling voltage caused by a single horizontal or vertical neighboring TSV is β, then the total inductively coupled voltage on a victim TSV ,$V_{Icoupl_{tot}}$, is proportional to $\alpha\beta$, where the value of integer α depends on the current flow direction and arrangement of active neighboring TSVs, as will be shown shortly. Assuming the electromagnetic proximity effect and other high-order effects are negligible, $V_{Icoupl_{tot}}$ is equal to sum of the voltages induced by each aggressor TSV, based on Faraday's law:

$$V_{Icoupl_{tot}} = \sum_{i=1}^{N} V_{Icoupl_{\mathrm{i}}} = \sum_{i=1}^{N} M_{\mathrm{v,i}} \frac{dI_i}{dt} \sim \alpha\beta \tag{7.1}$$

where N is the total number of aggressors, $V_{Icoupl_{\mathrm{i}}}$ is the voltage coupled on the victim by i^{th} aggressor, assuming all other aggressors have constant current. $M_{\mathrm{v,i}}$ is the mutual inductance between i^{th} aggressor and victim TSVs. I_i is the current of i^{th} aggressor TSV. The $M_{\mathrm{v,i}}$ is found from Equation 7.2 [107].

$$M_{\mathrm{v,i}} = \frac{\mu_0}{2\pi}\left[l\ln\left(\frac{l+\sqrt{{d_i}^2+l^2}}{d_i}\right) + d_i - \sqrt{{d_i}^2+l^2}\right] \tag{7.2}$$

where d_i is the distance of i^{th} aggressor from the victim TSV and l is the length of a TSV.

Since the effect of inductive coupling crosstalk caused by diagonal neighboring TSVs is less than that caused by adjacent TSVs, they are not considered in this analysis. With this assumption an ITTC is classified into five different classes based on the number and arrangement of active neighbors for a victim TSV. This classification is based on the absolute value of α for a given victim TSV caused by active neighbor TSVs. Each victim TSV has four neighbors in horizontal or vertical directions. These neighboring TSVs can be either active or inactive. An active TSV may have upward or downward current directions, while there is no current flow in an inactive TSV. As a result, there are $3^4 = 81$ possible configurations of TSV neighbors for a victim TSV. Since many of these 81 arrangements have similar $|\alpha|$ values, the behavior of all 81 configurations is summarized in Table 7.3. The absolute value of α is reported here since the sign of α is chosen by the direction of its neighbor TSVs' current direction which are all similar and symmetric. In other words, the same effect occurs if all adjacent neighbor TSVs of a victim TSV have upward or downward current flow. Based on this classification, there are 5 different types of ITTC as shown in the first row of Table 7.3, which are quantified in terms of their frequency and probability of occurrence. The $\odot$, $\otimes$ symbols in Table 7.3 represent the upward and downward TSV current flow directions, respectively. The $\oplus$ symbol represents a victim TSV regardless of its current direction which does not impact the proposed analysis.

A sample pattern resulting in each parasitic inductance in range of 0L to |4|L is also shown in Table 7.3. The maximum $V_{Icoupl_{tot}}$ in this representation is equal to |4|, if all adjacent neighboring TSVs have the same current flow, while it will be 0 if each pair of neighboring TSVs has reverse current flow or if all the adjacent neighboring TSVs are inactive. The frequency and probability of occurrence for each presented

types of ITTC are also reported in this table.

7.2.2 Proposed ITCM Technique

Shield insertion on a chip [97] and TSV placement with a safe distance [50] approaches have been proposed to mitigate mutual coupling, but they are not efficient. Similar to TCMA method the sequence of bits of each flits are adjusted in the proposed ITCM technique. This goal is not achievable for all data patterns due to intrinsic randomness property of the data, therefore the purpose of the proposed coding method is to rearrange the data bits to replace the majority of larger total inductive coupling values by smaller ones. As a practical approach for data-bit adjustments, the inversion operation is chosen. Data bit inversion is done at the receiver side to retrieve the original data-bit patterns. The overhead of this method is the extra bits, to decide the inversion process at the receiver side. Considering one bit for each of the data bits imposes a 100% information redundancy, in the proposed algorithm the inversion operation is performed for each row with a single bit overhead.

Algorithm 2 summarizes the proposed inductive coupling mitigation coding method for positive α values. There is a similar algorithm for negative α values, in which all the $IMAT[i][j] = BMAT[i][j]$ assignments from line 10 to line 50 of Algorithm 2 are replaced by $IMAT[i][j] = \overline{BMAT}[i][j]$. AMAT and BMAT input matrices are the two consecutive data flits, which AMAT has been sent while BMAT is planned to be sent. The proposed algorithm is able to invert the values of BMAT matrix based on the values of AMAT matrix, as shown in Algorithm 2. In this algorithm, first the current flow direction of each TSV is calculated by considering the data-bit pattern of the corresponding elements in AMAT and BMAT, resulting in matrix CMAT. Then, PMAT (α values for each TSV) is generated by adding the values of its corresponding

Algorithm 2 ITCM algorithm

```
AMAT ← Sent data bits
BMAT ← To be sent data bits
CMAT ← Current flow direction of each TSV
PMAT ← α values in Equation 7.1 for each TSV
IMAT ← Inversion intention for each TSV
for each R ∈ Rows do
    for each C ∈ Columns do
        UP ← CMAT[R-1][C]
        DOWN ← CMAT[R+1][C]
        if PMAT[R][C] == 1 then
            if UP == 0 then
                IMAT[R-1][C] = BMAT[R-1][C]
            else
                IMAT[R-1][C] = 1
            end if
        else if PMAT[R][C] == 2 then
            if UP == −1 or DOWN == −1 then
                if UP == −1 then
                    IMAT[R-1][C] = 1
                else
                    IMAT[R+1][C] = 1
                end if
            else if UP == 1 and DOWN == 1 then
                IMAT[R-1][C] = 1
                IMAT[R+1][C] = 1
            else if UP == 0 and DOWN == 0 then
                IMAT[R-1][C] = BMAT[R-1][C]
                IMAT[R+1][C] = BMAT[R+1][C]
            else if UP == 1 and DOWN == 0 then
                IMAT[R-1][C] = 1
                IMAT[R+1][C] = BMAT[R+1][C]
            else if UP == 0 and DOWN == 1 then
                IMAT[R-1][C] = BMAT[R-1][C]
                IMAT[R+1][C] = 1
            end if
        else if PMAT[R][C] == 3 then
            if UP == 1 and DOWN == 1 then
                IMAT[R-1][C] == 1
                IMAT[R+1][C] == 1
            else if UP == 0 then
                IMAT[R+1][C] == 1
                IMAT[R-1][C] == BMAT[R-1][C]
            else
                IMAT[R-1][C] == 1
                IMAT[R+1][C] == BMAT[R+1][C]
            end if
        else if PMAT[R][C] == 4 then
            IMAT[R-1][C] == 1
            IMAT[R+1][C] == 1
        end if
    end for
end for
for each Row ∈ IMAT do
    Decides whether the inversion is needed or not
end for
```

vertical and horizontal neighbors in CMAT based on Equation 7.1. There is a chance of row inversion for each row, if conditions of the proposed algorithm are satisfied, as shown in Algorithm 2 with the goal of replacing majority of larger values in PMAT by smaller ones.

In the proposed method, each TSV votes for inversion of its vertical neighbors to create IMAT. The decision of inverting data pattern of adjacent rows in IMAT matrix may conflict each other, as inversion of a single row of IMAT matrix is suggested for one of its adjacent row while it is not recommended for the other one. The proposed coding algorithm addresses these dependencies and takes the best decision considering the optimization of each row and their effect on the data. In this case, the net benefit of inversion of each row should be calculated to find the best inverting decision for different rows of IMAT matrix. There is a trade-off between the degree of inductive coupling mitigation and performance/complexity of the design. The proposed algorithm ignores the inverting process, if there is any conflict of decision in inverting adjacent rows of IMAT matrix in order to save the timing constraints. Finally, based on the number of inversion requests in each row the inversion decision is made. It should also be noted that the encoding and decoding processes are only needed at source and destination Network Interfaces (NI), also since NoC routers require the header information, header flit is not encoded; B-bit (B shows the number of rows) inversion indication bits are added to the header flit of the packet to prevent imposing extra TSVs.

Figure 7.7 illustrates an example of the proposed algorithm for given AMAT and BMAT as consecutive flits through a 4x4 mesh of TSV. Any changes from '0' to '1' in the corresponding elements of AMAT and BMAT matrices is represented as 1 in CMAT, while the reverse changes is shown by -1. A given element of CMAT will be 0 if it corresponding element in AMAT and BMAT are the same. Then

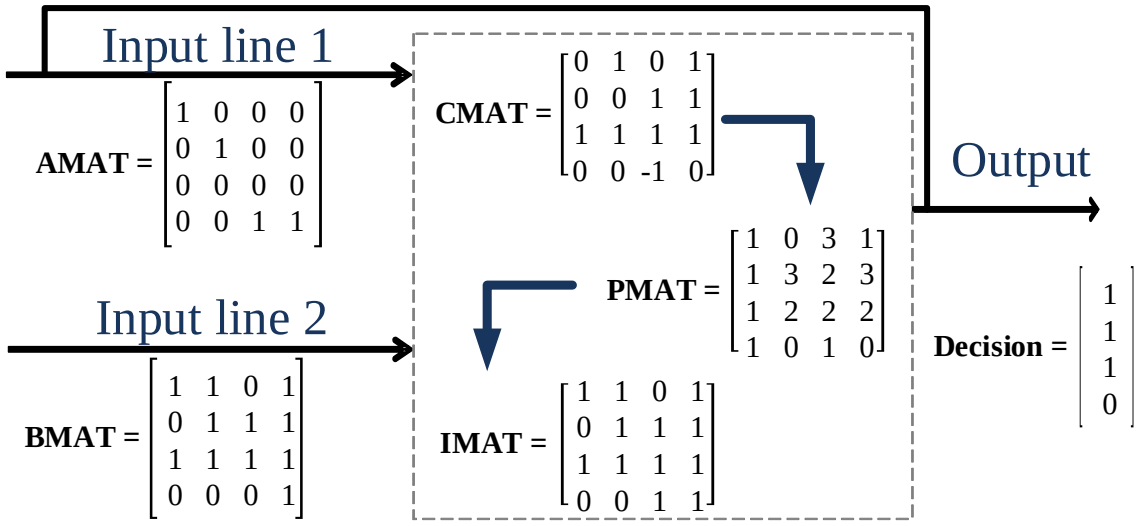


Figure 7.7: 4x4 mesh of TSV example of ITCM

PMAT is generated from CMAT by adding the neighbors' values in the corresponding element of CMAT matrix. For example, the PMAT [1][1] is equal to 3, which is the summation of CMAT[0][1], CMAT[1][0], CMAT[2][1], and CMAT[1][2]. Finally, IMAT matrix is created by examining all elements of PMAT through the *if* conditions of the Algorithm 2. The values of IMAT[0][1] and IMAT[2][1] are both 1 since the condition of line 36 in Algorithm 2 (PMAT[1][1] == 3) is satisfied. The remaining of values of IMAT matrix is completed by tracing the proposed algorithm. Decision vector which is sent to the receiver side is generated by voting the number of '1' values of each row in IMAT matrix. If the sum of all values in each row is greater than the half size of TSV array columns, the corresponding element of the decision vector becomes 1 and the inverted values of the corresponding row of BMAT are transmitted through the TSV lines.

7.2.3 ITCM Method Evaluation

In order to evaluate the efficiency of the proposed ITCM approach, two different designs have been modeled; one embedded with encoder/decoder to implement the

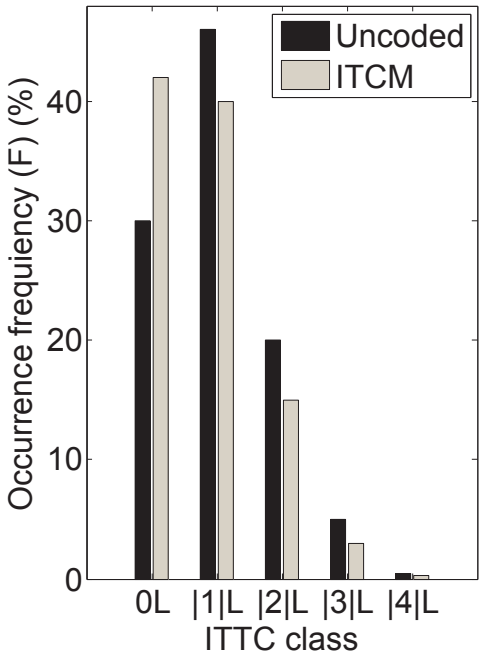
proposed ITCM approach, and the other without encoder/decoder components. Similar to evaluation of the CTCA both random and realistic applications (PARSEC benchmark traces) have been used.

Figure 7.8a shows the occurrence percentage of all possible ITTC categories in an 8×8 TSV bundle in both uncoded (left bar) and coded (right bar) models for random data traffic. In Figure 7.8a, there are 5 pairs of columns, where the left one refers to the uncoded system and the right column refers to the coded system. Figure 7.8b also reports the mitigation factor for each class of ITTC. As is evident from Figure 7.8a and Figure 7.8b, the majority of inductive coupling with larger α values are replaced by smaller ones by using the proposed ITCM method.Figure 7.8c shows the occurrence frequency of each of the ITTC classes for different applications of PARSEC benchmark workload traffic before and after using the ITCM approach.

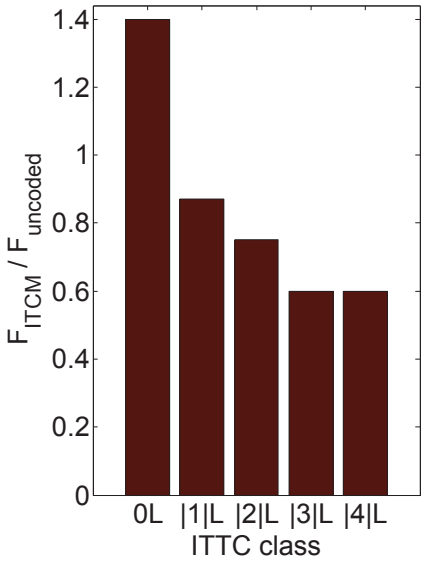
Similar to Figure7.8a the right column for each application reports the inductive coupling values in a coded system while the left one shows inductive coupling values for the uncoded one. In 7.8c, for each workload the left bar represents the uncoded and the right bar shows the coded approach results. As expected the percentage of the inductive coupling with smaller α values in the coded system is much higher than similar ones in the uncoded system. Encoder of the ITCM approach was implemented in Verilog and synthesized by Synopsys Design Compiler using 28nm TSMC library (1.05V, 25 °C), for estimating the overhead cost. The latency, power consumption, and area occupation of the proposed encoder are summarized in Table 7.4. But, the decoder unit synthesis result is not reported, since it is only composed of a comparator and a mix of inverter gates. It is considerably less complex than the encoder component in terms of area, power consumption, and latency.

Table 7.4: Hardware synthesis results of ITCM

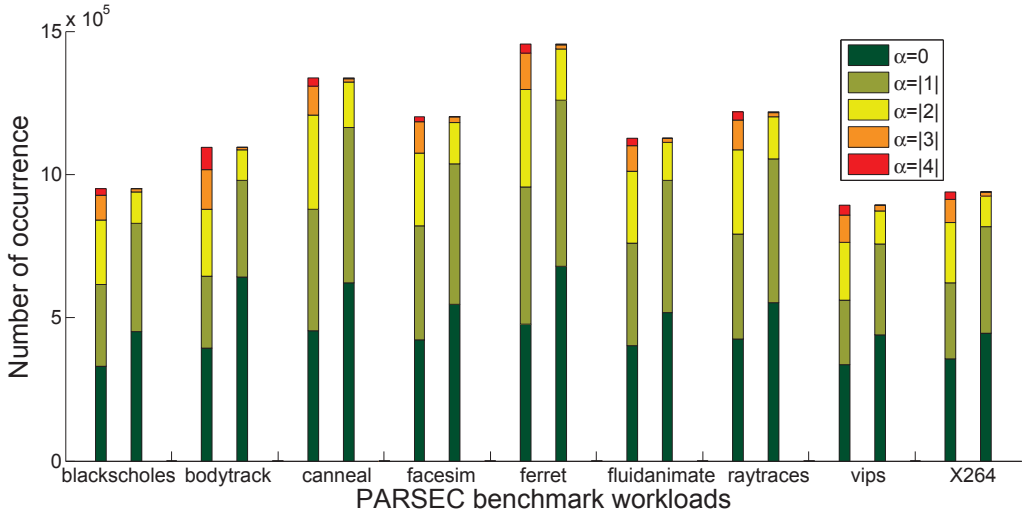
Bus size	Area	Latency	Power
64-bit	$7290\mu m^2$	$14ps$	$5.34mW$
128-bit	$13851\mu m^2$	$20ps$	$10.06mW$
256-bit	$30200\mu m^2$	$29ps$	$20.23mW$
512-bit	$61786\mu m^2$	$41ps$	$40.48mW$



(a) Random application



(b) Random application



(c) PARSEC data traffic

Figure 7.8: Evaluating the efficiency of ITCM for an 8×8 TSV bus

Chapter 8

Conclusion

3D technologies promise increased system integration at lower cost and reduced footprint. They support performance improvement such as increased bandwidth and easier reuse by mixing and matching existing silicon. The fabrication process is becoming somewhat mature. However, reliability issues need to be addressed for an eventual transition from laboratory to production. The goal of this project is to develop a Three-Dimensional NoC Reliability Evaluation Automatic Tool (TREAT), for the first time, as an analysis tool to analyze effects of static and dynamic faults in 3D NoC architectures. This approach allows injecting faults into the 3D NoC platform dynamically by monitoring the status of links and components to decide where and when to inject faults accurately. The proposed tool reports the strength of different components in terms of reliability based metrics. TREAT is useful for fault-tolerant designers at the early stage of the design to save time and cost.

In more detail, in this research first all possible potential sources of physical faults in 3D NoC environment have been studied. The impacts of all the potential physical faults on 3D NoC components are also addressed to use them as reliability metrics

in TREAT. All the main components of a 3D NoC router architecture and sensitive entities with their possible responses to physical faults are categorized in this dissertation. An HDL model of 3D NoC environment as a platform to run my experiments is also designed and implemented as a part of this project. Then, the physical causes of faults in logic-level are modeled and applied as fault libraries in TREAT. TREAT is able to model to support both static and dynamic fault injections. It has been developed in a structural way to support future plug-in fault models. In static fault injection phase TREAT runs the simulation for number of times which is defined by the user. TREAT guarantees to inject faults when the target system is stable. The life time of the static fault models are also defined by a random function with exponential distribution. By comparing the value of selected signals and their assertion time in 3D NoC components after running both faulty and faultless experiments the vulnerability of each component is reported. The effect of psychical fault models are used as evaluation metrics in TREAT. The presented tool is capable of evaluating any Verilog model of 3D NoC architecture or fault-tolerant techniques.

In addition, by exploring the physical fault in 3D NoC environment, both Capacitive and Inductive TSV-to-TSV coupling mitigation coding have been proposed as a side related topic to this project. Furthermore, an efficient routing technique has been developed as part of THeNoC simulator development.

Future work

From this thesis, major directions for future work can be derived. As discussed earlier TREAT is implemented in a modular structure. One interesting field would be adding more fault model libraries specially the dynamic fault models such as a thermal fault model. Similar to what has been discussed for TSV coupling issue, the output of a circuit-level model can be applied in system-level by TREAT to model the effect of thermal issues in 3D NoC environment. The other interesting direction would be to

add more reliability metrics which are introduced in Chapter 4, although they depend on the capability of the test bench of 3D NoC model.

Bibliography

[1] K. Aisopos, C.-H. Chen, and L.-S. Peh. Enabling system-level modeling of variation-induced faults in networks-on-chips. In *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, pages 930–935, 2011.

[2] S. Ashby, P. Beckman, J. Chen, P. Colella, B. Collins, D. Crawford, J. Dongarra, D. Kothe, R. Lusk, P. Messina, and Others. The opportunities and challenges of exascale computing. *summary report of the advanced scientific computing advisory committee (ASCAC) subcommittee at the US Department of Energy Office of Science*, 2010.

[3] A. Avizienis, J.-C. Laprie, and B. Randell. *Fundamental concepts of dependability*. University of Newcastle upon Tyne, Computing Science, 2001.

[4] L. T. B. Wu, X. Gu and M. Ritter. Electromagnetic modeling of massively coupled through silicon vias for 3d interconnects. In *Microwave and Optical Technology Letters*, page 12041206, 2011.

[5] J. H. Bahn and N. Bagherzadeh. A generic traffic model for on-chip interconnection networks. In *International Workshop on Network-on-Chip Architectures*, pages 22–29, 2009.

[6] J. Baraza, J. Gracia, D. Gil, and P. Gil. A prototype of a vhdl-based fault injection tool: description and application. *Journal of Systems Architecture*, 47(10):847 – 867, 2002.

[7] R. C. BAUMANN. Soft errors in commercial integrated circuits. *International Journal of High Speed Electronics and Systems*, 14(02):299–309, 2004.

[8] A. Benso and P. Prinetto. *Fault injection techniques and tools for embedded systems reliability evaluation*, volume 23. Springer Science & Business Media, 2003.

[9] D. Bertozzi, L. Benini, and G. D. Micheli. Error control schemes for on-chip communication links: The energy-reliability tradeoff. *IEEE Trans. CAD*, 24(6):818–831, 2000.

[10] C. Bienia. *Benchmarking Modern Multiprocessors*. PhD thesis, Princeton University, January 2011.

[11] M. Birner and T. Handl. Arrow - a generic hardware fault injection tool for nocs. In *Digital System Design, Architectures, Methods and Tools, 2009. DSD '09. 12th Euromicro Conference on*, pages 465–472, Aug 2009.

[12] P. Bogdan, T. Dumitras, and R. Marculescu. Stochastic communication: A new paradigm for fault-tolerant networks-on-chip. In *VLSI Design*, 2007.

[13] J. Borecky, M. Kohlik, P. Kubalik, and H. Kubatova. Fault models usability study for on-line tested fpga. In *Digital System Design (DSD), 2011 14th Euromicro Conference on*, pages 287–290, 2011.

[14] S. Borkar. 3d integration for energy efficient system design. In *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, pages 214–219, 2011.

[15] D. Brooks, R. Dick, R. Joseph, and L. Shang. Power, thermal, and reliability modeling in nanometer-scale microprocessors. *Micro, IEEE*, 27(3):49–62, 2007.

[16] V. Champac, J. Hernandez, S. Barcelo, R. Gomez, C. Hawkins, and J. Segura. Testing of stuck-open faults in nanometer technologies. *Design Test of Computers, IEEE*, 29(4):80–91, 2012.

[17] Y.-Y. Chang, Y.-C. Huang, V. Narayanan, and C.-T. King. Shieldus: A novel design of dynamic shielding for eliminating 3d tsv crosstalk coupling noise. In *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*, pages 675–680, Jan 2013.

[18] Y. Cheng, L. Zhang, Y. Han, and X. Li. Thermal-constrained task allocation for interconnect energy reduction in 3-d homogeneous mpsocs. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 21(2):239–249, 2013.

[19] K.-W. Chung, S. Shih, S.-T. Lu, T.-H. Chen, C.-T. Chen, J. Ho, J.-J. Chen, and J.-P. Lin. 3d stacking dram using tsv technology and microbump interconnect. In *Microsystems Packaging Assembly and Circuits Technology Conference (IMPACT), 2010 5th International*, pages 1–4, Oct 2010.

[20] r. Cota, A. de Morais Amory, and M. Lubaszewski. Noc basics. In *Reliability, Availability and Serviceability of Networks-on-Chip*, pages 11–24. Springer US, 2012.

[21] A. Dalirsani, M. Hosseinabady, and Z. Navabi. An analytical model for reliability evaluation of noc architectures. In *On-Line Testing Symposium, 2007. IOLTS 07. 13th IEEE International*, pages 49–56, July.

[22] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[23] A. DeOrio, D. Fick, V. Bertacco, D. Sylvester, D. Blaauw, J. Hu, and G. Chen. A reliable routing architecture and algorithm for nocs. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 31(5):726–739, May.

[24] R. Doe. Hmc specification, Aug. 2012.

[25] F. Dubois, A. Sheibanyrad, F. Petrot, and M. Bahmani. Elevator-first: A deadlock-free distributed routing algorithm for vertically partially connected 3d-nocs. *Computers, IEEE Transactions on*, 62(3):609–615, 2013.

[26] A. Eghbal, P. Yaghini, N. Bagherzadeh, and M. Khayambashi. Analytical fault tolerance assessment and metrics for tsv-based 3d network-on-chip. *Computers, IEEE Transactions on*, 64(12):3591–3604, Dec 2015.

[27] A. Eghbal, P. Yaghini, H. Pedram, and H. Zarandi. Fault injection-based evaluation of a synchronous noc router. In *On-Line Testing Symposium, 2009. IOLTS 2009. 15th IEEE International*, pages 212–214, 2009.

[28] A. Eghbal, P. M. Yaghini, and N. Bagherzadeh. Capacitive coupling mitigation for tsv-based 3d ics. In *VLSI Test Symposium (VTS), 2015 IEEE 33rd*, pages 1–6, April 2015.

[29] A. Eghbal, P. M. Yaghini, H. Pedram, and H. R. Zarandi. Designing fault-tolerant network-on-chip router architecture. *International Journal of Electronics*, 97(10):1181–1192, 2010.

[30] A. Eghbal, P. M. Yaghini, S. S. Yazdi, and N. Bagherzadeh. Tsv-to-tsv inductive coupling-aware coding scheme for 3d network-on-chip. In *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2014 IEEE International Symposium on*, pages 92–97. IEEE, 2014.

[31] A. Eghbal, H. Zarandi, and P. Yaghini. Fault tolerance assessment of pic microcontroller based on fault injection. In *Test Workshop, 2009. LATW '09. 10th Latin American*, pages 1–6, March 2009.

[32] L. Entrena, C. Lpez, and E. Olas. Automatic generation of fault tolerant vhdl designs. In *in RTL, in: Forum on Design Languages (FDL2001*, 2001.

[33] B. Feero and P. Pande. Networks-on-chip in a three-dimensional environment: A performance evaluation. *Computers, IEEE Transactions on*, 58(1):32–45, Jan 2009.

[34] P. Folkesson, S. Svensson, and J. Karlsson. A comparison of simulation based and scan chain implemented fault injection. In *Fault-Tolerant Computing, 1998. Digest of Papers. Twenty-Eighth Annual International Symposium on*, pages 284–293, June 1998.

[35] R. Gaillard. Single event effects: Mechanisms and classification. In M. Nicolaidis, editor, *Soft Errors in Modern Electronic Systems*, volume 41 of *Frontiers in Electronic Testing*, pages 27–54. Springer US, 2011.

[36] D. Gil-Tomas, J. Gracia-Moran, J.-C. Baraza-Calvo, L.-J. Saiz-Adalid, and P.-J. Gil-Vicente. Analyzing the impact of intermittent faults on microprocessors applying fault injection. *Design Test of Computers, IEEE*, 29(6):66–73, 2012.

[37] E. Guthmuller, I. Miro-Panades, and A. Greiner. Architectural exploration of a fine-grained 3d cache for high performance in a manycore context. In *Very Large Scale Integration (VLSI-SoC), 2013 IFIP/IEEE 21st International Conference on*, pages 302–307, 2013.

[38] W. L. Heimerdinger, W. L. Heimerdinger, C. B. Weinstock, C. B. Weinstock, T. R. Miller, and L. Col. A conceptual framework for system fault tolerance, 1992.

[39] D. Henry, S. Cheramy, J. Charbonnier, P. Chausse, M. Neyret, G. Garnier, C. Brunet-Manquat, S. Verrun, N. Sillon, L. Bonnot, A. Farcy, L. Cadix, M. Rousseau, and E. Saugier. Development and characterisation of high electrical performances tsv for 3d applications. In *Electronics Packaging Technology Conference, 2009. EPTC '09. 11th*, pages 528–535, Dec 2009.

[40] M. Horowitz and W. Dally. How scaling will change processor architecture. In *Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International*, pages 132–133 Vol.1, Feb 2004.

[41] J. Howard, S. Dighe, S. Vangal, G. Ruhl, N. Borkar, S. Jain, V. Erraguntla, M. Konow, M. Riepen, M. Gries, G. Droege, T. Lund-Larsen, S. Steibl, S. Borkar, V. De, and R. Van Der Wijngaart. A 48-core ia-32 processor in 45 nm cmos using on-die message-passing and dvfs for performance and power scaling. *Solid-State Circuits, IEEE Journal of*, 46(1):173–183, Jan 2011.

[42] K. Huang. Towards many-core real-time embedded systems. In *Zrich: TIK Institut fr Technische Informatik und Kommunikationsnetze, ETH Zrich*, page 2, 2010.

[43] Intel-cooperation. Pin - A Dynamic Binary Instrumentation Tool.

[44] ITRS. International technology roadmap for semiconductors 2011 edition system drivers.

[45] ITRS. International technology roadmap for semiconductors 2012 Executive Summary.

[46] ITRS. Table intc7-international technology roadmap for semiconductors 2013 executive summary.

[47] J. Jeddeloh and B. Keeth. Hybrid memory cube new dram architecture increases density and performance. In *VLSI Technology (VLSIT), 2012 Symposium on*, pages 87–88, June 2012.

[48] N. Julai, A. Yakovlev, and A. Bystrov. Error detection and correction of single event upset (seu) tolerant latch. In *On-Line Testing Symposium (IOLTS), 2012 IEEE 18th International*, pages 1–6, 2012.

[49] M. Jung, J. Mitra, D. Pan, and S. K. Lim. Tsv stress-aware full-chip mechanical reliability analysis and optimization for 3d ic. In *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, pages 188–193, June 2011.

[50] M. Jung, J. Mitra, D. Z. Pan, and S. K. Lim. Tsv stress-aware full-chip mechanical reliability analysis and optimization for 3d ic. *Commun. ACM*, 57(1):107–115, Jan. 2014.

[51] A. Kamto, Y. Liu, L. Schaper, and S. Burkett. Reliability study of through-silicon via (tsv) copper filled interconnects. *Thin Solid Films*, 518(5):1614 – 1619, 2009. Proceedings of the 36th International Conference on Metallurgical Coatings and Thin Films.

[52] A. Kauppila, G. Vaughn, J. Kauppila, and L. Massengill. Probabilistic evaluation of analog single event transients. *Nuclear Science, IEEE Transactions on*, 54(6):2131–2136, 2007.

[53] N. Khan, S. Alam, and S. Hassoun. Through-silicon via (tsv)-induced noise characterization and noise mitigation using coaxial tsvs. In *3D System Integration, 2009. 3DIC 2009. IEEE International Conference on*, pages 1–7, Sept 2009.

[54] D. H. Kim, K. Athikulwongse, M. Healy, M. Hossain, M. Jung, I. Khorosh, G. Kumar, Y.-J. Lee, D. Lewis, T.-W. Lin, C. Liu, S. Panth, M. Pathak, M. Ren, G. Shen, T. Song, D. H. Woo, X. Zhao, J. Kim, H. Choi, G. Loh, H.-H. Lee, and S. K. Lim. 3d-maps: 3d massively parallel processor with stacked memory. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, pages 188–190, Feb 2012.

[55] D. H. Kim and S.-K. Lim. Design quality trade-off studies for 3-d ics built with sub-micron tsvs and future devices. *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, 2(2):240–248, 2012.

[56] D. H. Kim, S. Mukhopadhyay, and S.-K. Lim. Fast and accurate analytical modeling of through-silicon-via capacitive coupling. *Components, Packaging and Manufacturing Technology, IEEE Transactions on*, 1(2):168–180, Feb 2011.

[57] J. Kim, C. Nicopoulos, and D. Park. A gracefully degrading and energy-efficient modular router architecture for on-chip networks. In *ISCA*, 2006.

[58] M. Koibuchi, H. Matsutani, H. Amano, and T. Pinkston. A lightweight fault-tolerant mechanism for network-on-chip. In *Networks-on-Chip, 2008. NoCS 2008. Second ACM/IEEE International Symposium on*, pages 13–22, April.

[59] M. Krasich. How to estimate and use mttf/mtbf would the real mtbf please stand up? In *Reliability and Maintainability Symposium. Annual*, pages 353–359, Jan.

[60] R. Kumar and S. P. Khatri. Crosstalk avoidance codes for 3d vlsi. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1673–1678, March 2013.

[61] M. Lee. *Electrical Design of Through Silicon Via*. SpringerLink : Bücher. Springer London, Limited, 2014.

[62] C. Liu, T. Song, J. Cho, J. Kim, J. Kim, and S.-K. Lim. Full-chip tsv-to-tsv coupling analysis and optimization in 3d ic. In *Design Automation Conference, 48th ACM/EDAC/IEEE*, pages 783–788, 2011.

[63] X. Liu, Q. Chen, P. Dixit, R. Chatterjee, R. Tummala, and S. Sitaraman. Failure mechanisms and optimum design for electroplated copper through-silicon vias (tsv). In *Electronic Components and Technology Conference, 2009. ECTC 2009. 59th*, pages 624–629, 2009.

[64] I. Loi, F. Angiolini, S. Fujita, S. Mitra, and L. Benini. Characterization and implementation of fault-tolerant vertical links for 3-d networks-on-chip. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 30(1):124–134, 2011.

[65] F. Lu, J. Shao, X. Liu, and X. Wang. Validation test method of tddb physics-of-failure models. In *Prognostics and System Health Management (PHM), 2012 IEEE Conference on*, pages 1–4, 2012.

[66] K. Lu, X. Zhang, S.-K. Ryu, J. Im, R. Huang, and P. S. Ho. Thermo-mechanical reliability of 3-d ics containing through silicon vias. In *Electronic Components and Technology Conference, 2009. ECTC 2009. 59th*, pages 630–634, 2009.

[67] Z. Lu and A. Jantsch. Trends of terascale computing chips in the next ten years. In *ASIC, 2009. ASICON '09. IEEE 8th International Conference on*, pages 62–66, Oct.

[68] M. Lundstrom and J. Guo. *Nanoscale Transistors: Device Physics, Modeling and Simulation*. Springer, 2006.

[69] S. Manolache, P. Eles, and Z. Peng. Fault and energy-aware communication mapping with guaranteed latency for applications implemented on noc. In *DAC*, 2005.

[70] R. Marculescu, U. Ogras, L.-S. Peh, N. Jerger, and Y. Hoskote. Outstanding research problems in noc design: System, microarchitecture, and circuit perspectives. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(1):3–21, 2009.

[71] S. Marella, S. Kumar, and S. Sapatnekar. A holistic analysis of circuit timing variations in 3d-ics with thermal and tsv-induced stress considerations. In *Computer-Aided Design (ICCAD), 2012 IEEE/ACM International Conference on*, pages 317–324, 2012.

[72] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood. Multifacet's general execution-driven multiprocessor simulator (gems) toolset. *SIGARCH Comput. Archit. News*, 33(4):92–99, Nov. 2005.

[73] A. Mercha et al. Comprehensive analysis of the impact of single and arrays of through silicon vias induced stress on high-k / metal gate cmos performance. In *Electron Devices Meeting (IEDM), 2010 IEEE International*, pages 2.2.1–2.2.4, 2010.

[74] F. Mohammadian. Falp: A fault adaptive and low power method for network on chip router. In *Architecture of Computing Systems (ARCS), 2014 27th International Conference on*, pages 1–7, Feb 2014.

[75] R. Moonen, P. Vanmeerbeek, G. Lekens, W. De Ceuninck, P. Moens, and J. Boutsen. Study of time-dependent dielectric breakdown on gate oxide capacitors at high temperature. In *Physical and Failure Analysis of Integrated Circuits, 2007. IPFA 2007. 14th International Symposium on the*, pages 288–291, 2007.

[76] S. Murali, T. Theocharides, N. Vijaykrishnan, M. Irwin, L. Benini, and G. De Micheli. Analysis of error recovery schemes for networks on chips. *Design Test of Computers, IEEE*, 22(5):434–442, 2005.

[77] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C. Das. Exploring fault-tolerant network-on-chip architectures. In *DSN*, pages 93–104, 2006.

[78] V. Pasca, L. Anghel, C. Rusu, and M. Benabdenbi. Configurable serial fault-tolerant link for communication in 3d integrated systems. In *On-Line Testing Symposium (IOLTS), 2010 IEEE 16th International*, pages 115–120, 2010.

[79] S. Pasricha and Y. Zou. A low overhead fault tolerant routing scheme for 3d networks-on-chip. In *Quality Electronic Design (ISQED), 2011 12th International Symposium on*, pages 1–8, March.

[80] L.-S. Peh and N. E. Jerger. *On-Chip Networks*. Morgan and Claypool Publishers, 1st edition, 2009.

[81] Y. Peng, T. Song, D. Petranovic, and S. K. Lim. On accurate full-chip extraction and optimization of tsv-to-tsv coupling elements in 3d ics. In *Computer-Aided Design (ICCAD), 2013 IEEE/ACM International Conference on*, pages 281–288, Nov 2013.

[82] M. Pirretti, G. M. Link, R. R. Brooks, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin. Fault tolerant algorithms for network-on-chip interconnect. In *IEEE Symp. VLSI*, 2004.

[83] I. Pomeranz and S. Reddy. A bridging fault model where undetectable faults imply logic redundancy. In *Design, Automation and Test in Europe, 2008. DATE '08*, pages 1166–1171, 2008.

[84] I. Pomeranz and S. Reddy. Transition path delay faults: A new path delay fault model for small and large delay defects. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 16(1):98–107, 2008.

[85] J. Pontes, N. Calazans, and P. Vivet. An accurate single event effect digital design flow for reliable system level design. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, pages 224–229, 2012.

[86] D. Powell, I. Bey, and J. Leuridan, editors. *Delta Four: A Generic Architecture for Dependable Distributed Computing*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1991.

[87] PTM. Predictive Technology Model.

[88] V. Puente, J. A. Gregorio, F. Vallejo, and R. Beivide. Immunet: A cheap and robust fault-tolerant packet routing mechanism. In *ISCA*, 2004.

[89] Y. Qian, Z. Lu, and W. Dou. From 2d to 3d nocs: A case study on worst-case communication performance. In *Computer-Aided Design - Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on*, pages 555–562, Nov 2009.

[90] A. M. Rahmani, K. R. Vaddina, K. Latif, P. Liljeberg, J. Plosila, and H. Tenhunen. Design and management of high-performance, reliable and thermal-aware 3d networks-on-chip. *Circuits, Devices Systems, IET*, 6(5):308–321, 2012.

[91] N. Ranganathan, K. Prasad, N. Balasubramanian, and K. L. Pey. A study of thermo-mechanical stress and its impact on through-silicon vias. *Journal of Micromechanics and Microengineering*, 18(7):075018, 2008.

[92] P. Roth, C. Jacobi, and K. Weber. Superprocessors and supercomputers. In B. Hoefflinger, editor, *Chips 2020*, The Frontiers Collection, pages 421–427. Springer Berlin Heidelberg, 2012.

[93] C. Rousselle, M. Pflanz, A. Behling, T. Mohaupt, and H. Vierhaus. A register-transfer-level fault simulator for permanent and transient faults in embedded processors. In *Design, Automation and Test in Europe, 2001. Conference and Exhibition 2001. Proceedings*, pages 811–, 2001.

[94] S.-K. Ryu, K.-H. Lu, T. Jiang, J.-H. Im, R. Huang, and P. S. Ho. Effect of thermal stresses on carrier mobility and keep-out zone around through-silicon vias for 3-d integration. *Device and Materials Reliability, IEEE Transactions on*, 12(2):255–262, 2012.

[95] K. Salah, Y. Ismail, and A. El-Rouby. *Arbitrary Modeling of TSVs for 3D Integrated Circuits*. Springer Publishing Company, Incorporated, 2014.

[96] C. Selvanayagam, J. Lau, X. Zhang, S. K. W. Seah, K. Vaidyanathan, and T. Chai. Nonlinear thermal stress/strain analyses of copper filled tsv (through silicon via) and their flip-chip microbumps. In *Electronic Components and Technology Conference, 2008. ECTC 2008. 58th*, pages 1073–1081, 2008.

[97] C. Serafy and A. Srivastava. Tsv replacement and shield insertion for tsv-tsv coupling reduction in 3d global placement. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, PP(99):1–1, 2014.

[98] S. Shamshiri, A.-A. Ghofrani, and K.-T. Cheng. End-to-end error correction and online diagnosis for on-chip networks. In *Test Conference (ITC), 2011 IEEE International*, pages 1–10, Sept 2011.

[99] L. Shang. Fault nature identification for single-phase adaptive reclosure on double circuit ehv transmission lines with shunt reactors. In *High Voltage Engineering and Application (ICHVE), 2010 International Conference on*, pages 509–512, 2010.

[100] A. Shayan, X. Hu, H. Peng, C.-K. Cheng, W. Yu, M. Popovich, T. Toms, and X. Chen. Reliability aware through silicon via planning for 3d stacked ics. In *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, pages 288–291, 2009.

[101] V. Sieh, O. Tschache, and F. Balbach. Verify: evaluation of reliability using vhdl-models with embedded fault descriptions. In *Fault-Tolerant Computing, 1997. FTCS-27. Digest of Papers., Twenty-Seventh Annual International Symposium on*, pages 32–36, June 1997.

[102] J. solid state technology association. Failure mechanisms and models for semiconductor devices. *JEDEC Publication JEP 122-G*, pages –, 2011.

[103] V. Soteriou, R. Ramanujam, B. Lin, and L.-S. Peh. A high-throughput distributed shared-buffer noc router. *Computer Architecture Letters*, 8(1):21–24, Jan 2009.

[104] M. Stanisavljevi, A. Schmid, and Y. Leblebici. Reliability, faults, and fault tolerance. In *Reliability of Nanoscale Circuits and Systems*, pages 7–18. Springer New York, 2011.

[105] H. Sun, J. Liu, R. Anigundi, N. Zheng, J. Lu, R. Ken, and T. Zhang. Design of 3d dram and its application in 3d integrated multi-core computing systems. *Design Test, IEEE*, PP(99):1–1, 2013.

[106] U. Tida, R. Yang, C. Zhuo, and Y. Shi. On the efficacy of through-silicon-via inductors. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, PP(99):1–1, 2014.

[107] A. Todri, S. Kundu, P. Girard, A. Bosio, L. Dilillo, and A. Virazel. A study of tapered 3-d tsvs for power and thermal integrity. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 21(2):306–319, 2013.

[108] W.-C. Tsai, D.-Y. Zheng, S.-J. Chen, and Y.-H. Hu. A fault-tolerant noc scheme using bidirectional channel. In *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, pages 918–923, June.

[109] K. Tu. Reliability challenges in 3d ic packaging technology. *Microelectronics Reliability*, 51(3):517 – 523, 2011.

[110] R. Ubar, S. Kostin, and J. Raik. Multiple stuck-at-fault detection theorem. In *Design and Diagnostics of Electronic Circuits Systems (DDECS), 2012 IEEE 15th International Symposium on*, pages 236–241, 2012.

[111] X. Wang, K. E. Holbert, and L. T. Clark. Single event upset mitigation techniques for {FPGAs} utilized in nuclear power plant digital instrumentation and control. *Nuclear Engineering and Design*, 241(8):3317 – 3324, 2011.

[112] W. Wolf. *High-Performance Embedded Computing: Architectures, Applications, and Methodologies*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[113] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta. The splash-2 programs: characterization and methodological considerations. In *Computer Architecture, 1995. Proceedings., 22nd Annual International Symposium on*, pages 24–36, 1995.

[114] P. Yaghini, A. Eghbal, S. Asghari, and H. Pedram. Power comparison of an asynchronous and synchronous network on chip router. In *Computer Conference, 2009. CSICC 2009. 14th International CSI*, pages 242–246, Oct 2009.

[115] P. Yaghini, A. Eghbal, H. Pedram, and H. Zarandi. Investigation of transient fault effects in an asynchronous noc router. In *Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on*, pages 540–545, Feb 2010.

[116] P. Yaghini, H. Zarandi, A. Eghbal, A. Jafarzadeh, and S. Eskandari. An investigation of fault tolerance behavior of 32-bit dlx processor. In *Dependability, 2009. DEPEND '09. Second International Conference on*, pages 93–98, June 2009.

[117] P. M. Yaghini and A. Eghbal. Three-dimensional hdl emulator noc (thenoc). In *gram.eng.uci.edu/comp.arch/lab/tools.html*.

[118] P. M. Yaghini, A. Eghbal, and N. Bagherzadeh. A gals router for asynchronous network-on-chip. In *Proceedings of International Workshop on Manycore Embedded Systems*, MES '14, pages 52:52–52:55, New York, NY, USA, 2014. ACM.

[119] P. M. Yaghini, A. Eghbal, and N. Bagherzadeh. On the design of hybrid routing mechanism for mesh-based network-on-chip. *Integration, the {VLSI} Journal*, 50:183 – 192, 2015.

[120] P. M. Yaghini, A. Eghbal, H. Pedram, and H. R. Zarandi. Investigation of transient fault effects in synchronous and asynchronous network on chip router. *Journal of Systems Architecture*, 57(1):61 – 68, 2011. Special Issue On-Chip Parallel And Network-Based Systems.

[121] P. M. Yaghini, A. Eghbal, S. S. Yazdi, and N. Bagherzadeh. Accurate system-level tsv-to-tsv capacitive coupling fault model for 3d-noc. In *Proceedings of the 9th International Symposium on Networks-on-Chip*, NOCS '15, pages 3:1–3:8, New York, NY, USA, 2015. ACM.

[122] P. M. Yaghini, A. Eghbal, S. S. Yazdi, N. Bagherzadeh, and M. M. Green. Capacitive and inductive tsv-to-tsv resilient approaches for 3d ics. *IEEE Transactions on Computers*, 65(3):693–705, March 2016.

[123] W. Yao, S. Pan, B. Achkir, J. Fan, and L. He. Modeling and application of multi-port tsv networks in 3-d ic. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 32(4):487–496, April 2013.

[124] L. Yavits, A. Morad, and R. Ginosar. 3d cache hierarchy optimization. In *3D Systems Integration Conference (3DIC), 2013 IEEE International*, pages 1–5, 2013.

[125] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor. Interconnect-aware and layout-oriented test-pattern selection for small-delay defects. In *Test Conference, 2008. ITC 2008. IEEE International*, pages 1–10, Oct 2008.

[126] K. Yoon, G. Kim, W. Lee, T. Song, J. Lee, H. Lee, K. Park, and J. Kim. Modeling and analysis of coupling between tsvs, metal, and rdl interconnects in tsv-based 3d ic with silicon interposer. In *Electronics Packaging Technology Conference, 2009. EPTC '09. 11th*, pages 702–706, Dec 2009.

[127] T. Yoshinaga and M. Nomura. Trends in r&d in tsv technology for 3d lsi packaging. In *Science and Technology Trends*, pages 26–39. NISTEP Science and Technology Foresight Center, 2010.

[128] B. Zhao, Y. Du, J. Yang, and Y. Zhang. Process variation-aware nonuniform cache management in a 3d die-stacked multicore processor. *Computers, IEEE Transactions on*, 62(11):2252–2265, 2013.