Efficient Post-Silicon Debug Platforms for Future Many-Core Systems

A THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF

Doctor of Philosophy

BY

SIDHARTHA SANKAR ROUT

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI

NEW DELHI– 110020

JUNE, 2023

Acknowledgements

First of all I would love to thank Almighty God for giving me such a wonderful life. This thesis has been made possible through the support and assistance of several individuals in one way or the other throughout my PhD journey. I extend my heartfelt thanks to each and every one of them.

I would like to express my deep gratitude to my advisor, Dr. Sujay Deb for providing me with the opportunity and his guidance through all these years. His valuable suggestions, technical inputs and continuous support through the years have encouraged me to learn and expand my knowledge. His insights during various stages of my research is what has made this thesis possible today. Besides his guidance in my work, he has always encouraged and motivated me to push beyond limits, create opportunities and make my own way in my career. I am forever indebted to him both for his professional and personal support through all these years. I also extend my gratitude to Dr. Sumit Darak (IIIT Delhi), Dr. Anuj Grover (IIIT Delhi), and Dr. Kanad Basu (University of Texas at Dallas) for their guidance and support in completion of this thesis.

I would also like to thank all the people I interacted with at IIIT Delhi. I had amazing colleagues who helped me through all these years. I wish to thank Hemanta Mondal, Wazir Singh, Harsha Gade, Mitali Sinha, Sneha Agarwal, Deepank Grover, Tarun Sharma, and Keshav Goel for their time in several research discussions and for keeping up a positive environment in the lab. I would also like to thank the undergraduate and graduate students who have worked with me on different projects. Last but not least, I would like to express my gratitude to administrative staffs at IIIT Delhi, for all their help in logistics and office related matters. Many of my research assistance was funded by DRDO and SPARC scheme. Their support and efficiency are commendable.

I am grateful to have a loving family who have given me the freedom to choose my career and have had faith in me, no matter what. My parents have always been my constant support throughout all my endeavours and I sincerely thank them from the bottom of my heart for their confidence in me. I am grateful to have a

caring wife who constantly boosts my energy and gives me a fresh perspective on any problem or challenge at hand. I am also thankful to my twin sons, who make this journey even more exciting. As I look back, countless people have contributed in several ways to enabling me. My sincere gratitude, love and thanks to all life. It has been an immensely satisfying and joyful journey.

Sidhartha Sankar Rout

Sidhartha Sankar Rout

Certificate

This is to certify that the thesis titled "Efficient Post-Silicon Debug Platforms for Future Many-Core Systems" being submitted by Sidhartha Sankar Rout to INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI, for the award of the DOCTOR OF PHILOSOPHY, is an original research work carried out under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

Dr. Sujay Deb

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI

Abstract

As the computation is moving towards the exascale era, more and more number of processing cores of heterogeneous natures are getting embedded in a System-on-Chip (SoC). The growing demands for high-performance and increased functionalities would further proliferate this trend in future SoCs. Such many-core systems require efficient and secured interconnection infrastructure for establishing low cost, high speed, and reliable on-chip communication. Thus, the state-of-the-art interconnect modules such as Networks-on-Chip (NoCs) are becoming extremely complex with advanced features like speculation, power management, redundancy, run-time controllability, encryption, etc. The high level of design complexity of the communication network leads to a situation where many functional bugs escape through the pre-silicon verification stage to the actual product on silicon. Even though the processing cores function correctly, bugs in interconnect can very well introduce faults like deadlock, dropped data fault, misroute, etc., which can lead to complete system failure. A substantial percentage of total system errors appear in the interconnect modules of the recent multicore architectures. This necessitates strong post-silicon debug platforms for the NoC subsystems to ensure minimal or no functional communication faults on the actual products.

While post-silicon debug provides an efficient platform to remove elusive design bugs, it suffers from very poor system observability and controllability, which is limited to the I/O pins of the chip. To enhance the system's internal observability during validation, Design for Debug (DFD) structures are instrumented to the original design that includes on-chip trace buffer, trigger unit, trace bus, etc. Traditionally, a trace-based post-silicon debug platform is used that stores the runtime traces in the embedded trace buffer and later forwards them to the debug analyzer through a trace port. The drawbacks of such methods are on-chip storage cost because of the trace buffer size and slow trace transfer because of the low bandwidth trace port. In this thesis, we have focused on the development of efficient DFD structures for post-silicon validation of NoC based many-core systems, both in terms of trace reduction and high speed trace communication. Moreover, after the system

validation and mass production, the DFD hardware remains vestigial on the system. Reuse of such modules for architectural purposes can compensate for the area overhead introduced by them. Therefore, we have proposed to reuse the DFD infrastructure during the in-field operation mode for the performance enhancement of the NoC based systems.

To improve the efficiency of the debug infrastructure, we propose Wireless enabled NoC Debug (WiND) and Redundant Trace Elimination (RTE) frameworks. WiND performs both trace reduction and high speed trace transfer by using the augmented Wireless Interfaces (WIs) in the debug hardware for both test data and trace data communication. RTE majorly focuses on eliminating the redundant traces without degrading the internal observability of the system. To improve the reusability of the debug infrastructure, we propose ReDeSIGN framework. In this proposal, we reuse trace buffer as extended Virtual Channels (VCs) of NoC routers for network throughput improvement, trace prioritization hardware for critical data prioritization, and trace capture modules for starvation control. This way, ReDeSIGN repurposes almost all the debug units for the performance enhancement of the system during the execution mode. On-chip router buffers consume a significant portion of the total system power. So, to minimize the buffer power in ReDeSIGN framework due to increased number of VCs, we propose DNoC, a dynamic VC power management scheme that activates only the required number of VCs based on the application need during runtime. On-chip wireless infrastructure forms the backbone for WiND and is pivotal for achieving higher debug efficiency. The wireless setup requires a Medium Access Control (MAC) mechanism for an interference free sharing of the wireless channel. The efficiency of a wireless-enabled system is majorly driven by the success of the MAC protocol, as its failure degrades the interconnect performance to a large extent. Towards this end, we propose 2DMAC and Secure MAC to improve the efficiency and security of the wireless communication respectively. 2DMAC can dynamically change the token arbitration pattern and tune the channel hold time of each WI based on its run-time traffic density and data criticality status, resulting in efficient wireless channel utilization. Moreover, 2DMAC prioritizes the critical traffic over the non-critical traffic during the wireless data transfer, leading to application speedup. Wireless channel being a shared medium, a corrupted WI can maliciously hold the channel, resulting in Denial of Service (DoS) or Spoofing in wireless communication. This leads to starvation of healthy WIs and under-utilization of wireless channel. Secure MAC illustrates the threat model and provides countermeasure to establish a secure MAC protocol for the wireless infrastructure embedded to the NoC based many core systems.

List of publications

Patents

1. Sidhartha Sankar Rout and Sujay Deb, “Method and System for Post Silicon Validation,” Indian Patent, Application Number: 201911006527, filed on :19/02/2019, published on: 21/08/2020.

Book Chapters

1. Sidhartha Sankar Rout, Mitali Sinha, Sujay Deb, "NoC Post-Silicon Validation and Debug," in Network-on-Chip Security and Privacy, Publisher: Springer Nature, 2021, pp. 339-369, ISBN 978-3-030-69131-8.
2. Mitali Sinha, Sidhartha Sankar Rout, Sujay Deb, "DoS Attack Models and Mitigation Frameworks for NoC-based SoCs," in Frontiers of Quality Electronic Design (QED), Publisher: Springer, 2023, pp. 575-609, ISBN 978-3-031-16343-2.

Journals

1. Sidhartha Sankar Rout, Mitali Sinha, Sujay Deb, “2DMAC: A Sustainable and Efficient Medium Access Control Mechanism for Future Wireless NoCs,” in ACM Journal on Emerging Technologies in Computing Systems (ACM JETC), November 2022.
2. Sidhartha Sankar Rout, M Badri, Mitali Sinha, Sujay Deb, "ReDeSIGN: Reuse of Debug Structures for Improvement in Performance Gain of NoC based MPSoCs," in IEEE Transactions on Emerging Topics in Computing (IEEE TETC), September 2022.
3. Sidhartha Sankar Rout, Sujay Deb and Kanad Basu, "WiND: An Efficient Post-Silicon Debug Strategy for Network-on-Chip," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (IEEE TCAD), 40(11), 2020, pp. 2372-2385.
4. Mitali Sinha, Setu Gupta, Sidharth Sankar Rout, Sujay Deb, "Sniffer: A Machine Learning Approach for DoS Attack Localization in NoC-based SoCs," in IEEE Journal on Emerging and Selected Topics in Circuits and Systems (IEEE JETCAS), 11(2), 2021, pp. 278-291.
5. Mitali Sinha, Pramit Bhattacharyya, Sidharth Sankar Rout, Neha Prakriya, Sujay Deb, "Securing an Accelerator-rich System from Flooding-based Denial-of-Service Attacks," in IEEE Transactions on Emerging Topics in Computing (IEEE TETC), 10(2), 2021, pp. 855-869.

Conferences

1. Sidhartha Sankar Rout, Akshat Singh, Suyog Bhimrao Patil, Mitali Sinha, Sujay Deb, "Security Threats in Channel Access Mechanism of Wireless NoC and Efficient Countermeasures," IEEE International Symposium on Circuits and Systems (ISCAS), Seville, Spain, 2020, pp. 1-5.

2. Sidhartha Sankar Rout, Badri M, Sujay Deb, "Reutilization of Trace Buffers for Performance Enhancement of NoC based MPSoCs," 25th Asia and South Pacific Design Automation Conference (ASP-DAC), Beijing, China, 2020, pp. 97-102.

3. Sidhartha Sankar Rout, Suyog Bhimrao Patil, Vaibhav Ishwarlal Chaudhari, Sujay Deb, "Efficient Router Architecture for Trace Reduction During NoC Post-Silicon Validation," 32nd IEEE International System-on-Chip Conference (SOCC), Singapore, 2019, pp. 230-235.

4. Sidhartha Sankar Rout, Vaibhav Ishwarlal Chaudhari, Suyog Bhimrao Patil, Sujay Deb, "RCAS: Critical Load Based Ranking for Efficient Channel Allocation in Wireless NoC," 32nd IEEE International System-on-Chip Conference (SOCC), Singapore, 2019, pp. 21-26.

5. Sidhartha Sankar Rout, Kanad Basu, Sujay Deb, "Efficient Post-Silicon Validation of Network-on-Chip using Wireless Links," 32nd International Conference on VLSI Design and 18th International Conference on Embedded Systems (VLSID), Delhi, India, 2019, pp. 371-376.

6. Sidhartha Sankar Rout, Hemanta Kumar Mondal, Rohan Juneja, Sri Harsha Gade, Sujay Deb, "Dynamic NoC Platform for Varied Application Needs," 19th International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, 2018, pp. 232-237.

7. Priyanshi Gaur, Sidhartha Sankar Rout, Sujay Deb, "Efficient Hardware Verification Using Machine Learning Approach," 5th IEEE International Symposium on Smart Electronic Systems (IEEE-iSES), Rourkela, India, 2019, pp. 168-171.

8. Mitali Sinha, Sidhartha Sankar Rout, Sri Harsha Gade, Sujay Deb, "Near Threshold Last Level Cache for Energy Efficient Embedded Applications," 9th International Green and Sustainable Computing Conference (IGSC), Pittsburgh, USA, 2018, pp. 1-6.

9. Sri Harsha Gade, Sidhartha Sankar Rout, Ravi Kashyap, Sujay Deb, "Reliability Analysis of On-Chip Wireless Links for Many Core WNoCs," 33rd Conference on Design of Circuits and Integrated Systems (DCIS), Lyon, France, 2018, pp. 1-6.

10. Sri Harsha Gade, Sidhartha Sankar Rout, Sujay Deb, "On-Chip Wireless Channel Propagation: Impact of Antenna Directionality and Placement on Channel Performance," 12th IEEE/ACM International Symposium on Networks-on-Chip (NOCS), Torino, Italy, 2018, pp. 1-8.

11. Sri Harsha Gade, Sidhartha Sankar Rout, Mitali Sinha, Hemanta Kumar Mondal, Wazir Singh, Sujay Deb, "A Utilization Aware Robust Channel Access Mechanism for Wireless NoCs," IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 2018, pp. 1-5.

12. Sri Harsha Gade, Mitali Sinha, Sidhartha Sankar Rout, Sujay Deb, "Enabling Reliable High Throughput On-chip Wireless Communication for Many Core Architectures," IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Hong Kong, 2018, pp. 591-596.

List of awards and honors

1. Dean IRD Research Excellence Award 2021, IIIT Delhi.
2. First Prize in Student Research Forum at the International VLSI Design & Embedded Systems Conference (VLSID), 2021.
3. ACM SIGDA Best Student Research Forum Poster Award at the Asia and South Pacific Design Automation Conference (ASP-DAC), 2021.
4. Semifinalists of McCluskey Doctoral Thesis Award at the IEEE European Test Symposium (ETS), 2021.
5. Third Prize at IVPC-2020, an initiative under ISEA Project Phase - II, MeitY Govt of India.
6. Selected as one of the best popular science stories under the PhD Category award by Augmenting Writing Skills for Articulating Research (DST AWSAR), 2019.

List of Acronyms

CDF.....Corrupt Data Fault

CUD.....Circuit Under Debug

DAP.....Debug Access Port

DDF.....Dropped Data Fault

DF.....Direction Fault

DFD.....Design for Debug

DFT.....Design for Test

FA.....Fault Analysis

HD.....Header Decoder

IP.....Intellectual Property

LLC.....Last Level Cache

MAC.....Medium Access Control

MC.....Memory Controller

MCSF.....Multiple Copies in Space Fault

MCTF.....Multiple Copies in Time Fault

MPSoC.....Multi-Processor System on Chip

NI.....Network Interface

NoC.....Network on Chip

OOK.....On Off Keying

PIR.....Packet Injection Rate

PMC.....Power Management Controller

PMU.....Packet Monitor Unit

PTC.....Packet Trace Collection

QoS.....Quality of Service

RC.....Routing Computation

RR.....Round Robin

SA.....Switch Arbiter

SoC.....System on Chip

TDT.....Trace Data Transfer

TPH.....Trace Priority Hardware

TrU.....Trigger Unit

TU.....Timestamp Unit

VA.....Virtual Channel Allocator

VC.....Virtual Channel

WI.....Wireless Interface

WNoC.....Wireless Network on Chip

Note: This list contains the list of acronyms that are common to the whole thesis. Chapter specific acronyms are tabulated in the respective chapter wherever required.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The tremendous advancements in CMOS manufacturing technology have navigated the rapid growth in semiconductor industry. The contemporary System-on-Chip (SoC) embeds multiple processing cores of heterogeneous nature (CPU/GPU/accelerators) on a single package to achieve high performance and high throughput operations. Such multi-core processing platforms are gaining significant interest for a wide range of applications, viz., Internet of Things, consumer electronics, single-chip cloud computers, supercomputers, automotive, defense applications etc. The growing demand for high performance and increased functionality in a wide variety of applications would undoubtedly proliferate the number of on-chip processing cores on future SoCs [1]. These cores communicate with each other using an interconnect fabric. The traditional systems use bus based communication infrastructure for this purpose. But with the shrinking technology and increased system size, bus interconnect does not scale efficiently. For the last two decades, Network-on-Chip (NoC) has been evolved as a better scalable interconnect solution for Multi-Processor SoC (MPSoC) [2]. It decouples the computation and communication and facilitates high throughput as well as parallel operation.

NoC is primarily comprised of routers, which are responsible for taking data routing decisions. Routers along with the wired links establish the whole data communication network. All the routers are connected to their neighbouring nodes through the wired links. Each processing core is attached to a router through a network interface (NI) and can communicate with other on-chip modules through multi-hop data communication driven by a routing algorithm. The shared and distributed nature of NoC allows multiple cores to transfer data simultaneously. Short length wired links and pipelined router architecture eliminates the performance bottleneck of long wired bus communication system to a great extent [3]. The state-of-the-art SoCs demand efficient and reliable communication within the on-chip modules. Therefore, performance and power efficiency along with a high level of Quality of Service (QoS) are desired during on-chip data transfer. To achieve these goals, the

NoC module is supported with multiple advanced features like power management, speculation, redundancy, run-time controllability, fault tolerance, etc. Several topological advancements are also made in the recent time by augmenting wireless [4, 5] and optical [6] infrastructure to the NoC module to improve its communication efficiency. Although these features result in efficient communication, it makes the contemporary NoCs extremely complex. Due to design intricacy, it is very difficult and time-consuming to detect all the functional bugs during the pre-silicon verification stage [7]. As a result, multiple errors may slip on to the fabricated interconnect module. A survey conducted by [8] shows that a significant percentage of total design errors in multi-core architectures come from the interconnection networks. The survey summarizes the design errors found in different components of recent multi-core architectures, collected from their respective errata sheets. It finds around 48%, 26%, 16.07%, and 13% errors appearing in the interconnect modules of ARM MX6, Intel Xeon E5, Intel Xeon Phi, and AMD Opteron respectively. The errors on interconnect may introduce several data communication faults, and lead to complete system failure. Therefore, efficient post-silicon validation is required to capture the escaped errors and provide a bug-free interconnect module for multi-core systems [9].

1.1 System Validation

Several validation steps are performed on a newly designed system to make it bug-free before its release to the market. Figure 1.1 shows the typical validation steps in a system development process. Pre-silicon verification is performed before fabrication, whereas post-silicon validation (alternatively known as post-silicon debug) and manufacturing tests are performed after the fabrication. Post-silicon debug is executed on a few initial prototype chips to find functional and electrical bugs in the design. However, manufacturing tests are carried out on each of the fabricated chips after the mass production to detect any manufacturing defect. Finally, the good chips are shipped, and the bad ones are discarded. Though post-silicon debug and manufacturing tests are performed after fabrication, it can be seen that the planning for them starts from the very beginning of the system development phase. Design for Debug (DFD) and Design for Test (DFT) infrastructures are augmented to the actual design, which facilitate observable and controllable points within the design during the post-fabrication validation steps. This thesis proposes three efficient post-silicon debug methods for NoC-based MPSoC systems.

Post-silicon debug is performed on a few fabricated prototypes of the design. A prototype chip is mounted on a test board with all the peripherals connected, and the test stimuli are applied for the validation. It is expected that the escaped design errors from the pre-silicon verification step should be detected during this phase. Pre-silicon verification is performed through simulators. The slow execution rate of such simulators

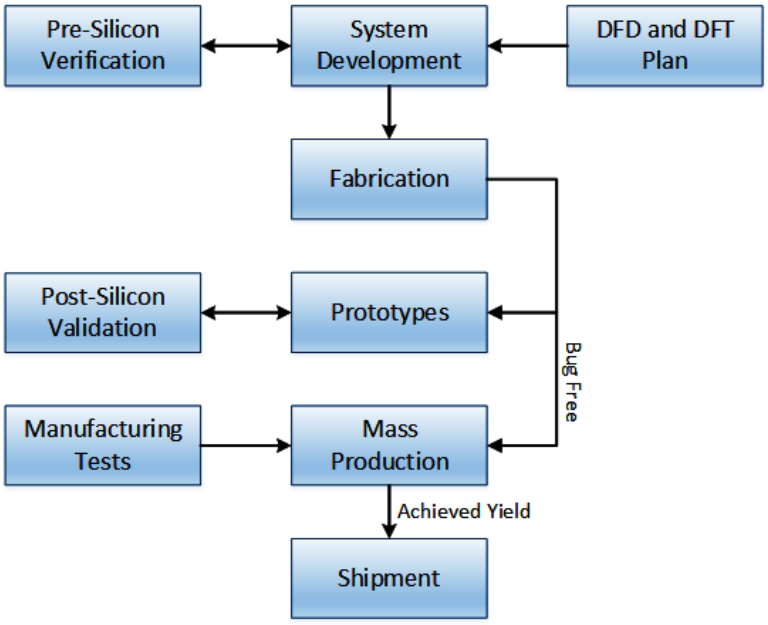


Figure 1.1: Broad validation steps in a system development cycle.

results in longer testing time. In contrast, post-silicon debug is a faster process, as it can run at the speed of system clock frequency [7]. A significant increase in test speed allows a debug engineer to validate the design for more test cases within a short period of time. Another advantage of post-silicon debug is that the design can be tested for electrical errors that may originate due to voltage fluctuation, noise, hotspot, process variation, etc. Thus, the correct behaviors of the system can be validated in actual application environments over specified operating conditions [10, 11]. Detecting electrical errors during pre-silicon verification is challenging, as it is cumbersome to model such operating environments on the simulator.

1.2 Post-Silicon Debug Challenges

Though post-silicon debug provides an efficient platform to remove elusive design bugs, it suffers from very poor system observability and controllability, which is limited to the I/O pins of the chip. This is because the tests in this phase are performed on the prototype chips and the debug engineer has access to the I/O pins only [8, 12]. As a result, it becomes very difficult to debug and localize a bug upon the detection of a fault. To enhance the system's internal observability during validation, DFD structures are instrumented to the original design during the development phase [7, 12]. Traditionally, a trace based post-silicon debug framework is adopted for system validation, as shown in Figure 1.2. The DFD structures include different debug modules such as on-chip trace buffer, trigger unit, trace bus, etc. During the DFD plan, a few observable points and trigger event conditions are decided. Through the validation phase, the system executes normal operations and

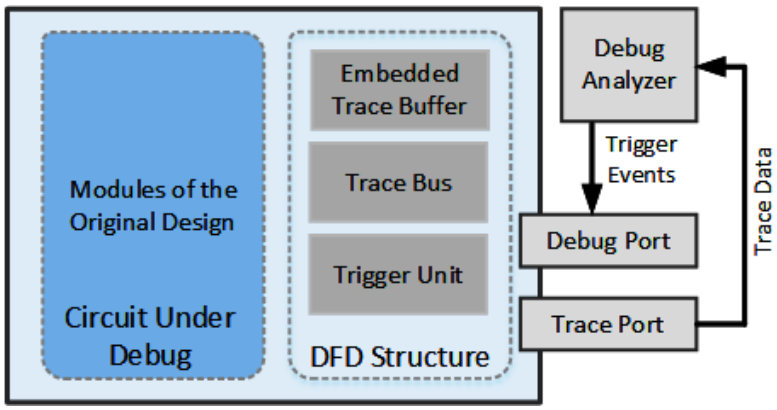


Figure 1.2: Trace based post-silicon debug architecture.

whenever trigger event conditions are satisfied, the run-time traces of the observable points are captured and stored in the trace buffer. Stored traces are later exported to an external debug analyzer through the trace bus and trace port for fault detection and localization. These traces improve the internal observability of the system and thereby help to detect as well as find the root-cause of the bugs during fault analysis [13, 14]. However, the conventional trace based post-silicon debug poses multiple challenges in the system validation process. Starting from DFD plan to debug signoff, there exist several implementation and execution challenges. We broadly present a list of potential challenges involved in conventional trace based post-silicon debug as follows.

- Cost of debug structures: System internal visibility and DFD overhead are competing in nature. The implementation cost of an integrated debug system depends largely on the size of trace buffer [10]. As the size and complexity of NoC based MPSoCs are increasing, the trace buffer footprint is also growing to maximize the system observability during debug. The size of trace buffer directly translates to the area and power overhead of the debug structure. Thus, the requirement of large size trace buffer is considered as a major challenge involved in post-silicon debug.

- Signal selection: Trace buffer width decides the number of observable points that can be traced during debug. As the observable points are limited, it is a challenging task to decide which signals to be traced among the plethora of signals in a design [13]. This is known as signal selection, and is performed during design time. Moreover, large industrial designs, increasing possibilities of electrical bugs and use of less standardized metric such as state restoration ratio [15] increase the challenges associated with signal selection.

- Speed of debug: The post-silicon debug consumes more than 50% of an SoC's overall design time and effort [16]. Exporting large amount of traces to the off-chip debug analyzer via a low bandwidth JTAG interface introduces substantial latency overhead to the debug process [13]. With stringent time-to-market

constraints, a prolonged debug stage may lead to late release of the product. Hence, speed of debug is another challenge associated with the post-silicon validation.

- Error reproducibility and bug localization: Error reproducibility is yet another challenge during post-silicon debug phase. Once a bug is detected during trace analysis, it is required to reproduce the same error to find out the root-cause of the bug. However, an error may be masked by a glitch or fluctuating voltage levels or fluctuating temperature while reproducing it [7]. Moreover, asynchronous I/Os and multiple clock domains make the bug reproduction even more challenging. Difficulty in error reproducibility leads to another challenge that is bug localization. The problem of bug localization is further exacerbated by long error detection latency [17]. This is the latency between the occurrence of an error and its manifestation as an observable failure. If this latency is too high, then it is very much difficult to trace back the location of the bug.

- Secured debug: The requirements of post-silicon debug and system security are conflicting in nature. The observability provided by the DFD structures are crucial for detecting and root-causing the existing bugs. However, increased observability brings up the threat of security attack. The structures inserted for the benefit of debugging can be a source of information leakage. Trace buffer attack can successfully leak secret key information from Advanced Encryption Standard (AES) [18]. Thus, designing secured debug structure is considered as a major challenge in post-silicon debug.

- Unused debug structures during in-field operation: Most of the debug structures becomes non-functional once the SoC goes into production [9]. The DFD hardware occupies a significant portion of the design area (up to 20% or more in silicon real estate [7]) without performing any useful function during the in-field operation. Therefore, the area overhead introduced by DFD hardware is considered as a major design concern associated with the post-silicon debug for another reason.

In our work, we primarily target three of the most important challenges, such as implementation cost of the debug structures, debug speed and unused debug hardware during in-field operation. Our contributions in this thesis provide the solutions to the targeted challenges primarily associated with the post-silicon debug for NoC based multi-core systems.

1.3 Research Contributions

The contributions of this thesis are mainly towards developing efficient debug structures and reuse of debug hardware for architectural purposes. Figure 1.3 summarizes the targeted challenges associated with the trace-

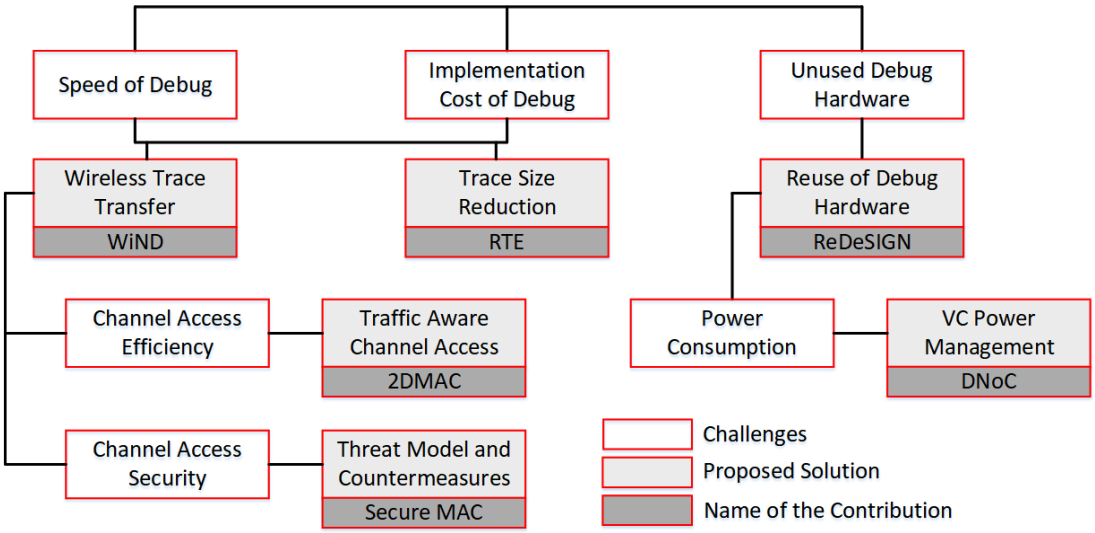


Figure 1.3: Targeted challenges associated with post-silicon debug and the summary of the thesis contributions.

based post-silicon debug, the proposed solutions and the name of the corresponding contributions in this thesis. The contributions are summarized as below.

1.3.1 Efficient post-silicon debug structures

Under the efficient post-silicon debug structures, we have considered two important challenges, such as high implementation cost of the debug structures and the speed of debug. As highlighted earlier, the implementation cost of an integrated debug system depends largely on the size of the trace buffer [10]. Several techniques such as signal selection [13, 14] and trace compression [19, 20] are proposed in literature to address this problem by reducing the level of trace amount during debug. However, these two techniques increase design complexity in-terms of the implementation of the restoration and the compression engine, respectively. Moreover, they do not guarantee complete system states restoration. Speed of debug is another challenge associated with the post-silicon validation [16]. Large amount of traces are exported to the off-chip debug analyzer for fault analysis during debug. Traditionally, this is done through a low bandwidth JTAG interface that introduces substantial latency overhead to the debug process [13]. To deal with this; in [9], a centralized on-chip debug analyzer is instrumented that eliminates off-chip trace transfer. But, the on-chip analyzer introduces large area and power overhead. To address both these debug implementation cost and debug speed challenges, two solutions named WiND and RTE are proposed in this thesis, as shown in Figure 1.3. WiND is a wireless enabled NoC debug framework that augments Wireless Interfaces (WIs) on top of the baseline wired NoC for validation purposes. The wireless medium is utilized for long-range intra-chip test payload communication and for high-speed inter-chip trace transfer during debug. Wireless enabled NoC platform reduces the average intermediate hop count

between source to destination by performing long-range data communication over the wireless medium [4]. Therefore, the WiND setup leads to less trace generation for the same amount of test payload during NoC debug. We further reduce the amount of trace by using the Redundant Trace Elimination (RTE) scheme. Many a time during debug phase, traces of observable points are captured periodically. If the state of a particular signal does not change frequently, then with the small trace capture interval, multiple redundant traces get generated for the same signal. This scenario is evident in case of an NoC-based system as NoC is a shared medium and network congestion is a frequent phenomenon on an NoC due to resource contention. As a result, packets need to wait in the router buffers without any change in their state till the resource allocation. This results in significant amount of redundant packet trace generation. The RTE scheme reduces the trace amount by discarding such redundant traces while keeping only one copy of the trace from each of the redundant group. Both WiND and RTE performs efficient trace reduction while maintaining the same level of system internal observability. Moreover, both WiND and RTE can be integrated to the existing signal selection and trace compression techniques to further enhance the trace reduction.

On-chip trace transfer in traditional debug method is performed through wired trace bus to the JTAG based trace port and again through wired interconnect till the off-chip debug analyzer [13]. The trace transfer bandwidth provided in this case is low. Therefore, this thesis proposes to replace the traditional interface used for exporting traces to the debug analyzer with high bandwidth WI. Moreover, both WiND and RTE reduce the trace amount and thus need to transfer less trace, resulting in higher debug speed.

The WiND framework augments the WIs on the NoC for debug purpose. The WIs use the wireless medium for both test payload and trace data transfer during debug. Moreover, the same wireless infrastructure can be reused for normal payload data transfer during in-field operation. Therefore, both debug efficiency during validation and system performance during in-field operation are largely influenced by the efficiency of the wireless communication on the NoC. Multiple WIs on a Wireless enabled NoC (WNoC) share a single wireless channel for simple and low overhead implementation [21]. All the WIs need to share the same wireless medium for data transmission and reception. Hence, to avoid interference and contention, a Medium Access Control (MAC) mechanism is used that arbitrates and allocates the channel to a particular WI at an instant, and decides the channel hold time for each of the WIs. The performance of WNoC is mostly dependent on the success of the MAC mechanism. Therefore, an efficient MAC is desired. Furthermore, tampering of MAC protocol by a malicious WI node may lead to unfair allocation of wireless channel among the WIs resulting in a degraded system performance. These challenges in WNoC MAC mechanism motivate another major contribution of this thesis that is summarized in Section 1.3.4, and the outcomes of the contribution named 2DMAC and Secure MAC are shown in Figure 1.3.

1.3.2 Reuse of debug structures

Post-silicon debug, though extremely important, but is a one time process at the end of the product development cycle. In-field debug does not require such a large debug setup as needed by post-silicon debug. Thus, most of the debug hardware become non-functional during in-field operation [9]. However, these unused hardware units consume a significant portion, around 20% of the actual design area [7]. This motivates to one of the notable contribution of this thesis work named as ReDeSIGN as shown in Figure 1.3. The proposed framework reuses the unused debug infrastructures on the design for different architectural purposes to enhance the system performance. Trace buffer is re-purposed as extended Virtual Channels (VCs) of NoC routers. Availability of additional VCs effectively reduces the packet drops and deadlock conditions and thereby improves the network throughput. The Trace Priority Hardware (TPH) implemented in router arbiter modules to speed up the trace transfer is reused for prioritization of critical data at the time of resource assignment during in-field operation. This provides faster forwarding of critical packets on the network, and effectively reduces the application run-time of the system. Moreover, the Packet Monitor Unit (PMU), the Trigger Unit (TrU), and the Timestamp Unit (TU) are re-employed for packet starvation control in the proposed framework. This way, the proposed ReDeSIGN framework enables the re-usability of almost all the DFD structures for the system performance enhancement, and thereby compensates for the area overhead associated with them.

ReDeSIGN framework reuses the trace buffer as extended VCs of the NoC routers. This extensively increases the total number of VCs on the network. VCs are essential for high performance of the interconnection network, but claim a significant portion of the total NoC power. This necessitates an efficient VC power management solution and motivates another contribution of this thesis work. The respective contribution is named DNoC as shown in Figure 1.3 and summarized in the following section.

1.3.3 VC power management in NoC

Among all the NoC components, VCs are the most power hungry modules [22]. VCs in the NoC routers store intermediate packets while they wait for the network resources, thereby reduce significant packet drops [23]. Moreover, they minimize head-of-line packet blocking, resulting in decreased deadlock conditions [24] and thus improve the NoC performance. However, the usage of the VCs in the NoC routers are application dependent and for most applications, not all the VCs are used. This motivates for a dynamically reconfigurable system that can switch between both high performance and low power modes to exploit the variable workload conditions provided by different applications. One of the contribution of this thesis proposes a Dynamic NoC platform (DNoC) that optimizes VC utilization for different applications using a smart router architecture. A

Utilization Computation Unit (UCU) is implemented within the NoC routers that can estimate the load density for the downstream routers in the upcoming cycles. Based on the load information provided by the UCUs of the neighbouring nodes, the Power Management Controller (PMC) can decide the number of active VCs within a router to achieve the required performance. The remaining VCs can either be kept in power gated mode or in drowsy mode by the PMC to save considerable VC power.

1.3.4 Efficient and secured wireless channel access in WNoC

A major contribution of this thesis work is development of efficient and secured wireless channel access control (alternatively known as medium access control (MAC)) mechanism for WNoC as highlighted in Section 1.3.1. In traditional MAC [25], a token is circulated among the WIs in a Round Robin (RR) manner. The WI with the token holds the channel for a fixed number of cycles. However, the channel requirement of the individual WIs dynamically changes over time due to the varying traffic density across the WNoC. Moreover, the conventional WNoCs give equal importance to all the traffic taking the wireless path and transmit them in an oldest-first manner. Nevertheless, the critical data can degrade the system performance to a large extent by delaying the application run-time if not served promptly. This thesis proposes a 2-stage Dynamic MAC (2DMAC) as one of its contribution to achieve a better MAC efficiency. In the first stage, 2DMAC ranks the WIs based on their run-time traffic density and criticality status. In the second stage, the proposed framework prioritizes the WIs for the wireless medium access by changing the token arbitration pattern and tuning the channel hold time according to their ranks. Moreover, 2DMAC prioritizes the critical traffic over the non-critical traffic during the wireless data transfer. As a result, the proposed 2DMAC framework considerably enhances the wireless channel utilization, network throughput, and reduces the critical data latency over the traditional MAC.

A centralized MAC incurs significant routing and performance overhead in transferring the control signals [26]. Therefore, typically the WNoC MACs are preferred to be implemented in a distributed manner governed by all the WIs collectively. However, a distributed WNoC MAC can be malfunctioned by a Hardware Trojan (HT) in a malicious WI or a rogue third party Intellectual Property (IP) core present on the same SoC. This may result in *Denial-of-Service (DoS)* or *spoofing* in WNoC leading to starvation of healthy WIs and under-utilization of wireless channel. To highlight the security issue in distributed MAC and its countermeasures, this thesis proposes Secured MAC, as an extension to the design of efficient MAC (2DMAC). Secure MAC demonstrates possible threat model on distributed MAC mechanism and the system performance degradation due to the attacks. Moreover, it discusses low overhead decentralized countermeasures for both *DoS* and *spoofing* attacks in WNoC MAC.

1.4 Organization of the Thesis

The remainder of this thesis is organized as follows:

1. In Chapter 2, we discuss the background of the NoC architecture and operation for both wired and wireless NoC. A fault model for NoC is discussed that includes both short-lived and permanent faults. This chapter also provides details on the generic debug infrastructure of the NoC based systems and their working principles.

2. Chapter 3 focuses on improving the efficiency of post-silicon debug structures for NoC based system. This chapter presents two debug frameworks named WiND and RTE that provide considerable trace reduction and enhanced debug speed.

3. Chapter 4 discusses the overhead challenges of debug hardware and how they are considered as vestigial after the debug phase. This chapter presents ReDeSIGN, that illustrates different debug structure reuse possibilities, which can enhance the system performance during the in-field operation. This chapter also proposes DNoC, a dynamic VC power management scheme that augments the REDeSIGN for better power management of the system during in-field operation.

4. WiND being a wireless-enabled debug structure, Chapter 5 emphasizes on the efficiency and security enhancement of the wireless infrastructure in the NoC based system. This chapter presents 2DMAC that considerably improves the wireless channel utilization and overall system performance of a WNoC based mixed-critical data system. It also demonstrates the threat model to WNoC MAC protocol and suggests Secure MAC, a security countermeasure to the discussed threats.

5. Chapter 6 finally concludes the contributions of this thesis and discusses future directions of our work.

Chapter 2

Background and Related Work

This thesis proposes efficient post-silicon debug platforms for NoC based systems. Therefore, this chapter presents the background of NoC architecture and operation, NoC fault model and the debug infrastructure of the NoC based systems and their working principles. Both wired only (electric) and wireless NoCs are discussed. The background related works present in literature are highlighted in this chapter. In addition, the following chapters highlight more state-of-the-arts related to the proposed solutions.

2.1 NoC Architecture and Operation

With the increasing number of cores, the contemporary SoCs demand intensive parallel communication to meet the performance requirements. Bus-based communication infrastructures are falling short to provide the required bandwidth, latency and power consumption in such systems. NoC, being an embedded switching network, connects the on-chip IP modules and is considered as a better scalable alternative compared to the buses [3]. NoC provides larger design space with the flexibility to implement different topologies, routing protocols, arbitration schemes, flow control mechanisms etc. This thesis focuses on the 2D-mesh NoC based systems with deterministic XY routing. The following two subsections detail the architecture and operation of the wired NoC and wireless NoC.

2.1.1 Wired NoC

The NoC is made up of three main components such as links, routers, and Network Interfaces (NI). Figure 2.1 (a) shows an NoC based multi-core system and highlights the main components of the NoC. The link (channel)

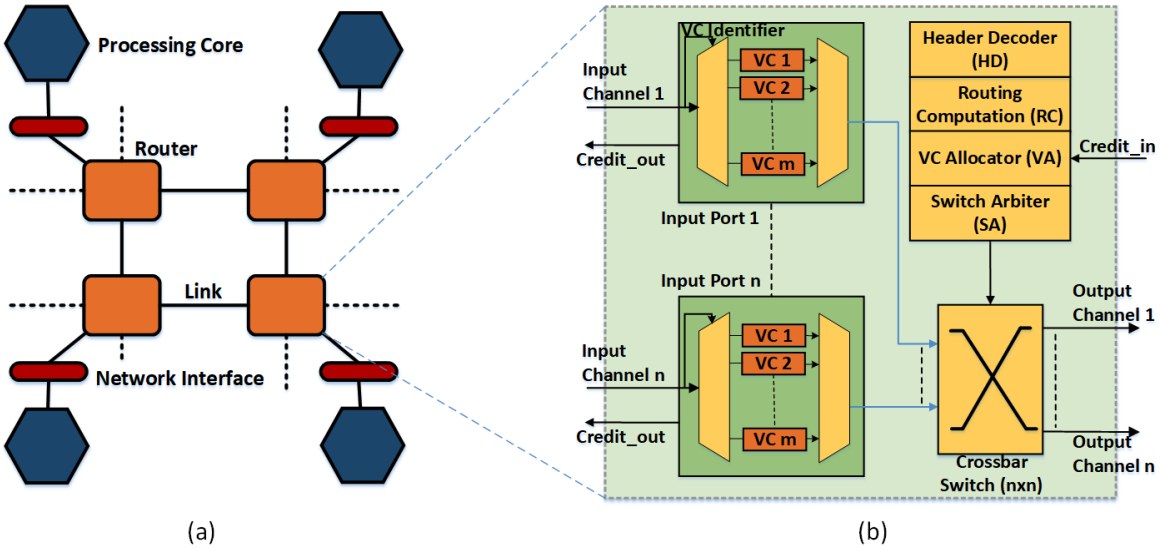


Figure 2.1: (a) Baseline NoC based multi-core system, (b) Architecture of NoC router.

is composed of physical wires that actually establishes the communication between the nodes. The router blocks implement the routing protocol and redirect the message towards its destination [2]. Finally, the NIs provide the connection between the processing cores and the network.

In an NoC-based system, messages are transferred in the form of packets, and packets are further divided into flits as the smallest flow control units. A conventional NoC router is demonstrated in Figure 2.1 (b). Whenever a header flit of a packet reaches the input port of a router node, the *VC Identifier* places the flit in the assigned Virtual Channel (VC). Header Decoder (HD) reads the destination address from the header flit. During the Routing Computation (RC) stage, the output port for the flit is decided based on the destination address and the routing algorithm. The VC Allocator (VA) assigns an input VC of the downstream router after arbitrating among all the competing flits. The *Credit_in* signal (which is the *Credit_out* signal from the downstream router) indicates the availability of the free VCs in the following router. The Switch Arbiter (SA) arbitrates among all the flits targeting a particular output port and allots the crossbar to the winning flit. Arbitration in both VA and SA are performed in a *round-robin* or *oldest-first* manner. Many a time, priority logics are implemented in these arbiters for early allocation of resources to critical data [27]. As soon as the reserved output port and VC in the down-stream router are available, the flit is removed from the current VC and forwarded to the allocated Crossbar input. The Crossbar unit establishes the routing path to the desired output port multiplexer. Finally, the multiplexer transfers the data flit available at its selected input to the output link. Till the time the reserved resources are not available, the flit has to wait in the current VC. In *wormwhole* routing (most popularly adopted flow control mechanism in NoCs) [28], the header flit reserves the VC, body flits follow the header flit, and finally the tail flit releases the VC while leaving the router. Once a VC is released or occupied, the *Credit_out*

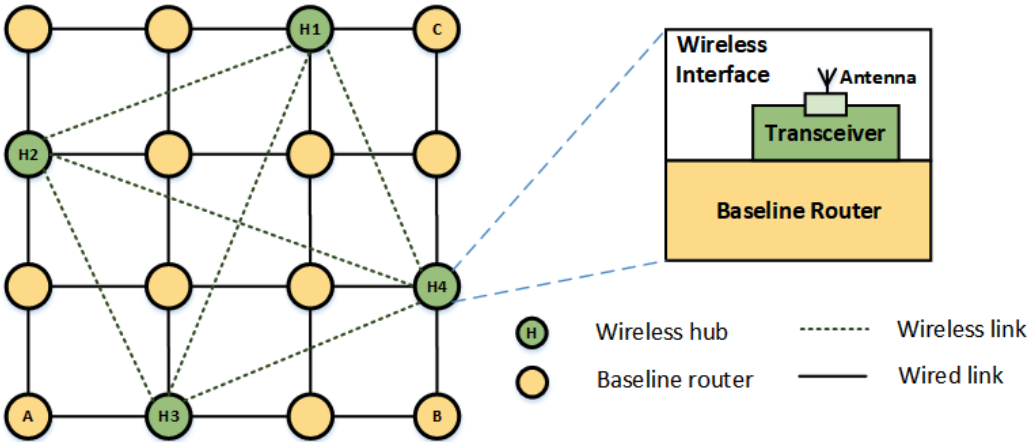


Figure 2.2: Wireless NoC architecture.

signal is updated accordingly. In the case of *wormwhole* routing, HD, RC and VA are performed only on the header flit, and SA is performed on all the flits.

2.1.2 Wireless NoC

For long-distance on-chip communications in a large SoC, packets travel through multiple hops, resulting in higher power and performance overhead. Wireless NoC (WNoC) alleviates this issue by providing single-hop communication between distant nodes [4], as shown in Figure 2.2. For an example, a packet communication from node A to node C would have taken the path all the way from node A to node B and then to node C in a traditional wired NoC. However, in case of the WNoC, the same packet communication would take the path from node A to node H3, then to node H1 over the wireless medium, and finally to the destination node C. This particular communication requires only three hops to reach its destination compared to six nodes in case of a wired NoC. The WNoC augments multiple CMOS compatible Wireless Interfaces (WIs) on top of the conventional wired NoC to establish the low latency and low energy wireless medium [29, 30]. The WIs are optimized to be placed far apart so that they can be used for long-range packet communication. The optimal number of WIs for a particular network size is computed according to simulated annealing based process [21, 31]. The nodes with the supplemented WI on top of the baseline router are known as wireless hubs. The WI is composed of a transceiver unit and an antenna operating in the millimeter-wave (mm-wave) frequency range. The transceiver module is used for converting the digital data to RF domain and vice versa, and the antenna is used to transmit and receive the radio waves. The long-range data transmission is performed over the wireless channel. Most of the WNoCs adopt a single wireless channel to achieve a simple and low overhead implementation [21]. The WIs need to share the available wireless channel for data transmission and reception. A Medium Access Control (MAC) mechanism is deployed that performs a fair arbitration and allocation of the

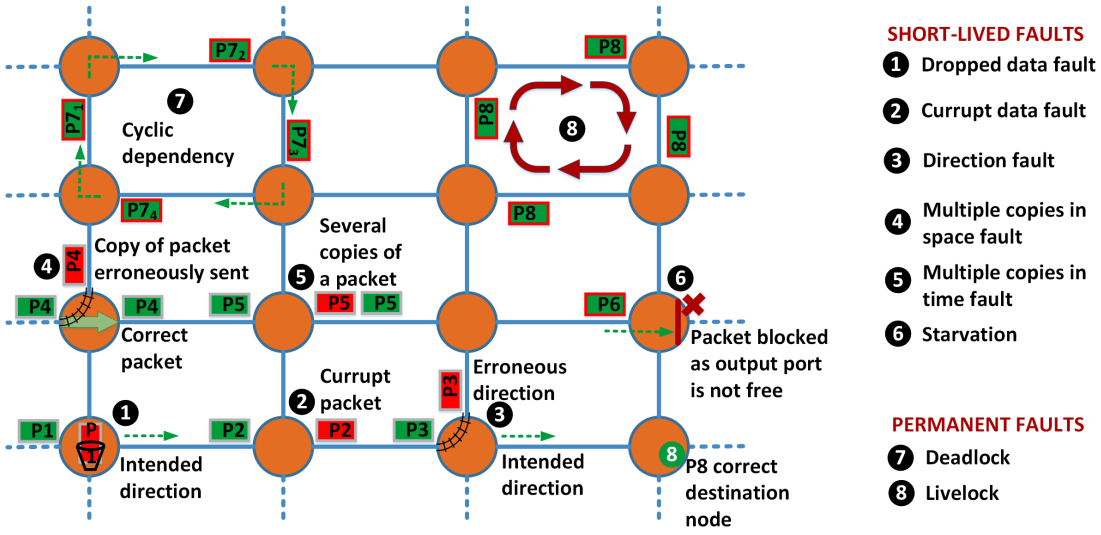


Figure 2.3: Different data communication faults on an NoC.

wireless channel among all the WIs. At a particular instant, the channel is assigned to a single WI for a fixed number of cycles, so that the conflict and contention on the wireless medium are avoided.

2.2 NoC Fault Model

NoC being an interconnect module, reliable and efficient data transfer is its main functionality. Anomalies in packet routing operation introduce several data communication faults to NoC. So, to evaluate the functional correctness of NoC, communication-centric validation methods are popularly adopted [32]. This section presents a fault model that illustrates commonly found data communication faults on an NoC [33, 9], which are illustrated in Figure 2.3. These faults can be of two different types, such as short-lived fault and permanent fault. Moreover, this section also indicates the probable buggy component(s) within the NoC router for which a particular fault may occur. The following subsections provide a brief discussion on these network faults.

2.2.1 Short-lived Faults

Short-lived faults exist on the network for a short span of time. These faults may occur due to bugs in different functional units of NoC. Even though the visibility period of such faults is small on the network, they can badly impact system performance. Moreover, if the functional bugs that are the root-cause of such faults remain unaddressed, the fault may affect a critical packet or multiple data packets, leading to a catastrophic situation. As an example, data drop is a short-lived fault. Let's say a particular input port of an NoC router is buggy, and the result is that any packet coming to that port is getting dropped. This may happen that memory access packets

for a critical application may get dropped, leading to significant performance degradation. Therefore, the root-cause of such faults should be carefully identified and debugged. This subsection discusses the commonly observable short-lived network faults.

2.2.1.1 Dropped Data Fault (DDF)

Whenever a packet drops midway before reaching its destination, the error is known as Dropped Data Fault (DDF). In this case, the packet is received at an input port of a router but is never forwarded out of its intended output port. Such fault may occur in a VC buffer or routing unit. For instance, a buggy head counter pointing to the first location of a VC buffer would increase by 2 or more each time a flit is removed. This will lead to multiple flit-drops. Moreover, an erroneous routing unit may remove the flit from the VC but would not issue an appropriate select signal for the output multiplexer. In this case, the flit will never come out of the output port and will be lost inside the router. DDF is visible on the network until the actual arrival time of the packet at its destination.

2.2.1.2 Corrupt Data Fault (CDF)

Whenever the actual information carried by the packet is altered, the fault is known as a Corrupt Data Fault (CDF). Such fault may occur due to error in the VC buffer. An error in one or more bits of buffer memory would change the data value from ‘0’ to ‘1’ or vice versa. This would change the payload information if any of the body flits is corrupted, or may change the destination if the head flit is corrupted. Such fault is active till the time the destination node consumes the packet.

2.2.1.3 Direction Fault (DF)

If a packet is sent to an output port other than the intended one, then the fault is known as Direction Fault (DF). Such fault occurs due to a bug in the routing unit or Crossbar component. If the RC module inside the routing unit decodes a wrong output port from the destination address, the packet flits would be transferred to an unintended output port. Similarly, a fault in the Crossbar module may forward a flit in a wrong direction. This type of fault exists in the network until the packet traverses on the unintended path.

2.2.1.4 Multiple Copies in Space Fault (MCSF)

Due to a buggy routing unit, it may happen that the select signal of multiple output multiplexers may get activated. Thereby, a flit would be forwarded to multiple output ports. Such fault is known as Multiple Copies in Space Fault (MCSF). The fault can be experienced on the network till the time all the flits taking the wrong paths are live.

2.2.1.5 Multiple Copies in Time Fault (MCTF)

Multiple Copies in Time Fault (MCTF) may originate from the VC buffer in the router. For instance, if the *empty* signal of a VC is faulty, it may happen that multiple copies of the same flit are being received from an upstream router. Though all the copies of the flit are taking the correct path, all of them are not intended to be on the network. Such a fault can be realized until the unwanted packets are on the network.

2.2.1.6 Starvation

A flit experiences starvation, when it does not get an NoC resource to move forward for a long time. Such type of fault may appear due to faulty arbitration mechanism in VA and/or SA modules. In general, whenever the VC waiting time of a flit exceeds beyond an allowable threshold, the anomaly is considered as starvation. This fault exists until the flit gets access of the resource to move forward.

2.2.2 Permanent Faults

A permanent fault exists on the NoC forever once it occurs, and continues to thwart the normal operation of the network. We discuss two commonly observed permanent faults on NoC such as deadlock and livelock.

2.2.2.1 Deadlock

Deadlock occurs whenever a cyclic dependency among packets is developed, as shown in Figure 2.3. This happens when all the packets wait for each other to release the resource in a cyclic manner, but none of them proceeds. Such type of scenario may occur due to error in VC allocator. Deadlock is a permanent fault and the affected packets keep on waiting in their respective VC forever.

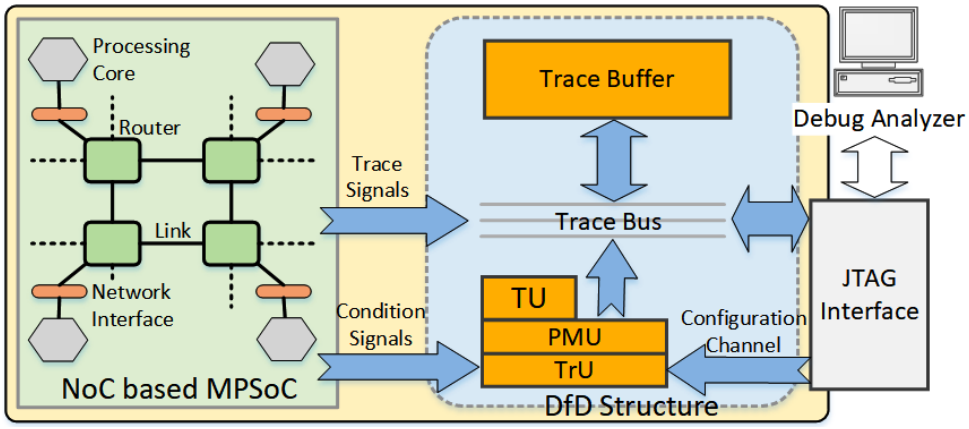


Figure 2.4: Trace based NoC debug platform.

2.2.2.2 Livelock

Livelock is another permanent fault, where the packet keeps on traversing across network routers, but never reaches its destination, as demonstrated in Figure 2.3. This may happen due to error in the routing unit, or Crossbar module.

The above discussion indicates the probable erroneous NoC component(s) that may result in a data communication fault. Therefore, in a communication-centric validation framework, whenever such a fault is detected and localized, the root-cause of the bug can be found by evaluating the suspected NoC components.

2.3 NoC Post-Silicon Debug Framework

Post-silicon debug framework enables to capture signal states during system run, and analyze the same for fault detection and localization. A trace-based debug platform inherited from [34, 35] is demonstrated as debug framework for NoC in Figure 2.4. As shown in the figure, an NoC-based multiprocessor system is considered as the Circuit Under Debug (CUD). The integrated DFD hardware provides the suitable infrastructure to observe and capture the NoC behaviour during normal operation. The off-chip debug analyzer examines the behavioral traces of the interconnect by running debugger software, and detects any existing network fault. Upon any fault detection, the localization process is carried out. Finally, the root-cause of the fault is diagnosed and fixed.

In Figure 2.4, it can be observed that the DFD structure includes an embedded trace buffer, a Trigger Unit (TrU), a Packet Monitor Unit (PMU), a Timestamp Unit (TU), a trace bus, and a JTAG interface. Trace buffer is an on-chip memory, where the run-time traces can be stored. TrU invokes the tracing operation, PMU builds the packet traces, and TU assigns the timestamp to individual traces. Trace bus provides the communication

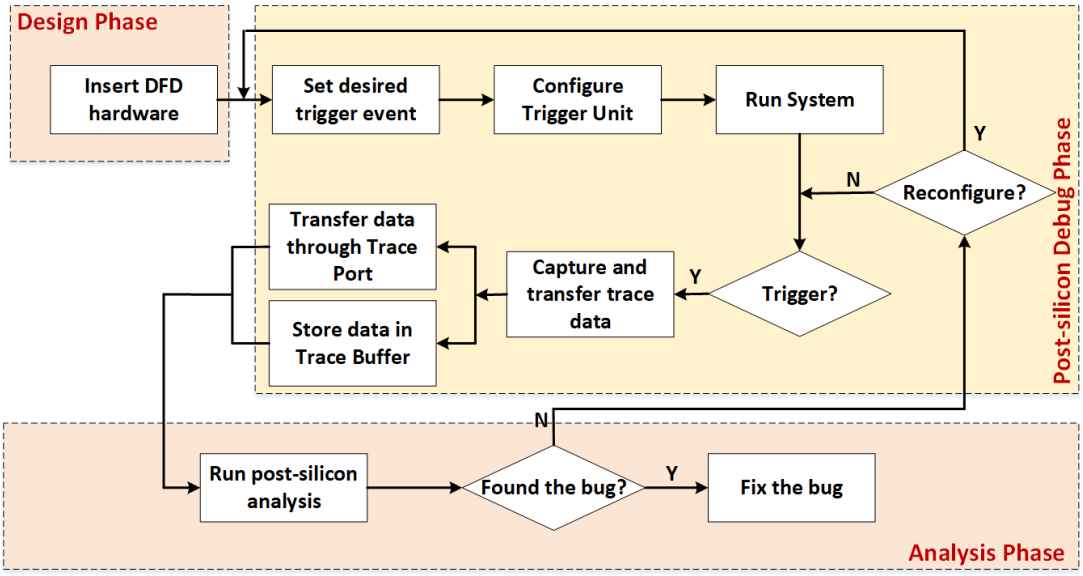


Figure 2.5: Debug flow during traced-based post-silicon validation.

medium for trace transfer. JTAG interface is there to configure the TrU (through debug access port (DAP)) and export the traces (through trace port) to the external debug analyzer.

The cost of debug hardware is majorly dependent on the size of the trace buffer [10]. This restricts the number of observable points, frequency of tracing, and the width of observation window during the debug phase. Trace buffer width decides the number of observable points, and its depth decides the width of the observation window. By varying the frequency of trace capture, the width of the observation window can also be varied. As the observable points are limited, it is a challenging task to decide which signals to be traced among the plethora of signals in a design. This is known as *Signal Selection*, and is performed during design time. Multiple researchers have proposed several efficient signal selection techniques that can provide greater visibility to the system's internal states [35, 36].

Figure 2.5 presents a generalized flow of trace-based post-silicon validation process. At the beginning of the debug phase, the TrU is configured through the DAP. Generally, the JTAG interface is used as the DAP. The TrU is configured so that signal traces can be captured whenever pre-set trigger events are detected. A trigger event can be expressed in terms of certain state conditions that are transferred by the condition signals from the CUD to the TrU. The efficiency of bug localization depends upon the event detection capability and amount of traced data [13]. The event detection capability depends on how well a trigger event is specified, and its relevance to a particular fault. Whereas, the amount of traced data is driven by the size of the trace buffer. The captured traces are either streamed to the off-chip analyzer via the trace port during run-time or stored in the on-chip trace buffer for later analysis. Due to the trace bandwidth limitation and low-speed I/O interface,

most of the debug framework follows the store and forward trace mechanism. In this mechanism, the stored traces in the trace buffer are accessed and transferred to the debug analyzer via the JTAG interface later. During post-silicon debug analysis, the traces are checked for any anomalous behavior. Upon the detection of any fault, the root-cause is investigated and fixed. If no fault is detected, then more traces are collected for the same or a reconfigured trigger setup.

As discussed earlier, a communication-centric debug is adopted for NoC post-silicon validation. Therefore, the packet movement is traced during NoC debug. The TrU is configured based on trace capture frequency. For short-lived faults, the capture frequency is kept high, whereas it can be kept low for permanent faults. The NoC validation process can broadly be divided into three steps namely Packet Trace Collection (PTC), Trace Data Transfer (TDT), and Fault Analysis (FA).

2.3.1 Packet Trace Collection

Trace collection is the most important step in post-silicon debug. In an NoC, payload data is transported in the form of packets. Thus, capturing packet traces can provide a globally consistent view of data communication across the interconnect. The following subsections discuss monitoring infrastructure for NoC-based system, and the process of packet trace collection.

2.3.1.1 NoC Monitoring Infrastructure

To capture packet movement activities, monitoring units are traditionally deployed at different interface levels of NoC as a significant part of debug infrastructure. Such monitors collect transaction-level packet traces, which are then analyzed to detect any anomaly in packet communication. Many research works propose such monitoring infrastructure for NoC that enhances the debug process of whole SoC [37, 38, 39, 40]. Figure 2.6 shows a basic monitoring infrastructure for transaction-based communication-centric system debug. The *read* and *write* transactions of processing cores initiate activities on interconnect. These transactions are fragmented into packets that communicate through the network. Based on the placement in the NoC architecture, the monitors can be broadly classified as transaction monitors and communication monitors. Transaction monitors are placed in between the processing core and NI. It inspects the transactions and detects either missing transactions or transactions with incorrect attributes. On the other hand, communication monitors residing between the routers observe the packet communication to detect faulty packets and paths.

Authors in [37] presents a monitoring infrastructure to carry out the debugging of the interactions among the

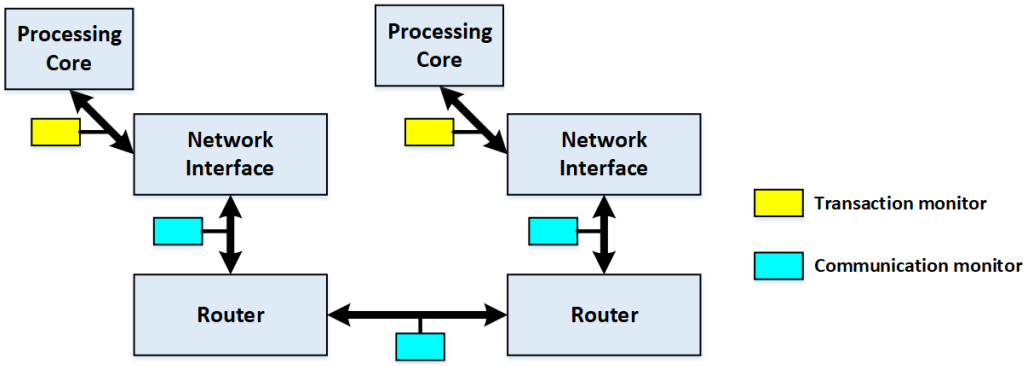


Figure 2.6: Monitoring infrastructure for transaction-based communication-centric NoC debug.

embedded processors, along with system performance analysis. An event-based NoC monitoring framework is proposed in [38] that integrates hardware probes to NoC's NI ports. The infrastructure provides run-time observability of NoC behavior and supports system-level and application debugging. In another work [39], a system-level debug agent is provided along with the hardware probes integrated between the processing cores and the NIs. The debug agent delivers several in-depth analysis features of an NoC-based system such as NoC transaction analysis, multi-core cross-triggering, and global synchronized timestamping for more effective debugging. Authors in [40] propose to integrate monitors and filters to NoC, which observe and filter transactions at run-time. This work also implements programmable finite state machines in the debug unit to validate the correct relation of transactions at run-time.

The above-mentioned proposals integrate monitoring infrastructure to NoC interfaces for providing SoC level debug solutions. Additionally, there are proposals [9], which embed monitors inside each NoC routers to provide dedicated NoC debug solutions. Such in-router monitoring setup can observe and capture the packet traces within the router. These packet traces can be analyzed to find any data communication fault discussed in Section 2.2. Upon the detection of any fault, the packet traversal path can be reconstructed using the corresponding packet traces, and thereby the buggy functional unit of the NoC can be localized. The following subsection discusses the process of packet trace collection using the in-router monitoring infrastructure.

2.3.1.2 Process of Trace Collection

During the packet trace collection phase, routing status of all the packets inside each router node is monitored. Periodic snapshots of packet states are captured and stored as traces in the trace buffer. The trigger unit is configured such that it can repeatedly generate the trigger signal for trace capture after a fixed time unit known as Snapshot Interval (SI). A packet trace is constructed by concatenating packet information extracted from its header flit, packet's routing status, and timestamp value. Figure 4.4 shows the pattern of individual packet

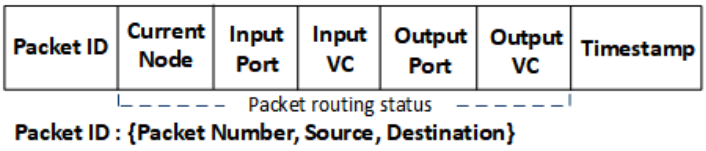


Figure 2.7: Individual packet trace pattern.

trace. Packet ID in the trace is a piece of static information and acts as the identifier of the corresponding packet. Packet number along with source and destination information from the header flit constitute the unique packet ID for each packet. At every core, a packet number is produced and embedded into the header flit whenever a packet is generated. Other than packet ID, the trace of a packet contains multiple dynamic information such as timestamp, current node, port and VC number. These entities change according to the packet routing status. Once the packet traces are collected, they are transferred via trace bus to the trace buffer for storage.

2.3.2 Trace Data Storage and Transfer

Both trace buffer and interconnection fabric are important components of DFD structure. DFD interconnection modules perform the trace data transfer both on-chip and off-chip, whereas trace buffer stores these trace data on-chip for further analysis.

2.3.2.1 Trace Storage

Trace buffer is an on-chip memory, which stores run-time traces for debugging purpose. One row of the trace buffer stores a single trace of all the traced signals. The depth of the trace buffer decides the number of traces that can be stored for a single signal. Therefore, the total amount of trace data that can be collected during a single post-silicon validation run is limited by the storage capacity of the on-chip trace buffer. To efficiently use the limited storage space in the trace buffer, the authors in [41] have proposed distributed as well as dynamic allocation of trace buffers at run-time. Moreover, authors in [9] have proposed a debug structure for NoC sub-system, where L2 cache of each processing core is effectively reused as local trace buffer of corresponding router node. To deal with the trace buffer storage constraint while maintaining the system's internal observability, several trace reduction techniques such as state restoration and trace compression are proposed in the literature. Special state restoration mechanisms [35, 36, 42] can enhance the amount of traced information by data expansion. This allows less number of observable signals to be selected without any visibility loss, which leads to a smaller trace buffer requirement. Moreover, trace compression techniques can reduce the total amount of trace by 20-30% [19, 20, 43].

2.3.2.2 Trace Transfer

DFD interconnection includes trace bus and JTAG interface for transfer of trace data on-chip and off-chip respectively. Trace bus provides a dedicated routing path for trace data transfer to trace buffer and/or to trace port. Standard solutions typically use pipelined multiplexer (MUX) trees to transfer traces [11, 44]. But, such MUX tree-based interconnection incurs high area overhead to provide increased trace bandwidth. To overcome this limitation, the authors in [45] propose a two-level interconnection network consisting of a MUX network and a non-blocking concentration network for trace transfer. In the case of NoC-based system debug, there is an opportunity to reuse the same NoC infrastructure for transferring the traces. Authors in [9] have reused the same NoC (which is the CUD) for on-chip trace transfer. Conventionally, a JTAG interface is used to transfer the trace data to an off-chip debug analyzer for trace analysis [46].

2.3.3 Fault Analysis

During the fault analysis phase, the collected traces are analyzed for fault detection, identification, root-causing, and localization. The root-cause of faulty behaviour needs to be located both in space and time. Bug localization being the most important step in debug [10], it requires extensive packet path reconstruction. In the case of an NoC debug process, the individual packet paths are reconstructed using packet ID and timestamp values once the traces are collected in the debug analyzer. All the traces related to a particular packet are extracted using the unique packet ID assigned to it. Then the traces corresponding to each packet are stitched together according to the increasing timestamp values. Thus, the paths traversed by every packet are reconstructed. The percentage of packet path reconstruction depends upon the amount of traces present for the respective packet. Each packet path is examined by several fault detection algorithms based on the faults for which the design is getting evaluated. Trace collection, transfer, and analysis phases are iterated multiple times till all the functional bugs are identified and located.

2.4 Conclusion

This thesis targets the problems related to NoC debug. Therefore, this chapter provides the background on NoC architecture and operation along with the details of basic debug infrastructure and trace-based debug flow. Moreover, this chapter discusses the possible communication faults such as short-lived and permanent faults on a network. The next three chapters, in this thesis, discuss different challenges associated with the post-silicon debug of NoC based multicore systems, as highlighted in Chapter 1 and the proposed solutions.

Chapter 3

Efficient Post-Silicon Debug Structures for NoC Based Systems

The complexity levels of modern interconnect modules are very high. Evaluating and maintaining the functional correctness of such modules are a real challenge. This leads to a situation, where multiple functional bugs on the interconnect might escape through the pre-silicon verification stage. Even though the processing cores function correctly, bugs on NoC can introduce various data communication faults. The faults on interconnect can lead to several system-level problems, such as functional failure, reliability issue, violation of power, performance constraints, etc. This necessitates a robust post-silicon debug framework for developing fault-free NoC modules. The traditional store and forward trace-based debug methods encounter the problems of large trace buffer requirement and limited availability of trace communication bandwidth. These constraints become more stringent for short-lived network faults (direction fault, dropped data fault, etc.), which demand more frequent trace collection for their detection and localization. This chapter discusses a Wireless enabled NoC Debug framework (WiND) that addresses the trace buffer size and trace communication bandwidth issues associated with the conventional trace-based debug methods. Moreover, WiND includes Redundant Trace Elimination (RTE) to further minimize the trace buffer constraint.

3.1 Motivation and Contribution

The proposed WiND framework [47, 48] uses Wireless Interfaces (WIs) to achieve an efficient trace-based post-silicon debug infrastructure for NoC. Recent research works show, adding WIs to baseline NoC (WNoC) improves the performance of SoC [4, 49]. In this work, WIs are integrated to the NoC in the same way as

WNoC, but they are used during NoC debug for both trace reduction and high-speed trace transfer. Wireless enabled NoC platform reduces the intermediate hop-count between source to destination by performing long-range data communication over the wireless medium. To demonstrate this, a simulation for *Random* traffic has been performed on a 2D mesh network of different sizes with varied wireless capabilities. Without loosing any generality, the latency at both wired and wireless nodes is considered as the same (one hop) for this simulation. It can be observed from the shown result in Figure 3.1 that average life of the in-flight packet reduces notably in a WNoC in comparison to wired NoC. Thus, WiND setup can lead to less trace generation for the same amount of test payload during NoC debug. WiND also replaces the traditional interface used for exporting traces to the debug analyzer with high bandwidth WI. This significantly increases the speed of trace data transfer.

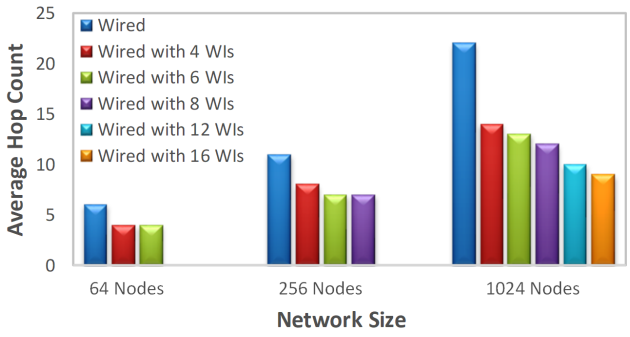


Figure 3.1: Reduced average hop-count for wireless enabled network.

NoC is a shared medium, and thus congestion on an NoC is a frequent phenomenon. This provides another opportunity to further reduce the total trace on an NoC. Traces of internal signals are captured periodically, once after each Snapshot Interval (SI). If the state of a particular signal does not change frequently, then with a small SI, multiple redundant traces get generated for the same signal. This scenario can easily be visualized for an NoC. Figure 3.2 shows the normalized actual maximum packet latency to ideal maximum packet latency for different workloads at their respective saturated Packet Injection Rate (PIR). An 8x8 baseline 2D-mesh NoC is modeled on Noxim (a cycle accurate network simulator) [50], and the workloads are executed on this platform to get the latency values. From the figure, it can be observed that in an average, the actual maximum latency on the network is 16.2 times of ideal maximum latency. This indicates that packets in network routers get stuck at different pipeline stages for multiple additional cycles than desired. This happens due to congestion in the network, which grows with increasing PIR value. During NoC validation, packet states are captured and stored as traces for all the packets present inside the routers [9]. Because of the fact that multiple packets do not change their state for several cycles, many redundant packet traces are generated for a small SI. Therefore, WiND performs redundant trace elimination (RTE) [51] that can distinguish these redundant traces and keep only one copy from each redundant groups. As a result, one instance of all unique trace is collected and thus, the system observability is maintained.

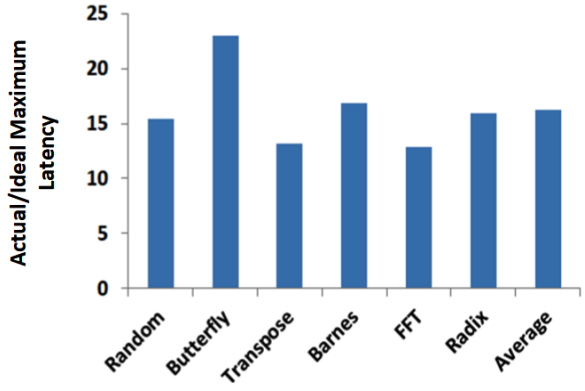


Figure 3.2: Actual maximum packet latency normalized to ideal maximum packet latency.

3.2 Background and Related Work

Various research works have developed monitoring platforms to improve the run time observability of the packet communication inside NoC [37, 39, 52]. Debug probes are inserted in between cores and their NIs in [39], whereas NoC monitors are attached to routers as well as to master-slave interfaces in [37, 52] to keep track of communication transactions. Though these methods provide high-level transaction information over the network, they do not discuss about using such information for detecting particular NoC bugs and localizing them. A recent work [9] has developed a complete NoC debug platform that instruments in-flight network traffic monitoring to detect bugs, and performs packet path reconstruction to localize them. This technique stores the runtime transaction traces of the packets traversing through the network routers in the local trace buffers. Whenever a local trace buffer is filled with traces, the network execution is halted, and a local check is performed in the attached core for fault detection. With no fault detected, the corresponding trace buffer is flushed. Upon the detection of a fault, traces from all the trace buffers are transported to an on-chip central debugger for analyzing the root cause of the bug. This method, however, does not discuss on the trace buffer size and trace communication bandwidth issues. Moreover, this method frequently interferes with the normal operation of the system during the local checks and analysis phase, resulting in degraded debug speed.

The implementation cost of an integrated debug system depends largely on the size of trace buffer [10]. Several methods such as signal selection [13, 14, 34, 35] and trace compression [19, 20, 43, 53] are proposed to reduce the level of trace during debug. However, these two techniques increase design complexity in-terms of the implementation of restoration and compression engine respectively. Also, they do not guarantee whole system states restoration. Speed of debug is another challenge associated with post-silicon validation [7, 17]. Exporting large amount of traces to off-chip analyzer introduces substantial latency overhead to the debug process. To deal with this; in [9], a centralized on-chip debug analyzer is instrumented that eliminates exporting

Table 3.1: List of Acronyms used for WiND Framework

Acronym	**Full Form**	**Acronym**	**Full Form**
AOF	Aggregate Objective Function	GSP	Global Snapshot Period
GTDT	Global Trace Data Transfer	HB	Header Buffer
LTDT	Local Trace Data Transfer	MAP	Memory Access Packet
RTE	Redundant Trace Elimination	SB	Snapshot Buffer
SI	Snapshot Interval	ST	Snapshot Time
TCB	Trace Channel Buffer	WiND	Wireless enabled NoC Debug

traces off-chip. But, the on-chip analyzer introduces large area and power overhead. The proposed framework (WiND) focuses both on trace reduction and quick trace transfer to achieve faster as well as efficient NoC debug. Moreover, unlike [9], WiND uses an off-chip debug analyzer for fault analysis, and ensures negligible interference with the normal execution of the network.

The intra-chip wireless communication, in case of WiND, reduces the trace amount by decreasing the average life of test payload on the network. This allows to capture more transaction-level traces with a smaller interval between two consecutive snapshots for the same trace buffer size. Frequently collected traces enable the framework to detect the short-lived network faults efficiently. After the trace collection, usually a dedicated trace bus is used for transferring traces to the trace buffer or to the output port. Authors in [11, 45] present special interconnection design for trace transfer. WiND reuses the existing network resources for intra-chip trace communication to reduce the debug overhead. However, while sharing the NoC bandwidth for both normal payload and trace transfer, the proposed method takes special care to minimize the interference with the normal execution of the system. The proposed scheme replaces the traditional JTAG interface used for off-chip trace transfer with a wireless interface. Inter-chip wireless communication exports the collected traces with very high data rate to the external debugger, leading to faster debug.

This chapter includes several acronyms for ease of writing. Table 3.1 lists all the acronyms used specifically for the WiND framework. The common acronyms used in the whole thesis can be referred from the "List of Acronyms" at the beginning of this thesis.

3.3 WI-Based NoC Post-Silicon Debug

This work establishes an efficient post-silicon debug framework for NoC (WiND) by using WIs in the debug platform. The whole network is divided into multiple clusters, and each cluster is augmented with one WI. WiND differs from the traditional debug platform broadly on the basis of five modifications, as shown in Figure

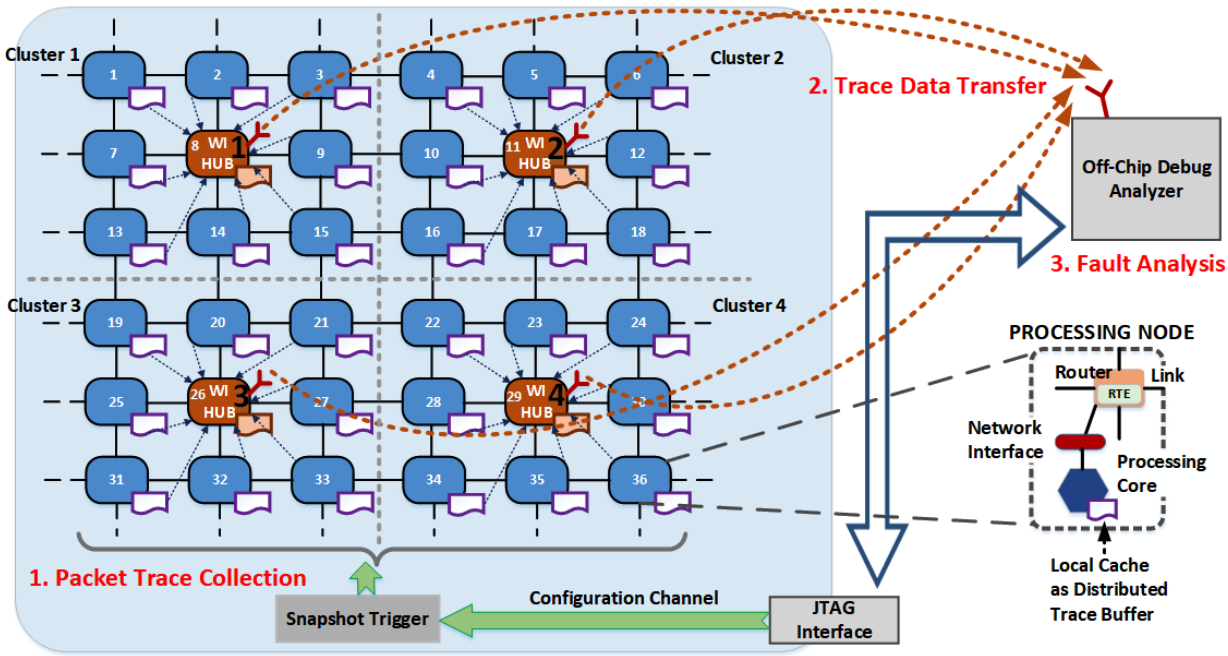


Figure 3.3: WiND: Wireless enabled NoC Debug platform.

3.3. These are (i) provision of test data communication through the wireless medium for trace reduction, (ii) incorporation of RTE scheme for further trace reduction, (iii) reuse of local cache memory of each core as distributed trace buffer, (iv) reuse of NoC infrastructure with modified router architecture for on-chip trace transfer, and (v) use of wireless links for exporting traces to the off-chip debug analyzer for increasing the bandwidth of trace communication. The NoC debug process is divided into three phases, such as: (i) packet trace collection (PTC); (ii) trace data transfer (TDT); and (iii) fault analysis (FA). The embedded DFD structure captures snapshots of the transaction level packet information from the NoC. These snapshots are considered as packet traces, which give a globally consistent view of the communication among multiple cores on the system.

3.3.1 Packet Trace Collection in WiND

During the PTC phase, each router takes the snapshot of its own state after each SI and stores the traces in the trace buffer. In this phase, traces can be collected concurrently at all the routing nodes in every cycle, which will provide complete transaction level details of each packet traversing through the network. But this generates a humongous amount of trace data that are difficult to store, transfer, and analyze. Hence, to control the frequency of packet tracing, the SI is regulated by configuring the trigger unit (TrU). This configuration depends on the type of suspected fault and the network node. For short-lived faults, the interval is kept low, whereas for permanent faults, it can be set with a high value. The wireless hubs (WI HUBs: nodes with WI in Figure 3.3), being the critical nodes in the proposed framework, are set with as low as 1 cycle SI to ensure enough visibility.

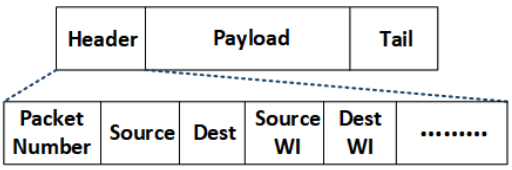


Figure 3.4: Format of packet and header flit content in WiND framework.

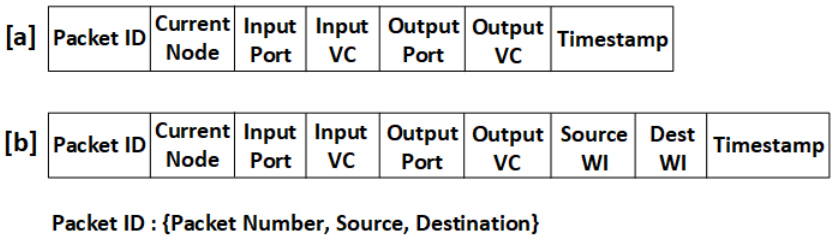


Figure 3.5: Complete packet trace pattern. (a) trace pattern for wired node, (b) trace pattern for WI HUB.

However, this does not overburden the available trace buffer, as the number of wireless hubs in the network is less, and the total wireless utilization is small with respect to the whole network communication. In contrast, SI of baseline wired nodes is controlled according to the available trace buffer space.

Trace buffer, an on-chip memory meant for storing the trace data, is used only during post-silicon debug and has no use thereafter. Hence, the size of the trace buffer is usually kept small [54]. The proposed method utilizes the local L2 cache of each core as distributed trace buffer, rather than having a dedicated one [9]. A specified portion of the local caches are temporarily used to store traces during debug phase and are released for normal operations after debug. The nodes not connected with a core are provided with dedicated trace buffer.

Traces are extracted from the router state and are stored in the associated trace buffer. Router state represents the packet information of all the packets present in the router at an instant. Packet information is composed of header flit data (packet ID and WI information) and its routing status (current node, port, and VC information). Figure 3.4 shows the format of the packet and header flit in the WiND framework. Each WI is assigned with a unique WI address. The packet header holds the WI addresses of both source and destination WI, along with the addresses of the actual source and destination node. The packet number is another information carried by the header flit, that combined with source and destination address forms a unique packet ID for the corresponding packet. Unused space in header flit can be used to hold packet number, source and destination WI. Patterns of complete packet trace at both wired node and wireless hub are shown in Fig. 3.5. For a wired node, packet ID is collected from the header flit, whereas the current node, port, and VC information are collected from the packet status inside the router. The physical timestamp at which the snapshot is taken is added by the timestamp unit (TU) to form a complete packet trace. For a packet in a wireless hub, the packet trace contains WI source and destination addresses fetched from header flit in addition to all other entries present in the case of a wired node.

NoC being a shared medium, many a time, packets keep on waiting for free resources in intermediate routers. This leads to no change in packet state for the waiting period, and snapshots taken during this period are considered redundant. Such redundant traces can be eliminated in the debugger by running a software program, or in the router node itself by introducing a small hardware change as discussed in sub-section 3.3.4. We have implemented the Redundant Trace Elimination (RTE) hardware that eliminates the redundant traces and keeps only one unique copy of the trace from each redundant group. This benefits in reducing the trace amount before trace storage and transfer. Moreover, our method collects an additional trace of each in-flight packets at the frequency of a large period (Global Snapshot Period (GSP)). These traces are retained regardless of redundancy or not for easy detection of permanent faults like deadlock, and livelock (shown in Algorithm 1). During the PTC phase, whenever any of the local trace buffers is filled, TDT phase gets started.

3.3.2 Trace Data Transfer in WiND

Trace data needs to be transferred to the debug analyzer for fault analysis. In the traditional method, a dedicated trace bus is used for on-chip trace communication [55], whereas a low bandwidth JTAG interface [46] is used for off-chip trace transfer. In the proposed method, we are reusing the NoC infrastructure for trace communication. Traces stored in the trace buffer of each router get transferred to a wireless hub using a wired NoC path. The incoming traces at each wireless hub are sent to the off-chip debug analyzer through the high bandwidth inter-chip wireless communication as shown in Figure 3.3.

TDT step is divided into two phases, such as Local Trace Data Transfer (LTDT), and Global Trace Data Transfer (GTDT). Whenever a trace buffer associated with a router node gets filled up, the LTDT phase starts, and the traces in the local trace buffer are transferred via wired path till the wireless hub of the corresponding cluster. During this period, normal execution of all the routers on the trace travel path is halted, and in-flight packets within those routers are paused in their VCs. Once, the trace transfer is over, these routers resume their operation. This way, the normal execution of other routers on the network are not affected, and the whole network operation is not required to be halted. Since the interference between debug and normal operation is minimized, this provides us two benefits such as (i) normal execution of the system nearly replicates in-field operation, and (ii) it adds very low overhead to debug latency. The LTDT phase is repeated whenever there is a trace buffer overflow on the network. The WI present in the cluster exports the incoming traces through a wireless channel to the external debug analyzer. GTDT phase starts after a pre-defined large time period. During this phase, traces from all the trace buffers on the network are exported to the debug analyzer. This is performed to collect complete network traces before starting the fault analysis phase. During the GTDT phase, the whole network is halted, and all the in-flight payload packets on the network are paused in the VCs of

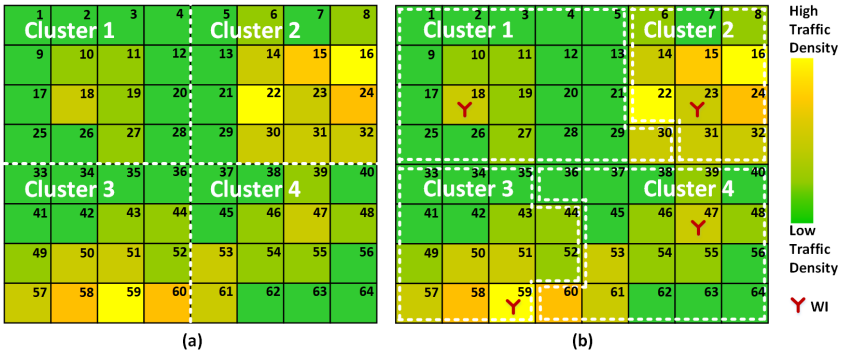


Figure 3.6: Network Clusters. (a) Equi-node, (b) Equi-load.

routers. Our proposed method provides a Trace Channel Buffer (TCB) inside each router (explained in subsection 3.3.3) that transfers the on-chip trace data without disturbing the halted payload packets in the VCs, both in LTDT and GTDT phases. Once the trace transfer is over, the debug analyzer starts analyzing the traces. In the CUD, trace buffers are flushed, timestamp counters are reset, and network operation as well as trace collection are resumed.

Clustering of the whole network. Clustering of the network is performed for efficient trace data communication. Each cluster is provided with a WI, which is responsible for exporting the traces of that cluster to the external debugger. For efficient trace transfer during the GTDT phase, an equal distribution of trace load across all the clusters is expected, which leads to parallel completion of trace transfer. Figure 3.6 (a) demonstrates a simple homogeneous network clustering that equally divides the total number of nodes per cluster. However, such clustering does not guarantee uniform traffic load distribution, as well as trace generation across the clusters. This is due to the nature of applications, the presence of heterogeneous cores, and the placement of memory controllers. For an example case, the average traffic density of each node of a 8x8 system is shown in Figure 3.6 found from simulations performed for six different applications (refer Table 3.2). For this, we executed the applications for 100,000 cycles and observed the total number of packets travelling across each of the 64 router nodes. It can be observed from the figure that the traffic density is not uniform over the network. For this purpose, an equi-load network clustering is performed to ensure near-equal traffic load on all the wireless hubs. Figure 3.6 (b) shows an example of such equi-load network clustering. Such clustering decisions can be taken during design time for a targeted set of applications, and specific hardware platform. Furthermore, it can also be observed from Figure 3.6 (b) that not all the nodes experience the same amount of traffic load within a cluster. For example, 18 in Cluster1 is the busiest node, similarly 59 in Cluster2, and so on. It is apparent that busy nodes must be generating a larger amount of traces, forcing their local trace buffers to overflow frequently. The traces in such trace buffers need to be transferred quickly to the debugger during LTDT so that they can be available for further trace collection. For this, optimal placement of the WI within a cluster is performed in

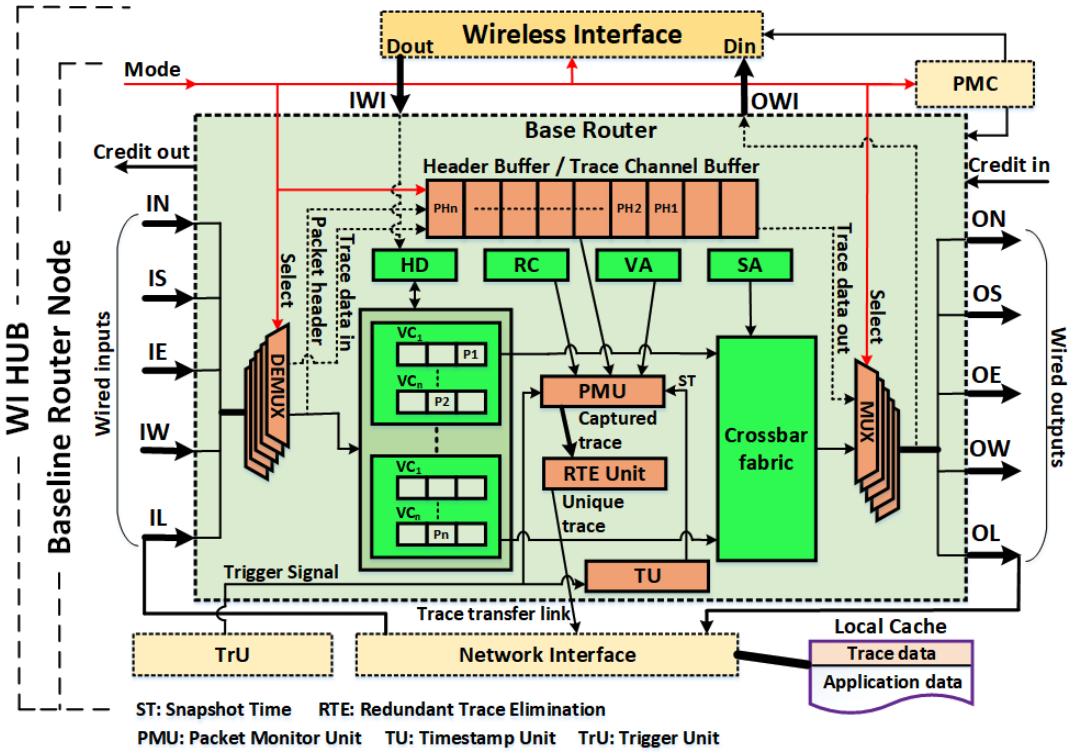


Figure 3.7: A modified router architecture for efficient trace data collection and communication.

close proximity to the busy nodes, as illustrated in Figure 3.6 (b) and discussed in sub-section 3.3.5.

3.3.3 Router Architecture and Control Circuitry

The router architecture in WiND is modified as shown in Figure 3.7 to realize the proposed method. Each router is equipped with an extra buffer space. The buffer behaves as a Header Buffer (HB) during the PTC phase to store the copies of headers of all the packets present in the router node. The same buffer behaves as a TCB during the TDT phase to transfer the traces inside the router.

During the PTC phase, whenever a packet header reaches an input port of a router, it is temporarily stored in an input VC and its copy is transferred to HB as shown in Figure 3.7. Further, the packet moves towards the desired output port through multiple pipeline stages, namely Header Decoder (HD), Routing Computation (RC), VC Allocator (VA), and Switch Arbiter (SA). The PMU shown in the figure continuously monitors the HB, RC, and VA units. As soon as the TrU sends the trigger signal, the PMU captures the snapshots of packet states and builds the packet traces for all the packets present inside the router. The packet states are the header information present inside HB and the routing status of the packet present inside RC and VA. TU provides the time of trace capture. RTE module eliminates the redundant traces, and the unique packet traces including the respective timestamps are sent to the local trace buffer (L2 cache) through a dedicated trace transfer link.

At the end of the PTC phase, the content of HB is cleared, so that the buffer can be used as TCB for trace communication during the TDT phase. With the beginning of the next PTC phase, the normal operation gets resumed and the header information of the packets in the router's VCs are fetched back to the HB.

During TDT phase, traces stored in the trace buffers, need to be transported to the wireless hub of the corresponding cluster. A fixed destination in every cluster allows the system to bypass the HD and RC pipeline stages during trace transfer. TCB is used as an express channel inside the router, which carries the traces to the output port without disturbing the halted packets inside the VCs. A credit-based flow control mechanism is used to forward the trace packets through the TCB from one node to the next downstream node. Since the trace packets do not take the packet-switched path, VA and SA pipeline stages are eliminated. Thus, the proposed method prioritizes traces to bypass all the routing pipeline stages and provides a fast on-chip trace communication path inside a router.

A mode signal is provided to drive a router to either the PTC phase or TDT phase. This signal acts as the select line for demultiplexers at the input side and for multiplexers at the output side, as shown in Figure 3.7. During the PTC phase, the packets coming on the input ports are considered as the payload data and are pushed to the respective VCs through the demultiplexer. Similarly, at the output side of the router, the payload data from the crossbar fabric get selected through the multiplexer, to be transferred to the output port. During the TDT phase, the incoming packets are considered as trace data. The mode signal indicates the demultiplexer to redirect the trace packets to the TCB. It also acts as the select line to the multiplexer at the output end, which chooses the TCB content to be forwarded to the output port.

3.3.4 Redundant Trace Elimination

The RTE unit implemented inside the router is able to save a significant amount of trace buffer space in an NoC-based system by discarding the redundant traces. Figure 3.8 demonstrates the structure and operation of both periodic trace collection, and redundant trace elimination. Whenever the PMU collects a new packet trace *Snapshot(t)* (Figure 3.8 [a]), it is compared with the corresponding packet trace *Snapshot(t-SI)* previously stored in the Snapshot Buffer (SB) (Figure 3.8 [b]). If both the traces are different, then the new trace *Snapshot(t)* would be overwritten in the SB. The output of the SB would be enabled to export the new trace through the trace transfer link to the trace buffer. Otherwise, if both the traces are the same, then the previously stored trace *Snapshot(t-SI)* will be retained in the SB. No trace will be transferred to the trace buffer. This ensures the system to discard all the redundant traces and store only the unique ones. Thus, the RTE mechanism reduces the total trace amount without degrading the network internal observability.

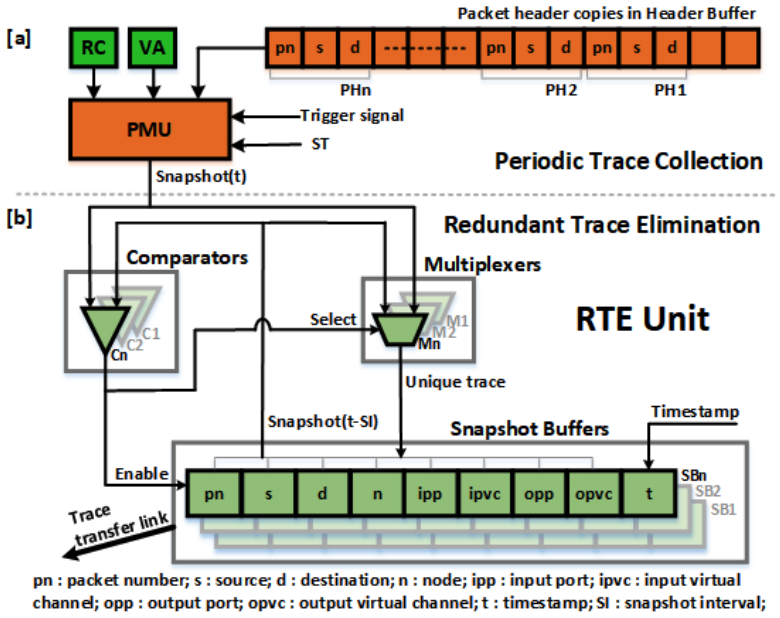


Figure 3.8: Periodic trace collection and redundant trace elimination.

3.3.5 Wireless Interface

In the proposed validation framework, WIs are used for both on-chip and off-chip data communication. The WiND structure augments WIs to a few baseline routers that are known as wireless hubs (WI HUBs in Figure 3.3). Baseline routers are interconnected using wired links, whereas the wireless hubs communicate with each other using the wireless channel.

We have used On-Off Keying (OOK) based on-chip mm-wave transceiver as WI, adopted from [21, 56], which is shown in Figure 3.9. The primary components of the transmitter circuit (T_X) are up-conversion mixer, and power amplifier, whereas the receiver circuit (R_X) consists of a low noise amplifier, down-conversion mixer, and baseband amplifier. A voltage controlled oscillator is used to generate the carrier frequency (f_0) of 60 GHz. The WIs are used for on-chip test data transfer during PTC phase and off-chip trace data transfer during TDT phase. Data amplification power required in both the phases are different, as the communication range changes. Hence, to meet the power demands of both intra and inter-chip communication, a variable gain power amplifier [57] is used. This can provide two different amplification levels depending on the data communication range, and can swiftly switch between both the levels. The mode signal is given to the power amplifier that indicates when to switch the amplification level. An on-chip omni-directional antenna at each wireless hub is used to transmit and receive data through the wireless channel. To keep the proposed debug structure simple, a single wireless channel is shared between all the WIs during data communication [58]. Each of the WIs gets access to the channel periodically for a fixed amount of time. Thus, at a particular instant, only one transmitter is

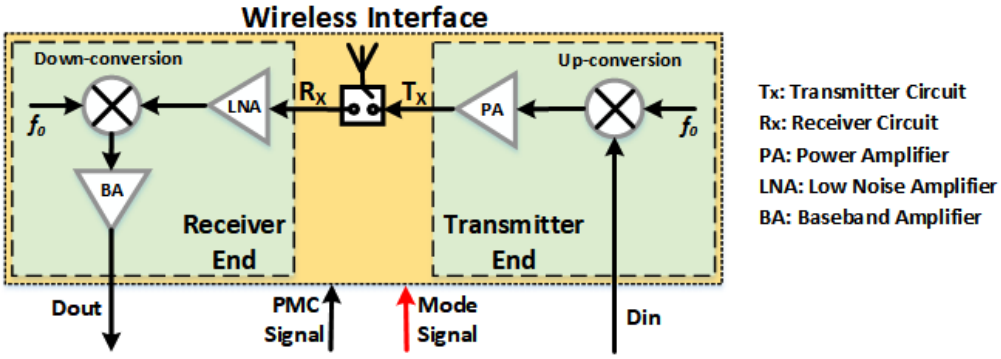


Figure 3.9: OOK based transceiver.

active. The benefit of such single-channel wireless infrastructure is considerably low area and power overhead. Moreover, this allows to power gate the inactive WI transmitters to save a considerable amount of static power.

Optimal number of WIs. To determine the optimal number of WIs in a network, we adopted a similar approach used in [21]. For this, we consider two metrics that account for the *less trace generation* and *cost of the DFD structure*. The first metric is proportional to the *total shortest path*, μ between all pairs of nodes, that can be calculated as shown in Equation 3.1.

$$\mu = \Sigma\, h_{ij} * f_{ij} \tag{3.1}$$

Here, h_{ij} is the shortest distance in-terms of hop-count between ith source and jth destination nodes. The frequency f_{ij} of communication between the ith source and jth destination is computed based on the traffic interactions between the nodes that depend on the applications mapped onto the NoC.

The second metric, that is *cost of DFD structure*, can be calculated using Equation 3.2.

$$Cost(\#ofWI) = A + P \tag{3.2}$$

Here, A and P are area and power respectively, arising from the WIs. An Aggregate Objective Function (AOF) is formed using the two metrics μ and *cost*. This function is then used in a *Simulated Annealing* [31] based optimization process, where the number of WIs (n) is swept through 1 to N. Here, N is the total number of router nodes in the NoC. Finally, an optimal number of WIs, n is selected that results in minimum value of the AOF. In our experiments, we find n value as 4 and 6 for network size of 64 and 256 nodes respectively for the applications mentioned in Table 3.2.

Optimal placement of WIs. As discussed in subsection 3.3.2, a WI is optimally placed so that it can be at close proximity to the busy nodes within the cluster. This can be determined based on a metric *total shortest path in the cluster*, $\mu_{cluster}$ between each node of the cluster and the corresponding wireless hub, that can be calculated as shown in Equation 3.3.

$$\mu_{cluster} = \Sigma\, h_{iWI} * f_{iWI} \tag{3.3}$$

Here, h_{iWI} is the distance in-terms of hop-count between the ith source of the cluster and the corresponding wireless hub. The frequency f_{iWI} of communication is computed based on the traffic interactions between both the nodes that depend on the number of trace buffer overflow of ith node. $\mu_{cluster}$ is computed for the placement of the WI at each node location within the cluster, and the minimum $\mu_{cluster}$ value decides the final location of the WI.

3.3.6 Fault Analysis in WiND

Upon the collection of traces, the debug analyzer processes them for fault detection, identification, and localization. Bug localization being the most important step in debug [10], it requires extensive packet path reconstruction. Detailed packet traversal paths can be reconstructed with the availability of frequent traces from all the routing nodes. Our method transfers traces from all the nodes to the debug analyzer for the whole test period. This enables our framework to provide high system observability during fault analysis. Fault like deadlock depends upon the traffic condition of both local and neighboring router nodes. Unlike the other existing NoC debug method [9], where fault analysis is performed individually in each node, WiND collects the traces from all the nodes and performs the fault analysis at an external debug analyzer. Thus, WiND provides greater visibility for the faults like deadlock.

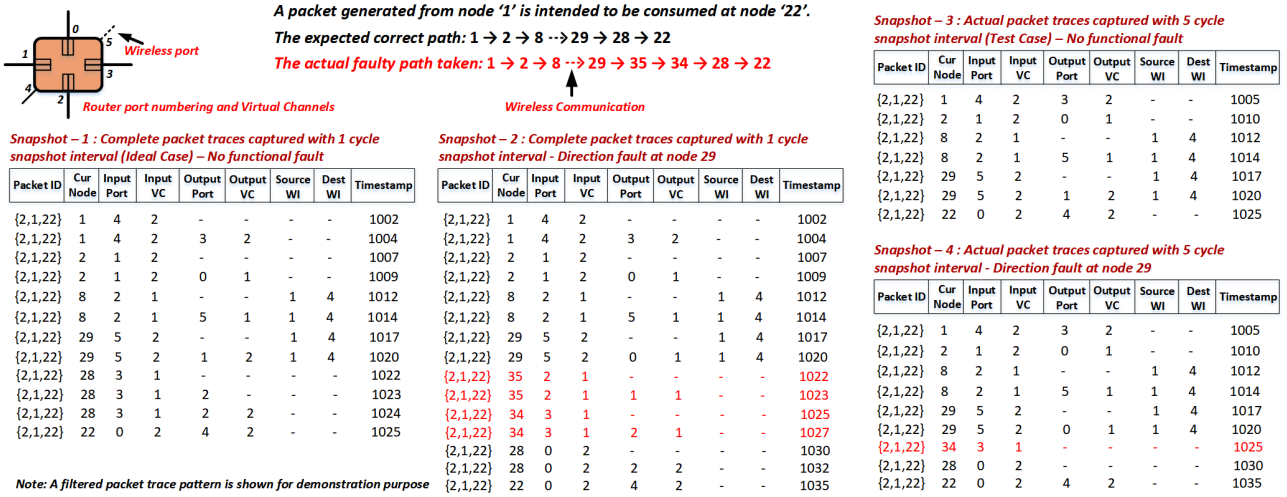
During the FA phase, all traces related to a particular packet are extracted using the unique packet ID and are arranged according to the timestamp values for path reconstruction. A fault detection and localization algorithm is presented in Algorithm 1, which can take this path as input to make a decision on the occurrence of any functional fault. This algorithm is capable of detecting either of the six NoC faults such as Dropped Data Fault (DDF), Direction Fault (DF), Multiple Copies in Space Fault (MCSF), Multiple Copies in Time Fault (MCTF), deadlock, and livelock that are discussed in Section 2.2. If the time difference between two consecutive traces are larger than the GSP value and still both the traces are same (no change in packet state), then deadlock is detected. However, when the timestamp of two consecutive traces are the same, then there are

two copies of the same packet on the network. The fault is MCTF, if both copies of the packet are on the desired path, else it is a MCSF. In case, the algorithm finds the last trace for a packet, but the trace does not indicate the desired destination node, then it flags a DDF. The algorithm detects a DF, if the trace is collected from a node other than the nodes on the desired path. In the earlier case, if the time difference between the trace timestamp and the packet start time is larger than GSP, then the fault is detected as livelock.

Algorithm 1 Fault Detection and Localization Algorithm for WiND Framework

```
1:  initial: Pre-computed time period: GSP                    ▷ GSP: Global Snapshot Period
2:
3:  Packet Path Fault Analysis:
4:  Src = GetSource(Packet)
5:  Dest = GetDestination(Packet)
6:  PST = FirstTrace timestamp                                ▷ PST: Packet Start Time
7:  for every packet trace do
8:      time = trace timestamp; Node = trace Current Node
9:      if (time − prev_time) >= GSP then
10:         if trace == prev_trace then
11:             DetectFault(Deadlock)
12:         end if
13:     else if (time − prev_time) == 0 then
14:         if Node && prev_node InPath(Src, Dest) then
15:             DetectFault(MCTF)
16:         else
17:             DetectFault(MCSF)
18:         end if
19:     else
20:         if LastTrace && (Node != Dest) then
21:             DetectFault(DDF)
22:         else if Node ! InPath(Src, Dest)
23:             if (time − PST) >= GSP then
24:                 DetectFault(Livelock)
25:             else
26:                 DetectFault(DF)
27:             end if
28:         end if
29:     end if
30:     if DetectFault then
31:         FaultLocation = Return(Node)
32:     end if
33:     prev_trace = trace
34:     prev_time = trace timestamp
35:     prev_node = Node
36: end for
```

Algorithm 1 also indicates the probable location of the fault, as shown in line number 31. The indicated fault location might be the source of the fault or a location to which the fault has already propagated (the shorter



Packet ID	Cur Node	Input Port	Input VC	Output Port	Output VC	Source WI	Dest WI	Timestamp
{2,1,22}	1	4	2	-	-	-	-	1002
{2,1,22}	1	4	2	3	2	-	-	1004
{2,1,22}	2	1	2	-	-	-	-	1007
{2,1,22}	2	1	2	0	1	-	-	1009
{2,1,22}	8	2	1	-	-	1	4	1012
{2,1,22}	8	2	1	5	1	1	4	1014
{2,1,22}	29	5	2	-	-	1	4	1017
{2,1,22}	29	5	2	1	2	1	4	1020
{2,1,22}	28	3	1	-	-	-	-	1022
{2,1,22}	28	3	1	2	-	-	-	1023
{2,1,22}	28	3	1	2	2	-	-	1024
{2,1,22}	22	0	2	4	2	-	-	1025

Packet ID	Cur Node	Input Port	Input VC	Output Port	Output VC	Source WI	Dest WI	Timestamp
{2,1,22}	1	4	2	-	-	-	-	1002
{2,1,22}	1	4	2	3	2	-	-	1004
{2,1,22}	2	1	2	-	-	-	-	1007
{2,1,22}	2	1	2	0	1	-	-	1009
{2,1,22}	8	2	1	-	-	1	4	1012
{2,1,22}	8	2	1	5	1	1	4	1014
{2,1,22}	29	5	2	-	-	1	4	1017
{2,1,22}	29	5	2	0	1	1	4	1020
{2,1,22}	35	2	1	-	-	-	-	1022
{2,1,22}	35	2	1	1	1	-	-	1023
{2,1,22}	34	3	1	-	-	-	-	1025
{2,1,22}	34	3	1	2	1	-	-	1027
{2,1,22}	28	0	2	-	-	-	-	1030
{2,1,22}	28	0	2	2	2	-	-	1032
{2,1,22}	22	0	2	4	2	-	-	1035

Packet ID	Cur Node	Input Port	Input VC	Output Port	Output VC	Source WI	Dest WI	Timestamp
{2,1,22}	1	4	2	3	2	-	-	1005
{2,1,22}	2	1	2	0	1	-	-	1010
{2,1,22}	8	2	1	-	-	1	4	1012
{2,1,22}	8	2	1	5	1	1	4	1014
{2,1,22}	29	5	2	-	-	1	4	1017
{2,1,22}	29	5	2	1	2	1	4	1020
{2,1,22}	22	0	2	4	2	-	-	1025

Packet ID	Cur Node	Input Port	Input VC	Output Port	Output VC	Source WI	Dest WI	Timestamp
{2,1,22}	1	4	2	3	2	-	-	1005
{2,1,22}	2	1	2	0	1	-	-	1010
{2,1,22}	8	2	1	-	-	1	4	1012
{2,1,22}	8	2	1	5	1	1	4	1014
{2,1,22}	29	5	2	-	-	1	4	1017
{2,1,22}	29	5	2	0	1	1	4	1020
{2,1,22}	34	3	1	-	-	-	-	1025
{2,1,22}	28	0	2	-	-	-	-	1030
{2,1,22}	22	0	2	4	2	-	-	1035

Figure 3.10: Walk-through example for DF detection and packet path reconstruction from the available traces.

the snapshot interval, the closer will be the captured fault location to the source of the fault).

Figure 3.10 demonstrates a DF case and its analysis for a particular packet by a walk-through example. The example considers a packet that has taken a wireless path to reach its destination. The second packet generated from node 1 is intended to be consumed at node 22 as shown in the figure. Thus, the packet ID assigned to the packet is {2,1,22}. For this example, the node and the wireless hub numbers can be referred from Figure 3.3, whereas the router port number can be viewed in the top left corner of Figure 3.10. This example considers the router with two VCs at each port. *Snapshot 1* and *Snapshot 3* in the figure show the fault-free packet traces collected for XY routing with snapshot interval of 1 cycle and 5 cycles respectively. The individual trace shows the exact status of the packet at the time of snapshot. The blank output port and/or output VC in few cases indicate that the corresponding resources are not allocated at the time of snapshot. Source and destination WIs are captured only at the wireless hubs. It can be noticed that *Snapshot 3* holds all the traces related to wireless communication as *Snapshot 1*. This is because wireless hubs are triggered to take the snapshot at every cycle to ensure the detection of any fault on the wireless path. *Snapshot 2* and *Snapshot 4* exhibits the traces for the same above packet with a DF at node 29 for snapshot interval of 1 cycle and 5 cycles respectively. Rather than moving to node 28 from 29, the packet moves to node 35 due to the fault. In this example case, the fault detection algorithm in the debug analyzer compares the traces in *Snapshot 4* with the correct traces in *Snapshot 3*. The DF is detected from the seventh trace in *Snapshot 4* indicated by a different node than the intended one.

Percentage of path reconstruction can be estimated from Equation 3.4. A packet trace collected from a wired node contains five routing components such as current node, input port, output port, input VC, and output VC. These five components of all the intermediate routers constitute the complete path of the packet.

The path also includes WI source and destination information, if the packet has taken any wireless path to reach its destination. Since the wireless hubs are captured in every cycle, there is no loss in information regarding the source and destination WI. But many a time, the wired nodes are captured with large snapshot interval as discussed earlier. This leads to incomplete path statistics with respect to the five wired components of each intermediate router. In the case of a mesh structure with XY routing, if the input port through which a packet has entered a node is known, then the information about the upstream node and the upstream node's output port can be extracted. This postulate put forward the following expression for calculation of percentage path reconstruction.

$$Percentage\ path\ reconstruction = \frac{(\Sigma\ input\ ports) * 3 + \Sigma\ input\ and\ output\ VCs}{total\ nodes\ in\ complete\ path * 5} \tag{3.4}$$

For an example, we have considered *Snapshot 4* from Figure 3.10 for calculation of percentage path reconstruction. Comparing with *Snapshot 2*, we find that *Snapshot 4* does not have any information regarding intermediate node 35. In its complete traversal, the packet covers eight nodes, whereas the collected trace gives information about seven nodes. Output port and VC information are also missing for a few of the existing nodes. For this example, Equation 3.4 returns an estimation of 82.5% path reconstruction for the corresponding packet.

Benefits of external debug analyzer. The use of an external debug analyzer reduces the on-chip DFD overhead. Most importantly, it minimizes the interference between debug and normal operation of the network. *ChipScope* from *Xilinx* [59] and *CoreSight* from *arm* [60] are examples of such industry-standard external debug analyzers. Being an external system, an off-chip debugger can be highly computation efficient for quick fault analysis and can have large memory space for acomodating an abundant amount of traces that would help in detailed path reconstruction.

3.3.7 Power Management of WiND

On-chip DFD structure is an integral part of design architecture. This allows the extension of DUT's power control mechanism for the power management of DFD structure. The Power Management Controller (PMC) deployed for baseline NoC router (Figure 3.7) can be used for augmented WI and other DFD components with some additional control. Normally, during in-field operations, DFD components are not used and become

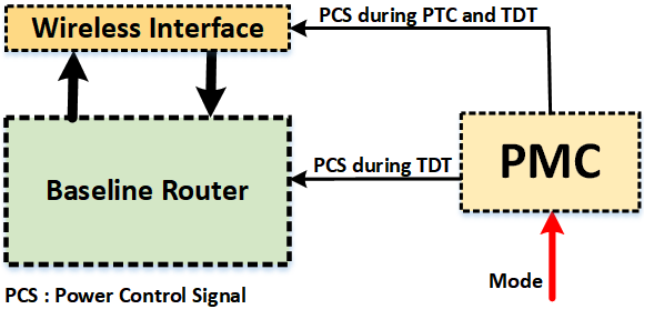


Figure 3.11: Power management of WiND.

vestigial. Hence, after the debug phase, such debug modules can be permanently power gated if they are not reused for any architectural purpose (few reuse cases are discussed in Section 3.5). During debug, the mode signal indicates PTC and TDT phase, and accordingly the power controls are changed as discussed below.

During the PTC phase, router components are active for test data communication, and debug components are active for trace collection. The WIs on the network are used for long-range test data transfer. However, a single wireless channel is shared between all the WIs for data transfer to minimize the overhead of debug structure as discussed in sub-section 3.3.5. Such wireless infrastructure keeps only one transmitter active at a time. The power control signal is supplied to WIs during PTC as shown in Figure 3.11, which can power gate the inactive transmitters to reduce the static power consumption.

During the TDT phase, the normal operation of either a few (during LTDT) or all (during GTDT) router nodes are paused. The in-flight payload packets in the associated routers are paused in the VCs of routers. Though the VCs are idle, power gating them would flush the payload data paused inside them. Therefore, PMC provides the scaled-down retention voltage to these VCs that drives them into drowsy mode to save power, while retaining the stored data at the same time. During this phase, traces are transferred through the TCB, bypassing all the pipeline stages. This allows the PMC to power gate all the router pipeline hardware, and the debug components such as PMU, TrU and TU (Figure 3.11). Only except the active WI transmitter, all others are power gated, similar to the PTC phase. During GTDT, all the WI receivers are power gated as traces are only transmitted out of the DUT. WiND adopts the technique presented in [61, 62] to reduce the wireless-hub energy by using sleep transistor based power-gated transceivers.

3.4 Results and Analysis

This section illustrates the experimental setup to evaluate the WiND platform and discusses the results. The multi-core interconnect fabric is modeled on cycle-accurate Noxim simulator [50]. We have considered 8x8

Table 3.2: Network Topology and Simulation Setup

Component	Configuration
Topology and Size	2D mesh NoC (4x4 with 2 WIs; 8x8 with 4 WIs; and 16x16 with 6 WIs)
Router	5 I/O wired ports, 1 I/O wireless port (for wireless hub), 2 virtual channels per port, 8 flit buffers, 8 flit packets, 32 bit flits
Wireless Link	60 GHz carrier, 16 Gbps bandwidth, single cycle latency
Workload	Synthetic - *Random*, *Transpose*, *Butterfly* SPLASH-2 - *Barnes*, *FFT*, *Radix*

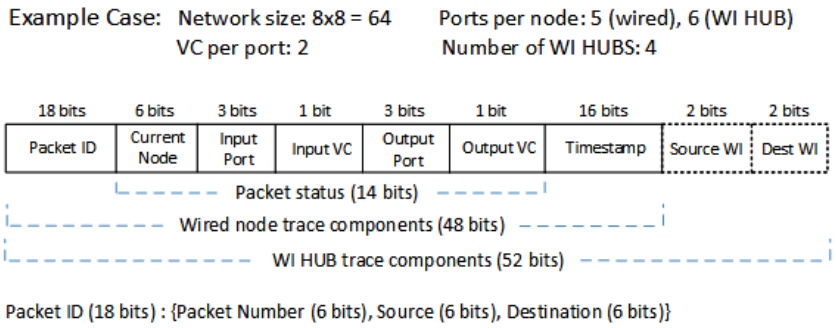


Figure 3.12: Packet trace size for a 8x8 mesh network.

baseline 2D mesh NoC with 4 wireless hubs, and deterministic XY routing for generating results. All cores and NoC wired modules are operated at a clock frequency of 1 GHz. The details of the network topology and simulation setup are shown in Table 3.2. Simulations are performed for 3 synthetic workloads and 3 applications from SPLASH-2 benchmark suite [63]. Full system simulation of SPLASH-2 applications is executed on Graphite simulator [64] to generate the network traces for the Noxim platform.

Few of our results are compared with [9], which is a state-of-the-art NoC debug solution. For the experiments, each node is allocated with 30KB memory from the attached L2 cache for trace storage similar to [9]. This leaves more than 90% of L2 cache space free for normal operation. In our proposed framework, only NoC is considered faulty, whereas cores, caches, and cache controllers are assumed to be bug-free. The router module in the network simulator is modified to capture periodic snapshots and to inject different faults.

3.4.1 Trace Amount

In this section, we have computed the total amount of trace generated on the WiND platform for the whole simulation period and compared the same with an equivalent wired debug platform. A single trace captured at a wired router node is 48 bits as shown in Figure 3.12. Trace of a wireless communication at wireless hub adds 4 more bits of WI address values on top of the wired trace and consumes a total of 52 bits. A network simulation

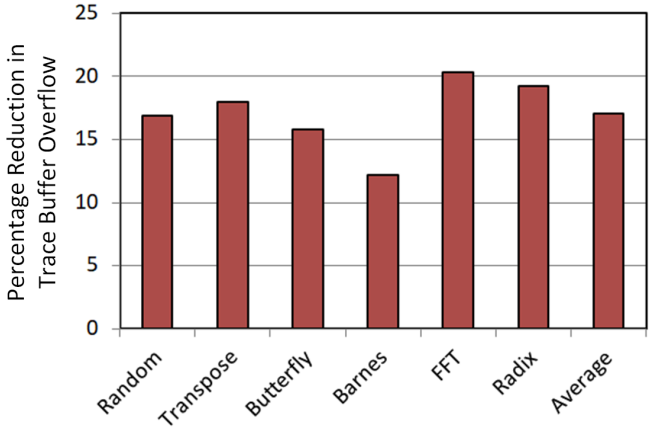


Figure 3.13: Percentage reduction in trace buffer overflow in WiND over [8].

is performed for 100K *Uniform Random* packets, and traces are captured once at each hop during a packet transmission. It is observed that a wired-only network generates 3.43MB of total trace, whereas the WiND platform generates only 2.38MB of total trace. The resultant trace reduction is because of the average life of the packet is considerably less in the wireless network than the wired-only network (shown in Figure 3.1). To highlight the role of wireless infrastructure in trace reduction, we deactivated the RTE module of WiND during the simulation. In the real scenario, RTE mechanism further reduces the trace amount. The contribution of RTE in trace reduction is discussed in sub-section 3.4.6.

3.4.2 Trace Buffer Overflow

Due to the disparity nature of network traffic, the trace buffers at different nodes fill up unevenly, and buffers associated with busy nodes would overflow frequently. Whenever there is a buffer overflow, network operation gets interrupted, leading to higher debug time. We observed that for the whole simulation period, the total number of trace buffer overflows reduces considerably (around 17%) in case of WiND compared to [9], as shown in Figure 3.13. This is because of two reasons such as 1) less trace generation in WiND and 2) wireless test data communication in case of WiND bypasses multiple intermediate busy router nodes. Less number of trace buffer overflows followed by non-interfering trace transfer (discussed in sub-section 3.3.2) result in faster execution in case of WiND.

3.4.3 Trace Data Transfer

To evaluate the efficiency of trace data transfer, we have divided the complete process into two steps. These are (i) on-chip communication, where the stored traces are transferred from the trace buffer till the I/O port, and

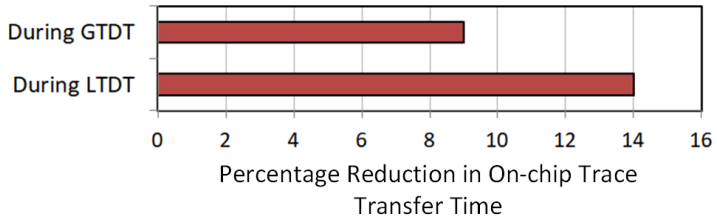


Figure 3.14: On-chip trace transfer efficiency.

(ii) off-chip communication for exporting the traces till the external debug analyzer. Traditionally, the former is accomplished through wired links till the JTAG port, and later the traces are exported again through wired interconnect till the off-chip analyzer. In the proposed WiND framework, trace data stored in distributed trace buffers are transferred using the NoC wired links till the on-chip wireless port of the nearest wireless hub, and then through wireless link till the off-chip debug analyzer. First, we evaluate the efficiency of WiND on the basis of off-chip trace communication and compare the same with the traditional wired framework. An average of 54% packet latency reduction in case of wireless inter-chip communication has been sown in [65] compared to wire-based silicon interposer communication. In terms of bit-rate capacity, the proposed wireless interface can export traces with 16Gbps data rate. Whereas a JTAG based ARM CoreSight debug and trace connector (ULINK*pro*) provides up to 800 Mbps of trace communication speed [60]. This ensures that the proposed framework is a suitable solution for fast off-chip trace transfer.

To enhance the on-chip trace communication, equi-load clustering and optimal WI placement are proposed in sub-section 3.3.2. Prioritized trace transfer through TCB inside router nodes further enhances the on-chip trace communication. WiND shows 14% and 9% reduction in on-chip trace transfer latency during LTDT phase and GTDT phase respectively in comparison to baseline 8x8 with 4 WIs systems. The baseline system's configuration includes equi-node clustering, placement of WI at the center of each cluster, and packet-switching transfer of traces inside the routers. The simulations are performed for 100,000 cycles for all six applications considered in this work, and the average results are shown in Figure 3.14. In case of WiND, the improvement during LTDT is majorly achieved due to the optimal placement of WIs that facilitates short travel path for the traces of the busy nodes that experience frequent trace buffer overflow. The improvement during GTDT is because of the equi-load clustering that results in parallel completion of the trace transfer from all the clusters.

3.4.4 Fault Detection

This sub-section evaluates the fault detection capability of the proposed WiND framework. Separate simulations each of 100,000 cycles are executed for 100 randomly injected faults from each fault type. Traces are

Table 3.3: Comparison of WiND with equivalent Wired Debug Setup

% Improvement	**Workload**	***DDF***	***DF***	***MCSF***	***MCTF***
Fault Detection	Random	27	36	13	27
	Transpose	11	25	14	13
	Butterfly	33	25	20	22
	Barnes	21	22	15	26
	FFT	18	17	10	34
	Radix	30	18	17	32
	Average	**23.3**	**23.8**	**14.8**	**25.7**
Path Reconstruction	Random	34	38	25	35
	Transpose	22	33	25	29
	Butterfly	39	48	33	39
	Barnes	31	34	30	30
	FFT	31	26	24	35
	Radix	17	24	26	28
	Average	**29**	**33.8**	**27.2**	**32.7**

collected with different SI for each injected fault, and the fault detection capability of WiND is evaluated by running the Algorithm 1 on the collected trace set. It is observed that with increasing SI, the percentage of short-lived fault detection degrades significantly. This is because the lifespan of such faults is very small on the NoC and snapshots after large intervals are unable to capture most of the traversal information of faulty packets. Hence, trace collection with very small SI is required for better short-lived fault detection. In this context, the WiND platform allows for smaller SI with the available room to accommodate more traces. Thus, the proposed validation framework shows considerable improvement over the wired counterpart in terms of short-lived fault detection and path reconstruction (discussed in sub-section 3.4.5) for same trace buffer size as demonstrated in Table 3.3. Permanent faults like deadlock and livelock can be detected even if the SI is very high, because of their perpetual nature on the network.

We have also evaluated WiND in-terms of fault detection reliability, as the framework reduces the amount of traces. One-hop wireless communication of long-range packets in WiND results in eliminating the traces for the intermediate hops that are bypassed. As a result, the corresponding packet traces will not carry the fault information related to the bypassed router nodes, which otherwise the traces would have carried in case of a wired platform. This will lead to degradation in fault detection reliability of WiND compared to wired counterpart for packet traces captured with same SI value as shown in Figure 3.15 (a). The result shows around 90% packet fault detection for all the short-lived faults in case of WiND. This 10% degradation is because of the absence of the packet traces for the router nodes which were bypassed due to wireless communication. In contrast, permanent faults are better detected as multiple nodes are involved in manifesting a single permanent

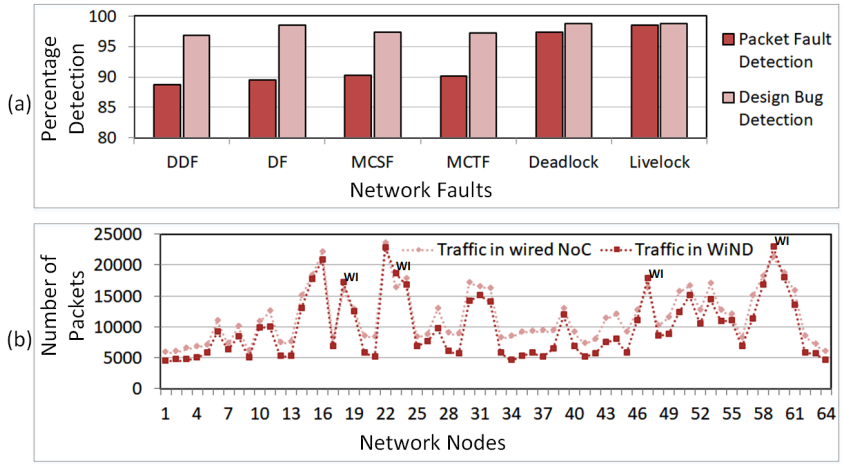


Figure 3.15: (a) Fault and bug detection reliability of WiND, (b) Traffic per node.

fault. However, post-silicon debug is performed to capture and localize the functional design bugs, which cause the packet communication faults. A detected fault provides indications to the root-cause functional bug. It is evident that a particular functional bug in a router module affects multiple packets traversing across the same router node. So, it can be comprehended that in case of WiND, a particular functional bug can be indicated at-least by the traces of short-range packets taking the wired path and traversed across the corresponding router node. Figure 3.15 (b) indicates the average traffic across all the router nodes for both wired and WiND framework simulated for all the six applications on an 8x8 NoC. It can be observed, even in case of WiND, each of the router nodes experiences a sufficiently large number of packets traversing across it. This results in high bug detection capability of the proposed debug framework, as shown in Figure 3.15 (a).

WiND framework performs an offline FA, resulting in delayed fault detection. However, real-time detection of faults is not the primary concern during post-silicon debug. This is because post-silicon debug is performed before mass production and not during in-field operation. Moreover, its primary objective is to detect and remove the existing design bugs. In this regard, percentage as well as reliability of fault detection are considered to be important. Thus, WiND framework is not evaluated on-the basis of timely fault detection.

3.4.5 Path Reconstruction

Percentage path reconstruction is the fraction of the route taken by a packet that can be rebuilt from the available traces. Reconstructed paths of erroneous packets help in localizing the faults on the network. In our experiments, we have checked the path reconstruction capability of WiND and compared it with [9] for three common faults considered in both the works such as DF, deadlock, and livelock. These faults are named as misroute-1, deadlock, and livelock1 respectively in [9]. Network simulations are run for saturated PIR, and

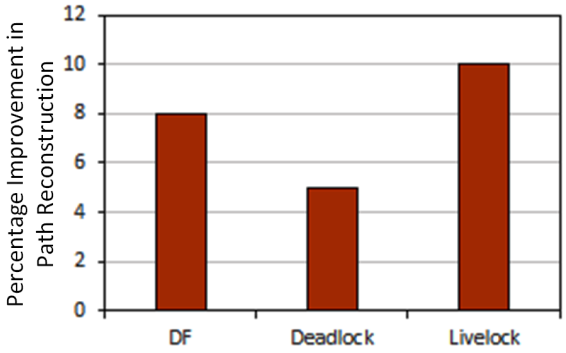


Figure 3.16: Percentage improvement in path reconstruction in WiND over [8].

traces are collected with SI of 10 cycles. For WiND, we have used Equation 3.4 for calculating percentage path reconstruction. Figure 3.16 demonstrates that WiND is better in terms of path reconstruction than [9]. This is because [9] discards the fault-free traces after each local check phase. However, since both the methods retain all the captured faulty traces, percentage fault localization for both the cases can be assumed to be the same. But, the smaller trace volume in the case of WiND allows it to lower the SI and capture more frequent traces for the same trace buffer size. This will enable WiND to reconstruct more detailed packet routing paths. Table 3.3 demonstrates such improved path reconstruction results for considered short-lived faults.

3.4.6 Trace Reduction by RTE Scheme

RTE hardware eliminates the redundant traces and thus reduces the total amount of trace generated. Figure 3.17 (a) demonstrates the fault detection and path reconstruction for *Random* traffic in case of DDF. It shows that with increasing SI, both percentage of fault detection and path reconstruction degrades significantly. So, to enhance the fault detection, the TrU needs to be configured with very small SI. Thereby, the PMU can capture the packet traces frequently and provide more visibility to the packet transactions. However, it generates huge amount of traces. In such case, the proposed RTE scheme considerably reduces the total trace amount, as shown in Figure 3.17 (b). Simulations are performed with and without RTE structure for all the workloads at their respective saturated PIR on a fault free network. It is observed that RTE scheme outperforms for small SI. This is because more redundant traces are generated when the packet snapshots are captured with small SI, and RTE structure can successfully discard them. Trace reduction capability of the RTE technique is also tested for different fault conditions, and the results are shown in Figure 3.17 (c). Faults are injected on 10% of total *Random* traffic payload and snapshots are captured per one cycle. Each fault category is simulated separately. In case of no fault condition, RTE scheme achieves around 36% of trace reduction for *Random* traffic and one cycle SI, as shown in Figure 3.17 (b). Results in Figure 3.17 (c) shows almost similar outcome

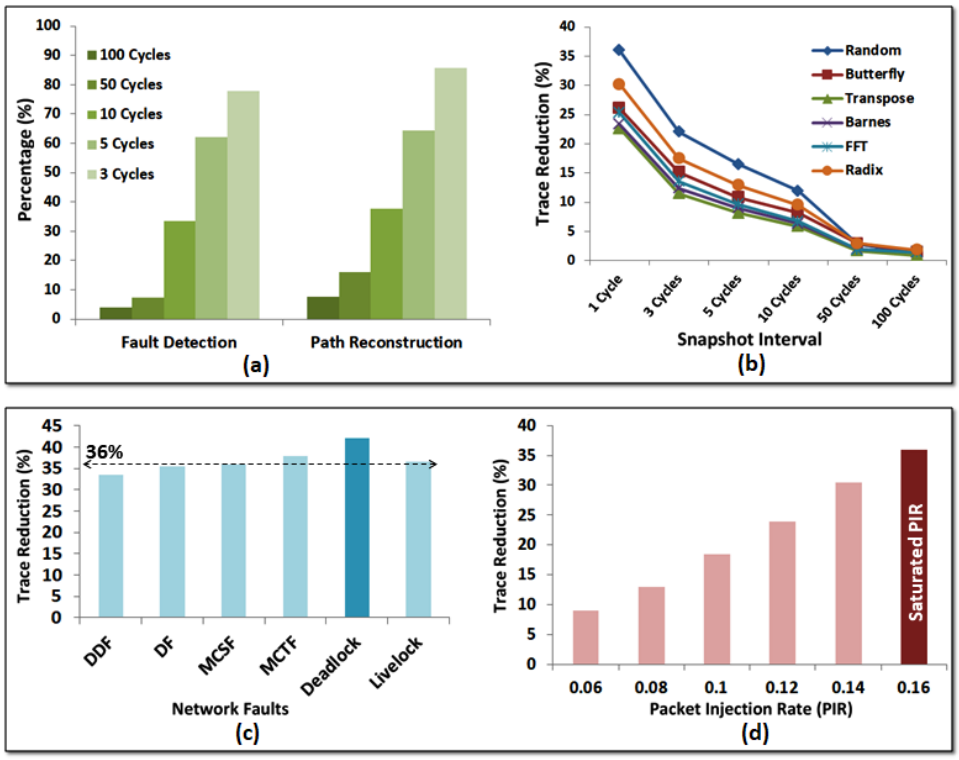


Figure 3.17: (a) Fault detection, and path reconstruction for different snapshot intervals, (b) Trace reduction of RTE scheme for different workloads, (c) Trace reduction for different network faults, (d) Trace reduction with increasing PIR value.

with DF, MCSF, and livelock. DDF shows a small decrease in trace reduction, which is due to 10% packet loss from the network. MCTF shows a small improvement in trace reduction. This is because multiple instances of the packet are using the same path till the destination, which makes the path congested and generates more redundant traces for the packet. The maximum benefit is observed in case of deadlock, as this permanent fault continues to generate redundant traces for the faulty packet.

With the rising PIR value, packet density on the NoC increases, and the network becomes congested. Thereby, packets spend more time in the intermediate router nodes while waiting for the network resources to become free. Thus, more redundant traces are generated at high PIR value. For such a scenario, the RTE scheme is more useful, as can be observed from Figure 3.17 (d). The results are shown for snapshots captured per cycle for *Random* traffic with different PIR values on a 8x8 mesh network.

3.4.7 Overhead and Scalability

In this section, we discuss the overhead associated with WiND and evaluate its scalability. The area and power consumed by the additional debug components introduced to the baseline NoC are considered as the

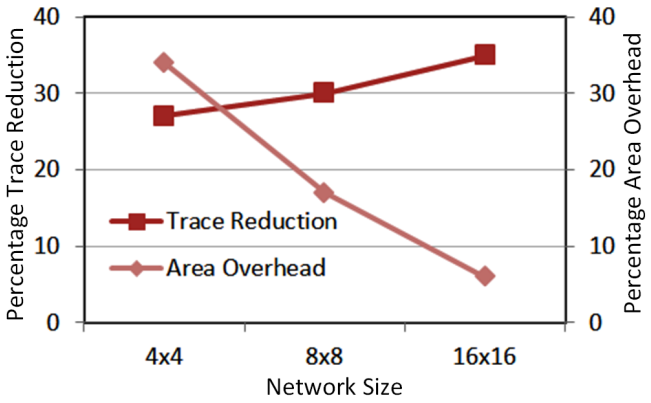


Figure 3.18: Scalability analysis of WiND platform.

hardware overheads of the proposed framework. The area of the transceiver circuit per WI designed in 65-nm CMOS process is 0.3 mm^2, and it consumes 36.7 mW of power [21]. However, the number of WI is much less in comparison to the whole system size, resulting in a smaller WI area overhead. Moreover, the power management scheme power gates all the inactive WIs leading to negligible power overhead introduced by the wireless infrastructure. Our framework introduces additional circuits viz. input demultiplexers, output multiplexers, HB, and RTE module to the router architecture. The size of the HB and RTE module depends on the maximum number of packets that can be accommodated by the router node at a particular time. For overhead calculation, we consider a 5 I/O port router with 2 VCs per port that can accommodate maximum of 10 packets (1 packet per VC). The circuit components are designed using 65 nm technology in the Synopsys design compiler. Synthesis results show that router additional circuits occupy 0.026 mm^2 of area and consume 0.8 mW of power, those are much less compared to the baseline router values. The operational power consumption of a 8x8 system during the debug phase is obtained for WiND platform. For a debug simulation period of 100,000 cycles and 32K injected *Uniform Random* packets, the system consumes 1.08 W of operational power.

As the system size grows rapidly, the proposed debug framework must be scalable to larger NoC. With increased network size, the cluster size would increase and a single WI would be responsible for transferring traces from more number of nodes. Otherwise, more WIs can also be augmented, keeping the cluster size small. Additional WIs would introduce area overhead to WiND, however the relative overhead would decrease as the number of WIs does not increase linearly with the system size. Moreover, the implemented power management scheme along with the single-channel wireless communication would keep the associated power consumption unchanged. With more number of WIs, the average hop-count reduces considerably. Hence, for a larger system, WiND would show a significant reduction in trace buffer requirement compared to the wired counterpart. Figure 3.18 presents percentage area overhead of WiND w.r.t. the whole system and its percentage trace size reduction w.r.t. wired counterpart for 3 different system sizes such as 4x4 with 2 WIs, 8x8 with 4

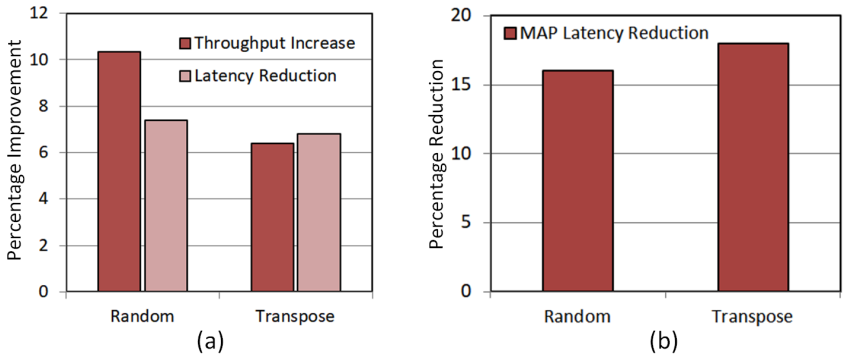


Figure 3.19: In-field system performance improvement by reusing WiND.

WIs, and 16x16 with 6 WIs. The trend shows smaller area overhead and larger trace benefits for increasing system size. Thus, WiND can be considered as a scalable debug framework for future NoCs.

3.5 Reusability of WiND Debug Structure

The majority of debug hardware become unused after the post-silicon debug phase, and are considered vestigial once the chip is sent for mass production. To compensate for the hardware overhead of the debug structure, different researchers have proposed reuse of debug components for architectural purposes or vice versa [66, 67]. To emphasize the usability of our proposal, we have discussed a few potential reuse opportunities for the proposed debug framework.

3.5.1 WIs for System Performance Improvement

In our proposal, WIs are augmented on top of the baseline NoC to provide high-speed trace transfer and trace reduction. WIs transfer the long-range test data during the debug mode to reduce the trace amount. The same WIs can be reused for normal payload transfer during in-field operation. This would significantly reduce the network average hop-count, and thereby provide better network throughput along with reduced average packet latency. Figure 3.19 (a) shows the supporting results for *Random* and *Transpose* traffic on a 8x8 network with 4 WIs, simulated for the saturated PIR values. Reuse of WIs for normal payload transfer would create a modified network similar to Wireless NoC (WNoC). However, the traces of wireless packets collected during PTC phase (discussed in Section 3.3.1) enables WiND to debug the modified network as well.

3.5.2 For Faster Off-chip Memory Access

The off-chip memory access latency problem is a growing concern for multi-core systems[27], which results in large main memory access time and thereby degrades system performance. This problem can be divided into two parts, such as (i) on-chip travel time of the packets that are meant for off-chip memory access (MAP), and (ii) access latency of the off-chip memory. Reducing the MAPs' on-chip travel time would lower the total main memory access time. Our proposed debug hardware can prioritize such MAPs over the normal packets, while accessing the shared resources of NoC. MAPs can access the TCB within the routers on its path for faster movement. This will considerably decrease the on-chip traversal time of MAPs till the memory controller, as shown in Figure 3.19 (b).

3.5.3 For Network Performance Monitoring

WiND structure can also be used as a network performance monitor. Buffer waiting time of each packet stored in the router VCs can be monitored using the WiND snapshot capture components. Such information helps in predicting network congestion status and can provide useful inputs to on-chip dynamic management controllers.

3.6 Conclusion

In this chapter, we propose WiND, a novel post-silicon debug framework for NoC using wireless infrastructure. This method utilizes the augmented WIs for efficient trace reduction, and trace data transfer. The scheme is shown to be more useful for detecting network short-lived faults by providing the flexibility of frequent trace collection without any additional overhead. Experiments exhibit that WiND debug framework shows an overall improvement of 15% to 26% on fault detection and 27% to 34% on path reconstruction in case of different short-lived faults. It is observed to have high bug detection reliability even in case of short-lived faults. The platform is highly efficient in-terms of both on-chip and off-chip trace transfer speed. The WiND framework is shown as a scalable solution and can be reused for multiple architectural purposes. This chapter also discusses RTE, another novel NoC post silicon debug framework that can reduce around 36% trace compared to baseline wired debug framework. Both WiND and RTE considerably improve the efficiency of NoC debug framework.

Chapter 4

Reuse of Debug Structures in NoC Based Systems

The Chapter 3 discusses on efficient NoC debug infrastructure and indicates a few debug hardware reuse possibilities. After the system validation and mass production, most of the DFD hardware remain vestigial on the design. Reusing the idle debug infrastructure for architectural purpose during in-field operation can significantly improve the system performance. This motivates the objective of Chapter 4. This chapter discusses ReDeSIGN, a framework to reuse the DFD infrastructure during the in-field operation for performance enhancement of the NoC-based MPSoCs. Major contributions of this work include reuse of (i) trace buffer as extended VC for network throughput improvement, (ii) trace prioritization hardware for critical data prioritization, and (iii) packet monitor module for packet starvation control. VCs are the most power consuming components in an NoC. Therefore, ReDeSIGN is augmented with a dynamic VC power management mechanism (DNoC) to control the power of the increased number of VCs in the system.

4.1 Motivation and Contribution

System internal visibility and DFD overheads are competing in nature. As the size and complexity of NoC-based MPSoCs are increasing, the DFD hardware footprint is also growing (upto 20% or more in silicon real estate [7]) to maximize the system observability during debug. The area overhead introduced by DFD hardware is considered as a major design concern. This is because many of the DFD modules become non-functional once the SoC goes into production [9], only except a few that are reserved for the in-field debug. An illustration is shown in Figure 4.1 for bug capture in system development steps and unused debug structures during in-field

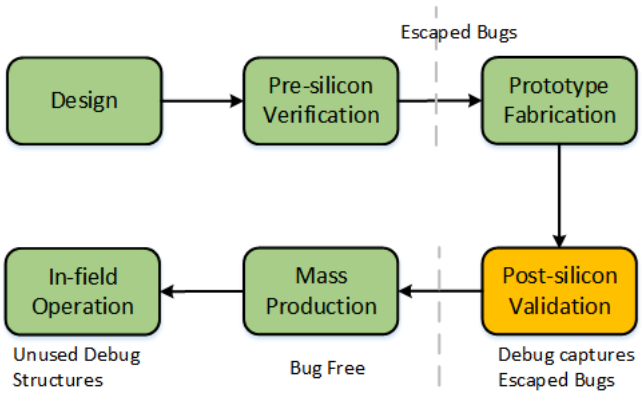


Figure 4.1: System development steps and bug capture.

operation. To address this issue, reuse of architectural component as DFD component or vice versa have been proposed in several research fronts [9, 66, 67, 68, 69, 70]. In this work, we propose ReDeSIGN [71, 72], a complete DFD structure reuse framework to improve the throughput and performance of NoC-based MPSoCs. Firstly, we propose to reutilize the NoC trace buffer, as it largely contributes to the implementation cost of the DFD hardware [10]. The NoC trace buffer in our ReDeSIGN framework is reused as the extended VCs of the network routers. The entire trace buffer is optimally distributed among all the routers to locally store the corresponding packet traces during the debug phase, as shown in Figure 4.2 (a). The trace buffer share at each router is reused as extended VCs of router input channels during in-field operation, as shown in Figure 4.2 (b). Availability of additional VCs effectively reduces the packet drops and deadlock conditions and thereby improves the network throughput. Secondly, ReDeSIGN framework re-purposes the Trace Priority Hardware (TPH) implemented in router arbiter modules for prioritization of critical packets at the time of resource assignment during in-field operation. This provides faster forwarding of critical data on the network, and effectively reduces the application run-time of the system. Originally, the TPH is designed for trace prioritization that provides faster trace transfer during debug phase. And finally, Packet Monitor Unit (PMU) and Trigger Unit (TrU) along with the Timestamp Unit (TU) are re-employed for packet starvation control in the proposed framework. PMU continuously observes the Buffer Waiting Time (BWT) of all the packets present inside the router. As soon as the BWT of any packet crosses the starvation threshold, the TrU assigns a temporary criticality status to the corresponding packet and the network resources are provided to it with priority to avoid starvation. This way, the proposed ReDeSIGN framework enables the re-usability of almost all the DFD structures for the system performance enhancement, and thereby compensates for the area overhead associated with them.

The architectural/functional extensions by the reuse of debug infrastructure are not debugged during the debug phase, as they are newly repurposed during the in-field operation. However, in most of the cases the reuse of debug structure majorly enhances an existing functionality such as extension of VCs in our proposal

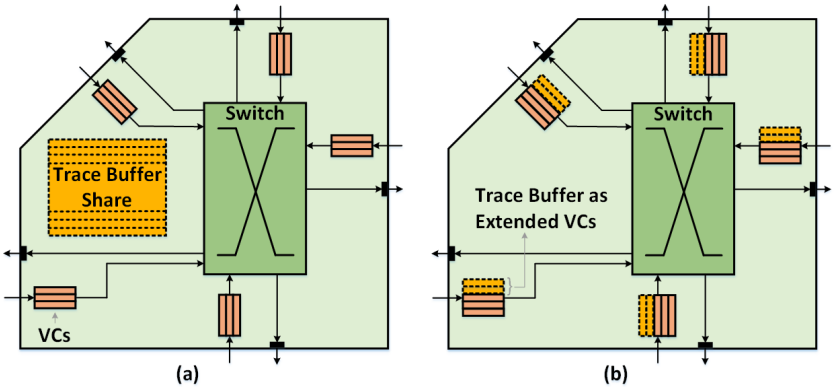


Figure 4.2: (a) Trace buffer share used for storing traces locally during debug, (b) Trace buffer used as extended VCs during the in-field operation.

rather than adding a completely new functionality. The existing functionality is anyway validated during the debug phase and can take care of the system operation in case of the failure of the extended functionality at the baseline system performance level. Moreover, any functional anomaly in these architectural extensions can be captured by the in-field debugger and can also be debugged during the run-time. Therefore, the reuse of the idle debug modules for architectural purposes during in-field operation provide a great opportunity for the compensation of the debug area overhead through the system performance enhancement.

In ReDeSIGN framework, reuse of trace buffer as extended VCs considerably increases the number of VCs at each router port. Among all the NoC components, VCs are the most power hungry modules [22]. Therefore, power management of VCs becomes an important aspect for ReDeSIGN framework. The usage of the routers and resources associated with it are application dependent and for most applications, performance requirements can be met without operating the whole communication infrastructure to its maximum limit. Dynamic reconfigurable system that can switch between both high performance and low power modes will be able to exploit the variable workload conditions provided by different applications. We propose a Dynamic NoC platform (DNoC) [73] that optimizes VC utilization for different applications using a smart router architecture. Power Management Controller (PMC) along with Utilization Computation Unit (UCU) predict and control the number of active VCs to achieve the required performance with low power consumption.

4.2 Background and Related Work

This section is broadly divided into two parts, such as "NoC Background and Related Work" and "Post-silicon Debug Background and Related Work". This thesis has already discussed the architecture and working principles of baseline 2D mesh NoC in Section 2.1. Similarly, it has also presented the background on the generalized

NoC post-silicon debug platform in Section 2.3. The primary objective of having another dedicated section on background of NoC operation and debug infrastructure in this chapter is to discuss the detailed background on the relevant topics with respect to the ReDeSIGN framework.

4.2.1 NoC Background and Related Work

The reuse of debug structures in this chapter are relevant to VCs, critical data priority, and starvation control in the NoC based multi-core systems. Therefore, this section provides the required background on these topics.

4.2.1.1 Virtual Channels in NoC Routers

While traversing from source to destination, packet routing decisions are taken at each intermediate router. Addition of buffers to the routers improves the network flow control, as they decouple the allocation of adjacent channels [28], and reduce the packet drop probability [23]. The network throughput can further be improved by dividing the buffer storage into multiple VCs that can avoid head-of-line blocking and provide a deadlock free message routing on the network [24]. Traditionally, static partitioning of buffer resources is adopted in router implementation [24]. In such cases, all the router nodes are assigned with an equal and fixed numbers of VCs, each with a fixed number of flit counts. Buffer resources consume large power and silicon real-estate [74]. Thus, it is required to restrict the buffer size to a minimum without severely affecting the performance. Therefore, the authors in [75] have proposed to allocate application specific buffer space to NoC routers. However, NoC used in the contemporary MPSoCs targets a wide variety of applications. And an application specific buffer allocation may considerably degrade the system performance when the allocated buffer size is too small or may leave most of the buffer resources unused when the allocated buffer size is too high. In the ReDeSIGN framework, a design can choose to have the number of VCs per router based on the requirement of the least demanding application. The additional VC requirements of the router node for the other high demanding applications can be compensated by the use of the unused trace buffer inside the router. This solves the area overhead issue significantly by re-employing the unused trace buffer as extended VCs of the router nodes during in-field operation.

4.2.1.2 Critical Data on NoC

Traditional NoC treats all the data packets equally on the network. However, there can be different packets that can have a varied degree of impact on the system performance. Certain packets may be performance critical, and

Table 4.1: L2 Cache Miss Rate

SPLASH-2 workloads	**% L2 miss rate**	**Rodinia workloads**	**% L2 miss rate**
barnes	7.34	backprop	28
fft	30	gaussian	20
radix	29.4	lud	18
raytrace	11	bfs	21

can speed up the application execution considerably if prioritized during resource allocation (critical packets). At the same time, other packets can be stalled on the network for a few more cycles without impacting the application run-time (non-critical packets). Authors in [27] dynamically establish the criticality level of each packet on the network based on its slack and provide higher priority to the packet having a lower slack value. The slack value of each packet is determined based on whether it is an L2 cache hit/miss and its distance in terms of hops from its destination. Since L2 cache misses are costly in terms of latency, we consider the associated packets as critical in this particular work. Moreover, from simulation, we observe a reasonable amount of load on the network are related to L2 cache miss, as shown for different workloads in Table 4.1. Therefore, by prioritizing the L2 cache miss packets on the network, the overall performance of the system can be improved. Dedicated priority hardware units are integrated in the router arbiter modules of an NoC to prioritize the critical data during resource allocation [27]. Our work proposes to reuse the existing TPH circuit for the critical data prioritization during in-field operation without embedding any additional hardware for the same purpose. The TPH circuits are implemented within the arbiter modules of the NoC routers and are idle during the in-field operation mode, which creates such reuse opportunity.

4.2.1.3 Starvation Control on NoC

The starvation of non-critical data packets increases with network congestion and stringent prioritization protocols. The starved packets take longer time to reach to their destination. During in-field operation, the order and the relative delay of data packets arrival at the destination nodes have significant impact on the system performance. Thus, with increasing number of starved packets on the NoC, the system performance degrades. There are several works that provide solutions to starvation avoidance on NoC [76, 77]. The proposal in [76] presents a source throttling based congestion control mechanism with application-level awareness that reduces network starvation. Authors in [77] adopt weighted Round Robin (RR) based arbitration during input selection for starvation control. NoC based systems incorporate dedicated starvation control hardware to minimize the amount of starved payload during the in-field operation [76]. This includes a monitor that can observe the BWT of the packets and can detect the starved packets that are waiting for the network resources beyond the star-

vation threshold. It also includes an event generator that creates the requests for the allocation of the network resources to the starved packets. In this regard, the ReDeSIGN framework provides idle DFD structures such as PMU, TU and TrU to be reused to monitor the packets' BWT in the router VCs and to provide a temporary criticality status to starved packets for their forward movement during in-field operation.

4.2.2 Post-silicon Debug Background and Related Work

This section provides a brief discussion on the recent NoC debug platforms. It highlights the existing works on the reuse of architectural and debug components. This section also discusses the comparison of the proposed framework with the state-of-the-art.

There are several research efforts, which demonstrate multi-core debug platform using NoC transaction monitoring [37, 39]. But, a recent work in [9] presented a structured post-silicon debug frameworks for NoC itself. This work utilizes a dedicated portion of local L2 cache as trace buffer, and uses an on-chip debugger for fault analysis. The proposal in [9] tries to enhance the NoC debug by exploiting the architectural components. In this work, we propose to reuse the NoC debug components as architectural modules for system performance enhancements. We adopt the generalized debug platform shown in Figure 2.4. However, we incorporate a few changes to this framework, as demonstrated in Section 4.3.1. First, we replace the centralized trace buffer with smaller size distributed trace buffers for trace storage at each router node. This would enable the framework to store the packet traces locally and would eliminate the cost involved in carrying a trace packet till the centralized trace buffer every time it gets generated. Second, we make use of the existing NoC infrastructure for on-chip trace transfer up to the trace bus, whenever the traces need to be exported to the debug analyzer.

4.2.2.1 Reuse of Architectural and Debug Components

Majority of debug hardware become unused and are treated vestigial once the chip is sent for production. So, many efforts have been made to reuse the architectural components for debug purpose or vice versa to overcome the area overhead introduced by DFD structure. Instead of using a dedicated trace buffer, the work in [67] uses a data cache to store both traces and data during debug. Authors in [68] propose to use L1 and L2 caches to store memory operation activity logs required for memory consistency and coherence validation. A dedicated portion of L2 cache of each node is used as trace buffer for NoC validation in [9]. While [67] uses write-back circuitry to export trace content, [9] utilizes existing wired links of NoC for trace transfer up to the on-chip debug analyzer.

Table 4.2: Summary of comparison with the related works

	[66]	[69]	[70]	ReDeSIGN
Debug hardware reuse objective	Improvement in cache access	Runtime verification	Malware detection	NoC performance improvement
Debug hardware reuse proposal	Trace buffer reused as victim cache	Trace buffer reused for online monitoring of program assertions	Trace buffer reused for real-time malware detection	Trace buffer reused as extended VCs, TPH reused for critical data prioritization, monitor unit reused for starvation control
Reused debug components	Trace buffer	Trace buffer and monitor unit	Trace buffer and the traced debug signals	Trace buffer, priority hardware, monitor unit, timestamp unit, trigger unit
Proposal Achievement	System speed up by 8.3%	Performance overhead reduction of runtime verification by 32%	Malware detection success rate more than 94%	Increase in network throughput by 11.46%, decrease in critical data latency by 34.93%, and decrease in packet starvation by 19.17%
Implementation overhead	Area overhead of 0.011 mm^2	Not reported	Zero silicon area overhead	Area overhead of 778.2 um^2

There are several efforts, which demonstrate the reuse of DFD hardware for some architectural enhancements. Authors in [69] have re-employed the debug structure for online monitoring during runtime verification. Embedded trace buffers are re-purposed for malware prevention in [70], and reused as victim cache to improve system performance in [66]. In this work, we have also proposed to reuse the debug components for architectural enhancements of NoC-based MPSoCs. A comprehensive summary of comparison of our proposed work with the above related works that reuse debug structures for architectural enhancements is presented in the following sub-section.

4.2.2.2 Comparison with the Related Works

The existing works on reuse of debug structures provide solutions in diverse application areas such as runtime verification, real-time malware detection and cache access improvement. The authors in [69] propose DHOOM, an architectural framework for runtime monitoring of program assertions by reusing the unused DFD hardware modules such as trace buffer and monitor unit. DHOOM considerably minimizes the overall performance over-

Table 4.3: List of Acronyms used for ReDeSIGN Framework

Acronym	Full Form	Acronym	Full Form
BWT	Buffer Waiting Time	DSU	Debug Support Unit
ED	Equal Distribution	FD	Fair Division
GTDT	Global Trace Data Transfer	IPRRA	Improved Parallel Round Robin Arbiter
LTDT	Local Trace Data Transfer	MFDD	Modified Fair Division Distribution
NTP	Network Trace Port	PI	Profile Index
PSB	Packet Status Bit	SU	Snapshot Unit
TBUF Ctrl	Trace Buffer Controller	UCU	Utilization Computation Unit

head of runtime verification compared with baseline software-only monitors. PREEMPT, a malware detection methodology is proposed in [70] that reutilizes the embedded trace buffer and the signals used for post-silicon debug to detect botnets of Gafgyt and Mirai families. PREEMPT claims to have high malware classification accuracy and low malware detection latency. The work in [66] proposes to reuse the trace buffer as the victim cache of a processor to enhance the cache access performance. The trace buffer is reused as a victim cache with set associative structure in a 2-way simultaneous multithreading processor core. In contrast, our ReDeSIGN framework reuses almost all the debug structures for performance improvement of NoC infrastructure. It reuses the trace buffers as extended VCs of NoC routers that can considerably improve the network throughput. In addition, the proposed framework reuses different debug components for critical data prioritization and starvation control for overall improvement in system performance. A summary of comparison with the related works is given in Table 4.2.

This chapter includes several acronyms for ease of writing. Table 4.3 lists all the acronyms used specifically for the ReDeSIGN framework. The common acronyms used in the whole thesis can be referred from the "List of Acronyms" at the beginning of this thesis.

4.3 Proposed ReDeSIGN Framework

This section discusses the details of the reuse methodologies proposed in ReDeSIGN framework. Moreover, this section illustrates the network operation and power management during both debug and in-field execution modes. The following sub-sections highlight the use of the debug modules for different purposes during different operation modes. A *mode* signal coming from Debug Support Unit (DSU) decides the mode of operation of the network. Figure 4.3 (a) shows a 64 node network and (b) illustrates the router architecture of the ReDeSIGN framework.

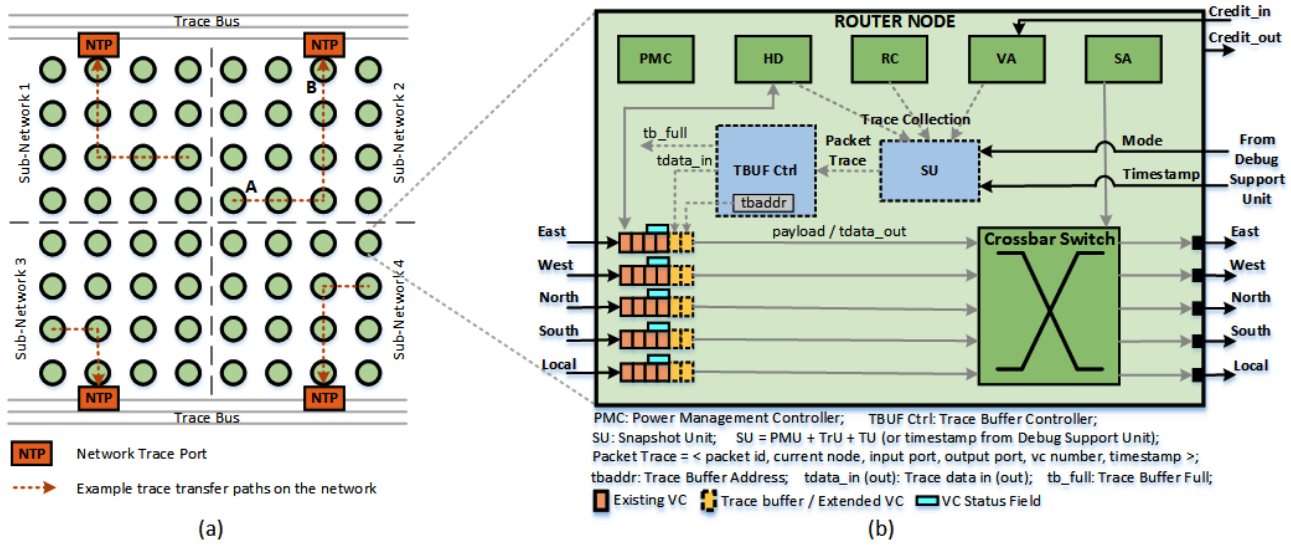


Figure 4.3: (a) A 64 node network, (b) Proposed router node architecture of ReDeSIGN framework.

4.3.1 Network Operation During Debug Mode

During post-silicon debug, the trace buffer is used for packet trace storage. To achieve an efficient debug framework, the total system trace buffer in ReDeSIGN platform is distributed among all the router nodes using a modified Fair Division (FD) algorithm that is discussed in Section 4.4. During debug mode, the Snapshot Unit (SU) collects the packet state information from HD, RC and VA and builds the respective packet trace by adding the corresponding timestamp value as shown in Figure 4.3 (b). Figure 4.4 shows the format of a complete packet trace pattern that is elaborated in Section 4.3.1.1. The SU is composed of the PMU and the TrU. Moreover, there can be a local TU as a part of SU in each router, or the global timestamp value can be referred from the DSU. Upon the generation of a trace, the trace buffer controller (TBUF Ctrl) generates the trace buffer address (*tbaddr*) and forwards the generated packet trace (*tdata_in*) to the corresponding *tbaddr* location. TBUF Ctrl asserts a *tb_full* signal, when it finds no free trace buffer space inside a router. This prompts to export the trace buffer content (*tdata_out*) of the filled router node to the external debug analyzer.

Trace transfer from routers is performed in two different phases similar to WiND framework, namely Local Trace Data Transfer (LTDT), and Global Trace Data Transfer (GTDT). The LTDT phase starts whenever a particular router node runs out of its local trace buffer space, for transferring the traces of the associated router. Whereas, the GTDT phase starts just before the fault analysis in the debug analyzer for collecting the traces of all the network routers. In Figure 4.3 (a), node A is a busy node that experiences large traffic across it, leading to a substantial amount of trace generation. Therefore, its trace buffer fills up quickly, and the LTDT phase starts. During this phase, payload data switching in router A is paused, and packet traces stored in its

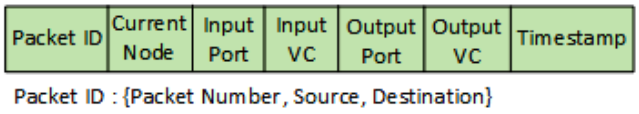


Figure 4.4: Complete packet trace pattern.

trace buffer are forwarded to the downstream router in the direction of Network Trace Port (NTP). The whole network is divided into multiple sub-networks and each of them has a NTP that provides an interface to the trace bus of the debug structure. The switching of trace packets is done in a similar way as the payload packets, buffered through the VCs of the intermediate routers (routers on the path from A to B). These routers give higher priority to the trace packets over the normal packets during resource allocation, and thus provide a faster trace transfer. This is performed using the TPH implemented in the router's arbiter circuit. After the completion of LTDT phase, node A resumes with its normal operation. The GTDT phase starts after a pre-defined large time period. During this phase, normal operation of the whole network is paused, and traces from all router nodes are collected through the NTP ports. This is performed to accumulate complete network traces, so that trace analysis starts in the debug analyzer. The network returns to its normal operation after the GTDT phase.

4.3.1.1 Packet Trace Pattern

All the components of a packet trace is shown in Figure 4.4. *Packet ID* is unique for each packet on the network and is constituted of *Packet Number*, *Source* address and *Destination* address. *Packet Number* is assigned to a packet at the *Source* at the time of generation. *Timestamp* indicates the time when the trace is captured. *Current Node* indicates the router number where the packet is present at the time of trace capture. The *Input Port*, *Output Port*, *Input VC* and *Output VC* specify the path taken by the packet within a router. A packet trace provides the complete information about the packet states at a time instant. In the debug analyzer, the packet path reconstruction is performed. This is done by grouping and arranging all the traces corresponding to a particular packet based on the increasing *Timestamp* values. Fault analysis is performed on the packet trace groups to find out any existing error such as DF, DDF, deadlock etc.

4.3.2 Network Operation during In-Field Execution Mode

The primary objective of this work is to reuse the debug modules as architectural components for the enhancement of network operation during in-field execution mode. As a result, improved system performance is achieved with the existing hardware only. Additionally, our framework guarantees the quality-of-service of the critical workloads while at the same time not overly-constraining the non-critical workloads in case of

Table 4.4: Debug Structures for Architectural Purposes

Debug Component	**Architectural Purpose**
Trace buffer	Extended VCs
TPH	Critical data prioritization
PMU, TrU, and TU	Starvation control

mixed-critical data on the system. The key components of the DFD structure are (i) trace buffer, (ii) PMU, (iii) TrU, (iv) TU, and (v) trace bus as shown in Figure 2.4. Moreover, to our debug platform, we introduce TPH for trace prioritization during debug mode. Our proposed ReDeSIGN framework reuses almost all the above DFD modules for architectural purposes, as listed in Table 4.4.

4.3.2.1 Trace Buffer as Extended VCs

During the in-field execution mode, the trace buffer is no more used for debug purposes. The share of trace buffer allotted to a router node is equally divided among all the input ports, and are re-utilized as extended VCs (Figure 4.3 (b)) in the proposed ReDeSIGN framework. The incoming payload flits now find additional buffer space at each router input port, leading to a reduced chance of packet drop and deadlock conditions. VA module checks the *Credit_in* information from the downstream routers to assign VCs to the header flits. Here, credit refers to the number of available buffer slots of VCs in the downstream router. The extended VC scenario in our method increases the possibility of VC credit having a positive value most of the time, and thus increases the throughput of the network.

4.3.2.2 TPH for Critical Data Prioritization

The proposed ReDeSIGN platform embeds priority logic in both the VA and SA units for trace prioritization during debug phase, known as TPH. The same hardware enables the framework to prioritize the critical data on the network during the in-field operation. The criticality status of a data packet is known from the header flit. A bit in the header flit is reserved as a Packet Status Bit (PSB) that is updated to '1' if the data packet is critical and to '0' if non-critical. Each VC is augmented with an additional one-bit *status field* that is updated according to the PSB, whenever a header flit reserves the VC. The body flits are also prioritized according to the VC *status field*, while competing for the network resources during the SA phase. The corresponding VC *status field* is cleared once the tail flit leaves the VC.

ReDeSIGN framework deploys a priority-based scheduling scheme (Algorithm 2) for assigning the network

Algorithm 2 Scheduling Scheme of ReDeSIGN Framework

```
1:  if there exists priority requests of critical data then
2:      if number of priority request == 1 then
3:          grant the request to the requesting flit;
4:      else
5:          grant the request to the oldest requesting flit;
6:      end if
7:  else
8:      if age of any flit > starvation threshold then
9:          invoke starvation control mechanism;
10:     else
11:         grant requests according to RR scheduling;
12:     end if
13: end if
```

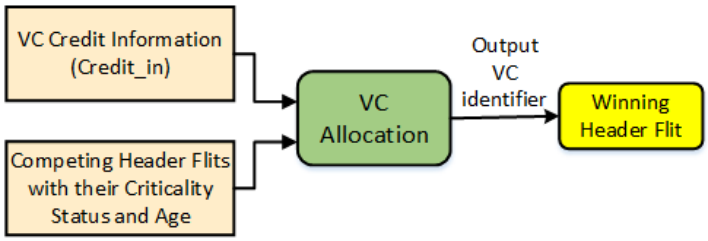


Figure 4.5: VC allocation in ReDeSIGN framework.

resources to the most deserving flit during both VA and SA phases. Whenever an arbitration starts, the priority requests generated by the competing flits are checked. If only one request by a critical flit is found, then it wins the arbitration. If there exist multiple such requests, then the oldest critical flit is assigned with the resources. In case where no critical flit is competing, then it is checked whether the BWT of any candidate is more than the starvation threshold. If yes, then a starvation control mechanism is invoked as discussed in Section 4.3.2.3. Otherwise, the resources are allocated according to the Round Robin (RR) scheduling method.

During the VA phase, the VC arbiter arbitrates among the requesting header flits for allocating a VC in the downstream router. For this, it checks the VC credit information and the criticality status as well as age of the competing flits as shown in Figure 4.5. Based on the rules defined in the scheduling scheme (Algorithm 2), a higher priority header flit wins the race, and the output *VC identifier* is embedded into it. It is to be noted that in the case of *wormwhole* routing, only the header flits compete during the VA phase to acquire a VC. The remaining flits of a packet obtain the same VC.

During the SA phase, each flit of a packet has to compete for getting the access to the switch and the desired output port. The switch arbitration is performed in two stages, as shown in Figure 4.6. In the first stage (VC Selection), a single VC is selected from each input port. In the second stage (Output Port Assignment), arbitration is done among the winning requests of the first stage to decide who wins each of the output ports.

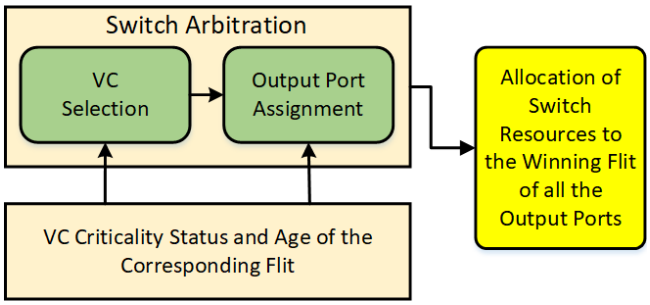


Figure 4.6: Switch arbitration in ReDeSIGN framework.

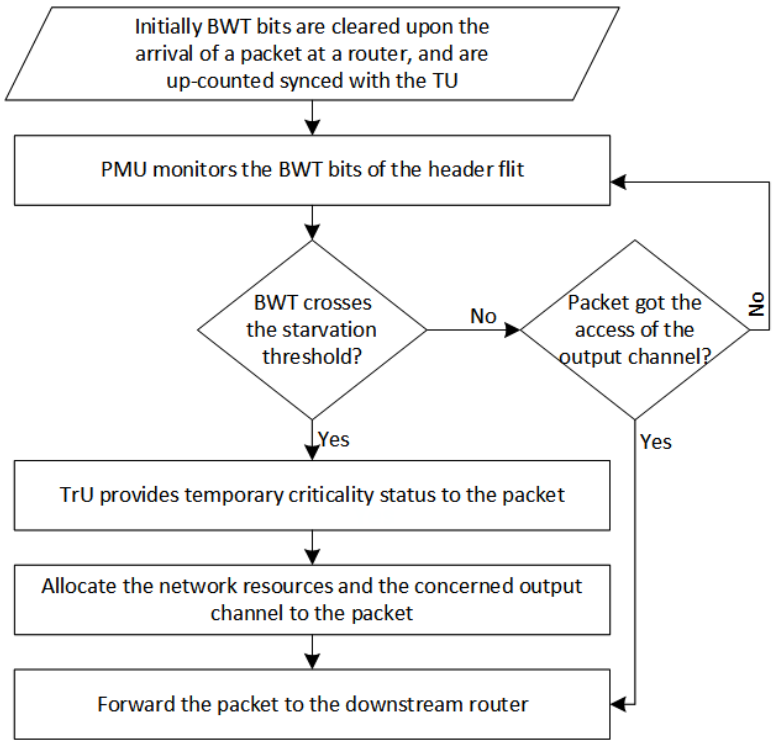


Figure 4.7: Starvation control mechanism of ReDeSIGN framework.

Both the stages perform the arbitration according to the scheduling scheme shown in Algorithm 2, and assigns the resources based on criticality status and the age of the corresponding flit.

4.3.2.3 BWT Monitoring for Starvation Control

Our proposed ReDeSIGN framework re-employs the SU (PMU, TrU, and TU debug units) for starvation control. The adopted starvation control mechanism is illustrated in Figure 4.7. The PMU continuously observes the BWT of all the packets stored in the VCs of the router node. BWT is updated in the header flit of a packet. The size of BWT bits depends upon the starvation threshold that is pre-decided based on the performance margin of non-critical data. Whenever, a header flit reaches a router node, the BWT bits are cleared, and are

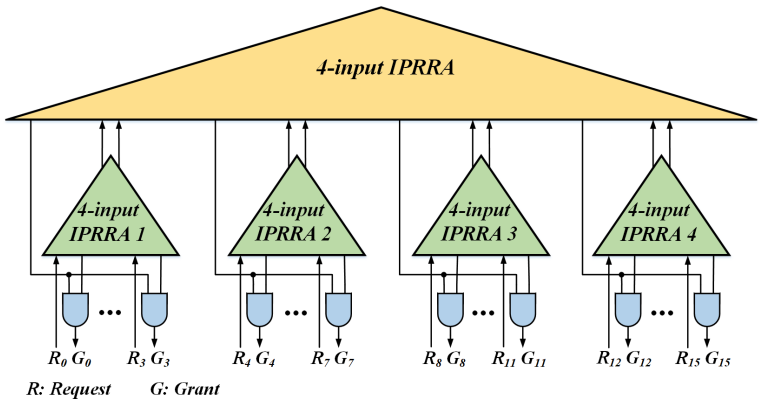


Figure 4.8: 16:1 Grouped IPRRA arbiter using 4:1 IPRRA.

up-counted synchronized with the TU. As soon as the BWT of any packet crosses the starvation threshold, the TrU triggers and assigns a temporary critical status to the packet. This enables the packet to get the access of network resources to move forward to the downstream router, and thereby avoid starvation. It is to be noted that while assigning the temporary criticality, the *status field* associated with the concerned VC is updated to '1', and not the PSB of the header flit. This means that at the downstream router, the packet is again considered as non-critical.

4.3.3 Arbiter Design

Arbiters are required to grant the access to any shared resource, when multiple input requests exist during VA and SA phases. The design methodology of arbiters is important, as their latencies determine the router clock cycle. The latency of VA and SA phases depend on the cardinality of the arbiters, or the worst-case delay of the critical path [74, 78]. The arbiter units in the ReDeSIGN framework are designed according to the total VC (existing + extended), so that it can perform the desired arbitration during in-field execution mode. However, the presence of additional VCs results in a larger arbiter design. Therefore, in our proposed framework, we have designed the arbiters using Improved Parallel Round-Robin Arbiter (IPRRA) architecture [79]. IPRRA establishes a binary search tree structure, where nodes at each level operate parallelly and provide quick arbitration. For large arbiter design, grouped IPRRA is adopted to reduce further delay [79]. A 16:1 arbiter using 4:1 arbiters is shown in Figure 4.8 by applying the grouped IPRRA technique.

4.3.4 Power Management of ReDeSIGN Framework

In ReDeSIGN framework, the Power Management Controller (PMC) deployed for the baseline NoC router is used for the power management of DFD structures, with some additional control, as shown in Figure 4.9.

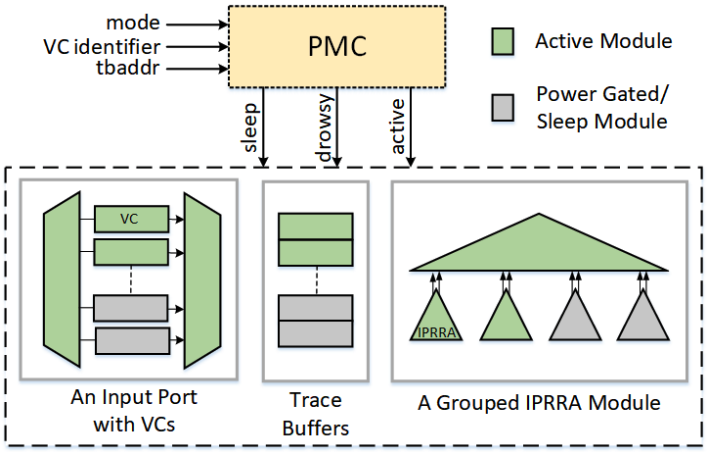


Figure 4.9: Power management of ReDeSIGN framework.

The *mode* signal indicates the PMC that whether the system is operating in the debug mode or in the in-field execution mode. Accordingly, the PMC provides power control signals to different modules within a router during different operating modes. The buffer units and the arbiters are the most power hungry components in a router [22, 80]. Our proposed framework provides a dynamic power management control [81] for both the buffer units (VCs and trace buffers) and the arbiter modules.

During debug mode, the full-fledged arbiters designed for in-field execution mode are not used. So, the unused grouped IPRRA modules are completely power gated by the PMC. Moreover, due to the varying traffic density with respect to time across the NoC routers, even all the baseline VCs are not used most of the time. To exploit this condition, the PMC initially power gates all the VC, trace buffer, and arbiter modules within the router. As soon as a packet arrives at the input port, the *VC identifier* stored in the packet header indicates the desired VC, and the PMC sends the *active* signal (V_{dd} = 1V) to the same VC as shown in Figure 4.9. To store the trace of the arrived packet, the TBUF Ctrl unit computes a trace buffer address indicated by *tbaddr*, and the PMC provides the *active* signal to the corresponding trace buffer location. However, the data stored inside the trace buffer do not switch till the time the trace buffer is not full. Hence, once the trace data is stored, the PMC provides the *drowsy* signal (lowered V_{dd} = 0.63 V) at the next timestamp to the corresponding trace buffer location so that the trace data can be retained within the trace buffer and at the same time the power consumption can be lowered. Furthermore, to provide the arbitration to the stored packets, the required portions of the baseline grouped IPRRA module are activated as indicated in Figure 4.9. Moreover, the TBUF Ctrl and SU modules are power gated during the trace transfer, as no packet traces are captured during these phases.

During the in-field execution mode, the power control scheme is exactly similar to the debug mode only except few changes because of the use of trace buffers as extended VCs. The TBUF Ctrl module is permanently power gated as *tbaddr* generation is no more required. Even though there is no trace generation, the SU is

kept active as the PMU, TrU and TU within it are used for starvation control during this mode. The VCs (existing + extended) and the arbiters (full-fledged) power management is performed dynamically according to the amount of payload data present inside the router. The following sub-section provides a detailed discussion on a proposed framework named DNoC, a dynamic power management scheme for VCs in NoC routers during in-field execution mode. The same mechanism drives the power management of the other components of the router module, both during the debug and the in-field execution modes.

4.3.5 DNoC: A Dynamic VC Power Management Scheme

The number of VCs per physical channel and VCs depth are two important parameters that interplay the buffer utilization, throughput, latency and energy consumption. Since buffer resources come at a premium, efficient management of it is desired. Fixed buffer structures will either be underutilized or will underperform at certain traffic conditions. DNoC architecture is proposed to best serve the varied workload demand. Different applications need different traffic condition and thereby claim different performance efficiency requirement. Routers in DNoC platform are provided with multiple VCs per channel to satisfy the maximum demanding application. But at the other extreme, tasks who claim less traffic are provided with few active VCs out of the multiple available ones. An efficient use of VC resources is done by calculating the runtime traffic demand for varied applications in terms of router utilization. A Utilization Computation Unit (UCU) in each router computes runtime utilization for all neighboring routers that are at one hop distance from it. The UCU observes activity of the router and computes runtime utilization of the neighboring routers based on the traffic condition. This utilization information is provided to the PMC unit of the downstream router. PMC reads necessary information about the status of router utilization and generates different operating voltage levels based on that.

The estimation of router utilization is done periodically on an epoch-to-epoch basis within the UCU. A fixed duration epoch is empirically set at 100 cycles for the set of benchmarks used in our simulation. At the start of an epoch, UCU in each router processes the header flits that are queued in its input VCs through the HD module. Thus, it determines the number of packets that will be routed to each output port of the router and thereby the utilization estimation of the routers connected to the output ports is done. Once UCU completes its operation, it transfers the estimated utilization information to the corresponding neighbor routers through the *Pilot_out* signal, as shown in Figure 4.10.

PMC receives the estimated router utilization information for all of its input channels through the *Pilot_in* signals. Utilization information are provided by the *Pilot_out* signals from the upstream neighboring routers. Initially, PMC provides sleep signals to all the VCs except the first one at each input port and drives them to

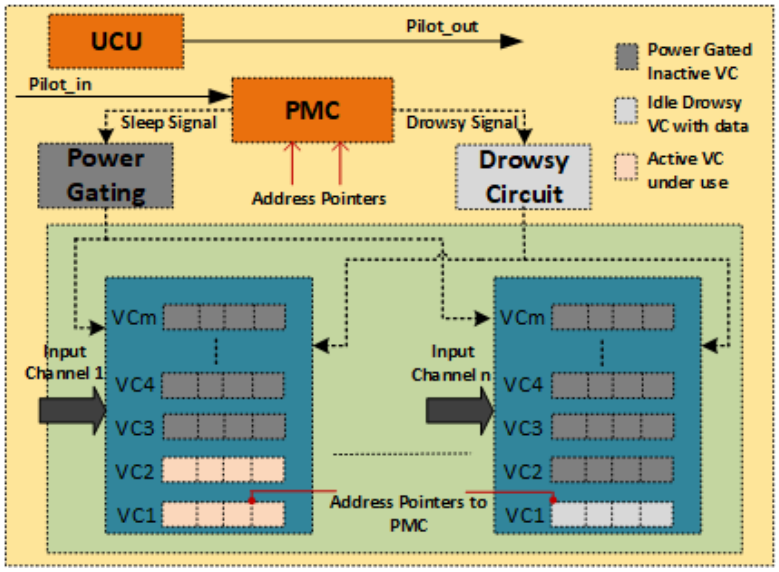


Figure 4.10: DNoC VC power management at router level.

power gating state to save the maximum possible power. At the same time, to reduce the wake-up penalty and avoid any data loss, the controller sends drowsy signal to the first VC of each port. This is shown for the input channel n in Figure 4.10. The utilization information on the *Pilot_in* signal instructs the PMC when to remove or activate the drowsy signal. PMC removes the drowsy signal from the first VC of a particular input port, when the router utilization information indicates that there are packets in the upstream router to be routed to the same input port. Once a particular port of a router starts getting utilized, the address pointers shown in Figure 4.10 indicate the exact location till which the VC is occupied. When the address pointer indicates the first buffer to be half/more filled, PMC removes the sleep signal from the second VC to save the wakeup latency of the subsequent one. This enables the second VC to receive the upcoming flits, as illustrated for the input channel 1 in the Figure 4.10. Work flow of the PMC in the proposed DNoC scheme is presented in Figure 4.11.

Whenever a VC is completely empty, it is unutilized and PMC sends the sleep signal to drive it to power gating state. VCs in DNoC router are connected to sleep transistors which couple them to ground rail. When PMC sends a sleep signal, the corresponding sleep transistor gets into cut-off region and thereby disconnects the associated VC from the ground terminal. This saves a significant amount of leakage power. Depending on channel occupancy, the address pointer may invoke the controller to remove the sleep signal and bring the subsequent VC to active state. When a VC holds message flits, but no active communication is going on because of the congestion in subsequent routers, PMC sends a lower level supply voltage to the VC to reduce the power consumption. The retention voltage that ensures no loss of stored data in VCs is applied, and the mode is called as drowsy mode. In our implementation, D flip-flop based buffer is used to design the VCs. From our experiment, we estimated the data retention voltage for DFF is 0.63V at 32nm technology node.

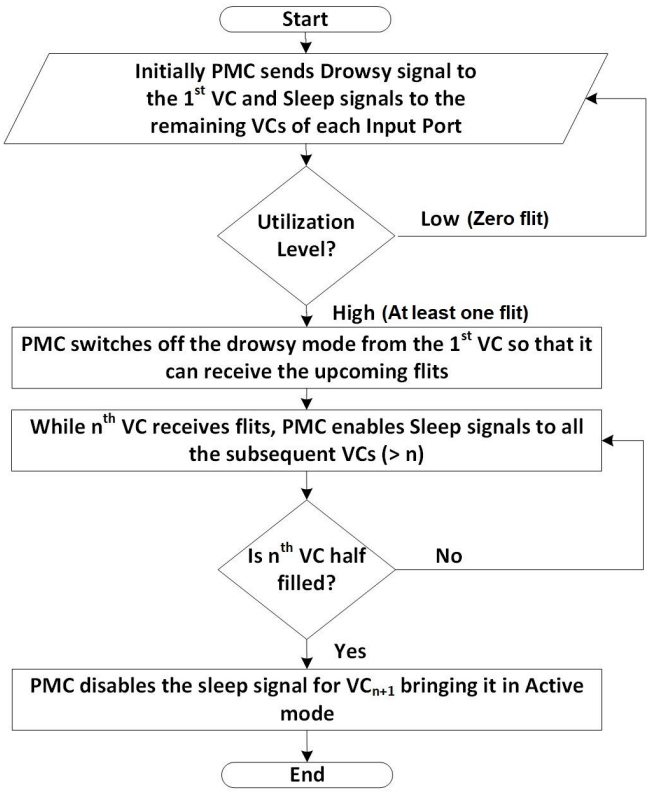


Figure 4.11: Work flow of the PMC unit in a router.

Consequently, the circuit is operated under two level of voltages; 0.63 (drowsy state) and 1V (active state). DNoC's architecture power gates the VCs while unutilized. Under low traffic condition, only a limited number of VCs are activated and utilized. This way, the total energy dissipation in the VCs for the whole network reduces drastically. With the rise in traffic, more VCs are activated dynamically to handle the increased load. This proves DNoC as a real time manager of buffer resources which provides power efficient solution for variable load condition without having any impact on the performance of the system.

4.4 Optimal Trace Buffer Distribution

In ReDeSIGN framework, the distribution of trace buffer among router nodes is performed using a Fair Division (FD) algorithm to achieve optimal results. A proportional FD method says every agent receives at least its due share of a set of resources according to its own value function [82]. In this context, routing nodes are agents, trace buffers are resources and Profile Index (PI) of each node based on traffic condition are considered to be respective value function $f(v)$. The following two sub-sections present the details about the calculation of PI of each node, and accordingly fair distribution of available trace buffer resources among all the nodes.

4.4.1 Profiling the Router Nodes

Each of the network router nodes is profiled based on the amount of traffic passing through it for a given application. Profiling of an individual node is done separately for all the applications of a given MPSoC. Finally, each node is assigned with a PI, which is the arithmetic mean of all its profiling values across the applications. For a network with n nodes $\{N_1, N_2, ... N_n\}$ and m applications $\{A_1, A_2, ... A_m\}$, PI of each node can be realized as below in Equation 4.1.

$$PI_{N_i}|_{i\in\{1\ to\ n\}} = \left\{\sum_{j=1}^{m} p_{i,j}\right\}/m \tag{4.1}$$

Here $p_{i,j}$ represents profiling of ith node for jth application, and PI_{N_i} represents the final profile index value of ith node. Value of $p_{i,j}$ becomes high if large number of packets traverse through the ith node while jth application is executed. PI value indicates an average occupancy level of the respective node by the payload packets. It can be concluded that the trace buffer requirement of each node is proportional to its PI value. So, value function $f(v)$ of each routing node is calculated based upon the corresponding normalized PI as shown in Equation 4.2.

$$f_i(v) = PI_{N_i} / \left\{\sum_{i=1}^{n} PI_{N_i}\right\} \tag{4.2}$$

These value functions are used in FD method for optimal distribution of the available trace buffers. All the applications mapped to the NoC are considered as of equal importance while calculating the $f(v)$ of the routing nodes. This is because the distribution of trace buffer is driven by the amount of traffic, but not by the type of traffic across nodes. Moreover, the traffic density per node is dependent on the routing algorithm implemented on the NoC, which is considered to be deterministic XY routing in our case.

4.4.2 Fair Division of Trace Buffers

The objective of fair division is to divide and allocate resource share to each candidate in such a way that everyone would receive their due share according to their value function. Several FD mechanisms are proposed in literature, from which we have adopted *Dubins-Spanier Moving-Knife Procedure* [83] for the fair distribution of trace buffer. The *Moving-Knife Procedure* is originally proposed to solve the fair division of cake problem among n people, where $n > 2$. A knife is slowly moved across a cake from its extreme left position, and

whenever a person feels that the knife has moved $1/n^{th}$ of the total cake according to his measure, he calls "*cut*". Then he takes that piece of cake and exits. If two persons call at the same time, then the piece is given randomly to any one of them. The process is repeated for $n-1$ participants.

Algorithm 3 Modified Fair Division Distribution Algorithm for Trace Buffer Distribution

assumption: $S_{tb} = p * tb_{N(min)}$;

▷ S_{tb}: trace buffer size; $tb_{N(min)}$: minimum trace buffer share per router; p: a whole number

initial: $m = n; j = 1;$ ▷ total n number of nodes in the network $N = \{N_1, N_2 \ldots N_n\}$, j represents node number

while $m > 0$ **do** ▷ **Trace buffer distribution**

call function ***tb_fair_div***

i = Node number that asked for the trace buffer partition

if $tb_{N_i} \leqslant tb_{N(min)}$ **then**

$tb_{N_i} = tb_{N(min)}$

else if $(tb_{N_i} \,\%\, tb_{N(min)}) \geqslant tb_{N(min)}/2$ **then**

$tb_{N_i} = \lceil tb_{N_i} \rceil$

else

$tb_{N_i} = \lfloor tb_{N_i} \rfloor$

end if ▷ $\lceil tb_{N_i} \rceil$ and $\lfloor tb_{N_i} \rfloor$ returns the next and previous whole number divisible by $tb_{N(min)}$ respectively

$S_{tb} = S_{tb} - tb_{N_i}$

$N = N - N_i$

$m - -$

end while

if $(q = \sum_{j=1}^{n} tb_{N_j} \;/\; tb_{N(min)}) > p$ **then** ▷ **Final trace buffer adjustment**

For top q-p nodes assigned with maximum trace buffer share are reduced by $tb_{N(min)}$ each

else if $q < p$ **then**

For top p-q nodes assigned with maximum trace buffer share are increased by $tb_{N(min)}$ each

end if

function ***tb_fair_div()*** ▷ **Function definition**

Start the S_{tb} division

Wait till any node asks for the trace buffer partition according to its value function f(v)

If more than one node ask for the same partition then assign randomly to any one of them

In the context of trace buffer distribution, a Modified Fair Division Distribution (MFDD) algorithm is illustrated in Algorithm 3. This is shown for an NoC comprised of n nodes with individual node's value function, $f_i(v)$, and total trace buffer size of S_{tb}. The cake problem deals with the division of continuous resources, whereas the trace buffer distribution is a discrete resource division problem. The smallest unit of trace buffer that can be assigned to a particular port of a router node should have the size equal to one VC. In case of a 2D mesh topology (considered for experiments in this work), each router has 5 ports that result into minimum trace buffer share per router node ($tb_{N(min)}$) as 5 times a VC. In a case, where the trace buffer share of a node (according to its $f(v)$) is not equal to a whole number multiple of $tb_{N(min)}$, the final share is rounded up to either the next or the previous whole number multiple of $tb_{N(min)}$. Final adjustment of trace buffer share to meet the size limit of available trace buffer is done on the nodes claiming the largest portions of it, as shown in Algorithm 3.

The principal criterias of fairness in resource allocation are proportionality and envy-freeness [84]. The analysis in [84] shows that the *Dubins-Spanier Moving Knife procedure* [83] guarantees a proportional allocation of the cake in a cake-cutting problem. It also maps every envy-free allocation of the cake to a pure Nash equilibrium for a positive value density function. The presence of Nash equilibrium state indicates the optimality of the solution where all the competitors are fulfilled with their desired proportion of resource (i.e., cake in the cake-cutting problem). As described in [84], our MFDD algorithm also follows the restrictions on the participants to use threshold strategy during the moving knife procedure. Each network node participating in the procedure has a fixed $f(v)$ satisfying the threshold strategy, while competing for a portion of the total trace buffer. Every node gets its desired trace buffer share, fulfilling the envy-free and proportionality criteria of the cake-cutting procedure. Hence, our MFDD algorithm follows the threshold-based *Dubins-Spanier Moving Knife* procedure and attains a Nash equilibrium for the trace-buffer allocation problem, thus meeting an optimal condition. As the MFDD method provides an optimal distribution of trace buffers, the busy nodes get a larger portion of it to store more traces during debug mode and to store more payload during in-field execution mode.

4.4.3 Walkthrough Example on Trace Buffer Distribution

An MFDD-based trace buffer distribution example for a 2x2 2D mesh network is demonstrated in Figure 4.12. The figure shows the corresponding value functions of each of the network node that are calculated using Equation 4.2 based on the traffic condition on the NoC. The total size of available trace buffer is S_{tb} = 85 units. One unit of trace buffer is 1 VC size. According to the $f(v)$ values, the trace buffer share ($tb''_N = S_{tb} * f(v)$) of each node are calculated, leading to an optimal trace buffer share estimation. These values are then rounded up (tb'_N) to the nearest value divisible by ($tb_{N(min)}$). For a 2D mesh network, $tb_{N(min)}$ is 5 units, as mentioned

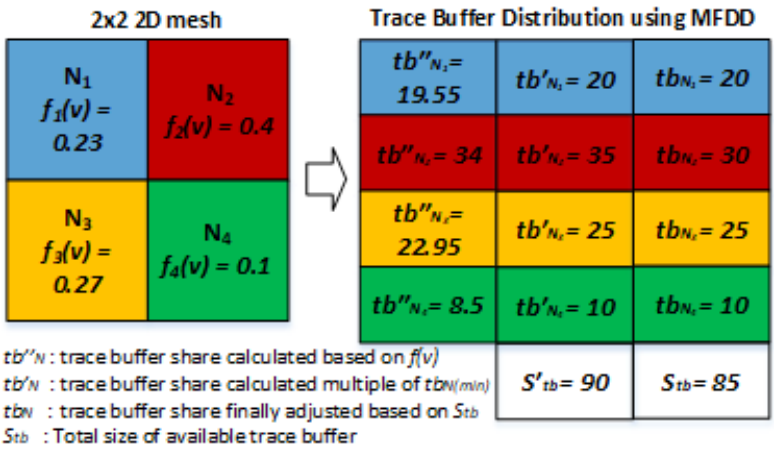


Figure 4.12: Example of MFDD-based trace buffer distribution.

Table 4.5: Network Topology and Simulation Setup

Component	**Configuration**
Topology	8x8 baseline 2D mesh NoC, XY routing, Wormhole flow control
Router	5 I/O ports, 4 existing VCs per port, extended VCs based upon value function, 2 flit VCs, 8 flit packets, 32 bit flits, 1 GHz operating clock
Debug Setup	8KB trace buffer size, single cycle snapshot, 32 bits packet trace size
Workload*	Synthetic - *Random*, *Transpose*, *Butterfly*; SPLASH-2 - *Barnes*, *FFT*, *Radix*; Rodinia - *Backprop*, *Gaussian*, *Lud*
* Synthetic and SPLASH-2 applications are executed on homogeneous system environment and Rodinia applications are executed on heterogeneous system environment.	

earlier. For this example, the summation of the rounded up trace buffer share (S'_{tb}) becomes 90 units, whereas the value of S_{tb} is 85 units. This requires the trace buffer adjustment with the step size of $tb_{N(min)}$ is to be done on the nodes having the highest allocations. In this example, only one node (N_2) that is having the highest trace buffer share (35 units) needs to be adjusted to 30 units to satisfy the S_{tb} value. The adjustment at the highest allocated nodes ensures the minimal deviation from the optimal condition.

4.5 Experimental Results

This section presents the simulation setup as shown in Table 4.5 and discusses experimental results for evaluating the ReDeSIGN framework. An 8x8 2D mesh NoC is modeled on the cycle-accurate Noxim simulator [50] for the generation of results. For scalability analysis, a 4x4 and a 16x16 NoCs are also modeled on the same simulator. To assess the efficacy of the ReDeSIGN framework, we have incorporated and evaluated it separately for both homogeneous and heterogeneous systems. Both system configurations are shown in Table 4.6

Table 4.6: Configuration of System Simulator

Component		Configuration
Core	Homogeneous system	48 x86 Out-of-Order cores
	Heterogeneous system	8 x86 Out-of-Order cores, 40 GPU cores
L1 Cache	Size	64 KB
	Line size	64 B
	Associativity	4
L2 Cache	Size	256 KB
	Line size	64 B
	Associativity	8
L1/L2 Cache replacement policy		LRU

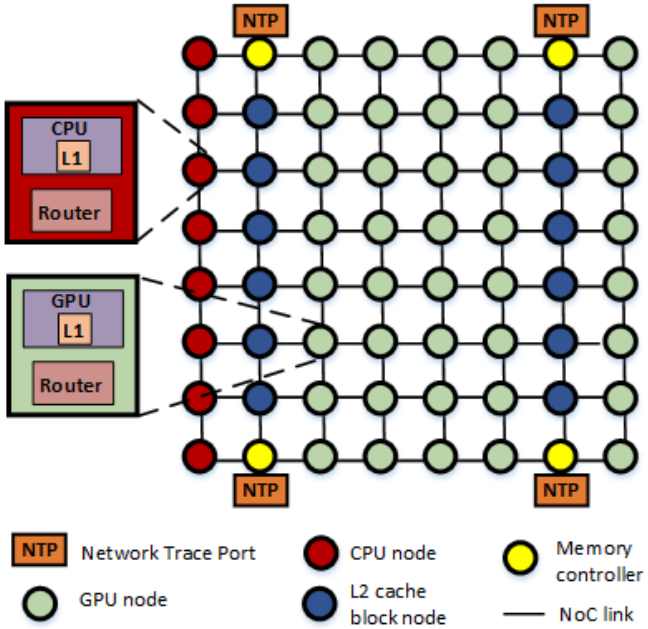


Figure 4.13: Considered system configuration and core placement.

and are modeled on gem5-gpu simulator [85] for the generation of network traces. Simulations are performed for Synthetic and SPLASH-2 [63] workloads on the homogeneous system environment and for Rodinia [86] workloads on the heterogeneous system environment. We have considered 8KB of total trace buffer size for our experiments. To demonstrate the benefit of the proposed MFDD algorithm, we have compared it with the Equal Distribution (ED) of trace buffers in the result.

4.5.1 Configurations of the Evaluated Systems

We have considered 8x8 2D mesh NoC for both homogeneous and heterogeneous systems for our experiments. Figure 4.13 shows the configuration and core placement of the evaluated heterogeneous system, whose details

are listed in Table 4.6. Our proposed framework can be used for any composition of CPUs/GPUs/Last Level Caches (LLCs)/Memory Controllers (MCs) and system size. It only uses knowledge of the CPU and GPU traffic characteristics and traffic density on the network for the trace buffer distribution. Hence, without loss of generality, in this work, we consider a heterogeneous architecture with 40 GPUs, 8 CPUs, 12 L2 cache blocks, and 4 MCs, which are connected through the 64 node NoC. All the cores have their private L1 cache, each of 64KB. The system has 12 shared L2 cache blocks, each of 256KB, which are used as the LLCs in the considered system. The placement of the cores is performed in such a way that the paths to memory routers (LLCs and MCs) from CPU and GPU cores would have reduced overlapping [87, 88]. This ensures that maximum requests are serviced from the memory-intensive GPU cores, while also servicing the latency critical requests from the CPU cores. The LLC blocks and the MCs are placed close to each other. The MC nodes are expected to be highly busy nodes that generate a lot of packet traces during debug. Therefore, MCs are placed at the router nodes that are connected to the NTP ports. This helps in faster trace transfer from the busy MC nodes during the debug phase. In case of a homogeneous system, the GPU cores are replaced by the CPU cores without changing the configuration of the rest of the system.

4.5.2 $f(v)$ Calculation and Trace Buffer Distribution

For the calculation of value function $f(v)$, profiling of each NoC node is performed based on the traffic conditions. All the workload patterns are executed on the 64 node system, and the profile index of each NoC node for individual workload, $p_{i,j}$ is calculated. Figure 4.14 (a) and (c) show the graphs of normalized PI of all the workload patterns for homogeneous and heterogeneous systems, respectively. The $f(v)$ of each node is calculated from Equation 4.2 mentioned in Section 4.4 and is represented by the red line in the figure. Based on these $f(v)$ values, each node gets its due share of trace buffer. Figure 4.14 (b) and (d) show the distribution of trace buffer as a measure of extended VCs at each node of the NoC for both ED and proposed MFDD. Figure 4.14 (b) shows that for an 8KB of total trace buffer size, an ED provides 15 additional VCs per router node, while MFDD allocates as high as 30 additional VCs and as low as 10 additional VCs to nodes of a homogeneous system. Similarly, Figure 4.14 (d) demonstrates that the additional VC distribution in a heterogeneous system varies from 5 to 60 across all the network nodes. This is because in a CPU-GPU heterogeneous system, an NoC mostly experiences *many-to-few* communication patterns, primarily between the GPUs and the LLCs [88]. Thus, a few nodes in such cases encounter the major percentage of the total network traffic, as can be observed from Figure 4.14 (c). Accordingly, few of the nodes get a large trace buffer share, whereas others get a smaller portion of it, as shown in Figure 4.14 (d). Furthermore, it can also be observed that even the nodes that experience very low traffic get 5 additional VCs (that is the value of $tb_{N(min)}$). This is adjusted from the

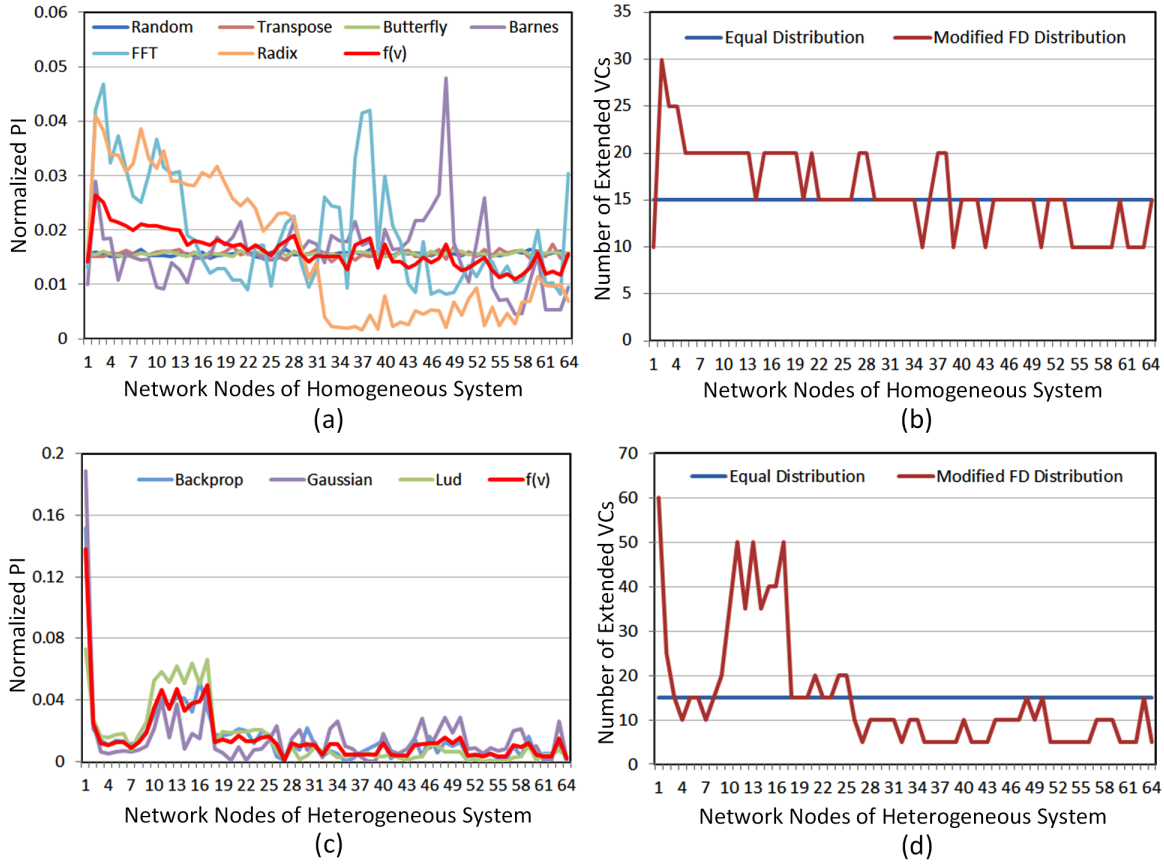


Figure 4.14: (a) Normalized PIs and value functions of each node in homogeneous system, (b) Trace buffer distribution in terms of extended VCs in homogeneous system, (c) Normalized PIs and value functions of each node in heterogeneous system, (d) Trace buffer distribution in terms of extended VCs in heterogeneous system.

ones having the largest share of VCs according to the MFDD algorithm. This results in an optimal trace buffer distribution for both debug and in-field execution modes.

4.5.3 Trace Buffer Overflow

Whenever the trace buffer of a router node fills up, the LTDT phase starts, as discussed in Section 4.3.1. This pauses the corresponding router operation and degrades the intermediate routers' operation while transferring the traces to the NTP. Therefore, it can be concluded that the more the number of trace buffer overflow, the longer the time period of debug process. Figure 4.15 shows an average of 7% and 5.6% reduction in trace buffer overflow in case of MFDD over ED for homogeneous and heterogeneous systems, respectively. This is because, the MFDD of trace buffer encourages parallel filling of buffer space in all the routers. While in case of ED, trace buffer in few of the routers fill up quickly, and in few others, they are mostly unused. Less number of trace buffer overflow followed by faster trace transfer (presented in Section 4.5.4) results in faster debug in case of ReDeSIGN framework.

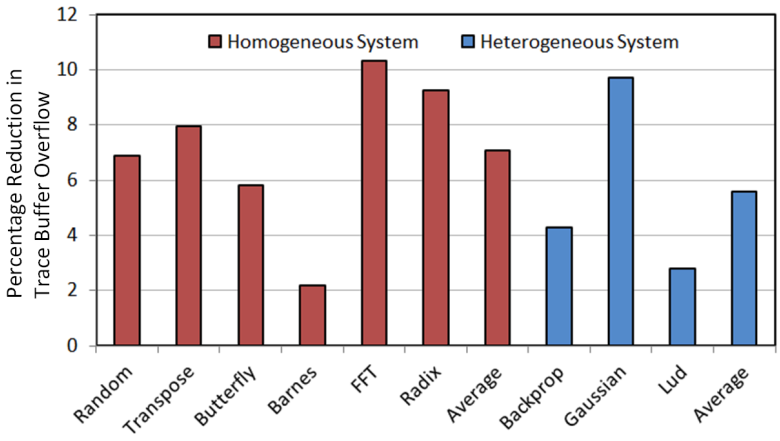


Figure 4.15: Percentage reduction in local trace buffer overflow in case of MFDD over ED.

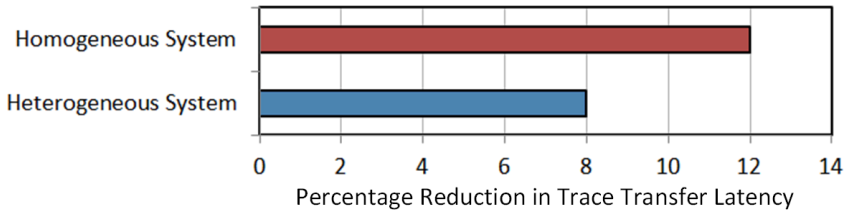


Figure 4.16: Trace transfer efficiency of ReDeSIGN framework.

4.5.4 Trace Data Transfer

In ReDeSIGN framework, trace packets take the NoC path to reach the NTP port. Trace data is prioritized during NoC resource allocations (as discussed in Section 4.3.1) for quick trace transfer. Experimental results show that in case of the proposed framework, there are 12% and 8% reduction in trace transfer latency for homogeneous and heterogeneous systems respectively in comparison to baseline 8x8 systems (Figure 4.16). The baseline systems' configuration do not include the TPH in the arbiter circuits. The simulations are performed for 100,000 cycles for all the applications considered in this work, and the average results are computed separately for homogeneous and heterogeneous systems. Slightly less benefit in case of heterogeneous system is observed because of its *many-to-few* communication patterns. In this scenario, there exist local congestion and trace buffers associated with the busy nodes overflow frequently. Due to the congestion, there is a higher number of competing flits requesting the network resources at particular router nodes during arbitration, leading to a comparatively longer trace transfer phase.

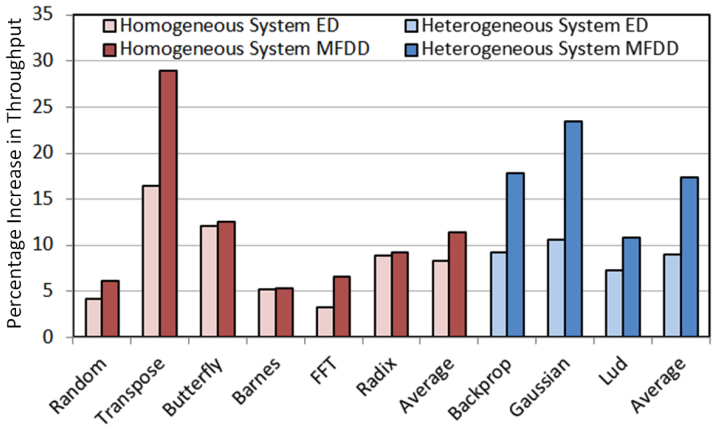


Figure 4.17: Percentage increase in network throughput.

4.5.5 Network Performance with VC Extension

The network performance of ReDeSIGN framework during in-field execution mode is evaluated in terms of throughput. The proposed scheme provides additional VCs to each port of every router. As a result, packet drop as well as network congestion reduces and thereby network throughput increases considerably. Results shown in Figure 4.17 demonstrate the percentage of throughput improvement in case of MFDD and ED of trace buffer. It can be observed that ED provides an average of 8.36% and 9.03%, whereas MFDD provides an average of 11.46% and 17.36% in throughput improvement over baseline architecture for homogeneous and heterogeneous systems respectively. The performance enhancement in case of MFDD over ED is achieved as the router nodes are allocated with additional VCs according to their requirements. Since, there is more disparity in traffic nature and therefore VC requirements in case of heterogeneous system than homogeneous system; the MFDD provides greater benefit to heterogeneous system in terms of network throughput.

4.5.6 Network Performance with Critical Load Priority

In ReDeSIGN framework, TPH is reused to prioritize the critical data (L2 cache miss packets) during network resource allocation of in-field execution mode for system performance improvement (as discussed in Section 4.3.2.2). This is achieved by reducing the critical packet latency on the network. Since, the proposed framework minimizes the waiting time of critical packets in the router nodes, it is expected that the total latency associated with critical network packets would reduce as shown in Figure 4.18. For simulation, the percentage of critical load for SPLASH-2 as well as Rodinia workloads are referred from Table 4.1, and is assumed to be 30% for Synthetic applications. We have generated results for critical load latency improvement in case of proposed MFDD and MFDD with data prioritization hardware (MFDD_TPH) over the baseline architecture for both

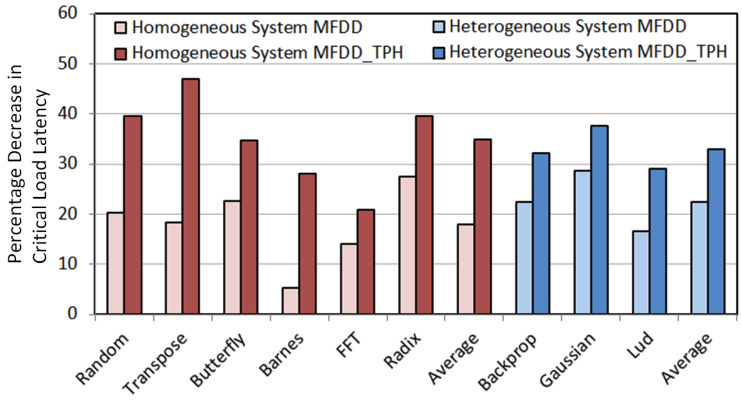


Figure 4.18: Percentage decrease in critical load latency.

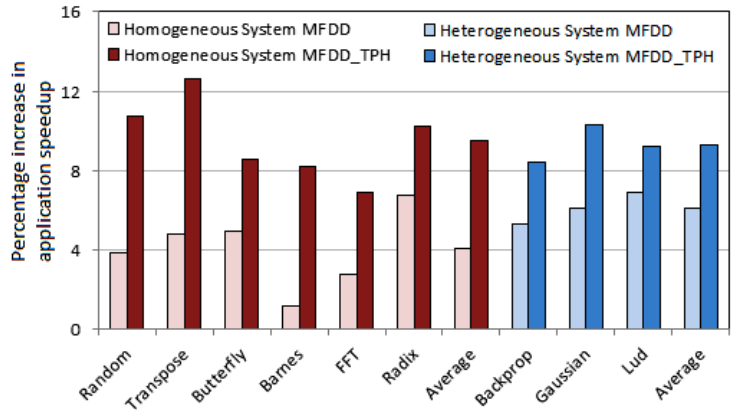


Figure 4.19: Percentage increase in application speedup.

homogeneous and heterogeneous systems. Due to the availability of additional VCs in case of MFDD, a flit is more likely to get the requested VC in its first attempt. This results in a reduction of average network latency, including critical load latency. From our experiments, the decrease in critical load latency is observed to be 18.02% and 22.49% in homogeneous and heterogeneous systems respectively. In case of MFDD_TPH, critical load is prioritized while traversing through the NoC that results in further reduction in critical load latency, that is observed to be 34.93% and 32.91% for homogeneous and heterogeneous system respectively.

Due to significant reduction in the critical load latency, the ReDeSIGN framework shows appreciable speedup in application runtime at the system level. From the experiments, it is observed that our proposed framework achieves 9.54% and 9.31% application speedup improvement in homogeneous and heterogeneous system respectively, as shown in Figure 4.19. We have considered only L2 cache miss packets as critical in this work to validate our framework. However, other important packets such as protocol requests, acknowledgment packets etc. [89] can also be considered as critical. Our ReDeSIGN framework is a generalized one. Depending upon the application requirements, one can define different packets as critical, and our framework would accordingly prioritize the critical packets during the data transfer on an NoC.

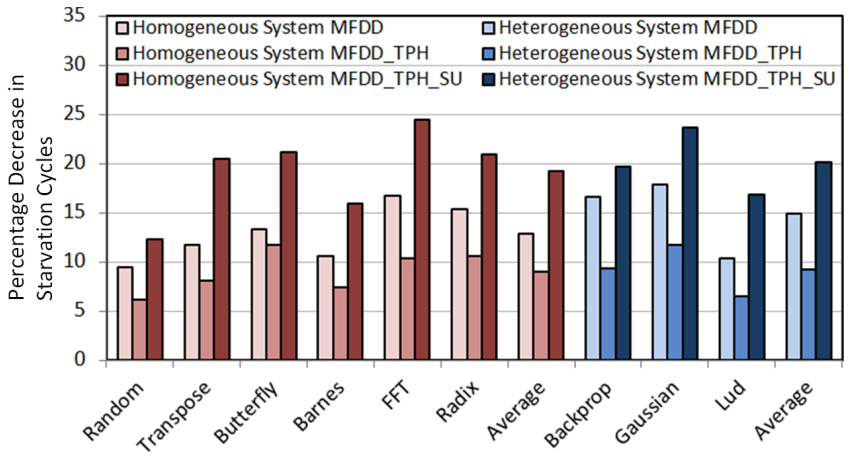


Figure 4.20: Percentage decrease in starvation cycles.

4.5.7 Starvation Control

In ReDeSIGN framework, the SU (PMU, TrU, and TU debug units) is reused to provide an efficient starvation control (as discussed in Section 4.3.2.3). Figure 4.20 shows the percentage reduction in starvation cycles for MFDD, MFDD_TPH, and MFDD_TPH_SU architectures compared to the baseline architecture for both homogeneous and heterogeneous systems. We have measured the starvation control efficiency of the above-mentioned three different architectures in terms of starvation cycles. In this work, the number of starvation cycles of an architecture is defined as the total BWT of all the packets in cycles beyond the starvation threshold. For our evaluation, we have considered the starvation threshold value as 8 cycles that is found from the empirical observations. For data forwarding, the baseline as well as the MFDD architecture follows the RR scheduling, MFDD_TPH uses prioritization for critical data and RR scheduling for non-critical data, and finally MFDD_TPH_SU adopts the scheduling scheme proposed in Algorithm 2. In case of MFDD, more packets find free VC slots in the downstream router, leading to an increased rate of packet forwarding. This decreases the starvation cycles compared to the baseline architecture that is observed to be 12.83% and 14.94% in case of homogeneous and heterogeneous systems, respectively. However, MFDD_TPH architecture increases the probability of resource starvation for non-critical data by prioritizing the critical ones. Thus, in many instances, the non-critical packets keep on waiting in the router buffers, leading to an increased value of starvation cycles. In contrast, MFDD_TPH_SU architecture provides the starvation control mechanism by continuously observing the packet BWT and prioritizing a packet for forwarding, whenever the starvation threshold is crossed. This considerably reduces the value of starvation cycles compared to the baseline architecture, which are found to be 19.17% and 20.07% for homogeneous and heterogeneous systems, respectively.

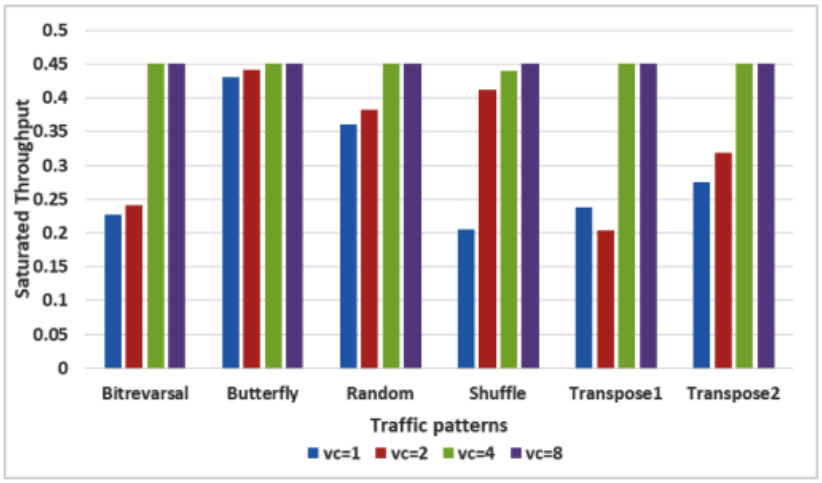


Figure 4.21: Throughput vs traffic patterns for different number of VCs.

Table 4.7: Power comparison for different number of used VCs

	Power consumption of active router port VCs (mW)			
Router Architecture	***1 VC***	***2 VCs***	***4 VCs***	***8 VCs***
Baseline Mesh Router	24.6	24.6	24.6	24.6
Proposed Router	4.10	7.18	13.3	25.7

4.5.8 VC Power Benefit using DNoC Scheme

We evaluated the DNoC framework on a 16x16 2D mesh NoC for 1, 2, 4 and 8 VCs at each port. We have used only Synthetic traffic patterns for DNoC evaluation. Figure 4.21 presents the throughput results with respect to different traffic patterns and for different number of VCs. Simulation result shows that for all the considered traffic patterns, 4 VCs provide equivalent throughput in comparison to 8 VCs. So, here in most of the cases only 4 VCs per router port perform with near equal efficiency as 8 VCs. The remaining unused VCs are power gated to save power. The result also presents that for the traffic pattern *shuffle*, 2 VCs and for *butterfly*, 1 VC serve the purpose with very less degradation in throughput. This leaves us with more number of unused VCs which are again power gated to save considerable amount of power with a little impact in throughput and latency.

Power measurements for the proposed DNoC router are done for 1, 2, 4 as well as 8VCs active along with the consideration of power overhead introduced due to the extra hardware used. Similar power measurements are done for a baseline mesh router with 8 VCs per port for the comparison purpose. For a baseline mesh router having 8 VCs per port shows channel power consumption of 24.6mW per port. In DNoC router the additional hardware including the UCU unit adds 0.31mW of power overhead. Sleep transistors, introduced in the design for power gating, consumes 0.712mW of extra power per port with 8 VCs. Table 4.7 shows the

power consumption per port for both baseline mesh and proposed router with variable numbers of VCs active at a time. The results show that for the workload, not requiring all the VCs for packet communication can reduce power consumption drastically at individual router level. Calculation shows that with 4 active VCs, the proposed router saves up to 46% of total power per port in comparison to the baseline router. For the traffic conditions which need only 2 or 1 active VC, DNoC provides power saving up to 71% and 83.3% respectively. With all the VCs in active mode in few cases, the proposed router consumes 4.47% extra power than the baseline architecture due to the power consumed in the additional hardware. Table 4.7 holds the power consumption data when the router port is engaged in packet transfer. The proposed router can save even more power while the router port is inactive. It would consume as low as 3.17mW of power, including the extra hardware in its idle state.

4.5.9 Overhead and Scalability

This section discusses the overhead associated with ReDeSIGN framework and evaluates its scalability. In this work, we have augmented minimal hardware as most of the existing hardware modules are reused. The newly introduced PSB, and BWT bits are stored in the free slots of the packet header flit, and therefore add no overhead. The VC *status field* adds one-bit of memory overhead per VC. In our experiment, the maximum VCs per a router node is 50 (20-baseline + 30-extended) for a homogeneous system, that adds 50 memory bit cells to a router node in the worst-case scenario. The major overhead associated with ReDeSIGN is the arbiter module that becomes larger because of the presence of more number of VCs. For VA stage, the baseline architecture uses a 16:1 arbiter, whereas in case of ReDeSIGN, a 40:1 grouped IPRRA arbiter is used. A total of fourteen 4:1 IPRRA arbiter modules are used to construct a 40:1 grouped IPRRA arbiter. For two-stage SA, the baseline architecture uses five 4:1 arbiters in *VC Selection* stage (as there are five input ports and each input port has four baseline VCs), and five 4:1 arbiters in *Output Port Assignment* stage (as there are five output ports). In contrast, for the worst-case scenario, ReDeSIGN uses five 10:1 grouped IPRRA (as each input port has four baseline VCs and six extended VCs) in *VC Selection* stage, and five 4:1 arbiters in *Output Port Assignment* stage.

We synthesized our design in Synopsys DC compiler for $32nm$ technology and found the latency of the 40:1 grouped IPRRA as $0.688ns$. The latency of the largest arbiter controls the router clock cycle, and we observed this to be well below the desired router clock cycle that is $1ns$. We found the total area occupancy of the arbiter modules in a router of the baseline architecture to be $264.88um^2$ and in the ReDeSIGN framework to be $522.68um^2$. The additional memory bit cells used as VC *status field* occupies $520.4um^2$. The total area overhead in case of ReDeSIGN framework is estimated as $778.2um^2$ that is very small in comparison to the baseline router area, which is observed to be 0.054 mm^2. The overhead results of ReDeSIGN framework

Table 4.8: Operational Power Consumption of ReDeSIGN Framework

Mode	**Without PMC**	**With PMC**
Debug	1.68W	1.45W
In-field	1.42W	1.26W

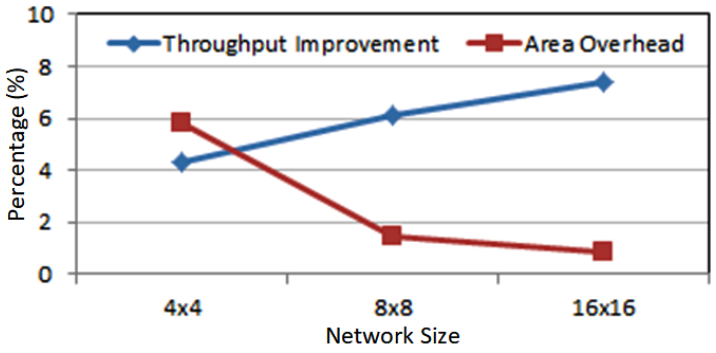


Figure 4.22: Scalability analysis of ReDeSIGN framework.

reported here correspond to the router node having the largest trace buffer share and thereby having the largest arbiter modules and maximum number of VC *status fields*. The overhead of the remaining router nodes of the system is even lower than the reported figures.

The operational power consumption values of a 8x8 homogeneous system for ReDeSIGN framework with and without the power management are computed. The experiments are performed using *Random* application for the simulation period of 100,000 cycles. The results are reported in Table 4.8 for both the debug and in-field operation modes. The results show higher operational power in debug mode than the in-field operation mode. This is because of the trace storage and trace transfer operations along with the normal operation during the debug mode. It is also observed that by using the proposed power management, 13.69% and 11.26% reduction in power consumption is achieved in debug and in-field operation mode respectively.

We have evaluated the scalability of ReDeSIGN framework using the *Random* application for three different system sizes, such as 4x4, 8x8, and 16x16 with 8KB of trace buffer for each system. Figure 4.22 shows a decreasing area overhead and increasing network throughput with the increasing size of the system. The significant decrease in area overhead is achieved due to the constant trace buffer size for all the systems, leading to smaller trace buffer share and smaller arbiters per router node in larger systems. In terms of network throughput, the ReDeSIGN framework would always benefit, as the number of final VCs would always be more than the baseline architecture. If the trace buffer size is increased for increased system size, then the area benefit would degrade, but the system performance would further increase. So, the proposed framework is considered as scalable and more beneficial for larger systems.

4.6 Conclusion

In this chapter, we propose ReDeSIGN, a novel debug structure reuse framework that enhances the performance of the system during the in-filed execution. The trace buffer is used as extended VCs of router nodes for improving the network throughput. Furthermore, the debug structures are also reused for critical data prioritization and starvation control. Experiments exhibit that ReDeSIGN framework provides an average of 11.46% increase in network throughput, 34.93% decrease in critical data latency, and 19.17% decrease in packet starvation for a 8x8 homogeneous system. The framework is observed to be beneficial for both homogeneous and heterogeneous systems, and is shown as a scalable solution. Moreover, this chapter also discusses DNoC, a dynamic VC power management scheme that is shown to be beneficial for large number of router VCs in variety of workload scenarios. Both ReDeSIGN and DNoC provide an efficient platform to reuse different debug modules during in-field operation to achieve enhanced system performance with very minimal overhead.

Chapter 5

Efficient and Secured WNoC MAC

The Chapter 3 establishes an efficient NoC debug platform WiND by augmenting wireless capability to the NoC. It also highlights the reuse of wireless infrastructure for payload communication during in-field operation mode. The data transfer on WiND platform works on the principles of Wireless NoC (WNoC) communication protocol. A Medium Access Control (MAC) protocol actually drives the data transfer efficiency of the WNoC. Therefore, an efficient MAC is desired that can eventually improve the debug capability of WiND and also can enhance the payload communication during the in-field operation mode. This motivates the objective of Chapter 5. This chapter proposes 2DMAC, a Dynamic MAC for the wireless channel allocation, to achieve the maximum gain. In traditional MAC, a token is circulated among the Wireless Interfaces (WIs) in a Round Robin (RR) manner. The WI with the token holds the channel for a fixed number of cycles. However, the channel requirement of the individual WIs dynamically changes over time due to the varying traffic density across the WNoC. Moreover, conventional WNoCs give equal importance to all the traffic taking the wireless path and transmit them in an oldest-first manner. Nevertheless, the critical data can degrade the system performance to a large extent by delaying the application run-time if not served promptly. Our proposed 2DMAC framework can change the token arbitration pattern and tune the channel hold time of each WI based on its run-time traffic density and criticality status. Moreover, 2DMAC prioritizes the critical traffic over the non-critical traffic during the wireless data transfer, leading to application speed up. Since, sharing of the channel among multiple WIs is controlled by the MAC, security threats to it can malfunction the system operation. Such attacks can be performed by a Hardware Trojan (HT) in a malicious WI or a rogue third party IP core present on the same SoC. This may result in *Denial-of-Service (DoS)* or *spoofing* in WNoC leading to starvation of healthy WIs and under-utilization of wireless channel. Therefore, this chapter demonstrates possible threat model on MAC and augments a security mechanism named Secure MAC to 2DMAC against *DoS* and *spoofing* attacks.

5.1 Motivation and Contribution

NoC is considered as a promising interconnect solution for on-chip data and protocol transactions on the modern multi-core systems [2, 3]. However, for long-distance on-chip communications, packets travel through multiple hops, resulting in higher power and performance overhead. WNoC alleviates this issue by providing single-hop communication between distant nodes [4]. WNoC augments multiple CMOS compatible Wireless Interfaces (WIs) operating in the millimeter-wave (mm-wave) frequencies on top of the conventional wired NoC to establish the low latency and low energy wireless medium [29, 30, 90]. The WIs are optimized to be placed far apart so that they can be used for long-range packet communication. The optimal number of WIs for a particular network size is computed according to simulated annealing based process [21, 31]. With the tremendous growth in VLSI technology, a large number of processing cores are getting integrated on a SoC [1]. Such advancements in the system integration would require the future NoCs to be extremely large in size and necessitate tens of WIs on the future WNoCs [91].

Most of the WNoCs adopt a single wireless channel for data communication to achieve a simple and low overhead implementation [21]. Even with the presence of multiple wireless channels [92], it is not possible to provide a dedicated channel to each WI. The WIs need to share the available wireless channel for data transmission and reception. A MAC mechanism is deployed that performs a fair arbitration and allocation of the wireless channel among all the WIs. At a particular instant, the channel is assigned to a single WI for a fixed number of cycles, so that the conflict and contention on the wireless medium are avoided. Traditionally, a Time Division Multiple Access (TDMA) based token passing MAC (T-MAC) is used for channel allocation [25, 93, 94] as shown in Figure 5.1. In this method, a token in the form of a physical packet is circulated among all the existing wireless hubs (routers augmented with the WIs) on a ring-based path, and the WI that holds the token gets the access of the channel. In general, the token passing method follows a static RR arbitration and is assigned to each WI for a fixed period of time (channel hold time). The maximum channel hold time of individual WIs are either kept the same or determined through the static profiling at the design time. The latter is performed based on the average bandwidth need of the WIs according to the applications mapped onto the WNoC. The efficiency of a wireless-enabled system is majorly driven by the success of the token passing mechanism, as its failure degrades the interconnect performance to a greater extent [95].

Though the implementation of T-MAC is simple, it is unable to cater the stringent power and performance needs of the state-of-the-art systems. The traffic pattern as well as the traffic density on an NoC is not uniform throughout the entire duration of the operation. It varies both spatially and temporally across the whole NoC. All the routers are not utilized identically even for *Uniform Random* traffic due to the implementation of a

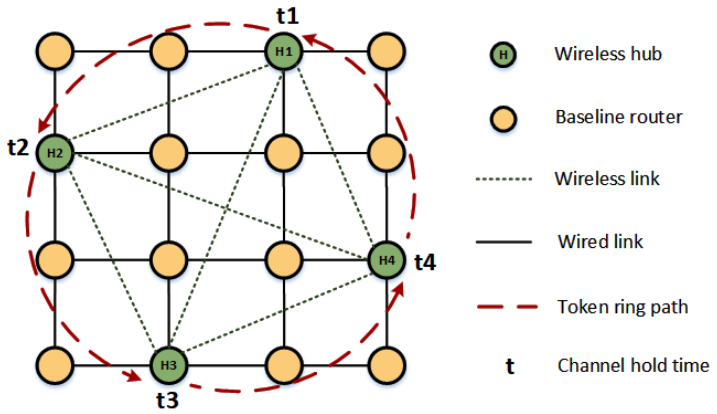


Figure 5.1: Conventional TDMA based token passing MAC (T-MAC) in WNoC.

particular routing algorithm [96]. Moreover, the data packet size varies in the number of flits depending on the type of payload. Furthermore, the contemporary heterogeneous system architectures introduce even more skewed traffic patterns [88]. The variation in traffic pattern across the NoC routers gets reflected in the load at each WI. This leads to the variation in wireless channel bandwidth demands by the WIs of the WNoC. Thus, a static T-MAC is considered to be inefficient that may lead to poor wireless channel utilization and degraded system throughput. Furthermore, the traditional T-MAC does not consider data criticality while transferring packets over the wireless channel. The conventional schemes forward packets to the transmitter based on their age in the wireless output buffer, in an oldest-first manner. But the system performance degrades considerably if the critical packets are not served on time. This leads to scenarios like application execution stall [27], and increased off-chip memory access time [97]. Therefore, a dynamic MAC is desired that can judiciously vary the token arbitration pattern and the channel hold time of WIs based on their run-time traffic density and data criticality to achieve a greater benefit. In this regard, a few dynamic MACs are already proposed in the literature [26, 98, 99, 100, 101]. However, most of them just vary the channel hold time according to the traffic density conditions and still follow a static token arbitration pattern for channel allocation. Furthermore, none of them have considered data criticality while allocating the wireless channel and transmitting packets over the channel.

The efficiency of a WNoC MAC is typically evaluated based on the wireless channel utilization, network throughput, average packet delay, and power consumption. To overcome the limitations associated with the conventional T-MAC and the existing dynamic MACs, we propose a congestion aware 2-stage Dynamic MAC (2DMAC) [102, 103] to achieve better efficiency in terms of the above-mentioned metrics. A centralized MAC structure can access to the global information of the bandwidth requirements of all the WIs, but it incurs significant routing and performance overhead in transferring the control signals [26]. Moreover, the whole system would malfunction, if the attacker can intrude the centralized controller. Therefore, 2DMAC is implemented as a distributed medium access control mechanism as discussed in Section 5.5. It is a ranking based mechanism,

where the wireless channel allocation and the channel hold time calculation for the WIs are performed according to their ranks. The proposed technique takes into consideration of the critical load density along with the non-critical load density at every WI, while computing the ranks. Critical packets are given priority over non-critical ones during data release at all the wireless transmitters. Our proposal is a truly dynamic MAC as both the token arbitration pattern and the channel hold time are dynamically varied to achieve improved wireless channel utilization and network throughput, while minimizing the latency of the critical traffic.

Wireless channel being a shared medium, security in WNoC channel access can be a major concern. While multiple WIs access the same wireless channel to transfer their data, a malicious wireless hub can hold the channel unauthorizedly to introduce *DoS* in system operation, or can *eavesdrop* or *spoof* to leak critical information. This way, either the system performance drastically degrades or privacy of the system is compromised. Integration of various untrusted third-party IP cores, as well as detection difficulties of small HT circuits on contemporary MPSoCs increase the probability of unauthorized wireless channel access. This motivates another contribution of this chapter, where we illustrate the possible security thread model for WNoC MAC and propose a low overhead decentralized solution as Secure MAC [104] augmented to 2DMAC system.

5.2 Background and Related Work

This section is broadly divided into two parts, such as "WNoC MAC Background and Related Work" and "WNoC Security Background and Related Work".

5.2.1 WNoC MAC Background and Related Work

Usually, a MAC mechanism is used that allows the on-chip WIs to transfer data over the shared wireless channel without any interference. Traditionally, a low overhead TDMA based MAC (T-MAC) is popularly used in WNoCs [25, 94]. In this case, a token is circulated among all the WIs for channel allocation, and is held by only one WI at a particular time instant. The corresponding WI with the token can transfer data over the wireless channel for a certain number of hold cycles without any interference. Though the implementation of the conventional T-MAC is simple, it is agnostic of the run-time bandwidth requirements of the WIs and therefore is considered to be inefficient in meeting the stringent demands of the contemporary systems. An efficient MAC mechanism can significantly reduce the congestion on a WNoC and improve the performance of the whole system [95]. Therefore, researchers have proposed different techniques related to architectural optimizations, suitable routing algorithms, application mapping, and congestion aware mechanisms to enhance

the efficacy of the MAC protocol [91, 105]. The design of congestion aware MAC mechanisms provides a noticeable opportunity to enhance the performance of a WNoC based system [26, 98, 99, 100, 101, 106, 107, 108]. The authors in [106] discuss a technique that dynamically switches to Carrier Sense Multiple Access (CSMA) MAC for low traffic loads and TDMA MAC for high traffic loads. Another congestion aware MAC is proposed in [98] that assigns the token to the WI with the largest buffered data waiting time, while ensuring the receiving WI is congestion free and can receive the transmitted data. An adaptive algorithm is developed in [99] that works along with a token sharing scheme, which dynamically allocates wireless channel bandwidth in response to demands from different cores. The authors in [107] present *NeuMAC*, a deep learning based adaptive MAC that can gain experience from the traffic patterns, and dynamically adjust the protocol to deal with different applications running on the WNoC-based system. A *Fuzzy Token* based WNoC MAC protocol is proposed in [108] that dynamically switches between token-based *focussed* and contention-based *fuzzy* mode to take care of the *bursty* and *hotspot* traffic.

A dynamic Radio Access Control Mechanism (Dynamic RACM) is proposed in [26] that distributes the unused clock cycles among the WIs in proportion to their channel utilization in the previous token round (a complete traversal of the token across all the WIs). However, this method does not take into consideration of the current channel requirements of the WIs. The authors in [100] have suggested a *moving average* based approach to predict a more accurate bandwidth demand by the WIs for the next token round. They have proposed two techniques such as Proportional Slot Allocation Mechanism (P-SAM) MAC and Demanded Slot Allocation Mechanism (D-SAM) MAC to consider both dynamic bandwidth requirements and dynamic token round period (total duration of a token round). The authors in [101] propose a dynamic and traffic aware MAC (Dynamic T-MAC), where they have used *exponential smoothing* method to predict the bandwidth requirements of the WIs for the next token round and claim that it has lower prediction error than the *moving average* method. The above discussed mechanisms rely on the previous bandwidth utilization data to predict the current bandwidth requirements of the WIs. However, with varying workloads and dynamic task mapping, the above methods are very much likely to introduce high prediction errors. To eliminate the prediction errors, we have proposed 2DMAC that computes the bandwidth requirements of the WIs based on the current congestion status and utilizes the channel bandwidth accordingly during the same token round.

There are several proposals that discuss the performance improvement of wired NoC based systems by prioritizing the critical data on the network during the resource allocation [27, 89, 109]. The work in [89] specifies different on-chip network packets into latency critical traffic and non-critical traffic. It shows that even though the percentage of on-chip critical traffic is low, by prioritizing it, the critical path delay is reduced and thereby the throughput of the system is significantly enhanced. The NoC prioritization policy in [109]

Table 5.1: Congestion Aware MAC Mechanisms : A Comparison

Scheme	**Bandwidth demand computation**	**Technique used**	**Parameters varied**	**Critical data**
Chang et al. [25] (T-MAC)	Statically computed at design time, fixed bandwidth allocation to each WI in every token round	Proportional to the average bandwidth needs based on the applications	Fixed channel hold time, fixed token arbitration	Not considered
Palesi et al. [26] (Dynamic RACM)	Dynamically computed based on congestion status in previous token round	Distribution of unused slots from previous token round	Varied channel hold time, fixed token arbitration	Not considered
Mansoor et al. [100] (P-SAM & D-SAM)	Dynamically predicted based on congestion status of previous token rounds	Moving average method	Varied channel hold time, fixed token arbitration	Not considered
Gao et al. [101] (Dynamic T-MAC)	Dynamically predicted based on congestion status of previous token rounds	Exponential smoothing	Varied channel hold time, fixed token arbitration	Not considered
Proposed work (2DMAC)	Dynamically computed based on congestion status of current token round	Ranking based mechanism	Varied channel hold time, varied token arbitration	Considered

accelerates the L2 cache access requests, resulting in overall program speedup. The authors in [27] discuss about dynamic priority of critical traffic based on packet latency slack and show considerable improvement in system throughput by prioritizing them. However, to the best of our knowledge, none of the existing work discuss on prioritizing the critical packets during the wireless channel allocation in a WNoC based system. In this proposed work, we have considered the critical traffic density both during the channel bandwidth allocation and wireless data transfer to achieve improved system performance. A brief comparison of the state-of-the-art MAC mechanisms along with the proposed 2DMAC is listed in Table 5.1.

5.2.2 WNoC Security Background and Related Work

There are few research works those have focused on some of the security aspects of WNoC. Authors in [110] have demonstrated a small-world network based *DoS* resilient wireless NoC architecture. But the proposed platform only deals with the *DoS* attack due to an HT in a core that triggers excessive injection of garbage traffic into the network. Work in [111] implements a secured broadcast-based coherence protocol for WNoC. A recent work has shown the use of machine learning classifiers to distinguish between jamming based *DoS* attack and random burst errors in WNoC [112]. The same work deploys a lightweight data scrambling method as a prevention mechanism against eavesdropping attack. A module named *Veritas* is implemented in [113] to prevent spoofing by RF power profiling. Authors in [114] have developed a security framework named

Prometheus to deal with *DoS*, *spoofing*, and *eavesdropping* attacks in WNoCs. However, none of these work talks about the security challenges in WNoC MAC. The MAC protocol can be malfunctioned owing to unauthorized access of the wireless channel by any of the malicious wireless hub present in the network. A hub is considered to be malicious if it has an HT present inside it, or it is connected to a rogue third party IP. Such a malignant wireless hub can either disturb the channel access arbitration pattern or it might manipulate the channel hold time. In this work, we illustrate the possible security thread model for WNoC MAC and propose low overhead decentralized countermeasures.

5.3 Traffic Disparity Across Wireless Hubs in WNoC

In an NoC based multi-core system, it is often observed that the traffic distribution is not uniform all over the network both spatially and temporally. This is majorly due to the nature of applications, dynamic task mapping, different data types, varied payload size, presence of heterogeneous cores, and the placement of shared caches and memory controllers. As a result, a significant load variation is also observed at the WIs in case of a WNoC. This section illustrates a few experimentally observed results highlighting the traffic disparity across the wireless hubs in a WNoC in terms of traffic density and data criticality. The experiments are performed on a 8x8 2D mesh WNoC with 8 WIs. The system configuration is given in Table 5.4 of Section 5.8.

5.3.1 Traffic Density on WNoC

The traffic density within the buffers of the NoC router nodes is typically skewed [88, 96]. Therefore, the level of wireless load experienced by all the WIs on the WNoC is expected to be dissimilar at a particular time instant [100]. Furthermore, the amount of traffic also varies across a single WI with respect to time. Figure 5.2 shows the buffered load condition across eight different WIs for *Uniform Random* and *Transpose* traffic for a duration of 2000 simulation cycles (from 2000 till 4000 simulation cycles). We have divided the total duration into 10 epochs of each 200 cycles length and reported the average load for each epoch. It can be observed that the traffic densities vary widely across the WIs both spatially and temporally. Therefore, a MAC that can dynamically sense the congestion status of the WIs during bandwidth allocation and accordingly tune the channel hold times would be beneficial. Another important observation highlighted by dotted circles in Figure 5.2 is that the traffic density is too much skewed in multiple instances. These are the occasions, where almost all the channel bandwidth is demanded by a very few number of WIs. Such a scenario can be efficiently exploited if the MAC can favor the corresponding WIs, while allocating the wireless bandwidth during that period. It is expected that in future large systems with many WIs and increasing heterogeneity, such extreme

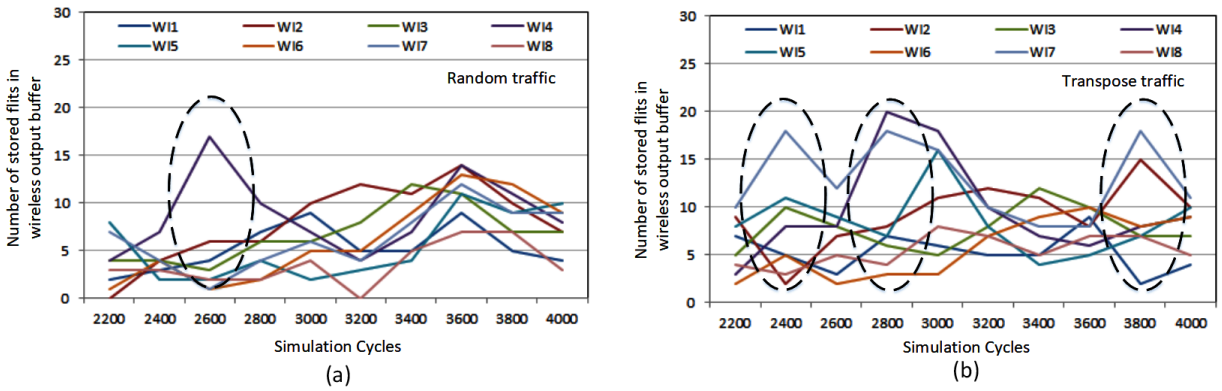


Figure 5.2: Varied traffic density both spatially and temporally for (a) *Random* and (b) *Transpose* traffic.

skewed traffic conditions would be more frequent. This encourages to design a dynamic MAC that can also provide a favored bandwidth allocation while ensuring minimum starvation in the non-favored WIs.

5.3.2 Critical Traffic on WNoC

Traffic on the contemporary SoCs is composed of packets with varied level of criticality in nature. There are certain packets that can hinder the application processing if delayed and thereby degrade the performance of the system to a large extent (critical packets). Whereas, there are other packets on the network that can be stalled for a few more cycles without having any impact on the application run-time (non-critical packets). Data packets in read, write transactions, acknowledgment signals, protocol requests, and words on a few cache lines are considered as performance critical [89]. On the other hand, the packets in eviction and write-back transactions, and words on a few other cache lines are considered to be non-critical. Furthermore, the packets associated with the off-chip memory access such as L2 cache miss packets are considered to be time critical [27] as the latency between request and response in such off-chip transactions are very high [115]. WNoC is a shared interconnection medium, where packets need to share network resources such as buffer space, wireless transceivers, and both wired as well as wireless links on their path. In case of resource contention, the detained critical packets may cause the processor to stall. Providing preferences to critical packets in terms of prioritized resource access would significantly enhance the execution speed of applications. Moreover, the latency critical packets are typically short, and the critical traffic accounts for a relatively small portion of the overall network traffic [89]. Table 5.2 indicates the observed percentage of Critical Load of the total Wireless Traffic (% CLWT) for four SPLASH-2 [63] and four Rodinia [86] benchmarks for the simulation length of 100,000 cycles. The table shows that on an average, only 19.21% of the total wireless traffic is found to be critical. This provides the opportunity to prioritize the critical traffic during wireless bandwidth allocation and at the same time ensuring the minimal non-critical packet starvation. Furthermore, the critical load density also varies both spatially

Table 5.2: Percentage Critical Load in Wireless Traffic (% CLWT)

SPLASH-2 workloads	**% CLWT**	**Rodinia workloads**	**% CLWT**
barnes	12.6	backprop	21
fft	26	hotspot	20.2
radix	22.6	lud	16.1
raytrace	15.8	bfs	19.4
Average % CLWT		19.21	

and temporally across the WIs. The WIs close to shared L2 caches and memory controllers are expected to experience more critical traffic. This supports the idea of designing a dynamic MAC that provides a favored wireless bandwidth allocation.

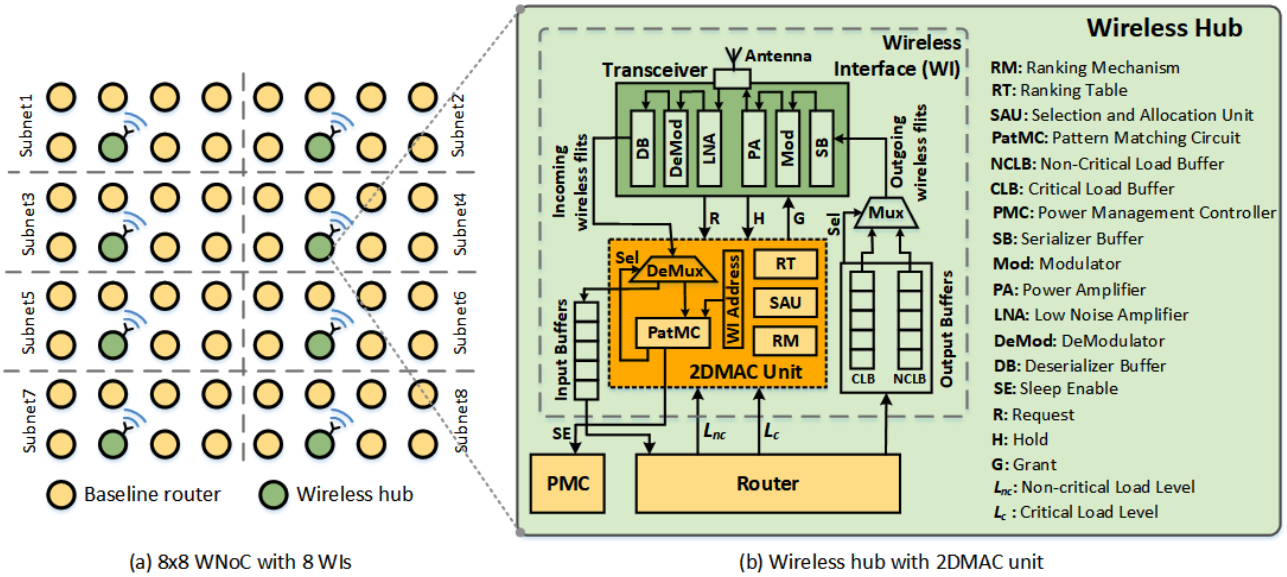
5.4 2DMAC System Architecture

This section discusses the considered system architecture and the structural modifications introduced to the system for the implementation of the 2DMAC framework in WNoC. We have considered a 8x8 2D mesh wireless network topology with eight WIs and a single wireless channel. The network is divided into eight *Subnets* with one wireless hub within each *Subnet* as shown in Figure 5.3 (a). The system clock frequency is set at 1GHz at which the baseline router along with the 2DMAC unit operate. The variation of packet size ranges from 4 to 16 flits on the considered system, and a flit size is taken to be 32 bits.

5.4.1 Modified Wireless Hub

Wireless hub integrates WI to the wired router and incorporates the MAC mechanism. The WI consists of a transceiver module for converting the digital data to RF domain and vice versa, an antenna to transmit and receive radio waves, and a MAC module to grant the channel access to the transceiver as shown in Figure 5.3 (b). The primary components of the transmitter module are serializer, modulator, and power amplifier, whereas the receiver side consists of low noise amplifier, demodulator, and deserializer. We have considered On-Off Keying (OOK) modulation and an omni-directional zig-zag antenna operating at 60 GHz for wireless data transmission with 16Gbps data rate [21, 116]. A flit transmission time (t_{fl}) over the wireless medium is computed to be 2 clock cycles as shown below in Equation 5.1.

$$t_{fl} = 32\ bits * (1GHz/16Gbps) = 2\ clock\ cycles \tag{5.1}$$



(a) 8x8 WNoC with 8 WIs

(b) Wireless hub with 2DMAC unit

Figure 5.3: (a) A 8x8 2D mesh WNoC, (b) Modified wireless hub with 2DMAC unit.

5.4.2 2DMAC Unit

The primary components of the 2DMAC unit are shown in Figure 5.3 (b). The proposed method introduces a Ranking Mechanism (RM) that performs a traffic aware ranking of all the WIs. This is performed locally within the 2DMAC unit of each of the wireless hub. The ranking of the WIs are maintained in a Ranking Table (RT). A Selection and Allocation Unit (SAU) is employed that selects a few higher ranked WIs for data transfer over the wireless medium in every token round, and allocates required channel hold cycles only to them. A Pattern Matching Circuit (PatMC) is implemented within the 2DMAC unit for the purpose of destination WI identification, virtual token arbitration, and *Sleep Enable* (*SE*) signal generation, which are discussed in subsequent sections.

5.4.3 Traditional Wireless Channel Allocation

A single wireless channel is shared among all the available WIs in a WNoC for data transfer. Conventionally, a token is circulated in a RR manner on a token ring path for channel allocation. Whenever a transceiver has data to transfer, it sends a $Request$ (R) signal to the MAC unit (Figure 5.3 (b)). As soon as the token is available with the corresponding wireless hub, it activates the $Grant$ (G) signal so that the transceiver will be able to access the channel. The transceiver asserts the $Hold$ (H) signal immediately after it gets the channel access and starts transferring the data packets. The $Grant$ signal is deactivated by the controller whenever either of channel hold time or WI load becomes zero. Consequently, the $Hold$ signal gets deasserted for the current

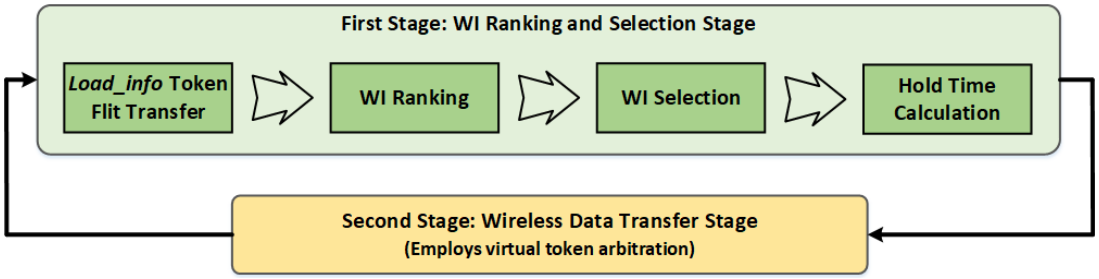


Figure 5.4: Different stages and operations in a single token round of 2DMAC framework.

token round, and the token is transferred to the next WI on the arbitration path. Traditionally, each WI is given a Static Hold Time (*SHT*) based on their average bandwidth requirement for the mapped applications. The Token Round Period (*TRP*) is defined as the summation of the *SHT*s of all the WIs present on the network, as shown in Equation 5.2.

$$TRP = \sum_{i=1}^{n} SHT_i \tag{5.2}$$

Here, i is the WI index, and n is the total number of WIs on the WNoC. The value of TRP is a constant to ensure that any jth WI can get the access of the wireless channel for SHT_j cycles for every $\sum_{i \neq j} SHT_i$ cycles in the worst case scenario. This results in avoiding any WI starvation. However, in case of such static MAC, any jth WI experiencing a lot of traffic at a particular time can at most use the channel for SHT_j cycles, even if the other WIs do not use the $\sum_{i \neq j} SHT_i$ cycles completely. This degrades the system performance and leaves a scope for improvement. In this work, a dynamic MAC mechanism is proposed that intelligently allocates the channel to the most deserving WIs so that the whole TRP can be utilized in a most effective manner resulting in a greater system performance. The next section discusses the details of the proposed 2DMAC framework.

5.5 2DMAC Framework for Wireless Channel Allocation

This work proposes 2DMAC, a unique traffic-aware dynamic MAC mechanism for WNoC based system. In this framework, a single token round is divided into two stages, as shown in Figure 5.4. The first stage performs a traffic aware WI ranking operation and then selects a few high ranked WIs that can access the wireless channel for data transmission in the second stage. The working principles of both the stages of 2DMAC are presented in the following subsections. Table 5.3 lists all the variables and parameters of the proposed 2DMAC framework. This table also lists all the acronyms used specifically for the 2DMAC framework. The common acronyms used in the whole thesis can be referred from the "List of Acronyms" at the beginning of this thesis.

Table 5.3: List of Acronyms, Different Parameters and their Definitions used in 2DMAC Framework

Parameters/ Acronyms	**Definitions / Full Forms**	**Parameters/ Acronyms**	**Definitions / Full Forms**
n	Total number of WIs	m	Number of selected WIs for 2^{nd} stage
WI_i	WI number i	$SWI_{p/c}$	Selection status of WI in the previous/current token round
SHT_i	Statically defined hold time of ith WI	AHT_i	Dynamically allocated hold time to ith WI
TRP	Total time period of a token round	$RTRP$	Remaining time of a token round
t_{fst}	Time consumed in 1^{st} stage by $load\_info$ token flits transfer	t_{fsc}	Time consumed in 1^{st} stage by all the computations
L_{nc}	Non-critical load level of a WI	L_c	Critical load level of a WI
α_1	transformation weight for L_{nc}	α_2	transformation weight for L_c
TL_{act}	Total actual load of a WI	TL_{tran}	Total transformed load of a WI
STL_{act}	Sum of all WI's total actual load	ATL_{tran}	Average of all WI's total transformed load
t_{TL}	Time required to transfer all WI's total actual load	t_{TLs}	Time required to transfer selected WI's total actual load
t_{fl}	Time required to transfer one flit	RHT	Remaining hold time
Tx_{Load}	Data transmitted through antenna	ST	Simulation time
$Load\_info$	Traffic Load Information	SE	Sleep Enable
LE_n	Load enable	${LE_n}^*$	Modified load enable
${LE_{nrht}}^*$	Load enable to RHT register	${LE_{nswi}}^*$	Load enable to Source WI register
CLB	Critical Load Buffer	CLWT	Critical Load in Wireless Traffic
CSMA	Carrier Sense Multiple Access	DoS	Denial of Service
D-SAM	Demanded Slot Allocation Mechanism	eNI	Network Interface to electrical router
HT	Hardware Trojan	LLB	Load Level Buffer
NCLB	Non-Critical Load Buffer	PatMC	Pattern Matching Circuit
P-SAM	Proportional Slot Allocation Mechanism	RACM	Radio Access Control Mechanism
RM	Ranking Mechanism	RT	Ranking Table
SAU	Selection and Allocation Unit	TDMA	Time Division Multiple Access
T-MAC	TDMA based token passing MAC	wNI	Network Interface to wireless interface

5.5.1 First Stage: WI Ranking and Selection Stage

The first stage of the proposed framework includes broadcasting of the traffic load information ($Load\_info$) of the individual WIs, then ranking of the WIs based on their $Load\_info$ and finally the WI selection and the hold time allocation. All the operations of the first stage are illustrated in Figure 5.5.

5.5.1.1 WI Ranking Scheme

The WI ranking computation in the 2DMAC unit of every wireless hub is done once at the beginning of each token round. Initially, all the WIs relay their own $Load\_info$ in the form of a token flit in a RR manner using the broadcasting capability of the wireless infrastructure (Figure 5.5 (a)). The controller constructs the token flit by inserting the *Source WI* address, the non-critical load level (L_{nc}) and the critical load level (L_c) of the

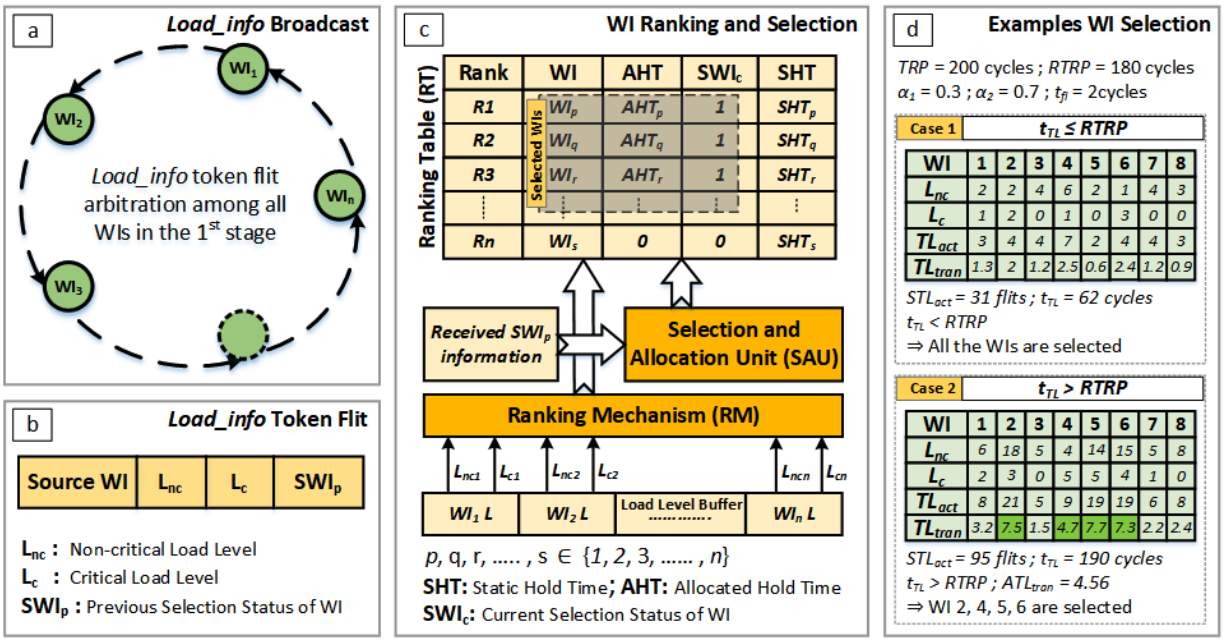


Figure 5.5: Detailed illustration of different operations in the first stage of 2DMAC framework.

corresponding hub as shown in Figure 5.5 (b). Moreover, the token flit carries an additional bit that indicates the Selection status of the WI (SWI_p) in the previous token round. All the receivers capture the L_{nc} and L_c values from the token flits and stores them in their own Load Level Buffer (LLB). The transmitting hub also latches its own L_{nc} and L_c values in its own LLB. In this way, all the WIs have homogeneous information about the current load status of all other WIs. These load statistics are used to construct a fresh WI Ranking Table (RT) using the Ranking Mechanism (RM) implemented in the 2DMAC unit of the each wireless hub. Since, all the WIs have the view of global load information (stored in local LLB), the RT is coherent across all the WIs.

Both L_{nc} and L_c contribute towards the total load in a wireless hub. Since, the L_c is crucial for improving the system performance, ranking the WIs based on the actual total load (TL_{act}) (Equation 5.3) will not provide the utmost performance. Therefore, we have considered a transformed total load (TL_{tran}) for the WI ranking as shown in Equation 5.4.

$$TL_{act} = L_{nc} + L_c \tag{5.3}$$

$$TL_{tran} = \alpha_1 * L_{nc} + \alpha_2 * L_c \tag{5.4}$$

Here, α_1 and α_2 are the $load\ transformation\ weights$ assigned to non-critical and critical loads respectively with the condition $\alpha_2 > \alpha_1$. These weights are user-defined and decided at the design time based on the

Algorithm 4 Algorithm for 2DMAC First Stage

1: **initial:** *i = 1 ; j = 1;* ▷ *i* represents WI number, and *j* represents RT index
2:
3: **while** $i <= n$ **do** ▷ Ranking Scheme
4: $L_{nc_i} \leftarrow$ non-critical load of i
5: $L_{c_i} \leftarrow$ critical load of i
6: $TL_{tran_i} \leftarrow \alpha_1 * L_{nc_i} + \alpha_2 * L_{c_i}$
7: *i++*
8: **end while**
9: $RT[1:n] \leftarrow sort(TL_{tran_{1:n}})$
10:
11: **if** $t_{TL} \leq RTRP$ **then** ▷ WI Selection Scheme
12: select all *n* WIs
13: **else**
14: select all WIs whose $((TL_{tran} \geq \ ATL_{tran})||((SWI_p == 0)\&\&(L_c \neq 0)))$
15: **end if**
16: assign *m = number of WI selected*
17:
18: **if** $t_{TL} \leq RTRP$ **then** ▷ Hold Time Allocation Scheme
19: **while** $j <= n$ **do**
20: assign $AHT_j = 2 * TL_{act_j}$ cycles
21: *j++*
22: **end while**
23: **else if** $t_{TLs} \leq RTRP$ **then**
24: **while** $j <= m$ **do**
25: assign $AHT_j = 2 * TL_{act_j}$ cycles
26: *j++*
27: **end while**
28: **else**
29: **while** $j <= m$ **do**
30: assign $AHT_j = RTRP * \frac{TL_{act_j}}{\sum_{j=1}^{m} TL_{act_j}}$ cycles
31: *j++*
32: **end while**
33: **end if**

number of WIs, application types, average critical load density and average critical load latency. For the experimental purpose, we have considered α_1 = 0.3 and α_2 = 0.7. The load transformation ensures that the ranking of WI is biased by the load criticality and not completely driven by the actual load size. Once the $Load\_info$ token flits are broadcasted from all the WIs, the RM block collects L_{nc} and L_c values of all the WIs from the LLB as shown in Figure 5.5 (c), and computes the TL_{tran} of individual WI. Then it prepares the RT for all the WIs by using the *Ranking Scheme* presented from line number 3 to 9 in Algorithm 4. The WI with the largest TL_{tran} gets the rank 1 and the WI with the smallest TL_{tran} gets the lowest rank. In case of equal TL_{tran}, the WI with higher L_c value is ranked higher, else assigned the same rank.

5.5.1.2 WI Selection and Hold Time Allocation Scheme

Once the ranking of WIs is performed, the next step is to select a few WIs that will be eligible for data transfer over wireless channel in the second stage. The channel hold time for the current token round is allocated only to the selected WIs. A *WI Selection Scheme* and a *Hold Time Allocation Scheme* are presented from line number 11 to 16 and line number 18 to 33 respectively in Algorithm 4. Both the schemes are explained below.

The complete duration of a token round (*TRP*) is decided at the design time by adding the *SHT* values (Equation 5.2). The RT holds the *SHT*s of individual WIs as shown in Figure 5.5 (c). The selected WIs can transfer wireless data at most for the remaining duration of the *TRP* (*RTRP*), after the first stage, as presented in Equation 5.5.

$$RTRP = TRP - (t_{fst} + t_{fsc}) \tag{5.5}$$

Here, t_{fst} and t_{fsc} are the time consumed by transferring the $Load\_info$ token flits and by all the computations respectively in the first stage. If the traffic density is low at all the WIs, and it is found that $t_{TL} \leq RTRP$, then all the WIs are selected for the second stage. Here, t_{TL} represents the time required to transfer the total wireless load ($\sum_{i=1}^{n} TL_{act_i}$). In this case, the number of hold cycles allocated to each of the WI is equal to the twice of its TL_{act}, as each flit takes two cycles to be transferred over wireless medium in our proposed architecture. In this way, the total wireless traffic stored in the output buffers of all the WIs can be transferred within the pre-defined *TRP*. A similar scenario can be seen in *Case 1* example of Figure 5.5 (d). However, in case t_{TL} is found to be larger than $RTRP$, the 2DMAC framework would select a few WIs for the second stage. The number of WIs selected would be biased by the skewed nature of the traffic in terms of both density and data criticality. The *Selection Mechanism* only selects the WIs whose $TL_{tran} \geq ATL_{tran}$. Here, ATL_{tran} is the average of TL_{tran} of all the WIs. In most of the cases, this mechanism would ignore the WIs with very low load density and with only non-critical data. Thus, it can provide a few more hold cycles to the WIs with high load density and with buffered critical data. Such a scenario can be observed in the *Case 2* example shown in the Figure 5.5 (d). The channel hold time for the selected WIs are computed as twice of the buffered flits if $t_{TLs} \leq RTRP$. Here, t_{TLs} is the time required to transfer the total load of all the selected WIs. In case $t_{TLs} > RTRP$, the hold times are calculated proportional to the respective TL_{act} of the selected WIs as shown in line 30 of Algorithm 4. The channel hold time is assigned to $Zero$ if the WI is not selected for the second stage. The allocated hold times (AHT_i) of the WIs are updated in the RT. The proposed WI selection may lead to a scenario where a few WIs may not be selected to the second stage for multiple consecutive token rounds. This is fine as long as the corresponding WI is having only non-critical data, as they can be stalled without

having any significant impact on the system performance. However, it raises a concern, if such a WI has a very little amount of critical data present in its output buffer. To avoid starvation of these WIs, the *Selection Mechanism* checks the corresponding SWI_p bits received from the *Load_info* token flits. If the SWI_p bit for such a WI is found to be '0' and it has some non-zero L_c value, then the corresponding WI is selected for the second stage, irrespective of its rank in the current token round. The Selection and Allocation Unit (SAU) also updates the corresponding SWI_c value of each WI in the RT based on their selection status in the current token round. The discussed schemes incorporated in the first stage of the 2DMAC framework enable the system to provide the access of the wireless channel to the WIs according to their bandwidth need.

In case of 2DMAC, not only the allocated hold time (AHT) of selected WIs are dynamic, but the duration of token round period is also dynamic. In 2DMAC, $TRP \leq \sum_{i=1}^{n} SHT_i$, unlike the static TRP ($= \sum_{i=1}^{n} SHT_i$) in case of T-MAC. This enables the framework to use every single cycle of the token round for data transmission of the selected WIs and at the same time ensuring minimum starvation of the non-selected WIs.

5.5.2 Second Stage: Wireless Data Transfer Stage

In the second stage, the selected WIs get the access of the wireless channel to transfer the buffered data for the allocated hold time. All the operations of the second stage are illustrated in Figure 5.6. The channel is first assigned to the top ranked WI for data transfer. The remaining selected WIs get the access of the channel according to their rank through a token arbitration as shown in Figure 5.6 (a). The token arbitration pattern is varied in the 2DMAC framework, as it follows the ranking sequence and not the traditional RR path. This is performed to provide quick access of the channel to the WIs that have more buffered critical data and/or having high load density. By assigning the wireless medium to the WIs in the order of their need, higher system performance is achieved. The physical token transfer is associated with performance penalty and consumes a significant energy especially in the case of low load scenarios [106] (for example, the scenario presented in *Case 1*, Figure 5.5 (d)). Therefore, we have implemented a virtual token passing mechanism for the second stage.

5.5.2.1 Virtual Token Passing Mechanism

The proposed method transfers the token virtually by augmenting few additional information other than the *Source* and *Destination* address of the packet to the unused bits of the packet header flit. The format of the wireless packet and the header flit structure is shown in Figure 5.6 (c). The first entry (P_{nc}/P_c) indicates that the packet is a critical load if its value is '1' and non-critical otherwise. Source and destination WI addresses

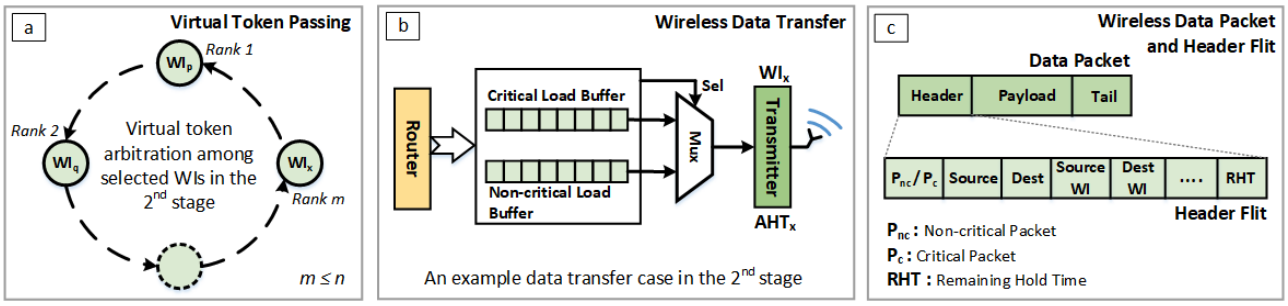


Figure 5.6: Detailed illustration of different operations in the second stage of 2DMAC framework.

($Source\ WI, Dest\ WI$) are inserted for token passing and receiver address matching respectively. Finally, *RHT* indicates for the remaining channel hold time of the transmitting WI in the current token round.

During wireless data transmission, all WIs except the transmitting one, act like receivers. The data packet received at the receivers is checked for address matching using PatMC circuit. If the WI address is matched with $Dest\ WI$ of the packet header, the respective wireless hub is known to be the desired receiver of the packet. The packet is further processed in this hub and is discarded from all other hubs. $Source\ WI$ of the packet header is also checked at each receiver to presume which WI would get the next token. The receiver hub that comes next to the transmitting hub in the ranking sequence processes the $Source\ WI$, and finds that it will be the next to get the token. The next WI candidate keeps monitoring the *RHT* value of the transmitted packets. As soon as *RHT* becomes *Zero*, automatically the transmitting WI releases the wireless channel and the next WI candidate holds the token as well as the channel.

5.5.2.2 Load Release Strategy

The transmitter starts sending data packets over the wireless channel as soon as it gets the token. The data transfer architecture is demonstrated in Figure 5.6 (b) and the scheme is discussed in Algorithm 5. The proposed 2DMAC framework gives preference to the critical load over the non-critical load by transmitting the critical packets first during the channel access, so that the application execution speed can be enhanced. This allows the framework to release the whole L_c in most of the instances, and a share of L_{nc} in the remaining channel hold time during a single token round. Total wireless load provided by the router is segregated into non-critical and critical load based on the P_{nc}/P_c value in the header flit, and are queued in the respective Non-Critical Load Buffer (NCLB) and Critical Load Buffer (CLB) as shown in Fig. 5.6 (b). The wireless output buffer can be divided into NCLB and CLB, while keeping the size of the CLB relatively smaller as the percentage and the length of the critical packets are smaller in comparison to non-critical packets [89]. Age of packets

Algorithm 5 Algorithm for 2DMAC Second Stage

initial: *i = 1 ; k = 1;* ▷ *i* represents WI number, and *k* represents RT index

assign token to WI_k ▷ Token Passing and Load Release
while $k <= m$ **do**
    grant channel access to WI_k
    $Tx_{Load_k} \leftarrow L_{c_k}$
    $RHT \leftarrow AHT_k$
    while $(RHT \;\&\&\; Tx_{Load_k})! = 0$ **do**
        release load L_c of WI_k
        decrement RHT of WI_k
    end while
    if $Tx_{Load_k} == 0$ **then**
        $Tx_{Load_k} \leftarrow L_{nc_k}$
        while $(RHT \;\&\&\; Tx_{Load_k})! = 0$ **do**
            release load L_{nc} of WI_k
            decrement RHT of WI_k
        end while
    end if
    release the wireless channel
    pass the token to WI_{k+1}
    k++
end while

are taken into consideration while queuing both type of loads. First, the critical data transmits through the antenna (Tx_{Load}) till the time the *RHT* becomes *Zero*. If the critical data finishes first, the non-critical data gets transmitted for the remaining hold time period, as shown in the Algorithm 5. Due to the prioritization of the critical packets in the adopted $Load\ Release\ Strategy$, there may be some instances, when a very small channel bandwidth is left for the non-critical packets. Since, the percentage of the critical traffic on the network is low, the non-critical load always finds at least a few slots to be transmitted from the selected WIs in every token round. However, the non-critical traffic in the WIs those are not selected for the second stage in consecutive token rounds may starve for the wireless bandwidth. In this case, such non-critical packets are transferred to their destination through the wired NoC path. Though the average delay of these packets increases, it has negligible impact on the overall system performance due to its non-critical nature.

The load release efficiency can further be improved by transferring one data packet along with the $Load\_info$ token in the first stage. The $Load\_info$ bits can be embedded as the trailing bits or can be accommodated in the available free space of the header flit of the transmitting data packet. This way, 2DMAC can utilize almost all the first stage cycles for data transmission and minimize the start-up latency.

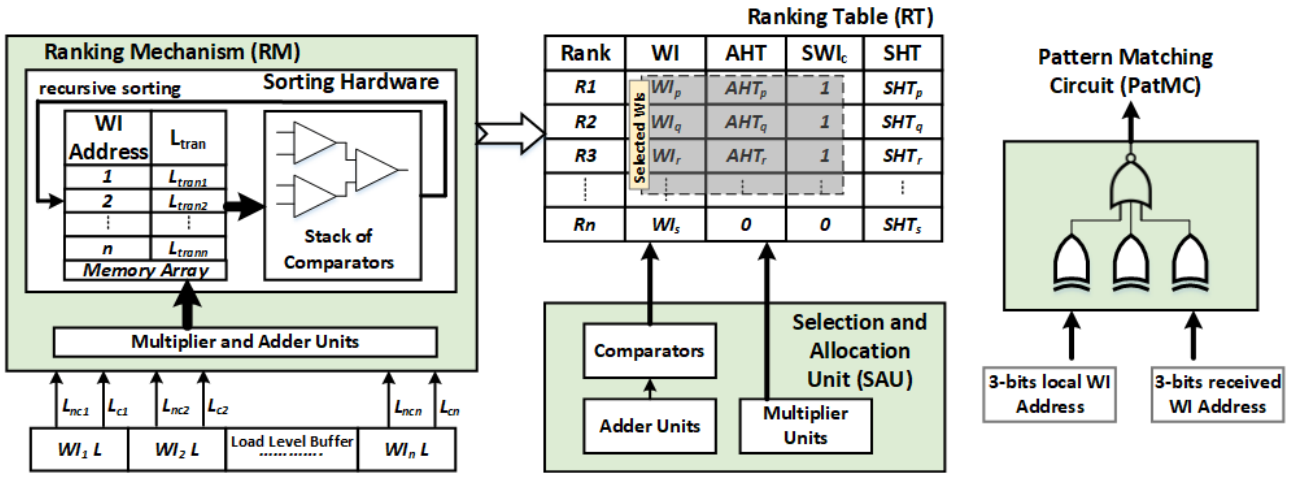


Figure 5.7: Hardware implementation of different units within 2DMAC module.

5.6 Hardware Details and Power Management of 2DMAC Framework

This section discusses hardware implementation details of every unit in the proposed 2DMAC, along with the power management scheme that is built around the framework.

5.6.1 Hardware Implementation details of 2DMAC

The 2DMAC module is composed of RM, RT, SAU, PatMC and the LLB units. The circuit details of all the units inside 2DMAC module is shown in Figure 5.7. The LLB is a memory array that stores the load status of individual WIs. In our experimental setup, we have considered 8 WIs and 32 flits wireless output buffer (refer Table 5.4). So, maximum 3 bits and 5 bits are required to represent the *WI Address* and load level respectively. The total size of the LLB is taken as 104 bits that stores the *WI Address* (3 bits), L_c (5 bits), and L_{nc} (5 bits) of all 8 WIs. RM module collects the load information and using the 5-bits multiplier and adder units computes the L_{tran} (6 bits) of individual WIs (refer Equation 5.4). These L_{tran} data are kept in the *Memory Array* of the *Sorting Hardware* inside the RM module. The stack of comparators recursively sort the L_{tran} values of the WIs in the *Memory Array*. For our experiments, 72 bits of *Memory Array* and 6-bit comparators are used inside the RM unit. RT module is again a 2-dimentional memory array of 110 bits size. Each row of RT holds *WI Rank* (3 bits), *WI Address* (3 bits), AHT (8 bits), SWI_c (1 bit), and SHT (7 bits). The SAU block incorporates 8-bit comparators for WI selection. It includes 8-bits arithmetic adders for calculating the total loads (TL_{act}, TL_{tran}) and 8-bits multipliers for calculating the WI hold time (AHT). We have adopted a low overhead Vedic multiplier from [117] to save area and power. SAU also includes an 8-bits shift register for

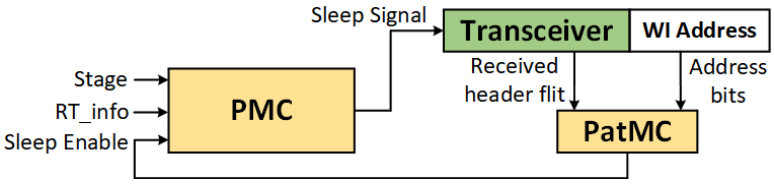


Figure 5.8: Power management of 2DMAC framework.

low overhead multiplication of the load with *'2'* while calculating the AHT in case $t_{TL} \leq RTRP$ (shown in Algorithm 4). The PatMC module is implemented using Ex-OR gates, that matches the 3-bit local *WI Address* with the 3-bit received *WI Address*.

5.6.2 Power Management of 2DMAC

The wireless hubs are more power hungry compared to baseline routers because of the presence of WI and large buffers [118]. Different techniques are proposed in the literature to control the power management of WIs [119, 58]. Our proposed 2DMAC framework exploits the PatMC unit used for virtual token passing for providing efficient power management control to the transceivers. The PMC unit shown in Figure 5.8 takes *Stage* as one of the input that indicates the system is operating in the first stage or in the second stage. In the first stage, all the transceivers are kept active as each has to broadcast a single *Load_info* token flit one after another and the remaining transceivers have to receive the flits. The PMC activates and power gates the transceiver module by sending a '0' or a '1' respectively over the *Sleep Signal*. The ranking as well as selection of WI and the hold time allocation is completed in the first stage. So, the RM and the SAU modules of 2DMAC unit are power gated after the first stage, and are activated again in the beginning of the next token round. In the second stage, only two transceivers get involved in the data transfer. So, as soon as a transceiver sends its header flit of the first data packet, the PatMC unit in the remaining WIs matches the *Dest WI* address with their own address and feeds its output to the PMC through the *Sleep Enable* signal. The PMC of the wireless hub keeps the corresponding transceiver active if its address matches with the *Dest WI*, and the remaining can power gate their inactive transceivers. The PMC gets the *AHT* information of the transmitting WI from the *RT_info* input, and reactivates the power gated transceiver at the completion of the corresponding *AHT*. Moreover, the selected WIs also check the *Source WI* address of the received header flit. If a WI find its address to be the next in the RT, then it assumes itself to be the next candidate to get the token for the data transmission. So, the PMC keeps the corresponding transceiver active so that it can continuously observe the *RHT* of the transmitting WI. As soon as the *RHT* becomes *Zero*, the token is virtually transmitted to the snooping WI candidate. The benefit of this method is *no token loss*, in contrast to the traditional method, where the token flit gets lost in multiple occasions when it reaches a power gated WI.

5.7 Secure MAC : A Security Mechanism for WNoC MAC Protocol

Distributed and dynamic MAC mechanism is preferred in WNoC over centralized and static MAC to achieve higher system performance. The proposed 2DMAC is such a distributed and dynamic framework, which is already discussed in Section 5.5. Here, the MAC controller inside each wireless hub computes the *AHT* of the corresponding hub and inserts the *RHT* of the hub to each transmitting packet. Every wireless packet communication on such network, informs the source address and the *RHT* of the transmitting WI to the remaining hubs. The packet header flit as shown in Figure 5.6 (c) is modified by the controller to embed the *Source WI* and *RHT* information that indicate who and when will get the next access of the channel. Whenever a packet header is received by a hub, its PatMC unit checks for the *Dest WI*, *Source WI*, and *RHT*. If the *Dest WI* matches to its own address, then it starts receiving the *payload* of the packet. If the *Source WI* appears to be the address of the previous hub in the RT, then the hub interprets that it is the next to get the access of the channel, and it holds the token if the *RHT* is found to be zero.

5.7.1 Possible Threat Model

In a decentralized MAC, if any local controller is corrupted by some means, then the *Source WI* and *RHT* information in packet header flit can be manipulated to hold the wireless channel in an unauthorized manner. The presence of a simple HT circuit can easily be triggered to change the header flit values. Such flit manipulation can lead to *DoS* or *spoofing* attacks. These attacks would result in poor utilization of wireless channel and would drive the victim hubs to starvation.

5.7.1.1 Denial-of-Service (DoS)

In this work, *DoS* refers to the unavailability of the wireless channel for a particular wireless hub whenever the channel is used by another hub in an illegitimate way. This can be done by a malicious hub by periodically changing its *RHT* in the header flit of the transmitting packets with an intention not to release the wireless channel. The *RHT* of the malignant hub would never be seen as zero, and the next wireless hub waiting for the channel would be misguided that the previous node has the rightful access of the channel.

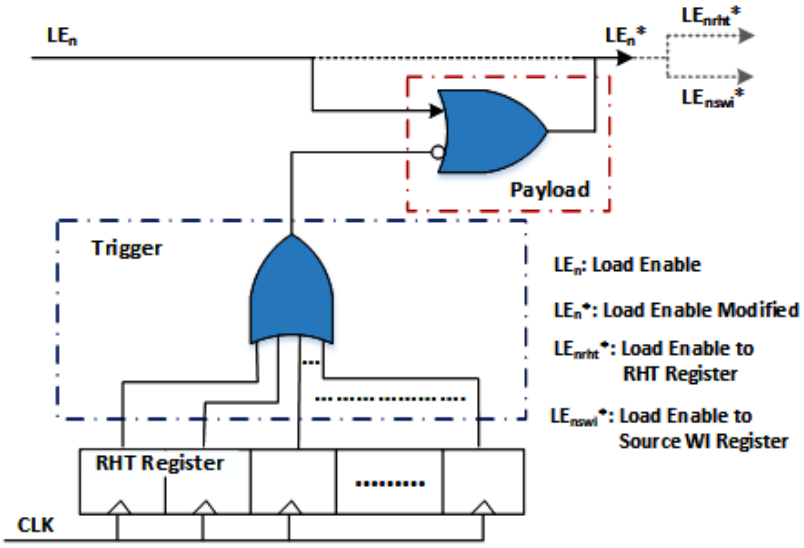


Figure 5.9: Time-bomb Trojan in the malicious hub.

5.7.1.2 Spoofing

In case of *spoofing*, the malicious hub would change the source address of the packet being transmitted on the wireless channel during its last access cycle. It is done by manipulating the *Source WI* in the packet header flit as the source address of the previous hub. This would mislead all the remaining hubs interpreting that the malicious one is the next to get the access of the channel. This way the malicious hub would continue to hold the channel.

5.7.1.3 Trojan Attack Activation

Both *DoS* and *spoofing* attack scenarios can be triggered by a simple Trojan circuit implanted inside the hub as shown in Fig. 5.9. The circuit used is a sequential *Time-bomb Trojan* [120]. *Trojan Trigger* takes the inputs from the *RHT Register* and generates the trigger signal when all bits of the register becomes zero. *Trojan Payload* modifies the *Load Enable* (LE_n) signal whenever there is a trigger. During the *First Stage* of the token round (As discussed in Section 5.5.1), LE_n signal in each of the hubs gets activated to load the *RHT Register* and *Source WI Register*. Both these registers provide values to be updated in the corresponding fields of packet header flit before transmission. The *RHT Register* is decremented based on load release. The Trojan circuit exploits the decrement nature of the *RHT Register* to generate a *Time-bomb Trigger* during the last channel access cycle of the corresponding hub in the *Second Stage* of the token round. The *Payload* circuit modifies the LE_n to generate ${LE_n}^*$ that enables unauthorized loading to either *RHT register* (${LE_{nrht}}^*$) leading to *DoS* attack or *Source WI Register* (${LE_{nswi}}^*$) leading to *spoofing* attack.

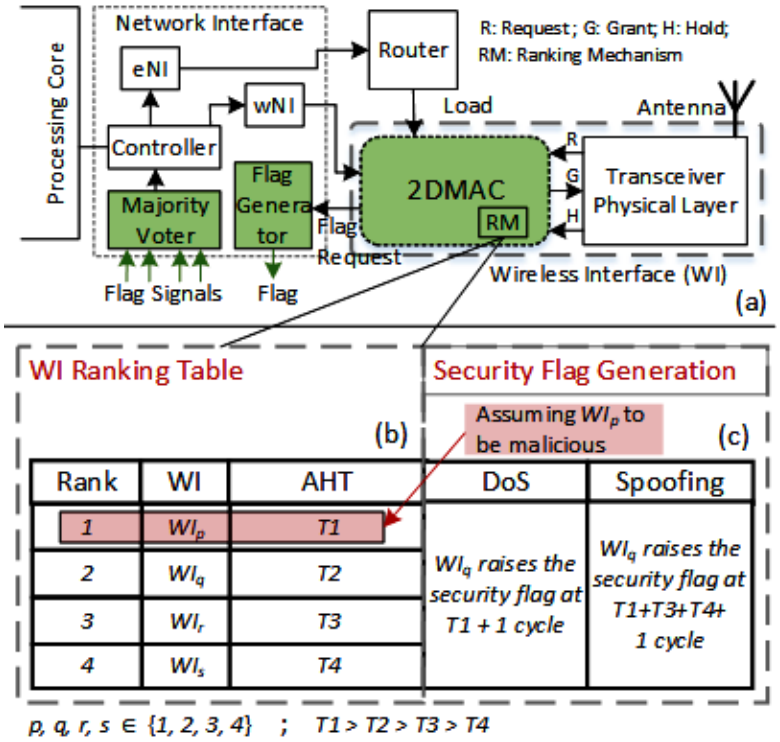


Figure 5.10: (a) Proposed ranking mechanism and security architecture, (b) Ranking table, (c) Security flag generation.

5.7.2 Proposed Security Countermeasure

This section discusses the security countermeasure based on the assigned *AHT* and the actual hold time values. Figure 5.10 (a) demonstrates our proposed Secure MAC that includes the security blocks such as *Majority Voter* and *Flag Generator* in the Network Interface (NI). The NI that provides separate interfaces to the electrical router (eNI) and to the WI (wNI) is assumed to be completely secured and free from attacks. Figure 5.10 (b) illustrates a part of RT that shows the respective *AHT* values, and Figure 5.10 (c) demonstrates the security flag generation for *DoS* and *spoofing* attacks.

Any malicious wireless hub can manipulate the *RHT* or *Source WI* information while transmitting a packet to hold the wireless channel in an unauthorized way, as discussed in Section 5.7.1. To deal with this, we have proposed a distributed self-defence mechanism by flag generation based on actual access time of any WI as shown in Figure 5.10 (c). The mechanism works during run-time and takes a corrective action whenever there is a violation between actual channel access duration by a WI and the corresponding *AHT* in the RT. For an example case, we have considered a 64 node, 4 WI, 2D mesh NoC whose RT is shown in Figure 5.10 (b). We are assuming WI_p is a malicious hub. Whenever WI_p holds the channel beyond T_1 by manipulating its *RHT*, *DoS* security threat is detected by WI_q (at $T1 + 1\ cycle$ as shown in Figure 5.10 (c)). The 2DMAC module

Table 5.4: System Configuration and Simulation Setup

System Component	**Configurations**	**Network Component**	**Configurations**
Core	8 x86 OoO cores, 40 GPU cores	Topology	8x8 baseline 2D mesh WNoC, 8 WIs, XY routing, 8x8 WNoC with 4 WIs for Secure MAC
L1 Cache	Size - 64 kB, Line Size - 64 B, Associativity - 4	Router	5 I/O ports, 1 I/O wireless port (for wireless hub), 8 flit router buffers, 32 flit wireless buffers, 4 to 16 flit packets, 32 bit flits, 1GHz clock
L2 Cache	Size - 256 kB, Line Size - 64 B, Associativity - 8	Wireless Interface	OOK modulation, zig-zag antenna, 60 GHz carrier, 16 Gbps bandwidth, 200 cycles *TRP*
L1/L2 Cache replacement policy	LRU	Workload	Synthetic - *Random, Transpose, Butterfly*; SPLASH-2 - *Barnes, FFT, Radix*; Rodinia - *Backprop, Hotspot, Lud*

in WI_q requests its *Flag Generator* to raise a *DoS attack flag* and sends the same to all other WIs over the wired NoC paths. To make an unbiased attack detection and correction, we have proposed a majority voting mechanism implemented in the NI module of each of the wireless hub. The *DoS attack flag* is received by all other WI hubs present in the network, and is evaluated locally based on the *AHT* information present in the local RTs. In case of a real violation, all other WIs would raise *Support flags*. All the flags would travel on the wired NoC path till the NI of the malicious hub. The associated *Majority Voter* module will collect all the flags and would disable the wNI if found guilty. This leaves the node with only the wired router for data transfer and suspends its wireless capability. In case of *spoofing* attack, the correction mechanism will be same with a small difference in the detection mechanism. In this scenario, the *spoofing attack flag* will be generated by WI_q at $T1 + T3 + T4 + 1\ cycle$. This is because at $T1$ time, WI_p would manipulate the packet source address (*Source WI*) as WI_s to confuse the other wireless hubs. While receiving the packet and matching the source address, WI_q would misinterpret that WI_p is the next candidate to have the token. So, it would not generate the flag at $T1 + 1\ cycle$. To detect such scenarios, we have proposed to consider the maximum possible idle time of a WI during a token round. In this case, when WI_q would not get the flag within $T1 + T3 + T4$ (maximum idle time of WI_q), it will raise a *spoofing attack flag* in the next cycle.

5.8 Experimental Results

This section presents the complete system configuration, network topology, and the simulation setup as shown in Table 5.4 and discusses experimental results for evaluating the 2DMAC and Secure MAC frameworks. A 48 core CPU/GPU heterogeneous system architecture is adopted for our experiments. A 8x8 baseline 2D mesh WNoC based system with 8 WIs is developed on the cycle-accurate Noxim simulator [50] for the generation

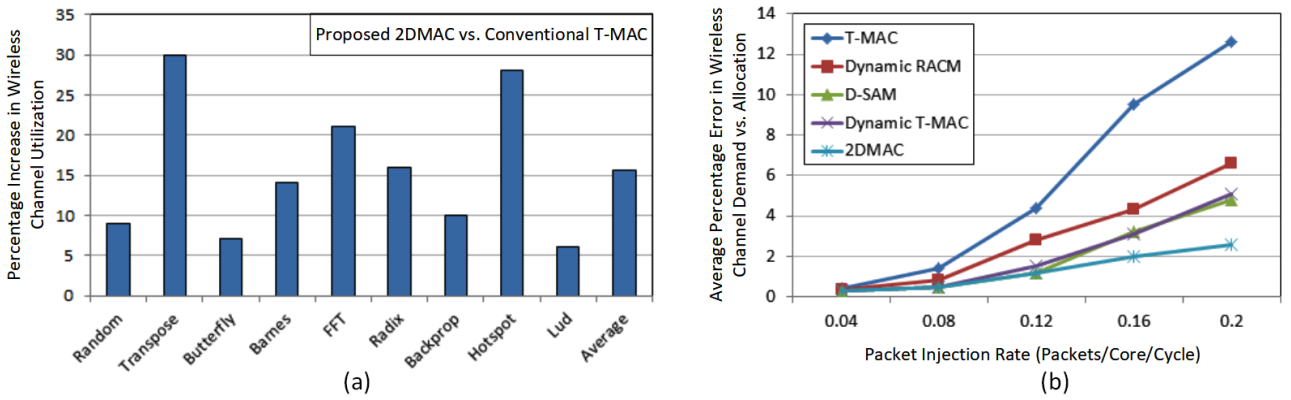


Figure 5.11: (a) Percentage increase in wireless channel utilization in proposed 2DMAC over conventional T-MAC approach, (b) Average percentage error in wireless channel demand vs. allocation for increasing packet injection rate under a *Uniform Random* traffic scenario.

of the results. For scalability analysis, a 4x4 WNoC with 4 WIs and a 16x16 WNoC with 16 WIs are also modeled on the same simulator. To evaluate the efficacy of the proposed method, we have executed 3 synthetic workloads, 3 applications from SPLASH-2 benchmarks [63], and 3 applications from Rodinia benchmarks [86] on the developed system. Full system simulations are performed on gem5gpu tool [85] for collecting the network traces. The network level experiments are performed for the total Simulation Time (*ST*) of 100,000 cycles. Results are generated and analyzed for wireless channel utilization, average packet latency, network throughput, communication energy etc. The proposed 2DMAC framework is compared with the conventional T-MAC [25], and the state-of-the-art Dynamic RACM [26], D-SAM [100], Dynamic T-MAC [101] methods.

5.8.1 Wireless Channel Utilization

The wireless capability of a WNoC is best exploited when the channel idle time is minimized. This means the channel should not be idle if any load meant for wireless transfer is waiting at any of the wireless hub. In case of conventional T-MAC approach, as each WI gets the access of the channel for a fixed period, there might be cases when the channel is assigned to a WI that has no packets to transfer. This results in higher channel idle time, which leads to poor wireless channel utilization. On the other hand, 2DMAC being a congestion-aware dynamic channel allocation mechanism, it allocates the channel bandwidth to the WIs according to their run-time demand. Thus, it increases the wireless channel utilization over the T-MAC approach, which is shown to be 15.67% on average, from our experiments in the Figure 5.11 (a).

To evaluate our proposed framework in comparison to the state-of-the-art techniques, we calculated the average percentage error in the wireless channel demand versus the bandwidth allocation across all the eight

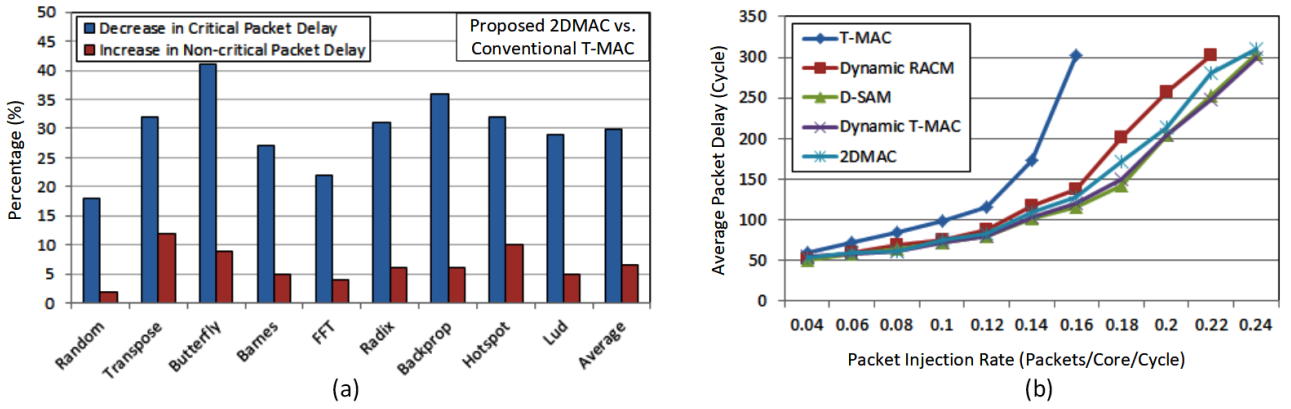


Figure 5.12: (a) Percentage decrease in critical packet delay and percentage increase in non-critical packet delay in proposed 2DMAC over conventional T-MAC approach, (b) Average packet delay for increasing packet injection rate under a *Uniform Random* traffic scenario.

WIs for a *Uniform Random* traffic scenario. This calculation was repeated for each of the considered techniques for different Packet Injection Rates (PIR) and the results are presented in Figure 5.11 (b). It can be observed from the result that at low PIR, the percentage error is very small for all the techniques. But, with the increase in PIR, the conventional T-MAC method is unable to sense the varied congestion status at different WIs, and thus the error between the bandwidth demand and the actual allocation increases considerably. The Dynamic RACM is a congestion aware technique, and it performs better than the T-MAC approach. However, it allocates the channel bandwidth to the WIs according to their congestion status in the previous token round. Therefore, in several instances, there exists a mismatch between the bandwidth demand and the allocation. Both D-SAM and Dynamic T-MAC also rely on the congestion status history of the WIs for multiple previous token rounds and applies different methods to predict their bandwidth requirements. We found, the percentage error is similar for both these techniques. On the other hand, the proposed 2DMAC technique computes the bandwidth requirements of the WIs based on their current load status. Therefore, it results in a very small error between the bandwidth demand and the allocation even for the high PIR.

5.8.2 Average Packet Delay and System Performance

The average packet delay of the network is evaluated in terms of average critical packet delay and average non-critical packet delay. For simulation, the percentage of critical load for SPLASH-2 and Rodinia workloads are referred from Table 5.2, and is assumed to be 20% for the Synthetic applications. Since the proposed scheme minimizes the waiting time of critical packets in the wireless hubs, it is expected that the total delay associated with the critical packets would reduce. It is found from the experiments that our method speeds up

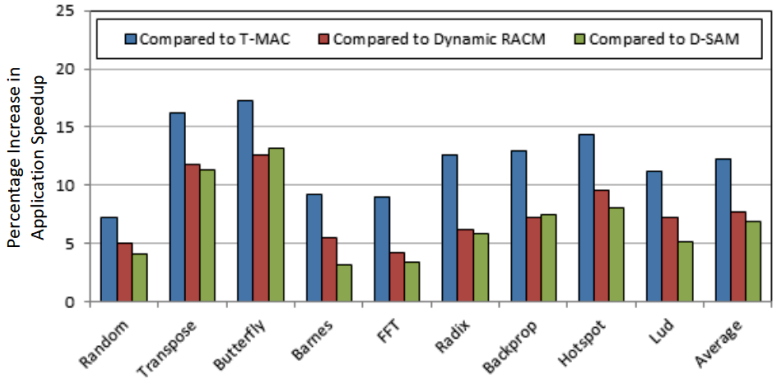


Figure 5.13: Percentage increase in application speedup in case of 2DMAC compared to the state-of-the-art.

the critical load to a great extent. The results presented in Figure 5.12 (a) shows an average of 29.77% decrease in critical packet delay on the wireless network in case of 2DMAC compared to the conventional T-MAC method. However, it can be observed in the same figure that 2DMAC shows 6.55% average increase in non-critical packet delay. This is because the non-critical load is transmitted from a WI after the critical load during its channel access period, and in some cases it takes the wired path if the WI is not selected for consecutive token rounds. Since, the non-critical packets have large slack values, their increased delay would have very minimal impact on the overall system performance. Whereas, the reduction in critical packet delay would speed up the execution of the applications and in return significantly improve the overall system performance.

We have also compared our framework with the other techniques in terms of average packet delay versus PIR, whose results are shown in Figure 5.12 (b) for a *Uniform Random* traffic. As expected, the result shows similar delay for all the techniques at low PIR. However, as soon as the PIR increases, the performance of T-MAC degrades and the network saturates early because of the higher average packet delay. This happens due to the poor wireless channel utilization of the conventional method. The remaining techniques, being congestion aware, utilize the wireless medium efficiently and have low average packet delay than the T-MAC approach. We found that both D-SAM and Dynamic T-MAC show similar and a little better delay figures than the Dynamic RACM technique. Our 2DMAC scheme performs close to the other state-of-the-art techniques and saturates almost at the similar time. With low average percentage error in wireless channel demand vs. allocation (as shown in Figure 5.11 (b)), it is expected that 2DMAC should show better latency-throughput characteristics compared to the other existing methods. However, which is not the case because of the high start-up latency consumed in the first stage (20 cycles in every 200 cycles TRP). To reduce the impact of the start-up latency and thereby further improve the latency-throughput characteristics, the possible solution is to broadcast the WI load information along with the actual data flits, as discussed in Section 5.5.2.2.

Though the proposed framework shows similar average packet delay with other dynamic MACs, the over-

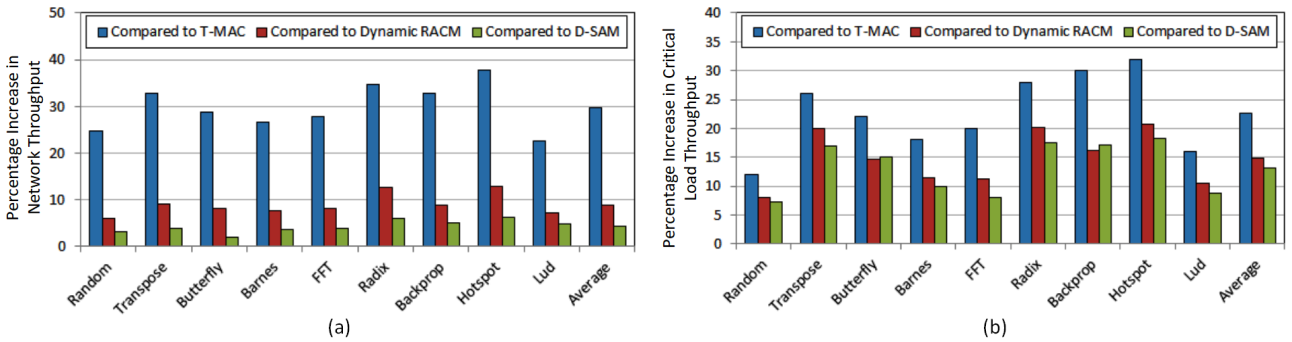


Figure 5.14: (a) Percentage increase in network throughput in 2DMAC over state-of-the-art, (b) Percentage increase in critical load throughput in 2DMAC over state-of-the-art.

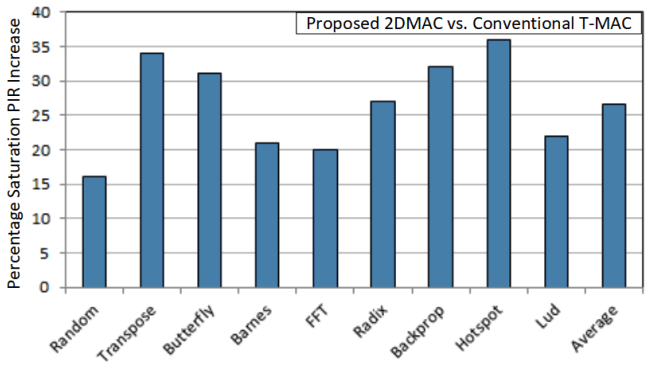


Figure 5.15: Percentage increase in saturation PIR of 2DMAC over T-MAC.

all system performance in case of 2DMAC is observed to be higher because of the appreciable speedup in application runtime at the system level. This is achieved due to the significant reduction in the average critical packet delay and improved wireless channel utilization. From the experiments, it is observed that our proposed framework achieves 12.21%, 7.71% and 6.86% application speedup compared to T-MAC, Dynami RACM and D-SAM respectively, as shown in Figure 5.13.

5.8.3 Network Throughput

2DMAC framework is also evaluated in terms of network throughput. Figure 5.14 (a) shows the percentage increase in network throughput in 2DMAC over the other state-of-the-art. It can be observed that our proposed framework achieves 29.83%, 8.94% and 4.32% higher throughput than the static T-MAC, Dynamic RACM and D-SAM respectively. These throughput improvements are achieved due to the better wireless channel utilization in case of 2DMAC. Even though the improvement in 2DMAC's network throughput over the other dynamic MACs are marginal, a considerable improvement in critical load throughput over all the considered MACs can be observed from the Figure 5.14 (b). Our scheme shows 22.67%, 14.8%, and 13.22% increase in

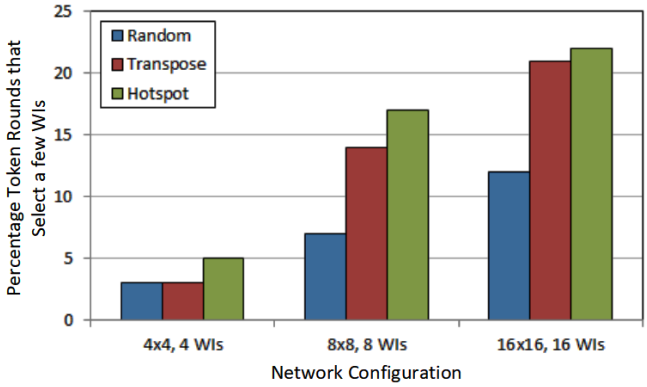


Figure 5.16: Percentage of token rounds that select a few WIs for wireless data transfer in the second stage of the 2DMAC.

critical load throughput compared to T-MAC, Dynamic RACM and D-SAM respectively. This is resulted as the proposed framework prioritizes the flow of critical packets over normal packets on the wireless path. The improvement in critical load throughput eventually contributes towards the application speedup and system performance improvement that we discussed in Section 5.8.2. We also observed, the throughput of the network saturated at a higher PIR value in 2DMAC than the T-MAC for all the workloads. The results are plotted in Figure 5.15 that demonstrates our framework shows an average of 26.55% increase in the saturation PIR.

5.8.4 Performance Evaluation with Skewed Traffic

In case of skewed traffic, the range of variation in the wireless channel bandwidth demand is high across the WIs. This scenario differs for different applications and becomes more prominent in heterogeneous system architecture. Though, the existing congestion-aware techniques adjust the channel hold time of the WIs depending on their requirements, the token needs to traverse across all the WIs even if a few of them do not need the wireless medium at that moment. This leads to a longer waiting time for the most demanding WIs, in case of a large system with many wireless hubs. One of the unique attribute of the 2DMAC framework is WI selection that picks a few deserving WIs in a token round for data transfer to deal with the skewed bandwidth requirement. To evaluate the framework in this regard, we have considered a *Uniform Random* traffic and two non-uniform traffic such as *Transpose* and *Hotspot*. In the *Transpose* traffic, each of the core communicates with the core that is located diagonally opposite to it, whereas in case of *Hotspot* traffic, multiple nodes communicate with a single core known as hotspot core. *Transpose* and *Hotspot* give rise to more skewed communication, resulting in congestion at a few of the WIs. Figure 5.16 shows the percentage of token rounds for which a few WIs (at most *n-1* WIs out of *n* WIs) are selected for data transfer in the second stage. The results are generated for 500 token rounds for all the three application traffic. We also considered three different system sizes such as 4x4

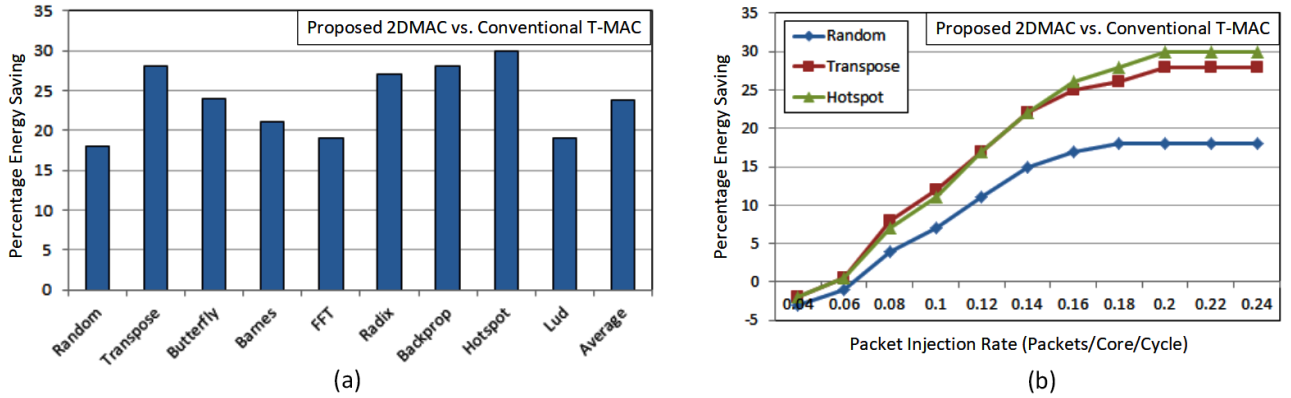


Figure 5.17: (a) Percentage increase in communication energy saving in proposed 2DMAC over conventional T-MAC, (b) Percentage increase in communication energy saving for different PIR of 2DMAC over T-MAC.

with 4WIs, 8x8 with 8 WIs and 16x16 with 16 WIs for our experiment. It is observed from the results that in case of more skewed traffic (*Transpose* and *Hotspot*), there exists more number of token rounds that selects a few WIs for data transfer compared to *Uniform Random* traffic. This becomes more prominent upto 21% and 22% for *Transpose* and *Hotspot* traffic respectively, when the system size and number of WIs increase. Thus, the proposed 2DMAC framework is effective in handling the skewed traffic scenarios which is more common in heterogeneous system architecture and also more suitable for future large systems with many WIs.

5.8.5 Communication Energy

The average communication energy of the network is observed for both the proposed and the T-MAC techniques. The introduction of the 2DMAC unit to the wireless hubs in our method adds power overhead to the system, as discussed in Section 5.8.7. However, the 2DMAC framework performs efficiently in terms of overall energy saving as shown in Figure 5.17 due to the considerable reduction in average network delay and efficient power gating scheme. We have divided the energy consumption in WNoC into three parts and highlighted the energy saving of 2DMAC in these different parts compared to conventional T-MAC. The three parts are (a) packet energy consumption while waiting in the wireless output buffer, (b) energy consumption during the packet communication, both during processing of the packet in the transmitter and during the packet transmission over the wireless channel, and (c) energy consumption by the MAC Unit, and the WI. For 2DMAC, the buffer waiting time of the packets significantly decreases compared to T-MAC due to the efficient channel allocation, as discussed in Section 5.5.1. This results in lower packet energy consumption in wireless buffers for 2DMAC in part (a). In part (b), energy consumption during the packet communication is the same for both 2DMAC and T-MAC assuming the transceiver and the wireless channel are the same for both the frameworks.

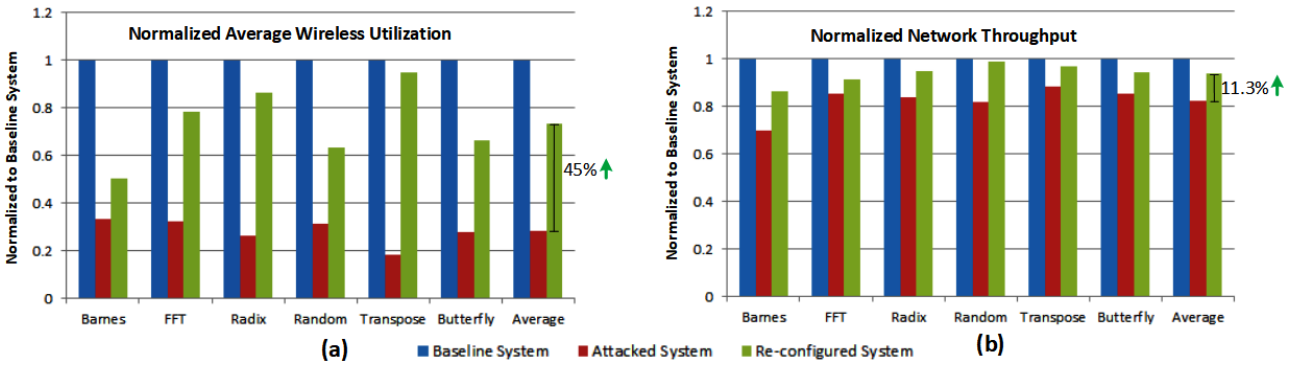


Figure 5.18: (a) Normalized average wireless utilization, (b) Normalized network throughput.

Lastly, in part (c) 2DMAC outperforms T-MAC in terms of power saving because of the power management scheme deployed in 2DMAC, as discussed in Section 5.6.2. During the inactive period, 2DMAC efficiently power gates different units of the MAC module and the WIs to save considerable amount of power. The results in Figure 5.17 (a) shows that 2DMAC consumes in an average of 23.77% less communication energy than the conventional T-MAC method.

We have also observed the energy consumption of both the frameworks for increasing PIR values for *Random*, *Transpose*, and *Hotspot* application traffic and presented the results in Figure 5.17 (b). The results show that 2DMAC consumes more energy compared to T-MAC in the low PIR region. This is because, for low value of PIR, the network load is very low and the channel allocation for both the schemes remains the same. This results in higher energy dissipation in case of the proposed method due to the additional power consumption of the 2DMAC unit. But, as soon as the PIR increases, the average packet communication delay of 2DMAC framework decreases drastically and thereby the energy consumption by the network decreases significantly. The result also demonstrates that for skewed traffic (*Transpose* and *Hotspot*), 2DMAC saves more energy as it is more responsive to the varied bandwidth demands of the WIs.

5.8.6 Performance Gain in Secure MAC Compared to Attacked System

The *DoS* and *spoofing* attacks degrade the system performance drastically due to the under-utilization of wireless medium. The proposed Secure MAC blacklists the malicious WI and re-configures the wireless network with the available healthy wireless hubs. This helps the system regains its performance close to the completely healthy one. We have modeled a 8x8 WNoC with 4 WIs for the simulation purpose. An HT is implemented by modifying the WI module, as discussed in Section 5.7.1. The simulation results are generated by executing 3 synthetic applications and 3 workloads from SPLASH-2 benchmark suite.

Table 5.5: Area Overhead Comparison with the state-of-the-art Dynamic MACs

	Dynamic RACM [26]	**D-SAM [100]**	**Dynamic T-MAC [101]**	**2DMAC** (Proposed work)
Technology node used	28nm	65nm	not reported	65nm
MAC unit area	$\sim$ 0.0077 mm^2	0.0017 mm^2	not reported	0.0082 mm^2
% of the wireless hub area	$\sim$ 1%	1%	not reported	2.31%

The experimental results are illustrated in Figure 5.18. We have executed the simulations in 3 system architectures such as (i) baseline system with 4 healthy hubs, (ii) malicious system with 1 attacked hub, and (iii) proposed reconfigured system with 3 healthy hubs. Figure 5.18 (a) represents the normalized average wireless utilization, and (b) represents the normalized network throughput for all the 3 system architectures. In case of a attacked system, the wireless channel is captured by the malignant hub all the time leading to drastic reduction in wireless utilization. By detecting and eliminating the attacked hub, the system regains a considerably high wireless utilization, which is around 45% higher than the attacked system. Due to the poor utilization of wireless channel, the whole network throughput also gets affected during the attack. Our proposed method re-configures the network with the available healthy hubs, and the wireless channel is shared between them. This enables the system to achieve around 11.3% better network throughput than the attacked system.

5.8.7 Overhead and Scalability Analysis

This section discusses the overhead associated with the 2DMAC framework and evaluates its scalability. The extra hardware modules introduced to 2DMAC (presented in Section 5.6.1) adds area and power overhead to the design. For the considered baseline WNoC (8x8 WNoC with 8 WIs), the 2DMAC module is synthesized using 65-nm technology in Synopsys design compiler. It is found that the 2DMAC unit occupies 0.0082 mm^2 of area and consumes 0.26 mW of power. The area of the transceiver module of the baseline WI designed using same technology node is 0.3 mm^2, and it consumes 36.7 mW of power [21]. The router module consumes 0.054 mm^2 of area and 1.56 mW of power. Thus, the area and power overhead by 2DMAC unit per wireless hub is found to be very small, that is 2.31% and 0.68% respectively. Table 5.5 provides a comparative analysis of MAC unit area overhead and its percentage with respect to the wireless hub area for all the considered state-of-the-art dynamic MAC mechanisms. It is observed that 2DMAC consumes a slight higher area than its counterparts. However, this area overhead can be overlooked, as the benefit in terms of system performance improvement is considerably higher for 2DMAC compared to its competitors, as shown in Figure 5.13. Moreover, at the system

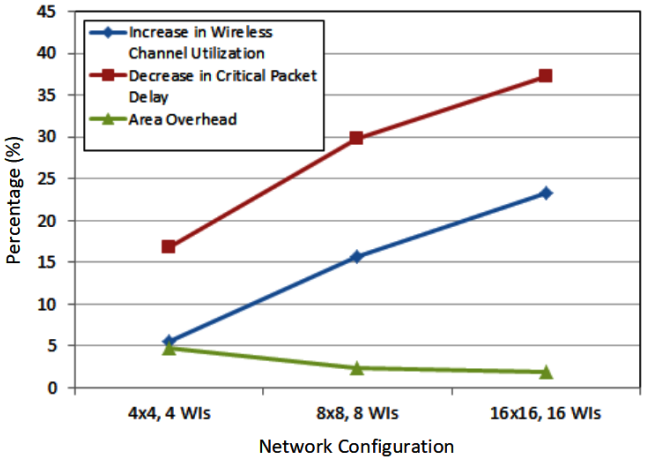


Figure 5.19: Scalability analysis of 2DMAC framework.

level, this area overhead becomes negligible as only 8 wireless hubs are present in a 64 nodes system. In terms of communication overhead, the 2DMAC unit consumes 3.62 ns of time for completing the WI ranking and hold time allocation operations, resulting in consumption of 4 clock cycles per one token round. It consumes $2 * n$ number of clock cycles for *Load_info* token flit circulation for n WIs in the first stage. So, for the considered baseline system, 20 cycles are consumed in the first stage and the remaining 180 cycles of a token round (*TRP* is considered as 200 cycles) are provided for wireless data transmission. The 2DMAC adopts a virtual token passing mechanism in the second stage, and it does not transfer any physical token flit during the payload data transfer. Considering the fact that all the state-of-the-art mechanisms transfer physical token flits during data transfer, the effective communication overhead of 2DMAC is considered to be only 4 clock cycles. In contrast, 2DMAC provides much higher application speedup benefit due to its effective channel allocation.

We have evaluated the scalability of the 2DMAC framework using the *Uniform Random* traffic for three different system configurations, such as 4x4 with 4 WIs, 8x8 with 8 WIs, and 16x16 with 16 WIs. The proposed framework is compared with the traditional T-MAC architecture and the results are presented in Figure 5.19. It shows that with increased system size, the area overhead of 2DMAC framework reduces. Though the overhead of the 2DMAC unit increases with more number of WIs, the WI count does not linearly increase with the size of the system. Therefore, the area overhead relative to the whole system size decreases. It can also be observed that with increased network configuration, the wireless channel utilization increases and the critical packet delay decreases. The WI ranking and selection technique in 2DMAC framework efficiently distinguish the WIs that need the wireless channel urgently and assign the channel accordingly. With more number of WIs on the network, these unique features of the proposed framework are proved to be more beneficial. Thus the 2DMAC framework is considered as a scalable and sustainable solution for the future WNoCs.

5.9 Conclusion

In this chapter, we have proposed 2DMAC, a novel congestion-aware MAC mechanism for WNoC based systems that dynamically allocates the channel to the WIs according to their requirements. The suggested framework incorporates a WI ranking scheme based on the load density and data criticality. It varies the token arbitration pattern by selecting a few higher ranked WIs and tunes their hold time based on their channel requirements. Experiments exhibit that 2DMAC framework improves the wireless channel utilization by 15.67%, the network throughput by 29.83%, and reduces the critical data latency by 29.77% for a 8x8 baseline system with 8 WIs. The framework is observed to be beneficial for skewed traffic as well as for many WIs system, and is shown as a scalable solution for the future WNoCs. Moreover, this chapter discusses Secure MAC, a distributed security framework for WNoC MAC protocol that provides around 11% performance benefit over the attacked system. Both 2DMAC and Secure MAC considerably improve the efficiency of WNoC and also secure the MAC protocol.

Chapter 6

Conclusions and Future Directions

In this work, we have focused on developing wireless enabled debug infrastructure for interconnection networks to improve the efficiency of post-silicon debug of multicore systems. We have also extended our work in exploring different reusability of the debug structures for system performance enhancement during in-field operation. Moreover, this thesis work presents efficient and secured MAC protocol for WNoC based systems. The major contributions of this thesis are:

1. WiND, a novel post-silicon debug infrastructure for interconnection network of multicore systems, is proposed in Chapter 3. It divides the whole NoC into sub-networks and embeds wireless interfaces to all the sub-networks. The wireless capability of the debug structure considerably enhances the trace transfer speed. It also reduces the generated trace size because of the minimized hop count in case of wireless test data transmission. WiND is augmented with RTE that enables the proposed debug structure to eliminate the redundant traces without compromising with the observability of the system. This further reduces the trace amount, leading to small size embedded trace buffer requirement.

2. Chapter 4 addresses the issue of debug hardware overhead by reusing the idle debug modules during in-field operation. The proposed ReDeSIGN framework reutilizes the embedded trace buffer as extended virtual channels, trace priority hardware for critical data prioritization and snapshot unit for starvation control. All these reuse capabilities of the framework enhances the application runtime and eventually the system performance significantly. Furthermore, ReDeSIGN is augmented with DNoC, a dynamic VC power management mechanism to minimize the power consumption of the increased number of VCs.

3. Considering the use of wireless infrastructure for debug purpose in WiND framework, Chapter 5 has focussed on the enhancement of the efficiency and security of MAC protocol of wireless NoC. 2DMAC is

developed as a 2-stage dynamic MAC that can change the token arbitration pattern and tune the wireless channel hold time depending on the rank of a wireless interface during the runtime. The ranking of all the wireless interfaces are computed based on the channel requirement and the data criticality of the buffered data within the hubs during each token round for efficient channel allocation. Moreover, the critical data is prioritized over the non-critical data during wireless data transmission for application speedup. Furthermore, Secure MAC is proposed that discusses possible threat models for the wireless NoC MAC and their countermeasures. The threat model shows how an HT in a wireless hub can enable the unauthorized access of the wireless channel and create *DoS* or *Spoofing* attack. The proposed countermeasure detects and localize the threat based on the computed and actual hold time of the wireless channel.

Some of the possible future research directions to extend our work are:

- In WiND framework, we have built the debug infrastructure for 2D Mesh topology and for deterministic XY routing. An obvious future extension is to validate the framework for different network topologies and routing algorithms considering the adaptive routing protocol. The framework can be made more efficient in terms of trace reduction by integrating suitable signal selection and trace compression techniques. There are future scope of improvements in the framework that can be worked upon in terms of establishing better path reconstruction strategies for larger snapshot intervals and with adaptive routing scenarios.
- Design for debug and design for security are conflicting in nature. The DFD modules gain access to several observable and controllable points. However, these hardware units can be exploited as vulnerable interfaces by the malicious third parties to leak secured information or degrade the system performance. In WiND, we have used wireless infrastructure for efficient debug capability. However, wireless communication is more vulnerable to eavesdrop and side-channel attacks. Moreover, in ReDeSIGN we propose to reuse the debug infrastructure for architectural purposes. This may provide the opportunities to an attacker to disturb the actual functionalities of the design through the debug modules acting as architectural components during the in-field operation. Hence, analyzing the threat models and developing security countermeasures for them is crucial and therefore can be considered as an important future work in this direction.
- In ReDeSIGN framework, we have demonstrated how almost all the debug modules are reused for architectural purposes to compensate for the debug hardware overhead. This considerably enhances the system performance, but reuses the debug modules for achieving gain during in-field execution that themselves are not validated for functional operations. Therefore, we have reused these debug modules mostly for functional extensions such as extended VCs and not as dedicated hardware for a

completely new functionality. This allows us to revoke the reuse capability of the debug modules if found any anomaly during in-field operation and continue the operation with the original hardware. This motivates a future research direction to come up with novel reverse solutions that can reuse the architectural components for all the debug operations to minimize the overhead of debug hardware.

- In ReDeSIGN framework, we have discussed the reuse of debug modules for functional purposes. There exists another scope to reuse design for debug (DFD) modules as design for test (DFT) modules or vice versa. DFT modules are inserted to the original design during design time to facilitate the test infrastructure during manufacturing tests. DFT performs a lot of scan flops insertion to the actual design to collect the design responses during the tests on silicon chips after the fabrication. This brings an opportunity to build another useful framework that can repurpose the scan flops as trace buffer or vice versa to reduce the debug and test hardware overhead.

Bibliography

[1] B. Bohnenstiehl, A. Stillmaker, J. J. Pimentel, T. Andreas, B. Liu, A. T. Tran, E. Adeagbo, and B. M. Baas, "Kilocore: A 32-nm 1000-processor computational array," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 4, pp. 891–902, 2017.

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

[3] H. G. Lee, N. Chang, U. Y. Ogras, and R. Marculescu, "On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 12, no. 3, pp. 1–20, 2008.

[4] S. Deb, A. Ganguly, P. P. Pande, B. Belzer, and D. Heo, "Wireless noc as interconnection backbone for multicore chips: Promises and challenges," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 2, no. 2, pp. 228–239, 2012.

[5] S. Deb and H. K. Mondal, "Wireless network-on-chip: a new era in multi-core chip design," in *2014 25nd IEEE International Symposium on Rapid System Prototyping*. IEEE, 2014, pp. 59–64.

[6] A. Shacham, K. Bergman, and L. P. Carloni, "On the design of a photonic network-on-chip," in *First International Symposium on Networks-on-Chip (NOCS'07)*. IEEE, 2007, pp. 53–64.

[7] P. Mishra, R. Morad, A. Ziv, and S. Ray, "Post-silicon validation in the soc era: A tutorial introduction," *IEEE Design & Test*, vol. 34, no. 3, pp. 68–92, 2017.

[8] P. Jayaraman and R. Parthasarathi, "A survey on post-silicon functional validation for multicore architectures," *ACM Computing Surveys (CSUR)*, vol. 50, no. 4, pp. 1–30, 2017.

[9] R. Abdel-Khalek and V. Bertacco, "Post-silicon platform for the functional diagnosis and debug of networks-on-chip," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 13, no. 3s, pp. 1–25, 2014.

[10] S. Mitra, S. A. Seshia, and N. Nicolici, "Post-silicon validation opportunities, challenges and recent advances," in *Design Automation Conference*. IEEE, 2010, pp. 12–17.

[11] M. Abramovici, "In-system silicon validation and debug," *IEEE Design & Test of Computers*, vol. 25, no. 3, pp. 216–223, 2008.

[12] D. Josephson, "The good, the bad, and the ugly of silicon debug," in *Proceedings of the 43rd annual Design Automation Conference*, 2006, pp. 3–6.

[13] H. F. Ko and N. Nicolici, "Automated trace signals identification and state restoration for improving observability in post-silicon validation," in *2008 Design, Automation and Test in Europe*. IEEE, 2008, pp. 1298–1303.

[14] X. Liu and Q. Xu, "On signal selection for visibility enhancement in trace-based post-silicon validation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 8, pp. 1263–1274, 2012.

[15] S. Ma, D. Pal, R. Jiang, S. Ray, and S. Vasudevan, "Can't see the forest for the trees: State restoration's limitations in post-silicon trace signal selection," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2015, pp. 1–8.

[16] A. Nahir, A. Ziv, M. Abramovici, A. Camilleri, R. Galivanche, B. Bentley, H. Foster, A. Hu, V. Bertacco, and S. Kapoor, "Bridging pre-silicon verification and post-silicon validation," in *Design Automation Conference*. IEEE, 2010, pp. 94–95.

[17] D. Lin, T. Hong, Y. Li, S. Eswaran, S. Kumar, F. Fallah, N. Hakim, D. S. Gardner, and S. Mitra, "Effective post-silicon validation of system-on-chips using quick error detection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 10, pp. 1573–1590, 2014.

[18] Y. Huang and P. Mishra, "Trace buffer attack on the aes cipher," *Journal of Hardware and Systems Security*, vol. 1, no. 1, pp. 68–84, 2017.

[19] E. Anis and N. Nicolici, "Low cost debug architecture using lossy compression for silicon debug," in *2007 Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2007, pp. 1–6.

[20] K. Basu and P. Mishra, "Test data compression using efficient bitmask and dictionary selection methods," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 18, no. 9, pp. 1277–1286, 2009.

[21] S. Deb, K. Chang, X. Yu, S. P. Sah, M. Cosic, A. Ganguly, P. P. Pande, B. Belzer, and D. Heo, "Design of an energy-efficient cmos-compatible noc architecture with millimeter-wave wireless interconnects," *IEEE Transactions on Computers*, vol. 62, no. 12, pp. 2382–2396, 2012.

[22] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *Proceedings of the 36th Annual International Symposium on Computer Architecture*, 2009, pp. 196–207.

[23] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani, "A network on chip architecture and design methodology," in *Proceedings IEEE Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design. ISVLSI 2002*. IEEE, 2002, pp. 117–124.

[24] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," 1988.

[25] K. Chang, S. Deb, A. Ganguly, X. Yu, S. P. Sah, P. P. Pande, B. Belzer, and D. Heo, "Performance evaluation and design trade-offs for wireless network-on-chip architectures," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 8, no. 3, pp. 1–25, 2012.

[26] M. Palesi, M. Collotta, A. Mineo, and V. Catania, "An efficient radio access control mechanism for wireless network-on-chip architectures," *Journal of Low Power Electronics and Applications*, vol. 5, no. 2, pp. 38–56, 2015.

[27] R. Das, O. Mutlu, T. Moscibroda, and C. R. Das, "Aergia: Exploiting packet latency slack in on-chip networks," *ACM SIGARCH computer architecture news*, vol. 38, no. 3, pp. 106–116, 2010.

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

[29] A. Ganguly, K. Chang, S. Deb, P. P. Pande, B. Belzer, and C. Teuscher, "Scalable hybrid wireless network-on-chip architectures for multicore systems," *IEEE Transactions on Computers*, vol. 60, no. 10, pp. 1485–1502, 2010.

[30] C. Wang, W.-H. Hu, and N. Bagherzadeh, "A wireless network-on-chip design for multicore platforms," in *2011 19th international euromicro conference on parallel, distributed and network-based processing*. IEEE, 2011, pp. 409–416.

[31] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[32] K. Goossens, B. Vermeulen, R. Van Steeden, and M. Bennebroek, "Transaction-based communication-centric debug," in *First International Symposium on Networks-on-Chip (NOCS'07)*. IEEE, 2007, pp. 95–106.

[33] N. Karimi, A. Alaghi, M. Sedghi, and Z. Navabi, "Online network-on-chip switch fault detection and diagnosis using functional switch faults." *J. Univers. Comput. Sci.*, vol. 14, no. 22, pp. 3716–3736, 2008.

[34] Q. Xu and X. Liu, "On signal tracing in post-silicon validation," in *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2010, pp. 262–267.

[35] H. F. Ko and N. Nicolici, "Algorithms for state restoration and trace-signal selection for data acquisition in silicon debug," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 2, pp. 285–297, 2009.

[36] K. Basu and P. Mishra, "Rats: Restoration-aware trace signal selection for post-silicon validation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 4, pp. 605–613, 2012.

[37] B. Vermeulen and K. Goossens, "A network-on-chip monitoring infrastructure for communication-centric debug of embedded multi-processor socs," in *2009 International Symposium on VLSI Design, Automation and Test*. IEEE, 2009, pp. 183–186.

[38] C. Ciordas, T. Basten, A. Radulescu, K. Goossens, and J. Meerbergen, "An event-based network-on-chip monitoring service," in *Proceedings. Ninth IEEE International High-Level Design Validation and Test Workshop (IEEE Cat. No. 04EX940)*. IEEE, 2004, pp. 149–154.

[39] S. Tang and Q. Xu, "A multi-core debug platform for noc-based systems," in *2007 Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2007, pp. 1–6.

[40] M. Dehbashi and G. Fey, "Transaction-based online debug for noc-based multiprocessor socs," *Microprocessors and Microsystems*, vol. 39, no. 3, pp. 157–166, 2015.

[41] H. F. Ko, A. B. Kinsman, and N. Nicolici, "Design-for-debug architecture for distributed embedded logic analysis," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 19, no. 8, pp. 1380–1393, 2010.

[42] K. Rahmani, S. Ray, and P. Mishra, "Postsilicon trace signal selection using machine learning techniques," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 2, pp. 570–580, 2016.

[43] E. Anis and N. Nicolici, "On using lossless compression of debug data in embedded logic analysis," in *2007 IEEE International Test Conference*. IEEE, 2007, pp. 1–10.

[44] A. B. Gomes, F. A. Alves, R. S. Ferreira, and J. A. M. Nacif, "Vericonn: a tool to generate efficient interconnection networks for post-silicon debug," in *2015 16th Latin-American Test Symposium (LATS)*. IEEE, 2015, pp. 1–6.

[45] X. Liu and Q. Xu, "Interconnection fabric design for tracing signals in post-silicon validation," in *2009 46th ACM/IEEE Design Automation Conference*. IEEE, 2009, pp. 352–357.

[46] K. Morris, "On-chip debugging—built-in logic analyzers on your fpga," 2004.

[47] S. S. Rout, S. Deb, and K. Basu, "Wind: An efficient post-silicon debug strategy for network on chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 11, pp. 2372–2385, 2020.

[48] S. S. Rout, K. Basu, and S. Deb, "Efficient post-silicon validation of network-on-chip using wireless links," in *2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID)*. IEEE, 2019, pp. 371–376.

[49] S. H. Gade, S. S. Ram, and S. Deb, "Millimeter wave wireless interconnects in deep submicron chips: Challenges and opportunities," *Integration*, vol. 64, pp. 127–136, 2019.

[50] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Cycle-accurate network on chip simulation with noxim," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 27, no. 1, pp. 1–25, 2016.

[51] S. S. Rout, S. B. Patil, V. I. Chaudhari, and S. Deb, "Efficient router architecture for trace reduction during noc post-silicon validation," in *2019 32nd IEEE International System-on-Chip Conference (SOCC)*. IEEE, 2019, pp. 230–235.

[52] C. Ciordas, K. Goossens, T. Basten, A. Radulescu, and A. Boon, "Transaction monitoring in networks on chip: The on-chip run-time perspective," in *2006 International Symposium on Industrial Embedded Systems*. IEEE, 2006, pp. 1–10.

[53] K. Basu and P. Mishra, "Efficient trace data compression using statically selected dictionary," in *29th VLSI Test Symposium*. IEEE, 2011, pp. 14–19.

[54] A. Martin, "Debugging with arm coresight," *LAUTERBACH NEWS*, 2009.

[55] W. Orme, "Debug and trace for multicore socs (arm white paper)," September 2008.

[56] X. Yu, S. P. Sah, H. Rashtian, S. Mirabbasi, P. P. Pande, and D. Heo, "A 1.2-pj/bit 16-gb/s 60-ghz ook transmitter in 65-nm cmos for wireless network-on-chip," *IEEE Transactions on Microwave Theory and Techniques*, vol. 62, no. 10, pp. 2357–2369, 2014.

[57] S. Kaushik, M. Agrawal, H. K. Mondal, S. H. Gade, and S. Deb, "Path loss-aware adaptive transmission power control scheme for energy-efficient wireless noc," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2017, pp. 132–135.

[58] S. H. Gade, S. S. Rout, M. Sinha, H. K. Mondal, W. Singh, and S. Deb, "A utilization aware robust channel access mechanism for wireless nocs," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2018, pp. 1–5.

[59] Chipscope pro and the serial i/o toolkit. [Online]. Available: https://www.xilinx.com/products/design-tools/chipscopepro.html

[60] armKEIL, CoreSight™ Technology. [Online]. Available: http://www2.keil.com/coresight/

[61] H. K. Mondal and S. Deb, "An energy efficient wireless network-on-chip using power-gated transceivers," in *2014 27th IEEE International System-on-Chip Conference (SOCC)*. IEEE, 2014, pp. 243–248.

[62] H. K. Mondal, G. N. S. Harsha, and S. Deb, "An efficient hardware implementation of dvfs in multi-core system with wireless network-on-chip," in *2014 IEEE Computer Society Annual Symposium on VLSI*. IEEE, 2014, pp. 184–189.

[63] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The splash-2 programs: Characterization and methodological considerations," *ACM SIGARCH computer architecture news*, vol. 23, no. 2, pp. 24–36, 1995.

[64] J. E. Miller, H. Kasture, G. Kurian, C. Gruenwald, N. Beckmann, C. Celio, J. Eastep, and A. Agarwal, "Graphite: A distributed parallel simulator for multicores," in *HPCA-16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*. IEEE, 2010, pp. 1–12.

[65] A. Ganguly, M. M. Ahmed, R. Singh Narde, A. Vashist, M. S. Shamim, N. Mansoor, T. Shinde, S. Subramaniam, S. Saxena, J. Venkataraman *et al.*, "The advances, challenges and future possibilities of millimeter-wave chip-to-chip interconnections for multi-chip systems," *Journal of Low Power Electronics and Applications*, vol. 8, no. 1, p. 5, 2018.

[66] N. Jindal, P. R. Panda, and S. R. Sarangi, "Reusing trace buffers as victim caches," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 9, pp. 1699–1712, 2018.

[67] C.-H. Lai, Y.-C. Yang, and J. Huang, "A versatile data cache for trace buffer support," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 11, pp. 3145–3154, 2014.

[68] A. DeOrio, I. Wagner, and V. Bertacco, "Dacota: Post-silicon validation of the memory subsystem in multi-core designs," in *2009 IEEE 15th International Symposium on High Performance Computer Architecture*. IEEE, 2009, pp. 405–416.

[69] N. Jindal, S. Chandran, P. R. Panda, S. Prasad, A. Mitra, K. Singhal, S. Gupta, and S. Tuli, "Dhoom: Reusing design-for-debug hardware for online monitoring," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2019, pp. 1–6.

[70] K. Basu, R. Elnaggar, K. Chakrabarty, and R. Karri, "Preempt: Preempting malware by examining embedded processor traces," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2019, pp. 1–6.

[71] S. S. Rout, M. Badri, M. Sinha, and S. Deb, "Redesign: Reuse of debug structures for improvement in performance gain of noc based mpsocs," *IEEE Transactions on Emerging Topics in Computing*, 2022.

[72] S. S. Rout, M. Badri, and S. Deb, "Reutilization of trace buffers for performance enhancement of noc based mpsocs," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2020, pp. 97–102.

[73] S. S. Rout, H. K. Mondai, R. Juneja, S. H. Gade, and S. Deb, "Dynamic noc platform for varied application needs," in *2018 19th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2018, pp. 232–237.

[74] C. A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, and C. R. Das, "Vichar: A dynamic virtual channel regulator for network-on-chip routers," in *2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)*. IEEE, 2006, pp. 333–346.

[75] J. Hu and R. Marculescu, "Application-specific buffer space allocation for networks-on-chip router design," in *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.* IEEE, 2004, pp. 354–361.

[76] G. P. Nychis, C. Fallin, T. Moscibroda, O. Mutlu, and S. Seshan, "On-chip networks from a networking perspective: Congestion and scalability in many-core interconnects," *ACM SIGCOMM computer communication review*, vol. 42, no. 4, pp. 407–418, 2012.

[77] M. Daneshtalab, M. Kamali, M. Ebrahimi, S. Mohammadi, A. Afzali-Kusha, and J. Plosila, "Adaptive input-output selection based on-chip router architecture," *Journal of Low Power Electronics*, vol. 8, no. 1, pp. 11–29, 2012.

[78] Y. Xu, B. Zhao, Y. Zhang, and J. Yang, "Simple virtual channel allocation for high-throughput and high-frequency on-chip routers," *ACM Transactions on Parallel Computing (TOPC)*, vol. 2, no. 1, pp. 1–23, 2015.

[79] S. Q. Zheng and M. Yang, "Algorithm-hardware codesign of fast parallel round-robin arbiters," *IEEE transactions on parallel and distributed systems*, vol. 18, no. 1, pp. 84–95, 2006.

[80] M. Oveis-Gharan and G. N. Khan, "Index-based round-robin arbiter for noc routers," in *2015 IEEE Computer Society Annual Symposium on VLSI*. IEEE, 2015, pp. 62–67.

[81] H. K. Mondal, S. H. Gade, S. Kaushik, and S. Deb, "Adaptive multi-voltage scaling with utilization prediction for energy-efficient wireless noc," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 4, pp. 382–395, 2017.

[82] S. J. Brams and A. D. Taylor, *Fair Division: From cake-cutting to dispute resolution*. Cambridge University Press, 1996.

[83] L. E. Dubins and E. H. Spanier, "How to cut a cake fairly," *The American Mathematical Monthly*, vol. 68, no. 1P1, pp. 1–17, 1961.

[84] S. Brânzei and P. B. Miltersen, "Equilibrium analysis in cake cutting," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 2013, pp. 327–334.

[85] J. Power, J. Hestness, M. S. Orr, M. D. Hill, and D. A. Wood, "gem5-gpu: A heterogeneous cpu-gpu simulator," *IEEE Computer Architecture Letters*, vol. 14, no. 1, pp. 34–36, 2014.

[86] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *2009 IEEE international symposium on workload characterization (IISWC)*. Ieee, 2009, pp. 44–54.

[87] J. Lee, S. Li, H. Kim, and S. Yalamanchili, "Adaptive virtual channel partitioning for network-on-chip in heterogeneous architectures," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 18, no. 4, pp. 1–28, 2013.

[88] W. Choi, K. Duraisamy, R. G. Kim, J. R. Doppa, P. P. Pande, D. Marculescu, and R. Marculescu, "On-chip communication network for efficient training of deep convolutional networks on heterogeneous manycore systems," *IEEE Transactions on Computers*, vol. 67, no. 5, pp. 672–686, 2017.

[89] Z. Li, J. Wu, L. Shang, R. P. Dick, and Y. Sun, "Latency criticality aware on-chip communication," in *2009 Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2009, pp. 1052–1057.

[90] S. H. Gade, S. S. Rout, R. Kashyap, and S. Deb, "Reliability analysis of on-chip wireless links for many core wnocs," in *2018 Conference on Design of Circuits and Integrated Systems (DCIS)*. IEEE, 2018, pp. 1–6.

[91] S. Abadal, A. Mestres, J. Torrellas, E. Alarcón, and A. Cabellos-Aparicio, "Medium access control in wireless network-on-chip: A context analysis," *IEEE Communications Magazine*, vol. 56, no. 6, pp. 172–178, 2018.

[92] D. Zhao, Y. Wang, J. Li, and T. Kikkawa, "Design of multi-channel wireless noc to improve on-chip communication capacity!" in *Proceedings of the Fifth ACM/IEEE International Symposium*. IEEE, 2011, pp. 177–184.

[93] A. Kumar, L.-S. Peh, and N. K. Jha, "Token flow control," in *2008 41st IEEE/ACM International Symposium on Microarchitecture*. IEEE, 2008, pp. 342–353.

[94] D. DiTomaso, A. Kodi, S. Kaya, and D. Matolak, "iwise: Inter-router wireless scalable express channels for network-on-chips (nocs) architecture," in *2011 IEEE 19th annual symposium on high performance interconnects*. IEEE, 2011, pp. 11–18.

[95] N. Mansoor, M. P. Yuvaraj, and A. Ganguly, "A robust medium access mechanism for millimeter-wave wireless network-on-chip architecture," in *2013 IEEE International SOC Conference*. IEEE, 2013, pp. 362–367.

[96] A. K. Mishra, N. Vijaykrishnan, and C. R. Das, "A case for heterogeneous on-chip interconnects for cmps," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 3, pp. 389–400, 2011.

[97] S. H. Gade, H. K. Mondal, and S. Deb, "High bandwidth off-chip memory access through hybrid switching and inter-chip wireless links," in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2018, pp. 100–105.

[98] Y. Ouyang, C. Sun, B. Jia, Q. Wang, and H. Liang, "Architecting a priority-based dynamic media access control mechanism in wireless network-on-chip," *Microelectronics Journal*, vol. 116, p. 105218, 2021.

[99] D. DiTomaso, A. Kodi, D. Matolak, S. Kaya, S. Laha, and W. Rayess, "A-winoc: Adaptive wireless network-on-chip architecture for chip multiprocessors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, pp. 3289–3302, 2014.

[100] N. Mansoor, A. Vashist, M. M. Ahmed, M. S. Shamim, S. A. Mamun, and A. Ganguly, "A traffic-aware medium access control mechanism for energy-efficient wireless network-on-chip architectures," *arXiv preprint arXiv:1809.07862*, 2018.

[101] Q. Gao, W. Song, Z. Lu, L. Li, and Y. Fu, "Dynamic and traffic-aware medium access control mechanisms for wireless noc architectures," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2021, pp. 1–5.

[102] S. S. Rout, M. Sinha, and S. Deb, "2dmac: A sustainable and efficient medium access control mechanism for future wireless nocs," *ACM Journal on Emerging Technologies in Computing Systems*, 2022.

[103] S. S. Rout, V. I. Chaudhari, S. B. Patil, and S. Deb, "Rcas: critical load based ranking for efficient channel allocation in wireless noc," in *2019 32nd IEEE International System-on-Chip Conference (SOCC)*. IEEE, 2019, pp. 21–26.

[104] S. S. Rout, A. Singh, S. B. Patil, M. Sinha, and S. Deb, "Security threats in channel access mechanism of wireless noc and efficient countermeasures," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2020, pp. 1–5.

[105] F. Rad, M. Reshadi, and A. Khademzadeh, "A survey and taxonomy of congestion control mechanisms in wireless network on chip," *Journal of Systems Architecture*, vol. 108, p. 101807, 2020.

[106] N. Mansoor and A. Ganguly, "Reconfigurable wireless network-on-chip with a dynamic medium access mechanism," in *Proceedings of the 9th International Symposium on Networks-on-Chip*, 2015, pp. 1–8.

[107] S. Jog, Z. Liu, A. Franques, V. Fernando, S. Abadal, J. Torrellas, and H. Hassanieh, "One protocol to rule them all: Wireless {Network-on-Chip} using deep reinforcement learning," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, 2021, pp. 973–989.

[108] A. Franques, S. Abadal, H. Hassanieh, and J. Torrellas, "Fuzzy-token: An adaptive mac protocol for wireless-enabled manycores," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 1657–1662.

[109] E. Bolotin, Z. Guz, I. Cidon, R. Ginosar, and A. Kolodny, "The power of priority: Noc based distributed cache coherency," in *First International Symposium on Networks-on-Chip (NOCS'07)*. IEEE, 2007, pp. 117–126.

[110] A. Ganguly, M. Y. Ahmed, and A. Vidapalapati, "A denial-of-service resilient wireless noc architecture," in *Proceedings of the great lakes symposium on VLSI*, 2012, pp. 259–262.

[111] F. Pereñíguez-García and J. L. Abellán, "Secure communications in wireless network-on-chips," in *Proceedings of the 2nd International Workshop on Advanced Interconnect Solutions and Technologies for Emerging Computing Systems*, 2017, pp. 27–32.

[112] A. Vashist, A. Keats, S. M. P. Dinakarrao, and A. Ganguly, "Securing a wireless network-on-chip against jamming-based denial-of-service and eavesdropping attacks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 12, pp. 2781–2791, 2019.

[113] B. Lebiednik, S. Abadal, H. Kwon, and T. Krishna, "Spoofing prevention via rf power profiling in wireless network-on-chip," in *Proceedings of the 3rd International Workshop on Advanced Interconnect Solutions and Technologies for Emerging Computing Systems*, 2018, pp. 1–4.

[114] ——, "Architecting a secure wireless network-on-chip," in *2018 Twelfth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*. IEEE, 2018, pp. 1–8.

[115] B. M. Rogers, A. Krishna, G. B. Bell, K. Vu, X. Jiang, and Y. Solihin, "Scaling the bandwidth wall: challenges in and avenues for cmp scaling," in *Proceedings of the 36th annual international symposium on Computer architecture*, 2009, pp. 371–382.

[116] S. H. Gade, S. S. Rout, and S. Deb, "On-chip wireless channel propagation: Impact of antenna directionality and placement on channel performance," in *2018 Twelfth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*. IEEE, 2018, pp. 1–8.

[117] M. Shoba and R. Nakkeeran, "Energy and area efficient hierarchy multiplier architecture based on vedic mathematics and gdi logic," *Engineering Science and Technology, an International Journal*, vol. 20, no. 1, pp. 321–331, 2017.

[118] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Improving energy efficiency in wireless network-on-chip architectures," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 14, no. 1, pp. 1–24, 2017.

[119] H. K. Mondal, S. H. Gade, R. Kishore, and S. Deb, "Adaptive multi-voltage scaling in wireless noc for high performance low power applications," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 1315–1320.

[120] R. S. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware trojan: Threats and emerging solutions," in *2009 IEEE International high level design validation and test workshop*. IEEE, 2009, pp. 166–171.