2016년 2월
박사학위 논문

An Autonomously Self-Aware and Adaptively Reconfigurable System for FPGA-Based Network-on-Chips

조선대학교 대학원

컴 퓨 터 공 학 과

Abba Sani

An Autonomously Self-Aware and Adaptively Reconfigurable System for FPGA-Based Network-on-Chips

2016년 2월 25일

조선대학교 대학원

컴 퓨 터 공 학 과

Abba Sani

An Autonomously Self-Aware and Adaptively Reconfigurable System for FPGA-Based Network-on-Chips

지도교수 이 정 아

이 논문을 컴퓨터공학 박사학위신청 논문으로 제출함

2015 년 10 월

조선대학교 대학원

컴 퓨 터 공 학 과

Abba Sani

Abba Sani의 박사학위논문을 인준함

위원장 조선대학교 교수 조 범 준 (인)

위 원 조선대학교 교수 김 판 구 (인)

위 원 조선대학교 교수 모 상 만 (인)

위 원 전남대학교 교수 김 철 홍 (인)

위 원 조선대학교 교수 이 정 아 (인)

2015년 12월

조선대학교 대학원

Table of Contents

List of Tables

List of Figures

List of Abbreviations

A	Total Number of Network Packets.
$\overline{L_m}(\boldsymbol{\tau})$	Autuator Input.
$\overline{K}_n(\boldsymbol{\tau})$	On-Chip Controller Input Signal.
2D	Two Dimensions.
ADC	Analog to Digital Converter.
AREA	Synthesized PFGA Area used as logic and routing
ARM	Advanced RISC Machine.
ASAART	Autonomous Self-Aware and adaptive Routing Technique.
Auto-Self-Awareness-Back-off$_{Delay}$	Auto Self-Awareness Back-off Delay Strategy.
Auto-Self-Awareness-Back-off$_{Delay-Slotted}$	Auto Self-Awareness Back-off Delay Slotted Strategy.
Bdca	Determines the Period in which Data are transmitted by the Cluster head to the Cluster Members.
BFT	Butterfly Fat-tree.
CAN	Controller Area Network.
CLK_FREQ, CLKF	Synthesized FPGA Clock Frequency.
CPU	Central Processing Unit.
DAG	Directed Acyclic Graph.
DAMQ	Dynamically Allocatable Multiple Queue.
d_o - d_3	Input Pins.
E	Set of Edges.
FBD	Flit Buffer Depth.

FDW	Flit Data Width.
FIFO	First in First Out.
FREE	Real-time Free Slot.
Global-Awareness-Back-off$_{Delay}$	Global Awarness Back-off Delay Strategy.
HDL	Hardware Description Language.
HoL	Head on Line.
HopCount$_{Dest}$	Number of Hop Counts from Current Node to the Destination.
i	Index Variable.
IEEE	Institute of Electrical and Electronic Engineering.
INST	Defines the Time Interval between the Transmission and Reception of the Sensor Data.
IP	Intellectual Property.
K	Network Packet.
K, t	Network Nodes.
$K_n(\tau)$	On-Chip Sensor Signal.
$L_m(\tau)$	Monitored Controller Output.
Local-Awareness-Back-off$_{Delay}$	Local Awareness Back-off Delay Strategy.
LUT	Look Up Table.
MHz	Megahertz.
MoT	Mesh of Tree.
MUX	Multiplexer Interface.
N, I	Index Variables.
NCAP	Network Capable Application Processor.
NI	Network Interface.
NoC	Network-on-Chip.

$NumHopCount_{expected}$	Expected Number of Hops Contained in the Destination Reference Packet Header.
P	Probability of an Event.
Pa	Probability of Parent of Random Variable (X_i).
$Packet_{id}$	Packet Identification Number.
PLUT	Synthesized PFGA Logic Area.
R, K_{ref}	Disturbance Signals or Noise.
RAM	Random Access Memory.
RAND	Random Function Generator.
$Route_{Metric}$	Route Computation Metric.
$Route_{MetricSeqn}$	Route Computation Metric Sequence Number.
RRP	Route Repair Packet.
Rxd	Receiver.
$Scale_{Factor}$	Parameter to Stretch the Random Function.
SHR	Self- Healing Routing.
Slot-Size	Slot Window Size.
$Source_{id}$	Source Node Identification Number.
SSR	Self-Selective Routing.
SYN	Defines the Real-time Synchronization Period between the Cluster Head (subcontroller) and Cluster Members.
T	Subset of Graphs from the Set V.
TI	Transducer Interface.
T_j, L_j	J^{th} Bit of the Source, Destination Address.
T_n, L_n	The n^{th} Radix-Q digit of the Source, Destination. Address.
Tx	Transmitter.
V	Set of Nodes.

VC	Virtual Channel.
Vcc	Source Voltage.
VHDL	Very High-Level Design Language.
VoQ	Virtual Output Ports.
WSN	Wireless Sensor Network.
X	Set of Random Variables.
β	Operational Mode.
B_1,..., β_N	Real-time Interval which the Cluster members tansmit data to the cluster head.
τ	Number of Switch Ports.

초 록

FPGA 기반의 네트워크 온 칩을 위한 자가인지 적응형 재구성시스템.

아바 사니

지도교수 : 이정아

컴퓨터공학과

조선대학교 대학원

본 논문에서는 FPGA 기반의 네트워크 온 칩의 설계구조 탐사, 성능 평가, 안정적인 작동 관련 인프라 프레임 워크를 제공하는 설계 패러다임으로 자율적인 자가 인식과 상황에 따라 적응적으로 동적 재구성 가능한 시스템을 제안하고, 온 칩 연결 및 센서 네트워크의 결함이 있는 경우에 런타임 라우팅을 보장하는 상세한 메커니즘과 알고리즘을 제시한다. 이러한 설계 패러다임은 확장적이며, 안정적인 동적 적응과 런타임 매개 변수의 제어가 가능함을 보인다.

본 논문에서는 먼저, FPGA 기반의 네트워크 온 칩의 확장성 및 적응형 맥락에서 매개 변수가 성능에 미치는 영향을 평가하기 위한 방법론을 제시한다. 온 칩 연결의 일반적인 구조, Torus, Mesh와 Fat-tree 세 가지를 고려하여, 네트워크 매개변수에 따른 FPGA의 성능을 평가하고 관련된 설계 상충관계를 분석한다. 또한, 네트워크 온 칩 구조가 상호 연결 및 라우팅 파라미터에 의해 어떻게 영향을 받는지, FPGA 합성 및 다양한 구성 테스트를 통해 유연성 및 성능에 관련된 분석을 제

시한다.

FPGA 기반의 네트워크의 성능 모델과 자가적응형 모델을 통하여, 자가 인식의 자율적인 적용과 적응적으로 재구성 가능한 시스템 패러다임을 보인다. 센서 데이터를 라우팅하기위한 기존의 자가치유 및 셀프 선택적 라우팅 기술의 한계를 보이고, 경로의 형성 오류와 실패에 탄력성을 제공하는 경로 수리를 위해 자율적인 자가인식 및 적응형 장애 감지 및 복구 기술을 통합한, 자율적인 자가인식 라우팅 및 적응형 기법 (ASAART) 프로토콜을 제안한다. 제안된 프로토콜은 네트워크의 순간 토폴로지 변경에도 효율적으로 적응한다. 연속적으로 우선 순위화된 슬롯을 이용한 백 오프 지연전송을 활용하여 로컬 및 글로벌 네트워크 상태 정보를 얻고, 다수의 랜덤 함수를 이용하여, 영구적 및 임시적 노드 실패에도 안정적인 경로 복구의 빠른 수렴을 달성한다.

자율적 자가인식과 적응형 재구성 시스템 패러다임의 개념적 검증을 위해, 본 논문에서 실시간 모니터링 및 FPGA 온칩 센서 네트워크의 효율적인 제어를 위한 다른 낮은 수준의 마이크로 구조 설계 및 프레임 워크가 제시된다. 주된 목표는 자율적 센서 에이전트를 사용하여 저전력, 저비용 고정밀 모니터링 및 제어 메커니즘을 설계하고, 온 칩 센서 네트워크의 제어를 동적으로 재구성하는 것이다. 전압과 온도와 같은 파라미터를 실시간 모니터링하고 수집하며, 시스템의 자가인식을 통하여 FPGA 로직 리소스 및 온 칩 센서 구성 모든 요소의 소비전력 이용을 효율적으로 향상시킬 수 있다. 하나의 사례로, 온칩 FPGA 센서를 동적으로 모니터링하고 제어하며, 사용자 대화식 웹 애플리케이션을 사용하여 보고하는 설계 사례를 구현한다. 실험 결과를 통하여 기존 방식에 비해 FPGA 로직 리소스 및 모든 온 칩 센서 부품의 전력 소비가 효율적임을 보인다. FPGA 온칩 센서 전압 및 온도계측 결

과는 높은 정밀도와 정확도를 보였고, 1000 ms에서 동적 재구성 리프레시 시간을 설정하면 100, 500 ms에 비하여 계측값의 정확도가 높아짐을 관찰했다.

본 논문에서 제안된 설계 기법과 프레임 워크는 복잡한 온칩 센서 네트워크 및 원격 감지 애플리케이션의 유연하고 효율적인 실시간 모니터링 및 제어를 가능하게 하여, 네트워크 엔지니어 및 시스템 설계자에게 도움이 될 것으로 기대된다.

Abstract

An Autonomously Self-Aware and Adaptively Reconfigurable System for FPGA-Based Network-on-Chips

Abba, Sani

Advisor: Prof. Lee, Jeong-A, Ph.D.

Department of Computer Engineering,

Graduate School of Chosun University

This dissertation proposes an autonomously self-aware and adaptively reconfigurable system for FPGA-Based network-on-chips design paradigm for providing an infrastructural framework for the exploration, performance evaluation, correctness and reliability issues of network-on-chips fabrics in the Field Programmable Gate Arrays (FPGAs). It presents a design and implementation of architectures that enable for self-adaptation, dynamic reconfiguration, autonomic formation and self-awareness of network-on-chip architectures. It considers the design of fault tolerant routing scheme and algorithms for both on-chip interconnect and sensor networks that provide a low level mechanism for ensuring runtime autonomous fault-tolerant routing in the presence of faulty network components. Furthermore, it presents a system level adaptation by runtime monitoring of environmental parameters and employing evolutionary optimization methods to intelligently adapt and optimize its operation. This paradigm, provides scalable, reliable and dynamic adaptation and control of runtime parameters for on-chip networks.

The study motivates the paradigm shift towards FPGA-Based networks-on-chip by examining the current and future technological trends in FPGA

architecture and embedded hardware system in order to address the limitations of the existing techniques for the on-chip networks. The dissertation introduces a methodology to evaluate the performance impacts of the NoC parameters in the context of scalable and adaptive FPGA-Based network-on-chips. It proposes a consistent parametric method and design trade-offs for evaluating the FPGA performance of the three common on-chip interconnect architecture namely, the Torus, Mesh and Fattree architectures. It also investigates how the NoC architectures are affected by the interconnect and routing parameters and demonstrate the flexibility and performance through FPGA synthesis and testing of different NoC configurations. The dissertation overview the design methodology for designing NoCs in the context of FPGAs and demonstrate the hardware and software procedures for the design and implementation of FPGA-Based NoCs. It then proposes two models namely, FPGA-Based network performance and self-adaptive models.

In order to demonstrate the flexibility and applicability of the autonomously self-awareness and adaptively reconfigurable system paradigm, the study addresses the limitations of Self-healing and Self-selective routing technique for routing sensor data and proposed the design of an autonomously self-aware and adaptively routing technique (ASAART) protocol. It examines the integration of autonomic self-awareness and adaptive fault detection and resiliency technique for route formation and route repair to provide resilience to errors and failures. This is achieved by using a combined continuous and slotted prioritized transmission back-off delay to obtain local and global network state information as well as multiple random functions for attaining faster routing convergence and reliable route repair despite transient and permanent node failure rates and efficient adaptation to instantaneous network topology changes.

To validate the autonomously self-aware and adaptively reconfigurable system paradigm as a proof of concept, the dissertation also proposed different low-level

micro-architectural design and framework for real-time monitoring and efficient control of on-chip sensor network using FPGAs. The main goals are to design a low-power, low-cost and highly accurate monitoring and control mechanism using autonomous sensor agents and to dynamically reconfigure control of on-chip sensor networks using FPGAs. By collecting dynamic and real-time monitoring parameters such as voltage and temperature the system becomes self-aware and adaptive to improve the utilization of FPGA logic resources and efficient power consumption of all on-chip sensor components. It then proposed a case study to dynamically monitor and control the on-chip FPGA sensors and reports the sensed parameters using an interactive user application webpage. Experimental results show a significant low usage of FPGA logic resources and efficient power consumption of all on-chip sensor components compared with previous approaches. Finally, results from the FPGA-measured on-chip sensor readings show high precision and accuracy in the measured voltage and temperature. We found that setting the dynamic reconfiguration refresh time at 1000 ms produces highly accurate FPGA-measured on-chip sensor readings compared with those at 100 and 500 ms. The proposed design technique and framework presented in this dissertation will assist network engineers and system designers by providing flexible and efficient real-time monitoring and control design of large and complex on-chip sensor networks and remote sensing applications.

I. Introduction

A. Overview

This chapter was written based on the published research papers [173], [174] and [175]. In embedded hardware systems the complexity of designing efficient, scalable, on-chip communication mechanisms is expected to increase as further technical advances allow more processors to be incorporated onto a single die, and as power dissipation and wire delay become more significant among design constraints [1-2]. Network-on-Chip (NoC) architectures help to mitigate this complexity. Their modular construction facilitates both design flexibility and high-performance, while their structured wiring allows better use of abundant wire resources and more control over electrical characteristics [2].

A NoC interconnect determines the physical layout and connections between the nodes and channels of an on-chip network used as a Field-Programmable Gate Array (FPGA). This, in turn, determines the number of nodes (or routers) a message must traverse and the interconnect lengths between these nodes, significantly affecting network latency and energy consumption [3]. Because the interconnect also defines the total number of alternate paths between nodes, it largely determines how well the network can spread traffic to support bandwidth requirements [4]. The operational complexity and cost of an NoC interconnect depends on two factors: (1) the number of channels at each node (*node degree*)

and (2) ease of layout (i.e., wire lengths and the number of metal layers required) [3-4].

This dissertation describes a new paradigm shift for FPGA-Based network-on-chip, namely autonomous self-aware and adaptive reconfigurable systems and proposes a parametric-based technique for evaluating the performance of packet-switched NoC interconnection architectures. The proposed approach incorporates design space attributes for topology, network, and router parameters, including the number of Virtual Channels (VCs), Flit Buffer Depth (FDB), Flit Data Width (FDW), pipeline router core, pipeline router allocator, and pipeline links. This approach allows customization of the network to make better application-specific trade-offs between resource usage and performance. The dissertation demonstrates that by strategically tuning multiple parameters of both the interconnect and routers, the system performance can be increased without adversely affecting the area or speed of the FPGA network. The dissertation also showed that the FDW and FBD parameters account for much more consumption of FPGA resources than the VCs parameter, and that, with proper tuning, Fat-tree networks offered much better performance in terms of silicon area, clock frequency, critical path delay, network saturation throughput, and latency under both benign and adversarial traffic patterns than the Mesh and Torus networks. The findings in this dissertation will help engineers and designers select the right interconnects and router parameters for large and complex NoCs, and that the parametric method described in this dissertation offers a better means of evaluating tiled on-chip interconnection architectures.

Autonomous self-aware and adaptive systems provide the means to solve the computational complexities of a dynamic environment. Autonomous self-aware

routing algorithms possess the capability of configuring, healing, optimizing, and protecting themselves autonomously [5-24]. This also entails the ability to find the best means for optimizing performance in resource-constrained environments. Sensor networks are exposed to inhospitable environments and need to be adapted to changes in the environmental parameters or users. Sensor networks that are resource constrained have low power, low communication, and minimal computational capabilities. In addition, they possess low storage capabilities, low cost, reliability and fault tolerance, which are the critical sensor network design considerations [5-8].

Sensor nodes may fail for various reasons. These include radio interference, battery failure, and synchronous and asynchronous communication mechanisms. These failures mainly arise as a result of software or hardware faults, malicious code, environmental changes, and timing closure. On the whole, the result of such an event is that a sensor node becomes inaccessible or disrupts certain operating conditions that are critical to providing the required service. Communication in sensor networks is highly critical because the links in a network are prone to faults and errors. For instance, the collision of network packets may result in a failure. The presence of such a failure in a routing protocol will lead to considerable network packet drops, resulting in excessive network delays and the consumption of a substantial amount of energy [25-55]. The traditional fault-tolerant approach to routing in a wireless sensor network introduces more

redundant transmission and retransmission of network packets and consumes an excessive amount of energy. In addition, they are also deficient in terms of their autonomous self-awareness and adaptive capabilities in routing packets from their sources to their destinations in the presence of transient and permanent node failures and in highly scalable and congested networks. The essential concerns here are how to avoid the problem of packet flooding because of broadcasting and network scalability issues. To address these issues, the dissertation attempts to reduce the number of redundant network packets caused by flooding. In addition, the dissertation focuses on the minimization of control packet overheads and efficient end-to-end data delivery. In doing so, we combined both the continuous and slotted prioritized transmission back-off delays to obtain local and global network state information and a multiple randomize function for faster routing convergence and efficient and reliable route repair in the presence of transient and permanent node failures. This approach will provide resiliency to errors and failures, minimize the end-to-end network delay, and improve the overall network routing performance and energy efficiency.

The complexity imposed by on-chip sensor network monitoring and control increases with the scalability of the on-chip sensor network. Therefore, there is the need to have an intelligent and reliable low-level design of intelligent monitoring and control systems using autonomous sensor agents. Because different runtime physical parameters must be monitored, the need for accurate

and efficient sensor communication mechanisms is very essential to ensure reliable monitoring and control of complex on-chip sensor network environment. This process also entails the design of low-power and high-speed circuits for efficient and reliable network monitoring and control. The use of field programmable gate arrays (FPGAs) for autonomous sensor network monitoring and control is an interesting research domain in which low power consumption and accurate sensor readings are the major metrics for evaluation [56-64], [65-68].

As the flexibility and reliability of FPGAs increases, autonomous sensor agents can be embedded in an FPGA to measure different runtime parameters. However, designing a sensor network on an FPGA is a very difficult task to achieve in a short time because designers are limited to do beyond the FPGA manufacturer design specification. Similarly, the degree of complexity and sophistication of design automation contained in the FPGA makes it difficult for designers to tweak the FPGA circuitry to design and implement sensor networks. Hence, the use of FPGA to implement sensor networks will be an efficient solution for a reliable on-chip sensor network packet transmission and retransmission scheme because logic and critical-path delays in FPGAs are minimal [61-63]. The dissertation proposes a low-level micro-architectural design infrastructure for real-time monitoring and control of on-chip sensor network-based systems using FPGA. The dissertation, then develops the design and implementation of efficient and reliable high-speed circuit technique of the on-chip sensor network runtime parameter

monitoring and control, which uses autonomous sensors that reside at the network interface (NI), to be dynamically configured and to communicate the runtime ambient parameters to the network monitor controller hardware. A detailed low-level design methodology and procedure and a user application for efficient and reliable on-chip sensor network monitoring and control using FPGAs was fully presented. In addition, a webpage interface for the user to dynamically reconfigure the FPGA to conduct on-chip sensor readings and adjust the sensor ranges and reconfiguration refresh time was provided.

B. Motivation

This dissertation considers the current and future trend in FPGA-Based NoC design and was motivated by the fact that the NoC interconnect architecture imposes new and complicated design issues in embedded hardware system. Engineers must decide what topology is suitable for their applications, they must design, network interfaces to access the on-chip network and routers, they must choose appropriate communication protocols (including routing, switching, buffer management, flow control, etc.), and they must find the right parameters for all of the above. To address these design issues, the dissertation tries to answer questions about the factors that affect the choice of an FPGA-Based NoC. There are several factors that affect the choice of NoC topology, including [4], [69]:

1. *Performance Requirements*: These requirements are typically characterized in terms of packet latency and throughput.

2. *Scalability*: A scalable architecture assumes that as more IPs are added, input and output bandwidth, and network bandwidth, will steadily increase.
3. *Simplicity*: Simpler designs typically yield higher clock frequencies and better performance.
4. *Distance Span*: Different topologies may have very different link lengths, in some cases producing problems with coupling, electromagnetic noise, and wire heaviness.
5. *Physical Constraints*: Packing components (e.g., processors, memory, i/o devices, etc.) into a network imposes certain physical constraints, such as limitations on space, wire length, and operating temperature.
6. *Reliability and Fault Tolerance*: A network should be able to deliver information reliably, providing continuous operation with a limited number of faults.

With regards to fault tolerance, the dissertation was motivated mainly by the observation that the traditional fault-tolerant approach to routing for both on-chip interconnect and wireless sensor network introduces more redundant transmission and retransmission of network packets and consumes an excessive amount of energy. In addition, they are also deficient in terms of their autonomous self-awareness and adaptive capabilities in routing packets from their sources to their destinations in the presence of transient and permanent node failures and in highly scalable and congested networks. The essential concerns here are how to avoid the problem of packet flooding because of broadcasting and network scalability issues. To address these concerns, the dissertation attempts to reduce the number of redundant network packets caused by flooding. The dissertation focuses on the minimization of these control packet overheads and efficient end-

to-end data delivery. In doing so, the proposed approach combined both the continuous and slotted prioritized transmission back-off delays to obtain local and global network state information and a multiple randomize function for faster routing convergence and efficient and reliable route repair in the presence of transient and permanent node failures. In the proposed scheme, the sensor node that transmits the packets contains information pertaining to its neighbors in the control header packets. The sender node is self-aware and adaptive if it knows both local and global state information contained in the control packets. In this case, the receiver node knows whether its neighbors have been covered based on the global information obtained in the neighbors' control packet list information. The sender node with the auto-self-awareness back-off information now has a higher priority to forward the network packet to the destination. This approach will provide resiliency to errors and failures, minimize the end-to-end network delay, and improve the overall network routing performance and energy efficiency.

Another motivation of this dissertation lies on the fact that as the flexibility and reliability of FPGAs and complexity and scalability of on-chip sensor network continue increases, autonomous sensor agents can be embedded in an FPGA to measure different runtime parameters. Therefore, a reliable and efficient low-level design that not only detects runtime ambient parameters, but also offers better network performance, scalability, and flexibility of on-chip sensor network design for FPGAs is required. Hence, the use of FPGA for sensor network monitoring and control is very important [61-63].

C. Dissertation Objectives

In embedded hardware systems, an autonomous self-aware and adaptive reconfigurable system for FPGA-Based network-on-chip is a promising paradigm shift aiming at providing an infrastructural framework for the exploration, performance evaluation, correctness and reliability issues of network-on-chips fabrics in the Field Programmable Gate Array (FPGA). The dissertation aimed to achieve the following objectives:

1. To propose, design and implement networks-on-chip architecture that provide flexible runtime, and autonomic formation for self-aware and adaptive reconfigurable networks-on-chip using FPGA. To develop methodologies and techniques for evaluating the performance and correctness of the NoC in FPGA that provides high throughput and clock frequency, low latency and efficient silicon area consumption.
2. To propose, design and implement fault tolerant routing schemes and algorithms to ensure low level self-aware and adaptive routing decision in the presence of faulty network components. To minimize network flooding and enhance throughput and energy efficiency.
3. To develop a mechanism for self-adaptation of NoC in complex and dynamic environments by runtime observability and monitoring of environmental parameters and employing evolutionary optimization techniques to intelligently adapt and optimize its operation and enhance system performance.
4. To develop working prototypes suitable to be employed in real system development. To employ the use of sensors to facilitate intelligent network monitoring and control of on-chip networks and other related applications.

5. To propose a low-level micro-architectural design infrastructure for real-time monitoring and control of on-chip sensor network-based systems using FPGAs. To design and implement efficient and reliable high-speed circuit technique for on-chip sensor network runtime parameter monitoring and control. To achieve low power, low-cost and highly accurate measured on-chip sensors reading accuracy.

In order to achieve the stated objectives, the dissertation through its new design paradigm shift, namely, autonomously self-aware and adaptively reconfigurable system, proposes the design and implementation of efficient and reliable system level architectures, methodologies, algorithms and prototypes for enhancing overall system performance in terms of throughput, latency, clock frequency, silicon area, critical path delay, and energy consumption.

D. Contributions

This dissertation contributed towards a state-of-the-art embedded hardware system. In order to achieve the stated objectives, the dissertation develops an autonomously self-aware and adaptively reconfigurable system for FPGA-Based network-on-chips design paradigm. The salient contributions of this dissertation are as follows:

1. A parametric-based methodology to perform a parametric performance evaluation and analysis of design trade-offs for the Mesh, Torus and Fat-tree network interconnect architectures using FPGAs was developed. In this process, the dissertation considers a number of salient NoC and router parameters, and the presented results should provide immediate value to

engineers and architects tasked with finding the right parameters for large and complex NoCs.

2. The dissertation achieved extensive design space coverage by synthesizing 392 different NoC configurations for the Mesh, Torus and Fat-tree networks using Xilinx ISE 14.3 design tools and a Virtex 7 FPGA device. In addition, the dissertation examined the impact of three basic NoC parameters using FPGA. Experimental results show that Fat-tree network produced the best utilization of the FPGA logic resources.
3. The dissertation demonstrated that for large and complex NoCs, an FDW of size 8 to 64 bits, VCs of size 2 to 4 and FBD of size 8 to 16 bits produced efficient FPGA area and clock frequency.
4. The dissertation investigated the network performance of the three NoC architectures under benign and adversarial traffic patterns. Simulation results showed that Fat-tree networks performed fairly well in terms of throughput and latency for both benign and adversarial traffic patterns, while the Mesh and Torus networks performed rather poorly for adversarial traffic patterns.
5. The dissertation demonstrated that the propose methodology substantially improves performance in a variety of experimental and simulated scenarios, suggesting its utility and applicability to real-world systems.
6. The dissertation develops an autonomous self-aware and adaptive fault tolerant routing technique (ASAART) as proof of concept for efficient and reliable route formation and faster routing decision-making for wireless sensor networks. This is achieved by combining both continuous and prioritized slotted back-off delay and multiple randomize function techniques for speedy routing convergence to obtain local and global network state information for the efficient and reliable routing of sensor data.

7. The dissertation develops a route repair technique using the greedy algorithmic approach for reliable transmission of sensor data in the presence of permanent and transient node failure rates, and efficient adaptation to simultaneous network topology changes.
8. An extensive simulation to demonstrate the routing performance, flexibility and efficiency of the autonomous self-aware and adaptive fault tolerant routing technique (ASAART) under five different simulated scenarios was conducted. The simulation results show that the ASAART performs better in terms of high resiliency against errors and failure, and has better routing performance and energy efficiency compared with SSR and SHR protocols in the presence of transient and permanent node failure rates and in highly congested, faulty, and scalable sensor networks.
9. As a proof of concept for the autonomous self-aware and adaptive reconfigurable systems paradigm, the dissertation proposes a low-level micro-architectural design and infrastructure for real-time intelligent network monitoring and control for reconfigurable on-chip sensor network using FPGA. The intelligent sensor agents collect runtime ambient parameters such as voltage and temperature for efficient monitoring and control of on-chip sensor networks.
10. The dissertation proposes a real-time triggered protocol for efficient communication between the sensors and the monitoring and control unit.
11. A detailed low-level design procedure and implementation of the on-chip sensor network, sensor NI, and monitoring and control unit using both Synopsys and Xilinx design tools was provided. The proposed design and framework was implemented as a case study using Xilinx Vivado 2013.3 integrated design environment and a Zynq 7000 FPGA device. The post synthesis and implementation results, including the device resource utilization, showed low FPGA logic resource consumption and efficient power utilization

by the on-chip sensor components compared with previous approaches.

12. The dissertation provided extensive experimental results using an FPGA to measure and controls the on-chip sensors. The experimental results from the FPGA-measured and controlled on-chip sensor readings show high precision and accuracy in the voltage and temperature readings. The dissertation demonstrates that setting the dynamic reconfiguration refresh time at 1000 ms produced accurate FPGA-measured on-chip sensor readings compared with those at 100 and 500 ms. This approach will help network designers and engineers design efficient and reliable intelligent monitoring and control system for on-chip sensors using FPGA and other related remote-sensing applications.

E. Dissertation Outline

The dissertation consists of six chapters, following the introduction that gives the general overview of the dissertation, the objectives, motivations behind the work and the major contributions. Chapter II motivates the proposed paradigm shift towards FPGA-Based network-on-chip in the field of embedded hardware systems. The chapter achieves one of the major objectives of the dissertation. The chapter describes the target NoC architectures namely, the Mesh, Torus and Fat-tree networks. Also the chapter discusses the NoC router and switch, micro-architecture and buffer management of the on-chip networks and scalable NoC and router configurations. The dissertation further describes an extensive literature review of related previous works and how they differ from the proposed

approaches.

Chapter III describes a parametric performance evaluation and design trade-offs for the FPGA-Based network-on-chip. It achieves the second objective of the dissertation by evaluating the performance impacts of NoCs parameters in the context of the FPGA. The chapter proposes two models namely, FPGA-based network performance and self-adaptive models. The chapter describes extensive experimental and simulation results of synthesizing NoC in FPGAs. The chapter evaluates FPGA synthesis results in terms of area and clock frequency, it then evaluates FPGA network cost performance and critical path delay analysis. The chapter further describes extensive simulation and comparison results under benign and adversarial traffic patterns for the three NoC architectures. The chapter describes qualitatively the benefit of the proposed evaluation methodology in the context of FPGA and its possible applicability in super computing and related application domains.

Chapter IV proposes the paradigm shift towards autonomous self-aware and adaptive routing technique ASAART. The chapter describes the proposed methodology as applied to fault tolerant routing in wireless sensor networks. It describes the technique for route formation and self-aware and adaptive route repair in the ASAART routing protocol. In addition, it discusses the reliable route formation in ASAART. The chapter describes an extensive simulation under five different simulated scenarios. It evaluates the routing performance of ASAART

compared with the state-of-the-art fault tolerant routing namely, SHR and SSR protocols. The chapter describes three scenarios of network density test and transient and permanent node failure test scenarios. The chapter clearly showed the benefit of the proposed approach in achieving high resiliency to errors and failure and high routing performance in terms of throughput, latency, energy efficiency and reduced MAC layer packet retransmissions.

Chapter V achieved the fourth and fifth objectives by proposing a proof of concept of the design and implementation of FPGA-based intelligent on-chip sensor network monitoring and control using dynamically reconfigurable autonomous sensor agents. The chapter describes the design methodology of the sensor network monitoring and controller interface design for various on-chip sensor components for the temperature, voltage and thermal sensors. The chapter then validates the proposed design by providing an RTL level simulation results in form of waveforms for the monitoring and control unit. The chapter describes a case study, an implementation of on-chip sensor network monitoring and control using dynamically reconfigurable autonomous sensor agents. The chapter describes the design of various hardware components and application user interface in the form of a web page for the proposed case study. It then further presents the FPGA utilization results of various on-chip sensor components and shows qualitatively the significance of the proposed approach in terms of fewer numbers of logic consumption. The chapter describes extensive experimental

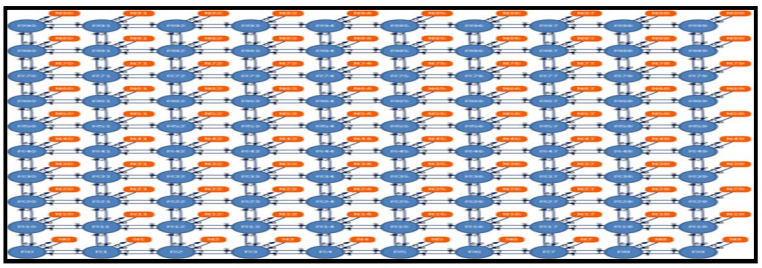
results for monitoring and control of FPGA on-chip sensor networks. The results show high accuracy in the monitored and control measured sensor reading with dynamic reconfiguration capability which confirms the applicability of the proposed paradigm shift to different application domains.

Chapter VI summarizes the autonomous self-aware and adaptive reconfigurable system paradigm and a discussion of its possible areas of applications in future embedded systems and other related applications.

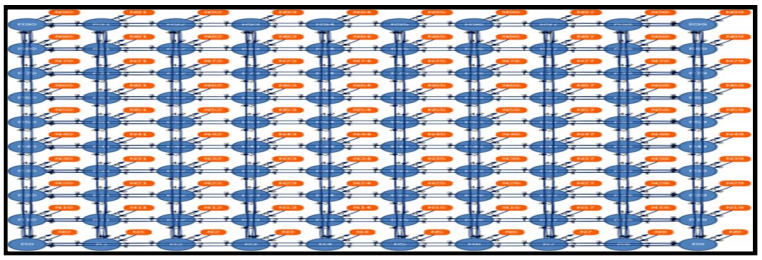
II. Background and Related Works

A. The Target NoC Architectures

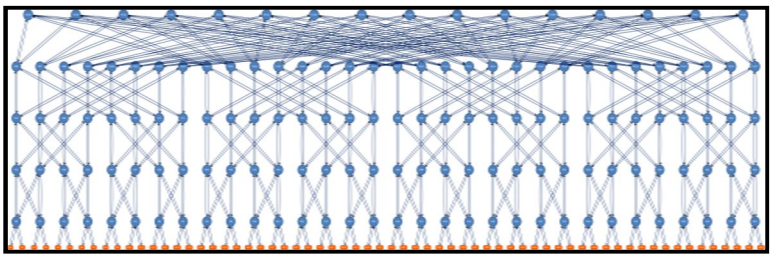
This chapter was written based on the published research papers [173], [174] and [175]. This subsection describes the topologies of the three NoC architectures evaluated in this dissertation. The topologies are illustrated in Figure 1 (a-c). These illustrations were designed using [70].



(a) 10x10 2D Mesh (Mesh100)



(b) 10x10 2D Torus (Torus100)



(c) Fat-Tree (Fat-tree144) networks

(Figure 1) Illustration of the NoC topologies.

1. 2D mesh network

Figure 1 (a) represents a k-array n mesh network (10x10 2D Mesh). A Mesh network consists of m columns and n rows. The routers are situated at each intersection of two wires and computational resources situated near these routers. The Mesh network topology is one of the key network architectures in which nodes are connected through many redundant routes between nodes. The 2D Mesh is an obvious choice for tiled architectures, since it matches very closely with the physical layout of the die [30]. Nodes in the Mesh network are connected in k-ary n-dimensional mesh (k-ary n mesh) formation, as shown in Figure 1 (a). The intellectual property nodes (IPs) are connected to a NoC switch by a Network Interface (NI) incorporating the connection management and data fragmentation functions. Because of the simple connection and easy routing provided by adjacency, the Mesh network is widely used in parallel computing platforms [16]. A Mesh network is orthogonal, providing a uniform interconnect length between nodes and uniform performance across nodes [71].

Routing: The Mesh network uses dimension order routing, in which a packet is routed towards the X axis and then towards the Y axis, to ensure reliable transmission and to avoid deadlock [69].

Flow Control: This refers to the protocol for managing and negotiating the available buffer space between routers. Due to physical separation and the speed of router operations, it is not always possible for the sending router to have up-to-date knowledge of the buffer status at the receiving router. Thus, a credit-

based flow control periodically sends credits that need to be tracked by the endpoints. This establishes a dialog between sender and receiver nodes, permitting and halting the advance of information. If a packet is blocked, it will require some buffer space in which to be stored. Under credit-based flow control, the sending router tracks credits from its downstream receiving routers. At any moment, the number of accumulated credits indicates the guaranteed available buffer space at the downstream router's buffer [4], [72].

2. 2D torus network

Figure 1(b) represents an m x n torus structure (10x10 2D Torus) based on an m x n mesh topology with a wraparound channel added to each row and column. The wraparound channels help reduce the diameter and the average distance per node. The IPs are connected to a NoC switch by a Network Interface (NI) incorporating the connection management functions. Each router is connected to four neighboring routers. As such, each router has five ports. Adding wraparound links to the mesh creates the torus topology, decreasing the average and maximum hop counts and doubling the bisection bandwidth. These links also double the number of wiring channels per tile edge. A Torus network has better path diversity and more minimal routes than a Mesh network [69], [71].

Routing: The Torus network uses dimension order routing.

Flow control: The Torus network uses a credit based flow control for efficient packet transmission.

3. Fat-tree network

In a Fat-tree topology nodes are routers and leaves are computational resources. The routers above a leaf are known as the leaf's *ancestors* and the leaves below an ancestor are known as its *children*. Each node has replicated ancestors, allowing many alternative routes between nodes [69], [73]. In this dissertation, the configurations of the Fat-tree provided in table 1 (an n-ary fat-tree with a height of k) is selected and used. Fat-tree topology has a unique characteristic: it allows all non-conflicting end-to-end connections to have full wire bandwidth. That is, any node can communicate with any other node in the network while keeping full per-channel bandwidth available for communication, provided no other nodes are communicating with the two nodes involved [71,73]. We should be aware that implementation of a direct fat-tree network does not require long wires; the maximum wire length is two tile spans. This is possible because the tiles can be connected so that routing resources are shared [69], [71,73]. This structure makes Fat-tree networks, superior in terms of hardware resource (i.e. FPGA utilization as will be demonstrated in chapter III).

Routing: The Fat-Tree network uses nearest-common ancestor routing (nca). Packets are adaptively routed upward to the common ancestor and deterministically downward to the destination [69].

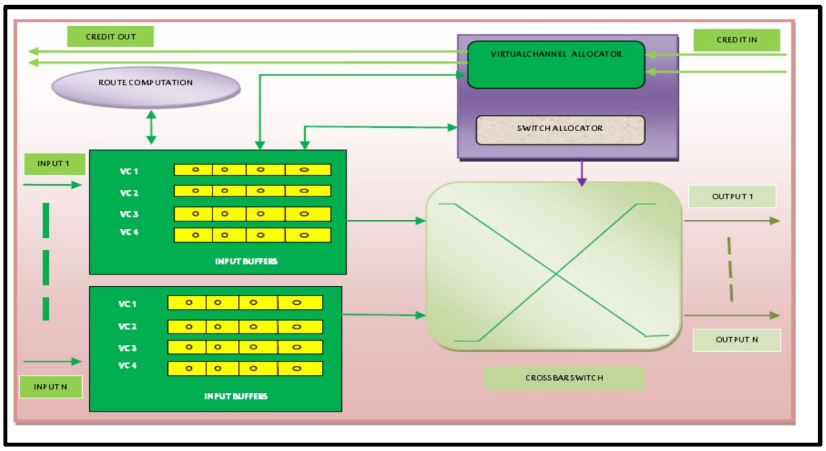
Flow control: The Fat-tree network uses credit-based flow control.

B. NoC Router Micro-Architecture

1. NoC credit-based virtual channel router micro-architecture

Network-on-chip routers forward packets from a source node to a destination node within the network. They must be designed to meet latency and throughput requirements while respecting the tight area and power constraints. This is the primary challenge facing designers of many-core systems [1], [71,73]. Even as bandwidth capacity and router complexity increase, very simple routers (e.g., wormhole, un-pipelined, no virtual channel, etc.) may still be used when high throughput is not needed, to keep the area and power overhead low. The challenge arises when throughput demand increases [4], [71-75]. A router's architecture determines its critical path delay, which in turn affects per-hop delay and overall network latency. Router micro-architecture also affects network energy, as it determines the logic circuit components of a router and their activity. The implementation of routing, flow control and the actual router pipeline will affect the efficiency at which buffers and links are used, and thus, overall network throughput. The area footprint of the router is directly determined by the chosen router micro-architecture and underlying logic circuits [1], [75]. Figure 2 shows the micro-architecture of a credit-based virtual channel router. The major components of the router are the input buffers, route computation logic, virtual channel allocator, switch allocator, and the crossbar switch. NoC routers are generally input-buffered, meaning packet storage occurs only in buffers at the input ports, allowing use of single-ported memory. In Figure 2, the dissertation

assumed four virtual channels at each input port, each with its own buffer queue four flits deep. The buffers are responsible for storing flits when they enter the router, and maintaining them for the duration of their stay at the router. The route computation block computes (or looks up) the correct output port for the packets. The virtual channel and switch allocators determine which flits are selected to proceed to the next stage, where they traverse the crossbar. Lastly, the crossbar switch is responsible for physically moving flits from the input port to the output port [4], [69], [74].

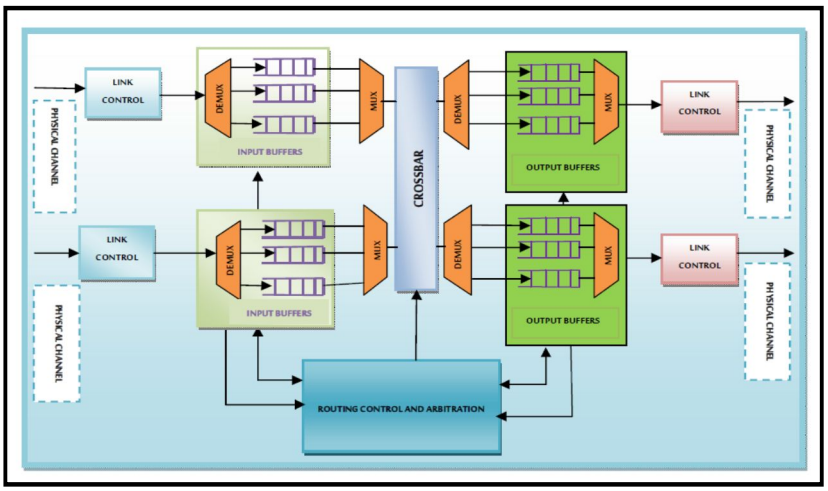


(Figure 2) Shows the NoC credit-based virtual channel router micro-architecture

2. NoC switch micro-architecture

A network switch is a device that channels incoming data from any of multiple input ports to the specific output port that will take the data toward its intended destination. Network switches implement the routing, arbitration, and switching functions of packet switched networks. The switches also implement buffer management mechanisms. For some networks, switches also implement part of the network management functions for exploring, configuring, and reconfiguring

the network topology in response to boot-up and failures [4], [74]. The internal data path of a switch provides connectivity among the input and output ports. Most high-performance switches implement an internal crossbar to provide non-blocking connectivity within the switch, allowing concurrent connections between multiple input-output port pairs [69], [74]. Buffering of blocked packets are performed by First in First Out (FIFO) or circular queues, which can be implemented as Dynamically Allocatable Multi-Queues (DAMQs) in static RAM to provide high capacity and flexibility [4], [74]. These queues can be placed at the input and output ports, centrally within the switch, or at both the input and output ports of the switch. Figure 3 shows an input-output buffered switch. The routing control unit is usually implemented as a centralized resource, although it can be replicated at every input port. Routing is done only once for every packet, and packets are usually large enough to need several cycles to flow through the switch. As a result, a centralized routing control unit (as illustrated in Figure 3) rarely becomes a bottleneck for the switch architecture [69], [74].

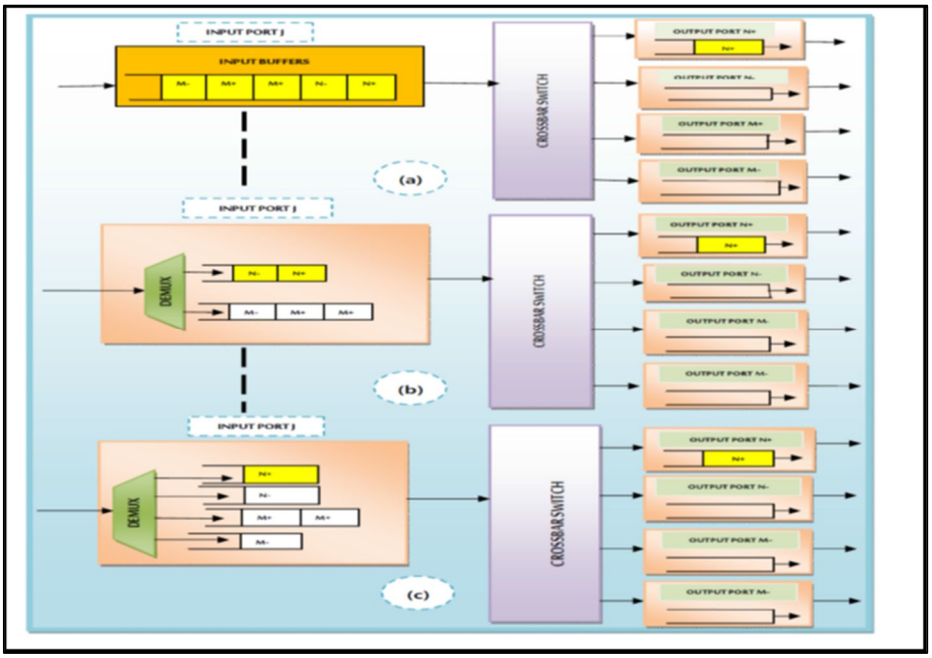


(Figure 3) Shows NoC input-output buffered switch micro-architecture.

3. NoC buffer management

A network buffer is a temporary storage area, usually in RAM. The purpose of most buffers is to act as a holding area, enabling the CPU to manipulate data before transferring it to a device. Buffer queues are memory space set aside specifically for either storing a packet that is awaiting transmission over a network or storing a packet that has been received over a network [4], [74]. Buffer queues can be located at the switch input and or output. Output-buffered switches have the advantage of completely eliminating Head-of-Line blocking (HoL). Head-of-line blocking occurs when two or more packets are buffered in a queue, and a blocked packet at the head of the queue blocks other packets in the queue that would able to advance where they at the head of the queue [4], [69], [74]. However, Head-of-Line blocking can occur within input port queues, as shown in Figure 4 (a). This can reduce switch output port utilization to less than sixty percent, even when packet destinations are uniformly distributed [74]. As shown in Figure 4 (b), the use of two virtual channels can diminish HoL blocking, but does not eliminate it. A more effective solution is to consolidate the input queues as Virtual Output Queues (VoQs), shown in Figure 4 (c). Under this solution, each input port implements as many queues as there are output ports, thus providing separate buffers for packets destined to different output ports. The major drawback of VoQ, however, is the cost of redundancy and coordinate lack of scalability, as the number of VoQs grows quadratically with the number of switch ports. One way to address this limitation is to dynamically assign only a fraction of the queues to store those

packets headed for congested destinations [4], [69], [74–75].



(Figure 4) NoC buffer management illustrating head–of–line blocking (HoL).

C. FPGA–Based NoC and Router Parameters

1. Configurable NoC and router parameters

This subsection gives a brief explanation of the NoC and router parameters used in this dissertation for the performance evaluation of the three target NoC architectures [4], [70], [76], [77]. These parameters are as follows:

1. *Flow control*: This parameter refers to the protocol for managing and negotiating the available buffer space between routers. This dissertation uses a credit–based flow control that periodically sends credits that need to be tracked by the endpoints. In contrast, peek flow control only exposes bit masks indicating which virtual channels can accept flits [4], [70].

2. *Router type*: This parameter specifies the router type used. Input-queued routers are simple and offer the lowest performance. VC-based routers offer higher performance and traffic isolation, whereas VoQ-based routers do not offer the highest performance, but do not require traffic isolation. This dissertation used VC-based routers based on their superior performance.
3. *Exposed edges port*: This parameter enables the unused ports at the edges and corners of Mesh networks. Each corner router exposes two additional ports. While the rest of the edge routers expose one additional port. This parameter is enabled for all networks.
4. *Virtual Channels* (VCs): This parameter reduces Head of Line (HoL) blocking, but increases hardware resource usage. This dissertation varied the number of VCs from 2 to 6 and measured the FPGA area (PLUT, %LUT) and clock frequency (CLK_FREQ) values as illustrated in tables 2-5.
5. *Flit Data Width* (FDW): This parameter specifies flit data width in bits. Flits maintain extra bits for flow control and bookkeeping purposes. This parameter is varied to obtain the synthesized FPGA area and clock frequency values in tables 2-5.
6. *Flits Buffer Depth* (FBD): This parameter specifies the depth of flit buffers in flits. Deeper buffers offer better performance by allowing packets to make more progress under heavy loads. This parameter is varied to optimize the synthesized FPGA area and clock frequency values shown in tables 2-5.
7. *Allocator*: This parameter specifies the type of the allocation algorithm used for matching input requests with outputs. This dissertation used the Separable Input -First Round Robin.
8. *Pipeline router core*: This parameter adds a pipeline stage right after the

router allocation unit. This parameter can produce a 1.5x to 2x clock frequency improvement for typical router configurations [4]. This parameter is enabled for all networks at design time to improve clock frequency performance during FPGA synthesis.

9. *Pipeline router allocator*: This parameter pipelined the router allocator that performs input-output matching and determines which flits will traverse the router each cycle. This parameter feature is especially useful for increasing frequency in networks containing many virtual channels or higher-radix routers (i.e. routers with many send/receive ports, such as the Mesh144, Torus144, and Fat-tree144 networks.), whose critical path is dominated by the allocation stage. This parameter is enabled for all networks at design time to offer higher frequencies during FPGA synthesis, as shown in tables 2-5.
10. *Pipeline links*: This parameter adds a pipeline stage to all flit and credit links in the network. It helps to increase clock frequency in large or congested designs where router inputs are placed far apart. This parameter is enabled for all networks at design time.
11. *Virtual links*: When this parameter is enabled, VCs and output ports are locked while a packet is transmitted, in order to avoid interruption of multi-flit packets. Only packets that belong to a higher-priority virtual channel are allowed to interrupt packets on lower priority virtual channels. Enabling the virtual links reduces the buffering requirements at the receiving endpoint since, for any virtual channel, all flits of a packet are guaranteed to arrive before any flits of another packet on the same virtual channel [4], [70]. Virtual links require a small increase in hardware, but in return, they can significantly reduce the reassembly buffering and logic requirements at endpoints. This parameter is enabled for all networks at design time.

In this dissertation, the analysis focuses on the three basic NoC parameters (FDW, VCs, and FBD) and how they affect the performance of the three targeted architectures on FPGA.

2. Scalable on-chip networks and router configurations

Table 1 gives the on-chip networks and router configurations for the different interconnect architectures analyzed in this dissertation. A point to note here is that the dissertation considered only the largest network grid sizes. For better understanding, the number next to each network name indicates the number of supported network endpoints (nodes). For example, Mesh100 corresponds to a network with one hundred connected routers, while Torus121 corresponds to a network with one hundred and twenty-one connected routers.

(Table 1) Router configurations for different network-on-chip architectures

2D Network Type	Number of Routers (Network Nodes)	Number of Send/Receive Ports	Number of Input/output Ports
10 x10 MESH (MESH100)	100	100	5/5
11x11 MESH (MESH121)	121	121	5/5
12 x12 MESH (MESH144)	144	144	5/5
10 x10 TORUS (TORUS100)	100	100	5/5
11x11 TORUS (TORUS121)	121	121	5/5
12 x12 TORUS (TORUS144)	144	144	5/5
FAT-TREE56	56	32	4/4
FAT-TREE100	100	56	4/4
FAT-TREE121	121	58	4/4
FAT-TREE144	144	64	4/4

D. Related Works

1. FPGA-based network-on-chips

A few number of recent efforts have shown ways to evaluate the performance of NoC interconnect architectures [1-4], [78-92],[168]. Ju and Yang [78] analyzed three common 2x4 topologies (2D Mesh, 2D Torus, and Hierarchical Mesh) and compared their performance and resource consumption, but with relatively few comparisons, the flexibility of their approach remains uncertain. Kundu et al. [79] presented a detailed study of the Mesh-of-Tree (MoT) topology and explored its potential in communication infrastructure design for 2-D NoCs, compare its performance and cost with those of the Butterfly Fat-Tree (BFT) topology and two variants of the mesh topology for a fixed number of cores under the same bisection width constraint. Their simulation results showed that MoT delivers better performance than other topologies, while consuming less energy per packet than mesh networks that connect single cores to each router. Although other efforts have shown ways to assess the performance of packet-switched FPGA NoC and routers, a systematic understanding of router performance evaluation [1-2], [78-87],[168]remains elusive. Abdelfattah and Betz [1] presented detailed design tradeoffs for hard and soft FPGA-based NoCs. Similarly, Papamichel and Hoe [82] presented CONNECT: Re-examining conventional wisdom of designing NoCs in the context of FPGAs. Lee and Shannon

[84] presented a predicting the performance of application specific NoCs implemented on FPGAs. The authors in the paper [82] performed systematic comparisons between ASIC NoC designs and their FPGA-based NoC optimizations, quantitatively demonstrating the benefits of the latter. Abba and Lee [92] presented a Bayesian networking approach and self-adaptive scheme for NoC parameter performance on FPGA-based NoC.

Most of the work on performance evaluation of NoCs is based on analytical models and simulation. In general, the traffic generator and the traffic meter are modeled at the transaction level using SystemC, and the network elements are modeled at the register-transfer level using either SystemC or a hardware description language, such as VHDL [93]. A number of previous works have used more abstract models in which all components are modeled at the transaction level [72]. The number of previous works in which NoCs are evaluated in hardware is still quite small. Wolkotte et. al [94] presented a platform for on-chip performance evaluation of an NoC. The traffic generator was run on an ARM9 processor and injected traffic into an FPGA on which a NoC was implemented. Only a single router was synthesized on this FPGA, and the full NoC was emulated sequentially. According to the authors, the proposed approach makes it possible to evaluate large NoCs, since the network size is not constrained by the logic density of the FPGA. Their on-chip evaluation was 80 to 300 times faster than evaluations using a SystemC model running on a computer. In Genko et. al [95], a hardware-based approach was used for performance evaluation of NoCs in FPGA.

They used instances of a synthesized traffic generator to create and inject packet flows into the NoC based on stochastic models (Uniform distribution or Burst mode) and on traces of actual traffic. They also used instances of a traffic receptor to eject these flows from the NoC and collect data for performance evaluation. Using their platform, a one billion packet emulation needed only three minutes and twenty seconds to run on the chip. In a SystemC simulation, the same experiment would take approximately six days to run, with a speedup of four orders of magnitude [95]. This dissertation tries to address the limitations of the aforementioned proposed approaches.

2. Self-adaptive and fault tolerant routing

Quite a few numbers of related works have been conducted with adaptive and fault tolerant routing algorithms for wireless sensor networks [16-29], [36-45], [96-98]. For instance, Gilbertt *et al.* [44,96] proposed a self-selective fault tolerant routing protocol for a wireless *ad hoc* network. The protocol employs a combined radio broadcast and autonomous programming technique to implement routing with reduced overhead. The routing protocol routes the packets via a near optimal path between nodes at runtime. Simulation results showed high routing performance under heavy network traffic, as well as network node and link failures. The authors in [28,97] presented a novel protocol for self-healing in sensor network routing. The approach employs broadcast transmission and prioritized slotted back-off delay for sensor nodes to use their hop distance from

the destination in order to determine how to transmit packets. By doing so, this ensures a dynamic traversal of minimum routes without nodes that specifies those neighboring nodes that transmit network packets. However, when faulty routes are encountered, the scheme locally and dynamically re-routes packets in order survive the shortest routes despite spontaneous network topology changes.

Other approaches that avoid neighbor state maintenance and allow destination nodes to contend with transmitting network packets are given in [30-31]. These two protocols are similar to SHR [28,97] and SSR [44,55,96], which use local information instead of global network state information. Unfortunately, none of the two proposed approaches provide route repair mechanisms. GRAB [30] employs a very complex fault tolerant mechanism by allowing the flow of redundant network packets to take multiple paths to destinations. Hellsenbttel *et al.* [32] employed a location coordinate system to allow only destination nodes within a given region to communicate with each other. Zori and Rao [33] used a concept similar to [32], and incorporated a back-off delay mechanism. Such mechanism makes use of a dual-radio concept with a busy tone signaling to enforce the channel to be clear before transmitting data in order to minimize the probability of network packet collisions. Another approach is given by Blum *et al.* [34], who employed an eligibility region given as a 60% fan shape that extends from a source towards a destination. Upon this, if a source node does not "hear" a response from any of its neighbors, it moves the eligibility region and looks for other neighbors. All these

approaches use packet forwarding techniques, *i.e.*, back-off delay and RTC/CTS schemes that result in packets that forward overhead.

Other methods for direct routing are given in DSR [99], AODV [100], and DSDV [35]. The work presented by Chokers and Elizabeth [100] is regarded as the most popular directed routing protocol. Such protocol uses the route request and a reply mechanism in order to set up network paths between sources and destinations. This process yields a routing table by each node that contains a path from the sources to destinations. Johnson *et al.* [99] employ a technique where the source node should contain the complete route information in the packet header. This also means that multiple routes are stored at the source node in order to avoid node failure. The limitation of the DSR protocol is its increases in the amount of memory use of the source node because of intermediate node storage and routing overhead. AODV [100] employed the use of a hello message in order to detect and fix broken network links. This results in an increase in network latency, particularly when dynamically looking for new routes.

Gelenbe and Liu [55] presented a cognitive and quality of service approach to routing in wireless sensor networks. They used the concept of smart packets for network path discovery coupled with reinforcement learning and neural networks. In addition, they used the ant colony concept to mimic the pheromone-based technique ants use to find a given path and communicate the information to colony members. Experimental results showed the efficiency of their approach in

adapting to network changes over time. Self-adaptive routing in multi-hop sensor network was presented by Bourndenas *et al.* [36]. A study of an effective method that uses time series analysis to forecast the occurrence of errors and faults was presented. The work considered an autonomic routing service through adaptation to avoid areas where failure or errors are expected. Simulation results showed the benefit of their approach. The authors in [42,43] compared the performance of three routing protocols, namely, the Multi-Parent Hierarchical (MPH), AODV, and DSR protocols. The protocols are evaluated both when exposed to different "jamming" attacks and without attacks, and considering diverse locations of the jammer. Simulation results show that MPH routing has greater immunity to tolerating attacks than DSR and AODV. This is because MPH reduces and encapsulates the network segment when subjected to a network attack. In addition, the self-configuring features of MPH yield higher resiliency and better routing performance compared with AODV and DSR protocols. Del-Valle-Soto *et al.* [43] proposed metrics for efficient energy consumption in wireless sensor networks. They compared the performance of three routing protocols, namely MPH, AODV, DSR, and Zigbee Tree Routing (ZTR). Simulation results demonstrated the impacts of communication metrics on performance, throughput, reliability, and energy consumption. They showed that MPH achieved a 19.3 % reduction of network packet retransmissions, 26.9% reduction in routing overhead, and 41.2% increase in protocol recovery from topology failure compared with

AODV routing. In addition, their approach achieved a 15.9 % decrease in energy consumption compared with AODV, 13.7% compared with DSR, and 5% compared with ZTR protocols. Apart from the few related works mentioned above and in the bibliography, this dissertation proposes an autonomously self-aware and adaptively fault-tolerant routing techniques for wireless sensor networks. The dissertation integrates the autonomic self-aware and adaptive mechanism for route formation and repair in the presence of transient and permanent node failure rates in order to achieve high routing performance, energy efficiency and resiliency against errors and failure of sensor nodes, and adapt to simultaneous network topology changes.

3. Self-selective routing

The SSR routing [44,45,55,96] has the characteristics that network nodes know their distance through the mechanism of route request and reply. In this scheme, the nodes broadcast packets with an estimated distance to the destination neighbors. The node then employs the SSR protocol to determine autonomously the node that should transmit a given packet to the next node using a prioritized transmission backup delay. Upon receiving the packet, the node, then schedules further packet transmission immediately with a random delay that is proportional to its path length of the destination. The transmission back-off delay is expressed using the mathematical expression given in Equation (1) as [44,45,55,96]:

$$Trans\_Back\_off_{\_Delay} = \begin{cases} \beta * \left(\alpha - \alpha_{expected} * \text{Random}(0,1)\right) \text{ if } \alpha > \alpha_{expected} \\ \frac{\beta}{\alpha_{expected} - \alpha + 1} * \text{Random}(0,1) \text{ if } \alpha \leq \alpha_{expected} \end{cases} \quad (1)$$

Where α is the hop count of the destination node, and $\alpha_{expected}$ is the sender node expected hop counts in the destination reply packets. The random function is the random number generator that produces real values uniformly distributed between the interval [0,1]. This random function is used to randomize the delays and reduce packet collisions. The parameter β is a scaling factor used to stretch the random delay values generated by the randomize function. Equation (1) assigns a $Trans\_Back-off_{\_Delay}$ greater than β to those nodes with hop counts that are greater than $\alpha_{expected}$. In this case, the smaller the value of α, the least is the $Trans\_Back\_off_{\_Delay}$, and the node has the highest probability of transmitting the packet [44,45,55,96]. The limitations of SSR are:

1. The SSR routine computes a node's transmission back-off delay in continuous time, in contrast to slotted time. The disadvantage is that it has a higher rate of packet collisions.
2. Because of non-zero probability, the packets may travel a longer route while a shorter route exists that increases network delay.
3. In SSR, there is no route repair mechanism for propagating packets around faulty routes.

4. Self-healing routing

SHR routing [28,97] employs the concept of a prioritized time-slotted transmission back-off delay and route repair routine. The first improvement on

SHR is as follows: when a given node receives a packet, the node employs Equation (2) instead of Equation (1) to compute the delay before transmitting the packet:

$$Trans\_Back\_off\_{}_{Delay} = \begin{cases} \beta * \left(\alpha - \alpha_{expected} + Random\,(0,1)\right) \; if \; \alpha > \alpha_{expected} \\ \frac{\beta}{\alpha_{expected} - \alpha + 1} * Random(0,1) \; if \; \alpha \leq \alpha_{expected} \end{cases} \tag{2}$$

From Equation (2), we can determine that the computed delay is such that the nodes closest to the destination forward their packets, rather than those far away. Furthermore, Equation (2) assumes that there are delays, and thus ensures the response of those nodes farther away from the destination than the sender, according to their distance from the destination. In doing so, no packets travel further than the necessary routes, even if there are no nodes closer to the destination: Another addition to SHR is the transmission back-off delay in the slotted time, in contrast to the continuous time used in SSR [44,45,55,96]. The transmission back-off delay is given by Equation (3) as [28,97]:

$$Trans\_Back\_off\_Delay_{SLOTTED} = \left\lfloor \frac{Trans\_Back\_off\_{}_{Delay}}{Wds} \right\rfloor * Wds \tag{3}$$

Where Trans_Back_off_Delay is as given in Equation (2), and *Wds* is the width of the used slot.

This dissertation addresses the limitations of the SHR and SSR routing protocols by using combined continuous and slotted prioritized transmission back-

off delay to obtain local and global network state information and multiple randomize function for faster routing convergence and efficient and reliable route repair technique in the presence of transient and permanent node failure rates.

5. On-chip sensor network monitoring and control using dynamically reconfigurable autonomous sensor agents

The on-chip sensor network runtime parameters such as voltage, current, and crosstalk noise are used in controlling and monitoring information from the on-chip sensor network environment. Therefore, as the signals are transmitted within the logic blocks, they encounter delays. This delay component of the logic circuit should be minimized. Hence, the use of FPGA to implement sensor networks will be an efficient solution for a reliable on-chip sensor network packet transmission and retransmission scheme because logic and critical-path delays in FPGAs are minimal. Several methods to overcome these limitations include the use of feedback control system [62-64], [101-103].

Chengun et al. [56] proposed a multiplexing scheme that is simple enough to synchronize the sampling in a multichannel acquisition system with different sampling rates and many combined analog inputs as well as an improved sampling control to enhance system performance. FPGA was used in their design implementation. They used a sampling lookup table (LUT) to design a rule-based synchronous sampling frame to reduce the internal wiring and enhance resource utilization. The results of their experiments demonstrated the benefit of the

proposed approach. The proposed scheme can be employed in other related applications. Perera et al. [57] presented the design of low-cost, reconfigurable, and programmable smart sensor node using ZigBee. Their design was implemented using FPGA that incorporated the basic functionalities of the IEEE 1451 standard. The design of the sensor nodes comprised a processing unit and transducers with control capabilities in a single core to ensure reliable processing speed as a result of interprocess communication within a die. The experimental results on the basis of the measured pH value and temperature of water samples demonstrated the benefit of their approach.

Nídia et al. [58] proposed a distributed autonomic inference machine. This machine could allow the sensor nodes to self-manage and to contextualize the tasks based on fuzzy logic. The results of their experiments demonstrated that the proposed machine is energy efficient by minimizing the number of messages to 48.8% and achieving a 19.5% reduction in the energy consumed by the network. The review paper by Jesús et al. [59] provided an overview of the recent wireless sensor network (WSN) middleware systems that addressed the autonomic properties. Their aims were to determine the best autonomic computing method that will provide a self-management feature of WSNs for middleware systems and to study the various interactions and behavior in WSN components. Their conclusions were summarized as follows: first, they addressed the major concerns about self-configuration, self-healing, self-optimization, and self-protection

properties of the autonomous systems. Second, their investigation used diverse methods to manage the dynamic behavior of middleware systems for WSN, which included policy-based reasoning, context-based reasoning, feedback control loops, mobile agents, model transformations, and code generation. Finally, they identified the lack of complete system architecture design that provides full autonomy to the sensor network.

Designing an intelligent monitoring and control system requires the use of smart or intelligent and reliable sensors. The recent work by Echanobe et al. [60] has proposed a system-on-chip-based intelligent multiprocessor embedded system to control ambient environmental parameters. They achieved the intelligent capability of their proposed approach by employing the concept of Neuro-fuzzy system, which can reason and adapt to situational environmental changes. The authors Huijsing et al. [101] and Kirianaki et al. [104] proposed the concept of intelligent sensors. These are sensors with embedded intelligence and control system that can perform diverse functionality for environmental monitoring, self-adaptation, runtime observability, and network packet monitoring. Approaches to provide a smart interface for application-specific integrated circuits and FPGAs are provided by the authors of [105-112].

Other approaches to intelligent sensor system monitoring are presented in [113-116]. These approaches focused on the functionalities and communication protocols using transducers to make network communication possible. Ring and

delay sensors have been used to study temperature and variations in FPGA [61,63]. Franco et al. [63], a ring oscillator was proposed to monitor the temperature in an FPGA. The work considered a quadratic frequency in relation to a temperature−sensor transfer function and examined how voltage variations affect the sensitivity of a sensor in relation to the number of stages in the oscillator. The authors found that for larger oscillating chains, the voltage variations can be easily analyzed. Xi et al. [105] used ring oscillators to measure variability and temperature within the processor die. They also measured the power consumption when the system processor die is operating and in idle state.

The use of sensor networks to measure diverse runtime parameters such as crosstalk noise, on−chip interconnect temperature, switching activity, clock duty cycle, and other parameters is given in [113−119]. Petrescu et al. [117] proposed a signal integrity and efficient architecture to monitor diverse network−on−chip physical parameters, particularly temperature and voltage. Similarly, McGowan et al. [31] described the control system of a 90−nm Itanium processor, which utilizes on−chip sensors to measure power and temperature and modulates voltage and frequency to optimize the system performance. Sohn et al. [119] proposed a sensor−based solution for static random access memory to mitigate the uncertainties and fluctuation that exist among different device parameters.

A few number of methods have been proposed to collect useful information at runtime for on−chip sensor networks. For instance, Chan et al. [120] proposed an

approach to use system management bus for communication between IP cores and a thermal-aware power management IP for low-level power management functions. Furthermore, Alexander et al. [113] proposed a hierarchical architecture to collect runtime parameters using network on-chip. Another approach was proposed by Ciordas et al. [114], which was a monitoring service framework, to support runtime observability of network-on-chip NoC behaviors, and application debugging was proposed.

The authors in [121,122] proposed a design of a web server for remote monitoring and control using FPGA. They provided a procedure for the implementation and application of their approach to remote monitoring of sensor networks on FPGA. Gomez-Ouna et al. [122], the authors proposed a monitoring infrastructure for FPGA self-awareness and dynamic adaptation. They employed sensor networks to provide dynamic adaptation and obtain data from the FPGA with reduced cost and easy implementation. They showed the significant benefit of their approach. Finally, the authors Georgios and Dionisions [102] and Gabriel et al. [103] provided a detailed survey of the taxonomy of network-on-chip monitoring and control as well as FPGA-based design of sensor systems. However, the limitations of the previous approaches are as follows: (1) they did not provide real-time monitoring and a control mechanism for accurate measurement of the on-chip sensor network runtime parameters, and (2) they did not provide efficient and convenient interface between the user and the FPGA

to dynamically tune the operating ranges and reconfiguration refresh time to measure different on-chip sensor values.

In order to address the limitations of the previously proposed approaches. This dissertation, propose a low-level micro-architectural design infrastructure for real-time monitoring and control of on-chip sensor network-based systems for FPGAs.

E. Chapter Summary

This chapter describes the basic background on FPGA-Based networks-on-chip. It gives an explanation on the three basic NoC interconnect architectures as used in the dissertation. Also describe are the router and switch, micro-architectures and buffer management. Furthermore, the chapter describes the eleven configurable NoC and router parameters as used in the study. Finally, an extensive review of the related works and how they are different from the proposed methodologies are clearly explained and provided in the chapter.

III. A Parametric-Based Performance Evaluation and Design Trade-offs for the Interconnect Architectures using FPGAs for Network-on-chips

A. FPGA-Based Network Performance and Self-adaptive Models

This chapter was written based on the published research papers [92] and[173].

1. Bayesian networking model

The dissertation defines a Bayesian network as a Directed Acyclic Graph (DAG) where nodes in the graph represent a set of random variables X = {X_1, X_2,... X_n}. It is represented as a set T = (V, E) where V represents a set of nodes or vertices and E the set of arcs or edges that define conditional independence assumptions. The DAG defines a factorization of the joint probability distribution of the modeled random variable X. This joint distribution is obtained as a product of all conditional probabilities specified for each variable given its parent in the DAG [123]. We denote the vertices or nodes by X_i, i = 1,..., n. The joint probability of any X can be represented as follows:

$$P(X_1, X_2, X_3, \ldots X_n) = \Pi^{N}_{i=1} P(X_i / Pa(X_i)) \quad (4)$$

2. Proposed FPGA-Based network performance model

Herein, the dissertation presents the proposed methodology. Let define the following parameters:

VC: Virtual channel (size from two to six units)

FBD: Flit Buffer Depth (varies from two to six units in bits)

FDW: Flit Data Width (varies from two to six units in bits)

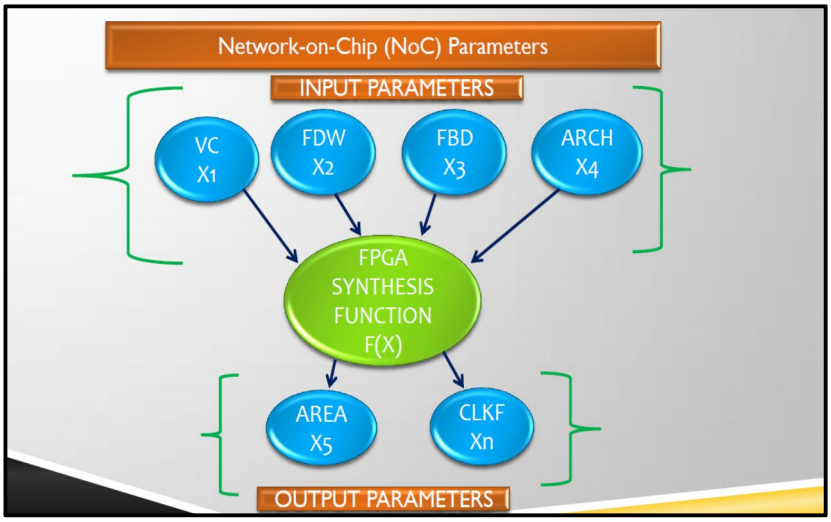
ARCH: Architectural parameters (router and switches, topology and flow control etc.)

AREA: synthesized FPGA area PLUT (%LUT, slice LUT used as logic and routing)

CLKF: synthesized FPGA clock frequency CLK_FREQ in (MHz)

P: Probability

$X_1, X_2, \ldots X_n$: are random variables representing these parameters.



(Figure 5) Shows the proposed network performance model.

In the proposed approach using the Bayesian network, the four parameters VC, FBD, FDW and ARCH influence the FPGA network performance and impacts both the area and clock frequency of the FPGA. The values of these parameters are

varied on the FPGA that is, X_1, X_2, X_3 and X_4 forming a different configurations and the resulting area X_5 and clock frequency X_n are observed. Figure 5 above shows the proposed model, each variable representing a unique on−chip parameter. Using the Bayesian network approach we define the following probabilities:

P (VC) = P (number of Virtual channels)

P (FBD) = P (size of Flit buffer depth)

P (FDW) = P (size of Flit data width)

P (ARCH) = P (architectural parameters router and switches, routing algorithms, topology and flow control etc.)

P (AREA) = P (synthesized FPGA area due to VC, *FBD, FDW* and *ARCH* parameters)

P (CLKF) = P (synthesized clock frequency due to *VC, FBD, FDW* and *ARCH* parameters)

From equation (4), the probability of the synthesized FPGA area and clock frequency are given as follows:

$$P(AREA) = \Pi^{N}_{i=1} P(AREA / parents(AREA_i)) \quad (5)$$

$$P(CLKF) = \Pi^{N}_{i=1} P(CLKF / parents(CLKF_i)) \quad (6)$$

Now putting the parameters in the equation (5) and (6) we get

$$P(AREA) = \Pi^{N}_{i=1} P(AREA / VC, FBD, FDW) \quad (7)$$

$$P(CLKF) = \Pi^{N}_{i=1} P(CLKF / VC, FBD, FDW) \quad (8)$$

The probability of the synthesized FPGA area and a clock frequency depends on

the variation of the probability of any of the parameter more than one or all the four parameters as in figure 5. More VCs reduce head_of_line (HoL) blocking, but increases hardware resource usage. The FDW specifies flit data width in bits internally, in addition to the bits used for data; flits maintain extra bits for flow control and bookkeeping purposes. The FBD parameter specifies the depth of flit buffers in flits, deeper buffers offer higher performance by allowing packets to make more progress under heavy loads.

Definition 1.

The dissertation defines each network configuration as a set of 3−tuple meaning that, each network configuration consists of different combination of the three basic parameters (FDW, VC and FBD). When these configurations are synthesized using the Xilinx ISE tools yields the resulting FPGA area (AREA %LUT, slice LUT used as logic and routing) and clock frequency (CLKF) in MHz. Therefore, we define each network configuration as:

Network_configuration (Parameter_1, Parameter_2, Parameter_3) = {AREA, CLKF} *(9)*

Where Network_configuration is the on−chip network name and grid (node size) of the network, Parameter_1 is the Flit Data Width (FDW) in bits ranges from 8 to 64 bits, Parameter_2 is the size of Virtual channel (VC) ranges from 2 to 6 units and Parameter_3 is the Flit Buffer Depth (FBD) in bits ranges from 8 to 64 bits respectively. Therefore, we can write equation (9) as:

Network_configuration (FDW, VC, FBD) = {AREA, CLKF} (10)

Equation (10) can also be written as:

$Network\_configuration\ (X_1, X_2, X_3) = \{X_5, X_n\}$ (11)

3. Proposed Self-adaptability model

Herein, the dissertation presents the proposed self-adaptive model [123-144]. In a more simplified form of the proposed model, as shown in figure 5, the set X = { X_i / i = 1, ..., n} denotes all n parameters that are varied on the FPGA. During run-time, different combinations of these parameters can be executed on the available architecture, each representing an operational mode β of the system. Ideally, each possible combination of parameters could be run as a network configuration on the FPGA (i.e. X_1, X_2, X_3 and X_4). Therefore, the idea is to additionally equip the system with an architectural parameter (X_4) to detect environmental changes (router and switches, topology, routing algorithm, flow control etc.) and degeneration of the system's performance. The system can then react by modifying the network configuration by changing the value of different parameters. The operational mode of the proposed model in an instant of time τ is represented by the set of active parameters β (τ) ⊆ X, where each parameter X_i ε β (τ) calculate the synthesized results $AREA_i$ (τ) and $CLKF_i$ (τ). These results are computed by the FPGA synthesis function F (X) to produce the resulting synthesized AREA (τ) and CLKF (τ) of the overall system at time τ. Therefore, we have:

$AREA\ (\tau) = F\{X_i\ (\tau)\}\ X_i\ \varepsilon\ \beta$ (12)

$CLKF\ (\tau) = F\{X_i\ (\tau)\ \}\ X_i\ \varepsilon\ \beta$ (13)

The FPGA function evaluates the quality of each active parameter $X_i \varepsilon \beta\ (\tau)$ by

the FPGA function F*(X$_i$ (τ), AREA (τ), CLKF (τ)),* which measure how well the FPGA function is synthesizing the area and clock frequency of each network configuration. Therefore, we have:

$$F\ (X_i(\tau), AREA(\tau), CLKF(\tau)) \quad (14)$$

B. FPGA Experimental Results

This subsection demonstrates the experimental methodology carried out to evaluate the performance of the Mesh, Torus and Fat−Tree architectures using FPGA. A total of 392 different NoC configurations was designed using [70], [144−146] on−chip network design tools. Each NoC architecture was redesigned in the Verilog Hardware Description Language (HDL) and then synthesized each NoC configuration using Xilinx ISE 14.3 to obtain the synthesis results in tables 2−5. The experiment targeted a large Xilinx FPGA Virtex−7 board (Device XQ7VX980T, Package RF1930, speed −2L), [146]. Since the intention was to employ a medium−sized FPGA device to observe the effects of varying parameters, the higher Virtex−7 device with a large grid size was chosen so that the impact of the parameters could be seen clearly as memory registers (routing resources) are abundantly allocated on this device. The values of the router parameters are tuned in order to optimize the area and clock frequency across different network configurations. The synthesis results include FPGA area, resource usage PLUT (percentage LUTs, slice LUT used as a routing and logic) and clock frequency (CLK_FREQ) in MHz. A timing analysis was performed using the Xilinx timing analyzer to determine the critical path delay for different

network configurations, and then evaluated the FPGA network costs for different network configurations. The dissertation employed the use of a flexible cycle-accurate simulation system [77] and simulate the NoCs under the benign (Nearest Neighbor and Uniform) and adversarial (Tornado and Random Permutation) traffic patterns. The resulting load-delay curves are provided in Figures 16-23.

1. FPGA synthesis results

Herein, the dissertation provides the FPGA synthesis results from the conducted experiment. Tables 2 to 5 show the results of synthesizing different network configurations using the Mesh, Torus and Fat-tree architectures. We tuned three of the parameters (FDW, VC, and FBD) to optimize PLUT and CLK_FREQ across different network configurations. For each combination of the three parameters (FDW, VC and FBD) is a different network configuration and the resulting synthesized FPGA area (PLUT) and clock frequency (CLK_FREQ) are observed. As a result, for each table, we have ninety eight (98) different on-chip network configurations and a total of three hundred and ninety two (392) networks. As can be seen in the table data, increasing the FBD, FDW, or VCs yields higher LUTs and affects the area and clock frequency of the FPGA networks. Note in Tables 2-5 that the Fat-tree network configurations showed better utilization of FPGA area (PLUT) and better clock frequency (CLK_FREQ) across different combinations of the three parameters. From the conducted experiment using the Xilinx ISE design tools it was discovered that the FDW and FBD parameters have the largest effect on FPGA resources, followed by the VCs parameter. This is due to the fact that, the parameters affect the FPGA routing

and logic resources. It is interesting to observe that the FBD and FDW parameters affect the LUTs count in an indeterminate manner, due to the quantization effects of distributed RAMs. This is also in line with the findings by Huan and DeHon [81], Papamicheal and Hoe [82], Chung et. al [83] and Abba and Lee [92]. To further analyze the results from the four tables and figures in this subsection, the dissertation provides the following definition of network configuration.

Definition:

A network configuration is a set of 3−tuples {FDW, VC, FBD} meaning that, each configuration is a function of the three basic parameters (FDW, VC, FBD). When these configurations are synthesized using the Xilinx ISE tools, they yield PLUT and CLK_FREQ in MHz.

Network_configuration (Parameter_1, Parameter_2, Parameter_3) =

{PLUT, CLK_FREQ} (15)

Where Network_configuration is the on−chip network name and grid (node size) of the network, Parameter_1 is the Flit Data Width (FDW) in bits ranging from 8 to 64, Parameter_2 is the size of the Virtual Channel (VC) ranging from 2 to 6 units, and Parameter_3 is the Flit Buffer Depth (FBD) in bits ranging from 8 to 64. Therefore, equation (15) can be written as:

Network_configuration (FDW, VC, FBD) = {PLUT, CLK_FREQ} (16)

A similar definition was used by Abba and Lee [92]. The dissertation presents an example from table 2. Applying definitions (15) and (16) above, we have the following network configurations and resulting FPGA area (PLUT, %LUT) and

clock frequencies (CLK_FREQ in MHz) as follows:

Mesh144 (8, 6, 32) = {27%, 205 MHz}

Torus144 (8, 6, 32) = {29%, 219 MHz}

Fat-tree144 (8, 4, 64) = {23%, 309 MHz}

2. Performance comparison of the synthesized FPGA network clock frequency (CLK_FREQ) results

Table 2 and Figures 6 (a-b) show the experimental results of synthesizing different network configurations using the Mesh, Torus and Fat Tree architectures. In table 2, the FDW was fixed at 8 bits, and varied VC from 2 to 6 and FBD from 8 bits to 64 bits. Then each network configuration was synthesized and obtained the synthesized PLUT, and CLK_FREQ (in MHz) across ninety-eight different configurations. As can be seen in Figure 6 (a-b), the Fatttree144 network configuration showed higher clock frequencies (above 300MHz) for different configuration of FDW, VC and FBD parameters. This is because the maximum frequency exhibited by Fat-tree network configurations is directly related to the FDW, FBD, VC parameters. Similarly, enabling the pipeline router allocator increases the clock frequency in the large Fattree144 network. The pipeline router allocator adds pipeline stages right after the allocation unit, this can produce 1.5x to 2x clock frequency improvements. This depends on the transistor switching and net delay or routing delay, which is minimal in Fat-tree network configurations. We observed a slight degradation in clock frequency due

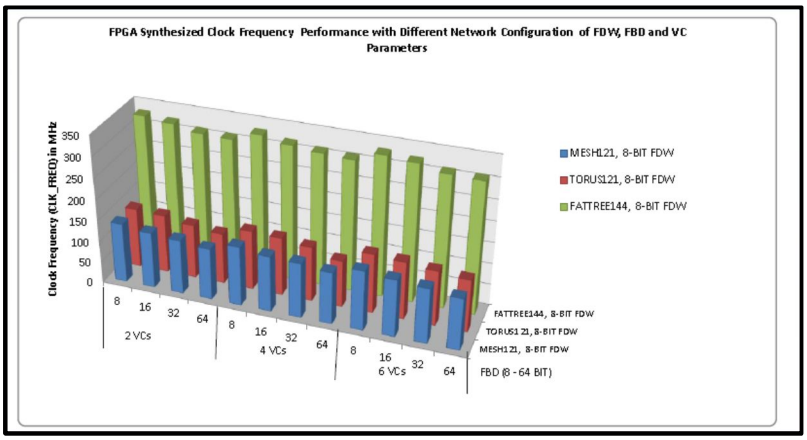
to the FBD parameter. With an FBD between 32 bits and 64 bits, the clock frequency degrades slightly from 331 MHz to 310 MHz. Similarly, increasing the number of VCs produces no performance degradation in Fat-tree network configurations. For the Mesh and Torus network configurations, the effect of the three parameters can be seen quite clearly, as the clock frequency degrades more sharply than it does for Fat-tree network configurations (from 141 to 121 MHz in Mesh121 and from 141 to 122 MHz in Torus121 network configurations). This is due to the higher FBD value, which forces the use of more buffers and increases the consumption of logical and routing resources along the critical path. A point to note that the Fattree144 network configuration offered the best overall FPGA clock frequency performance. In Fig. 6 (b) we see the expected improvement of the clock frequency due the increase in network grid size to 144 nodes. Mesh144 showed an increase from 141 MHz to 218 MHz and Torus144 showed an increase from 141 MHz to 244 MHz. Here, the FBD parameter becomes the dominant factor influencing clock frequency. We also observed that setting FBD to 8 bits produced a higher clock frequency in Torus144 than in other architectures. The Fattree144 network offered better overall performance than the Mesh144 and Torus144. The summary of the major findings are as follows:

- In Mesh configurations with FDW and FBD of size 8 bits, increasing the VCs does not produce any performance gain in FPGA area and clock frequency utilization.

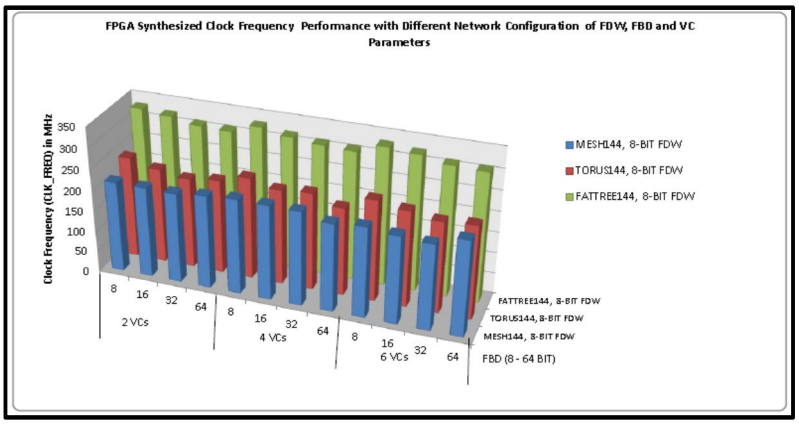
- A VC of the size 2 and FBD of size 64 bits produced the same area and clock frequency as a VC of the size 4 and FBD of size 64 bits.
- The FBD parameter significantly increases the FPGA area and degrades clock frequency performance.
- In both Mesh and Torus configurations, an FDW of size 8 bits, a VC of size 2 to 4, and an FBD of size 8 bits produced the best network configuration in terms of FPGA area and clock frequency.
- In a Fattree56 network configuration with FDW and FBD of the same size, increasing the VCs has no performance impact on the FPGA area and clock frequency.
- The Fattree144 network configuration with FDW of 8 bits, VC of 2 to 6, and FBD of 8 bits produced the best network configuration in terms of FPGA area and clock frequency.

(Table 2) Synthesis results of 96 different on-chip Interconnect network configurations with (FDW of 8 bits, VCs from 2 to 6, and FBD of 8 bits to 64 bits) on Virtex-7 FPGA device.

Flit Data Width (FDW)		8bits											
Number of Virtual Channels (VCs)		2VCs				4VCs				6VCs			
Flit Buffer Depth (FBD)		8	16	32	64	8	16	32	64	8	16	32	64
10X10 MESH (MESH100) NoC	PLUT (%LUTs)	12	14	18	24	12	14	18	24	12	15	18	26
	CLK_FREQ (MHZ)	285	267	253	243	285	269	253	243	285	265	251	236
11X11 MESH (MESH121) NoC	PLUT (%LUTs)	14	19	25	42	15	20	26	42	15	19	26	43
	CLK_FREQ (MHZ)	141	132	128	121	140	131	129	122	140	134	129	121
12X12 MESH (MEASH144) NoC	PLUT (%LUTs)	18	20	26	36	18	21	26	37	19	22	27	37
	CLK_FREQ (MHZ)	218	217	216	223	228	227	226	211	217	210	205	223
10X10 TORUS (TORUS100) NoC	PLUT (%LUTs)	13	16	20	27	14	16	20	28	14	17	21	27
	CLK_FREQ (MHZ)	259	263	276	248	258	262	276	248	258	261	273	239
11X11 TORUS (TORUS121) NoC	PLUT (%LUTs)	16	21	27	44	16	22	28	46	16	19	28	48
	CLK_FREQ (MHZ)	141	138	128	120	140	137	129	110	140	136	129	123
12X12 TORUS (TORUS144) NoC	PLUT (%LUTs)	20	24	27	40	21	24	28	40	21	25	29	41
	CLK_FREQ (MHZ)	244	228	217	225	244	228	233	211	243	231	219	224
FATTREE56 NoC	PLUT (%LUTs)	3	4	5	8	3	4	5	9	3	4	5	9
	CLK_FREQ (MHZ)	367	329	312	309	355	329	312	310	355	329	312	310
FATTREE144 NoC	PLUT (%LUTs)	10	12	15	23	10	12	15	23	10	13	15	23
	CLK_FREQ (MHZ)	331	324	312	310	331	319	312	309	331	325	312	310



(a) For Mesh121, Torus121, and Fattree144



(b) For Mesh144, Torus144, and Fattree144

(Figure 6) Shows the FPGA synthesized clock frequency performance with 8-bit FDW.

Table 3 and Figure 7 (a-b) show the experimental results of synthesizing ninety eight different network configurations using Mesh, Torus, and Fat-tree networks. Here the size of the FDW parameter was fixed at 16 bits and varied VC from 2 to 6 and FBD from 8 bits to 64 bits. We measured the resulting FPGA area (PLUT, %LUT) and clock frequency (CLK_FREQ) across different network

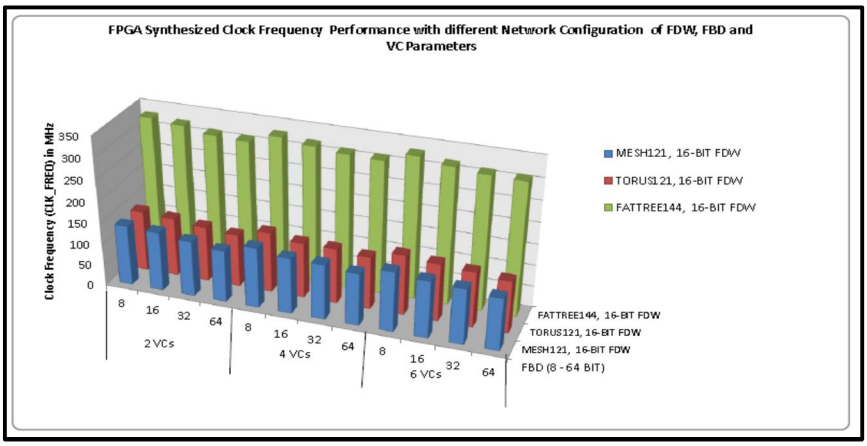
configurations using Xilinx ISE design tools.

In Fig. 7 (a) we expected an increase in clock frequency performance with the FDW parameter scaled to 16 bits. But, we observed that there is no any significant performance gain due to the FDW parameter. Again, the Fattree144 network configuration showed better performance than Torus121 and Mesh121, due to the effects of the FBD and VC parameters. We also observed that Torus121 and Mesh121 showed significant degradation in clock frequency. Increasing VC does increase clock frequency, but at the expense of a slight increase in area utilization. In fig 7 (b), we scaled the networks to grid size of 144 nodes. Again, Fattree144 outperformed Mesh144 and Torus144. While there is a slight improvement in both Mesh144 and Torus144 network when increasing FDW to 16 bits, Fattree144 again outperformance all others. From the conducted experiment we conclude that, due to changes in FDW, VC and FBD parameters, the Fattree144 architecture offered the best FPGA network synthesis in terms of area and clock frequency. The key findings are as follows:

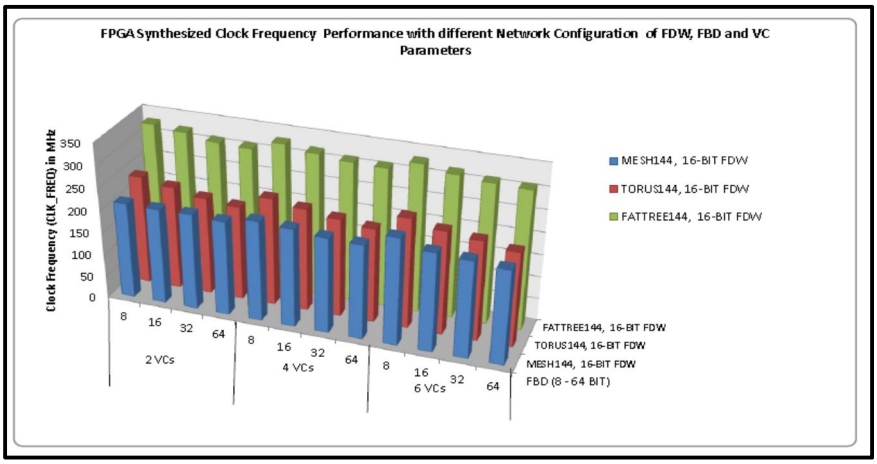
- For Mesh and Torus networks with FDW and FBD of 16 bits, increasing the VCs from 4 to 6 does not produce any performance gain in FPGA area and clock frequency.
- A VC of size 2 to 6, FDW of 16 bits, and FBD of 8 bits produced the same FPGA area and clock frequency.
- The FDW and FBD parameters have the greatest impact on FPGA area and clock frequency for a given network size.
- Fattree56 and Fattree144 with FDW of 16 bits, VC from 2 to 6, and FBD of 8 bits produced the best overall FPGA area and clock frequency.

(Table 3) Synthesis results for 96 different on-chip interconnect network configurations with (FDW of 16 bits, VCs from 2 to 6, and FBD of 8 bits to 64 bits) on Virtex-7 FPGA device.

Flit Data Width (FDW)		16bits											
Number of Virtual Channels (VCs)		2VCs				4VCs				6VCs			
Flit Buffer Depth (FBD)		8	16	32	64	8	16	32	64	8	16	32	64
10X10 MESH (MESH100) NoC	PLUT (%LUTs)	14	15	18	28	15	18	21	28	15	17	21	28
	CLK_FREQ (MHZ)	289	258	244	235	328	287	264	235	328	285	253	235
11X11 MESH (MESH121) NoC	PLUT (%LUTs)	17	23	28	45	17	24	28	45	18	25	29	45
	CLK_FREQ (MHZ)	141	138	130	121	141	130	129	122	140	133	129	122
12X12 MESH (MEASH144) NoC	PLUT (%LUTs)	22	25	29	40	22	25	29	40	22	25	29	40
	CLK_FREQ (MHZ)	215	214	215	211	224	220	213	211	228	220	216	210
10X10 TORUS (TORUS100) NoC	PLUT (%LUTs)	16	18	23	29	16	18	24	29	17	19	23	29
	CLK_FREQ (MHZ)	260	261	259	239	260	261	259	239	263	262	276	239
11X11 TORUS (TORUS121) NoC	PLUT (%LUTs)	19	23	30	47	19	25	31	47	19	26	31	47
	CLK_FREQ (MHZ)	141	137	129	123	141	130	129	123	141	134	129	122
12X12 TORUS (TORUS144) NoC	PLUT (%LUTs)	24	26	31	44	24	27	31	44	24	27	31	44
	CLK_FREQ (MHZ)	243	231	218	211	241	230	220	211	247	231	223	211
FATTREE56 NoC	PLUT (%LUTs)	4	4	5	9	4	4	5	9	4	5	6	9
	CLK_FREQ (MHZ)	357	329	312	309	357	329	312	310	357	329	312	310
FATTREE144 NoC	PLUT (%LUTs)	11	13	16	25	11	14	16	25	11	13	16	25
	CLK_FREQ (MHZ)	331	324	312	309	330	320	312	310	331	319	312	310



(a) For Mesh121, Torus121 and Fattree144



(b) For Mesh144, Torus144 and Fattree144

(Figure 7) Shows FPGA synthesized clock frequency performance with 16-bit FDW.

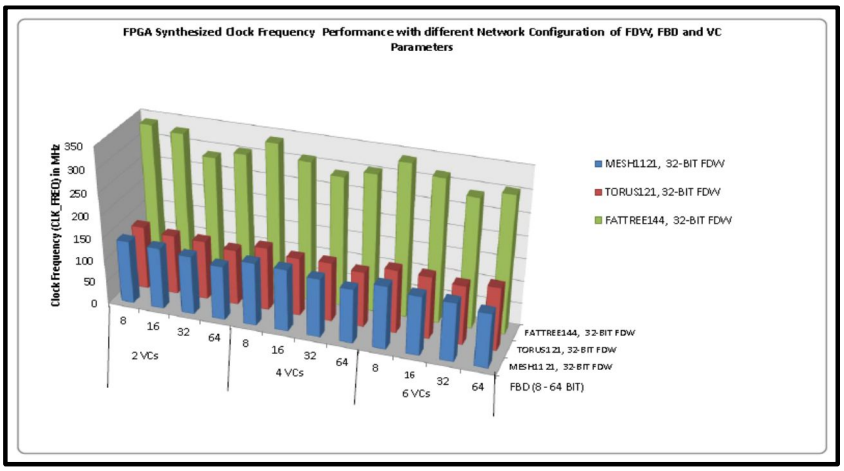
In table 4 and figure 8 (a-b) we increased the value of the FDW parameter to 32-bit and varied the values of VC from 2 to 6 and FBD from 8 to 64-bit. From Figure (a-b) due to FDW scaled to 32 bits, we expected an increase in the clock frequency performance. As observed, the Fattree144 network configurations

with 8-bit FBD and 2 to 6 VCs showed an increase in clock frequency to 340 MHz. This increase is due to the effect of FDW parameter scaled to 32-bit. Similar observation can be seen in the Table 4 for both Mesh100 and Torus100 network configurations with 2 to 6 VCs and 8-bit FBD, showed an increase in clock frequency of 320 MHz and 302 MHz respectively. This is also due to the effect of the FDW parameter scaled to 32-bit and FBD of size 8-bit respectively. The Fattree144 network configuration offered best FPGA synthesized clock frequency as compared to Mesh144 and Torus144 network configurations. Finally, from conducted experiment due to the effect of FDW parameter and FBD the Mesh and Torus network configurations showed a degradation in clock frequency (CLK_FREQ) performance. However, the Fattree network configurations showed a better performance and efficient utilization of FPGA resources. The key findings are:

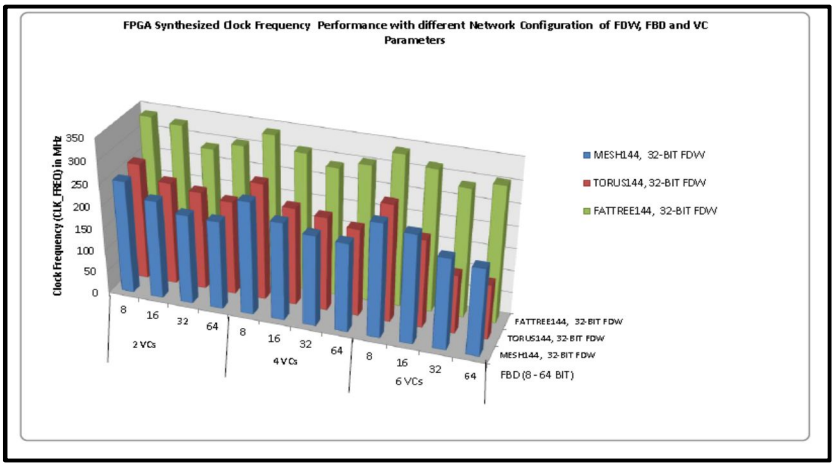
- The Mesh, Torus and Fat-tree networks with 32-bit FDW, 2 to 6 VC and 8-bit FBD offer the best utilization of the FPGA area and clock frequency.
- Fattree144 network configuration with 32-bit FDW, 2 VCs and 8-bit FBD showed the highest synthesized FPGA clock frequency.

(Table 4) Synthesis results of 96 different on-chip Interconnect network configurations with (FDW of 32 bits, VCs from 2 to 6, and FBD of 8 bits to 64 bits) on the Virtex-7 FPGA device.

Flit Data Width (FDW)		32bits											
Number of Virtual Channels (VCs)		2VCs				4VCs				6VCs			
Flit Buffer Depth (FBD)		8	16	32	64	8	16	32	64	8	16	32	64
10X10 MESH (MESH100) NoC	PLUT (%LUTs)	18	20	25	32	18	21	25	32	18	21	24	32
	CLK_FREQ (MHZ)	320	201	267	236	320	202	267	236	320	301	245	236
11X11 MESH (MESH121) NoC	PLUT (%LUTs)	21	26	33	48	21	27	33	48	21	27	33	48
	CLK_FREQ (MHZ)	141	137	131	121	141	139	131	121	141	132	131	121
12X12 MESH (MEASH144) NoC	PLUT (%LUTs)	27	30	33	45	27	29	32	45	28	31	35	46
	CLK_FREQ (MHZ)	254	221	201	199	254	220	204	199	255	243	204	195
10X10 TORUS (TORUS100) NoC	PLUT (%LUTs)	20	24	28	34	20	24	29	35	20	24	28	35
	CLK_FREQ (MHZ)	302	287	276	263	302	285	276	239	302	286	245	239
11X11 TORUS (TORUS121) NoC	PLUT (%LUTs)	23	27	35	53	23	27	35	54	24	29	31	56
	CLK_FREQ (MHZ)	141	133	132	124	141	130	132	124	141	139	132	141
12X12 TORUS (TORUS144) NoC	PLUT (%LUTs)	29	33	36	49	29	34	49	54	28	35	44	66
	CLK_FREQ (MHZ)	263	231	221	211	263	220	211	197	263	198	132	124
FATTREE56 NoC	PLUT (%LUTs)	5	6	5	11	5	6	7	11	5	6	7	11
	CLK_FREQ (MHZ)	342	333	342	306	342	333	301	306	342	333	306	306
FATTREE144 NoC	PLUT (%LUTs)	14	16	19	29	14	16	19	29	14	17	19	29
	CLK_FREQ (MHZ)	340	331	289	306	340	311	289	306	340	319	289	306



(a) For Mesh121, Torus121 and Fattree144



(b) For Mesh144, Torus144 and Fattree144

(Figure 8) Shows the FPGA synthesized clock frequency performance with 32-bit FDW.

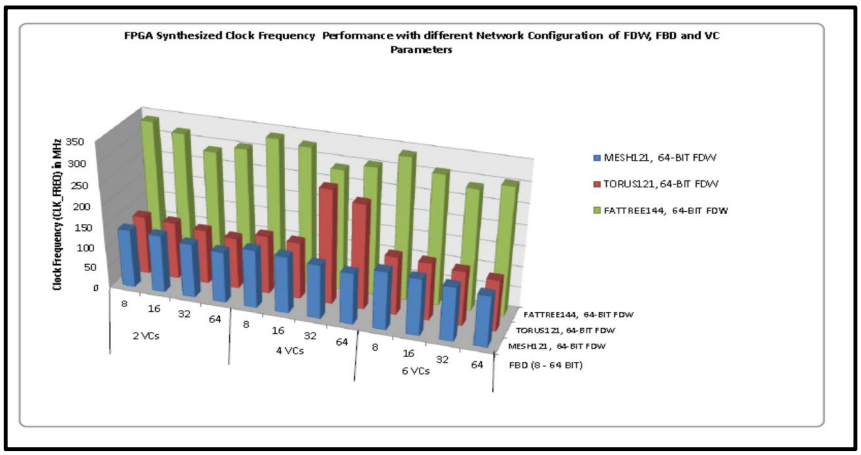
In table 5 and figure 9 (a-b) we fixed the size of the FDW parameter for 64-bit and varied the VCs from 2 to 6 and FBD from 8 to 64-bit. An interesting observation here is in figure 9 (a) shown in the Torus121 network configuration with 64-bit FDW, 4 VCs and 32 to 64-bit FBD. An increase in clock frequency to

275 MHz with 32−bit FBD and 252 MHz with 64−bit FBD respectively. However, in Figure 9 (b) we observed that the Torus144 network configuration showed an interesting result with an increase in clock frequency from 263 MHz to 300 MHz. This is due to the effects of FDW of size 64−bit, VC of the size 4 and FBD of sizes 8 to 16−bit respectively. However, we observed that, the clock frequency degraded significantly with 2 VCs. This clearly shows the effect of fewer number of VCs in large scaled Torus144 network configuration. In conclusion, from the conducted experiment an interesting observation is that, the trend in an FPGA area and clock frequency performance is the same for all the three network architectures. Meaning that, an FDW of size 64−bit, VC of size 2 to 6 and FBD of size 8 to 16−bit produces the best FPGA clock frequency in large scaled NoC architectures. The major findings are:

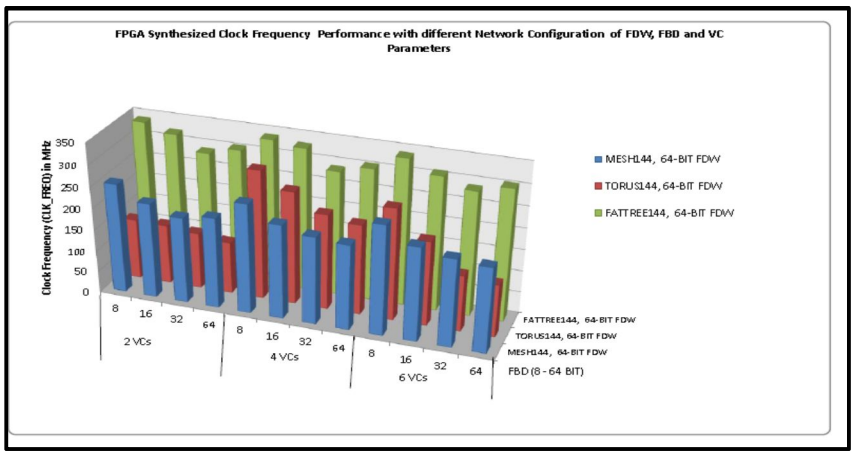
- A VC of the size 2 and FBD of size 64−bit produced the same area and clock frequency with a VC of size 4 to 6 and FBD of size 64−bit. FDW of size 64 bits, VCs of size 2 to 4 and FBD of size 8 to 16 produced efficient FPGA area and clock frequency for a given network size in the three architectures.

(Table 5) Synthesis results of 96 different on-chip Interconnect network configurations with (FDW of 64 bits, VCs from 2 to 6, and FBD of 8 bits to 64 bits) on the Virtex-7 FPGA device.

Flit Data Width (FDW)		64bits											
Number of Virtual Channels (VCs)		2VCs				4VCs				6VCs			
Flit Buffer Depth (FBD)		8	16	32	64	8	16	32	64	8	16	32	64
10X10 MESH (MESH100) NoC	PLUT (%LUTs)	26	28	31	38	26	29	34	43	26	29	33	42
	CLK_FREQ (MHZ)	323	293	221	215	323	298	267	247	323	294	253	236
11X11 MESH (MESH121) NoC	PLUT (%LUTs)	30	36	42	59	30	37	43	60	30	38	43	51
	CLK_FREQ (MHZ)	140	138	130	123	140	137	130	123	140	136	130	123
12X12 MESH (MEASH144) NoC	PLUT (%LUTs)	39	41	44	58	40	41	45	59	40	42	45	59
	CLK_FREQ (MHZ)	255	221	200	211	255	219	204	198	255	218	204	198
10X10 TORUS (TORUS100) NoC	PLUT (%LUTs)	29	34	38	48	29	34	40	48	30	32	35	45
	CLK_FREQ (MHZ)	302	285	250	252	302	286	275	252	302	295	245	239
11X11 TORUS (TORUS121) NoC	PLUT (%LUTs)	33	38	45	65	34	39	49	58	34	39	46	66
	CLK_FREQ (MHZ)	140	137	130	123	140	137	275	252	140	138	131	123
12X12 TORUS (TORUS144) NoC	PLUT (%LUTs)	41	46	55	59	40	44	47	63	40	48	56	80
	CLK_FREQ (MHZ)	140	138	131	121	300	263	222	210	260	197	131	123
FATTREE56 NoC	PLUT (%LUTs)	7	8	9	14	7	8	9	14	7	8	9	14
	CLK_FREQ (MHZ)	342	333	306	306	342	333	306	306	342	333	306	360
FATTREE144 NoC	PLUT (%LUTs)	19	21	25	37	20	22	25	37	20	23	25	37
	CLK_FREQ (MHZ)	340	322	289	306	340	331	289	306	340	311	289	306



(a) For Mesh121, Torus121 and Fattree144



(b) For Mesh144, Torus144 and Fattree144

(Figure 9) Shows FPGA synthesized clock frequency performance with 64-bit FDW.

3. FPGA network cost performance evaluation

The dissertation considers the FPGA silicon area (PLUT) as a metric for the evaluation of the network cost [1], [82], [147]. Considering tables 2-5. Figures 10-13 show the percentage of FPGA area (PLUT) utilization across different network configurations. From Figures 10-13, we observed that, the FPGA area

usage increases with the increase in the VC, FBD and FDW parameters. Correspondingly, the FDW and FBD parameters considerably increase the percentage of FPGA area (PLUT). As observed, the Fat-tree network configuration outperform all other networks in terms of efficient FPGA area utilization. This is mainly due to the fact that in fat-tree network resources increases in stages closer to the root node and routing resources are shared, fewer levels of logic are needed to implement the network. Additionally, the FDW parameter offers higher performance by allowing packets to make more progress under heavy loads. Furthermore, enabling the virtual link parameter in Fattree144 require a small increase in hardware. This significantly reduces the buffering and logic requirements.

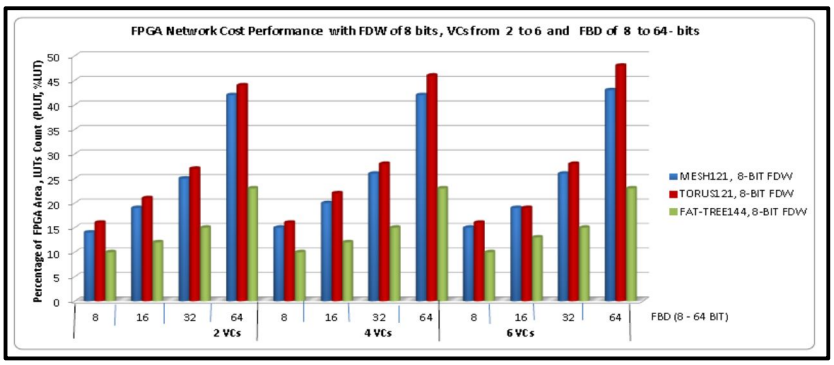
Figure 10 (a-b) shows the network cost in terms of percentage of LUT counts for Mesh, Torus and Fat-tree networks. Here we fixed the size of the FDW parameter for 8-bit and varied the values of VC from 2 to 6 and FBD from 8 to 64-bit. As observed, the Fat-tree144 network configuration outperforms Torus144 and Mesh144 in terms of efficient FPGA area (PLUT) utilization. The flit buffer depth (FBD) parameter is the dominant factor for increasing the percentage of LUTs consumption in both Torus144 and Mesh144 network configurations. In figure 10 (b), the Torus144 network used too much of the FPGA resources as compared to Mesh144 this is due the path diversity of the Torus network more logics and routing resources are needed to provide efficient

flow of packets from source to the destination and the effect of the FBD parameter of size 8 to 64 bits.

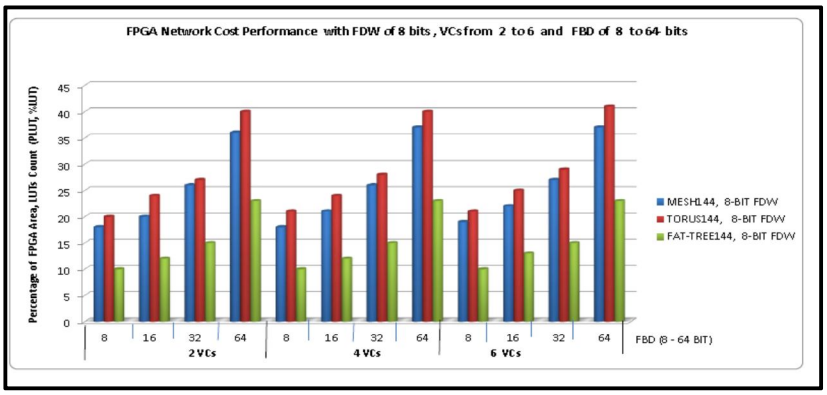
Fig. 11 (a-b) shows the percentage of LUT utilization of the three network architectures. Here, we fixed the size of the FDW parameter for 16-bit and varied the values of VC from 2 to 6 and FBD from 8 to 64-bit. As observed, the Fat-tree144 network is the best in terms of network cost it uses fewer percentage of LUT counts across different network configurations as compared to the Mesh144 and Torus144 network configurations. The FBD parameter becomes the major factor in increasing the FPGA area consumption. However, both Torus144 and Mesh144 networks with the VC of the size 4 and FBD of sizes 8 to 16-bit are showing good FPGA area utilization.

Fig. 12 (a-b) shows the network cost comparison for Mesh, Torus and Fat-tree networks. In the conducted experiment, we fixed the size of the FDW parameter for 32-bit and varied the values of VC from 2 to 6 and FBD from 8 to 64-bit. As observed, the Fat-tree144 outperforms both the Mesh144 and Torus144 network configurations. However, the FDW and VC parameters are showing significant impact in both Torus144 and Mesh144 networks. This is illustrated in figure 12 (b), a VC of size 2 to 6 and FBD of sizes 8 to 16-bit produced efficient utilization of the FPGA area in both Mesh and Torus networks. This shows a significant reduction of the FPGA area cost.

Fig. 13 (a-b) shows the network cost performance comparison between the three architectures. We fixed the size of the FDW parameter for 64-bit and varied the values of the VC from 2 to 6 and FBD from 8 to 64-bit. As observed, the Fat-tree144 network outperforms the Torus144 and Mesh144 networks. This comes with no surprise because in Fat-tree network communication can be scaled freely from the number of processors; as a result, considerable hardware resources can be saved, fewer levels of logic are needed for the implementation [73]. Additionally, the pipeline router core parameter adds pipeline stages right after the router allocation unit. This can offer 1.5 to 2x clock frequency improvement. The pipeline router allocator is also useful in increasing frequency in networks containing many VCs as in Mesh144 and Torus144 networks. We conclude that, the Fat-tree144 network outperforms the Mesh144 and Torus144 networks in terms of the FPGA network cost. These findings also suggest that, large scaled Fat-tree networks reduce the FPGA hardware cost and are compliant with the current Xilinx FPGA devices.

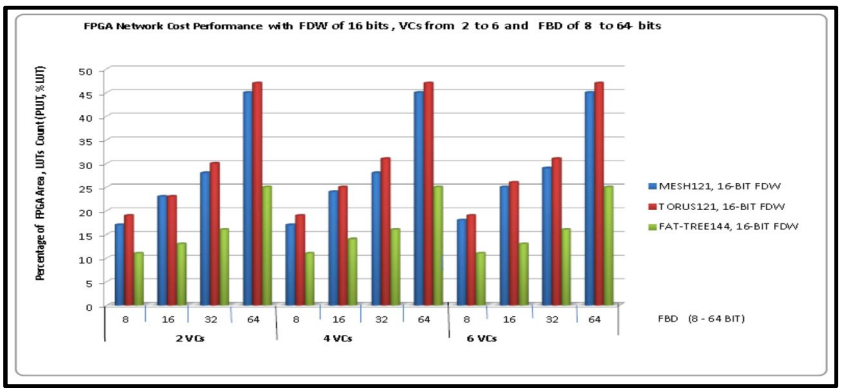


(a) For Mesh121, Torus121, and Fat-tree144 networks

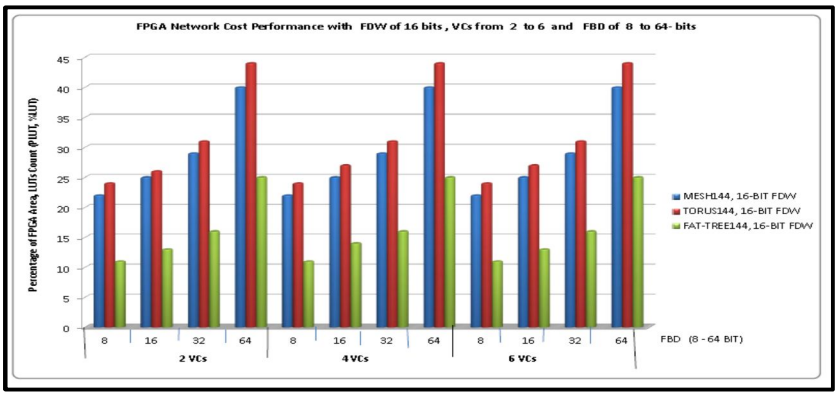


(b) For Mesh144, Torus144, and Fat-tree144 networks

(Figure 10) Illustration of the FPGA Network Cost Performance with 8-bit FDW

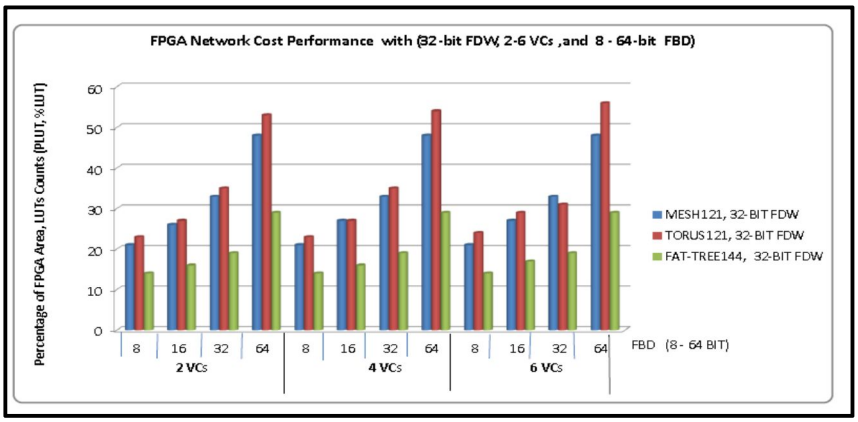


(a) For Mesh121, Torus121, and Fat-tree144 networks

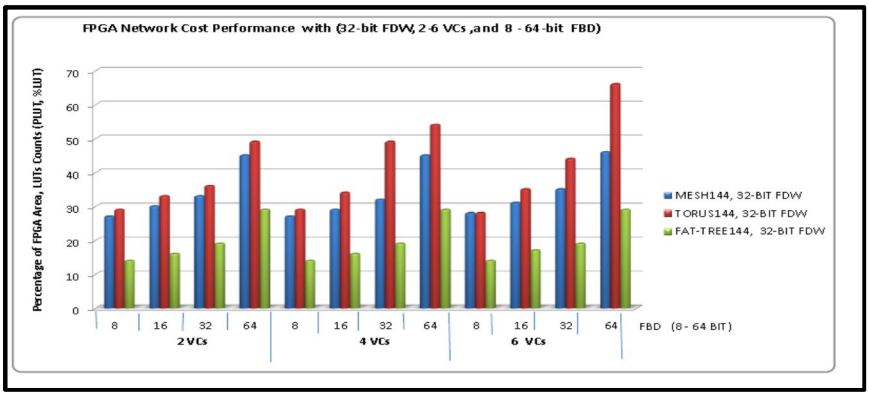


(b) For Mesh144, Torus144, and Fat-tree144 networks

(Figure 11) Illustration of the FPGA Network Cost Performance with 16-bit FDW

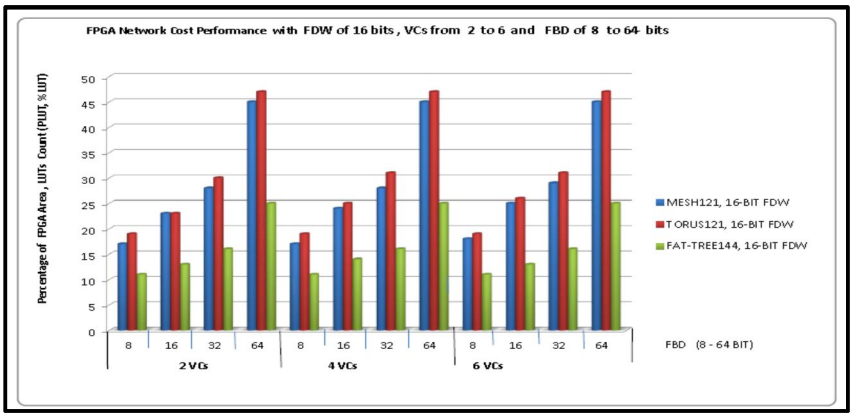


(a) For Mesh121, Torus121, and Fat-tree144 networks

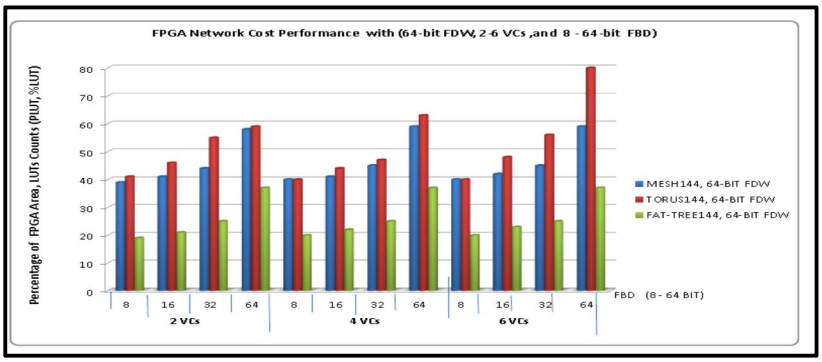


(b) For Mesh144, Torus144, and Fat-tree144 networks

(Figure 12) Illustration of the FPGA Network Cost Performance with 32-bit FDW



(a) For Mesh121, Torus121, and Fat-tree144 networks



(b)For Mesh144, Torus144, and Fat-tree144 networks

(Figure 13) Illustration of the FPGA Network Cost Performance with 64-bit FDW

4. Performance analysis of the critical path delay

Herein, the dissertation presents the experimental results of critical path delay analysis. An essential metric for NoC performance evaluation using FPGA is the speed measured by the critical path delay [78-80], [147]. The total critical path delay is defined as the total delay due to the logic and the routing delays. In the conducted experiment using the Xilinx ISE design tools we carried out timing analysis using the Xilinx timing analyzer to compare the critical path delay of the three interconnect architectures. From the findings over 80% of the total critical path delays are from routing and the remaining less than 20% are from logic. The results of the comparison are shown in Tables 6-7 and Figures. 14-15. The Tables 6-7 show the critical path delay across network configurations. Observing the tables, it's clear that increasing the FDW, FBD and VC parameters slightly increase the critical path delay. The critical path delays in Fat-tree network

configurations produced the best (minimum) critical path delays this is due to the fact that, there is a decrease in the number of logic elements since logic elements are shared with Fat-tree networks. Accordingly, there will be fewer logic levels to implement the network. From table 6 we fixed the size of the FDW parameter for 32-bit and varied the values of VC from 2 to 6 and FBD from 8 to 64-bit and observed the critical path delay across 72 network configurations. As shown in Table 6, the Fat-tree144 network with grid size of 144 nodes outperforms the Mesh144 and Torus144 networks by producing the minimal critical path delays across different network configurations. Another interesting observation is with the Mesh144 and Torus144 networks produced minimal critical path delays as compared to Mesh121 and Torus121 networks. This is mainly attributed to the on-chip cores (die) structure looks more like a perfect square grid that matches very closely with the actual on-chip core structure as compared to the Mesh121 and Torus121 networks. This agrees with the findings by Lee and Shannon [80] and Genko et. al [95].

In table 7, we fixed the size of the FDW parameter for 64-bit and varied the values of VC from 2 to 6 units and FBD from 8 to 64 bits across 72 network configurations. As observed increasing the FDW to 64-bit has no significant impact on the critical path delay. As can be seen, the Fat-tree network produced the minimum critical path delays across different network configurations. An interesting observation is in Torus121 (64, 4, 32-64) and Torus (64, 4, 32-64)

network configurations produced the minimum critical path delay. This shows that, at FDW of size 64-bit, VC of the size 4 and FBD of sizes 32 to 64-bit produced the minimum critical path delay in the larger scaled Torus network.

(Table 6) The critical path delay of 78 different NoC configurations with (FDW of 32 bits, VCs from 2 to 6, and FBD of 8 bits to 64 bits) on the Virtex-7 FPGA device.

CRITICAL PATH DELAY													
Flit Data Width (FDW)		32bits											
Number of Virtual Channels (VCs)		2VCs				4VCs				6VCs			
Flit Buffer Depth (FBD)		8	16	32	64	8	16	32	64	8	16	32	64
11X11 MESH (MESH121) NoC	Delay (ns)	7.117	7.231	7.650	8.243	7.117	7.211	7.651	8.238	7.117	7.213	7.656	8.243
12X12 MESH (MEASH144) NoC	Delay (ns)	3.932	3.991	4.980	5.033	3.932	3.994	4.907	5.033	3.920	3.997	4.907	5.117
11X11 TORUS (TORUS121) NoC	Delay (ns)	7.088	7.241	7.582	8.084	7.088	7.242	7.582	8.085	7.093	6.994	7.587	7.088
12X12 TORUS (TORUS144) NoC	Delay (ns)	3.808	4.012	4.524	4.748	3.808	4.013	4.749	4.749	3.808	4.213	7.594	8.090
FATTREE56 NoC	Delay (ns)	2.922	2.999	3.267	3.267	2.922	2.999	3.320	3.267	2.922	2.999	3.272	3.267
FATTREE144 NoC	Delay (ns)	2.938	3.124	3.457	3.267	2.938	3.132	3.457	3.267	2.938	3.132	3.457	3.267

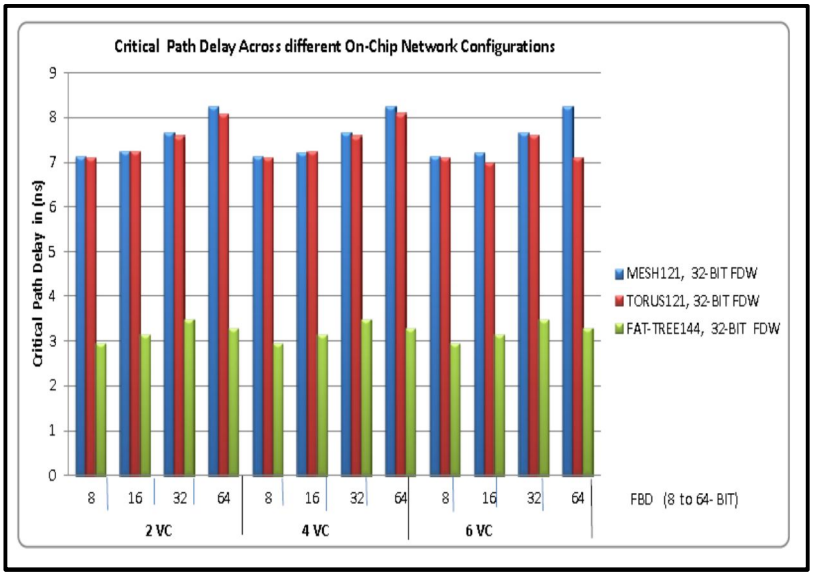
(Table 7) The critical path delay of 78 different NoC configurations with (FDW of 64 bits, VCs from 2 to 6, and FBD of 8 bits to 64 bits) on Virtex-7 FPGA device.

CRITICAL PATH DELAY													
Flit Data Width (FDW)		64bits											
Number of Virtual Channels (VCs)		2VCs				4VCs				6VCs			
Flit Buffer Depth (FBD)		8	16	32	64	8	16	32	64	8	16	32	64
11X11 MESH (MESH121) NoC	Delay (ns)	7.146	7.213	7.690	8.133	7.144	7.312	7.695	8.134	7.150	7.391	7.701	8.132
12X12 MESH (MEASH144) NoC	Delay (ns)	3.925	4.211	4.736	4.994	3.925	4.217	4.908	5.045	3.925	4.317	4.908	5.045
11X11 TORUS (TORUS121) NoC	Delay (ns)	7.123	7.289	7.690	8.114	7.127	7.287	3.640	3.966	7.132	7.299	7.633	8.124
12X12 TORUS (TORUS144) NoC	Delay (ns)	7.137	6.993	7.626	5.133	7.137	6.245	4.502	4.760	3.851	6.453	7.638	8.124
FATTREE56 NoC	Delay (ns)	2.922	2.991	3.277	3.260	2.922	2.997	3.272	3.267	2.922	2.998	3.272	3.267
FATTREE144 NoC	Delay (ns)	2.938	3.012	3.457	3.267	2.938	3.021	3.457	3.267	2.938	3.022	3.457	3.267

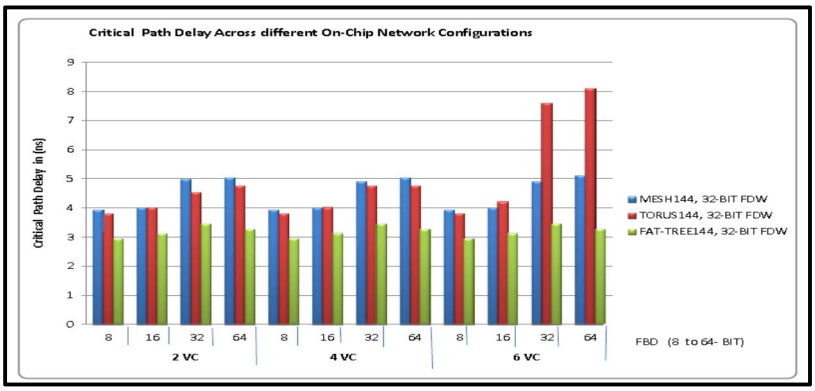
Figure 14 (a-b) shows the critical path delay for different network configuration of Mesh, Torus and Fat-tree networks. We observed that, the use of large FBD and FDW produces good performance results, but with higher critical path delays. From Figure 14 (a) the Mesh121 and Torus121 networks showed very poor performance in terms of critical path delays. This is due to the fact that, more logic is used on the critical path and the logic grows exponentially with the increase in the FDW, VC and FBD parameters. However, the Fat-tree144 network configuration produced the minimum critical path delay of 2.938 nanoseconds. In Figure 14 (b), we scaled the three architectures to grid size of 144 nodes and compare the performances. As observed, the Mesh144 and Torus144 networks showed an interesting result by reducing the critical path delay this reduction is very significant as it enhances the network performance. This is attributed to the increase in FBD and FDW parameters.

In Figure 15 (a-b), we increased the size of the FDW parameter to 64-bit and observe the impact of the critical path delay across different network configurations. As shown in figure 15 (a) both Mesh and Torus networks performed very poorly and produced higher critical path delays. This is due to the effect of FDW parameter scaled to 64-bit. The higher the critical path delay significantly degrades the clock frequency performance. Another interesting observation is in Torus121 network configuration with 64-bit FDW, 4 VC and 32

to 64 bits FBD produced the minimal critical path delay of 3.64 nanoseconds as compared to 7.695 nanoseconds of Mesh121 network. Another interesting observation is in figure 15 (b) the Mesh144 network showed significant reduction in the critical path delay as compared to Torus144. This is due to the effect of FDW and FBD parameters and the mesh size (grid) structure. As we mentioned earlier, the mesh size (grid) structure of the interconnection affects the area, and critical path delay this agrees with the findings of Abdelfattah and Betz [1], Lee and Shannon [80], Coll et. al [73] and Genko et. al [95]. Finally from the conducted experiment, it is clear that the Fat-tree network proved to be efficient in terms of critical path delay due to the influence of FDW, FBD and VC parameters. Hence the architecture is suitable for large and complex NoCs.

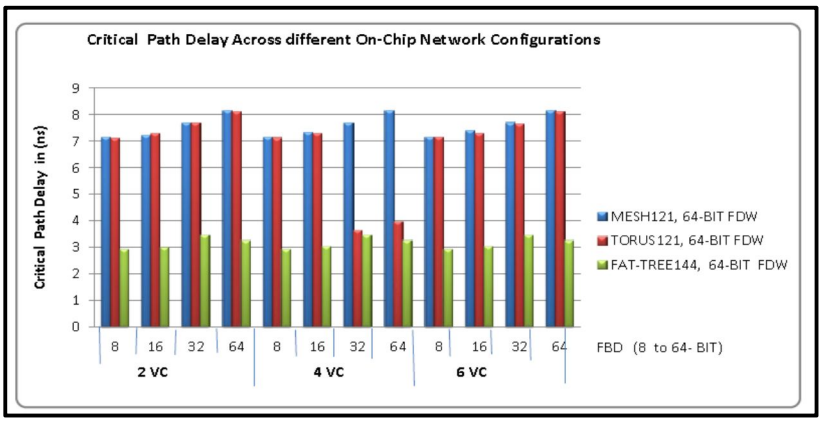


(a) For Mesh121, Torus121 and Fattree144 networks

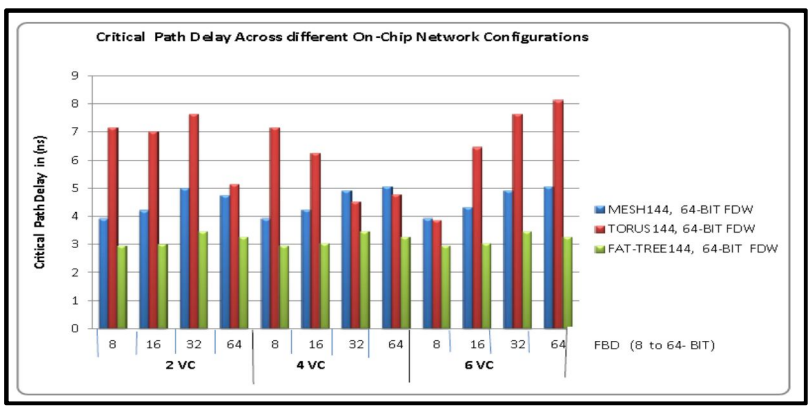


(b) For Mesh144, Torus144 and Fattree144 networks

(Figure 14) Illustration of the critical path delay with 32-bit FDW



(a) For Mesh121, Torus121 and Fattree144 networks



(b) For Mesh144, Torus144 and Fattree144 networks

(Figure 15) Illustration of the critical path delay with 64-bit FDW.

C. Networks-on-Chip Simulation Results

1. Simulation environment

The dissertation introduces the cycle-accurate simulation methodology used to evaluate the performance of the three interconnection architectures used in this study, namely, the Mesh, Torus and Fat-tree networks. The Network performance results are collected through a detailed and flexible cycle-accurate simulation system [77]. In the simulation system, each simulation has three phases, namely, warmup phase, measurement phase and drain phase. The length of the warmup and measurement phases is a multiple of the sample period defined by the sample_period parameter in the network configuration file. After the warmup period have passed the simulator reset all the simulation statistics and measurement phase begins and statistical data begin to be reported after each sample period. Once the measurement periods have passed, all the measurement packets drain from the network before final latency and throughput results are reported [77]. The dissertation used the following parameter configurations.

1. A packet size of ten flits (10 flits / Packet, packet_size = 10) for all network configurations.
2. We set the virtual channel buffer size (depth) for sixteen flits (vc_buffer_size = 16 flits).
3. We set the Injection rate from 0.1 to 1.0 in step of 0.05 (i.e. 0.1, 0.15, 0.2, 0.25,…, 1.0) flits/cycle/node. (Packet Injection Interval rate of

0.05 flits/cycle/node).

4. The Routing functions: Dimension order routing (dor) for Mesh and Torus networks and Nearest Common Ancestor (nca) for Fattree network.
5. The Traffic Patterns used: Nearest Neighbor, Uniform, Tornado and Random Permutation.
6. Number of virtual channels per physical channels used: (2 to 6 varied).
7. Injection Rate is flits/cycle (injection_rate_uses_flits = 1).
8. Injection Process Bernoulli (injection_process = Bernoulli).

2. Synthetic workload and traffic pattern

The network throughput and latency are greatly affected by the traffic pattern. The workload model for NoC is basically defined by the three parameters, namely, traffic pattern, packet injection rate, and packet length. The traffic pattern indicates the destination for the next packet at each network node. The most commonly used traffic pattern is the uniform traffic. In this traffic pattern, the probability of node k sending a message to node t is the same for all k and t, $k \neq t$. In synthetic workloads, the injection rate is usually the same for all the nodes. In most cases, each node is chosen to generate packets according to an exponential distribution [4], [69]. Table 8 shows the four traffic patterns and their description as used in this dissertation. Two benign traffic (Nearest Neighbor and Uniform) and two adversarial (Tornado and Random Permutation) traffic patterns. The Nearest Neighbor and Uniform traffic patterns are termed as benign traffic pattern, meaning that they naturally load balance the network and hence give good throughput with simple routing function. The Tornado and

Random Permutation traffic patterns are adversarial patterns, they are hard traffics and cause load imbalance in the network. To describe the traffic patterns as used in this dissertation, we let T_j, L_j denotes the j^{th} bit of the source, destination address and T_n, L_n denotes the n^{th} radix−Q digit of the source, destination address [4], [69].

(Table 8) Traffic Patterns used in this dissertation.

	Traffic Pattern	Explanation
1.	Uniform	Source nodes send an equal amount of traffics to each destination.
2.	Nearest Neighbor	$L_n = T_n + 1 \bmod Q$
3.	Tornado	$L_n = T_n + \left\lceil \frac{Q}{2} \right\rceil - 1 \bmod Q$
4.	Random Permutation	In this traffic pattern, a fixed permutation pattern is chosen uniformly at random from the set of all permutations. The perm−seed parameter is used to generate this permutation. Therefore, randomly chosen the values for perm−seed parameter gives a random sampling of permutations. On the other hand, a fixed value of the perm − seed parameter allows the same permutation to be used several times [77].

3. Performance evaluation metrics

One of the must basic performance measures of any interconnection network is its latency versus offered load [1−2], [78−80]. The performance evaluation and comparison of the three architectures was performed with respect to the

following performance metrics:

1. *Latency*: This is defined as the time in clock cycles that passes between the emergence of a header flit injected into the network at the source node and the emergence of a tail flit arrival at the destination node [4]. If source and destination overheads are considered the latency can be further defined as: let k be a packet the overall packet latency L_k is defined as L_k = *Source node overhead + Transport latency + Destination node overhead.* Similarly, let assume α be the total number of packets reaching their destinations, the average packet latency is given as [79], [91].

$$L_{average} = \sum \frac{L_k}{\alpha} \quad where \ \ k = 1, \ldots, \alpha \tag{17}$$

2. *Throughput*: This is defined as the total number of flits received at the destination node in a clock cycle. Throughput can be further defined as [4], [79].

Throughput = {(Total Packets Received) x *(Packet Size)} / {(Connected IP Cores)* x *(Simulation Time)}* (18)

Where total packet received is the number of packets successfully transmitted and arrived at the destination node. Packet size is the number of flits in the packet. The connected IP cores are the number of IP cores engaged in the communication. The simulation time is the total time for the simulation in clock cycles.

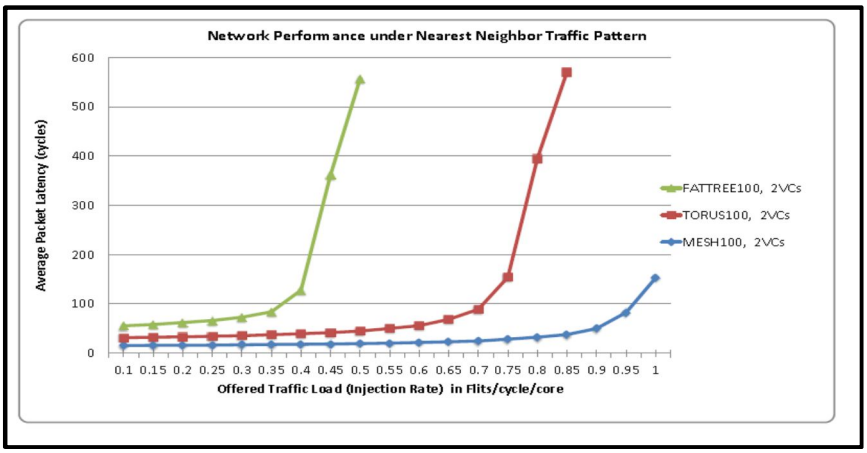
4. Comparison of the On-chip network latency and throughput

The dissertation compares the latency and throughput of the three network architectures described in Figure 1 (a-c) on the benign and adversarial traffic patterns. From the simulation, we observed that, the trends in network

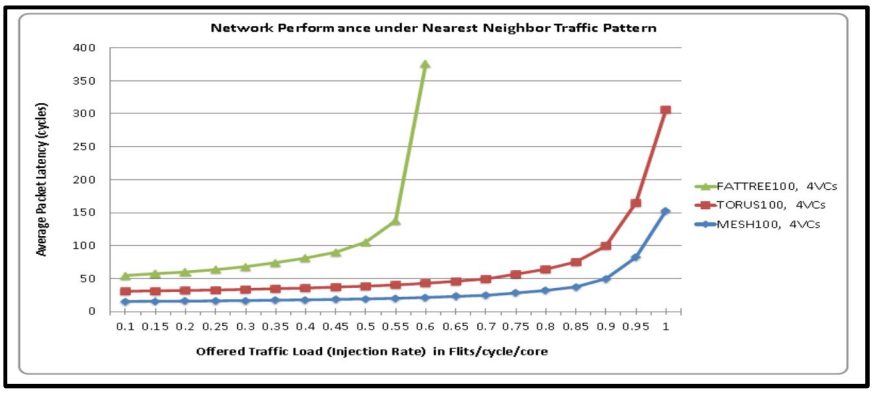
performance under the four traffic patterns with 121 and 144 network grid sizes are the same. Therefore, the network performance results for grid size of 100 nodes is provided.

Figure 16 (a-c) shows the network performance (load-delay curves) of Mesh100, Torus100 and Fattree100 network configurations under Nearest Neighbor traffic pattern and a varying number of virtual channels (VCs). As observed increasing the number of the VCs increases the network performance. The average packet latency depends on the number of virtual channels and the injection loads. In fig 16 (a) the Fattree100 network is the first to saturate at about 35% of the traffic loads. This is due to the effect of VC parameter more VCs are needed to reduce the packet contention in the network. Similarly, when the number of connected cores increases significantly the Fattree100 network needs many logical resources for routing purposes. Additionally, the tree-based routing also contributed to the poor performance when one branch is blocked the entire tree is blocked, increasing contention considerably. However, the Mesh100 network configuration saturate at 70% of the traffic loads followed by the Torus100 network configuration with 95% of the traffic loads. This is due to the fact that the Nearest Neighbor traffic pattern suit well with both Torus and Mesh networks with fewer numbers of VCs. In figure 16 (b) we observed that the Fattree100 network showed an improvement in the saturation throughput with an increase of 10% (i.e., from 35% to 45%). This is due to the effect of increasing

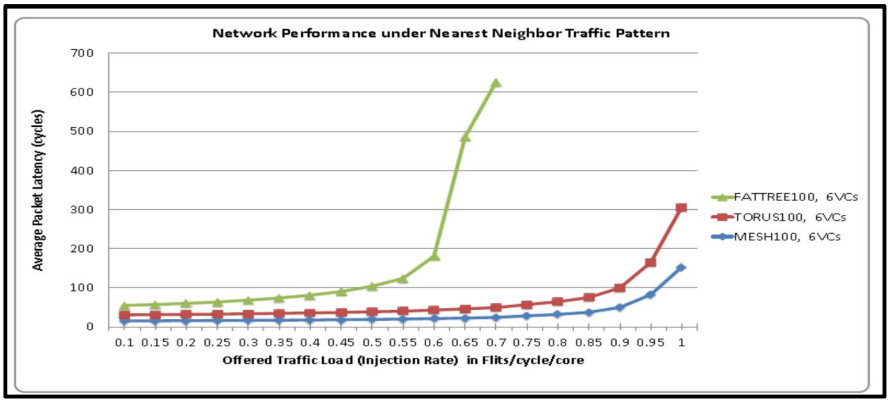
the size of the VC parameter. Furthermore, both Mesh100 and Torus100 network configurations saturated at 95% and 90% respectively. Showing a significant decrease in packet latency across all loads. In Figure 16 (c) the Fattree100 network configuration showed an increase in saturation throughput by saturating at 50% which shows an increase of 5%. This is due to the effect of the VC parameter and the nearest common ancestor routing. However, it is interesting to observe that both Mesh100 and Torus100 networks offered better performance by saturating at 95% and 90% respectively. This is attributed to the size of the VC parameter and dimension order routing in both Mesh100 and Torus100 networks. The dimension order routing exhibits lower latency than nearest common ancestor routing because path tends to be shorter, generating less traffic. Hence the network will not saturate quickly. Finally, from the conducted simulation under Nearest Neighbor traffic pattern the Mesh100 with 4 VCs outperforms Torus100 and Fattree100 network configurations and hence is the best network.



(a) With 2 VCs

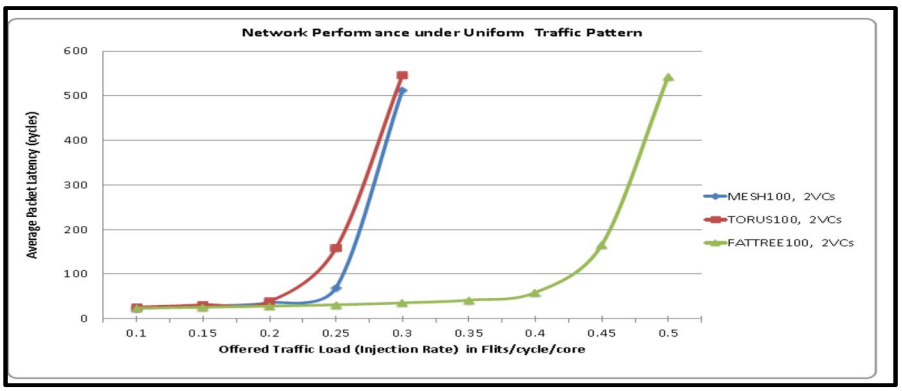


(b) With 4VCs

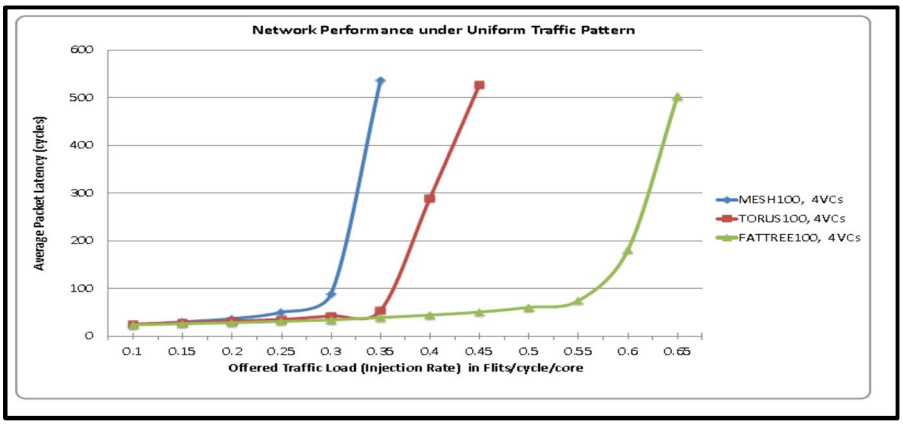


(c) With 6VCs

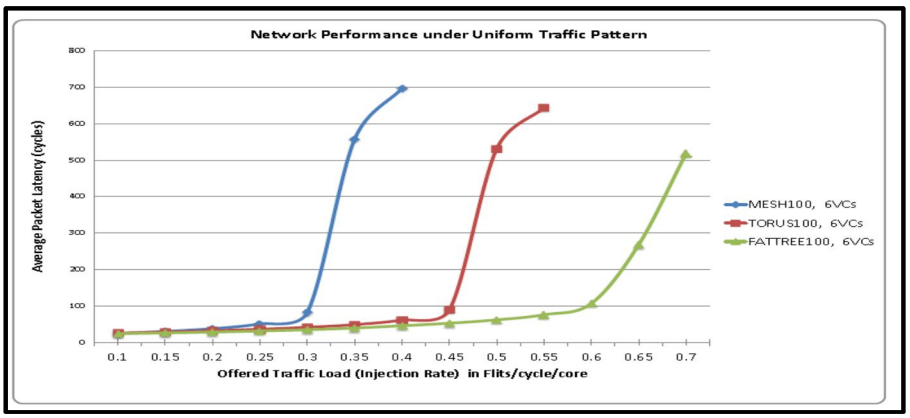
(Figure 16) Shows the network performance under nearest neighbor traffic pattern for Mesh100, Torus100 and Fattree100 network configurations.



(a) With 2 VCs



(b) With 4VCs

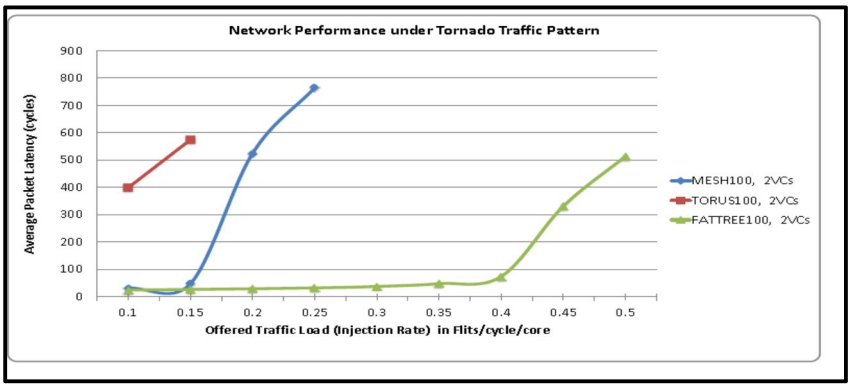


(b) With 6VCs

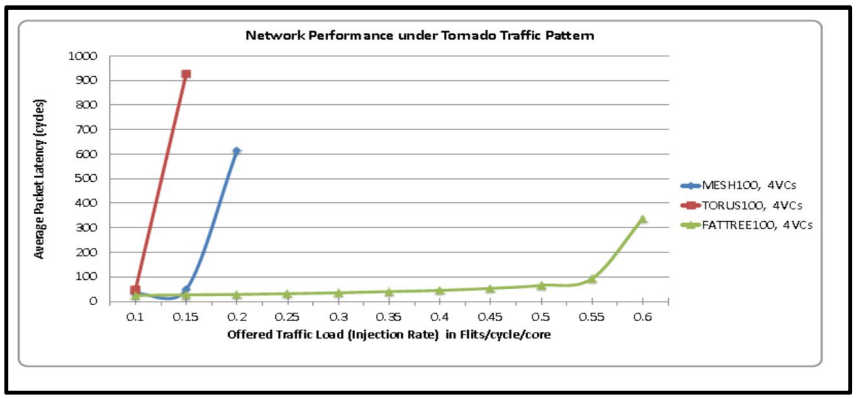
(Figure 17) Shows the network performance under uniform traffic pattern for Mesh100, Torus100 and Fattree100 network configurations.

Figure 17 (a-c) shows the network performance under the Uniform traffic pattern for the three architectures. From figure 17 (a), we observed that the Torus100 network configuration is the first to saturate at 20% of the traffic loads followed by Mesh100 at 25% respectively. This is attributed to the least number of VCs; more VCs are needed for efficient routing of packets in both Mesh100 and

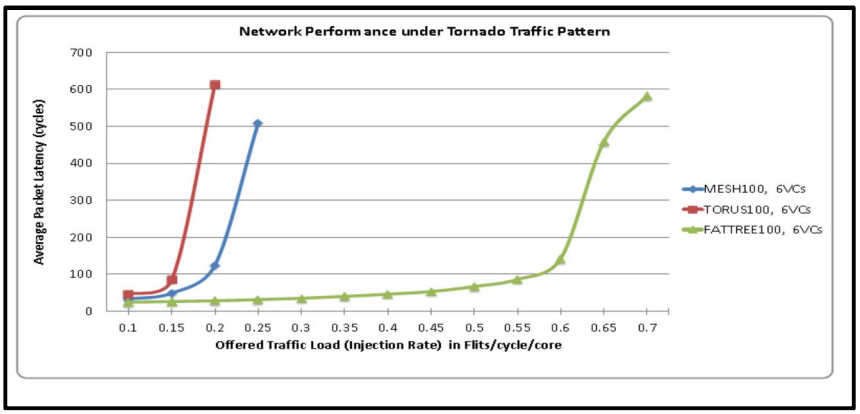
Torus100 network configurations and the traffic pattern used. The Fattree100 network configuration showed a fairly good performance by saturating at 40% of the traffic loads. This is attributed to the traffic pattern and the nearest common ancestor routing. In figure 17 (b) we observed that, there is an increase in the saturation throughput by all networks. The Mesh100 showed an increase of 5%, the Torus100 of 10% and Fattree100 of 15%. This reduction in latency and increase in the throughput is due to the effect of the number of VCs from two to four. Furthermore, in Figure 17 (c) we observed that for Mesh100 there is no decrease in the latency or the increase in the saturation throughput. This shows that, increasing the number of VCs to six has no any performance gain in Mesh100 network configuration. However, in Torus100 there is a decrease in latency and increase in the saturation throughput by 10% across different traffic injection loads. This is due to the number of VCs and the path diversity of Torus100 network. For Fattree100 the increase in the number of VCs to six has no any performance gain. From the conducted simulation, we conclude that, an Fattree100 under Uniform traffic pattern outperforms Mesh100 and Torus100 networks and hence is the best.



(a) With 2 VCs

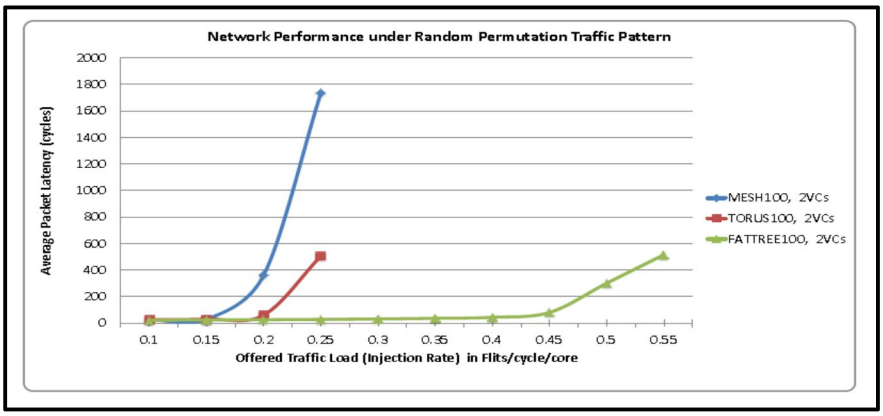


(b) With 4VCs

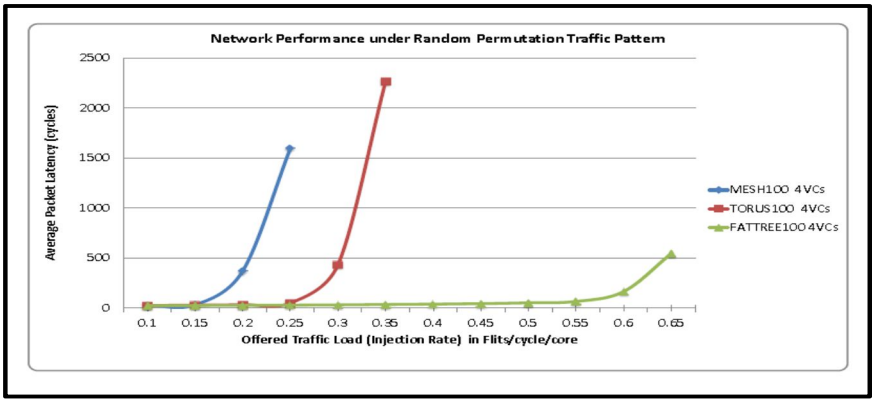


(c) With 6VCs

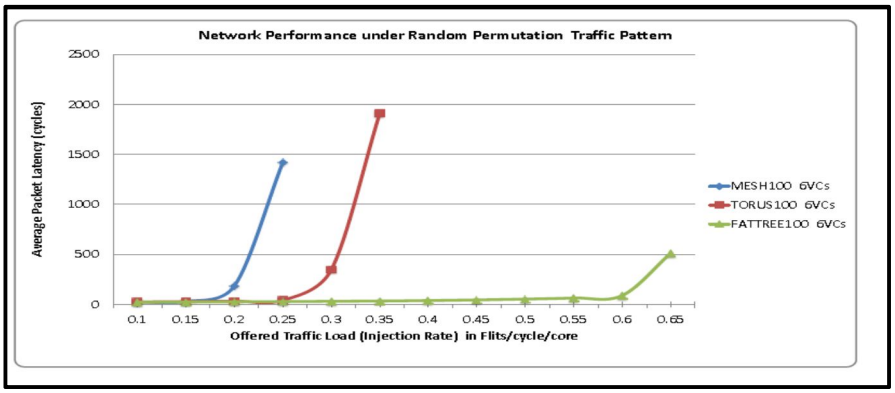
(Figure 18) Shows the network performance under tornado traffic pattern for Mesh100, Torus100 and Fattree100 network configurations.



(a) With 2 VCs



(b) With 4VCs



(c) With 6VCs

(Figure 19) Shows the network performance under random permutation traffic pattern for Mesh100, Torus100 and Fattree100 network configurations.

Figure 18 (a-c) shows the network performance under Tornado traffic pattern with varying number of VCs for large network grid size of 100 nodes for the three architectures. As observed, the performance of the three networks under Tornado traffic pattern is very poor, particularly in Mesh100 and Torus100 network configurations. Tornado traffic pattern does not favor Mesh-like architecture, even with six VCs. However, Fattree100 network showed a fairly good performance as compared to the Mesh100 and Torus100 with 2 to 6 VCs. With 2 VCs the Fattree100 saturate at 40% of traffic loads. However, with 4 VCs there is an increase in saturation throughput by 15% (i.e., saturation at 55%) additionally, with 6 VCs the Fattree100 network saturate at 55% of the traffic load. In conclusion, from the conducted simulation the Fattree100 network configuration under Tornado traffic pattern with 4 VCs offers the best network performance.

Figure 19 (a-c) shows the network performance under Random Permutation traffic pattern with varying number of VCs from 2 to 6. We found that the performance trend is similar to Figure 18 (a-c) under Tornado traffic pattern. There is also a significant increase in the network latency in both Mesh100 and Torus100 networks. Additionally, we observed that increasing the number of virtual channels does not make any performance impact in the two networks. This is mainly attributed to the Random permutation traffic pattern; the pattern does not favor Mesh and Torus networks. On the other hand, the Fattree100 network configuration showed a fairly good performance with 4 to 6 VCs saturating at 60%

of traffic loads. In conclusion, from the conducted simulation, we found that the Fattree100 under Random permutation traffic pattern with 4 to 6 VCs offered the best network performance.

5. Packet latency versus accepted network traffic

The dissertation presents the results of variation of packet latency versus the accepted network traffics for different configuration of Mesh100, Torus100 and Fattree100 networks.

Figure 20 (a–c) shows the average packet latency versus accepted traffic for the three architectures under Nearest Neighbor traffic pattern. As observed from the Figure 20 (a), the Mesh100 with 2 VCs outperforms Torus100 and Fattree100 networks with lower latencies and higher number of accepted packets. This is attributed to the nearest neighbor traffic pattern. Traffics are balanced in Mesh100 network with 2 VCs because each source node sends an equal traffic amount to each destination. This minimizes the packet contention and hence more packets are transmitted to the desired destinations. In figure 20 (b) we observed that there is an increase in the accepted network traffics and reduction in packet latency in both Torus100 and Fattree100 network configurations. The Fattree100 showed an increase in accepting network traffic from 0.5395 to 0.6377 an increase of 0.0982 (i.e., approx 10%). However, the Torus100 network configuration showed a significant increase in the accepted network traffic from 0.7816 to 0.93187 an increase of 0.15027 (approx. 15%). This is mainly attributed to the increase in the number of VCs from 2 to 4. The number of VCs helps to reduce the packet contention in the network. Therefore, more packets can

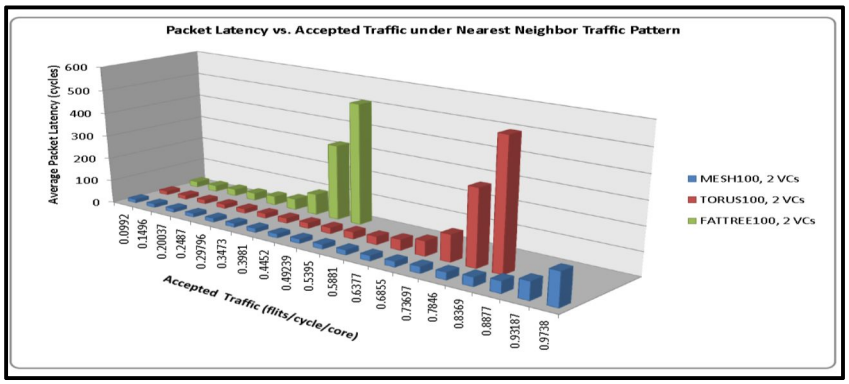
be transmitted to the required destinations. From the conducted simulation the Mesh100 with 2 to 4 VCs offered the lowest latency and highest amount of accepted network traffic.

Figure 21 (a-c) shows the average packet latency versus accepted traffic under Uniform distribution of traffics with 2 to 6 VCs. As expected, the Fattree100 network configuration with 2 to 6 VCs outperforms the Mesh100 and Torus100 networks. This comes with no surprise as uniform distribution of traffic works fairly well with tree-based routing. The poor performance exhibited by both Mesh100 and Torus100 network configurations is due to the uniform distribution of traffic from source to destinations and the dimension order routing. Congestions are too much propagated in the networks this resulted in packet contention and poor acceptance rate. However, in Figure 21 (c), the Torus100 network configurations with 6 VCs showed an improvement in packet acceptance rate from 0.35 to 0.45 (i.e., an increase of 10%). Therefore, addition of few virtual channels reduces contention and increases channel utilization. Conclusively, the Fattree100 network configuration offers the best network performance with higher number of accepted traffic and lower packet latencies across different injection loads.

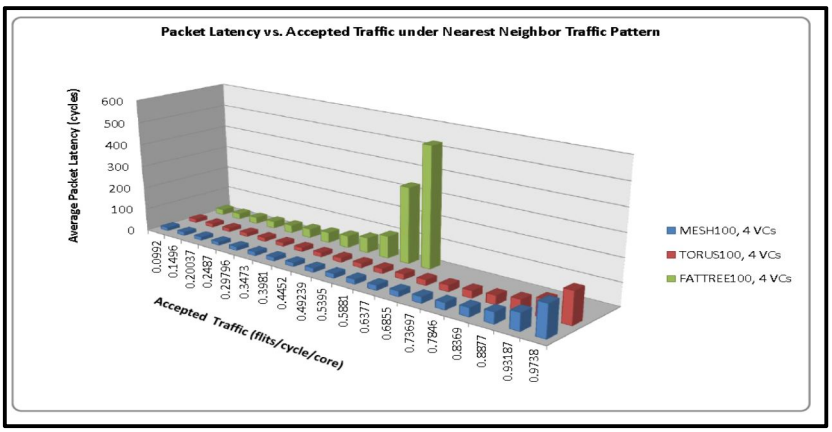
Figure 22 (a-c) shows the average packet latency versus accepted traffic under Tornado traffic pattern. The Tornado traffic pattern is a hard traffic pattern because the traffic cause load imbalance in the network. The Mesh-like structures (i.e. Mesh and Torus) perform very poorly under Tornado traffic pattern irrespective of the number of virtual channels. This is attributed to the

dimension order routing (dor) in both Mesh100 and Torus100 networks. We observed that the dimension order routing performed poorly on the Tornado traffic pattern because (dor) route all of the traffic in the short direction, leaving the channels in the other direction of the network idle.

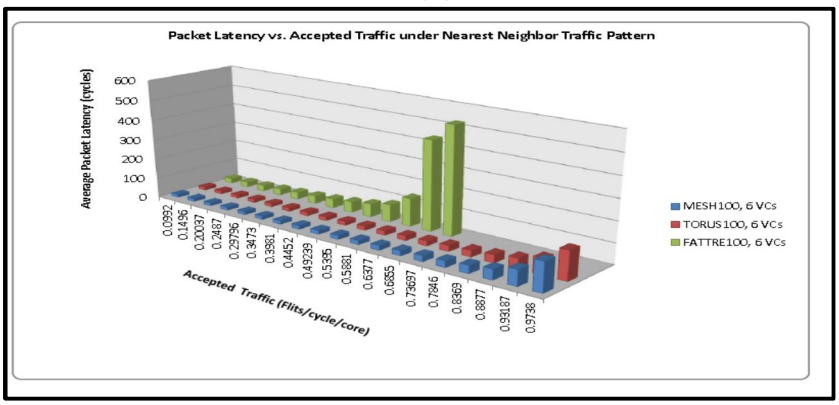
Figure 23 (a–c) shows the variation of average packet latency versus accepted traffic under Random Permutation traffic pattern with varying number of VCs from 2 to 6. In Random Permutation of traffics a random permutation of traffics is chosen from the set of all permutations. Due to this distribution of traffics the Mesh100 and Torus100 network configurations suffered tremendously under Random Permutation traffic pattern. However, the Fattre100 with 2 to 6 VCs offer the best network performance with lower packet latencies and higher number of accepted traffic. This is attributed to the nearest common ancestor routing (nca) and lower hop counts in the network. This leads to reduced packet contention times, thus reducing the packet latency and increasing the number of accepted traffic. In conclusion, from the conducted simulation, we observed that, the Fattree100 network configuration produced better network performance under both benign and adversarial traffic patterns. This makes the Fattree network to be better in terms of lower network latency and high amount of accepted network traffic. This suggests that, the Fattree network can be employed in the complex NoC design and in parallel and real–time applications like supercomputing. These findings agree with the findings by Coll et. al [73], Leiserson [148] and Wang et al. [149].



(a) With 2VCs

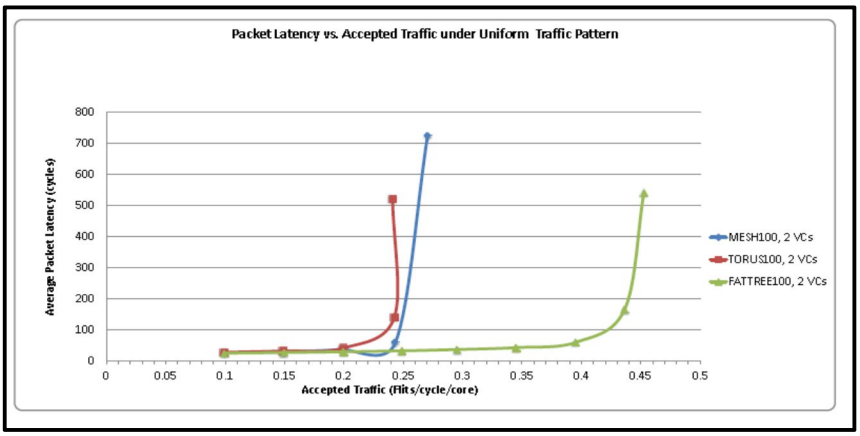


(b) With 4VCs

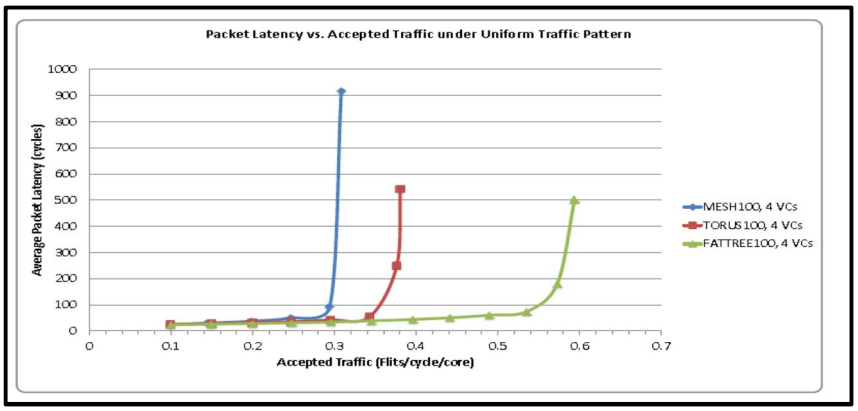


(c) With 6VCs

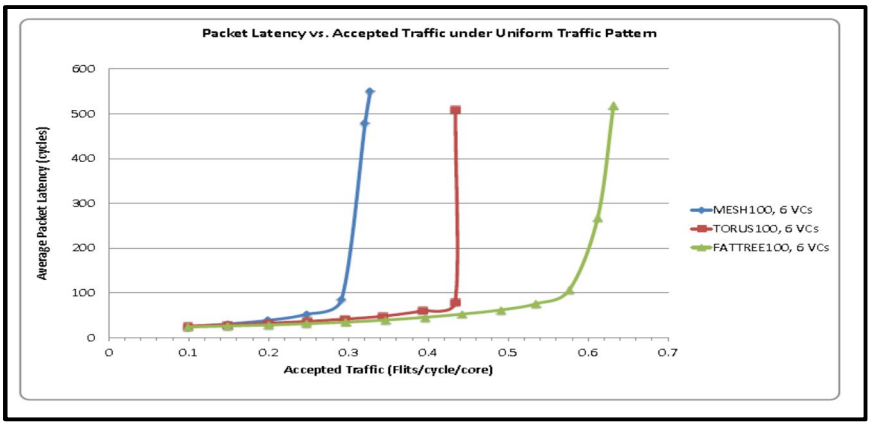
(Figure 20) Shows packet latency versus accepted traffics under nearest neighbor traffic pattern for Mesh100, Torus100 and Fattree100 network configurations.



(a) With 2VCs

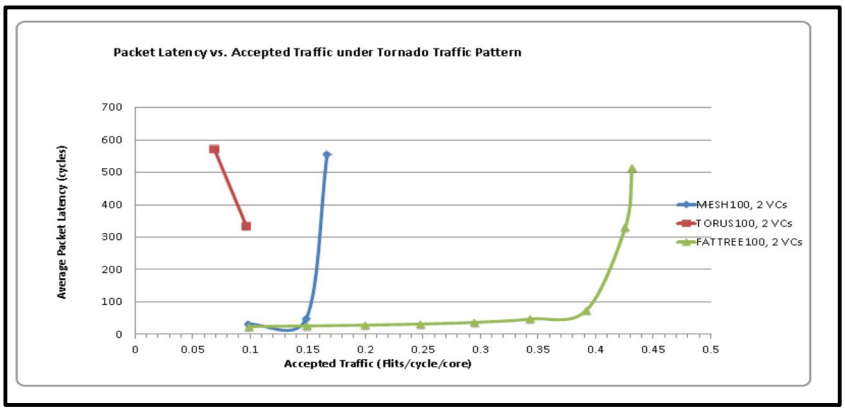


(b) With 4VCs

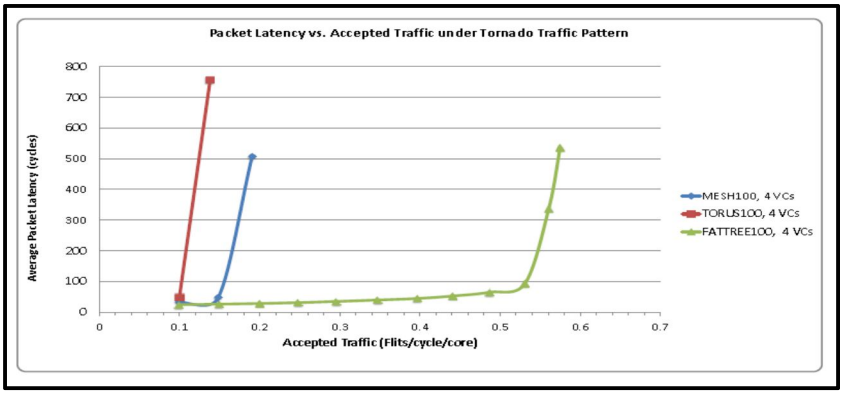


(c) With 6VCs

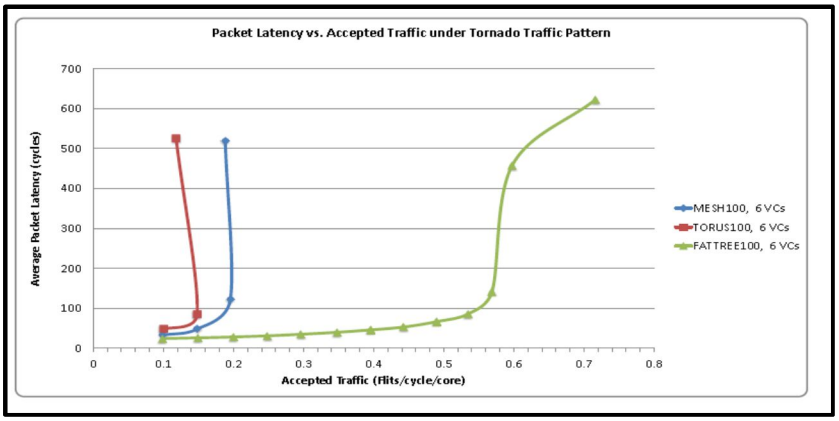
(Figure 21) Shows packet latency versus accepted traffics under uniform traffic pattern for Mesh100, Torus100 and Fattree100 network configurations.



(a) With 2VCs

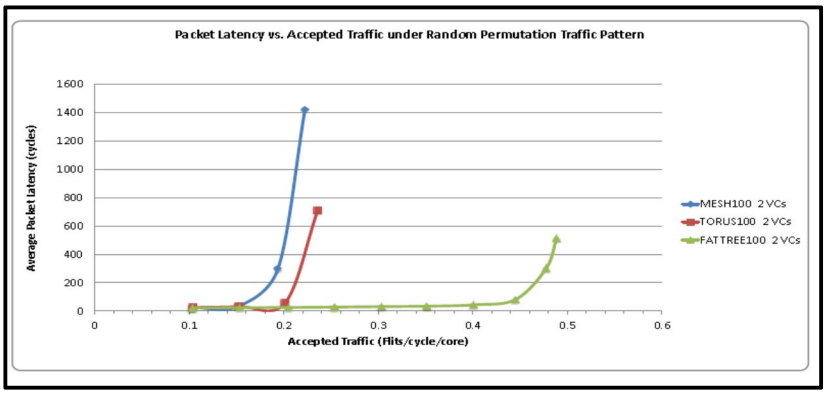


(b) With 4VCs

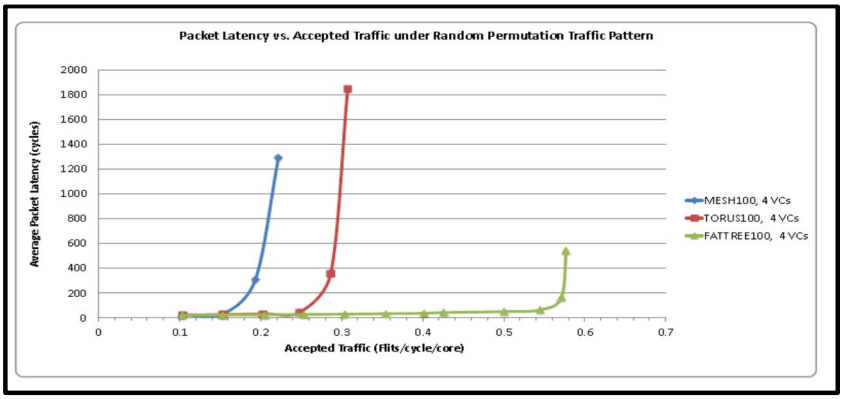


(c) With 6VCs

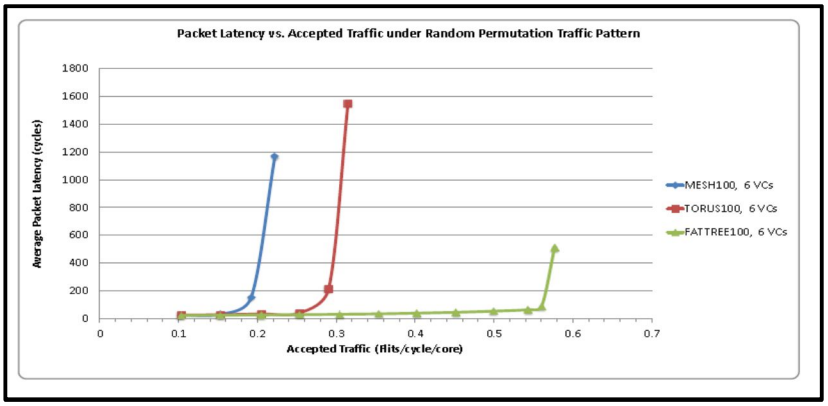
(Figure 22) Shows packet latency versus accepted traffics under tornado traffic pattern for Mesh100, Torus100 and Fattree100 network configurations.



(a) With 2VCs



(b) With 4VCs



(c) With 6VCs

(Figure 23) Shows packet latency versus accepted traffics under random permutation traffic pattern for Mesh100, Torus100 and Fattree100 network configurations.

D. Chapter Summary

This chapter describes a detailed parametric evaluation technique and used it to compare the performance of Mesh, Torus, and Fat-tree NoC architectures using FPGA. It demonstrated that the Virtual Channels (VCs), Flit Buffer Depth (FBD), and Flit Data Width (FDW) parameters all contributed to higher LUTs and influenced the area (PLUT) and clock frequency (CLK_FREQ) of the FPGA network, with FDW and FBD having the greatest impact on FPGA resources. We found that these three parameters significantly influenced reassembly buffering and routing and logic requirements at the NoC endpoints. To demonstrate the flexibility and design space coverage of three target NoC architectures, the chapter reported their results synthesizing of 392 different FPGA network configurations. It was shown that by strategically tuning the router and interconnect parameters, Fat-tree networks offered the best utilization of FPGA resources in terms of silicon area, clock frequency, critical path delay, network cost, overall performance under saturation throughput, and lower latencies under both benign and adversarial traffic patterns. In contrast, we consistently found that the Mesh and Torus networks consumed too much FPGA resources and produced poor network performance under adversarial traffic patterns. Through these findings, also demonstrated that the evaluation technique can substantially improve performance under a large variety of experimental conditions, confirm its suitability for real system development. The proposed methodology forms an important foundation for engineers to make informed early decisions about which interconnects and router parameters to use in large and complex NoCs for FPGA.

IV. An Autonomous Self-aware and Adaptive Fault Tolerant Rounting Technique (ASAART)

A. Autonomous System

This chapter was written based on the published research paper [174]. This subsection gives a brief overview of autonomous systems and how it used in the dissertation. An autonomous system is one which identifies its operating environment and senses the operating parameters, transforms its behavior in that environment, and adapts to the situational changes of the environment [10-24]. Autonomic systems provide a means to address the system's complexities by employing technology to manage and control complex and dynamic systems. Such systems work with independent and predefined conditions, policies, rules, etc. without human involvement. They can configure, optimize, manage, and control their actions based on predetermined conditions, rules, and acquired knowledge of the operating environment for a given period. An autonomous self-aware system can be employed in wireless sensor network to solve routing of sensor data without human intervention. In autonomic computing, self-managing capabilities in the system are achieved by employing suitable actions according to situational changes that arise in a dynamic environment. The main goal of autonomic systems is to control the loop that sources and gathers detailed information from the

environment, and then take appropriate actions. According to Kephart and Chess [10], there are four characteristics of an autonomous system, as follows:

Self-Configuration: In wireless sensor networks, self-configuration is the ability of sensor nodes to adapt dynamically to environmental and situational changes based on the predefined stated policies and rules that govern the operating environment. This entails the deployment of new sensors, removal of faulty components, or unpredicted situational changes that may arise in a dynamic environment. The use of dynamic adaptation provides continuous strength in sensing and enhances sensor node productivity, which gives rise to the scalability and flexibility of sensor networks.

Self-Healing: This is one of the characteristic features of autonomous systems that provide the means for autonomic route discovery, fault detection, and recovery in wireless sensor networks. Such feature is the ability to discover, diagnose, and react to unpredicted situations that may arise in a dynamic environment. Self-healing sensor network components and routing techniques can detect network malfunction and initiate a route repair technique without affecting the entire sensor network. The route repair technique makes the network resilient against errors and faults.

Self-Optimization: This attribute provides autonomic monitoring and control to facilitate fair and optimal use of available resources. Self-optimizing sensor components have the ability of controlling themselves to achieve the goals and

policies stated in a given dynamic environment. Such control measures are the reallocation and rescheduling of available resources caused by, arising situations in the dynamic environment in order to enhance overall network utilization and real-time transmission of sensory information [10].

Self-protection: This characteristic of autonomic systems provides sensor networks with the capability of anticipating, detecting, identifying, and providing protection mechanisms against threats and other variant attacks. Self-protecting sensor components can detect malicious and suspicious behavior within the network, and take reactive measures to counterattack actions in order to make the network less vulnerable to threats. In conclusion, when sensor network components possess these four characteristics, the network can configure, heal, optimize, and protect its operation, and enhance routing performance [10-15].

1. Methodology to autonomous self-aware and adaptation for routing in wireless sensor networks

Herein, the dissertation describes a methodology to autonomous self-aware and adaptation for routing sensor data. Autonomous self-aware and adaptive systems are a proven solution for overcoming the complexity imposed by sensor network routing algorithms [5-50],[168] and so are self-healing systems [10-28], [28,55,97]. Self-aware and adaptive systems are better methods for enhancing and improving the potential for understanding environmental situations and complex dynamic environments. In order to have a better understanding of the overall systems' capability and awareness to acquire and process information, five

levels of awareness need to be considered. Such levels describe the extent and worthiness to which a system can adapt, and the extent that the respective adaptation effects have on the system [11−16]. In this dissertation, we consider the five levels of self−awareness given in [13], which are:

Event Awareness: This is the main level of self−awareness. This level focuses on the self−description of sensor network entities and how the events associated with these entities relate to other entities in the network system. It involves using basic information processing for sharing among network entities in the entire network. For dynamic adaptation, the self−aware routing technique must not establish a local routing decision only, but consider both local and global state information, and a defined boundary between self−awareness and adaptation. This boundary has to be considered at run−time based on the routing function and tuned parameters [17−24].

Situation Awareness: At this level of awareness, several techniques and principles are joined to form a consistent reasoning both at the local and global level situational awareness. At this level, the routing function is advanced to specify a property between the sensor's network entities that are nodes and links. Several methods and techniques can be employed to ensure reliable and efficient understanding of the situational system. For example, fuzzy logic and probabilistic models can be used at this level [37−55].

Adaptability Awareness: This level of awareness provides support for network entities to detect the adaptability of other network components and provide their own adaptation interface. By doing so, the sensor nodes can select the best neighbors to meet the self-adaptation goal, *i.e.*, find the best routing path. At this level of awareness, different methods and techniques can be employed to ensure that the routing algorithm possesses the capability of adapting to situational changes [22].

Goal Awareness: This level of awareness defines the goals or the objective function to be achieved by the routing algorithm. This involves combining several different actions, goals, adaptations, and conflicting goals for the efficient routing of packets in the network, and for resource utilization. The routing function needs to specify an objective function that can be minimized or maximized at runtime. Goal awareness entails runtime goal checking to determine dynamically those goals that can be satisfied for a given constraint resource and threshold limit [10,22].

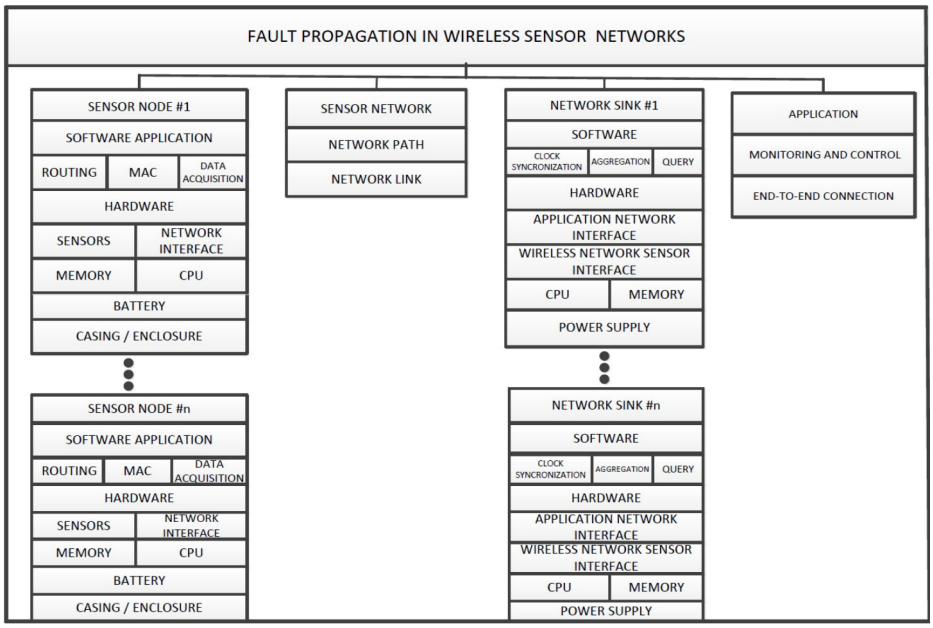
Future Awareness: This level of awareness is concerned with a thorough knowledge of the entire sensor network state. These include network cost, energy consumption, overhead, reliability, *etc.* This information helps reduce the number of actions to be taken in the future. Similarly, monitoring the network state can lead to the underlying data for future resource volume and utilization. Therefore, at this level, adaptive identification of the network state information is the major

goal associated with this level [13−16], [22]. By way of integrating these awareness levels into SHR, the problem associated with the routing of sensor data in wireless sensor networks can be solved.

2. Modular approach to fault propagation in wireless sensor networks

In order to integrate autonomous self−aware and adaptive mechanisms into fault tolerant routing for sensor networks, it is essential to explain the basic difference between faults, errors, and failure [14−16]. In sensor networks, a fault is any type of sensor node defect that leads to error, for instance, a loose sensor node connection in the network. An error is an undefined state of the node condition such that the condition or state leads to a node failure, *i.e.*, the state of the system when transmitting and retransmitting sensor data. In a sensor network, sensor node failure implies the manifestation of error. This occurs when a sensor node cannot perform its functions and violates the network design specification, which happens when sensor nodes cannot transmit their data within a specified time interval. To integrate self−awareness functionalities into sensor networks in order to provide resiliency against faulty conditions, fault detection and recovery measures are required. In fault detection, some design techniques need to be incorporated into the routing algorithm in order to detect that a specific functionality of the sensor nodes is or will be faulty. On the other hand, after a fault is detected, the routing algorithm should be able to prevent or recover from

it. This ensures smooth and steady packet routing from sources to destinations [17−50]. In order to illustrate how a fault is propagated in different sections of the sensor network system, we propose a modular architecture in contrast to the layered architecture presented in [16]. As illustrated in Figure 1, a fault in each module cannot be propagated to other modules in the system. For instance, when the node battery power is down, this causes only the node to be down without affecting other modules. In layered approach, the failed node affects the operation of other nodes. If this node is in the critical path of the network, the packets of other nodes that depend on the critical path will not reach their destination until the path is recovered or restored. In addition, if the application module components, *i.e.*, data, and users suffer a faulty software bug or hardware failure for the sensor node, the overall system is considered to be faulty as well. With the modular approach to fault propagation in sensor networks, fault occurrences are limited within their module without affecting other module functionalities.



(Figure 24) Shows modular approach to fault propagation in wireless sensor network.

B. Proposed Methodology to Autonomous Self-Aware and Adaptative Fault Tolerant Routing Technique (ASAART)

In this subsection, the dissertation demonstrates the proposed approach using autonomous self-awareness and adaptive routing techniques, namely ASAART. The dissertation considers low duty-cycle wireless sensor networks [18-23], [37]. In this type of network, the sensor nodes remain sleeping most of the time and wake up asynchronously in the case of event transmission. Therefore, the concept of probabilities that forwards a decision based on the delay distribution of the destination nodes can be employed [5-151]. In doing so, only packets are transmitted in order to achieve shorter delays and reduce packet redundancy. The

probabilistic approach signifies that when a sensor node receives a network packet, the packet is forwarded with given probability δ. Such probability δ is computed by the network state information obtained from individual neighboring nodes. The routing decision is computed using the improved formula given in Equations (19) -(25). The SHR [28,97] and SSR [44,55,96] protocols use more redundant transmission and retransmission of packets. The proposed approach attempts to reduce such redundant packets. SHR and SSR make use of prioritized back-off delay for forwarding packets, and only the receivers of the forwarding packet reach the destination. The proposed approach focuses on the reduction of these control packet overheads and efficient data delivery ratio. Rather than using local information to route a given packet to the desired destinations, the proposed approach combines both the local and global state information in order to avoid faults and collisions in the network. The most important issues to address are: (1) at the initial phase of network construction, how to avoid the problem of packet flooding because of broadcast and (2), when the network is scaled to a large network, the broadcast packet will overflow the network, thus causing too much delay and packet collisions that consume much energy. In order to address these issues, the autonomous self-awareness and an adaptive scheme proposed in this dissertation will ensure reliable and efficient transmission and retransmission of network packets. For the self-awareness scheme, the sensor node that transmits the packets contains information pertaining to its neighbors in the control packet.

The node is self-aware and adaptive if it knows all the local and global information on the packet. Then, the receiver node knows whether its neighbors have been covered based on the global information obtained in the neighbor's control packet list information. In this case, the sensor node will compute the auto self-awareness back-off delay, according to Equations (19) -(25) in order to obtain the entire network state information. The sensor node with the auto self-awareness back-off information now has a higher priority to forward the network packet. In this case, the sensor node that uses the global information of its neighbors has a very low back-off delay. The self-awareness and adaptive features of the node's neighbor network state information minimizes the network end-to-end delay and enhances the entire network routing performance. Here, we define global, local, and auto-self-awareness back-off delays as given in Equations (19) -(25). Ideally, this combines both the local and global network state information to have full autonomic self-awareness of the entire network state information, and reduces the time to converge to minimal routing path for efficient packet routing throughout the entire network [28], [45-55], [96-97].

Herein, we define the parameters used in Equations (19) -(25). $Hopcount_{Dest}$ is the number of hop counts from the current node to the destination; $NumHopCount_{expected}$ is the expected number of hops contained in the destination reference packet header information; $Scale_{Factor}$ is a parameter used to stretch the random function in order to avoid network packet collision; Rand

(seed) is a random function generator that produces random floating point numbers within the interval [0,1] based on the value of the seed; and $Slot\_size$ is the size of the slot or the transmission window width. The global awareness transmission back-off delay is computed for each node using Equations (19) and (20):

$$\text{If } Hopcount_{Dest} > NumHopCount_{expected}$$

$$Global-Awareness-Back-off_{Delay} = Scale_{Factor} * (Hopcount_{Dest} - NumHopCount_{expected} * Rand(0,1) \quad (19)$$

$$Else\ Hopcount_{Dest} \leq NumHopCount_{expected}$$

$$Global-Awareness-Back-off_{Delay} = \frac{Scale_{Factor}}{NumHopCount_{expected} - Hopcount_{Dest} + 1} * Rand(0,1) \quad (20)$$

The local awareness transmission back-off delay is computed for each node using Equations (21) and (22), depending on whether the hop counts from the destination parameter is greater, less than, or equal to the sender's node number of expected hop counts:

$$\text{If } Hopcount_{Dest} > NumHopCount_{expected}$$

$$Local-Awareness-Back-off_{Delay} = Scale_{Factor} * (Hopcount_{Dest} - NumHopCount_{expected} + Rand(0,1)) \quad (21)$$

$$Else\ Hopcount_{Dest} \leq NumHopCount_{expected}$$

$$Local-Awareness-Back-off_{Delay} = Global-Awareness-Back-off_{Delay} \quad (22)$$

The autonomous self-awareness technique combines both the local and global back-off delay transmission strategies in order to find the entire network state information for efficient transmission and retransmissions of packets. Equation (23) combines two randomized functions to allow both the near and far nodes minimize the back-off delay. Based on this, the sender node has a greater probability of transmitting the packet:

If $Hopcount_{Dest} > NumHopCount_{expected}$

$AutoSelf-Awareness-Back-off_{Delay} =$

$$Scale_{Factor} * (Hopcount_{Dest} - NumHopCount_{expected} * Rand(0,1) + Rand(0,1)) \quad (23)$$

Else $Hopcount_{Dest} \leq NumHopCount_{expected}$

$$AutoSelf-Awareness-Back-off_{Delay} = Global-Awareness-Back-off_{Delay} \quad (24)$$

From Equation (23), it is clear that the auto self-awareness back-off delay assigns a delay greater than $Hopcount_{Dest}$ to nodes with a hop count greater than $NumHopCount_{expected}$ parameters. Therefore, the smaller the value of the $Hopcount_{Dest}$ parameter, the smaller is the $AutoSelf-Awareness-Back-off_{Delay}$ parameter and the node has the highest probability of transmitting the packet. The slotted autonomous self-aware back-off slotted formula is expressed in Equation (10) as [28,44,55,],[96-97]:

$$AutoSelf-Awareness-Back-off_{Delay\_Slotted} = \left\lfloor \frac{\text{AutoSelf-Awareness-Back-offDelay}}{Slot\_Size} \right\rfloor * Slot\_size \quad (25)$$

Where the parameters $AutoSelf-Awareness-Back-offDelay$ are given in Equations (23) (24) and (25), and $Slot\_size$ is the size of the slot window used. Equation (25) provides the autonomous self-aware and adaptive back-off delay that combines both the continuous and slotted time interval. This approach lowers the back-off delay, and the sensor nodes have the greatest probability of transmitting the packets. In addition, the approach provides the current node with both local and global network state information in order to reduce end-to-end delay for efficient and reliable packet transmission.

1. Autonomous self-aware and adaptive route repair in ASAART protocol

The dissertation employs the greedy algorithm [37,152] to implement the autonomous self-awareness and adaptive route repair technique. By implementing the concept of the self-awareness principle defined in this dissertation, the sensor nodes become self-aware and adaptive for both local and global network state information. During route maintenance, the source node initiates a new route repair packet (*RRP*). The route repair packet *RRP* consists of the source node identification ($Source_{id}$) number, packet identification number ($packet_{id}$), route metric ($Route_{Metric}$), back-off delay ($Back-off_{Delay}$), and route metric sequence number of sources to destination ($RouteMetric_{Seqn}$). In this case, when a sensor

node *k* receives an *RRP* packet, it processes the packet according to the algorithm given in Figure 25.

```
// Autonomous Self-aware and Adaptive Route Repair Technique (in ASAART)
1.  Let Sensor_Network = sensor network consisting of sensors
2.  Let Sensor_node = sensor node
3.  Let d = destination node
4.  Let k = intermediate node
5.  Let s = source node
6.  Let Route_Metric = route computation metric
7.  Let AutoSelf-Awareness-Back-offDelay = use equations 19 to 25
8.  Let RR_ACK = route repair acknowledgement packet
9.  Let RRDP = route repair discovery packet
10. Let Sequence-Number_s-d = sequence number from source to destination
11. Let Sequence-number_-d = sequence number of intermediate node to the destination node
12. Let Local-Node-Neighbor_info = set of nodes within the sender's node transmission range
13. Let Global-Node-Neighbor_info = set of nodes within the sender's node outside transmission range
14. Begin {
15. With s as starting node update the Global-Node-Neighbor_info (k) using the Local-Node-Neighbor_info (s)
16.   If (k not equal to s) and (k not equal to d) then
17.       If RRDP then
18.           If Sequence-Number_k-d greater than Sequence-Number_s-d then
19.               K transmit RR_ACK together with Sequence-Number_k-d
20.           Else if ∀ neighbors in Sensor_Network receive the RRDP then
21.               Discard RRDP
22.           Else
23.             Compute AutoSelf-Awareness-Back-offDelay to get the global update information
24.   If Sensor_node transmits RRDP and Global-Node-Neighbor_info (k) = null using equation 19 to 25 then
25.               Discard RRDP
26.           Else
27.               Global-Node-Neighbor_info (s) = Global-Node-Neighbor_info (k)
28.                 Route_Metric = Route_Metric + 1
29.         End if
30.       End if
31.     End if
32.   Else if k = d and RRDP is initiated then
33.       Transmit RR_ACK
34. End if
35.   }
36. End
```

(Figure 25) Shows the proposed autonomous self-aware and adaptive route repair technique in (ASAART).

There are two sets of nodes, one that consists of the node's neighbors within its transmission range ($Local-Node-Neighbor_{info}$), and another that is outside the transmission range that leads to the minimal routing path ($Global-Node-Neighbor_{info}$). At the initial stage of network creation, sensor node *k* updates its global state information $Global-Node-Neighbor_{info}$ (*k*) using the source node's

local neighborhood information ($Local-Node-Neighbor_{info}$) to update its global node information. This approach helps to prevent and avoid both transient and permanent faults and failure in the network. A detail of the route repair technique in ASAART is illustrated in Figure 25.

2. Reliable and secure route formation in ASAART

In the proposed ASAART routing protocol, the route discovery packets are transmitted by the source node through the packet broadcast mechanism [36-51]. In this case, there is a need to control the continuous flooding of the route discovery packets, particularly in a very dense network. In the proposed scheme, shown in Figure 25, ASAART uses the sender node local information that is contained in the control packet. Upon receiving the packets from the sender node, it uses the local neighbor information to update its global network state information contained in the control packet header information. The node then computes the auto self-awareness back-off delay based on the expected number of hops to the destination, as given by Equations (19) -(25). During the computation of the auto-self-awareness back-off delay, the sender node listens to the channel and prevents the transmission of the data packets. This happens when the sender node overhears that one of its neighboring nodes with the expected hop count to the destination has already transmitted the data packets. However, if the sender node does not overhear its neighbor's transmission and its back-off timer fires, the sender node will now update its expected hop count in the data packets and then re-transmit the data packets to the destination. The sender node with the auto self-awareness back-off information will have the

highest priority to forward the packets to the next intermediate or destination node. To reduce the problem of redundant transmission, after the destination node receives a route discovery packet, it sends a route discovery acknowledgement to the source node. ASAART uses an efficient flooding control scheme that minimizes the redundant transmission of network packets. In this case, the intermediate and destination nodes have the complete network state information for a reliable route formation based on the discovery packet's auto self-awareness back-off information received from the source node. This scheme will minimize the redundant transmission of packets and ensure reliable end-to-end data delivery. The concept of secure route formation is beyond the scope of this dissertation.

C. Performance Evaluations

In this subsection, the dissertation provides routing performance results that compare the ASAART, SHR and SSR protocols. Because the proposed routing technique (ASAART) integrates the autonomous self-awareness and adaptive feature of routing path discovery and route repair into SHR, the dissertation uses the Sensor Network Simulator and Emulator (SENSE) [40] for performance evaluation. We consider both local and global network state information to compute the routing decision using the modified self-aware formula and backup strategies and efficient and reliable route repair technique as given in this dissertation.

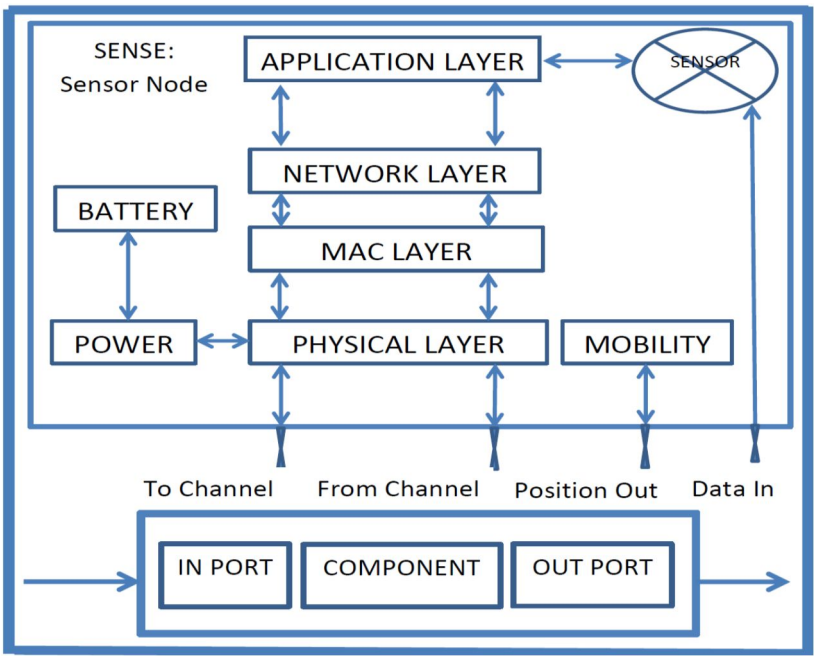
We performed five sets of simulations under a large and scalable sensor

network with different simulated scenarios. In the first to third scenarios, we evaluate the network density test under a large and scalable sensor network. We consider a sensor network with 100 to 1000 nodes in step of 100 nodes. We vary the number of source nodes from 10 to 20 and 40 nodes in increasing network traffics. We evaluate protocol performance under increasing network density to observe the extent to which the proposed protocol is fault tolerant and resilient against errors with different network configurations. In the fourth and fifth scenarios, we evaluate the performance of the three protocols under transient and permanent node failure rates. In the conducted simulation, we consider a 2000 × 2000 m^2 terrain densely populated with 100 to 1000 nodes in step of 100 nodes, all which have a nominal transmission range of 250 m. We employed the free space propagation model in order to simulate the wireless medium [41]. We used the Constant-Bit-Rate (CBR) model to simulate the bi-directional traffic between the sources and destinations. The simulation at each scenario was performed at least five times with a different number of seeds to ensure accurate and reliable measurements and avoid errors. The 95% confidence intervals are shown in the figures.

1. Simulation environment

The purpose of the simulation is to analyze the performance of the proposed autonomous self-aware and adaptive fault tolerant routing technique (ASAART) compared with SHR and SSR protocols. This allows us to study WSN behavior in

the event of faults associated with the sensor nodes and the network, and to determine the extent to which the proposed scheme is fault tolerant. SENSE is used for the simulation [40,46].



(Figure 26) Shows SENSE simulator internal structure of sensor node.

2. Simulation parameters

The following parameter values in Table 9 are used for the simulation. The values of these parameters are fixed and varied for different scenarios in order to provide efficient and reliable self-aware and adaptive fault tolerant routing techniques, and for better evaluation of the results.

(Table 9) Simulation Parameters

Parameter(s)	Value
Network Size Terrain (m^2)	2000 × 2000
Number of Sensor Nodes	100-1000 varied 600 nodes fixed
Number of Source Nodes	10, 20, 40 varied
Data Packet size (byte)	1000
Active Percentage	100%, 20% fixed 10%-60% varied
Active Cycle	100%, 20% fixed 10%-60% varied
Transmission Time (s)	0.002
Transmission Power (Tx) (Watt)	0.0290
Channel Model	Radio
Broadcast data packet	CBR
Slot Width (s)	0.0001
Simulation Time (s)	200
Scale Factor (s)	0.001
Traffic Direction	Bi-directional
Route Repair	SHR, ASAART (True) and SSR (False)

3. Performance evaluation metrics

Herein, the dissertation provides a brief explanation of the performance evaluation metrics used. The dissertation evaluates the three routing protocols in terms of the sensor network density test and transient and permanent node failure rate tests. In order to have consistent and reliable routing performance evaluation, the five evaluation metrics listed below are used. Such metrics are widely accepted by the research community and are used for wireless sensor network performance evaluations [5-68]:

1. **Average End-to-End Data Packet Delivery (Success Rate):** This is the fraction or percentage of success among a number of attempts to transmit a packet.
2. **Average Packet End-to-End Delay:** This refers to the time required for a packet to be transmitted across a network from source to destination.
3. **Energy Consumption:** Defined as the ratio between the number of packets transmitted to the amount of energy consumed, where the amount of energy consumed is the difference between the initial energy before simulation to the energy used after the simulation. The unit is expressed in bits/joule.
4. **Average Number of Transmitted MAC Packets:** Total number of packets received at the MAC layer.
5. **Packet Drop Rate/Error Rate:** The Packet Error Rate (PER) is the number of incorrectly received data packets divided by the total number of received packets. A packet is declared incorrect if at least one bit is erroneous.

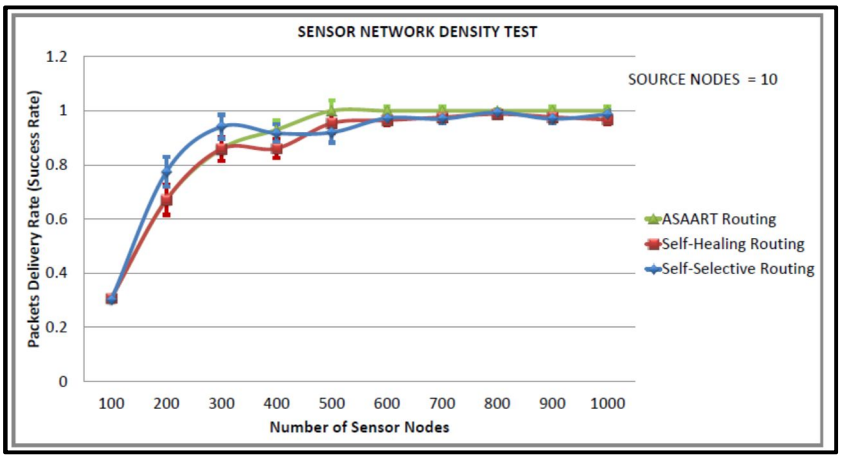
D. Sensor Network Simulation Results

In this sub-section, the dissertation provides the results of the simulation conducted based on five different simulated scenarios. These comprise of three different simulated scenarios for sensor network density test, as well as transient and permanent node failure rate tests. Details of the simulated scenarios are provided in the subsequent subsections.

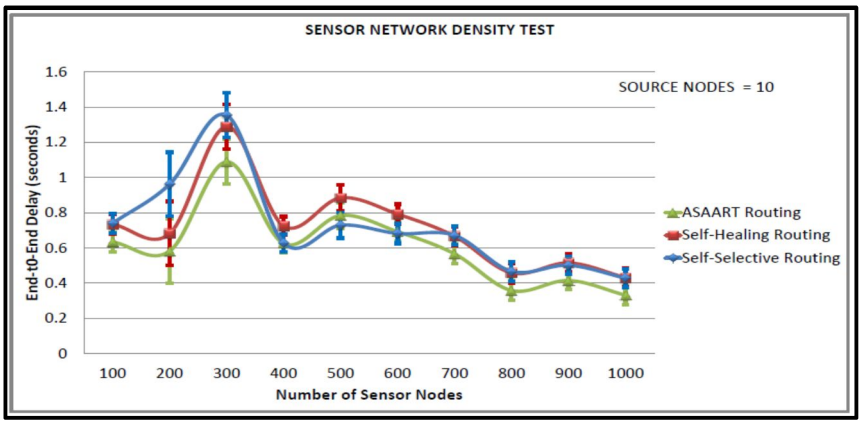
1. Sensor network density test (First scenario)

In this scenario, we set the number of source communicating nodes to ten, varied the network density from 100 to 1000 nodes in step of 100 nodes, and observed the routing performance. As observed from Figure 27 (a-e), the success rate drops to less than 100% between 100 to 1000 nodes by SSR and SHR protocols. However, ASAART maintain a 100% success rate between 500 to 1000 nodes compared with SSR and SHR protocols. This is attributed to the fewest number of communicating source nodes. There are fewer packets that reach the destination. However, the ASAART protocol slightly outperforms both SHR and SSR in terms of successful number of delivered network packets.

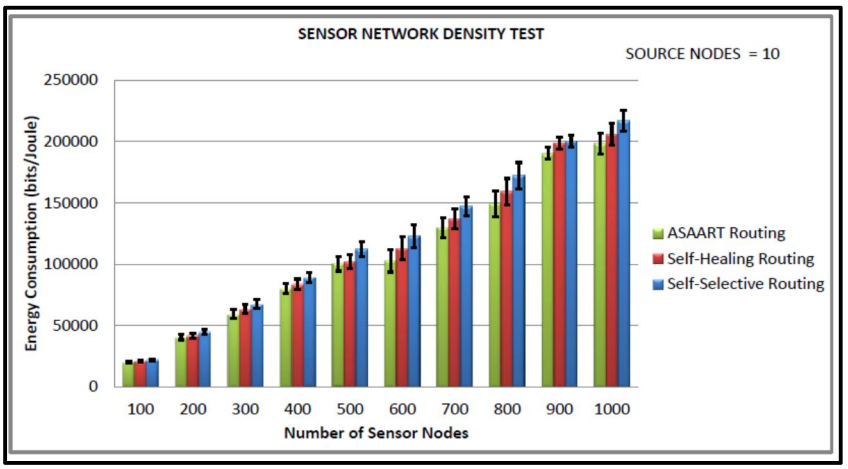
In Figure 27 (b), ASAART end-to-end network delays show a slight reduction compared with SSR and SHR protocols, which means better reduction in the end-to-end delay. In Figure 27 (c, d) we can observe that ASAART protocol shows better energy consumption and a total number of reduced MAC layer packet's delivery compared with the SSR and SHR protocols. This is attributed to its enhanced route repair mechanism that tries to control flooding at the MAC layer. In Figure 27 (e) we can observe that ASAART packet error rate (PER) is lower at higher network densities compared with the SSR and SHR protocols. Showing a zero percent of error in packet delivery between the network of sizes 500 to 1000 nodes.



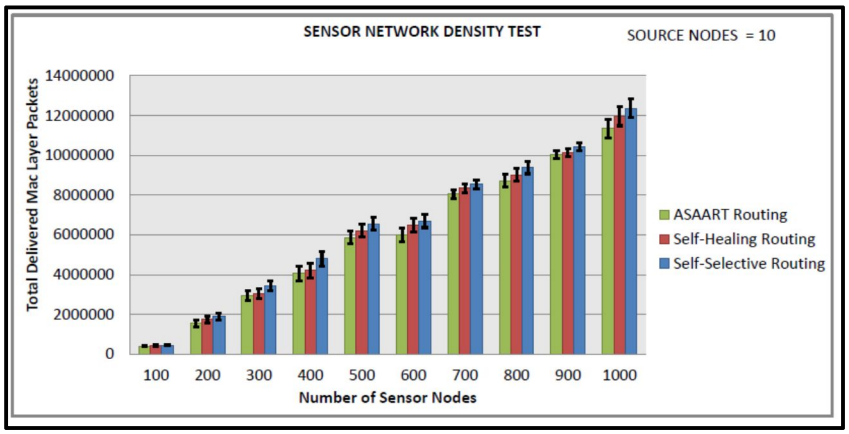
(a) Average packet delivery rate



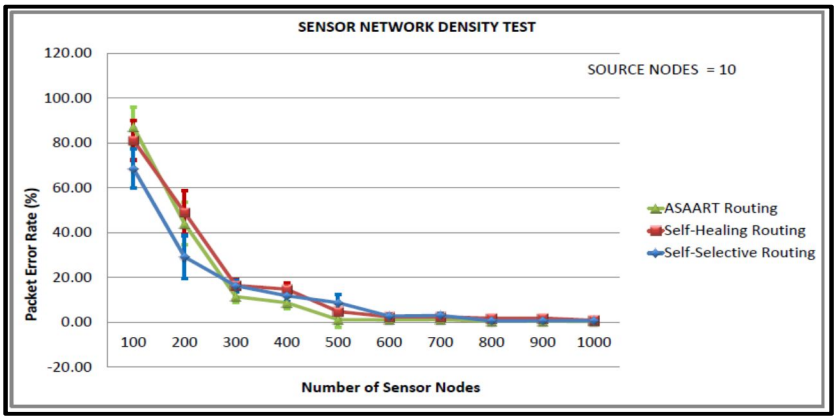
(b) End-to-end network delay



(c) Energy consumption



(d) Transmitted MAC layer packets



(e) Packets error rate

(Figure 27) Shows sensor network density test (first simulated scenario) with source node = 10

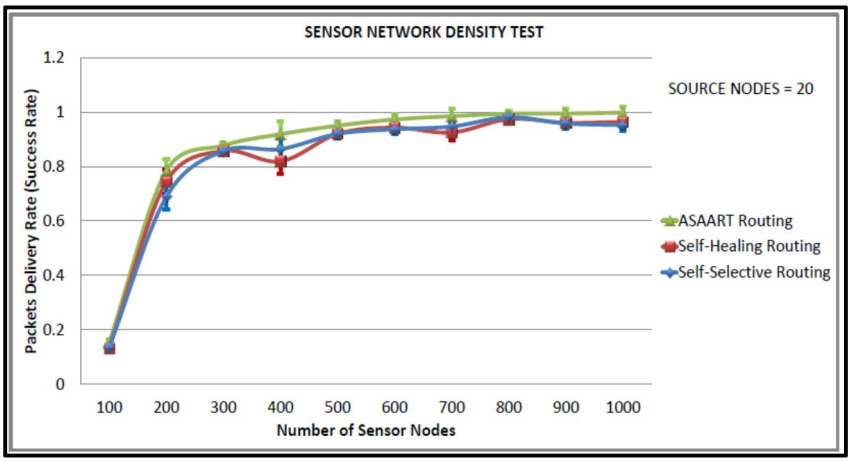
2. Sensor network density test (Second scenario)

In this scenario, we fixed the number of source communicating nodes to 20 and varied the network size from 100 to 1000 nodes in step of 100 nodes. As can be observed from Figure 28 (a-e), the ASAART protocol provides the highest packet

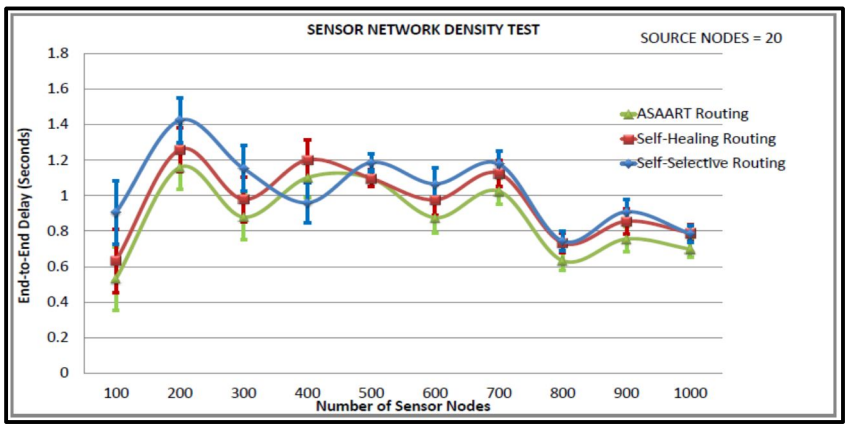
delivery rate or success rate compared with the SHR and SSR protocols. When the number of sensor nodes reaches 800 nodes, the success rate becomes stable with 100% packet delivery shown by ASAART protocol. However, at 400 node network size, SSR outperforms SHR protocol. This is because of its continuous back-off delay scheme that allows more flooding of network packets. However, at higher network density, both SSR and SHR protocols are competing, achieving a stable success rate with over 95% delivered network packets.

We can observe that the end-to-end delay reduces significantly when the source nodes are set to 20 compared with the third simulated scenario with source equal to 40 nodes. We can also observe that between 100 to 400 nodes, the end-to-end delay is higher in all three routing protocols, with a maximum of 1.4 s exhibited by the SSR protocol. However, at a higher network density, *i.e.*, from 400 to 1000 nodes, the end-to-end delay reduces significantly with the ASAART protocol, showing better reduction in end-to-end delay.

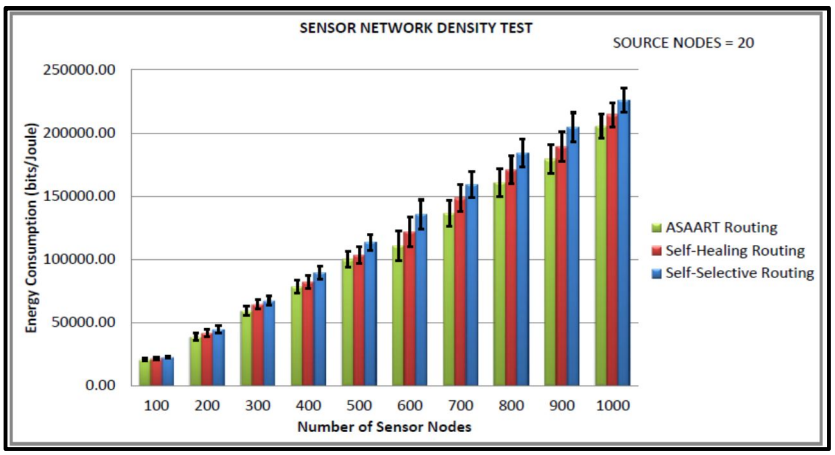
From Figure 28 (c), we can observe that the energy consumption is minimal compared with the SSR and SHR protocols at different network densities. This is because there is a reduced number of delivered MAC layer packets that reduce network congestions and save energy consumption. In addition, in Figure 28 (e), we can observe that PER for the three routing protocols at higher network density becomes stable particularly, between 600 to 1000 network sizes. Any further increase in network density has no impact on performance.



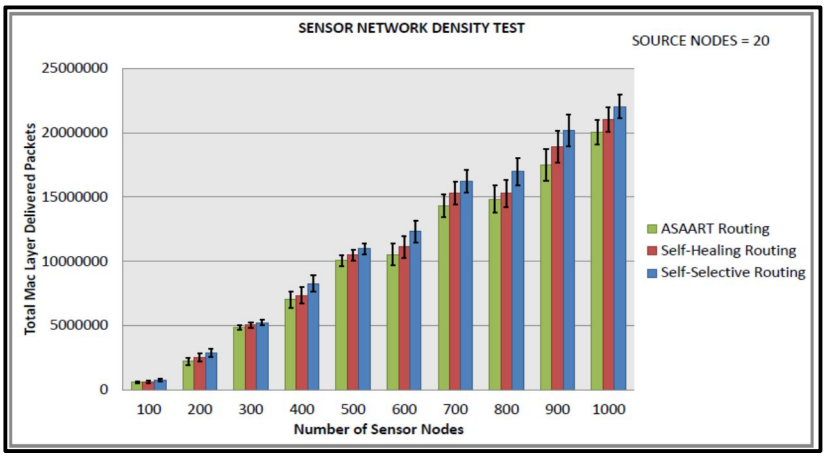
(a) Average packet delivery rate



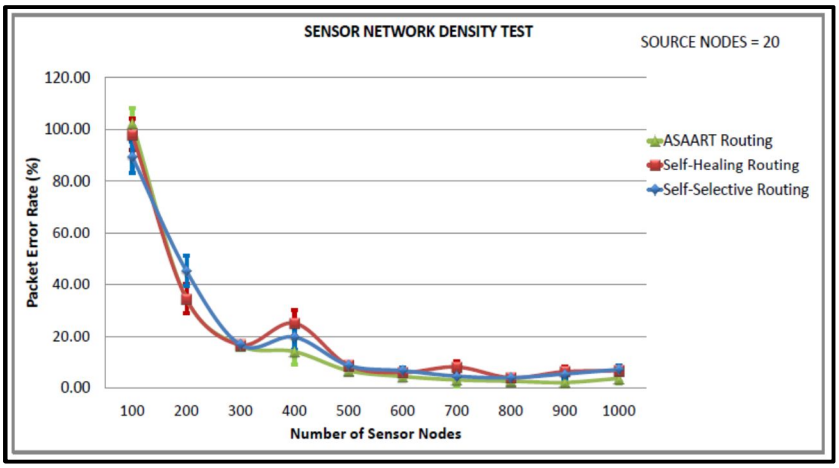
(b) End-to-end network delay



(c) Energy consumption



(d) Transmitted MAC layer packets



(e) Packets error rate

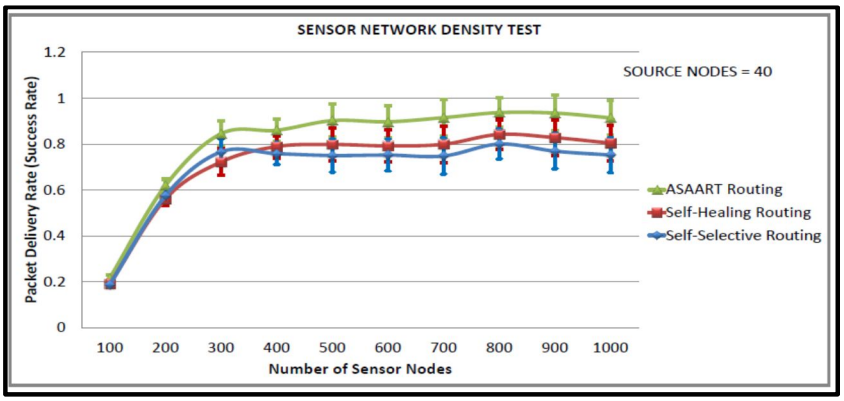
(Figure 28) Shows sensor network density test (second simulated scenario) with source node = 20

3. Sensor network density test (Third scenario)

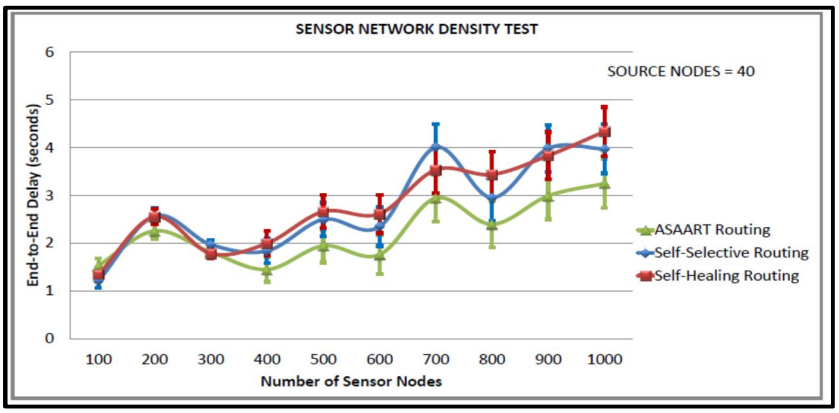
In the third scenario, we evaluated the impact of increasing the network density from 100 to 1000 in step of 100 nodes with 40 nodes designated as the source communicating nodes. The results of the simulation are illustrated in

Figure 29 (a-e). As can be observed in the figures, increasing the network traffics and density causes too much traffic oscillations in both SSR and SHR, and decreases the success rate in the network, particularly between 100 to 300 node network sizes. However, at higher network density, the success rate of the delivered packets becomes stable, with SHR showing slightly higher packet success rate compared with SSR. The proposed ASAART protocol maintains higher packet success rates at different network densities. In addition, the time required to converge to minimal routing path is minimal, which makes the end-to-end delay lower than for SHR and SSR. At the higher network density, SHR spends too much time on topology maintenance, which causes its performance to reduce drastically compared with the ASAART technique. Despite the higher packet success rate and lower end-to-end delay exhibited by ASAART, it also uses minimum energy and reduced MAC layer packet delivery compared with SHR and SSR protocols. In Figure 29 (e), we can observe that PER for ASAART protocol is lower with a network of size between 200 to 1000 nodes compared with the SSR and SHR protocols. The SHR protocol competes against SSR by showing a slightly higher packet delivery ratio at higher network densities particularly between 400 to 1000 nodes. It is interesting to observe that when the network is scaled from 400 to 1000 nodes, PER becomes stable for the ASAART routing protocol. In conclusion, the ASAART protocol proves to be resilient against errors, and energy consumption and offers better routing performance

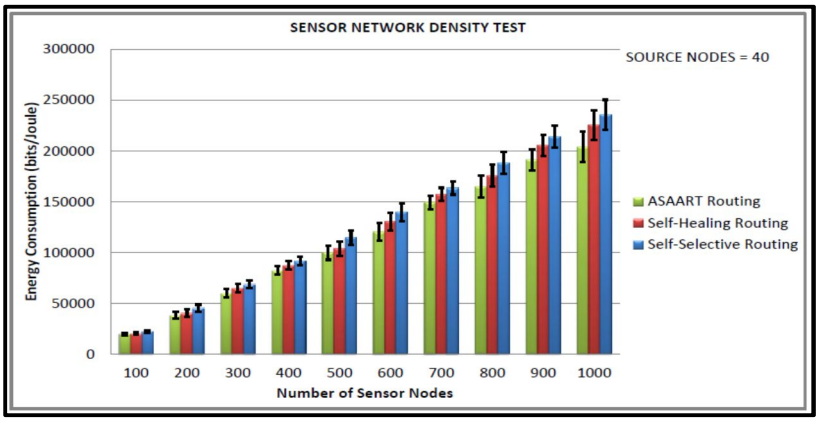
compared with the SHR and SSR protocols in a scalable and high traffic sensor network.



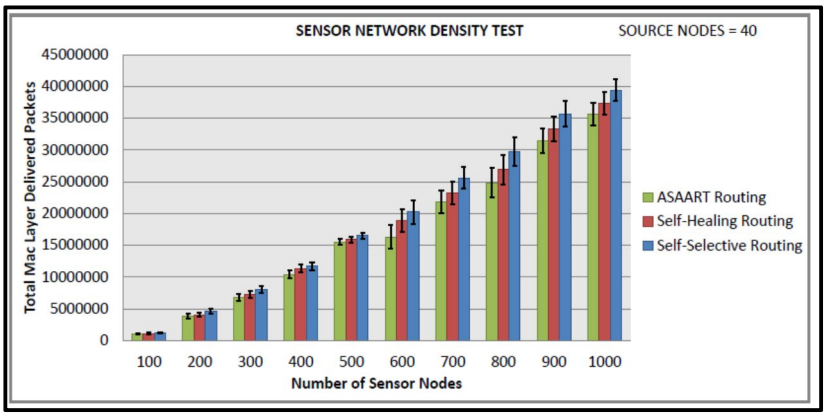
(a) Average packet delivery rate



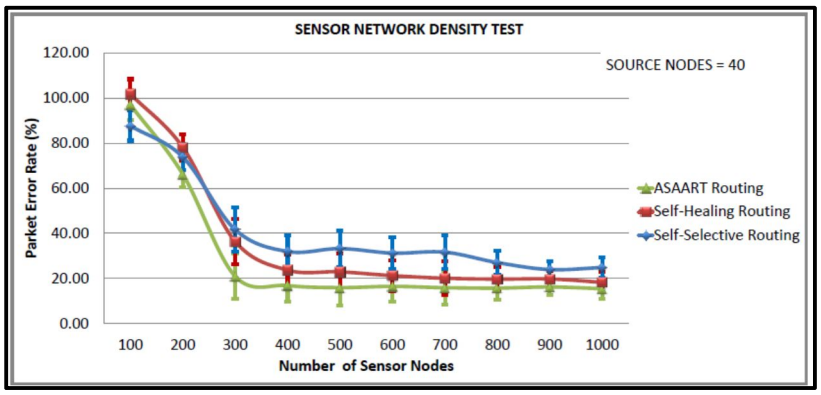
(b) End-to-end delay



(c) Energy consumption



(d) Transmitted MAC layer packets



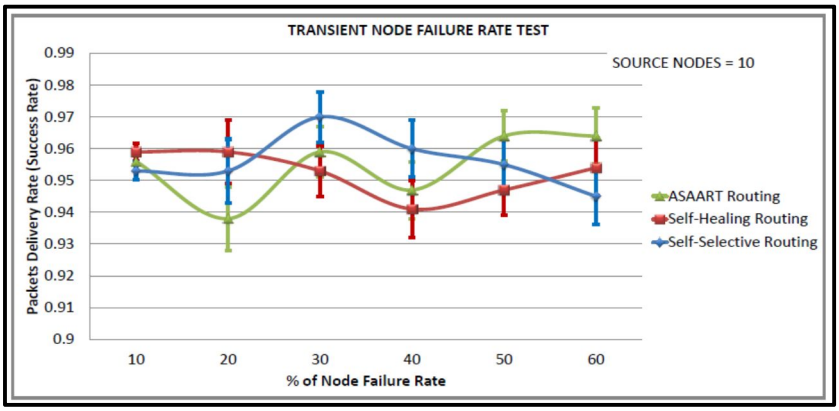
(e) Packet error rate

(Figure 29) Shows sensor network density test (third simulated scenario) with source node = 40

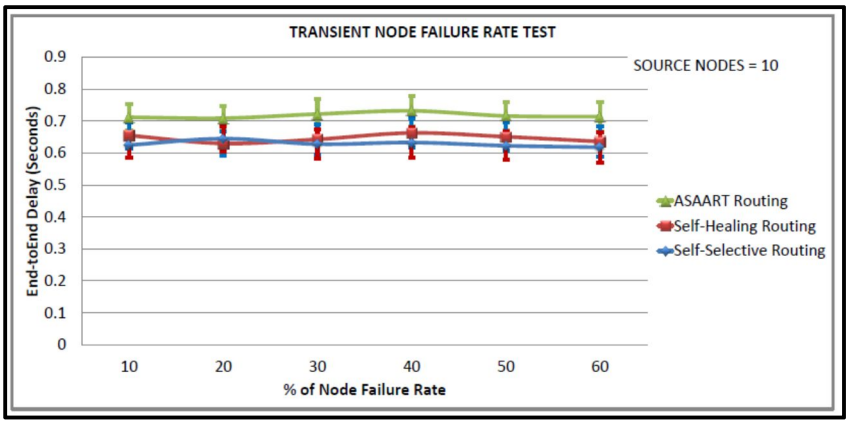
4. Transient node failure test

In this scenario, we consider a sensor network of size equal to 600 nodes. In order to simulate the transient node failure rate, we fixed the permanent node failure rate to 20% and varied the transient node failure rate from 10% to 60% in step of 10% in order to obtain routing protocol performance at different percentages of the transient node failure rate. We performed three sets of experiments with a different number of seeds, and varied the number of source

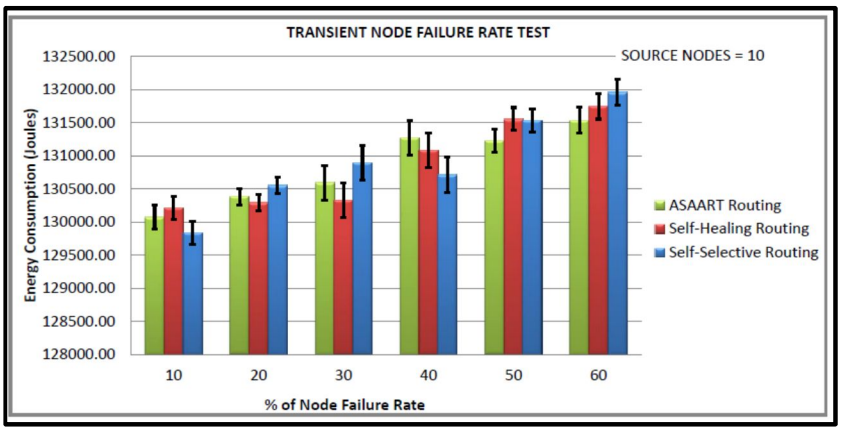
communicating nodes in each scenario. The simulation results and 95% confidence intervals are shown in Figures 30-33.



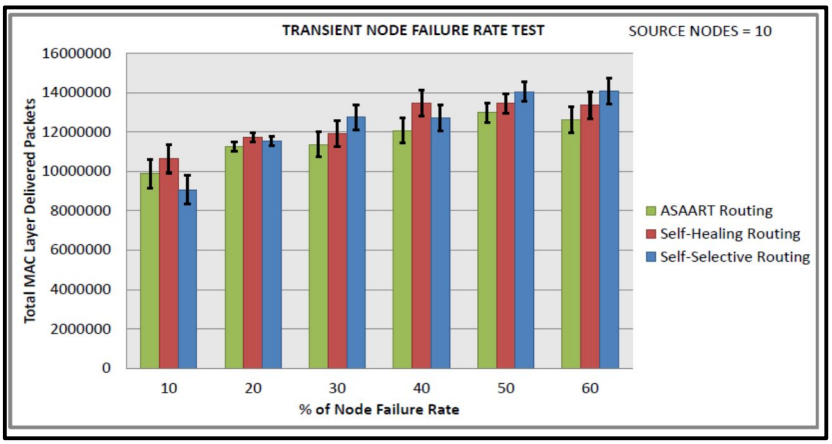
(a) Average packet delivery rate



(b) End-to-end network delay



(c) Energy consumption



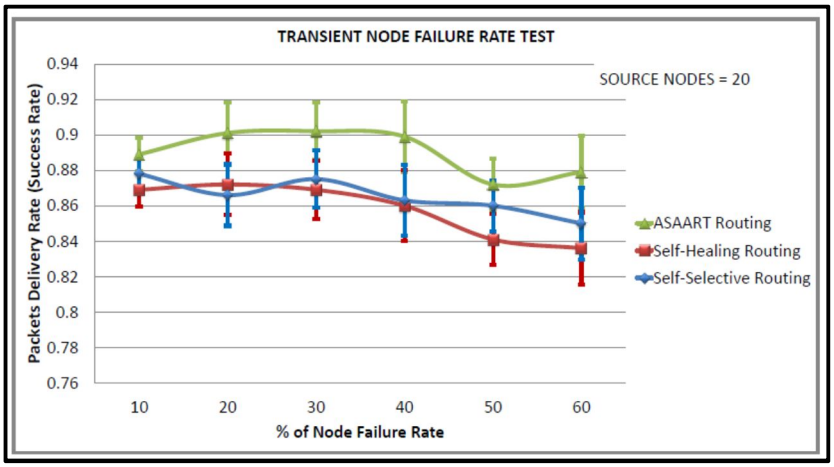
(d) Transmitted MAC layer packets

(Figure 30) Shows transient node failure rate test (fourth simulated scenario) with source node = 10

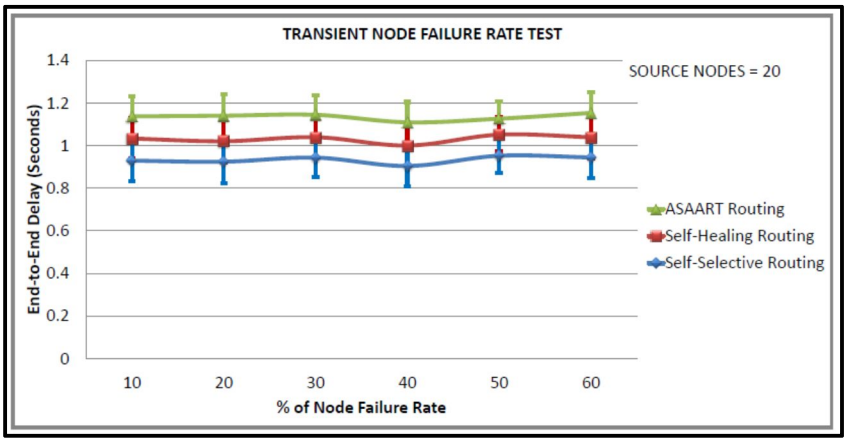
From Figure 30 (a-d), we can observe that the three routing protocols perform very well with a higher percentage of the packet delivery ratio. The three protocols achieve above 90% success rate because of less traffic congestion in the network. The ASAART performance drops to nearly 94% at 20% of the transient node failure rate. However, it slightly outperforms the SHR protocol when the transient node failure rate is between 30% and 60%, and the SSR protocol is between 50% and 60%. We observe that the ASAART protocol end-to-end delay is slightly higher than the SSR and SHR protocols. Nonetheless, this is negotiated by having the least number of MAC layer transmissions and energy consumption. ASAART attempts to minimize the retransmission of packets caused by flooding at the MAC layer, which makes it an energy efficient protocol.

Figure 31 (a-d) shows the performance of the three protocols with 20 nodes as

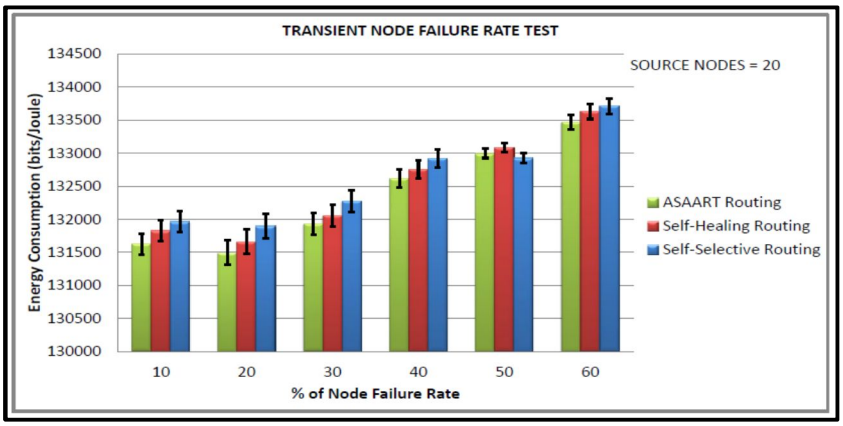
the source communicating node under different transient node failure rates. We can observe that there is high routing performance in terms of the success rate exhibited by ASAART. The ASAART protocol achieves a success rate above 90% when the transient node failure rate is between 20% and 40%. However, when the failure rate is between 50% and 60%, the success rate drops to 87%. This is at the expense of slightly higher end-to-end delay compared with the SHR and SSR protocols.



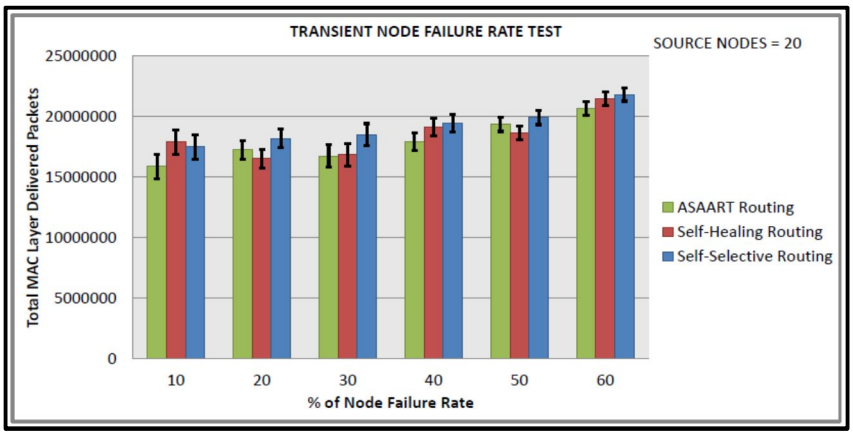
(a) Average packet delivery rate



(b) End-to-end network delay



(c) Energy consumption



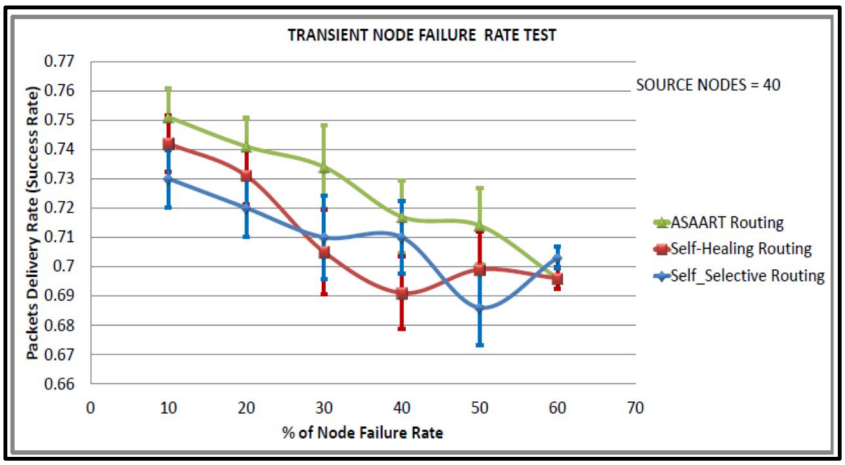
(d) Transmitted MAC layer packets

(Figure 31)Shows transient node failure rate test (fourth simulated scenario) with source node = 20

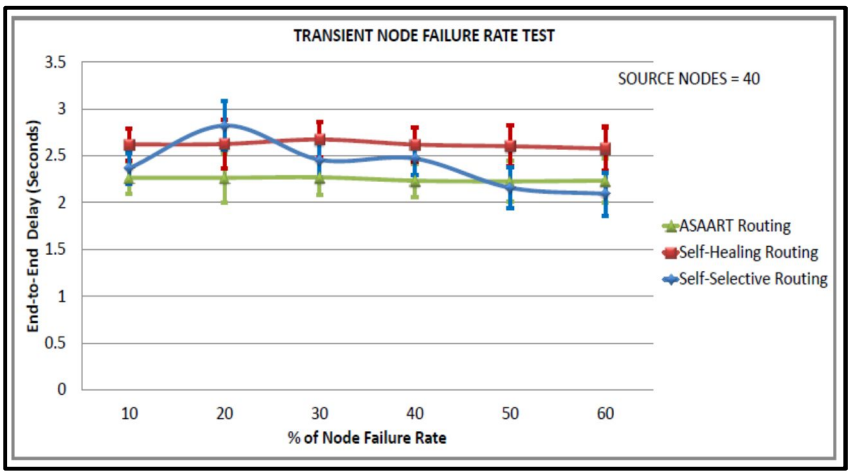
We can also observe that the SSR protocol success rate slightly outperforms the SHR protocol. This is because of its prioritized continuous back-off delay strategy. However, SSR shows slightly higher MAC layer packet transmission because of flooding compared with the SHR and ASAART protocols. It is interesting to observe that the SSR protocol shows a better reduction in the end-

to-end delay compared with the ASAART and SHR protocols. From Figure 32 (a-d), we can observe that the ASAART protocol outperforms both the SHR and SSR protocols at a transient node failure rate from 10% to 50%, respectively. The SHR protocol success rate drops to 69% when the transient node failure rate is 40%. However, at 50%, it increases to 70%. The SSR protocol end-to-end packet delivery drops to 67% at 50% of the transient node failure rate. Surprisingly, however, it outperforms both the ASAART and SHR protocols when the transient node failure rate is 60%.

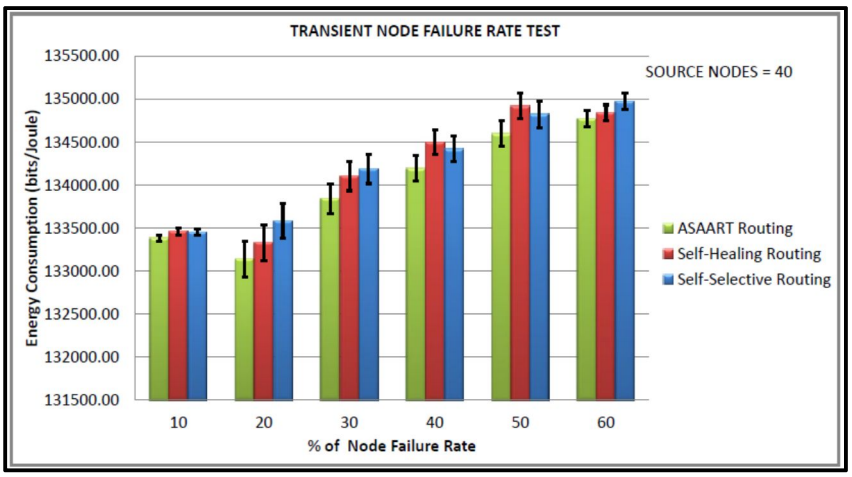
From Figure 32 (b), we can observe that ASAART end-to-end delay outperforms both the SHR and SSR protocols at a different percentage of the transient node failure rate. However, the SSR protocol outperforms ASAART at a transient node failure rate between 50% and 60%. The SHR protocol has the highest delay because of its route repairing mechanism. Too much time is spent repairing and retransmitting network packets. From Figure 32 (c-d), we can observe that ASAART uses slightly lower energy consumption and a reduced number of total MAC layer delivered packets compared with the SHR and SSR protocols. This reduces the network packet collision and contention, and saves energy. In conclusion, the ASAART technique demonstrates better fault tolerant capability in the presence of the transient node failure rate compared with the SHR and SSR protocols, particularly when there is too much traffic congestion in the network.



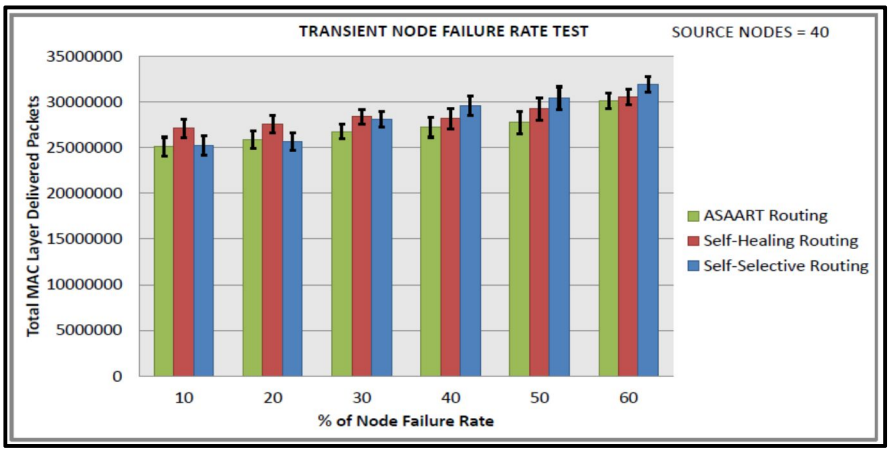
(a) Average packet delivery rate



(b) End-to-end network delay



(c) Energy consumption



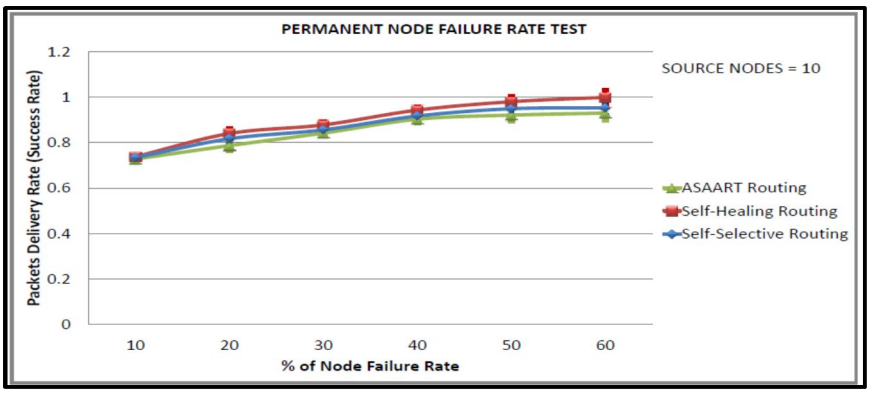
(d) Transmitted MAC layer packets

(Figure 32) Shows transient node failure rate test (fourth simulated scenario) with source node = 40

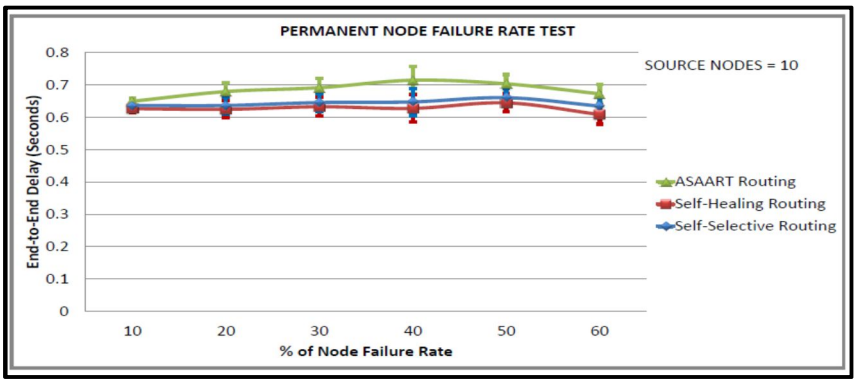
5. Permanent node failure test

In order to simulate the permanent node failure rate test, we set each sensor node to fail for a fixed instance of time. The start of each sensor node failure rate was chosen randomly at the start of the simulation using the uniform distribution of the failure rate start time against the time of failure occurrences [28,44,55], [96-97]. In this scenario, we consider a sensor network of size equal to 600 nodes. We set the transient link failure rate at 20%, and the source communicating nodes to be 10, 20, and 40 in order to ensure the protocol's performance under high and low network traffics. We varied the permanent failure rate from 10% to 60% in step of 10% in order to obtain the routing performances at different percentages of the permanent node failure rate. We conducted three different sets of simulations with a different number of seeds, and varied the number of source communicating nodes in each simulated scenario. The results of

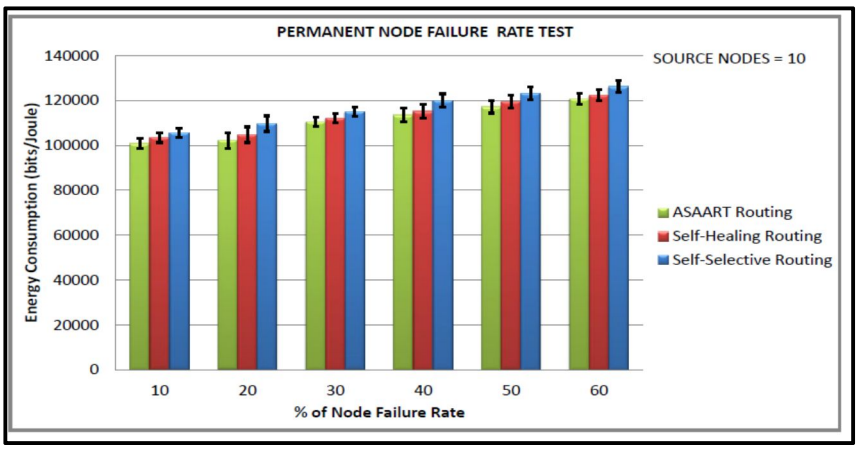
the permanent node failure rate with 95% confidence intervals are illustrated in Figures 33-35, which show the results of the permanent node failure rate performance evaluation of ASAART compared with the SHR and SSR protocols.



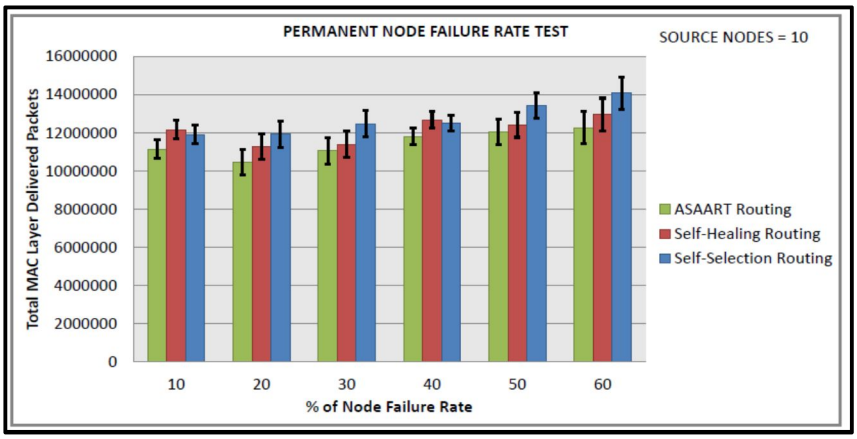
(a) Average packet delivery rate



(b) End-to-end network delay



(c) Energy consumption



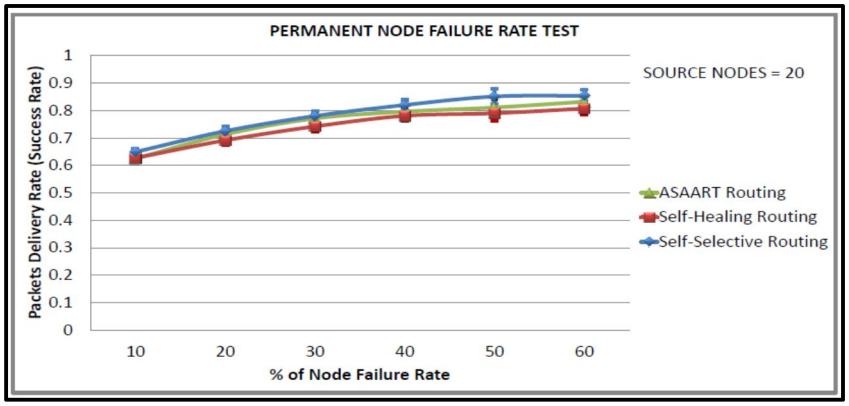
(d) Transmitted MAC layer packets

(Figure 33) Shows permanent node failure rate test (fifth simulated scenario) with source node = 10

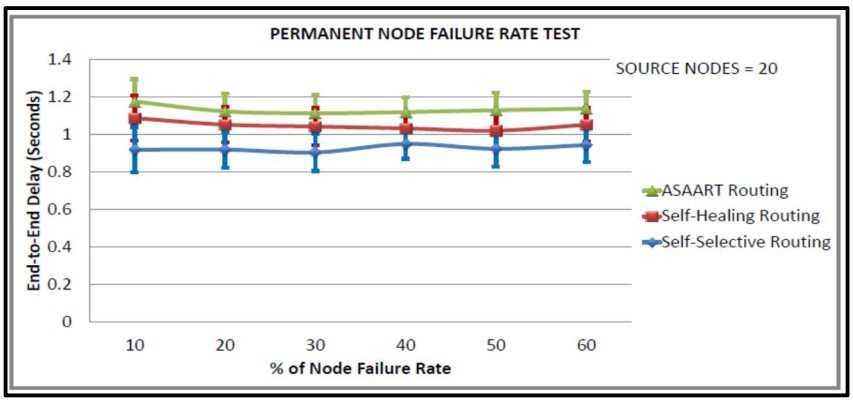
Figure 33 (a-d) shows the three routing protocols under permanent node failure rates with source communicating nodes of size equal to ten. Here, we can observe that because of less traffic congestion in the network, the three protocols compete very closely in terms of the end-to-end packet delivery ratio. All three protocols attain 90% to 95% efficient packet delivery with a permanent node failure rate between 40% and 60%. The ASAART protocol shows slightly higher end-to-end delay and low packet delivery rate compared with the SHR and SSR protocols. However, this is compensated by having a lower MAC layer packet transmission and low energy consumption. ASAART achieves lower MAC layer transmission because of its flooding control technique that attempts to reduce the retransmission of redundant network packets.

From Figure 34 (a-d), we can observe an increase in the packet end-to-end

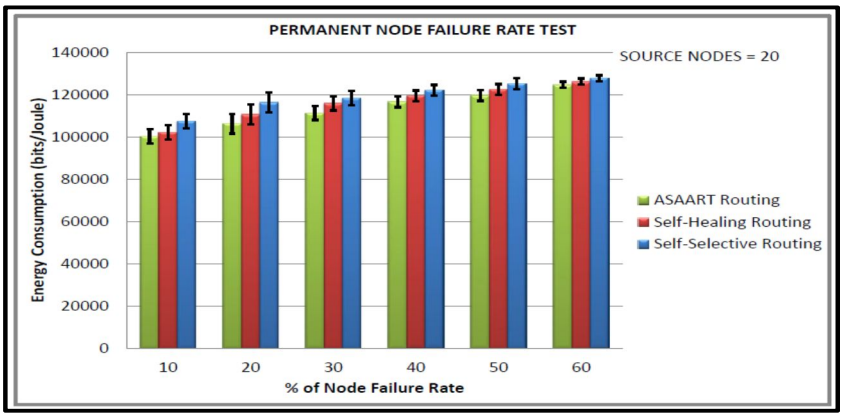
delivery rate exhibited by the three routing protocols when the source communication nodes equal 20 compared with the source node being equal to 40 nodes. The three routing protocols achieve a success rate above 80% when the permanent node failure rate is between 40% and 60%. This is attributed to less network traffic, which helps reduce the packets' contention and collision on the network. ASAART shows a higher end-to-end delay compared with the SHR and SSR protocols; this is because of its strategy of combined local and global network state information. However, ASAART exhibits better MAC layer packet transmission and energy efficiency compared with the SSR and SHR protocols.



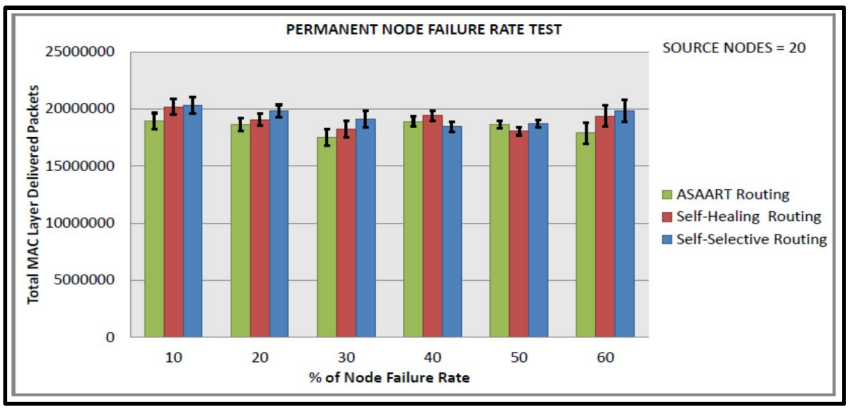
(a) Average packet delivery rate



(b) End-to-end network delay



(c) Energy consumption

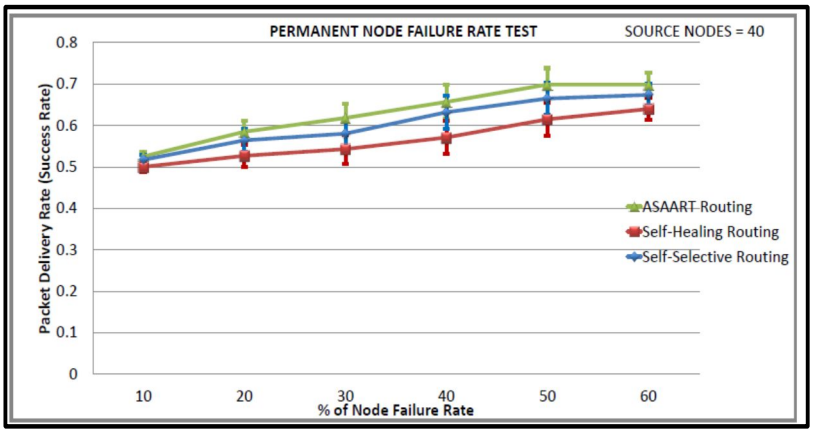


(d) Transmitted MAC layer packets

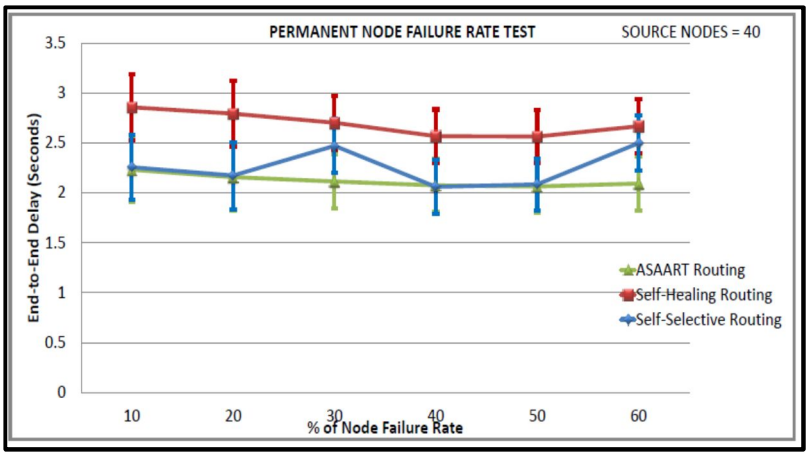
(Figure 34) Shows permanent node failure rate test (fourth simulated scenario) with source node = 20

In Figure 35, we can observe that the ASAART technique slightly outperforms both the SSR and SHR protocols. In Figure 35 (b), the SHR and SSR protocols exhibit higher end-to-end delay compared with the ASAART protocol. The ASAART protocol shows slightly lower end-to-end delay that remains almost stable when the permanent node failure rate is between 10% and 60%. ASAART uses slightly lower energy consumption compared with SSR and SHR, particularly at higher permanent node failure rates. This is because of its efficient route repair

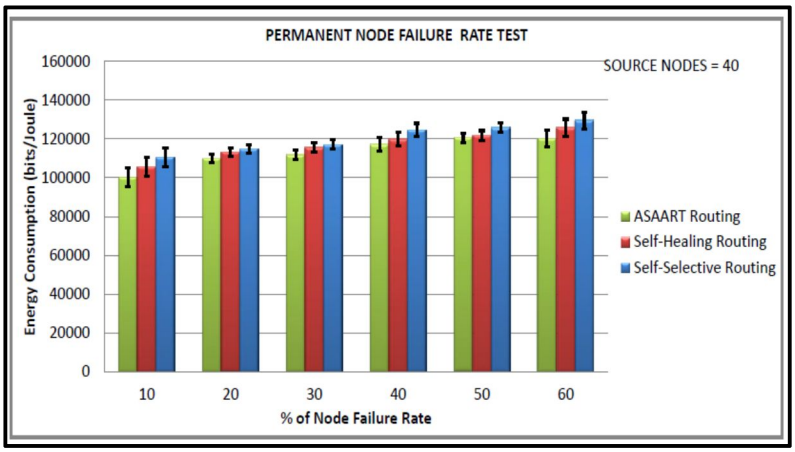
technique that ensures reliable end-to-end packet delivery. However, we can observe that ASAART uses the least number of delivered MAC layer packets at different percentages of the permanent node failure rate because of its efficient flooding control technique and faster routing decision that quickly converges to the minimum routing paths. In conclusion, ASAART protocol proves to be energy efficient and offer a high resiliency against faults and errors in the presence of permanent node failure rate and in highly congested and scalable network compared with SSR and SHR protocols.



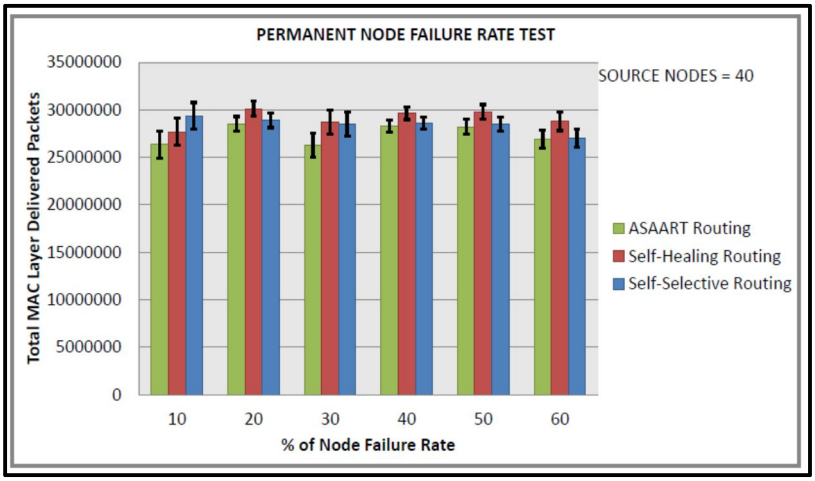
(a) Average packet delivery rate



(b) End-to-end network delay



(c) Energy consumption



(d) Transmitted MAC layer packets

(Figure 35) Shows permanent node failure rate test (fifth simulated scenario) with source node = 40

E. Chapter Summary

This chapter describes the proposed ASAART routing technique for wireless sensor networks to address the limitations of the SSR and SHR protocols. This is accomplished by integrating autonomous self-awareness and adaptive features

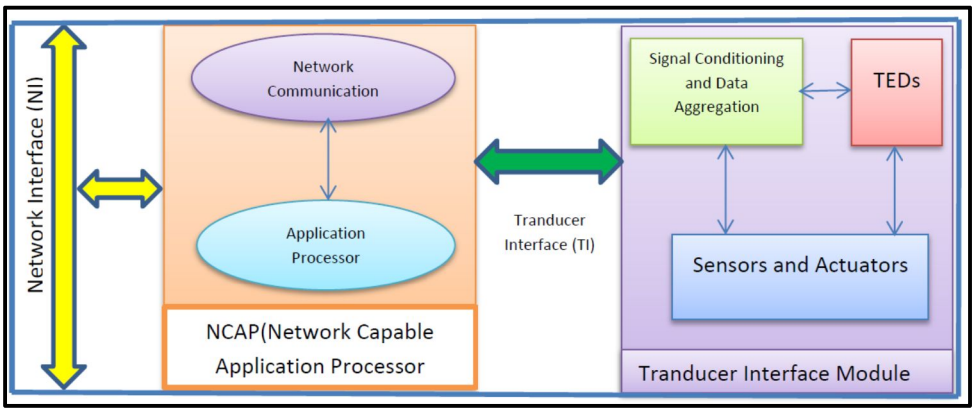
into the SHR protocol to make it more resilient to faults and errors and energy efficiency for efficient and reliable routing of sensor data in wireless sensor networks. The dissertation achieved this by combining both continuous and prioritized slotted back-off delay, and multiple randomize function techniques to obtain local and global network state information for faster routing path formation and speedy convergence to the minimum routing path. The chapter proposed a route repair technique for reliable transmission of sensor data in the presence of permanent and transient node failure rates, and efficient adaptation to simultaneous network topology changes. The chapter describes an extensive simulation under five different scenarios. The simulation results showed that the ASAART technique performed better in terms of high resiliency to errors and failure and better routing performance and energy consumption compared with the SSR and SHR protocols in the presence of transient and permanent node failure rates and in a highly congested, faulty, and scalable sensor network. The proposed approach is energy efficient because it consumes less energy compared to the SSR and SHR protocols.

V. FPGA-Based Design of an Intelligent On-Chip Sensor Network Monitoring and Control using Dynamically Reconfigurable Autonomous Sensor Agents

A. The IEEE 1451 Standard for Smart Sensor Interface

This chapter was written based on the published paper [175]. In this subsection, the dissertation describes the IEEE 1451 standard. The IEEE 1451 standard provides fast, efficient, and reliable means of converting transducer components into an intelligent sensor interface for efficient communication and transmission of sensor data to various NIs. A transducer is an electronic device that converts electrical signals from one form of energy to another. Therefore, a sensor is a special type of transducer that produces analog or digital electrical signals that represent the sensed physical, biological, or other environmental parameters. On the other hand, an actuator is a transducer that uses an electrical signal as an input and performs the required physical functionality [65-68]. An intelligent or smart sensor is a combination of both analog and digital transducer components, a central processing unit, and a network communication interface such as controller area network (CAN), inter-integrated circuit (I2C), local interconnect network (LIN), universal asynchronous receiver and transmitter (UART), etc. It is also composed of hardware or software and conditioning

circuits for sensor diagnostics and calibration, in addition to the communication mechanism and interface. Figure 36 shows the IEEE 1451 standard for a smart transducer interface (TI). The architecture, comprises four subsystems: transducers consisting of sensors and actuators, signal conditioning circuit, data-conversion subsystems, and application processor and network communication [65-68].



(Figure 36) Illustration of the IEEE 1451 standard for smart TI [65].

1. On-chip sensor network monitoring and control system

The on-chip sensor network environment is a very complex and dynamic system. Monitoring of the on-chip sensor network as a dynamic system demands communication of the sensed data to a controller and the basic information regarding the controller input parameters to the actuators. Therefore, the structure, topology, switching, and arbitration of the on-chip sensor network solely depend on the monitoring, control, and communication mechanism or protocol used by the system. Hence, two major challenges need to be addressed:

provision of efficient and reliable communication mechanism used by the network and provision of central and distributed control system that ensures real-time sensing, monitoring, and efficient processing of the on-chip sensory information. Therefore, an autonomous monitoring and control system for on-chip sensor network must consider the communication protocol and the constraints in relation to the clock time, network topology, switching, and arbitration [153-155].

On-chip sensor network monitoring and control can be regarded as a closed-loop system, which means that controllers are provided to connect to a feedback system, and sensors and actuators are installed to determine the overall function of the whole system. The on-chip sensor network must consider and address the following fundamental issues in a controlled loop system [107-112], [153-156].

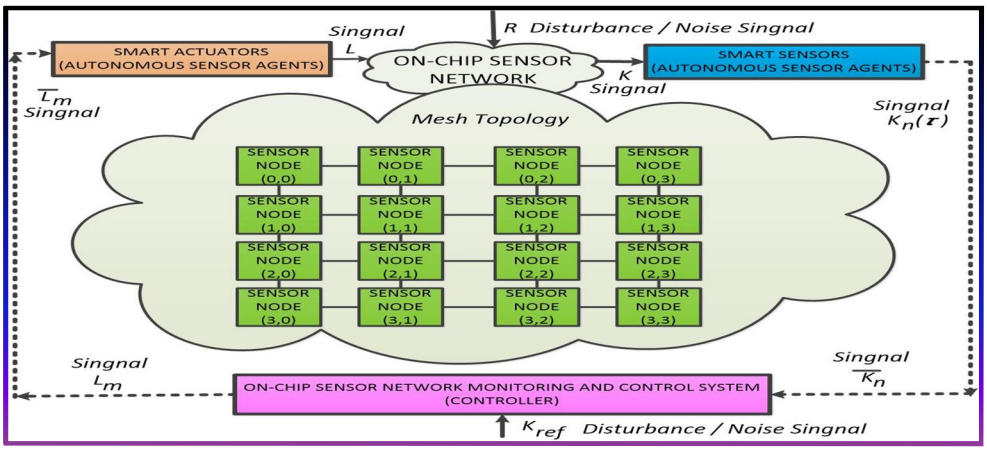
- Suitability of the on-chip sensor network communication topology for a particular application.
- Address the constraints imposed by the communication mechanism, such as packet loss, network latency, delay, routing, and switching, among other constraints.
- Determine the monitoring and control goals that must be observed to facilitate proper coordination of each component in the whole network monitoring and control system.

To address these issues, we must have an efficient and reliable monitoring and control infrastructure that will provide all the needed design methodology and framework to achieve the stated goals and objectives.

Figure 37 shows an intelligent monitoring and control system for on-chip

sensor network using autonomous sensor agents. It shows the data transfer among different nodes in the system, i.e., from the sensors to the controller and from the controller to the actuators. The on−chip sensor signal $K_n(\tau)$ is different from the on−chip network controller input signal $\widehat{K_n}(\boldsymbol{\tau})$, and the monitored controller output $L_m(\tau)$ is different from the actuator input $\widehat{L_m}(\tau)$. This information implies that signals $\widehat{K_n}(\tau)$ and $\widehat{L_m}(\boldsymbol{\tau})$ are the original network signals $K_n(\tau)$ and $L_m(\tau)$, respectively, which are transmitted through the communication links for the sensors, actuators, and controllers. Signals R and K_{ref} are disturbances or noise that must be controlled both in the input and output to obtain the desired results. Figure 37 shows single−headed arrow lines representing continuous time signals, whereas the dashed arrow lines represent the data transmission links at a given transmission time, e.g., τ_h, for h = 0, 1, 2,... During the transmission of the sensor data, the process produces a time delay that must be controlled and monitored. Similarly, time is another issue that needs to be considered, e.g., at what time interval is the communication needed and required. The transmission of the sensor data is constrained to a few number of bit rates, which implies that the values of signals $K_n(\tau)$ and $L_m(\tau)$ transmitted over the on−chip sensor network are constrained to a fixed bit length. Therefore, the monitoring and control system should ensure reliable and efficient communication at different time instances [153−155]. On the basis of these limitations, an FPGA−based design and implementation of an intelligent on−chip sensor network monitoring and control is

essential. Hence, the proposed design methodology in this dissertation provides a better solution for on-chip sensor network monitoring and control using autonomous sensor agents.



(Figure 37) Shows on-chip sensor network monitoring and control system using autonomous sensor agents

B. Proposed Design Methodology

In this subsection, the dissertation presents the design methodology of the proposed on-chip sensor network monitoring and control system with dynamic reconfigurable capabilities [113-169]. The proposed design detects runtime ambient parameters and communicates the sensed data to the monitoring and control hardware unit. Figure 38 shows the architectural design of the proposed monitoring and control system using autonomous sensor agents that reside in the NI to sense and report various parameter values to the hardware controller for onward analysis and processing. The architecture is a mesh network that is well

suited for on-chip sensor die structure [102-103], [164,166]. It consists of 12 nodes (gateway) and IP cores. The NI provides a means of communication among the sensors and actuators, network nodes, and hardware monitor controllers. At each NI, three intelligent sensors are installed, namely, voltage, thermal, and temperature sensors. These sensors continuously sense and transmit the sensed data to the hardware controller that then processes the data and provides the necessary actions to keep the system in a smooth and healthy condition [56-64], [104-110]. The sensors communicate the sensed data via a standard protocol. This protocol is shown in Figure 39.

Figure 38 shows that the intelligent monitoring and control system is composed of smart sensors and actuators that represent the autonomous agents, subcontrollers, main control unit, system monitor module, high-speed transceiver, and CAN controller. The monitoring and control system performs both central and distributed monitoring. Each sensor can communicate with its own cluster members and can perform data aggregation and forwarding to the main monitoring and control unit. Therefore, a distributed monitoring and control system is well established between the autonomous sensor agents and the main controller unit.

Figure 39 shows the real-time triggered protocol [64,116,165]. This protocol is deadlock and contention-free. The sensor nodes in each cluster are provided with real-time slots in which they receive sensed data from other nodes to communicate with the cluster head (subcontroller) or to transmit its own sensed

data to the subcontroller. Communication between the sensors is synchronized in real-time to avoid delay and loss of sensor data. The subcontroller triggers a real-time message to all its cluster members to query if sensed data are available for transmission. The members will respond to the cluster if they have data to transmit, and a real-time frame will be allocated and synchronized for each member to avoid communication delays and loss of sensor information. The explanation of the variables used in the proposed protocol shown in Figure 39 is presented below.

SYN: defines the real-time synchronization period between the cluster head (subcontroller) and cluster members.

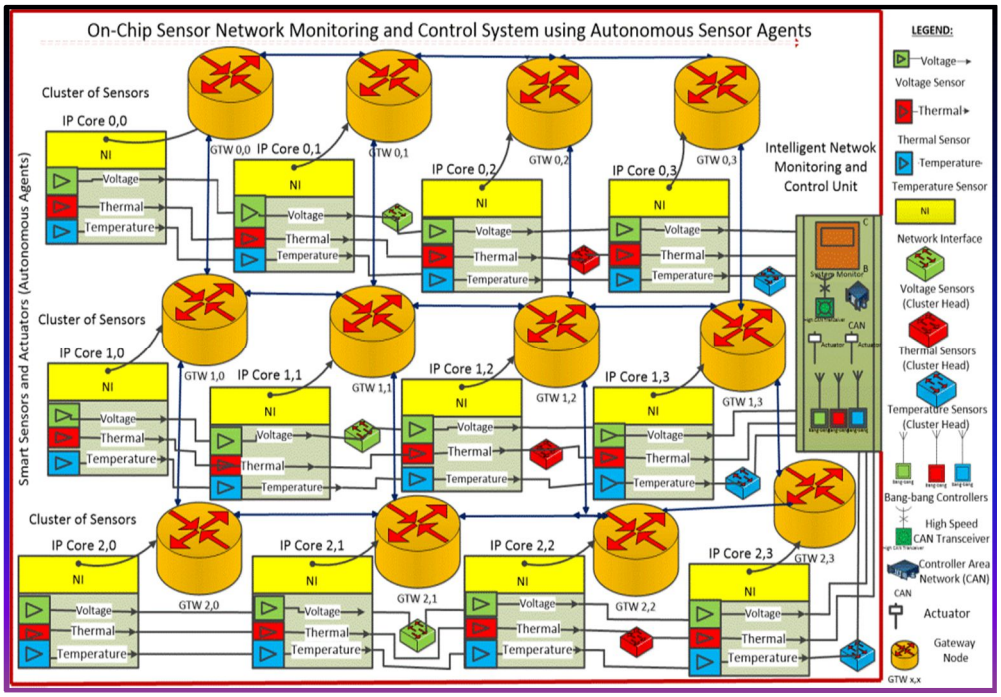
INST: defines the real-time interval between the transmission and reception of the sensor data.

BDCA: determines the real-time in which data are transmitted by the cluster head to the cluster members.

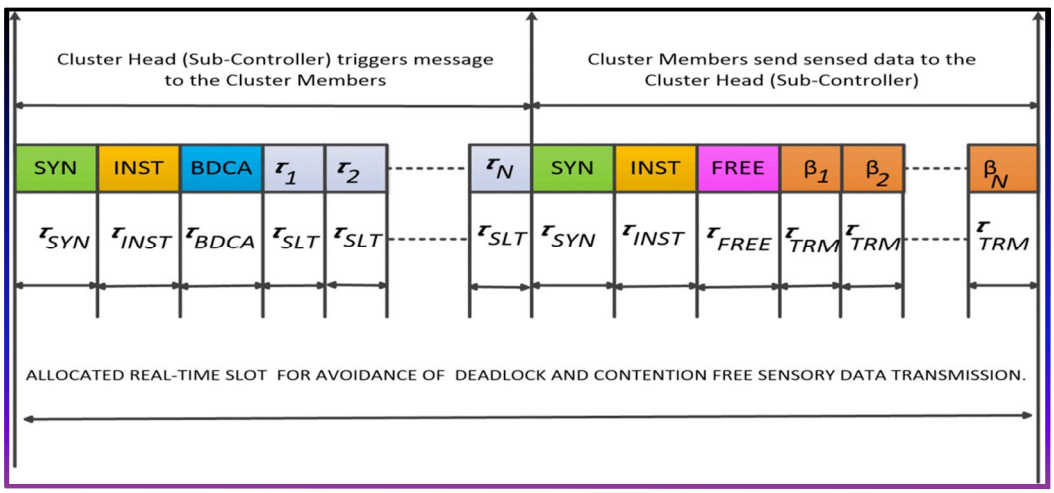
$\tau_{1\ ,\ldots,\ }\tau_n$: define the real-time slot during which the cluster members listen to instructions from the cluster head.

FREE: defined as a *real*-time slot in which the cluster head (subcontroller) receives data from the cluster members; the cluster head uses the FREE slot time to communicate with the on-chip monitor controller hardware on the status of the network.

$\beta_1,\cdots,\ \beta_N$: define the real-time interval in which the cluster members transmit data to the cluster head.



(Figure 38) Shows the proposed on-chip sensor network monitoring and control system using autonomous sensor agents on 4 × 3 mesh architecture



(Figure 39) Shows the proposed real-time-triggered on-chip sensor network communication protocol

1. On-chip sensor network monitoring and controller interface design

Figure 40 shows the logic design of the main on-chip sensor network monitoring and controller unit. The design consists of several components that form the controller unit and interface to the different sensor networks. The main monitoring and controller design include the following: three bang-bang (on/off) controllers, a time-domain system analyzer implemented as the main monitoring unit, an 8:1 multiplexer (MUX) interface, an ADC, smart sensors, actuators, a high-speed CAN transceiver, autonomous voltage sources, a CAN engine controller, clock sources, and voltage-monitoring circuits.

The reference voltage source supplies the required voltage to power the controllers and all the associated components of the monitoring and control system. The source is configured to generate different output sources such as a step and a periodic pulse. The initial pulse parameter and the amplitude values are set to 0.0 and 0.5V, respectively. The 8:1 MUX interface provides the input from the output of the different sensors to the main controller system. It also acts as a conditioning circuit. Different sensor signals are conditioned (aggregated) to form a single strong analog signal, which is then amplified or buffered, as shown in the logic design. The use of a buffer is beneficial for interfacing with the sensors. It buffers the voltage to be fed into the ADC input. The output of the MUX is then fed into the ADC, which converts the analog signal into a digital form.

Two clock sources are connected to the ADC to provide a clocking mechanism for data conversion and transmission. The clock sources of the ADC are configured with the following settings: on−time = 10.0 μs, off−time = 10.0 μs, initial time = 0 μs, and delay = 0.0 μs). Similarly, the clock source connected to the start pin of the ADC is configured with the following settings: on−time = 50 μs, off−time = 20 μs, initial time = 1 μs, and delay = 10 μs. These parameter values can be tuned to control both the input and output of the ADC. The output of the ADC is fed into an AMIS−3660 high−speed CAN transceiver, which serves as an interface between the ADC and the CAN engine. The *eoc* pin of the ADC is connected to the *s* pin of the CAN transceiver. The 8−bit digital output of the ADC is connected to the receiver (Rxd) input of the CAN controller. The Vcc input is connected to the Vcc source, and the ground pin is connected to the earth (ground). The transmitter (Tx) pin of the CAN transceiver is connected to the Tx pin of the CAN engine. The receiver (Rx) pin of the transceiver is connected to the Rx pin of the CAN engine, which provides a means of transmission and re−transmission of sensor data and direct monitoring and control of sensor information [108], [157−158], [167].

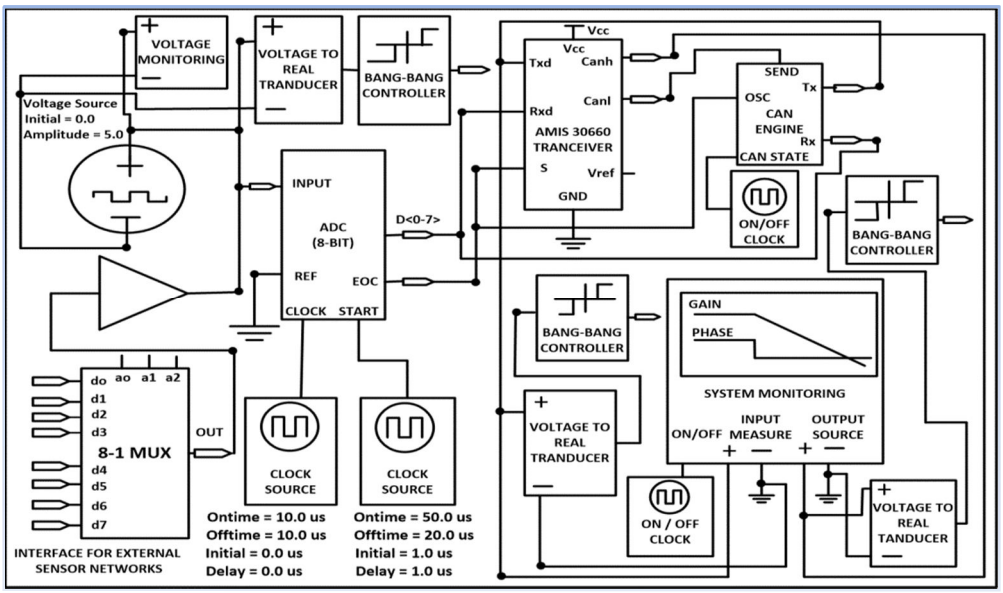
The CAN engine, together with the transceiver, provides a communication interface to the entire system and serves as an NCAP, as defined by the IEEE 1451 smart TI standard [65−68]. The voltage−monitoring module block measures the voltages from the sensor input to the control unit. We set a limit for

the upper and lower voltages. If the voltage falls below a certain level (0-5 V), it is reported to the control system, and appropriate actions are taken. Smart voltage transducers are attached to the controller input and output pins to monitor the variation in the input and output voltages from the sensors and actuators to observe and control the transmitted information.

The time-domain system analyzer is the main monitoring interface that provides voltage analysis of the converted signals from the sensor NI. The voltage can be swept within a given frequency range while performing both transient and time-domain analysis of the monitored control system. The controller compares the converted digital input signals from the CAN transceiver and the output signal from the CAN engine. It then calculates the phase and gain responses of the voltage frequencies during the sweeping process. This information is fed into the simulator, and waveform plots of the signals are displayed in the waveform window of the simulator for further analysis. Figure 40 shows two actuators attached to the system analyzer input and output ports to measure the actual transmitted sensor signals. This process will ensure that exact sensory information is transmitted and received at both input and output of the monitored and control system.

The bang-bang (on/off) controllers were used to control and monitor the sensor information between the sensor interface and the main hardware controller unit. We set the positive_threshold voltage to 5.0 V and the constant_output_value

to 1.0 V in order to control the output voltages from the sensors, actuators, and main control unit [157-161]. The pseudocode for the bang-bang controller is shown in Figure 41.



(Figure 40) Shows on-chip sensor network monitoring and controller hardware interface design

```
1.  Pseudocode for the bang-bang controller module operation
2.  //*****************************************************//
3.  Var  Positive_Threshold_value = 5.0v;
4.  Var Constant_output_value = 1.0 v;
5.  Begin {
6.  If Controller_Input  >= Positive_threshold
7.       Then
8.            Controller_Output = Positive_Constant_value;
9.  If Controller_Input <= Negative_Threshold
10.     Then
11.        Controller_Output = Negative_Constant_Value;
12.   Else if Controller_Input  is between Positive_Threshold  and Negative_Threshold
13.      Then
14.         Controller_output   = 0;
15.       End if
16.    End if
17.  End if
18.  End
```

(Figure 41) Illustration of the pseudocode for the bang-bang (on/off) controller module operation

2. On-chip temperature sensor NI design

Figure 42 shows the on-chip temperature sensor NI design. The design consists of several components: an autonomous temperature source, four smart transducers (temperature sensors), four smart TIs to the temperature sensors, four operational amplifiers, a 4-1 MUX, an ADC, and two clock sources.

The temperature source is the main source of thermal energy for the sensors and the TI, which generates a temperature difference across its input ports in a form of a pulse or a pulse train. Furthermore, the positive polarity of the source defines the temperature differential between the highest ($Temp_{high}$) and lowest ($Temp_{low}$) values of the port of the temperature source [66-68],[157-161]. The temperature differential is expressed by the following equation:

$$Temp_{diff} = Temp\left(Temp_{high}\right) - Temp\left(Temp_{low}\right) \quad (26)$$

where $Temp_{diff}$ is the temperature difference and $Temp\left(Temp_{high}\right)$ is the temperature in degree Celsius at the port $(Temp_{high})$. $Temp\left(Temp_{low}\right)$ is the temperature at the port $(Temp_{low})$. The two parameters of the temperature source are set at the following conditions: initial = 0.0 V and pulse = 5.0 V. This condition provides a continuous temperature source for the four smart temperature sensors and the TI. The control sensor senses the temperature signals from the source and converts them into a variable signal. The temperature model is expressed by the following equation:

$$Output_{temp} = R * (Temp\left(Temp_{high}\right) - Temp\,(Temp_{low}) \quad (27)$$

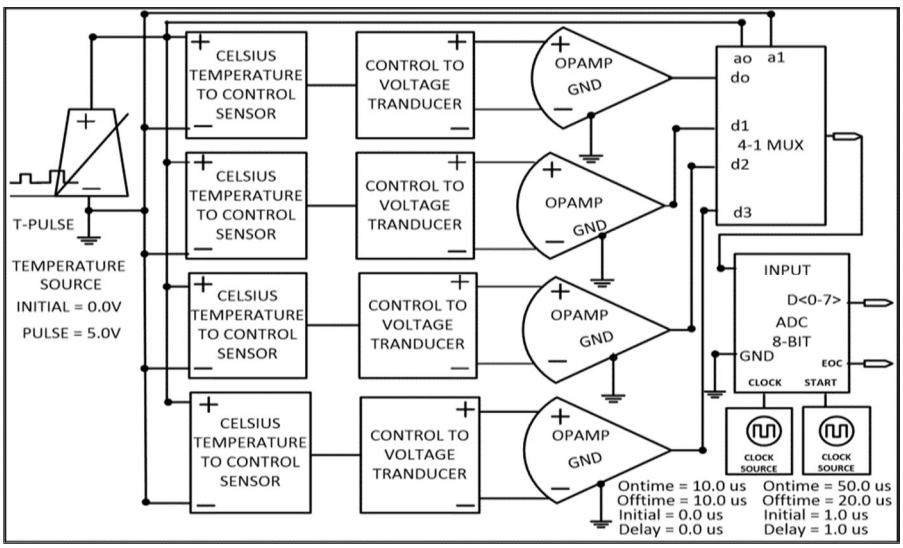
where the two pins $(Temp_{high})$ and $(Temp_{low})$ are terminal pins. The difference in the two temperature values defines the input signal. Parameter R is the gain, which can take any arbitrary values. Here, we set the default value of R to be one. The output parameter has no specific signal value. Therefore, we need a TI to convert the output signal into an electrical signal. To achieve this, we interface the output of the temperature sensor with the transducer (control to voltage interface) to convert the temperature into electrical signals. The TI is based on the characteristics expressed by the following equation:

$$\mathrm{Input}_{\mathrm{Sensor}_{\mathrm{Signal}}} = R * (\,Volt\,(\beta) - Volt\,(\alpha)) \quad (27)$$

where the difference $Volt\,(\beta) - Volt\,(\alpha)$ is the differential voltage from the sensor output and R is the gain that is set to one. The TI output is converted into a voltage and amplified using an operational amplifier. The converted output voltage from the TI ($\mathrm{Input}_{\mathrm{Sensor}_{\mathrm{Signal}}}$) is used as an input to the operational amplifier because the operational amplifier is suitable for interfacing the sensor data and buffering the input voltage for maximum gain. The amplified voltages from the four operational amplifiers are used as input to the 4−1 MUX, which serves as a conditioning circuit for the sensor interface. The four inputs are multiplexed to form a single output that is fed into the ADC for conversion into digital signals. The 4−1 MUX has four data inputs, namely, d_0 to d_3, and two address input pins a_1 and a_0, as shown in Figure 42. The 4−1 MUX functions as follows: one of the

four inputs is chosen to drive the output state *y*. The selection is based on which among the four possible input combinations are available [56−65],[157−161].

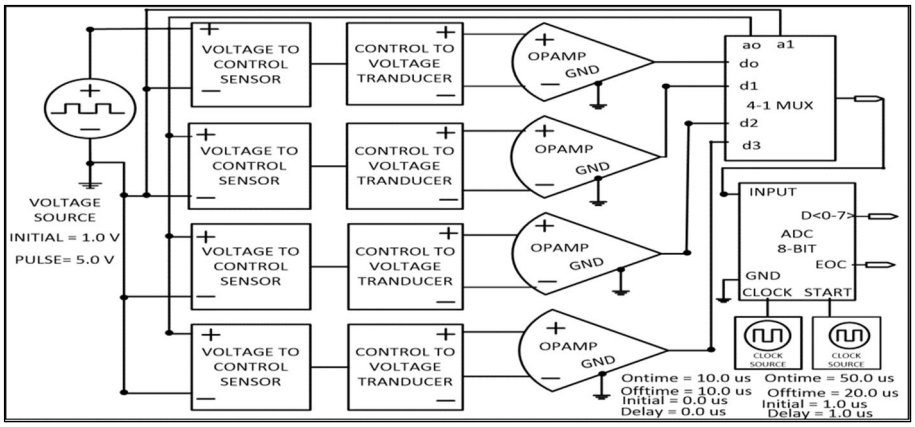
The output of the 4−1 MUX (*y*) is used as an input to the ADC. The ADC converts the analog sensor data into a digital form. However, the output (*y*) from the 4−1 MUX can be sent to the main controller representing one cluster of the four sensor data. In this case, the conversion from analog to digital is performed by the ADC on the main monitor controller unit. To ensure timely and accurate data conversion by the sensors, two clock sources are connected to the *start* and *clock* pins of the ADC. The clk1 source is configured with the following parameter configurations: on−time = 10.0 μs, off−time = 10.0 μs, initial time = 0.0 μs, and delay = 0.0 μs. The start clock source is configured with the following settings: on−time = 50 μs, off−time = 20 μs, initial time = 1 μs, delay = 1.0 μs.



(Figure 42) Shows on−chip temperature sensor NI design

3. On-chip voltage sensor NI design

Figure 43 shows the design of a voltage sensor NI. The design consists of autonomous voltage source, smart voltage sensors, four TIs, four operational amplifiers, one MUX, an ADC, and two clock sources. The autonomous voltage source is used as the main source for power supply to the voltage sensors and TIs. The two parameters of the voltage source are configured to supply continuous voltage to all the components in the system. The values of the parameters are as follows: initial = 0.0 V and amplitude = 5.0 V. The characteristic equations for the voltage sensors (voltage to control interface) and the TI (control to voltage) are similar to equations (26)-(28). The output of the TI is fed to the operational amplifiers, which is then amplified and sent to the d_0 to d_3 input pins of the 4-1 MUX [66-68],[157-161].



(Figure 43) Shows on-chip voltage sensor NI design

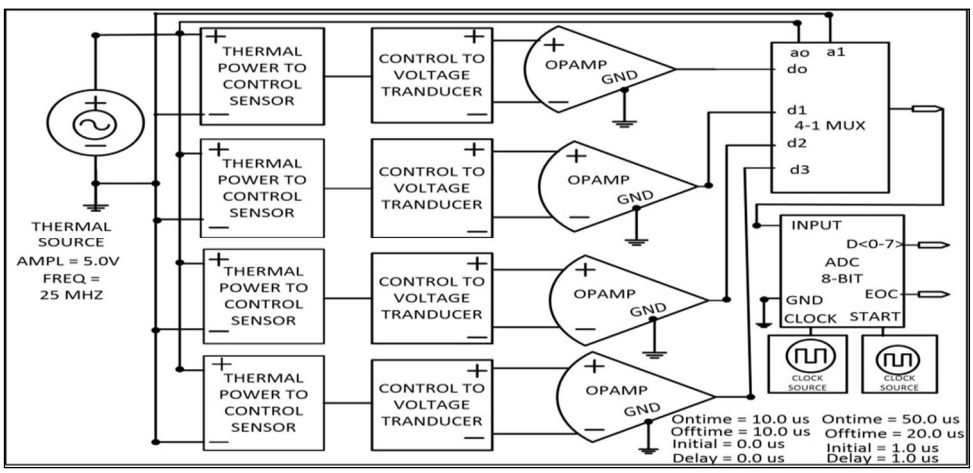
4. On-chip thermal sensor NI design

Figure 44 shows the smart thermal sensor NI of the main controller unit. The interface consists of several components similar to the two interfaces for the

temperature and voltage sensors described in the previous subsections. A thermal power source is used to power-on the four sensors and the TI. The two parameters for the thermal power source are configured with the following settings: amplitude = 5.0 V and frequency = 25 Hz. The source generates a sinusoidal heat at its input ports. The thermal sensors (thermal to power control) sense and transmit the sensor data to the smart TI. The characteristic equation for the thermal sensor model is expressed as

$$Output_{Sensor_{data}} = K * (\, flux\,(Thermal_{high} - Thermal_{low}) \qquad (29)$$

where $Thermal_{high}$ and $Thermal_{low}$ are the thermal pins with the temperature, as well as the thermal power flux, is expressed in degree Celsius. The input signal to the sensor is expressed by the function $flux\,(Thermal_{high} - Thermal_{low})$. Parameter K is the gain constant. We set $K = 1$ during the simulation. The sensor output is then sent to the input port of the TI (control to voltage), which converts the sensed data into electrical voltage and then amplifies them using the operational amplifiers and used as input by the 4-1 MUX [157-161], [167].



(Figure 44) Shows on-chip thermal sensor NI design

C. On-Chip Sensor Network Simulation Results

1. Simulation procedure

In this subsection, the dissertation explains the detailed simulation procedure carried out to verify the proposed design. The design and implementation of the register transfer level (RTL) modules, models, and schematics of the proposed intelligent network monitoring and control system presented in this chapter were performed using the Synopsys integrated design environment and tools [157-158]. We verified the functionality of the proposed design specification in the time domain using transient analysis to determine the variation in the sensor NI voltage signals and that in the monitoring and control unit over time. The simulation was run by setting the following parameter configurations:

Simulation time (end time) = 10,000 ms (10 s)
Time step = 10 ns
Start time = default = 0 s
Threshold voltage = 0-5 V
ADC clock signal setting: on-time =10 μs, off-time = 10 μs, initial = 0 μs, delay = 0 μs), ADC start signal setting (on-time = 50 μs, off-time = 20 μs, initial = 1 μs, delay = 1 μs).

The simulation time (end time) defines the time in which the simulator finishes the transient analysis. Meanwhile, the time step is the period in which the simulator finds an initial solution point during the simulation to project the next solution point in the simulation process. In addition to the end time of the

simulation process, the simulator also outputs the execution time. Four RTL gate-level netlist files for the entire design are provided with a simulator during the compilation and simulation process. Three netlist files are provided: one for each NI and the other one for the monitoring and control unit. The simulation results in waveforms are shown in Figures 45 and 46.

2. Discussion of simulation results

Herein, the dissertation provides the simulation results in waveforms using the Synopsys integrated design environment and tools [157]. Figures 45 and 46 show the transient analysis results in the time domain of the variation in sensor voltages over time for the different parts of the monitoring and control unit.

Figure 45 (a) -(c) shows the transient time-domain analysis for the three on-chip sensor NI designs. Figure 45 (a) shows the simulation results of the voltage sensor NI and the voltage signal variation at different time instances. We observed that the sensor signals are alternating with an amplitude of 0-5V corresponding to the reference voltage of 5V. The four sensor signals are combined (multiplexed and amplified) to form an aggregated signal. As expected, the aggregated sensor signals are measured to be 5.0 V, which shows a high accuracy in the measurement of the combined sensor signals with a very negligible error in the sensor measurement.

Figures 45 (b) show the variation in the sensor voltage at different time instances for the thermal sensor NI. As observed, no difference exists between

the reference temperature source (voltage) and the measured voltage of the four aggregated sensors. As expected, the reference voltage and the measured sensor voltages are the same (i.e., 5.0 V), which shows accurate precision in the sensor measurement of the NI and the efficiency of the intelligent thermal sensor network design.

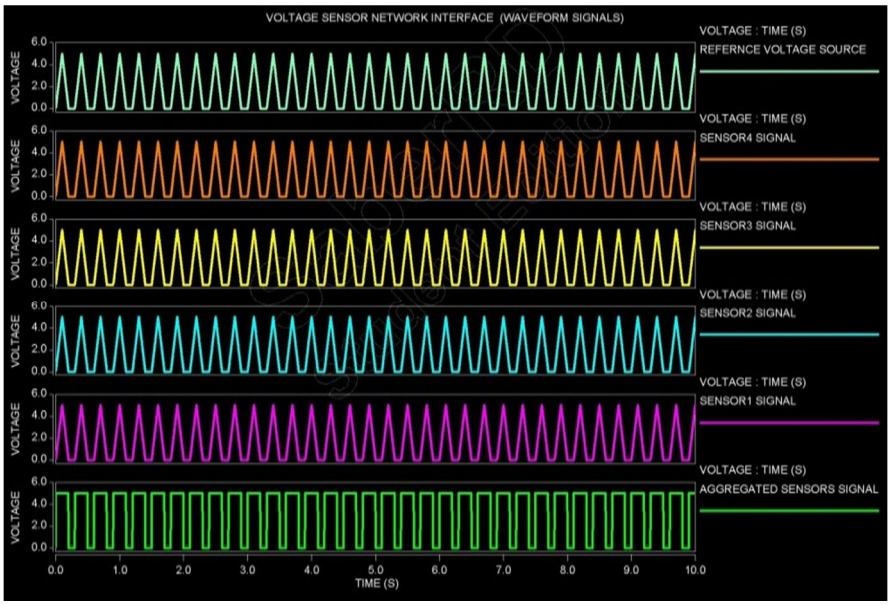
Figures 46 (a) and (b) show the simulation results for the variation in the sensor signals (voltage) over time for the main monitoring and control unit. Figure 46 (a) shows the measured analog signals at different time instances of the various components of the monitor controller unit compared with the input sensor measured signals from the sensor NIs. The figure shows that the bang-bang_1 output signal oscillates between 0 and 5 V (maximum), which indicates the same measured input sensor signals of 5 V.

The transceiver_txd signal remains stable and is measured to be 5 V at different time interval. This indicates a continuous conversion of signals and a high accuracy in the sensor signal conversion from analog to digital by the ADC. The output signals for both the high-speed transceiver and the CAN controller engine are controlled and regulated by the bang-bang_2 and bang-bang_3 controllers. We can observe that the controlled voltage measurement of the bang-bang_2 controller is 5 V measured at different time intervals. The bang-bang_3 controller voltage connected to the system monitor unit to control the monitor input and output is also measured to be 5 V, which shows that the input sensor

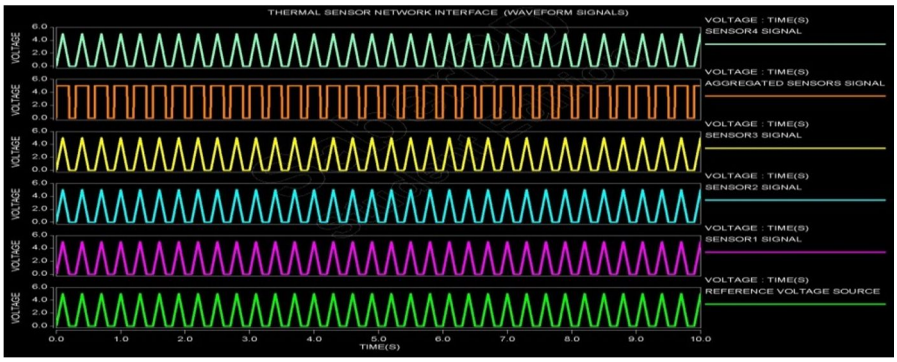
signal from the NI is well monitored and performs control without losing sensor precision and accuracy.

Figure 46 (b) shows the digital signal output of the converted sensor signals by the monitoring and control unit. The different digital bits of the converted sensor analog signals are output through the digital bits (0-7) of the ADC output. The transceiver signal was observed to be stable and high at different time instances.

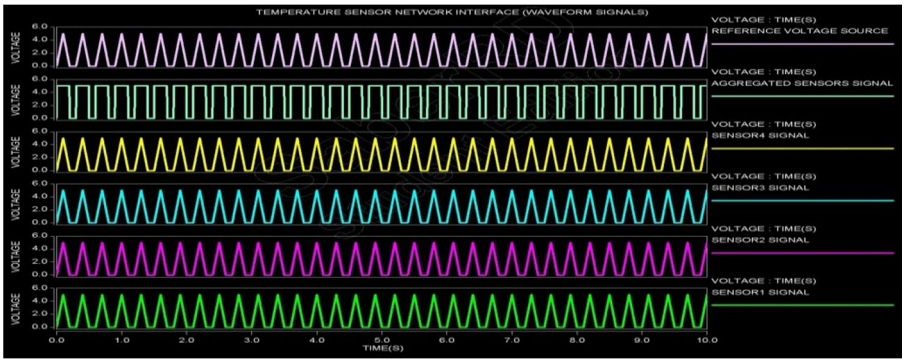
In conclusion, the simulation results verified the functionality of the various design components of the sensor NIs and the monitoring and control unit. The actual application of this proposed approach is implemented using FPGA as a case study. The details of the case study and FPGA implementation are presented in the experimental results subsection.



(a) For voltage sensors

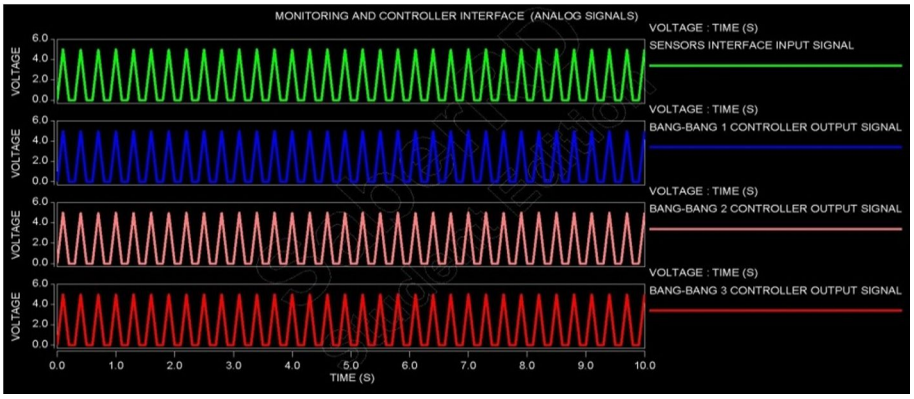


(b) For thermal sensors

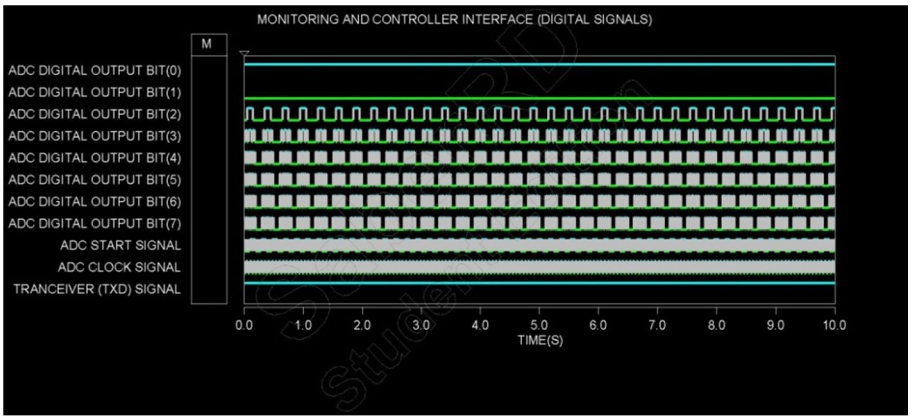


(c) For temperature sensors

(Figure 45) Illustration of the simulation waveforms for transient analysis of the variation in the reference voltage and the measured voltages over time



(a) For analog signals



(a) For digital signals

(Figure 46) Ilustration of the simulation waveforms of the transient analysisthe voltagesvariation in the voltage source and measured voltages over time.

D. FPGA On-Chip Sensor Network Experimental Procedure and Case Study

1. Experimental setup

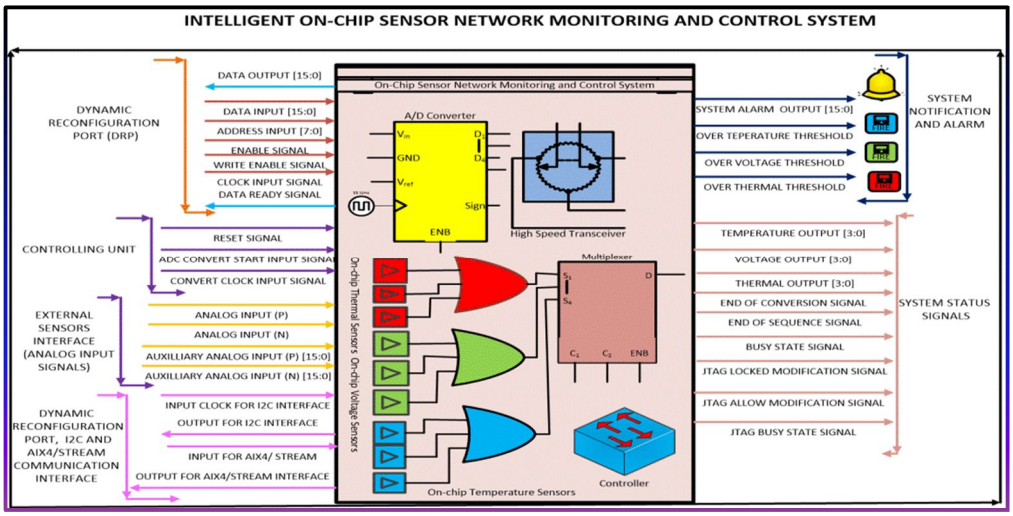
In this subsection, the dissertation explains the experimental procedure used to design, implement, and verify the proposed on-chip sensor network monitoring and control system using an FPGA. To validate the approach through a real-life scenario, we implemented a case study for on-chip sensor network monitoring and control using FPGA. We used the Xilinx Vivado 2013.3 integrated design environment and the Zynq 7000 XC7z020 CLG484-1 AP SoC FPGA device for synthesis, placement, routing, and implementation of the proposed system. Figure 47 shows the on-chip sensor network monitoring and control unit implemented

using Xilinx ADC (XADC) and SYSMON IP cores [160-161]. The XADC and SYSMON IP cores provide an interface for designing on-chip sensors and user-defined external sensors to the main monitoring and control unit. The system provides capabilities for self-awareness and adaptation to dynamically tune the different parameter values at runtime to measure ambient parameters such as temperature, voltage, and thermal energy variations. It is based on a highly precise analog measurement system consisting of a 12-bit ADC and other circuit elements. Several input and output pins are provided to achieve high performance and accurate measurements from the on-chip sensors and the remote sensors connected to the system interface [113-122], [159-163].

Figure 48 shows the RTL netlist schematic after synthesis, placement, and routing of the implemented monitoring and control system on Zynq 7000 XC7z020 CLG484-1 FPGA board using the Xilinx Vivado 13.3 integrated design environment.

The architecture shown in Figure 47 consists of the Xilinx XADC and SYSMON IP interface that are present in all Xilinx 7 series FPGA devices, including the Zync 7000 families [160-161]. The central processing unit is based on ARM cortex-A9 processing system and is used to monitor and control the on-chip sensor variation in the measured temperature, voltage, and power supply. Similarly, the remote sensors connected to the main unit can also be monitored and controlled. Furthermore, the system will set an alarm when over temperature,

over voltage, and under voltage, etc. occur. This feature will ensure smooth running and proper functioning of the whole system.

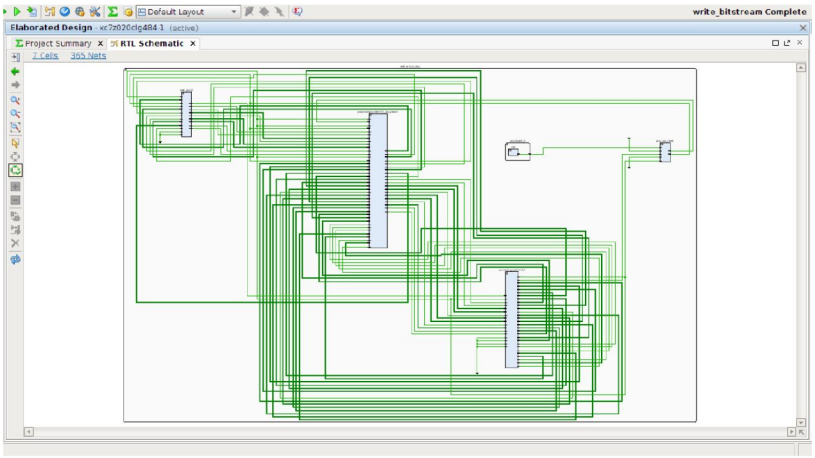


(Figure 47) Shows the block design of on-chip sensor network monitoring and control system using XADC and SYSMON IP cores

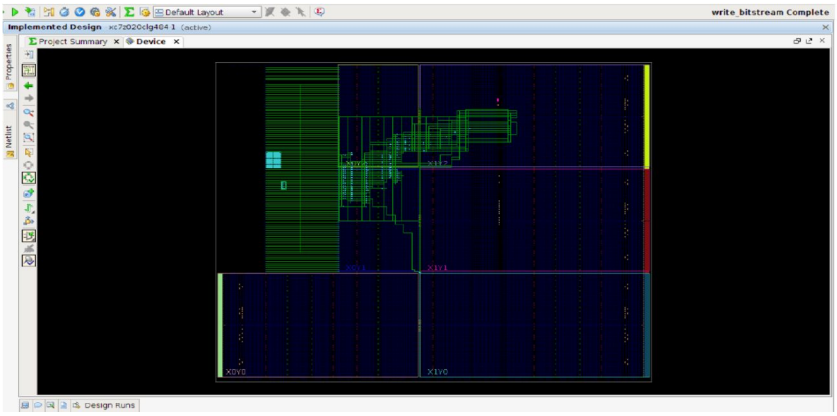
2. Case study

In this subsection, the dissertation provides the detailed explanation of the proposed case study using the Zynq-7000 FPGA board [160-164]. This case study uses a central processing unit (ARM cortex-A9 core) to monitor and control the variation in temperature and voltage of the on-chip sensor and remote sensors connected to the FPGA fabric. The Xilinx Zynq-7000 is a flexible and efficient architecture based on all programmable system-on-chip architecture. The device consists of a dual-core ARM Cortex-A9 MP Core processing system and enhanced Xilinx programmable logic in one integrated FPGA fabric. In the proposed design, the processing system is the main monitoring unit that provides

basic functionalities with the Cortex-A9 MP Core central processing unit. The XADC acts as the control unit, which works together with the processing system. The other components installed in the processing system include the on-chip memory, on-chip sensors, input and output devices, auxiliary memory, and interface to the remote-sensing devices.



(a) RTL schematics



(b) Routing resources of the Zynq 7000 AP SoC FPGA device

(Figure 48) Illustration of the on-chip sensor network monitoring and control system.

The monitoring and control system provides a communication using different interfaces such as I2C, DRP, and JTAG. The control system uses another form of communication interface, which is the industrial input and output (IIO) framework-based Linux driver [161-163]. This driver works as a device driver for an application that uses a control system and the AXI and DRP communication interfaces. An interesting functionality of this driver is its ability to configure the control system to be used in different functionalities, e.g., receiving data from the control system and providing information for different parameter states and configurations in the user space provided in the intended application. In this case study, we demonstrate how the IIO-based Linux driver is used for on-chip sensor network monitoring and control application. We show how the control system can be used to provide a hardware design in the programming logic, which creates a dedicated communication between the control and monitoring system using the DRP and AXI port communication interfaces. To make the application more user friendly, a web server-based design is provided for the user to interact with the monitoring and control system [60-107], [160-164]. The results of the dynamic monitoring and control system are displayed on a webpage for the users to see and change the configuration parameters to observe the variations in the measured on-chip temperature and voltage parameters, as shown in Figures 49 and 50. The architecture of the hardware monitoring and control system as well as the connectivity between the control system in the programming logic and the

processing system is shown in Figs. 49 and 50.

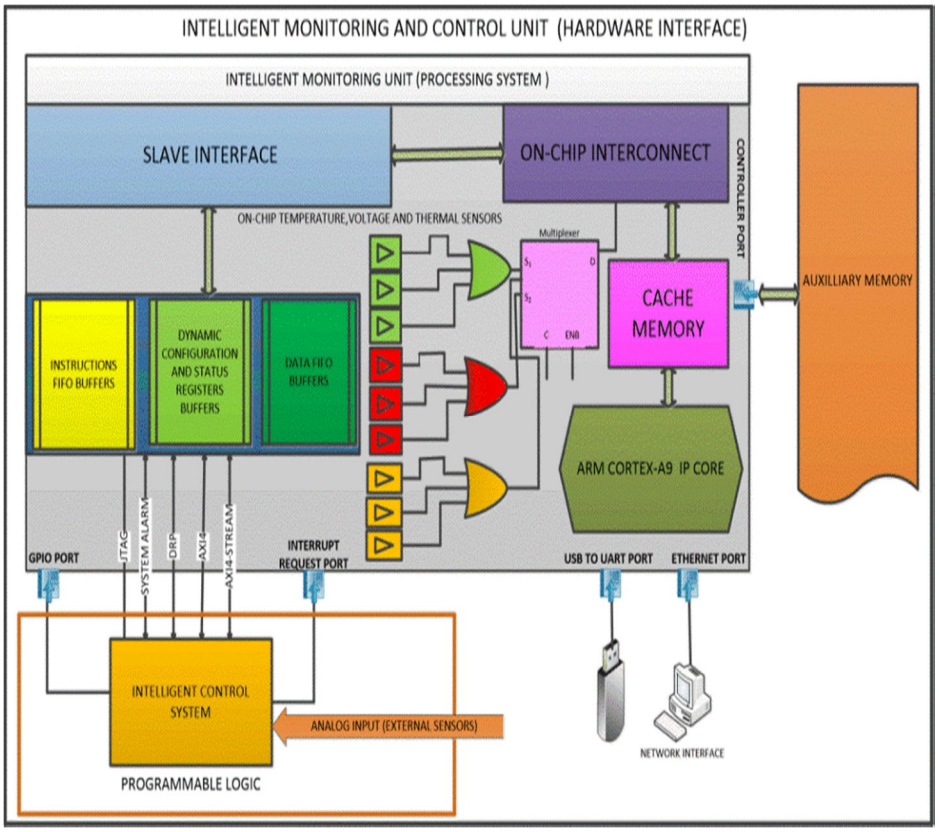
The web server or user application in this case study employs the IIO framework device driver, as mentioned earlier. This is a standard means of providing support for ADCs. It provides two basic functionalities: file system interface for communication with various devices (sensors) connected to the system and the character driver interface to receive event information from the subsystems to the user application space in the monitoring and control system [160-164]. The basic sections of the on-chip sensor network monitoring and control user application are briefly explained as follows:

i) On-chip sensor network monitoring and control: this subsystem provides a means of communication with the IIO system Linux device driver for functionalities like retrieving data from sensors, hardware configuration, synchronous/asynchronous event handling, and control mechanism.

ii) Web server subsystem: this subsystem of the monitoring and control unit addresses the connection request from remote web users. The users can acquire sensor data and other information through the web server interface. Sensory information and event notification and updates are provided to the server through the monitoring and control unit.

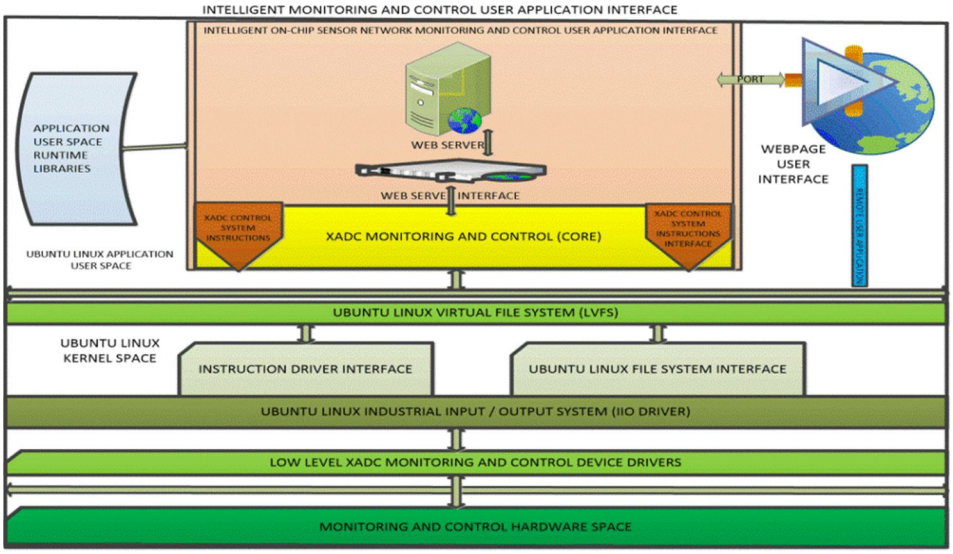
iii) Web interface subsystem: this subsystem provides the basic connection and interface for the web server and the monitoring and control system. The interface provides the functionality to set the threshold values of the different alarms and communicates these values to and from the web server interface for efficient monitoring and control.

iv) Web client subsystem: this subsystem provides functionality to users to tune the different parameter values of the temperature and voltage thresholds. The web client operates in a web browser and provides communication with the web server using a communication port. A graphical user interface is provided to obtain the sensed data from the sensors and the possibility of tuning the threshold values. Figure 50 shows the implemented monitoring and control user application and web interfaces.

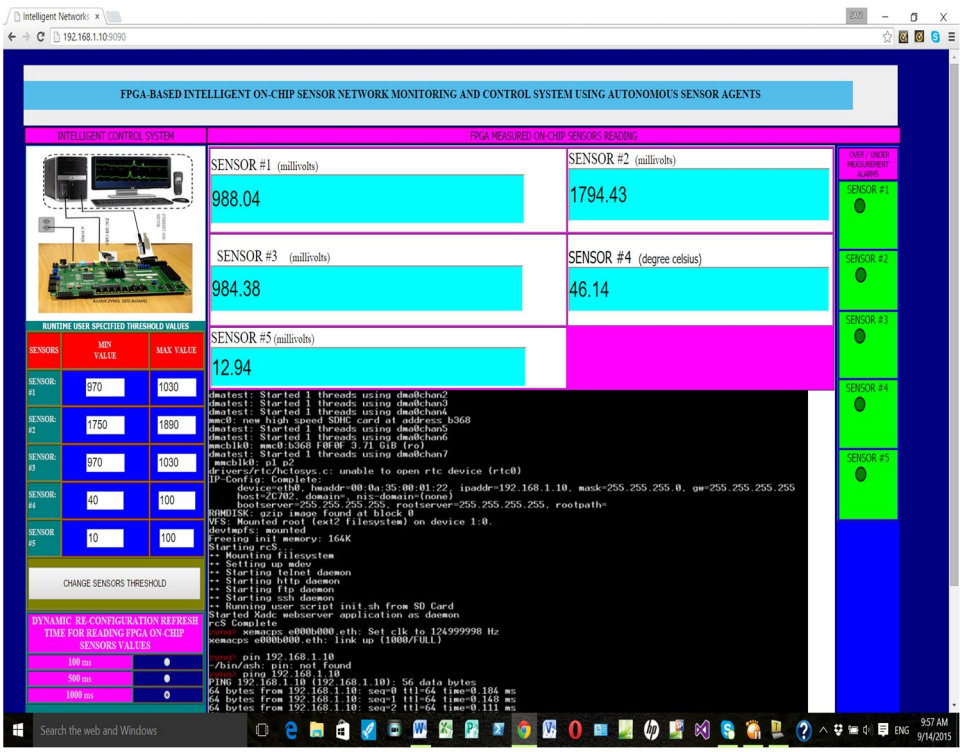


(a)

(Figure 49) Shows on-chip sensor network monitoring and control system (hardware interface)



(a) User-application interface design



(b) Webpage user interface

(Figure 50) Shows on-chip sensor network monitoring and control system.

3. FPGA logic resource utilization

In this subsection, the dissertation provides the FPGA logic resource utilization for post synthesis and implementation (i.e., placement and routing) on the Xilinx Zynq evaluation kit ZC702 XC7Z020 CLG484-1 Zynq-7000 AP SoC FPGA device. Tables 11-14 list the resource usage at different design implementation stages. Tables 11 and 12 list the FPGA implementation results of the logic components for the on-chip sensor network monitoring and control system. We observe that the implemented design uses very small amounts of FPGA logic resources, i.e., 2823 out of 482901, which is equivalent to 0.5851%, compared with the results presented by the authors in [56,57], [60,61] and [110,122]. This result indicates low-cost utilization of the logic resources when implemented in a medium-sized seven-series FPGA device. This result validates our stated objective of achieving a low-cost intelligent monitoring and control system. After the design synthesis, we optimize the design using the Xilinx Vivado tool to efficiently route the design into the target architecture. In Table 12, we can observe that because of the optimization done to route the design, a slight reduction in the amount of logic consumption occurs. Table 13 lists the low-level primitives used by the implemented design and their utilization numbers. Table 14 lists the power distribution and utilization of the various on-chip sensor components after placement and routing. This distribution consisted of both dynamic and static power consumptions. We can see from the table that all on-

chip sensor components show very low power consumption. We achieved 0.118 W of static power and 0.01W of dynamic power consumption. A total of 0.129 W, which represents a 2.26% of both dynamic and static power consumption, for all on-chip sensor components as compared with the results presented in [57,60]. This result also agrees with our stated objective of achieving low power consumption.

(Table 10) FPGA post synthesis (device placement) resource utilization implemented on Xilinx Zynq evaluation kit ZC702, XC7Z020 CLG484-1 FPGA device.

Resource type	Used	Available	Percentage utilization
Slice LUTs	641	53200	1.20%
LUTs as logic	571	53200	1.07%
LUTs as memory	70	17400	0.40%
LUT as shift register	70	-	-
Slice registers	770	106400	0.72%
Registers as flip-flops	770	106400	0.72%
Registers as latch	0	106400	0.0%
F7 Muxes	8	26600	0.03%
F8 Muxes	0	13300	0.00%
XADC	1	1	100%
TOTAL	2901	482901	0.601%

(Table 11) FPGA post implementation (device routing) resource utilization implemented on Xilinx Zynq evaluation kit ZC702, XC7Z020 CLG484-1. FPGA device.

Resource type	Used	Available	Percentage utilization (%)
Slice LUTs	607	53200	1.14
LUTs as logic	545	53200	1.02
LUTs as memory	62	17400	0.35
LUT as shift register	62	-	-
Slice registers	769	106400	0.72
Registers as flip-flops	769	106400	0.72
Registers as latch	0	106400	0.00
F7 Muxes	8	26600	0.03
F8 Muxes	0	13300	0.00
XADC	1	1	100%
TOTAL	2823	482901	0.585%

(Table 12) FPGA low-level primitive resource utilization implemented on Xilinx Zynq Evaluation kit ZC702, XC7Z020 CLG484-1 FPGA device.

Name	FDRE	UT3	LUT6	LUT5	BIBUF	FDSE	LUT4	LUT2	SRLC32E	SRL16E	CARRY4	MUXF7	LUT1	XADC	PS7	BUFG
Used	677	216	206	131	130	92	77	65	47	23	18	8	8	1	1	1

(Table 13) FPGA on-chip sensor component post placement and routing power resource utilization implemented on Xilinx Zync evaluation kit ZC702, XC7Z020 CLG484-1 FPGA device.

Resource type	Used	Available	Power (W)	Percentage utilization (%)
Clocks	1	-	0.001	-
Slice logic	1757	-	0.001	-
LUTs as logic	545	53200	0.001	1.02
Register	769	106400	0.001	0.72
CARRY4	18	13300	0.001	0.14
F7/F8 Muxes	8	53200	0.001	0.02
LUT as Shift Register	62	17400	0.001	0.36
Others	189	-	0.001	-
Signals	1225	-	0.001	-
XADC	1	1	0.001	-
PS7	1	1	0.001	-
Static Power	-	-	0.118	-
	TOTAL		0.129 watt	2.26%

4. FPGA logic resource utilization comparison with previous works

Table 5 shows the FPGA utilization results comparison with previous works [56,57], [60,61] and [110,122]. The comparison with the state-of-the-art seems to be difficult due to the fact that there are a lot of differences in terms of the FPGA implementation architectures, sensors and sensor networks and some of the proposed approaches did not provide FPGA power utilization result. Furthermore, the type of logic resources provided by the previous approaches are different from one implementation to another. However, the utilization results shown in Table 14 helps to acknowledge the efficiency and flexibility of the proposed approach compared with the state-of-the-art.

(Table 14) FPGA logic resource utilization comparison with previous works.

FPGA RESOURCE UTILIZATION COMPARISON TABLE					
Chengqun et al. [56]					
Logic Control Method					
Flip Flops (%)	4-Input LUTs (%)	Slices (%)	Block RAM (%)	Power Utilization	Implementation Architecture
19%	35%	41%	25%	Not provided	XILINX SPARTAN-3 FPGA DEVICE (3XC3S400).
Lookup Table Method					
12%	29%	31%	37%	Not provided	XILINX SPARTAN-3 FPGA DEVICE (3XC3S400).
Perera et al. [57]					
Slice Registers (%)	4-Input LUTs (%)	Occupied Slices (%)	Bonded IOBs (%)	Static power = 460.9 mW Dynamic power = 491.4 mW	XILINX SPARTAN-3E FPGA DEVICE (XC3S250E).
50%	52%	97%	8%		
Echanobe et al. [60]					
FFs (%)	LUTs (%)	BRAM (%)	DSPs (%)	Static power = 3.071 mW Dynamic power = 0.649 mW	XILINX VIRTEX-5 FPGA DEVICE (XCVSX50T).
41%	52%	50%	83%		
Carlos et al. [61]					
LUT + LATCH Pairs (#)	Loop Count (#)	Actual Delay (ns)	LUT + D-FF Pairs (#)	Not provided	XILINX SPARTAN-3E FPGA DEVICE (XC3S100E).
80	1024	180,000	110		
Kornaros et al. [110]					
Configuration	Slices (#)	RAMB16s (#)	Frequency (MHz)	Not provided	XILINX VIRTEX-4 FPGA DEVICE.
CAM Plugin (16 x 192) Control Monitor	1364	18	211		
Manager (Depth 32, Trace, 16 bits	3872	1	216		
Monitor (32 bit) with FSL Interface	432	3	418		
Eduardo et al. [111]					
Implemented prototype uses 2065 LEs a total of 7 % logic utilization.				Not provided	CYCLON II ALTERA FPGA DEVICE (EP2C35).
Gomez-Ouna et al. [112]					
Delay stages (#)	Loop counter (#)	Measured Delay (ns)	Total Area Slices (#)	Delay Area (%)	Not provided	XILINX SPARTAN-3E FPGA DEVICE (XC3S100E).
10	2,560	124.170	34	29%		
20	1,280	117,500	43	47%		
30	853	116,340	50	60%		
4	4,096	92,370	14	29%	Not provided	XILINX VIRTEX-5 FPGA DEVICE (LX50T).
Proposed Approach							
Slice LUTs (%)	LUT as Logic (%)	LUT as Memory (%)	Slice Registers (%)	Registers as Flip Flops (%)	F7 Muxes (%)	Static power = 0.118 W (118 mW). Dynamic power = 0.011 W (11 mW)	XILINX ZYNQ FPGA-7 DEVICE ZC702 (XC7Z020 CLG484-1).
1.14%	1.02%	0.35%	0.72%	0.72%	0.03%		
Total logic utilization (implemented design) = 0.585%							

%: Percentage of logic resource utilization.
#: Number of logic resources.
Not provided: The authors did not provide the power utilization result.

E. FPGA On-Chip Sensor Network Experimental Results and Discussion

In this subsection, the dissertation provides the results of the experiments to monitor and control the FPGA on-chip sensor readings. The essential metric for evaluating the FPGA on-chip sensor readings is its accuracy, i.e., how accurate does the FPGA report the sensed on-chip voltage and temperature readings. Although the FPGA on-chip sensors are specified in terms of accuracy based on the manufacturer datasheet, the real significance is the on-chip sensor error. For instance, the temperature sensor specification for accuracy is ± 4% full scale (FS) within the range −40°C to +100°C, which means that the on-chip sensor error will not exceed ±4% of its FS within its calibrated range. Therefore, the evaluation of the FPGA on-chip sensor accuracy represents the process of determining its maximum error [160,161,167].

We conducted several experiments using five different sensors, namely, Sensors #1-#5. Sensor #4 is a temperature sensor that measured the FPGA on-chip die temperature in degree Celsius and operates within a set threshold limit. The remaining four sensors are power sensors, which measure the on-chip voltage variations in millivolts within a monitored and control threshold value. Tables 15-29 list the FPGA-measured on-chip sensor readings. Each table represents the four experiments conducted, taking 10 readings in each experiment for a total of 40 measured readings from the FPGA on-chip sensors. The readings

were averaged to remove the gross and systematic errors from the observed readings. The FS percentage error or accuracy was calculated for each observation and averaged across the 40 readings. We varied the dynamic reconfiguration refresh time to query the FPGA on-chip sensor readings from the webpage at 100, 500, and 1000 ms to determine the effect of the reconfiguration time with respect to the sensor accuracy. The details of the experimental results are presented in the next subsection.

F. Effect of Dynamic Reconfiguration Refresh Time on an FPGA-measured On-chip Sensor Reading Accuracy

Tables 15-19 list the FPGA-measured on-chip sensor readings with the dynamic reconfiguration refresh time set at 1000 ms. The range of operation and the ideal output of each sensor are specified and listed in the tables. The averaged FS percentage errors of the five sensors are as follows: Sensor #1 = - 0.7%FS, Sensor #2 = -0.1%FS, Sensor #3 = 0.8%FS, Sensor #4 = 0.1%FS, and Sensor #5 = 0.4%FS. This result indicates that the accuracy of the measured sensor values is influenced by the reconfiguration refresh time, which shows an accuracy range between - 0.7%FS and +0.8%FS.

(Table 15) FPGA-measured on-chip Sensor #1 reading with dynamic reconfiguration refresh time of 1000 ms

S/N	EXPT. 1	EXPT. 2	EXPT. 3	EXPT. 4	AVERAGE	RANGE		IDEAL	ERROR	FULL SCALE
	SENSOR #1 MEASURED READING (mV)	SENSOR #1 MEASURED READING (mV)	SENSOR #1 MEASURED READING (mV)	SENSOR #1 MEASURED READING (mV)	MEASURED READING (mV)	HIGH (mV)	LOW (mV)	OUTPUT (mV)	VALUES	%ERROR
1.	988.77	984.38	990.23	986.57	987.49	1030.00	970.00	987	0.49	0.8
2.	982.91	985.84	988.77	986.57	986.02	1030.00	970.00	987	-0.98	-1.6
3.	987.30	988.04	988.04	986.30	987.42	1030.00	970.00	987	0.42	0.7
4.	985.84	985.11	986.57	987.30	986.21	1030.00	970.00	987	-0.80	-1.3
5.	988.04	988.77	986.57	988.04	987.86	1030.00	970.00	987	0.86	1.4
6.	987.30	985.11	986.57	990.23	987.30	1030.00	970.00	987	0.30	0.5
7.	985.84	985.84	987.30	988.77	986.94	1030.00	970.00	987	-0.06	-0.1
8.	985.84	983.64	985.84	986.57	985.47	1030.00	970.00	987	-1.53	-2.5
9.	985.84	984.38	984.38	986.64	985.31	1030.00	970.00	987	-1.69	-2.8
10.	985.11	985.84	985.84	985.84	985.66	1030.00	970.00	987	-1.34	-2.2
								AVERAGE	-0.43	-0.7

(Table 16) FPGA-measured on-chip Sensor #2 reading with dynamic reconfiguration refresh time of 1000 ms

S/N	EXPT. 1	EXPT. 2	EXPT. 3	EXPT. 4	AVERAGE	RANGE		IDEAL	ERROR	FULL SCALE
	SENSOR #2 MEASURED READING (mV)	SENSOR #2 MEASURED READING (mV)	SENSOR #2 MEASURED READING (mV)	SENSOR #2 MEASURED READING (mV)	MEASURED READING (mV)	HIGH (mV)	LOW (mV)	OUTPUT (mV)	VALUES	%ERROR
1.	1788.57	1791.50	1795.17	1792.24	1791.87	1890.00	1750.00	1793	-1.13	-0.8
2.	1789.31	1794.43	1794.43	1792.24	1792.60	1890.00	1750.00	1793	-0.40	-0.3
3.	1795.17	1791.50	1796.63	1792.24	1793.89	1890.00	1750.00	1793	0.88	0.6
4.	1792.24	1792.24	1797.36	1788.57	1792.60	1890.00	1750.00	1793	-0.40	-0.3
5.	1788.57	1792.24	1795.90	1791.50	1792.05	1890.00	1750.00	1793	-0.95	-0.7
6.	1792.24	1795.90	1792.97	1792.97	1793.52	1890.00	1750.00	1793	0.52	0.4
7.	1790.04	1792.97	1801.03	1792.97	1794.25	1890.00	1750.00	1793	1.25	0.9
8.	1785.64	1795.17	1793.70	1793.70	1792.05	1890.00	1750.00	1793	-0.95	-0.7
9.	1793.70	1792.24	1791.50	1790.04	1791.87	1890.00	1750.00	1793	-1.13	-0.8
10.	1794.43	1792.97	1795.90	1793.70	1794.25	1890.00	1750.00	1793	1.25	0.9
								AVERAGE	-0.10	-0.1

(Table 17) FPGA-measured on-chip Sensor #3 reading with dynamic reconfiguration refresh time of 1000 ms

S/N	EXPT. 1	EXPT. 2	EXPT. 3	EXPT. 4	AVERAGE	RANGE		IDEAL	ERROR	FULL SCALE
	SENSOR #3 MEASURED READING (mV)	SENSOR #3 MEASURED READING (mV)	SENSOR #3 MEASURED READING (mV)	SENSOR #3 MEASURED READING (mV)	MEASURED READING (mV)	HIGH (mV)	LOW (mV)	OUTPUT (mV)	VALUES	%ERROR
1.	986.57	986.57	987.30	987.30	986.94	1030.00	970.00	986	0.93	1.6
2.	985.11	987.30	985.11	985.84	985.84	1030.00	970.00	986	-0.16	-0.3
3.	983.64	987.30	986.57	989.50	986.75	1030.00	970.00	986	0.75	1.3
4.	986.57	987.30	988.04	985.84	986.94	1030.00	970.00	986	0.94	1.6
5.	985.11	984.38	988.04	986.57	986.03	1030.00	970.00	986	0.02	0.04
6.	985.84	985.84	987.30	988.77	986.94	1030.00	970.00	986	0.94	1.6
7.	989.50	986.57	986.57	986.57	987.30	1030.00	970.00	986	1.30	2.2
8.	983.64	988.77	985.84	986.57	986.21	1030.00	970.00	986	0.21	0.3
9.	984.38	987.30	985.11	987.30	986.02	1030.00	970.00	986	0.02	0.04
10.	985.84	984.38	988.04	985.11	985.84	1030.00	970.00	986	-0.16	-0.3
								AVERAGE	0.48	0.8

(Table 18) FPGA-measured on-chip Sensor #4 reading with dynamic reconfiguration refresh time of 1000 ms

S/N	EXPT. 1	EXPT. 2	EXPT. 3	EXPT. 4	AVERAGE	RANGE		IDEAL	ERROR	FULL SCALE
	SENSOR #4 MEASURED READING (℃)	SENSOR #4 MEASURED READING (℃)	SENSOR #4 MEASURED READING (℃)	SENSOR #4 MEASURED READING (℃)	MEASURED READING (℃)	HIGH (℃)	LOW (℃)	OUTPUT (℃)	VALUES	%ERROR
1.	55.12	47.49	47.62	47.25	49.37	100.00	40.00	50.00	-0.63	-1.1
2.	54.26	48.36	47.49	48.23	49.59	100.00	40.00	50.00	-0.41	-0.7
3.	55.49	48.23	47.74	47.74	49.80	100.00	40.00	50.00	-0.20	-0.3
4.	55.12	48.85	47.74	49.09	50.20	100.00	40.00	50.00	0.20	0.3
5.	55.00	49.09	48.23	48.23	50.14	100.00	40.00	50.00	0.14	0.2
6.	55.37	49.22	47.74	48.48	50.20	100.00	40.00	50.00	0.20	0.3
7.	55.12	49.09	47.99	48.48	50.17	100.00	40.00	50.00	0.17	0.3
8.	54.88	50.20	47.86	48.48	50.36	100.00	40.00	50.00	0.35	0.6
9.	55.49	50.08	47.99	48.60	50.54	100.00	40.00	50.00	0.54	0.9
10.	55.25	49.83	48.23	48.72	50.51	100.00	40.00	50.00	0.51	0.8
								AVERAGE	0.09	0.1

(Table 19) FPGA-measured on-chip Sensor #5 reading with dynamic reconfiguration refresh time of 1000 ms

S/N	EXPT. 1	EXPT. 2	EXPT. 3	EXPT. 4	AVERAGE	RANGE		IDEAL	ERROR	FULL SCALE
	SENSOR #5 MEASURED READING (mV)	SENSOR #5 MEASURED READING (mV)	SENSOR #5 MEASURED READING (mV)	SENSOR #5 MEASURED READING (mV)	MEASURED READING (mV)	HIGH (mV)	LOW (mV)	OUTPUT (mV)	VALUES	%ERROR
1.	13.43	13.18	13.43	13.43	13.37	100.00	10.00	13.00	0.37	0.4
2.	13.43	13.18	13.43	13.43	13.37	100.00	10.00	13.00	0.37	0.4
3.	13.43	12.94	13.18	13.43	13.25	100.00	10.00	13.00	0.24	0.3
4.	13.43	13.43	13.18	13.43	13.37	100.00	10.00	13.00	0.37	0.4
5.	13.67	13.18	12.70	13.43	13.25	100.00	10.00	13.00	0.24	0.3
6.	13.67	13.43	12.94	12.94	13.25	100.00	10.00	13.00	0.24	0.3
7.	13.67	13.43	13.18	13.67	13.49	100.00	10.00	13.00	0.49	0.5
8.	13.43	13.18	13.43	13.18	13.31	100.00	10.00	13.00	0.31	0.3
9.	13.43	13.43	13.18	13.18	13.31	100.00	10.00	13.00	0.31	0.3
10.	13.92	13.43	12.70	13.67	13.43	100.00	10.00	13.00	0.43	0.5
								AVERAGE	0.34	0.4

Tables 20 to 24 list the FPGA-measured on-chip sensor readings with the dynamic reconfiguration refresh time set at 500 ms. Here, the averaged FS percentage errors for the five sensors are as follows: Sensor #1 = -0.6%FS, Sensor #2 = -0.6%FS, Sensor #3 = +0.2%FS, Sensor #4 = +1.1%FS, and Sensor #5 = 0.3%FS. The averaged FS accuracy lies between -0.6%FS and +1.1%FS, which illustrates a better accuracy compared with reconfiguration refresh time of 100 ms.

(Table 20) FPGA-measured on-chip Sensor #1 reading with dynamic reconfiguration refresh time of 500 ms

S/N	EXPT. 1	EXPT. 2	EXPT. 3	EXPT. 4	AVERAGE	RANGE		IDEAL	ERROR	FULL SCALE
	SENSOR #1 MEASURED READING (mV)	SENSOR #1 MEASURED READING (mV)	SENSOR #1 MEASURED READING (mV)	SENSOR #1 MEASURED READING (mV)	MEASURED READING (mV)	HIGH (mV)	LOW (mV)	OUTPUT (mV)	VALUES	%ERROR
1.	986.57	983.64	988.77	987.30	986.57	1030.00	970.00	987	-0.43	-0.7
2.	985.11	985.84	986.57	987.30	986.21	1030.00	970.00	987	-0.80	-1.3
3.	985.84	987.30	988.77	986.57	987.12	1030.00	970.00	987	0.12	0.2
4.	985.11	985.84	985.11	988.77	986.21	1030.00	970.00	987	-0.79	-1.3
5.	985.84	986.57	985.84	986.57	986.21	1030.00	970.00	987	-0.79	-1.3
6.	986.57	986.57	988.77	987.30	987.30	1030.00	970.00	987	0.30	0.5
7.	988.77	988.04	988.04	985.11	987.49	1030.00	970.00	987	0.49	0.8
8.	987.30	985.11	985.11	988.04	986.39	1030.00	970.00	987	-0.61	-1.0
9.	985.84	985.11	988.04	987.30	986.57	1030.00	970.00	987	-0.43	-0.7
10.	988.04	985.84	984.33	988.04	986.56	1030.00	970.00	987	-0.44	-0.7
								AVERAGE	-0.34	-0.6

(Table 21) FPGA-measured on-chip Sensor #2 reading with dynamic reconfiguration refresh time of 500 ms

S/N	EXPT. 1	EXPT. 2	EXPT. 3	EXPT. 4	AVERAGE	RANGE		IDEAL	ERROR	FULL SCALE
	SENSOR #2 MEASURED READING (mV)	SENSOR #2 MEASURED READING (mV)	SENSOR #2 MEASURED READING (mV)	SENSOR #2 MEASURED READING (mV)	MEASURED READING (mV)	HIGH (mV)	LOW (mV)	OUTPUT (mV)	VALUES	%ERROR
1.	1790.04	1792.97	1797.36	1792.97	1793.34	1890.00	1750.00	1793	0.34	0.2
2.	1792.97	1792.97	1793.7	1794.43	1793.52	1890.00	1750.00	1793	0.52	0.4
3.	1793.70	1792.24	1790.77	1792.97	1792.42	1890.00	1750.00	1793	-0.58	-0.4
4.	1793.70	1795.90	1794.43	1795.90	1794.98	1890.00	1750.00	1793	1.98	1.4
5.	1793.70	1796.63	1792.97	1795.17	1794.62	1890.00	1750.00	1793	1.62	1.2
6.	1795.90	1792.24	1791.5	1795.90	1793.89	1890.00	1750.00	1793	0.89	0.6
7.	1794.43	1792.97	1795.17	1790.77	1793.34	1890.00	1750.00	1793	0.34	0.2
8.	1791.50	1792.97	1793.7	1792.97	1792.79	1890.00	1750.00	1793	-0.21	-0.2
9.	1791.50	1792.24	1792.97	1795.17	1792.97	1890.00	1750.00	1793	-0.03	0.0
10.	1792.97	1790.77	1792.24	1791.50	1791.87	1890.00	1750.00	1793	-1.13	-0.8
								AVERAGE	-0.34	-0.6

(Table 22) FPGA-measured on-chip Sensor #3 reading with dynamic reconfiguration refresh time of 500 ms

S/N	EXPT. 1	EXPT. 2	EXPT. 3	EXPT. 4	AVERAGE	RANGE		IDEAL	ERROR	FULL SCALE
	SENSOR #3 MEASURED READING (mV)	SENSOR #3 MEASURED READING (mV)	SENSOR #3 MEASURED READING (mV)	SENSOR #3 MEASURED READING (mV)	MEASURED READING (mV)	HIGH (mV)	LOW (mV)	OUTPUT (mV)	VALUES	%ERROR
1.	986.57	987.30	986.57	983.64	986.02	1030.00	970.00	986	0.02	0.03
2.	985.11	985.11	986.57	986.57	985.84	1030.00	970.00	986	-0.16	-0.3
3.	984.38	987.30	983.64	985.11	985.11	1030.00	970.00	986	-0.89	-1.5
4.	987.30	987.30	986.57	986.57	986.94	1030.00	970.00	986	0.94	1.6
5.	985.11	986.57	985.84	987.30	986.21	1030.00	970.00	986	0.20	0.3
6.	984.38	982.18	987.3	985.84	984.93	1030.00	970.00	986	-1.08	-1.8
7.	986.57	984.38	988.77	985.11	986.21	1030.00	970.00	986	0.21	0.3
8.	985.84	987.30	985.84	988.77	986.94	1030.00	970.00	986	0.94	1.6
9.	989.50	988.04	986.57	987.30	987.85	1030.00	970.00	986	1.85	3.1
10.	983.64	986.57	985.11	984.38	984.93	1030.00	970.00	986	-1.07	-1.8
								AVERAGE	0.10	0.2

(Table 23) FPGA-measured on-chip Sensor #4 reading with dynamic reconfiguration refresh time of 500 ms

S/N	EXPT. 1	EXPT. 2	EXPT. 3	EXPT. 4	AVERAGE	RANGE		IDEAL	ERROR	FULL SCALE
	SENSOR #4 MEASURED READING (℃)	SENSOR #4 MEASURED READING (℃)	SENSOR #4 MEASURED READING (℃)	SENSOR #4 MEASURED READING (℃)	MEASURED READING (℃)	HIGH (℃)	LOW (℃)	OUTPUT (℃)	VALUES	%ERROR
1.	54.75	50.20	48.23	48.72	50.48	100.00	40.00	50.00	0.48	0.8
2.	55.37	50.32	48.48	48.36	50.63	100.00	40.00	50.00	0.63	1.1
3.	55.49	49.83	48.36	48.97	50.66	100.00	40.00	50.00	0.66	1.1
4.	55.98	50.32	48.11	48.67	50.77	100.00	40.00	50.00	0.77	1.3
5.	55.86	49.71	48.36	48.48	50.60	100.00	40.00	50.00	0.60	1.0
6.	55.61	50.57	47.86	48.72	50.69	100.00	40.00	50.00	0.69	1.2
7.	55.61	50.20	48.29	48.60	50.68	100.00	40.00	50.00	0.67	1.1
8.	55.37	50.20	47.86	48.60	50.51	100.00	40.00	50.00	0.51	0.8
9.	55.61	50.08	47.62	48.85	50.54	100.00	40.00	50.00	0.54	0.9
10.	56.48	50.57	47.99	49.22	51.07	100.00	40.00	50.00	1.07	1.8
								AVERAGE	0.66	1.1

(Table 24) FPGA-measured on-chip Sensor #5 reading with dynamic reconfiguration refresh time of 500 ms

S/N	EXPT. 1	EXPT. 2	EXPT. 3	EXPT. 4	AVERAGE	RANGE		IDEAL	ERROR	FULL SCALE
	SENSOR #5 MEASURED READING (mV)	SENSOR #5 MEASURED READING (mV)	SENSOR #5 MEASURED READING (mV)	SENSOR #5 MEASURED READING (mV)	MEASURED READING (mV)	HIGH (mV)	LOW (mV)	OUTPUT (mV)	VALUES	%ERROR
1.	13.43	13.43	13.18	13.18	13.31	100.00	10.00	13.00	0.31	0.3
2.	12.94	12.70	13.43	13.43	13.13	100.00	10.00	13.00	0.13	0.1
3.	13.43	12.94	12.70	12.70	12.94	100.00	10.00	13.00	-0.06	-0.1
4.	13.67	13.43	13.18	13.18	13.37	100.00	10.00	13.00	0.37	0.4
5.	13.67	13.43	13.18	12.94	13.31	100.00	10.00	13.00	0.31	0.3
6.	13.67	13.43	12.94	13.43	13.37	100.00	10.00	13.00	0.37	0.4
7.	13.67	12.94	12.94	13.43	13.25	100.00	10.00	13.00	0.24	0.3
8.	13.92	13.18	13.43	12.94	13.37	100.00	10.00	13.00	0.37	0.4
9.	13.67	13.18	12.94	12.94	13.18	100.00	10.00	13.00	0.18	0.2
10.	13.43	13.18	13.18	13.18	13.24	100.00	10.00	13.00	0.24	0.3
								AVERAGE	0.24	0.3

Tables 25-29 show the FPGA-measured and controlled on-chip sensors readings with the dynamic reconfiguration refresh time set at 100 ms. The averaged percentage accuracy of Sensor #1 FS is -0.6%FS, Sensor #2 = -0.3%FS, Sensor #3 = -0.024%FS, Sensor #4 = 1.3%FS, and Sensor #5 = 0.4%FS. From Tables 25-29, we can observe that the FS percentage accuracy lies from −0.6%FS to + 1.3%FS. This result shows a low accuracy at FS as compared with the dynamic reconfiguration refresh times of 500 and 1000 ms. Table 30 shows the comparison results of the sensor reading accuracy versus the dynamic reconfiguration refresh time. We can observe that the FS percentage error with dynamic refresh time of 100 ms is between -0.6%FS and + 1.3% FS, that of 500 ms is between -0.6%FS and +1.1%FS, and that of 1000 ms is between - 0.7%FS

and + 0.8FS. This result shows that the lower FS percentage error range is almost the same (i.e., -0.6%FS and -0.7%FS) for all sensors irrespective of the dynamic refresh time. However, at higher FS percentage error rate, a significant difference in accuracy exists with different dynamic refresh time, i.e., +0.8%FS, +1.1%FS and +1.3%FS. Therefore, the dynamic reconfiguration refresh time of 1000 ms produces highly accurate FPGA-measured on-chip sensor readings.

In conclusion, the dynamic reconfiguration refresh time affects the accuracy of the FPGA-measured on-chip sensor readings. We determined that the dynamic reconfiguration refresh time of 1000 ms provides highly accurate FPGA-measured and controlled on-chip sensor readings across the five different on-chip sensors.

(Table 25) FPGA-measured on-chip Sensor #1 reading with dynamic reconfiguration refresh time of 100 ms

S/N	EXPT. 1	EXPT. 2	EXPT. 3	EXPT. 4	AVERAGE	RANGE		IDEAL	ERROR	FULL SCALE
	SENSOR #1 MEASURED READING (mV)	SENSOR #1 MEASURED READING (mV)	SENSOR #1 MEASURED READING (mV)	SENSOR #1 MEASURED READING (mV)	MEASURED READING (mV)	HIGH (mV)	LOW (mV)	OUTPUT (mV)	VALUES	%ERROR
1.	985.11	985.11	990.23	986.57	986.76	1030.00	970.00	987	-0.25	-0.4
2.	987.30	987.30	986.57	986.57	986.94	1030.00	970.00	987	-0.06	-0.1
3.	986.57	986.57	985.11	988.77	986.76	1030.00	970.00	987	-0.25	-0.4
4.	987.30	988.04	988.04	985.84	987.31	1030.00	970.00	987	0.31	0.5
5.	986.57	986.57	986.57	985.11	986.21	1030.00	970.00	987	-0.79	-1.3
6.	987.30	986.57	987.30	985.11	986.57	1030.00	970.00	987	-0.43	-0.7
7.	986.57	987.30	986.57	986.57	986.75	1030.00	970.00	987	-0.25	-0.4
8.	982.91	987.30	986.57	988.04	986.21	1030.00	970.00	987	-0.79	-1.3
9.	986.57	987.30	989.50	985.11	987.12	1030.00	970.00	987	0.12	0.2
10.	984.38	985.84	987.30	985.11	985.66	1030.00	970.00	987	-1.34	-2.2
								AVERAGE	-0.37	-0.6

(Table 26) FPGA-measured on-chip Sensor #2 reading with dynamic reconfiguration refresh time of 100 ms

S/N	EXPT. 1	EXPT. 2	EXPT. 3	EXPT. 4	AVERAGE	RANGE		IDEAL	ERROR	FS
	SENSOR #2 MEASURED READING (mV)	SENSOR #2 MEASURED READING (mV)	SENSOR #2 MEASURED READING (mV)	SENSOR #2 MEASURED READING (mV)	MEASURED READING (mV)	HIGH (mV)	LOW (mV)	OUTPUT (mV)	VALUES	%ERROR
1.	1793.70	1791.50	1789.31	1794.43	1792.24	1890.00	1750.00	1794	-1.76	-1.3
2.	1791.50	1796.63	1794.43	1792.97	1793.88	1890.00	1750.00	1794	-0.12	-0.1
3.	1791.50	1791.50	1792.24	1792.97	1792.05	1890.00	1750.00	1794	-1.95	-1.4
4.	1792.24	1795.17	1795.17	1795.90	1794.62	1890.00	1750.00	1794	0.62	0.4
5.	1793.70	1793.70	1794.43	1795.17	1794.25	1890.00	1750.00	1794	0.25	0.2
6.	1789.31	1795.90	1794.43	1795.17	1793.70	1890.00	1750.00	1794	-0.30	-0.2
7.	1795.17	1790.04	1792.97	1792.97	1792.79	1890.00	1750.00	1794	-1.21	-0.9
8.	1793.70	1793.70	1797.36	1792.97	1794.43	1890.00	1750.00	1794	0.43	0.3
9.	1794.43	1792.97	1791.50	1794.43	1793.33	1890.00	1750.00	1794	-0.67	-0.5
10.	1796.63	1795.90	1794.43	1792.24	1794.80	1890.00	1750.00	1794	0.80	0.6
								AVERAGE	-0.39	-0.3

(Table 27) FPGA-measured on-chip Sensor #3 reading with dynamic reconfiguration refresh time of 100 ms

S/N	EXPT. 1	EXPT. 2	EXPT. 3	EXPT. 4	AVERAGE	RANGE		IDEAL	ERROR	FS
	SENSOR #3 MEASURED READING (mV)	SENSOR #3 MEASURED READING (mV)	SENSOR #3 MEASURED READING (mV)	SENSOR #3 MEASURED READING (mV)	MEASURED READING (mV)	HIGH (mV)	LOW (mV)	OUTPUT (mV)	VALUES	%ERROR
1.	985.84	985.84	985.84	984.38	985.48	1030.00	970.00	986	-0.52	-0.9
2.	985.11	986.57	985.84	987.30	986.21	1030.00	970.00	986	0.20	0.3
3.	984.38	986.57	986.57	987.30	986.21	1030.00	970.00	986	0.20	0.3
4.	986.57	984.38	986.57	986.57	986.02	1030.00	970.00	986	0.02	0.04
5.	988.04	987.30	987.30	986.57	987.30	1030.00	970.00	986	1.30	2.2
6.	985.84	983.64	982.18	986.57	984.56	1030.00	970.00	986	-1.44	-2.4
7.	985.84	988.04	985.11	986.57	986.39	1030.00	970.00	986	0.39	0.7
8.	986.57	986.57	987.30	987.30	986.94	1030.00	970.00	986	0.93	1.6
9.	985.84	983.64	985.84	986.57	985.47	1030.00	970.00	986	-0.53	-0.9
10.	985.84	986.57	984.38	984.38	985.29	1030.00	970.00	986	-0.71	-1.2
								AVERAGE	-0.01	-0.024

(Table 28) FPGA-measured on-chip Sensor #4 reading with dynamic reconfiguration refresh time of 100 ms

S/N	EXPT. 1	EXPT. 2	EXPT. 3	EXPT. 4	AVERAGE	RANGE		IDEAL	ERROR	FULL SCALE
	SENSOR #4 MEASURED READING (°C)	SENSOR #4 MEASURED READING (°C)	SENSOR #4 MEASURED READING (°C)	SENSOR #4 MEASURED READING (°C)	MEASURED READING (°C)	HIGH (°C)	LOW (°C)	OUTPUT (°C)	VALUES	%ERROR
1.	55.49	50.45	47.99	48.85	50.70	100.00	40.00	50.00	0.70	1.2
2.	55.98	50.20	48.23	48.72	50.78	100.00	40.00	50.00	0.78	1.3
3.	56.23	50.08	47.49	49.46	50.82	100.00	40.00	50.00	0.82	1.4
4.	56.61	49.96	48.60	48.48	50.91	100.00	40.00	50.00	0.91	1.5
5.	55.49	50.45	48.23	48.48	50.66	100.00	40.00	50.00	0.66	1.1
6.	55.74	50.57	48.36	48.60	50.82	100.00	40.00	50.00	0.82	1.4
7.	55.74	50.57	47.37	48.97	50.66	100.00	40.00	50.00	0.66	1.1
8.	55.74	50.69	47.62	48.72	50.69	100.00	40.00	50.00	0.69	1.2
9.	55.37	50.20	48.23	48.48	50.57	100.00	40.00	50.00	0.57	0.9
10.	55.74	50.32	48.23	49.34	50.91	100.00	40.00	50.00	0.91	1.5
								AVERAGE	0.75	1.3

(Table 29) FPGA-measured on-chip Sensor #5 reading with dynamic reconfiguration refresh time of 100 ms

S/N	EXPT. 1	EXPT. 2	EXPT. 3	EXPT. 4	AVERAGE	RANGE		IDEAL	ERROR	FULL SCALE
	SENSOR #5 MEASURED READING (mV)	SENSOR #5 MEASURED READING (mV)	SENSOR #5 MEASURED READING (mV)	SENSOR #5 MEASURED READING (mV)	MEASURED READING (mV)	HIGH (mV)	LOW (mV)	OUTPUT (mV)	VALUES	%ERROR
1.	12.94	13.18	12.94	13.18	13.06	100.00	10.00	13.00	0.06	0.1
2.	13.92	13.18	13.43	13.43	13.49	100.00	10.00	13.00	0.49	0.5
3.	13.67	13.18	13.18	12.94	13.24	100.00	10.00	13.00	0.24	0.3
4.	13.67	13.43	13.43	13.18	13.43	100.00	10.00	13.00	0.43	0.5
5.	13.67	12.94	13.18	13.43	13.31	100.00	10.00	13.00	0.31	0.3
6.	13.18	13.67	13.18	13.18	13.30	100.00	10.00	13.00	0.30	0.3
7.	13.67	13.67	12.94	13.18	13.37	100.00	10.00	13.00	0.37	0.4
8.	13.43	13.18	13.43	13.43	13.37	100.00	10.00	13.00	0.37	0.4
9.	13.43	13.43	13.67	13.18	13.43	100.00	10.00	13.00	0.43	0.5
10.	13.43	13.43	12.94	12.94	13.19	100.00	10.00	13.00	0.18	0.2
								AVERAGE	0.32	0.4

(Table 30) Comparison results of the sensor reading accuracy versus dynamic reconfiguration refresh time.

	SENSOR #1 (mV)	SENSOR #2 (mV)	SENSOR #3 (mV)	SENSOR #4 (°C)	SENSOR #5 (mV)
	RANGE 970-1030 (mV)	RANGE 1750- 1890 (mV)	RANGE 970-1030 (mV)	RANGE 40-100 (°C)	RANGE 10-100 (mV)
DYNAMIC RECONFIGURATION REFRESH TIME = 1000 Milliseconds (ms)					
SENSORS'S ACCURACY / PRECISION					
FULL SCALE %ERROR	- 0.7	- 0.1	+ 0.8	+ 0.1	+ 0.4
ERROR VALUES	- 0.43	- 0.10	+ 0.48	+ 0.09	+ 0.34
DYNAMIC RECONFIGURATION REFRESH TIME = 500 Milliseconds (ms)					
SENSORS'S ACCURACY / PRECISION					
FULL SCALE %ERROR	- 0.6	- 0.6	+ 0.2	+ 1.1	+ 0.3
ERROR VALUES	- 0.34	- 0.34	+ 0.10	+ 0.66	+ 0.24
DYNAMIC RECONFIGURATION REFRESH TIME = 100 Milliseconds (ms)					
SENSORS'S ACCURACY / PRECISION					
FULL SCALE %ERROR	- 0.6	- 0.3	- 0.024	+ 1.3	+ 0.4
ERROR VALUES	- 0.37	- 0.39	- 0.01	+ 0.75	+ 0.32

G. Chapter Summary

In summary, the chapter describes the proposed low-level micro-architectural design infrastructure for real-time monitoring and control of on-chip sensor network-based systems using FPGA. The chapter discusses the development, design and implementation issues for efficient and reliable high-speed circuit technique of the on-chip sensor network runtime parameter monitoring and control, which use autonomous sensors that reside at the NI, to be dynamically configured and to communicate the runtime ambient parameters to the network monitor controller hardware. A detailed low-level design methodology and procedure and a user application for efficient and reliable on-chip sensor network monitoring and control using FPGAs was provided. In addition, a webpage interface for the user to dynamically reconfigure the FPGA to conduct on-chip sensor readings and adjust the sensor ranges and reconfiguration refresh time was presented. The chapter showed that the proposed approach uses low FPGA logic resources and low power consumption, validating its suitability in real-system development. Furthermore, it demonstrated that by collecting the dynamic and real-time monitoring parameters in terms of temperature and voltage variations, the system can adapt and improve the utilization of FPGA logic resources and energy consumption. Experimental results from the FPGA-measured on-chip sensor readings showed high precision and accuracy in the measured voltage and temperature. We found that a dynamic refresh time of 1000 ms produces the best FPGA-measured and controlled on-chip monitored sensor readings as compared with the 100 and 500 ms refresh time.

VI. Conclusions and Future Work

This dissertation proposed a new paradigm for enhancing system design, reliability and performance of FPGA-Based networks-on-chip in embedded hardware systems. A new paradigm, namely, autonomously self-aware and adaptively reconfigurable system for FPGA-Based network-on-chip was proposed. Autonomous self-aware and adaptive reconfigurable system for network-on-chip (NoC) is a promising paradigm aiming at providing an infrastructural framework for the exploration, performance evaluation, correctness and reliability issues of network-on-chips fabrics in the Field Programmable Gate Arrays (FPGAs). The dissertation considers a design and implementation of architectures that enable for self-adaptation, dynamic reconfiguration, autonomic formation and self-awareness of network-on-chip architectures. The dissertation also considers the design of fault tolerant routing scheme and algorithms for both on-chip interconnect and sensor networks that provide a low level mechanism for ensuring runtime autonomous self-aware fault-tolerant routing in the presence of faulty network components. Furthermore, the dissertation considers system level adaptation by runtime monitoring of environmental parameters and employing evolutionary optimization methods to intelligently adapt and optimize system performance in terms of throughput, latency, clock frequency, silicon area and energy consumption.

In order to achieve the first objective of the dissertation, a detailed parametric evaluation technique was proposed and used it to compare the performance of Mesh, Torus, and Fat-tree NoC architectures using FPGA. It demonstrated that the VCs, FBD, and FDW parameters all contributed to higher LUTs and influenced the area (PLUT) and clock frequency (CLK_FREQ) of the FPGA network, with FDW and FBD having the greatest impact on FPGA resources. The dissertation also found that these three parameters significantly influenced reassembly buffering and routing and logic requirements at the NoC endpoints. To demonstrate the flexibility and design space coverage of the three target NoC architectures, the dissertation reported their results synthesizing of 392 different FPGA network configurations. It showed that by strategically tuning the router and interconnect parameters, Fat-tree networks offered the best utilization of FPGA resources in terms of silicon area, clock frequency, critical path delay, network cost, overall performance under saturation throughput, and lower latencies under both benign and adversarial traffic patterns. In contrast, the dissertation consistently found that the Mesh and Torus networks consumed too much FPGA resources and produced poor network performance under adversarial traffic patterns. Through these findings, the dissertation also demonstrated that the evaluation technique can substantially improve performance under a large variety of experimental conditions, confirm its suitability for real system development. Thus, the presented methodology forms an important foundation for

engineers and designers to make informed early decisions about which interconnects and router parameters to use in large and complex NoCs for FPGA.

As a proof of concept for the autonomous self-aware and adaptive reconfigurable system paradigm, the dissertation proposed ASAART for wireless sensor networks to address the limitations of the SSR and SHR protocols. We integrated autonomous self-awareness and adaptive paradigm into the SHR protocol to make it more resilient to faults and errors and energy efficiency for efficient and reliable routing of sensor data in wireless sensor networks. This is achieved by combining both continuous and prioritized slotted back-off delay, and multiple randomize function techniques to obtain local and global network state information for faster routing path formation and speedy convergence to the minimum routing path. The dissertation proposed a route repair technique for reliable transmission of sensor data in the presence of permanent and transient node failure rates, and efficient adaptation to simultaneous network topology changes. The dissertation presented an extensive simulation under five different scenarios. The simulation results showed that the ASAART technique performed better in terms of high resiliency to errors and failure and better routing performance and energy consumption compared with the SSR and SHR protocols in the presence of transient and permanent node failure rates and in a highly congested, faulty, and scalable sensor network. The proposed approach is energy efficient because it consumes less energy compared to the SSR and SHR protocols

confirms its suitability in sensor networks and other remote sensing applications.

As a proof of concept for the proposed paradigm, the dissertation further proposed the design of efficient and high-speed circuit for real-time monitoring and control of on-chip sensor network using FPGA. It developed the autonomous sensor agents implemented in the FPGA-based NI to be dynamically configured and to communicate the dynamic ambient parameter changes to the monitoring and control hardware unit. The ultimate goal is to design a low-cost, low-power, and high-accuracy real-time monitoring mechanism using autonomous sensor agents as well as a dynamic reconfiguration control of on-chip sensor network environment using FPGAs. The dissertation presented a detailed design procedure and a case study to demonstrate the applicability of the proposed paradigm. It showed that the proposed approach uses low FPGA logic resources and low power consumption, validating its suitability in real-system development. Furthermore, it's shown that by collecting the dynamic and real-time monitoring parameters in terms of temperature and voltage variations, the system can adapt and improve the utilization of FPGA logic resources and energy consumption. Experimental results from the FPGA-measured on-chip sensor readings showed high precision and accuracy in the measured voltage and temperature. Through the experiment, it was found that a dynamic refresh time of 1000 ms produces the best FPGA-measured and controlled on-chip monitored sensor readings as compared with the 100 and 500 ms refresh time. The proposed design techniques, framework, and protocol will assist network engineers and system designers with a flexible and efficient real-time monitoring and control scheme for large and complex FPGA-based on-chip sensor networks and other related remote-sensing applications.

Future work

The dissertation has achieved its stated objectives in the field of embedded hardware systems. However, as the technology trend is fast moving towards embedded internet of things devices and software defined networking and big data paradigms, the future application of the proposed paradigm will be targeted towards the embedded internet of things (IoT) devices. It has been forecasted that around 50 billion devices will be connected by 2020. This means that the data generated by these devices can be autonomously uploaded into the cloud. This will enable us to keep track and analyze our key information and performance measurements such as heart rate, electric bills, the distance covered by cars, etc. The requirements of an embedded IoT device seems to be similar to the microprocessor systems. You have one or more sensors that are read by an MCU, the data may then be processed locally prior to sending it off to another application or trigger another event to occur. However, there are decisions to be made regarding the communication protocol, energy consumption and security consequences of the connected devices as well. To address these issues, the future direction will focus on a security functions to be performed by future IoT embedded devices, strategies to reduce energy consumed in executing complex data (Big Data), creation of a trusted execution environment on a single ultra-low power processor, protecting an embedded processor platform from malicious attacks and IP theft, managing the complexities in on-chip interconnect networks (Software Defined Networking) among others [169], [170], [171] and [172].

Bibliography

[1] M. S. Abdelfattah and V. Betz, "Design Tradeoffs for Hard and Soft FPGA-based Networks on-Chip", in an 11th International conference on Field programmable technology (FPT11), pp. 95-103, 2012.

[2] R. Ho, K. Mai, and M. Horowitz, "Managing Wire Scaling: A Circuit Perspective", in Proceedings of the IEEE 2003 International Interconnect Technology Conference, pp. 177-179, 2003.

[3] W. J. Dally and B. Towles, "Route Packets, not Wires: On-chip Interconnection Networks", in DAC '01, Proceedings of the 38th conference on Design automation, ACM Press New York, NY, USA, pp. 684-689, 2001.

[4] J. Duato, S. Yalamanchili, and L. Ni, "Interconnection Networks: an Engineering Approach", Morgan Kaufmann Publishers, pp. 62-98, 2003.

[5] Santambrogio, M.D., Hoffmann, H., Eastep, J., Agarwal, A., "Enabling Technologies for Self-Aware Adaptive Systems", in Proceedings of the NASA/ESA Conference on Adaptive Hardware and Systems, Anaheim, California, 15-18 June, pp. 149-156, 2010.

[6] S. Cheng, D. Garlan, B. Schmerl, P. Steenkiste, "Software Architecture-based Adaptation for GRID Computing", in Proceedings of the 11th International Symposium on High-Performance Distributed Computing (HPDC), Edinburg, UK, 24-26 July, pp. 389-398, 2002.

[7] P. Dini, et al. "Internet GRID, Self-Adaptability and Beyond: Are we Ready", in Proceedings of the 15th International Workshop on Database Expert Systems and Applications, Zaragoza, Spain, 30 August-3 September, pp. 782-788, 2004.

[8] G. Rengugadevi, M. G. Sumithra, "Hierarchical Routing Protocols for Wireless Sensor Network-A Survey", International Journal of Smart Sensors and Ad Hoc Networks, Volume *2*, pp. 71-75, 2012.

[9] A. Braman, G. R. Umapathi, "A Comparative Study on Advances in LEACH Routing Protocol for Wireless Sensor Networks: A survey", International Journal of Advanced Research in Computer Communication and Engineering, Volume *3*, pp. 5683-5690, 2014.

[10] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing", Computer Journal, Volume 36, pp. 41-50, 2003.

[11] HP Labs. "HP Open View Self-healing Services: Overview and Technical Introduction", in *User's Guide, Software Version: 2.60*, Hewlett Packard Development Company, California, USA, pp. 1-20, 2007.

[12] D. Breitgand, M. Goldstein, E. Henis, O. Shehory, Y. Weinsberg, "Panacea towards a self-healing development framework", in Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management, Munich, Germany, 21-25 May, pp. 169-178, 2007.

[13] S. Dustdar, C. Dorn, F. Li, L. Baresi, G. Cabri, C. Pautasso, F. Zambonelli, "A Roadmap towards Sustainable Self-aware Service Systems", in Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, Cape Town, South Africa, 1-8 May, pp. 10-19, 2010.

[14] A. Birolini, "Quality and Reliability of Technical Systems: Theory, Practice and Management", Springer-Verlag, Berlin, Germany, pp. 13-502, 1997.

[15] A. S. Tanenbaum, M. van Steen, "Distributed Systems: Principles and Paradigms", 2nd ed., Pearson Education Inc. Prentice Hall, Upper Saddle River, NJ, USA, pp. 321-373, 2007.

[16] L. M. S. Souza, H. Vogt, M. Beigl, "A Survey on Fault Tolerance in Wireless Sensor Network", Internal Report, Faculty of Computer Science, Department of Informatics, University of Karlsruhe, Germany, January 2007.

[17] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, F. Zambonelli, "A survey of autonomic communications", ACM Transactions on Autonomous and Adaptive Systems, Volume 1, 223-259, 2006.

[18] E. Gelenbe, R. Lent, "Power-aware ad hoc cognitive packet networks" Ad Hoc Network Journal, Volume 2, 205-216, 2004.

[19] K. Han, J. Luo, Y. Liu, A. V. Vasilakos, "Algorithm design for data communications in duty-cycled wireless sensor networks: A survey", IEEE Communication Magazine, Volume *51*, pp. 107-113, 2013.

[20] E. Gelenbe, R. Lent, A. Nunez, "Self-aware networks and QoS", Proceedings of the IEEE Journal, 16 August, Volume 92, Issue, pp. 1478-1489, 2004.

[21] Y. Zeng, K. Xiang, D. Li, A. V. Vasilakos, "Directional routing and scheduling for green vehicular delay tolerant networks",Wireless Network Journal, Volume *19*, 161-173, 2013.

[22] E. Gelenbe, "Steps toward Self-aware Networks", *Communication of the ACM, Volume 52*, pp. 66-75, 2009.

[23] L. Li, "Reliable Multicast with Pipelined Network Coding Using Opportunistic Feeding and Routing", IEEE Transactions on Parallel and Distributed Systems, Volume 25, 3264-3273, 2014.

[24] E. Gelenbe, M. Gellman, R. Lent, P. Liu, P. Su, "Autonomous Smart Routing for Network QoS", in Proceedings of the First International Conference on Autonomous Computing, New York, NY, USA, 17-19 May, pp. 232-239, 2004.

[25] J. Staddon, D. Balfanz, G. Durfee, "Efficient Tracing of Failed Nodes in Sensor Networks", in Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, Atlanta, GA, USA, 28 September, pp. 122-130, 2002.

[26] J. Tateson, C. Roadknight, A. Gonzalez, T. Khan, S. Fitz, I. Henning, N. Boyd, C. Vincent, "Real World Issues in Deploying a Wireless Sensor Network for Oceanography", in Proceedings of the Real-World Wireless Sensor Networks, Stockholm, Sweden, pp. 307-322, 2005.

[27] R. Szewczyk, J. Polastre, "Mainwaring, A.M.; Culler, D.E. Lessons from a Sensor Network Expedition", in *EWSN, LNCS 2920*, Springer-Verlag: Berlin, Germany, pp. 307-322, 2004.

[28] K. Wasilewski, J. W. Branch, M. Lisee, B. K. Szymanski, "Self-Healing Routing: A Study in Efficiency and Resiliency of Data Delivery in Wireless Sensor Networks", in *Proceedings of SPIE* 6562, Unattended Ground, Sea, and Air Sensor Technologies and Applications IX, 656218 Orlando, Florida, USA, May 11, 2007, doi: 10.1117/12.723515.

[29] W. Alec, T. Tong, D. Culler, "Taming the Underlying Challenges of Reliable Multi-Hop Routing in Sensor Networks", in Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, Los Angeles, CA, USA, 5-7 November, pp. 14-27, 2003.

[30] D. P. Robert, "Gradient Routing in Adhoc Networks", Massachusetts Institute of Technology Publication, XPOOR2419524, Cambridge, MA, USA, 2000. Available online:http://www.media.mit.edu/pia/Research/ESP/texts/poorieeepaper.pdf (accessed on: 03 June, 2015).

[31] F. Ye, G. Zhang, S. Lu, L. Zhang, "Gradient Broadcast: A Robust Data Delivery Protocol for Large Scale Sensor Network", ACM Wireless Network Journal, Volume11, 285-298, 2005.

[32] M. Hellsenbttel, T. Braun, T. Bernoulli, M. Waelchli, "BLR: Beaconless Routing Algorithm for Mobile Adhoc Networks", Journal of Computer Communication and Applied Service Wireless Network, Volume *27*, 1076-1086, 2004.

[33] M. Zori R. R. Rao, "Geographic Random Forwarding (GeRaf) for Adhoc and Sensor Networks Multihop Performance", IEEE Transactions on Mobile Computing, Volume 2, pp. 349-365, 2003.

[34] B. Blum, T. He, S. Son, J. A. Stankovic, "IGF: A Robust State-free Communication Protocol for Sensor Networks", University of Virginia, Department of Computer Science Technical Report, Virginia, USA, Retrieved from http://libra.virginia.edu/catalog/libraoa:1152. (Accessed on: 5 June 2015).

[35] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance Vector Routing (DSDV) for Mobile Computers", in Proceedings of the ACM SIGCOMM'94, London, UK, 31 August-2 September, pp. 234-244, 1994.

[36] T. Bourndenas, D. Wood, P. Zerfos, F. Bergamschi, M. Sloman, "Self-Adaptive Routing in Multi-hop Sensor Networks", in Proceedings of the 2011 7th International Conference on Network and Service Management (CNSM), Paris, France, 24-28 October, pp. 1-9, 2011.

[37] F. Hu and Q. Hao, "Intelligent Sensor Networks: The Integration of Sensor Networks, Signal Processing and Machine Learning", CRC Press, Taylor and Francis Group, Boca, Raton, FL, USA, pp. 411-589, 2013.

[38] B. Krishnamachari and S. Iyengar, "Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks", IEEE Transactions on Computers, pp. *53*, 241-250, 2004.

[39] D. S. Park, "Fault Tolerance and Energy Consumption Scheme of a Wireless Sensor Network", International Journal of Distributed Sensor Network, pp. 1-7, 2013.

[40] G. Chen, J. W. Branch, M. Pflug, L. Zhu, B. K. Szymanski, "SENSE: A Wireless Sensor Network Simulator", in Advances in Pervasive Computing and Networking 2005; Lisee, M., Chen, G.,Yener, B., Szymanski, B.K., Eds.; Springer: New York, NY, USA, pp. 249-267, 2006.

[41] T. S. Rappaport, "Wireless Communications Principles and Practice, 2nd Ed.; Prentice Hall, Upper Saddle River, NJ, USA, Dorling Kindersley, London, UK, pp. 1-709, 2009.

[42] C. Del-Valle-Soto, C. Mex-Perera, R. Monroy, J. A. Nulazo-Flores, "On Routing Protocol Influence on the Resilience of Wireless Sensor Networks to Jamming Attacks", Sensors Journal, Volume*15*, pp. 7619-7649, 2015.

[43] C. Del-Valle-Soto, C. Mex-Perera, O. Olmedo, A. Orozco-Lugo, G. Galván-Tejada, M. Lara, "On the MAC/Network/Energy Performance Evaluation of Wireless

Sensor Networks: Contrasting MPH, AODV, DSR and ZTR Routing Protocols", Sensors Journal, Volume14, 22811-22847, 2014.

[44] G. C. Gilbert, J. W. Branch and B. K. Szymanski, "A Self-Selection Technique for Flooding and Routing in Wireless Ad-Hoc Networks", Journal of Network and System Management Volume 14, 359-380, 2006.

[45] T. A. Babbitt, C. Morrell, B. K. Szymanski, "Self-Selecting Reliable Paths for Wireless Sensor Network Routing. Computer Communication Journal, Volume 31, pp. 3799-3809, 2008.

[46] G. Chen, B. K. Szymanski, "Component Oriented Simulation Architecture towards Interoperability and Interchangeability". In Proceedings of the Winter Simulation Conference, Arlington, VA, USA, 9-12 December, pp. 495-501, 2001.

[47] P. B. F. Duarte, Z. M. Fadlullah, A. V. Vasilakos, N. Kato, "On the Partially Overlapped Channel Assignment on Wireless Mesh Network Backbone: A Game Theoretic Approach", IEEE Journal on Selected Areas of Communications, Volume 30, pp. 119-127, 2012.

[48] L. Liu, Y. Song, H. Zhang, H. Ma, "Physarum Optimization: A Biology-inspired Algorithm for the Steiner Tree Problem in Networks", IEEE Transactions on Computers, Volume, 64, pp. 819-832, 2015.

[49] Y. Yao, Q. Cao, A. V. Vasilakos, "EDAL: An Energy-Efficient, Delay-Aware, and Lifetime-Balancing Data Collection Protocol for Wireless Sensor Networks", *IEEE/ACM Transactions on Networking, Volume 23*, pp. 182-190, 2015.

[50] E. Gelenbe, "A Diffusion Model for Packet Travel Time in a Random Multihop Medium. *ACM Transactions on Sensor Network.* Volume 3, pp. 1-19, 2007.

[51] X. Liu, J. Luo, A. V. Vasilakos, "Compressed Data Aggregation for Energy Efficient Wireless Sensor Networks", in Proceedings of the 2011 8th Annual IEEE Communication Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), Salt Lake City, UT, USA, 27-30 June, pp. 46-54, 2011.

[52] A. Attar, H. Tang, A. V. Vasilakos, F. R. Yu, "A Survey of Security Challenges in Cognitive Radio Networks: Solutions and Future Research Directions", Proceedings of the IEEE Journal, Volume *100,* pp. 3172-3186, 2012.

[53] E. Gelenbe, G. Sakellari, M. D'Arienzo, "Admission of QoS-aware Users in a Smart Network", *ACM Transactions on Autonomous and Adaptive Systems,* Volume *3*, pp. 1-28, 2008.

[54] A. Hind, and A. Agarwal, "A Multipath Routing Approach for Secure and Reliable Data Delivery in Wireless Sensor Networks". *International Journal of Distributed Sensor Network, Volume 2013*, pp. 1-10, 2013.

[55] E. Gelenbe and P. Liu, "Cognitive and Self-Selective Routing for Sensor Networks", *Computer Management Science Journal, Volume 8*, pp. 237-258, 2011.

[56] C. Chengqun, R. Yongfeng, L. Xin, Z. Yongqiu, and F. Wei, "An Efficiency Multiplexing Scheme and Improved Sampling Method for Multichannel Data Acquisition System", International Journal of Distributed Sensor Networks, Volume 2015, Article ID 626307, 9 pages, 2015.

[57] M. D. R. Perera, R. G. N. Meegama, and M. K. Jayananda, "FPGA Based Single Chip Solution with the 1-Wire Protocol for the Design of Smart Sensor Nodes", Journal of Sensors, Volume 2014, Article ID 125874, 11 pages, 2014.

[58] G. S. C. Nídia, G. G. Daniel, C. D. Flávia, J. V. N. Augusto, P. Luci, and N. de S. José, "Autonomic Context-Aware Wireless Sensor Networks", Journal of Sensors, Volume 2015, Article ID 621326, 14 pages, 2015.

[59] M. T. P. Jesús, C. D. Flávia, F. P. Paulo, G. Nadia, F. Lidia, and L. David, "Autonomic Wireless Sensor Networks: A Systematic Literature Review", Journal of Sensors, Volume 2014, Article ID 782789, 13 pages, 2014.

[60] J. Echanobe, I. Jelcampo, K. Basterretzea, M. V. Martinez, F. Doctor, "An FPGA-based Multiprocessor-architecture for Intelligent Environment", Microprocessors and Microsystems Journal, Volume 38, Issue 7, pp. 730-740, 2014.

[61] G. O. Carlos, I. Pablo, L. V. Marisa, "A Self-timed Multipurpose Delay Sensor for Field Programmable Gate Arrays (FPGA)", Sensors Journal Volume 14, pp. 129-143, 2014.

[62] M. Virgilio, M. Alessio M. Barbara, F. Gabriele, D. Paolo, "A universal Intelligent System-on-chip based Sensor Interface", Sensors Journal Volume 10 pp. 7716-7747, 2010.

[63] J. Franco, E. Boemo, E. Castilloo, L. Parrilla, "Ring Oscillator as Thermal Sensors in FPGAs: Experiments in Low Voltage", in Proceedings of Programmable Logic Conference (SPL), Ipojula, Brizal, March, pp. 133-147, 2010.

[64] R. K. Umanath, "Designing Next Generation Low Power Autonomous Sensor Nodes using Systems-on-chip based Solutions", Cypress Semiconductor, Published in EE Times Design, pp. 2-7, 2011.

[65] Y. S. Eugene, L. Kang, "Understanding IEEE 1451 Networked Smart Transducer Interface Standard", IEEE Instrumentation and Measurement Magazine pp. 11-17, 2008.

[66] W. Elmenreich, S. Pizek, "Smart Tranducers Principles Commmunications and Configurations", available: ttp://www.vmars.tuwien.ac.at/~wilfried/papers/2003/rr-10-2003.pdf.

[67] Smart Transducer Interface Specification Available: http://www.omg.org/docs/formal/03-01-01.pdf.

[68] Standard for a Smart Transducer Interface for Sensors and Actuators, Common Functions, Communication Protocols and Transducer Electronic Data Sheet (TEDs) Formats, IEEE STD 1451.0-2007, IEEE Instrumentation and Measurement Society, TC-9, the Institute of Electrical and Electronics Engineers, inc., New York, NY, 2007.

[69] W. J. Dally and B. Towles, "Principles and Practices of Interconnection Networks", Morgan Kauffman, 2004.

[70] M. K. Papamichael and J. C. Hoe, CONNECT: CONfigurable NetworkCreationTool. Available: "http://users.ece.cmu.edu/~mpapamic/connect/ (2013).

[71] H. S. Wang, L. S. Peh, and S. Malik, “Power-driven Design of Router Microarchitecture in On-chip Networks”, International Symposium on Microarchitecture, pp. 105-116, 2003.

[72] M. E. Kreutz, C. M. Marcon, L. Carro, F. Wagner, A. A. Susin, “Design Space Exploration Comparing Homogeneous and Heterogeneous Network-on-Chip Architectures”, in Proceedings of SBCCI, 2015.

[73] S. Coll, F. J. Mora, J. Duato, F. Petrini, "Efficient and Scalable Hardware-Based Multicast in Fat-tree Networks", IEEE Transactions on Parallel and Distributed Systems, Volume 20, Issue 9, pp. 1285 - 1298, 2009.

[74] T. M. Pinkson and J. Duato, “Appendix E Final Draft Interconnection Networks”, 2006.

[75] J. Kim, C. Nicopoulos, and D. Park, “A Gracefully Degrading and Energy-Efficient Modular Router Architecture for On-Chip Networks”, SIGARCH Computer Architecture News, Volume 34, Issue 2, pp. 4-15, 2006.

[76] D. Becker et al., “Open Source Network-on-a-chip Router”, Available:https://nocs.stanford.edu/cgibin/trac.cgi/wiki/Resources/Router,StanfordConcurrentVLSIArchitectureGroup, 2012.

[77] N. Jiang et al. “A detailed and Flexible Cycle-accurate Network-on-chip Simulator”, IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 86-96, 2013.

[78] X. Ju and L. Yang, “Performance analysis and comparison of 2x4 network on chip topology”, Microprocessors and Microsystems Journal, Volume 36, pp. 505 - 509, 2012.

[79] S. Kundu, J. Sounya and S. Chattopadhyay, “Design and evaluation of Mesh-of-Tree based Network-on-Chip using Virtual Channel router”, Microprocessors and Microsystems journal, Volume 36, pp. 471-488, 2012.

[80] J. Lee and L. Shannon, “The Effect of Node Size, Heterogeneity, and Network Size on FPGA based NoCs”, in an International Conference on Field-Programmable Technology (FPT), pp. 479-482, 2009.

[81] Y. Huan and A. DeHon, "FPGA Optimized Packet-Switched NoC using Split and Merge Primitives", in an 11th International Conference on Field Programmable Technology (FPT11), pp. 47-52, 2012.

[82] M. K. Papamichael and J. C. Hoe, "CONNECT: Re-examining Conventional Wisdom of Designing NoCs in the Context of FPGAs", in FPGA 12 ACM, pp. 37-46, 2012.

[83] E. S. Chung, M. K. Papamichael, E. Nurvitadhi, J. C. Hoe, and K. Mai, "ProtoFlex: Towards Scalable, Full System Multiprocessor Simulations Using FPGAs", ACM Transactions on Reconfigurable Technology and Systems, Volume 2, Issue 15, 2009.

[84] G. Schelle and D. Grunwald, "Exploring FPGA Network-on-Chip Implementations across Various Applications and Network Loads", in International Conference on Field Programmable Logic and Applications (FPL), pp. 41- 46, 2008.

[85] J. Lee and L. Shannon, "Predicting the Performance of Application-Specific NoCs Implemented on FPGAs", in Nineteenth ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA), pp. 23-32, 2011.

[86] N. Kapre, N. Mehta, M. deLorimier, R. Rubin, H. Barnor, M. J. Wilson, M. Wrighton, and A. DeHon, "Packet Switched vs. Time-multiplexed FPGA Overlay Networks", in FCCM, IEEE, pp. 205-213, 2006.

[87] M. K. Papamichael, J. C. Hoe, and O. Mutlu, "FIST: A Fast, Lightweight, FPGA-Friendly Packet Latency Estimator for NoC Modeling in Full-System Simulations", in Fifth IEEE/ACM International Symposium on Networks on Chip (NoCs), 2011.

[88] M. Moadeli, A. Shahrabi, W. Vanderbauwhede, and P. Maji, "An Analytical Performance Model for the Spidergon NoC with Virtual Channels", Journal of Systems Architecture, Volume 56, Issue 1, pp. 16-26, 2010.

[89] P. Del Valle, D. Atienza, I. Magan, J. Flores, E. Perez, J. Mendias, L. Benini, and G. Micheli, "A Complete Multiprocessor System-on-Chip FPGA-Based Emulation Framework", in IFIP International Conference on Very Large Scale Integration, pp. 140-145, 2006.

[90] G. Schelle and D. Grunwald, "Onchip Interconnect Exploration for Multicore Processors Utilizing FPGAs", in 2nd Workshop on Architecture Research using FPGA Platforms (WARFP), 2006.

[91] P. P. Pande, C. G. Grecu, M. Jones, A. Ivanor, R. Saleh, "Performance Evaluation and Design Trade-offs for Network-on-chip Interconnect Architectures", IEEE Transaction on Computers Volume 54, Issue 8, pp. 1025-1040, 2005.

[92] S. Abba and J. Lee, "Examining the Performance Impact of NoC Parameters for Scalable and Adaptive FPGA-Based Network-on-Chips", IEEE Fifth International Conference on Computational Intelligence, Modelling and Simulation (CIMSim), IEEE Computer Society, pp. 364-372, 2013.

[93] L. P. Tedesco, A. V. Mello, D. Garibotti, N. L. V. Calazans, F. G. Moraes, "Traffic Generation and Performance Evaluation for Mesh-based NoCs", Proceedings of the 18th Symposium on Integrated Circuits and Systems Design - SBCCI, pp. 184- 189, 2005.

[94] P. T. Wolkotte, K. F. Hölzenspies, J. M. Smit, "Fast, Accurate and Detailed NoC Simulations", in Proceedings of the International Symposium on Networks-on-Chip, pp. 323-333, 2007.

[95] N. Genko, D. Atienza, G. De Micheli, J. M. Mendias, R. Hermida, F. Catthoor, "A Complete Network-On-Chip Emulation Framework", in Proceedings, of Design, Automation and Test in Europe, pp. 246-251, 2005.

[96] G. C. Gilbert, J. W. Branch B. K. Szymanski, "Self-Selective Routing for Wireless Adhoc Networks", in Proceedings of the IEEE International Conference on Wireless and Mobile Computing, Networking and Communication, Montreal, Canada, 22-24 August, pp. 57-64, 2005.

[97] J. W. Branch, m. Lisee, B. K. Szymanski, "SHR: Self-Healing Routing for Wireless Adhoc Sensor Networks", in Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems, SPECTS'07, San Diego, CA, USA, 16-18 July, pp. 5-14, 2007.

[98] D. Johnson, D. Maltz, J. Broch, "DSR: The Dynamic Source Routing Protocol for Multihop Wireless Adhoc Networks", In *Ad Hoc Networking*; Perkins, C.E., Ed., Addison-Wesley, Boston, MA, USA, pp. 139-172, 2001.

[99] I. D. Chokers, M. B. R. Elizabeth, "AODV: Routing Protocol Implementation Design", in Proceedings of the 24th International Conference of Distributed Computing Systems and Workshops. W7-EC (ICDCSW'04), Tokyo, Japan, 23-26 March, pp. 698-703, 2004.

[100] W. R. Heinzelman, J. Kulik, H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks", in Proceedings of the ACM MobiCom, Seattle, WA, USA, 15-20 August, pp. 174-185, 1999.

[101] J. H. Huijsing, F. R. Riedijk and G. Van der Horn, "Development in Integrated Smart Sensors", Sensors and Actuators A-Physical, Volume 43, pp. 276-288, 1994.

[102] K. Georgios and P. Dionisions, "A Survey and Taxonomy of On-chip, Monitoring of Multi-core Systems-on-chip", ACM Transactions on Design Automation of Electronic Systems, Volume 18, Issue 2, 2013.

[103] J. G. Gabriel, A. J. Carlos, P. Jorge, A. Aiman, M. P. Lucas, T. Fernando, T., "A Survey on FPGA-based Sensor Systems: Towards Intelligent and Reconfigurable Low-power Sensors for Computer Vision, Control and Signal Processing", Sensors Journal, Volume 14, pp. 6247-6278, 2014.

[104] N. V. Kirianaki, S. Y. Yurish, N. O. Shpak, V. P. Deynega, "Data Acquisition and Signal Processing for Smart Sensors", John Willey and Sons: Hoboken, NJ, USA pp. 320-350, 2002.

[105] J. Xi, C. Yang, A. Mason, P. Zhong, "Adaptive Multisensor Interface System-on-Chip", in Proceedings of the 5th IEEE Conference on Sensors, Deegu, Korea, pp. 50-53, 2006.

[106] S. Y. Yurish, "Digital Sensors Design based on Universal Frequency Sensors Interfacing IC", Sensors and Actuators, A-Physical, Volume 132, pp. 265-270, 2006.

[107] G. Song, A. Song and W. Huang, "Distributed Measurement System Based on Networked Smart Sensors with Standardized Interfaces", Sensors and Actuators A-Physical, Volume 120, pp. 147-153, 2005.

[108] Smart Sensors Interface for IEEE 1451. Available online http://www.jlm-innovation.de/products/ieee1451.

[109] K. M. Zick and J. P. Hayes, "Online Sensing for Healthier FPGA Systems", in Proceedings of the 18th Annual ACM/SIDA International Symposium on Field Programmable Gate Arrays, ACM, New York, NY, USA, pp. 239-248, 2010.

[110] G. Kornaros and D. Pnevmatikatos, "Real time Monitoring of Multicore SoCs through Specialized Hardware Agent on NoC Network Interfaces", in Proceedings of the 26th IEEE International Parallel and Distributed Processing Symposium, Workshops and PhD Forum, pp. 249-255, 2002.

[111] L. Fiorin, G. Palermo, C. Silvano, "MPSoCs Runtime Monitoring through Networks-on-chip", in Proceedings of the Design Automation and Test in Europe Conference, Leuven, Belgium, pp. 558-561, 2009.

[112] J. T. Aquirre, V. Baena-Lecuyer, J. Mora, J. Carrasco, A. Torralba, L. Franquelo, "Microprocessors and FPGA Interfaces for Insisting Co-debugging in Field Programmable Hybrid Systems", Microprocessors and Microsystems Journal, Volume 29, 75-85, 2005.

[113] W.Y. Alexander, L. Pasi, R. Pekka, N. Ethiopia, I. Jouni, T. Hannu, "Hierarchical Agent Architecture for Scalable NoC Design with Online Monitoring Sensors", in Proceedings of MICRO 41.

[114] C. Ciordas, T. Basten, A. Radulescu, K. Goossens, J. Meerbergen, "An Event-Based Network-on-Chip, Monitoring Service", in Proceedings of the 9th IEEE International High-level Design Validation and Test Workshop, pp. 149-154, 2004.

[115] Y. Wang, X. Jiang, H. Shengxi, L. Weichen, Y. Huazhong, "A Case Study of On-chip Sensor Network in Microprocessor System-on-Chip", CASES 09, Grenoble, France, ACM, pp. 11-16, 2009.

[116] S. Andrew, "PSoC-Based Low-cost, Intelligent Network: Physical and Data Link Layer", Cypress Semiconductor, Application Note AN2346, Revision A, pp. 1-16, 2006.

[117] V. Petrescu, M. Pelgrom, H. Veendrick, P. Pavithran, J. Wieling, "Monitors for a Signal Integrity Measurement System", in Proceedings of the 32nd European Solid-State Circuits Conference, pp. 122-125, 2006.

[118] R. McGowen, C. Poirier, C. Bostak, J. Ignowski, Millican, M.; Parks, W.; S. Naffziger, "Power and Temperature Control on a 90nm Itanium Family Processor", Solid-State Circuits, IEEE Journal, Volume 41, pp. 229-237, 2006.

[119] K. Sohn, N. Cho, H. Kim, K. Kim, H. S. Mo, Y. B. Suh, H. G. Byun, H. J. Yoo, "An Autonomous SRAM with On-Chip Sensors in an 80nm Double Stacked Cell Technology", VLSI Circuits Symposium on Digest of Technical Papers, pp. 232-235, 2005.

[120] C. Chan, Y. Chang, H. Ho, H. chiueh, "A Thermal-aware Power Management Soft IP for Platform-based SoC Design", in proceedings of International Symposium on Systems-on-chip, pp. 181-184, 2004.

[121] M. Eduardo, R. Manuel, P. Fernando, H. David, G. Enrique, "An FPGA Embedded Web Server for Remote Monitoring and Control of Smart Sensors Networks", Sensors Journal, Volume 14, pp. 416-430, 2014.

[122] C. Gomez-Ouna, M. A. Sanchez, P. Ituero, M. Lopez-Vallejo, "A Monitoring Infrastructure for FPGA Self-Awareness and Dynamic Adaptation", in Proceedings of the 9th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Sevilla, Spain, pp. 10-13, 2012.

[123] R. E. Neapolitan, "Probabilistic Methods for Bioinformatics: with an introduction to Bayesian Network," Morgan Kaufman publishers, Elsevier, 2009.

[124] M. A. Al Faruque et al., "AdNoC: Runtime Adaptive Network-on-Chip Architecture" IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Volume 20, Number 2, pp. 257 - 269, 2012.

[125] D. Zoni, S. Corbetta and W. Fonaciari "Thermal / Performance Trade-off in Network-on-Chip Architectures", IEEE Internation Symposium on SoC, 2012.

[126] J. R. Venkateswara and P. R. Sudhakara, "Design and Implementation of Efficient On-Chip Crosstalk Avoidance CODECs using Fibonacci Numeral System", IEEE

Computer Society Fourth International Conference on Computational Intelligence, Modelling and Simulation, CIMSim, pp. 352-356, 2012.

[127] P. Gratz et al., "On-chip interconnection networks of the TRIPS chip," IEEE Micro Journal, Volume 27, Number 5, pp. 41-50, 2007.

[128] D. Wentzlaffet al., "On-chip Interconnection Architecture of the Tile Processor," IEEE Micro Journal, Volume 27, Number 5, pp. 15-31, 2007.

[129] R. Marculescu et al., "Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives," IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, Volume 28, Number 1, pp. 3-21, 2009.

[130] F. Angiolini et al., "Networks on chips: From research to products," in Proceedings of DAC 2010, pp. 300-305, 2010.

[131] K. Goossens et al., "The Aethereal network on chip after ten years: Goals, Evolution, Lessons, and Future," in Proceedings of DAC, pp. 306-311, 2010.

[132] J. Howard et al., "A 48-core IA-32 Message-passing Processor with DVFS in 45 nm CMOS," in Proceedings of ISSCC, pp. 108-109, 2010.

[133] L. P. Carloni et al., "Networks-on-chip in Emerging Interconnect Paradigms: Advantages and Challenges," in Proceedings of NOCS 09, pp. 93-102, 2009.

[134] W. J. Dally et al., "Route packets, not wires: On-chip Interconnection networks," in Proceedings of Design Automation Conference, pp. 684- 689, 2001.

[135] U. Y. Ogras et al., "Key Research Problems in NoC Design: A holistic Perspective," in Proceedings of CODES+ISSS, pp. 69-74, 2005.

[136] W. Karl et al., "SCI monitoring hardware and software: Supporting Performance Evaluation and Debugging," in Proceedings of SCI, pp. 417-432, 1999.

[137] C. Nicopoulos et al., "ViChaR: A Dynamic Virtual Channel Regulator for Network-on-chip Routers," in Proceedings MICRO, pp. 333-34, 2006.

[138] V. Nollet et al., "Operating-system Controlled Network on chip," in Proceedings of DAC 2004, pp. 256-259, 2004.

[139] Xilinx, Inc., VirtexT and XTFPGAs Datasheet [Online]. Available:http://www.xilinx.com/support/documentation/user_guides/ug470_7Series_Config.pdf 2013.

[140] V. Betz and J. Rose, "FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density", ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA. 1999.

[141] V. Betz and J. Rose, "Directional Bias and Nonuniformity in FPGA Global Routing Architecture", International Conference on Computer-Aided Design, San Jose, CA, USA, 1996.

[142] D. Lewis et al. "The Stratix™ Routing and Logic Architecture", ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Montery, CA, USA, 2003.

[143] C. Schafer, M. Stojilovic and L. Saranovac, "Analysis of Impact of FPGA Routing Architecture Parameters on Area and Delay" in an IEEE 19th Telecommunications Forum, TELFOR 2011, Serbia Belgrade, November 22-24, pp. 924-927, 2011.

[144] ModelSim SE (6.5c) Mentor Graphics Corporation. Available at: http:www.mentor.com.

[145] Bluespec, Inc., Bluespec System Verilog, Available: http://www.bluespec.com/products/bsc.htm.

[146] Xilinx, Inc., Virtex-7T and XTFPGAs Datasheet [Online]. Available: http:// http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/xst_v6s6.pdf (2013).

[147] E. Ahmed and J. Rose, "The Effect of LUT and Cluster size on Deep-submicron FPGA Performance and Density", IEEE Transaction on Very Large Scale Integration (VLSI) Systems, Volume 12, Issue 3, 288-298, 2004.

[148] C. E. Leiserson, "Fat-Tress: Universal Network for Hardware-Efficient Supercomputing ", IEEE Transaction on Computers, C-34, Issue 10, pp. 892-901, 1985.

[149] D. Wang, N. Jerger, and J. Steffman, "DART: A programmable Architecture for NoC Simulation of FPGAs", In Fifth IEEE/ACM International Symposium on Networks on Chip (NoCs), pp. 145-152, 2011.

[150] C. Intanagonwiwat, R. Govinda, D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks". In Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, Boston, MA, USA, 6-11 August, pp. 56-67, 2000.

[151] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "A survey on sensor networks", in *IEEE Communication Magazine*, IEEE Communication Society, 7 November, Volume 40, Issue 8, pp. 102-114, 2002.

[152] Greedy Algorithm. Available online: http://www.encyclopediaofmath.org/index.php?title=Greedy_algorithm&oldid=34629 (accessed on: 6 June, 2015).

[153] J. Lunze, L. Grune, "Control Theory of Digitally Networked Dynamic Systems", Springer International Publishing, Switzerland, 2014.

[154] K. J. Phillip, "Feedback Control of Computer Systems (Introducing Control Theory to Enterprise Programmers)", O'REILLY Media, Inc. Publisher, California USA, 2014.

[155] L. H. Joseph, D. Yixin, P. Sujay, M. T. Dawn, "Feedback Control of Computing Systems", A John Wiley and Sons Publication, 2004.

[156] R. Dafal, J. P. Diquet, "Self-adaptive Network Interface (Sani): Local Component of a NoC Configuration Manager", in Proceedings of International Conference on Reconfigurable Computing and FPGAs, ReConFig 09, 296-301, 2009.

[157] Synopsys Inc., SaberRD Integrated Design Environment (IDE), Student Edition, available at: http://www.synopsys.com/cgi-bin/saberrd/reg1.cgi, 2015.

[158] Synopsys Inc. "SaberRD User Guide", Version V-2004.06-SP1, Document Order Number: 00000-000 VA, Version W-2004.09, 2004.

[159] Xilinx Inc., Xilinx Vivado 2013.3 Integrated Design Environment (IDE), Available at: http://www.xilinx.com/support/download.html, 2015.

[160] Xilinx Inc., SYSMON User Guide, UG58 (v1. 2), Available at: http://www.xilinx.com/support/download.html, 2015.

[161] Xilinx Inc., XADC User Guide, UG480 Available at: http://www.xilinx.com/support/download.html (2015).

[162] J. S. Mrinal, S. P. Radhey, "System Monitoring using Zynq-700 AP SoC Processing System with the XADC AXI Interface", Xilinx inc. Application Note: Zynq-7000 All Programmable SoC, XAPP1182, V1.0, 2013.

[163] J. Pallav, A. Srinivasa, J. S. Mrinal, "Using the Zynq-7000 Processing System (PS) to Xilinx Analog to Digital Converter (XADC) Dedicated Interface to Implement System Monitoring and External Channel Measurements", Xilinx Inc. Application Note: Kintex-7 Family and Zynq-7000 AP SoC, XAPP1172 V1. 01 2014.

[164] S. Hauck and A. DeHon, "Reconfigurable Computing the Theory and Practice of FPGA-Based Computation", Morgan Kaufmann Publishers, Elsevier Inc. 2008.

[165] K. Hermann, B. Gunther, "The Time-Triggered Architecture", Proceedings of the IEEE Journal, Volume 90, Issue 1, pp. 112-126, 2003.

[166] M. Platzner, J. Teich, N. Wehn,"Dynamically Reconfigurable Systems Architectures, Design Methods and Applications", Springer Publishers, Dordrecth Heidelberg, 2010.

[167] F. D. Patric, "Measurement and Data Analysis for Engineering and Sciences", Second Edition, Taylor and Francis Publisher, Indiana, USA, pp. 58-134, 2010.

[168] L. Guang, "Hierarchical Agent-Based Adaptation for Self-Aware Embedded Computing Systems", PhD Thesis, Department of Information Technology, University ofTurku,Finland.Availableat:www.doria.fi/bitstream/handle/10024/86210/thesis_Liang__Guang.pdf?sequence=1. 2012.

[169] K. Karimi et al. "White Paper: Integrating the Internet of Things: Necessary Building Blocks for Broad Market Adoption ", Atmel Corporation, Revatmel-0776-corporation-IOT-Whipepaper-us-102014, San Jose, CA 95110 USA, pp. 1-18, 2014.

[170] J. A. Stankovic, "Research Directions for Internet of Things", IEEE Journal of Internet of Things, IEEE Communications Society, Volume 1, Issue 1, pp. 3-9, 18 March 2014.

[171] B. A. A. Nunes et al. "A Survey of Software-Defined Networking: Past, Present and Future of Programmable Networks ", IEEE Communication Surveys and Tutorials, IEEE Communications Society, Volume 16, Issue 3, pp. 1617-1634, 13 February, 2014.

[172] J. Changqing, L. Yu, Q. Wenming, U. Awada, L. Keqiu, "Big Data Processing in Cloud Computing Environments", in 12th International Symposium om Pervasive Systems, Algorithms and Networks (ISPAN), IEEE, pp. 17-23, 13-15 December 2012.

[173] S. Abba and L. Jeong-A, "A Parametric-based Performance Evaluation and design Trade-offs for Interconnect Architectures using FPGAs for Network-on-Chips" Microprocessor and Microsystems, Embedded Hardware Design: EUROMICRO Journal, Vol. 38, pp. 375-398, 2014.

[174] S. Abba and L. Jeong-A, "An Autonomous Self-Aware and Adaptive Fault-Tolerant Routing Technique for Wireless Sensor Networks", Sensors Journal, Vol. 15, Issue 8, pp. 20316-20354, 2015.

[175] S. Abba, and L. Jeong-A, "FPGA-Based Design of an Intelligent On-chip Sensor Network Monitoring and Control using Dynamically Reconfigurable Autonomous Sensor Agents". Int. Journal of Distributed Sensor Networks (Accepted 28, Dec. 2015. In press). Available at: www.hindawi.com/journal.ijdn/aip/201609/.

Acknowledgement

The journey begins when I received an email message from my PhD supervisor Prof. Jeong-A Lee from Korea. She informed me that I was selected as a recipient of the Korean Government Scholarship Award (KGSP 2010). Being a PhD student is a challenging and complex task to be achieved. It involves learning, experience, hard working, humility, honesty, patience and perseverance. First of all, I would like to thank the God Almighty for making my dream becomes true and sparing my life to achieved a Doctoral Degree. Glory be to Him. I would like to thank my supervisor for giving me the opportunity to work and serve under her supervision. It was a great time and life long experience. Special thanks are owed to the dissertation committee members. Professor Beom-Joon Cho, the committee chair for his support and elderly advice. Prof. Pankoo Kim, Prof. Sangman Moh, Prof. and Prof. Cheol Hong Kim of Chonnam National University, Korea, for their insightful and constructive comments and advice on my research work. Thank you all for your time, understanding and making this work in better shape.

Special thanks are owed to the Korean Government through the National Institute for International Education and Development (NIIED) for providing the financial support for my PhD study. These include the monthly living allowance, tuition fees, research allowance, health insurance, transportation, etc. Thank you very much for the kindness and generosity. Also, I would like to express my sincere gratitude to the Chosun University for giving me the opportunity to study and providing the financial support for my research work.

I would like to thank the management of Abubakar Tafawa Balewa University (Federal University of Technology) Bauchi, Nigeria, for granting the approval and permission to study abroad to obtain a PhD degree in Computer Engineering from

Chosun University, Gwangju, South Korea.

Special thanks are owed to the Heidelberg Laureate Forum Foundation (HLFF) Germany for the inspired selection and invitation as a Young Researcher to attend the 2014 Heidelberg Laureate Forum in Germany. An international gathering of the Laureates of computer science and mathematics and selected young researchers from all over the world to meet in a conducive and intellectual atmosphere for a period of one week to inspire the young researchers. Thank you very much for all the supports, including; travel grant, hotel accommodation, meals, transportation, insurance, etc. to mention but a few. I would like to thank all the participating laureates, the distinguished scientist, the German governments, and to all the supporters and contributors of the HLFF 2014. Sincere thanks are owed to Professors Leslie Lamport and Manuel Blum the Turing Award Laureates for their inspired advice during our discussion at the 2nd Heidelberg Laureate Forum 2014 in Germany. Sincere thanks are owed to the Professors Vicki L. Hanson, vice president of the association for computing machinery (ACM) and John Richards of IBM research for their inspired advice on-board cruise at the 2014 Heidelberg Laureate Forum. Thank you for organizing this wonderful intellectual gathering.

Special thanks are owed to the Schloos Dagstuhl Liebnitz Centre for Informatics Germany, for the selection and invitation as a Young Researcher to attend the Dagstuhl GI Seminar "Control Theory meets Software Engineering" in September 2014. Sincere thanks are owed to the Seminar organizers, Associate Professors Antonio Filieri of Stuttgart university Germany and Martina Maggio of Lund university Sweeden. I was inspired, motivated and learnt a lot from the seminar. Thank you very much for the kindness and generosity.

Special thanks are owed to the organizers of the IEEE Communications Society Summer School university of Trento, Italy for the selection and invitation to attend the summer school. Sincere thanks are owed to Prof. Fabrizio Granelli the

chair and organizer of the summer school university of Trento. Thank you very much for your kind assistance. I would like to extend my sincere thanks and gratitude to the potential speakers during the summer school. Prof. Lajos Hanzo (University of Southhampton, U.K.) for his lecture on "*Cooperative communications*", Prof. Giuseppe Bianchi (University of Rome Tor Vergata, Italy) for his lecture on "*From dumb to smarter switches in software defined networks: an overview of data plane evolution*", Prof. Andrea Goldsmith (Stanford University, U.S.A.) for her lecture on "*The next wave of wireless communications*" and Prof. Nelson L.S. da Fonseca (State University of Campinas, SP, Brazil) for his lecture on "Networking for big data". Thank you all, for your time and the inspired lectures.

Finally, I would like to express my sincere gratitudes and thanks to my parent for their support and understanding throughout my study. My mother Maryam Abubakar for nurturing and bringing me up. My late father Abba Abdullahi for his support and encouragement. It was a very hard situation when I received the sad message about his death while I was writing this dissertation. My father, may your gentle soul rest in perfect peace and Jannatul Firdaussi (the highest paradise) be the final abode Ameen. I would like to dedicate this work to my late father. Sincere thanks to my uncle Alhaji Mustapha Abdullahi for his support through my educational career. Thank you all.

List of Publications

1. Abba, S. and Jeong-A, Lee. "A Parametric-based Performance Evaluation and design Trade-offs for Interconnect Architectures using FPGAs for Network-on-Chips" Microprocessor and Microsystems, Embedded Hardware Design: EUROMICRO (SCI / SCI-E) Journal, Vol. 38, pp. 375-398, 2014. Available: doi http://dx.doi.org/10.1016/j.micpro.2014.04.011.
2. Abba, S. and Jeong-A, Lee. "An Autonomous Self-Aware and Adaptive Fault-Tolerant Routing Technique for Wireless Sensor Networks", Sensors (SCI-E) Journal, Vol. 15, Issue 8, pp. 20316-20354, 2015. Available: doi: 10.3390/s150820316.
3. Abba, S. and Jeong-A, Lee, "FPGA-Based Design of an Intelligent On-chip Sensor Network Monitoring and Control using Dynamically Reconfigurable Autonomous Sensor Agents". Int. Journal of Distributed Sensor Networks. (SCI-E) Journal (Accepted 28, Dec. 2015. In press). Available at: www.hindawi.com/journal.ijdn/aip/201609/.).
4. Abba, S. and Jeong-A, Lee "Self-aware and Adaptive Fault Tolerant Routing Protocol for Networks-on-Chip Architectures using Particle Swarm Optimization" Microprocessor and Microsystems, Embedded Hardware Design: EUROMICRO (SCI / SCI-E) Journal (To be Re-submitted).
5. Abba, S. and Jeong-A, Lee. "Run-time Firmware-based Approach for a Minimization of Dynamic Partial Reconfiguration Overhead for Self-Adaptive Reconfigurable Systems, " IEEE Transactions (SCI/SCI-E) Journal (To be Re-submitted).
6. Abba, S. and Jeong-A, Lee., "GSASRA: Globally Self-Adaptive and Scalable Routing Algorithm for Network-on-Chips Architecture using Particle Swarm Optimization", IEEE International Conference on Artificial Intelligence, Modeling and Simulation (AIMS2013), IEEE, Computer Society, pp. 393-399, 3-5 Dec. 2013. Available: doi 10.1109/AIMS.2013.72.
7. Abba, S. and Jeong-A, Lee, "Examining the Performance Impact of NoC Parameters for Scalable and Adaptive FPGA-Based Network-on-Chips" in the 5th IEEE International Conference on Computational Intelligence Modeling and Simulation", IEEE Computer Society, pp. 384-372, 24-25 Dec. 2013. Available: doi 10.1109/CIMSim.2013.65.