Real-Time Embedded Application Core Mapping for NoC with Enhanced Performance and Energy Efficiency

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Electronics and Communication Engineering

Submitted by

Mr. Aruru Sai Kumar

(Roll No.: 717129)

Under the Supervision of

Dr. T.V.K. Hanumantha Rao

Associate Professor

Department of Electronics and Communication Engineering

National Institute of Technology Warangal, India

November 2021

Department of Electronics and Communication Engineering
National Institute of Technology Warangal
Warangal, Telangana, India-506004

Certificate

This is to certify that the work contained in this thesis entitled **"Real-Time Embedded Application Core Mapping for NoC with Enhanced Performance and Energy Efficiency"** is submitted by **Mr. Aruru Sai Kumar (Roll No.: 717129)** to the Department of Electronics and Communication Engineering, National Institute of Technology Warangal, for the partial fulfillment of the requirements for the degree of **Doctor of Philosophy** in **Electronics and Communication Engineering**.

He has carried out his work under my supervision. This work has not been submitted else-where for the award of any other degree or diploma.

The research work in our opinion, has reached the standard fulfilling of the requirements for the degree of Doctor of Philosophy in Electronics and Communication Engineering in accordance with the regulations of the Institute.

....

Dr. T.V.K. Hanumantha Rao(Supervisor)
Associate Professor, Department of ECE
NIT Warangal

....

(HoD)
Department of ECE
NIT Warangal

Declaration

This is to certify that the work presented in this thesis entitled **"Real-Time Embedded Application Core Mapping for NoC with Enhanced Performance and Energy Efficiency"** is a bonafied work done by me under the supervision of **Dr. T.V.K. Hanumantha Rao** and was not submitted elsewhere for the award of any degree.

I declare that this written submission represents my own ideas and even considered others ideas which are adequately cited and further referenced the original sources. I understand that any violation of the above will cause disciplinary action by the institute and can also evoke panel action from the sources or from whom proper permission has not been taken when needed. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea or data or fact or source in my submission.

Aruru Sai Kumar

Place:

Research Scholar

Date:

Roll No.: 717129

Abstract

Name of the student: **Mr. Aruru Sai Kumar** Roll No: **717129**

Degree for which submitted: **Ph. D.**

Department: **Electronics and Communication Engineering**

Thesis title: **Real-Time Embedded Application Core Mapping for NoC with Enhanced Performance and Energy Efficiency**

Thesis supervisor: **Dr. T.V.K. Hanumantha Rao**

Month and year of thesis submission: **November 2021**

Abstract

As going with the present trend, the number of components on the On-chip architectures have been increased drastically when compared with previous years. The types of On-chip architectures were distinguished into 3 types, namely: a) System on Chip (SoC), b) Multi-Processor System on Chip (MPSoC), and c) Network on Chip (NoC). SoC offers products with high complexity, high computational capacity and high-value semiconductor. One of the core aspects of assessing a network's performance is the efficiency of on-chip architectures. Due to the increase of the perpetual scaling in the VLSI technology, cores are getting less complex. As a result, many cores are incorporated onto a single processor for simultaneous development and performance enhancement, leading to the origination of MPSoC and NoC architectures. In the development of high-performance parallel processors, multiprocessing chips (MPSoC) play a crucial role. The NoC accommodate modularly and the scalable nature and its contribution to the most efficient on-chip communication possibly leads to NoC-based multiprocessor systems. In the NoC design architecture, one of the pre-eminent part is the routing algorithm which assures a low communication latency makes the implementation of the hardware effortless and a high network throughput.

This thesis proposes an efficient core mapping algorithm named BMAP, which involves a standard topology selection and customization of NoC platforms. The proposed algorithm was applied to various benchmark applications as case studies and was compared with previous algorithms. A great improvement in reliability, delay, area and power was observed where minimal objective function is achieved even for larger complex networks in 2D topologies. The SPLASH-2 Benchmark synthesis provides the performance evaluation of the proposed mapping, where the experimental outcomes reveals that the performance metrics such as Speed-up Execution Time increased by 40%, 30% and 20%, Latency reduced by 42%, 34% and 28%, Energy efficiency improved by 36%, 30%, 26% and Power consumption reduced by 32.6%, 28.2%, and 26.4% when compared with NMAP, MMAP, and EMAP algorithms respectively.

The proposed ACM, mainly involves in two steps 1) Core mapping and 2) Spare core placement. The core mapping methodology considered the complete performance evaluation for different routing algorithms. The proposed ACM can dynamically react and recover from the failed core through spare core replacement to maintain system functionality. Before performing the core mapping, we need to find out the mapping region through NAD and PVR. The Weighted Communication Energy (WCE) for each mapped region is been calculated, so therefore, the obtained minimum WCE provides the best core mapping. Once the mapping is completed, if a fault occurs at any core, perform the fault diagnosis method, which determines the location of the damaged resource and correct the error using error detection and correction mechanism. If the faults occur even after applying the fault diagnosis method, perform task migration using spare core placement.

Mapping simulations executed on the NoC Simulator (Noxim), adjusted to perform a re-enactment for various routing algorithms, such as XY, WEST-FIRST, NORTH-LAST, and ODD-EVEN routings. A significant improvement observed in energy consumption, average delay, and throughput in the proposed work compared with existing algorithms. The proposed adaptive core mapping algorithm is synthesized and simulated using Vivado Design Suite 2018.3 and verified on the Kintex-7 FPGA KC705 board. The results implicate a dramatic decrease in area, power consumption, and an increase in throughput. The core mapping is applied to the PARSEC Benchmark using GEM5 simulator which outperform the latency and system performance.

In this Fault-tolerant mapping algorithm (FTMAP) is implemented that focuses predominantly on replacing the faulty cores and assessing the communication and the execution time of the network by employing it on various multimedia benchmarks. Each mapped core is associated with a router, i.e. responsible for the data transfer from source via destined core. In contrast, the unmapped cores will be available as the free spare cores. During the application task execution, the permanent faults were addressed through the proposed fault-tolerant mechanism, where the faulty cores were substituted with the nearest accessible free core to perform the tasks smoothly and efficiently. Considering the failure probabilities, FTMAP provides the outcomes of NFT, 1FT and 2FT that improves the system's performance, minimizes the communication energy and the execution time.

In this an efficient mapping strategy implemented on the real-time embedded applications named ERTEAM. In this algorithm, based on the minimum Node Average Distance (NAD) the mapping region is finalized, ensuring the overall mapping area reduced. The PE's mapped according to the minimum communication energy in the selected mapping region. This research evaluated a set of embedded applications, which reveals a reduction in latency at 12.3% against BBPCR and 8.4% against SBMAP. The simulation time reduces at an average of 19% against BBPCR and 9.6% against SBMAP. The throughput increases at an average of 14.5% against BBPCR and 7.8% against SBMAP and reduces the communication energy by 15.6% against BBPCR and 5.2% against SBMAP.

Index Terms: System on Chip (SoC), Network on Chip (NoC), Core, Application Core Mapping, Spare Core Placement, Fault tolerance, Failure Recovery, Processing Core Failure, Performance metrics, Kintex-7 FPGA KC705 Evaluation Kit.

Acknowledgements

The past four years time in National Institute of Technology Warangal are one of the most valuable, unforgettable and successful periods in my life. I gain a lot of novel knowledge and skills from my research, and harvest precious friendships that I will cherish forever. I must express my appreciation to all people who helped me and guided me in the past.

First and foremost, I would like to express my sincerest gratitude to my supervisor, Dr. T.V.K Hanumantha Rao has supported me throughout my doctoral study, which allows me to explore promising research topics freely and flexibly. With his solid knowledge and high-level guidance, I have been able to tackle challenging issues and make progresses in my research.

I take this privilege to thank all my Doctoral Scrutiny Committee members Prof. L. Anjaneyulu, Prof. N. Bheema Rao, Dr. P. Srihari Rao, and Prof. N. Vishwanathan (Dept. of EEE) for their detailed review, constructive suggestions, and excellent advice during the progress of this research work. I would also like to thank all the faculty of Dept. of ECE who helped me during the course.

I would like to convey my sincere gratitude to Dr. B. Naresh Kumar Reddy, Ms. P. Rajeshwari, Sri K. Sarangam, Dr. M. Satish, Dr. N. Srinivas, Dr. P. Siva Rama Krishna, Dr. Santhosh Kumar, Mr. B. Roshan, Mr. K. Rambabu, and Dr. Suman Bhowmik for thier invaluable guidance and support. I would also like to extend my heartfelt appreciation to my family, colleague scholars, friends, and well-wishers who helped to write my thesis with their support. Finally, I thank my nation India, for giving me the opportunity to carry out my research work at the NIT Warangal. A special thanks to MHRD for its financial support.

Date: ....

Place: (**Aruru Sai Kumar**)

Contents

List of Figures

List of Tables

Abbreviations

ACG	**A**pplication **C**ore **G**raphs
ACM	**A**daptive **C**ore **M**apping
AF	**A**cceleration **F**actor
ASIC	**A**pplication **S**pecific **I**ntegrated **C**ircuit
AV	**A**udio **V**ideo.
BT	**B**inary **T**ree
BFT	**B**inary **F**at **T**ree
CG	**C**ore **G**raph
CMOS	**C**omplementary **M**etal **O**xide **S**emiconductor
CMP	**C**hip **M**ulti **P**rocessor
DVFS	**D**ynamic **V**oltage and **F**requency **S**caling
ERTEAM	**E**fficient **R**real-**T**ime **E**mbedded **A**pplication **M**apping
FIFO	**F**irst **I**n **F**irst **O**ut
FPGA	**F**ield **P**rogrammable **G**ate **A**rray
FTMAP	**F**ault **T**olerant Core **MAP**ping
FTNoC	**F**ault **T**olerance **N**etwork **o**n **C**hip
GALS	**G**lobal **A**synchronous, **L**ocally **S**ynchronous
HDL	**H**ardware **D**escription **L**anguage
IC	**I**ntegrated **C**ircuit
IP	**I**ntellectual **P**property
ITRS	**I**nternational **T**echnology **R**oadmap for **S**emiconductors
MMS	**M**ulti **M**edia **S**ystems
MPEG	**M**oving **P**icture **E**xpert **G**roup

MPSoC	**M**ulti **P**rocessor **S**ystem **on** **C**hip
MWD	**M**ulti **W**indow **D**isplay
NAD	**N**odes **A**verage **D**istance
NFT	**N**on **F**ault **T**olerance
NI	**N**etwork **I**nterface
NoC	**N**etwork **o**n **C**hip
NTG	NoC **T**opology **G**raph
PE	**P**rocessing **E**lement
PVR	**P**lacing unmapped **V**ertices **R**egion.
QoS	**Q**uality **o**f **S**ervice
SA	**S**witch **A**llocation
SDR	**S**oftware **D**efined **R**adio
SoC	**S**ystem **o**n **C**hip
TCH	**T**otal **C**ore **H**ours
TG	**T**ask **G**raph
TGFF	**T**ask **G**raph **F**or **F**ree
VA	**V**irtual **C**hannel **A**llocation
VC	**V**irtual **C**hannel / **V**irtual **C**ircuit
VLSI	**V**ery **L**arge **S**cale **I**ntegration
VOPD	**V**ideo **O**bject **P**lane **D**ecoder
WCE	**W**eighted **C**ommunication **E**nergy
XB	**C**rossbar
1FT	**O**ne **F**ault **T**olerance
2FT	**T**wo **F**ault **T**olerance

Symbols

A	Area
A_{NoC}	Total NoC area
c_i	Core number i
c_f	Free cores
c_{ij}	Number of hops between cores i and j
C_{um}	Unmapped Core
D	Communication Distance
E	Directed Edge
e_{ij}	Directed edge between the cores i and j
F_{xy}(t)	Failure probability of the processing core,
L_n	The clock cycle latency for the n^{th} node
M	Mapping region
N	Number of cores in the application
N_b	Number of bits per packet
N_f	Number of failed cores
N_p	number of clocks cycles lapsed from the first generated packet to the last received packet
P	Failure Probability
P_{NoC}	Total NoC power consumption
R	Reliability
R_{NoC}	Overall NoC reliability
R_p	total number of received packets
S	Spare core
T	Temperature of the core

t_{xy}	The x^{th} row and y^{th} column of the tile
V	Vertex
V_n	Number of nodes or vertices in task graph
V_{um}	Unmapped Vertices
λ_{xy}	failure rate
μJ	Micro Joules
$\alpha(t)$	Number of bits that reaches till time t

Dedicated to my parents and teachers

Chapter 1

Introduction

This chapter highlights challenges in computer platform design and introduces the Network on-Chip concept. We also give an overview of the research presented in the thesis and outline the authors contributions to the enclosed papers

1.1 Challenges in computing platform design

Embedded applications have grown in popularity and demand as chip technology has evolved, particularly System on Chip (SoC) prevailing due to its compactness. A new ecosystem has originated for semiconductor devices, which allows complex tasks and features to be integrated into a single package, referred to as SoC. As per International Technology Roadmap for Semiconductors, named ITRS 2.0, an emerging ecosystem comprises heterogeneous implementation with electronic components linked to various application domains, including High-Performance Computing (HPC), IoT, Big Data, including Cloud Computing [1]. The architecture utilized in SoC design is bus-based structures that could not evolve well as an application's communication needs an expansion. The communication between the cores of these architectures is accomplished through the following techniques, namely (i) Bus based approach, i.e. intended as a traditional method that permits the communication of each core at a time which generally utilized in SoC-based architectures, (ii) Point to Point communication links contains a designated link to each core to communicate with the other core, which generally utilized in MPSoC based architectures, and (iii) Router-based communication utilized in NoC architectures [2], [3].

Multiple processors can be integrated on a single chip evolving as MPSoC's (Multi Processor System on Chip), that involves tens or hundreds processor elements, memory blocks, ASIC acceleration engines to be inter-connected together on chip. Each processing element performs its tasks in a parallel way taking the advantage of the parallelism either in task, thread or system level. Many recent chips have already switched to the paradigm of multi-core based platform for this purpose. For example, in [4], [5], [6] a 16-core heterogeneous digital base band IC for MIMO 4G Software Defined Radio (SDR) is proposed. Their proposed NoC-based prototype doubles the throughput and consumes only 39% power over the previous MPSoC solutions. Another example is the Intel 80-tile Teraflops processor [7], which is a homogeneous NoC-based CMP platform and delivers up to 1.28 TFlops of performance. Therefore, MPSoC based architectures provide great improvement in the performance and reduces the energy dissipation.

The Network on Chip (NoC) connectivity approach is developed as a solution to resolve the limitations of communication, performance metrics and energy efficiency [8], [9]. NoC is mainly classified into two types based on its characteristics: 1) General purpose and 2) Application specific. The communication between the cores are dynamically designed for the application specific NoC [10].

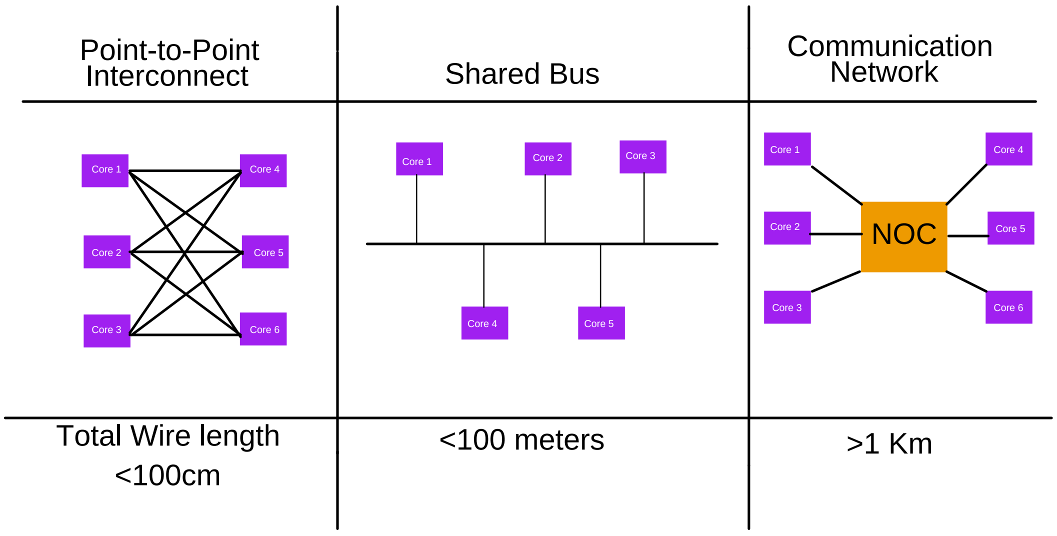

FIGURE 1.1: The trend of on-chip interconnection.

1.2 NoC for multi-core communication

NoC provides a versatile and customizable communication network that substitutes designated point-to-point connectivity among the cores, resulting in inter-core communication [11], [12]. The communication between the cores is achieved through the routers linked to each core. In evaluating the performance of the multi-core architectures, NoC router efficiency plays a prominent role [13], [14]. The NoC architectures comprise of three main processes for the application design, namely; (i) Scheduling Tasks, (ii) Application Partition, and (iii) Mapping the application. Every task in a directed graph is allocated to the separate cores during the task scheduling stage. Perhaps, the timeline of their operations is specified whenever two tasks along the same core are scheduled. After completing this stage, the core graph and the communication energy between the vertex's are obtained. The next stage involves the application mapping of the cores, where each core is mapped onto the available PE [15].

Figure 1.1 represents the trend of on-chip interconnection and compares the total wire length under different technology nodes [16]. For the NoC-based multicore system, besides the latency and throughput improvement, it also brings the following advantages:

1) High reliability: For Multicore systems, the complex system is highly susceptible to faults [17]. Compared to point-to-point dedicated links and buses, NoC can achieve higher reliability by providing redundant paths among the cores. If some of the cores fell into permanent or temporary faults, the other cores can be utilized to remap the packets to the destinations and hence packet acceptance rate will not drop dramatically.

2) Modular design : NoC provides sufficient bandwidth for communication, while the processors can be designed without considering the network; therefore it supports modularity design [18]. Moreover, the global clock synchronization is not necessary in NoC which increase the overall system yield [19].

3) Global asynchronous, locally synchronous (GALS) design: For multicore systems, it is difficult to distribute a single clock over thousands processor cores. To deal with these issues, NoC offers a good platform for the GALS design style because each tile (processor elements and the cores) can work separately within its own clock domain. By employing GALS design, multiple Voltage-Frequency islands can be developed in different regions of NoC so as to achieve lower power consumption [19].

4) Power and area efficiency: Compared to the buses, the arbitration time for contention is much smaller as each core only needs to handle local contention scenario [17].

Therefore, large buffers to store the unserved packets are not needed in NoC cores, which result in a more compact core mapping and reduces the area/power overhead. For power dissipation, because the buses are connected to all the cores in the system, while the links in NoC only need to connect two neighboring cores. Therefore, with proper floorplanning, NoCs uses shorter wire length and occupies less load per transition [17]. Moreover, NoCs provide a variety of efficient power management strategies to further reduce power. This is because the NoC can be partitioned into sub-networks and each region can be powered-off or slowed down via dynamic voltage and frequency scaling (DVFS) individually [17]. High power efficiency can be achieved without significant degradation in the overall system performance.

1.2.1 Network Terminology

Some network terminologies which are commonly used are described below.

Message : The message is actual data to be transferred from source core to destination core in NoC. The size of message may be fixed or variable and it depends on the application.

Packet : The message can be broken into several packets. Packet is a small formatted block that can be transmitted from a source core to destination core. The packet can move independently in the network. Every packet consists of control information and data (payload). The packet header carries the control information. The size of packet may be fixed or variable.

Flow Control Digit (Flit) : A packet may be broken down into several flits. The size of flit consist usually one or several bytes. It fits the storage resources in switches in the network.

Physical Transfer Digit (Phit) : It is the smallest physical unit of information at physical layer. It consists of a constant number of bits. It is transferred as a unit across a channel from one router to the next router. Size of phit may or may not be equal to the size of flit.

Communication units clearly shown in Figure 1.2

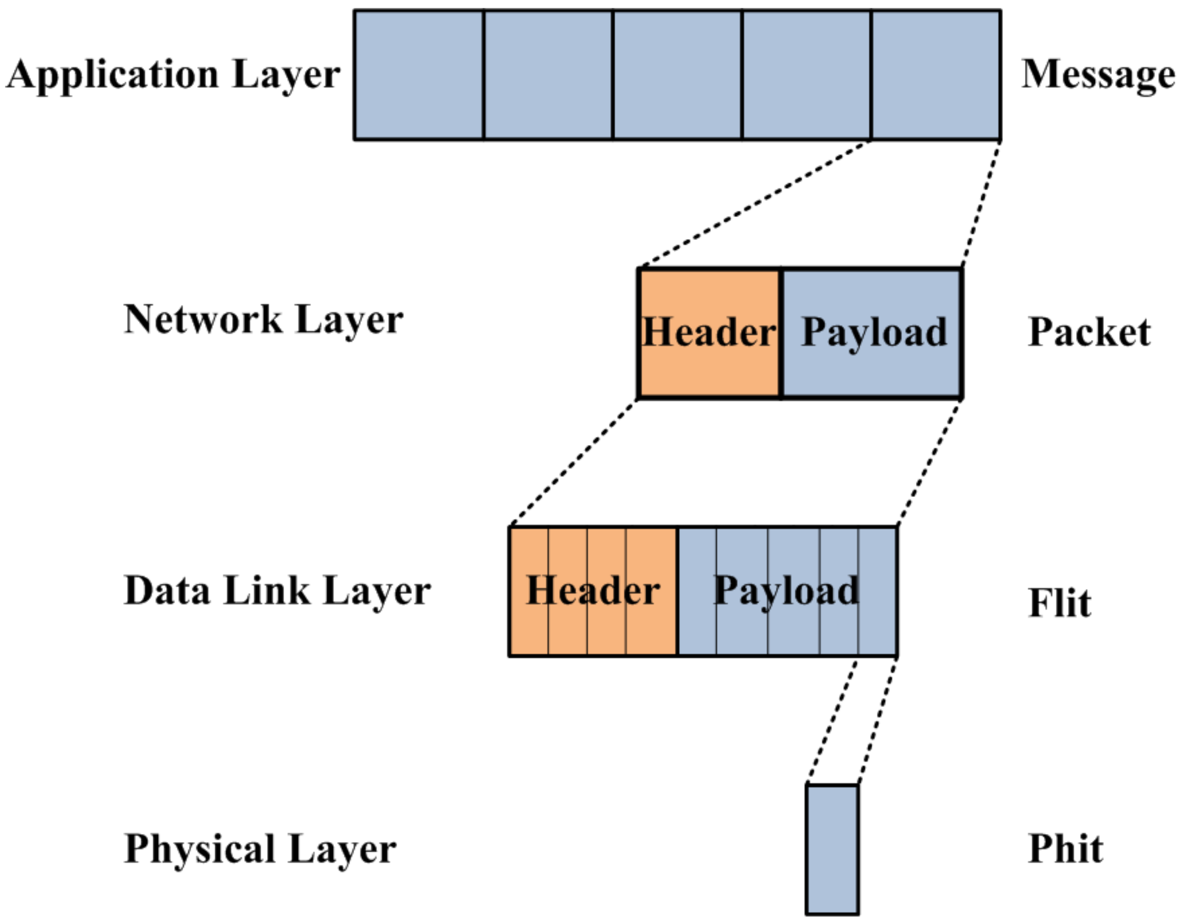

FIGURE 1.2: Communication Units.

1.2.2 Basic Advantages of NoC

NoC design paradigm has many advantages over the previous paradigm of bus SoC interconnection. The following subsections discuss the basic advantages of NoC design paradigm and how those advantages contribute to the electronic devices industry [20].

1.2.2.1 Scalability

Scalability is the ability to increase the number of modules on the SoC. In bus based SoC, it takes long time to add new function units to an existing design because designers had to design and test the system from scratch. Bus arbitration, data integrity, loads, and others have to be tested. In NoC, adding new modules or function units became easier. You only have to add a new router and connect this router to the network by small bus or even connect the module to an existing router depending on the network design. On the other hand, companies can buy a ready designed and tested functions, IPs from other third-party companies and commercial off-the shelf (COTS) modules from IP vendors then use them in their projects. That effectively reduces the strong pressure time-to-market (TTM) demand. So, scalability is the ability to increase the number of modules easily by just inserting a router and a network interface and we do not have to rebuild the design from scratch (placing the modules and routing the buses) to extend our design.

1.2.2.2 Reusability

The concept of reusability emerged with NoC, where in order to design a totally new product or a new application, designers may use many simple predesigned modules to make larger systems. This concept can reduce time of design and test where designers do not have to think about new architectures or testing the system from scratch, and instead, they test the interconnection communication in the network. In other words, reusability means design once and use many. Reusability and scalability reduces the time of testing the design by just testing the traffic instead of testing all the system specially the low level modules.

1.2.2.3 Reduced Communication Delay

Contrary to the nature of communication in bus based SoC where only one communication channel can be established between two nodes at a time. Connecting a SoC as a network enabled the nodes to establish many successful communication channels every time slot. That in most cases reduces the communication latency and the total operation time.

1.2.2.4 Reduced Communication Power Consumption

In bus based SoC, every message has to be delivered to all nodes on the bus, and every node has to check whether the message is destined to its own or not. But in NoC, messages are routed inside the network only to its destination by the communication routers. That reduces the unwanted power consumption in transmitting the messages to unwanted destination and also the power consumption in checking the destination in each node.

1.3 NoC Architecture

NoC Architecture contains three main elements [21] such as 1) Processing Core, 2) Network Interface and 3) Router, clearly represented in Figure 1.3.

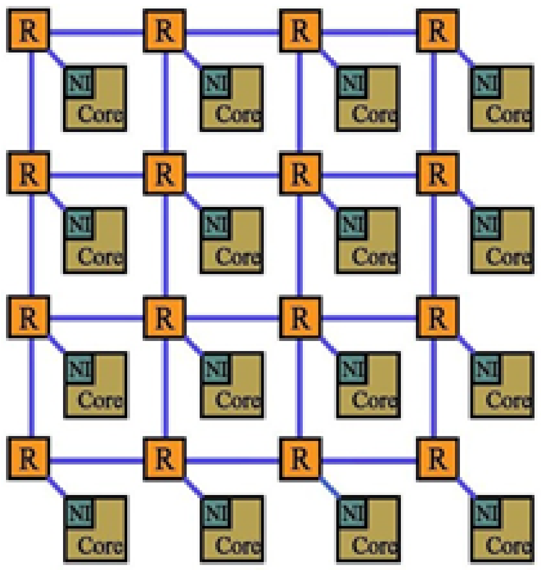

FIGURE 1.3: NoC Architecture.

1.3.1 Processing Core

The cores in NoC communicate with each other through interconnection links using a technique named packet-based switching. The processing cores in NoC are mainly classified into 3 types based on its action [22], which is clearly depicted in Figure 1.4.

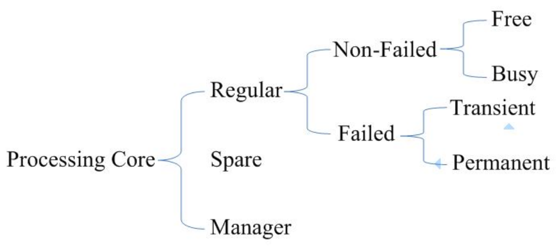

FIGURE 1.4: Processing Core Types.

1) Regular Core: The regular core is further classified into 2 types namely; Failed and Non-Failed core. The Failed cores can either be transient faults or permanent faults

whereas the Non-Failed cores are divided into either free or a busy core. Therefore, all such type of cores are considered as the regular cores [23].

2) Spare Core: It is responsible for recovering the failed cores in a network. Suppose if any of the core gets failed permanently in an application, the spare core is replaced in the place of the faulty core to complete the tasks of the faulty core. Therefore, all such spare cores are responsible for serving the tasks of failed cores [23], [24].

3) Manager Core: It contains the status of the busy cores present in the network and performs task migration to the spare core, whenever a faulty core occurs. Therefore, these type of cores are termed as manager core [23].

The communication between the core and the router are carried out through the network interface as well as each router is connected with the neighboring router in order to form a packet based NoC architecture.

1.3.2 Network Interface

The term network interface is basically defined as the communication between the core and the router that primarily performs the data packetization and depacketization as shown in Figure 1.5. The data is portioned into packets on basis of certain length which transmits them to the connected network, this is termed as packetization [25]. At the receiving terminal end, the depacketisation is responsible to reconnect the fragmented packets from router [26].

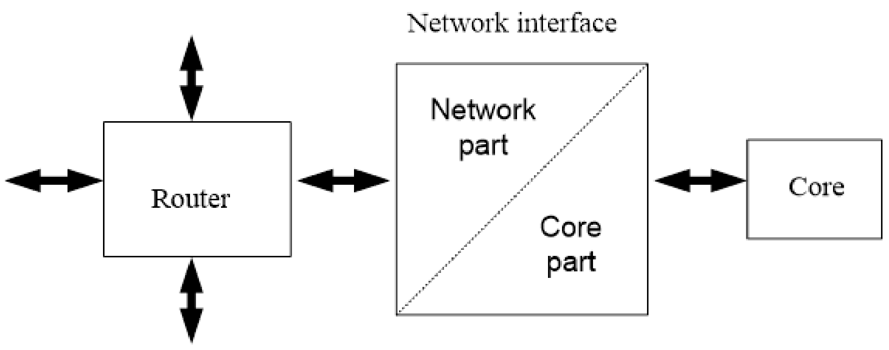

FIGURE 1.5: Basic Network Interface.

1.3.3 Router

Router is one of the main element and is called as the heart for the NoC architecture. Basically router contains five number of input and the output ports each, out of 5 ports, 4 ports are aligned in the directions such as north, east, west, and south, whereas the other port is between the local core and router. Every input port contains 'V' virtual channels (1:V demultiplexer, V:1 multiplexer), hence in router as it contains 3 virtual channels it is considered as 1:3 demultiplexer, 3:1 multiplexer. The router can utilize virtual channels as buffers to store the packets and each virtual channel contains its own state (virtual channel state). The router's control logic is made up of router computation (RC), virtual channel allocation (VA), and switch allocation (SA), which are in need of verifying that all of a packet's required resources were assigned to the downstream router [27]. The connection between the routers input and output ports, allowing packets to transit from an input port's buffer to a downstream router more efficiently is performed by the Crossbar switch (XB). The four phases of router pipeline are RC, VA, SA, and XB. A packet in NoC contains one head flit that allocates the resources for a packet, one or more body flits which comprises of packet payload and one tail flit that free up the packet resources [28]. In order to decrease the latency of a network, we implemented a VIP based Virtual channel architecture which improves the system performance as well. This is applied on 4x4 mesh based NoC and the simulated outcomes were more favorable than wormhole routing design.

1.4 NoC Topology

The physical architecture and interconnections among nodes and routers in the NoC network are determined by the term topology. The number of hops or the routers required for the transfer of message via source to destination can be determined using the topology. The network's length is determined that impacts the latency and also determines the throughput. Topology enables alternative pathways among nodes, allowing traffic to be distributed equally over the NoC, lowering network latency and increasing network connection bandwidth usage [29]. The different types of NoC topology are Mesh (M), Torus (T), Ring (R), Polygon (P), Binary tree (BT), 2-ary 3-fly (BFT23), 2-ary 2-stage clos (CLOS), 2-ary 2-fly (BFT22), clearly shown in Figure 1.6.

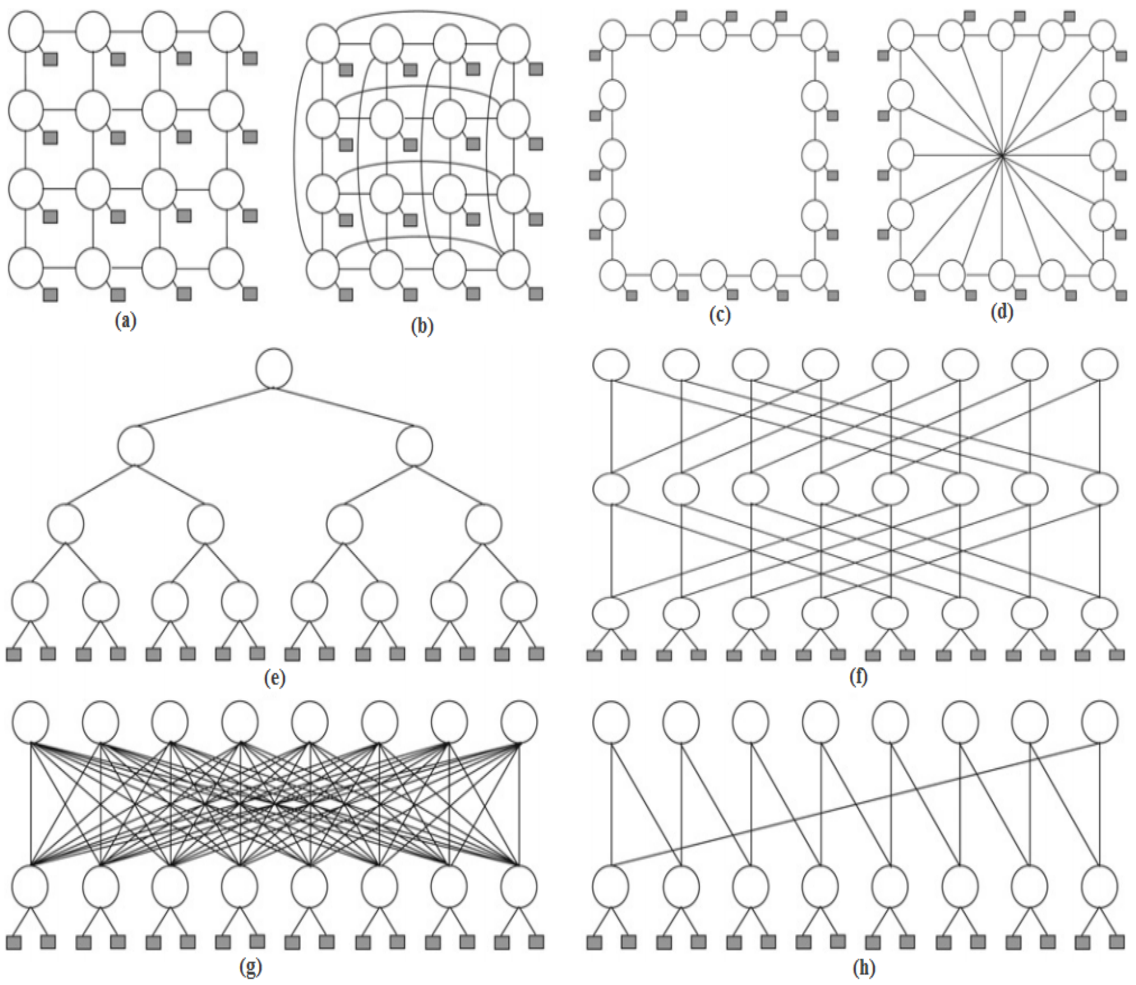

Figure 1.6: NoC Topologies: (a) Mesh (M), (b) Torus (T), (c) Ring (R), (d) Polygon (P), (e) Binary tree (BT), (f) 2-ary 3-fly (BFT23), (g) 2-ary 2-stage clos (CLOS), (h) 2-ary 2-fly (BFT22).

1.5 Challenges in the Design of NoC

The application mapping is considered one of the key elements for improving the performance parameters and reducing the communication energy of the network [30]. The application mapping is considered as an example for the NP-hard optimization issue and is further classified into two categories, namely; (i) Static mapping and (ii) Dynamic mapping, where static mapping contains static paths which are used for transferring information from specific source via destination and doesn't contain the network's current state. Dynamic mapping actions were decided based on the network's current state, and paths among both source and destination can evolve over time based on the traffic requirements. The failure probability has been increased, which results as one of the main drawback of NoC. Furthermore, due to the complexity of production and testing the system configuration, effective communication is a key challenge, that results more number of faults [31], [32]. The faults in a network are classified into 3 types namely: (i) Transient faults, (ii) Intermittent faults and (iii) Permanent faults, where the faults appeared can be resolved permanently are termed as transient faults, the faults that occurring frequently are termed as intermittent faults and finally the permanent faults are persistent that can be replaced to perform the assigned tasks. The fault-tolerant methods can be used to overcome the faults in a network [33], [34].

1.6 NoC and Application model

In this section, we discuss about the background of multi-applications, NoC Core graph, topology, the assessment of reliability, energy, and performance of the NoC system. The background of application core mapping is discussed, where NoC core graph and topology acts as its input. Every application has multiple parallel tasks. By using static analysis, the data traded among cores can probably be regulated. The data indicate how frequently a packet is transmitted between the source and destination in the network.

1.6.1 Background

As presented in [35], the communication rate between the vertices denoted by a core graph [36]:

Definition 1: An Application Core Graph is a directed graph ACG(V, E) as shown in Figure 1.7 (a). 'V' represents the vertex ($\forall V_i \in V$) and the tasks as in the vertex are assigned to the same IP, each directed edge 'e_{ij}' in E characterizes the communication from vertex V_i to vertex V_j, while 'W(e_{ij})' characterizes communication rate from vertex V_i to vertex V_j.

According to [37], the connectivity and communication of the NoC is represented by a topology graph [38]:

Definition 2: The architecture of the NoC platform (concentrated on cores only) is presented in Figure 1.7 (b). The NoC is represented as topology graph TG (IP, D), where IP represents the core. $\forall t_{xy} \in$ IP, 't_{xy}' represents the x^{th} row and y^{th} column of the tile, 'D' represents the communication distance $\forall IP_{ij} \in$ D, and 'IP_{ij}' denotes the distance between core (IP_i) and core (IP_j).

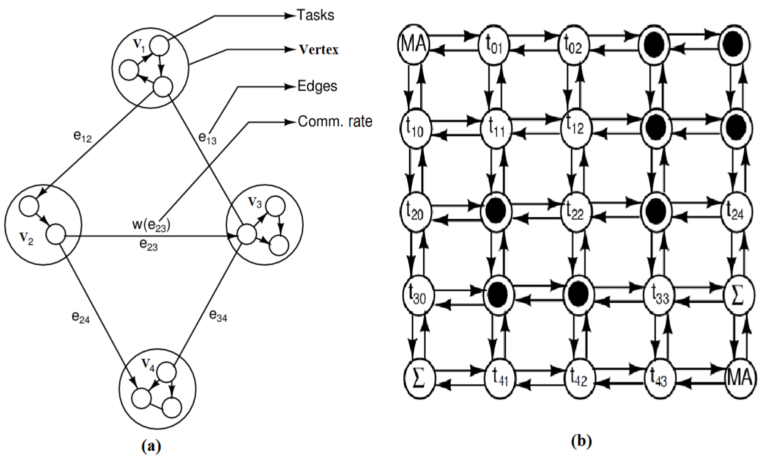

FIGURE 1.7: (a) An example of ACG, (b) An example of NoC.

1.6.2 Application Mapping and Scheduling

An efficient real-time embedded application mapping problem is defined as:

Definition 3: Given a set of Application Core Graph (ACG), A = ACG(V, E) and NoC Topology Graph (NTG),T = TG (IP, D) , finding a mapping function M(IP, D) that maps an IP core $ip_i \in$ IP in the ACG to a PE in the NoC.

$\forall V_i \in V$,

$\forall IP_i \in IP$,

$\Omega(V_i) \in IP$,

$V_i \neq V_j \Rightarrow \Omega(V_i) \neq \Omega(V_j)$

$\forall \mathrm{IP}_{ij} \in D$

Let $e_{ij} \in$ E be mapped to some $V_{xy} \in$ IP then $V_{xy} = \Omega(\mathrm{IP}_i) \in$ D.

The above mentioned core graph ACG(V, E), and the topology TG(IP, D) are considered as the primary inputs for application mapping. Each node present in the task graph i.e. $v_i \in \{$ V $\}$ is allocated to the free cores present in the topology based on the designed algorithm in order to minimize the total communication energy. It is important that the number of free cores present in the selected topology should always be more, when compared to the total number of nodes/vertices present in the selected task graph [39]. It is mathematically represented in Eq. (1.1).

$$V_n \leq IP(c_f) \tag{1.1}$$

where V_n is represented as number of nodes or vertices in task graph and IP(c_f) is represented as number of free cores in selected topology.

1.6.3 Efficiency Model

In this section, we mainly discuss the evaluation of reliability, communication energy and the performance metric. Each mentioned metric plays a vital role in the assessment of application core mapping.

1.6.3.1 Reliability Assessment

Reliability is assessed in two steps: 1) Identifying the most favorable mapping algorithm with respect to performance and weighted communication energy and 2) Assessing the reliability of faulty core mapping. The reliability of fault core mapping of the i^{th} faulty core among the number of failed cores 'n' for a particular application is represented by $R^A_{i,n}$. The reliability of core mapping for an application [40] is as follows:

$$R^A = \sum_{n=0}^{N} \sum_{i=1}^{M} R^A_{i,n} P_{I,n} \tag{1.2}$$

Where $P_{I,n}$ denotes the faulty probability when the i^{th} faulty core occurs, which is indicated in Eq. (1.3), and I indicates a set of 'n'faulty cores for the i^{th} faulty condition.

$$P_{I,n} = \prod_{j=1, j\in I}^{N} (p_j) \prod_{j=1, j\notin I}^{N} (1 - p_j) \tag{1.3}$$

Where P_j represents the faulty probability of the j^{th} core.

Then the reliability is normalized by a normalization factor N_R, which is obtained above equation when considering the network that can tolerate the maximum number of faulty links. The normalized reliability is defined as

$$\text{NR} = R/N_R \tag{1.4}$$

1.6.3.2 Evaluation of Communication Energy

Communication energy is considered as the distance between two tiles or nodes [41]. It is calculated as the sum of differences between their corresponding modules determines the distance among two vertices, i.e. V_i and V_j, where V_i having parameters as $(x_1,\ y_1)$ and V_j having parameters as (x_2,y_2).

Therefore, the Weighted Communication Energy (WCE) calculated as mentioned in Eq. (1.5).

$$WCE = \sum_{\forall t_i \in \{T\}} W(E_{ij}) \times \{|(x_2 - x_1)| + |(y_2 - y_1)|\} \tag{1.5}$$

where $W(E_{ij})$ is illustrated as the communication weight between any two nodes in a network.

Problem 1 : To find out total communication energy of the network (CE_{Total}).

To perform the mapping of an NoC application, let's consider task graph G(V, E), and the topology N(IP, D); find the mapping function MF : V $\rightarrow$ IP , where $v_i \in \{\ \text{V}\ \}$ and $ip_{ij} \in$ IP.

Let us map the respective v_i to ip_{ij}.

$\Rightarrow$ MF$(v_i) = IP(c_f) \qquad \forall\ v_i \neq v_j$: MF$(v_i) \neq$ MF(v_j)

The total communication energy (CE_{Total}) is given as

$$CE_{Total} = \sum_{\forall v_i \in \{V\}} W(E_{ij}) \times X_{ij} \tag{1.6}$$

$$CE_{Total} = \sum_{\forall v_i \in \{V\}} W(E_{ij}) \times \{|(x_2 - x_1)| + |(y_2 - y_1)|\} \tag{1.7}$$

where the distance between two nodes v_i and v_j, with v_i parameters as $(x_1,\ y_1)$ and v_j parameters as $(x_2,\ y_2)$ is represented as X_{ij} and $W(E_{ij})$ as communication energy between two nodes.

1.6.3.3 Evaluation of Performance

The throughput and latency considered important metrics for performance improvement [42]. Since network congestion significantly impacts latency, avoiding congestion for each node is an efficient way to minimize latency. Simultaneously, less congestion will result in increased throughput. As a result, the bandwidth limitation, which is interrelated to congestion, is considered the performance limitation. Therefore, the communication volumes for each node managed through bandwidth restrictions, so congestion is diminished, and performance, including latency and throughput, is guaranteed [43].

Throughput is considered to be the first order performance metrics in systems. It is the maximum traffic from the network that can be cleared.

$$Throughput = \frac{(number\ of\ completed\ messages\ X\ packet\ length)}{(number\ of\ cores\ X\ total\ times)} \tag{1.8}$$

In our experiment, we expected the message length to be equal to packet length. The number of completed messages denotes the number of messages that successfully reach the destination. Packet length is measured in terms of flits/phits. The number of cores denotes the number of cores involved in a given application, and total time is the time between the first message transmission and last message reception.

1.7 Importance of Application Core Mapping

Application core mapping is considered as one of the key factor in order to improve the performance of the real-time NoC applications. Based on the requirements the application mapping can be either a static or a dynamic. NoC provides the required resources to establish the communication between the cores and computing the application. The NoC application comprises of various elements such as DSP's, FPGA's, IP cores, Memory blocks, CPU's. Due to the increase in the transistors on a single On-chip, there is a high probability of faults in an application which results in core crashes, non-transmission of network messages etc. So, in order to decrease the faults and improve the system performance, the spare core replacement methodology is proposed.

1.8 Problem Statement

In this research work, we proposed a real-time embedded application core mapping techniques. It incorporates a core graph unit, which is responsible for mapping and scheduling the core graph on the NoC architecture. In order to increase the performance and reduce the communication energy of the NoC applications, mainly concentrated on 3 main approaches: 1) Core mapping, 2) Spare core replacement (enhances the reliability of the processor) and 3) Fault-tolerant core mapping. In this research, we proposed two types of mapping, one is system level mapping using preliminaries like Nodes Average Distance (NAD) and Placing unmapped Vertices Region (PVR). Another is BMAP algorithm which provides an efficient application core mapping. After providing an effective core mapping, the spare core replacement is used for replacing the spare core (free core) in the place of faulty core that helps in continuing the tasks of the faulty cores, resulting in the completion of the assigned tasks and run the applications smoothly. This leads to the performance improvement of an application along with other metrics. Fault tolerance unit collects all the fault information from the mapped NoC platform and provides various solutions for different faults.

1.9 Authors Contributions

The thesis is based on a collection of papers. The detailed materials, experiments, results and other related work are referred to the papers. In the following, we summarize the enclosed papers highlighting the authors contributions. These papers are also listed in the publications.

- International Journals:

    1. **Aruru Sai Kumar** and T.V.K. Hanumantha Rao, “Scalable benchmark synthesis for performance evaluation of NoC core mapping,” *Microprocessors and Microsystems*, 2020.**(Elsevier, SCI Indexed)**

       This paper proposes an efficient core mapping technique that is used for customizing the NoC platforms and synthesizing the SPLASH-2 benchmark for producing scalable performance evaluation. This algorithm comprises of standard selection of the topology and performing the efficient core mapping. This mapping algorithm outperformed the performance metrics such as Speed-up

Execution Time increased by 40%, 30% and 20%, Latency reduced by 42%, 34% and 28%, Energy efficiency improved by 36%, 30%, 26% and Power consumption reduced by 32.6%, 28.2%, and 26.4% when compared with NMAP, MMAP, and EMAP algorithms respectively.

Authors Contribution: The author developed and proved the algorithm, formulated the problem, wrote the program, conducted experiments and wrote the manuscript. The implementation and experiments were conducted on Noxim Simulator.

2. **Aruru Sai Kumar** and T.V.K. Hanumantha Rao, "An Adaptive Core Mapping Algorithm on NoC for Future Heterogeneous System-on-Chip," *Computers and Electrical Engineering*, 2021. **(Elsevier, SCI Indexed)**

   This paper proposes an Adaptive Core Mapping technique comprising mapping cores and spare core replacement on NoC. The ACM technique showed an increased performance for various sizes of NoC cores. The experiments conducted on 6 x 6 mesh NoCs revealed that the adaptive core mapping technique exhibited greater throughput, lesser delay, and energy consumption than other related algorithms. The current research also addressed the spare core placement issue, which replaces the faulty core with the available free core, therefore enhancing the reliability of the processor. An experimentation environment verified on the Kintex-7 FPGA KC705 board, which elucidates the faulty cores, recovery cores and spare cores in the network. The results implicate a dramatic decrease in area, power consumption, and an increase in throughput, which illustrates the efficiency of the proposed ACM algorithm.

   Authors Contribution: The author formulated an adaptive core mapping (ACM) algorithm, wrote the program, conducted experiments and wrote the manuscript. The implementation and experiments were conducted on Kintex-7 (KC705) FPGA board.

3. **Aruru Sai Kumar** and T.V.K. Hanumantha Rao, "Performance Assessment of Adaptive Core Mapping for NoC-based architectures," *International Journal of Embedded Systems*, 2021. **(Inderscience Publishers, ESCI and Scopus Indexed) (Accepted)**

In this paper, we implemented an Adaptive Core Mapping technique comprises of mapping cores, where the mapping region is obtained from NAD and PVR, whereas the communication energy between the cores obtained through WMD. The sequence for mapping the cores implemented through the lowest communication energy, i.e. in ascending order and this mapping technique applied to the PARSEC benchmark suite for the evaluation. The Proposed Adaptive Core Mapping technique evaluated using the GEM5 Simulator. The simulated outcome of this technique has outperformed both the latency and system performance compared with FASA, FARM, and NMAP techniques. The simulations and synthesis carried out through the Vivado Design Suite 2018.3 outperformed the metrics such as total latency, power consumption and the core mapping time compared to other algorithms.

Authors Contribution: The author formulated the system level core mapping algorithm. wrote the program, conducted experiments and wrote the manuscript. The implementation and experiments were conducted on GEM5 Simulator.

4. **Aruru Sai Kumar** and T.V.K. Hanumantha Rao, "Performance and Communication Energy constrained Embedded Benchmark for Fault Tolerant Core Mapping onto NoC architectures," *International Journal of Ad Hoc and Ubiquitous Computing*, 2021. **(Inderscience Publishers, SCI Indexed) (Accepted)**

   In this paper, an effective algorithm named as FTMAP (Fault-tolerant mapping algorithm), that exemplifies the core mapping on the basis of selected task graph, and replaces the faulty cores with the available free core termed as core replacement. This implementation focuses predominantly on the replacement of the faulty cores and assessing the communication as well as the execution time of the network by employing it on various multimedia benchmarks. The experimentation was carried out for NFT, 1FT, 2FT where the communication energy and execution time were outperformed.

   Authors Contribution: The author proposed a fault-tolerant core mapping (FTMAP) algorithm, developed solutions for the minimization of the communication energy and execution time using Noxim Simulator, conducted the case study, and wrote the manuscript.

5. **Aruru Sai Kumar** and T.V.K. Hanumantha Rao, "An Efficient Real-Time Embedded Application Mapping For NoC Based Multiprocessor System on Chip," *Wireless Personal Communications*, 2021. **(Springer, SCI Indexed) (Under Review)**

   In this paper an efficient mapping strategy was implemented on the real-time embedded applications named ERTEAM. In this algorithm, based on the minimum Node Average Distance (NAD) the mapping region is finalized, ensuring the overall mapping area reduced. The PE's mapped according to the minimum communication energy in the selected mapping region. The resultant outcome of the proposed mapping technique provides low latency, less simulation time, less communication energy and the overall throughput increased compared to BBPCR and SBMAP when applied to the mentioned embedded real-time applications.

   Authors Contribution: The author developed the idea, implemented the proposed techniques, conducted experiments using Noxim Simulator, and wrote the manuscript.

- International Conferences:

1. **Aruru Sai Kumar** and T.V.K. Hanumantha Rao, "An efficient low latency router architecture for mesh based NoC," *International Conference on Communications, Signal Processing and VLSI (**IC2SV-2019**), NIT Warangal, India, pp. 241-248, 2019.*

   In this paper, a Virtual router architecture is introduced which yields low latency resulting in improving the performance of a network. The proposed VIP based VC architecture for a 4x4 mesh NoC has experimented for 128-bit wide system targeting up to 250 MHz. The experimental outcome exhibits a low latency that requires 500 to 600 cycles on an average with respect to other router architecture. This outperforms 33% of low latency when compared to the Wormhole router architecture.

   Authors Contribution: The author contributed with the idea, evaluated experimentation methods, and wrote the manuscript.

2. **Aruru Sai Kumar** and T.V.K. Hanumantha Rao, "Efficient Core Mapping on Customization of NoC Platforms," *IEEE International Symposium on Smart Electronic Systems (**iSES-2019**), NIT Rourkela, India, pp.57-62, 2019.*

   This paper proposes an efficient core mapping technique and customization of NoC platforms are presented. The proposed mapping algorithm was assessed by applying it to different NoC benchmarks. The proposed algorithm incorporates standard topology selection and efficient core mapping. This algorithm was evaluated and compared with previous algorithms, and a great improvement in reliability, delay, area and power was observed where minimal objective function is achieved even for larger complex networks in 2D topologies.

   Authors Contribution: The author contributed with the problem formulation, conducted the experiments of various topologies using Noxim Simulator and wrote the manuscript.

3. **Aruru Sai Kumar**, T.V.K. Hanumantha Rao and B. Naresh Kumar Reddy, "Exact Formulas for Fault Aware Core Mapping on NoC Reliability," *IEEE 17th India Council International Conference (**INDICON-2020**), New Delhi, India, pp. 1-5, 2020.*

   In this paper, the exact formulas for the functional metrics in core mapping of the fault aware and failure probability in NoC were derived. Particularly to decrease the area of the mapping utilizing NAD & PVR, communication cost efficiently measured utilizing weighted communication energy. In the process of mapping the core, suppose if any of the faults that occurred at any of the cores effectively find out faulty core using failure probability. Finally, this metrics best in regard to reliability for core mapping of the fault aware in Network on Chip.

   Authors Contribution: The author discussed the problem formulation of the functional metrics utilized for the core mapping in NoC, and wrote the manuscript.

1.10 Thesis Overview

In our work, we investigated the core mapping and fault tolerance issues on NoC. We studied relevant works in application core mapping, spare core replacement and the fault-tolerance techniques used in Network-on-Chips (Chapter 2); Efficient Core Mapping using BMAP Algorithm (Chapter 3); Adaptive Core Mapping and its Hardware verification (Chapter 4); Fault-Tolerant Core Mapping (Chapter 5); Efficient Real-Time Embedded Application Mapping (Chapter 6); Conclusion and Future work (Chapter 7). To be more precise, the outline and contributions of this thesis are summarized below:

Chapter 2 : Literature Survey

This chapter provides the related works of core mapping techniques and the fault-tolerant mechanism for the real-time embedded NoC applications, spare core replacement and the FPGA-based NoC Models.

Chapter 3 : Efficient Core Mapping using BMAP Algorithm

This chapter presents the Efficient BMAP algorithm and explains about the core mapping, standard topology selection, the customization of NoC platforms and evaluating the proposed approach through SPLASH-2 Benchmarks. This technique improves the performance and cost metrics of the system.

Chapter 4 : Adaptive Core Mapping and its Hardware verification

This chapter presents the Adaptive core mapping (ACM) and spare core replacement, which explains the mapping of the application and replace the faulty core with spare core. The Hardware verification of the proposed ACM is synthesized through Kintex-7 FPGA KC705 board. The results implicate a dramatic decrease in area, power consumption, and an increase in throughput, which illustrates the efficiency of the proposed ACM algorithm. The experimental evaluation for the proposed core mapping carried through the PARSEC benchmark suite by using GEM5 Simulator.

Chapter 5 : Fault-Tolerant Core Mapping

This chapter presents the Fault-Tolerant Core Mapping technique (FTMAP), focuses predominantly on the replacement of the faulty cores and employing it on multimedia benchmarks. This outperforms the improvement in the communication energy and execution time.

Chapter 6 : Efficient Real-Time Embedded Application Mapping

This chapter presents the Efficient Real-Time Embedded Application Mapping entitled as ERTEAM, which provides an efficient mapping strategy implemented on the real-time embedded applications. The resultant outcomes of proposed algorithm provides low latency, less simulation time, increased throughput and less communication energy.

Chapter 7 : Conclusion and Future work

A consolidation of the results is presented in this chapter. The results indicate that the proposed methodologies provide a great improvement in the terms of system performance and the communication energy between the cores. Also presents the two possible extensions to the current research work.

Chapter 2

Literature Survey

Several publications have highlighted the need for solutions to pressing problems in various domains in the broad area of Network on Chips. This chapter introduces relevant works in the real-time embedded and fault-tolerant core mapping techniques used in Network on Chips, effects of communication on energy and performance trade-offs in Multi Processors, spare core replacement and FPGA based NoC models.

In order to optimise various factors,such as performance, communication energy consumption and chip area or a combination of them, several mapping techniques have been discussed (like Re-mapping, Re-configuration, fault-aware methodologies and spare core).

2.1 Mapping Algorithms

Chang et al., [44] introduced a framework called ETAHM which is used for solving task mapping and scheduling through ACO while producing the effective use of performance and energy consumption. This is used for allocating the tasks for a multi-processor NoC which is targeted to be composed of inner list scheduling and the outer evaluation loops. This provides less energy consumption and likely more power computation.

Wooyoung Jang et al., [45] proposed an A3MAP approach where the scheduling tasks are compatible mapped on a mesh NoC platform that targets general-purpose computing. This is for both homogeneous and heterogeneous cores regarding regular and irregular mesh. In this method, tasks are mapped to a respective core, which eliminates the problems by reducing the traffic amount. The quality of task mapping is measured by the total distortion of metric embedding. Through this formulation, A3MAP can map a task adaptively to any different sized tile on a custom network.

Ou He et al., [46] proposed the UNISM algorithm, which involves unified task scheduling and mapping on various NoC architectures. It is used for various NoC architectures, including regular mesh, irregular mesh, and custom networks, where both the task scheduling and core mapping operations are performed. A new labelled graph is implemented which provides the calculation of communication energy as well as latency. The results outperform the execution time and energy consumption. Area-overhead is considered as one of the limitation that the designer have to pay to gain reconfigurabilty feature.

Xinyu Wang et al., [47] introduced a method which is a novel mapping strategy on the basis of discrete PSO mechanism that minimizes the communication cost with less CPU time and evenly distributes communication over a chip. The stability analysis is presented which provides the robust behavior of this algorithm.

Ahmed A. Morgan et al ., [48] introduced a methodology (GAOPT) that is a GA based technique for optimizing the NoC architectures. This technique mainly deals with the performance metrics and cot metrics. Initially this algorithm selects the best standard topology followed by the application core mapping on these selected topology. This technique is applied on all the NoC benchmarks and the experimental outcomes outperforms the metrics such as area, power, delay and reliability. This algorithm requires extra computation time for finding out the optimum mapping when compared with NMAP and MMAP.

Pradeep Kumar Sharma et al., [49] implemented a mapping algorithm, i.e., a heuristic for low time complexity application of weighted graph containing acceptable bandwidth constraints, which minimizes the consumption of communication energy for a 2-D mesh. The approach mainly deals with lower time complexity to reduce the communication energy. The two techniques involved in this research for mapping an application are constructive mapping and reallocation. These techniques are responsible for providing a better outcome in reducing communication energy compared to other related techniques. This proposed algorithm limits only for 2-D Mesh topologies but does not consider other topologies.

Srinivasan Murali et al., presented a fast core mapping approach on an NoC platform with respect to bandwidth constraints termed as the NMAP algorithm [50]. This algorithm is for a single minimum path and split-traffic routing by taking bandwidth constraint into consideration through reducing the Delay. The traffic in NoC was split, having significant savings in the bandwidth and cost metric. This NMAP methodology can't be applied to dynamic mapping due to its high run time overhead and is applied to only mesh topology.

Amirali Habibi et al., introduced a mapping algorithm named MMAP [51], this is utilized in considering the Multicast along with the unicast communication flows. Initially the bandwidth values are characterized and these values along with the NoC configuration are considered as inputs for various heuristic mapping techniques. This technique improved performance and energy saving for synthetic as well as the CMP applications.

BNK Reddy et al., proposed an energy-efficient mapping algorithm named EMAP [52], which is used for mapping the cores in a network by considering the constraints of the communication. This improved in minimizing the total communication energy. The hardware verification is also performed for the proposed algorithm on Kintex-7 (KC705) FPGA board. The resultant outcome provides less hardware utilization as well as power consumption. This EMAP algorithm does not provide less communication energy for large size networks and results in more complexity.

Chen Wu et al.,[53] proposed a CoREP methodology which is used to optimize the reliability, energy consumption and the performance for the NoC architectures. Further an efficient mapping technique is introduced which provides an optimal mapping solution. The experimental outcomes outperform the reliability, throughput, energy efficiency and latency. If this proposed algorithm is extended to 3-D topologies, then there will be high probability for the link failures and results in more thermal dissipation.

Pradip Kumar Sahu et al., [53] proposed a mapping methodology which involves the PSO and ILP techniques. Initially to map the cores in an application in 2D and 3D mesh based NoC architectures the PSO technique is utilized. Later the obtained communication metric is compared with the exact methods of ILP. The obtained outcomes show an improvement in communication cost and less CPU time.

Upadhyay et al.,[55] proposed a two phase PSO for MoT topology on the basis of multiple application mapping. Initially by the combination of multiple applications the mapping is performed. Later the cores are reconfigured to the nearest routers. The experimental results show a significant improvement in communication cost. During the reconfiguration of cores onto the routers, there might be a high probability of faults in a network.

Bing Li et al.,[56] implemented a runtime mapping that is a thermal-aware algorithm that optimizes the overall performance for 3D NoC. The available core regions restored through the defragmentation algorithm introduced in this mapping. The experimental outcomes reveals that the running time is decreased to 48% and the communication cost to 44%.

LI Guangshun et al.,[57] implemented a mechanism for mapping the irregular IPs embedded on a regular 2D mesh topology for NoC architectures. The core principle is to break down each big IP into several smaller dummy IPs, each of which can move into a single tile, reducing energy consumption and avoiding congestion.

Weichen Liu et al.,[58] proposed a TopoMap algorithm for the SMART NoC architectures to improve performance. The topology of the architecture is selected dynamically based on the configuration by the thermal aware task mapping algorithm.

Guoyue Jiang et al.,[59] developed a mapping strategy based on the BB algorithm to provide both the core and the communication mapping. The application mapping contains packet switching, circuit switching as well as virtual circuit switching. This scheme reduces the overall latency and the energy of the hybrid NoC and optimizes the overall mapping. As the application mapping is done statically, the cores are allocated one after the other which increases the time complexity of the system.

Leibo Liu et al.,[60] proposed a BBPCR algorithm to find the optimal mapping for an application. Firstly, a PCM model that is highly accurate and flexible developed, containing both the energy and reliability parameters. Later, using this model, BBPCR is implemented for figuring out the best mapping solution for an application. Therefore, it significantly impacts the improvement of reliability, low energy consumption, and low latency. This technique does not resolve the failures occurred in a network and also does not provide a fault tolerant network.

Sarzamin Khan et al., [61] implemented the SBMAP mechanism by considering bandwidth constraints to minimize energy consumption and computational complexity. This mapping mechanism used the modular systematic searching technique. The system is divided into small possible modules and performs the mapping on it, resulting in high performance and less simulation time. SBMAP algorithm is applicable only for mesh and torus, it is not resilient to all the other topologies.

2.2 Spare core placement

Fatemeh Khalili et al., [62] developed a Fault Aware Spare Core Allocation entitled as FASA, is used to map the cores and replace the free core in place of the faulty core dynamically known as spare core replacement which improves reliability. Before the execution of the mapping technique, the preliminaries such as WMD, ANT and UNV are calculated. This results in the reduction of fault contamination area, enhancing performance compared to FARM.The communication energy also decreased through this technique. The limitation of this FASA is the Communication energy saving and performance improvement for low and high traffics is not significant.

Chen-Ling Chou et al., [63] introduced a Fault Aware Resource Management (FARM) at its system level, which is used to map the cores and dynamically embed the free cores at the faulty position. Initially, the failure probability is investigated based on that the spare core replacement is explored. Next the metrics such as network contention and the system fragmentation is measured by finding out the performance impact. Finally, the proposed mapping algorithm is evaluated. The technique provides high reliability, performance and low energy consumption. FARM tries to locate spare cores near processing cores, which have high probability to get failed.

P.V Bhanu et al., [64] proposed a technique to provide a fault-tolerant system and verified it through both simulations and FPGA validation. Firstly, the mapping of an application on the Torus topology is performed through the mathematical formulation of ILP and PSO to resolve the issues of fault tolerant mapping. A NoC router architecture named Virtual channel is proposed which is used for the FPGA implementation. The experimental evaluation were performed on the multimedia and synthetic application benchmarks. The scaling of the router faults have shown a significant impact.

Naresh Kumar Reddy Beechu et al., [65] proposed a technique named as FTCM in the system level entitled as Fault tolerance core mapping. The technique is composed of two parts, initially, the core are mapped onto the NoC network based on the ascending order of communication energy. The mapping region is finalized based on the calculation of NAD and PVR. After mapping the cores, if any core in a network fails, initially error detection and correction mechanism is performed to resolve the faulty cores. Even after applying it, if core still fails then the spare core replacement is performed. The experimental results outperforms the communication energy and system performance. The reliability and area are the two constraints for this method, as area increases and reliability decreases.

2.3 FPGA-based NoC Models

FPGAs are programmable devices that are used to implement digital circuits. They have traditionally been used to prototype circuit designs for final-stage verification. Modern FPGAs have the dynamic logic capacity to execute a whole digital onto one chip, due to advancements in processing technologies. FPGAs provide two benefits against software simulators as a performance test platform for NoCs. Firstly, a significant number of specialised function units may take use of the fine-grained parallelism for NoC simulation. Secondly, the profusion of wires accessible to interconnect the function units conveniently accommodates the high volume of communication which is costly to develop inside a coarse-grained thread architecture. FPGA-based NoC models could therefore be substantially faster compared to the software simulation.

Genko et al., [66] describes a configurable traffic generators and receptors-based emulation platform which drives a 6-switch NoC and therefore is 2600 times fast when compared to the SystemC simulation from the similar network. Since this platform allows you to configure traffic patterns and statistics counters, modifying the network configuration needs re-synthesising the emulator.

Yana E. Krasteva et al., [67] introduces DRNoC, a solution that avoids this requirement by taking advantage of Xilinx FPGA's partial reconfigurability. Every grid slot in the DRNoC host FPGA which can be dynamically modified to introduce a new element to mimic multiple networks.Partial reconfiguration, on the other hand, involves a unique design flow that imposes area overheads and this is also limited to a few devices. Therefore, the DART's configuration interface is generally based on the generic shift register, as well as it is adaptable to any FPGA.

NoCem [68] increases the density of the emulation when compared to the design of Genko et al., [66]. By eliminating the details of router pipeline as well as virtual channels, a 9-node mesh topology can be implemented on just a single FPGA. We choose a clean layout for every DART Router rather than compromising such crucial details such as: each have several input ports and a single output port, and in a simulated router, it replicates all-to-all switching by routing one input port for every DART cycle. The FPGA-based architecture and the modelled NoC architecture are not distinguished in any of the mentioned NoC assessment platforms. As an outcome, the emulator should always be updated as well as the FPGA synthesis place-route processes should be performed to examine a another distinct NoC. This technique is time consuming and requires large number of labor. Furthermore, it do not permit the trade-off of emulation performance for other key parameters like the area of the implementation.

Wolkotte et al., [69] virtualizes a router on FPGA that perform the performance or the area trade-off. Iterating numerous circumstances by the router type simulates a NoC with various routers. An off-chip ARM processor manages the emulation of the N-node network and saves N contexts for the router model. This method enables for a considerably more complex router model, while modifying the router settings typically needs hardware modifications. Furthermore, the ARM/FPGA communication link off-chip is a performance constraint.

Naresh Kumar Reddy Becchu et al., [70] proposed a FTNoC algorithm entitled as Fault tolerance Network On Chip, which mainly involves in mapping the cores and scheduling on the NoC architectures. The fault tolerant unit gathers all the information regarding the faults in a mapped network and therefore provides different solutions to overcome the different faults. The hardware verification is carried out through the Kintex-7 FPGA board to evaluate the FTNoC algorithm. The experimental outcomes outperform the area reduction, system performance and power consumption. This algorithm is limited for only few faults and does not specify the fault type and number of faults it can deal.

These studies evaluated the overall benefits and drawbacks of the defective core mapping in NoC. Earlier research has mostly emphasized on the fault model notation for the core, error detection, and spare core position identification. As a result of the mapping algorithm as well as spare core placement, the fault contamination area is reduced while energy is conserved and performance is improved. Proposed algorithm is verified on Kintex-7 (KC705) FPGA board. It also works with both random as well as distributed core graphs.

2.4 Fault-Tolerant Core Mapping

Lei Zhang et al., [71] introduced a task mapping mechanism that indulges a variation aware scheme that contains two stages. Initially, multiple mapping approaches were created through the genetic and static task mapping algorithm which outperforms the performance metrics. Secondly, during the run time execution, one among the various mapping approaches is selected as the efficient mapping approach which outperforms the communication cost. The proposed method takes into the consideration of the core symmetry as well as the topology modifications.

Zheng Wang et al.,[72] proposed a divide and conquered mechanism to build up a commercial fault tolerant architecture through fault tolerance concept which reduces the total edges, and it also imposes the rerun mechanism when faults occurred during the task execution. In this paper the generic graph is divided into number of simple subgraphs that could be processed separately. This approach is validated using a task mapping algorithm which is based on exhaustive search and also the failure injections were verified that shows the efficiency of this approach.

Colin Bonney et al.,[73] implemented APG which provides various fault tolerant mapping approaches for NoC application. This utilizes a multi-objective mapping mechanism known as EA and injected various faults to determine the overall performance impact. The remapping mechanism is followed in this implementation, i.e., if any faults occur during the task execution, it again reruns to obtain the new mapping structure. This exhibited a significant improvement in terms of performance metrics.

Navonil Chatterjee et al.,[74] proposed an effective mapping strategy of fault-tolerant, where the tasks that need to be executed in an application are induced dynamically, where the incoming tasks stagnant time were manipulated to indulge the fault-tolerant mechanism in which the faulty core reallocated with the free core. An appropriate fault tolerant technique is chosen by keeping in mind that periodic features of application tasks remain unknown at the outset. This proposal had a significant improvement in achieving the deadlines of the tasks and the network latency.

Fatemeh Khalilia et al.,[75] provided a heuristic mapping technique that is fault-tolerant on an NoC system architecture. After performing the core mapping using the heuristic technique, the spare core location is identified for every ACG (Application core graph) by taking into the consideration of transient faults as well as the permanent faults in a network. The execution of mapping technique and providing exact spare core placement, there was an improvement in terms of failure reduction, resource allocation during the failure of tasks, system performance and minimization of communication energy.

Suleyman Tosun et al., [76] introduced an algorithm termed FTTG, which mainly concentrated on the irregular topologies by providing a fault-tolerant mechanism. This technique provides at least 2 distinct paths, where every router in a topology access other router in those distinct paths. If there is any failure in a specific path, the specified topology picks up another route for mapping the application. It exhibited a significant outcome on the topologies respective to the fault-tolerant compared with the irregular topologies of non-fault tolerant in metrics such as performance, lower communication between cores, and area. FTTG can only generate one-fault tolerant topologies and cannot be applied toward generating multiple-fault tolerant network topologies.

Song Chen et al., [77] implemented the K-FTTG mapping methodology and the link failures of the physical NoC and the failures between the switches. In this method, where the ILP technique provides the required topology for the specific application, next to the sizes of the network port were minimized, and the issue of sharing the ports was resolved using the heuristic methodology. This technique reduces the overall energy consumption of the cores. K-FTTG does not resolve the failures, if any hardware faults occur in the switches or links. This method does not depend on the fault diagnosis methodology.

Naresh Kumar Reddy Becchu et al.,[78] proposed a EMAP algorithm entitled as energy-efficient fault-aware core mapping which mainly concentrated on mapping of the cores in an application through communication rate constraints and providing the spare core replacement. Initially the core mapping is performed using the functional metrics such as NAD, PVR and WCE. After the core mapping, the faults in the network are identified through the failure probability, where if faults occur, the error detection and correction mechanism is applied. Even if the faults persist, then spare core replacement is performed. This outperforms the reduction in area (A), power consumption for NoC architectures (P_{NoC}) and improves system performance.

2.5 Summary

This chapter provides the literature survey of the research work that involves the core mapping of the real-time embedded NoC applications, the spare core replacement when the faults occurs, the fault-tolerant core mapping technique and FPGA-based NoC Models.

Chapter 3

Efficient Core Mapping using BMAP Algorithm

In general, the objective of application core mapping is to determine the distribution of set of tasks across set of cores in order to enhance the design metrics. The application mapping contains two inputs namely; 1) application and 2) NoC platform. An application is comprised of a number of tasks that can run concurrently. Data or control interdependence among tasks are represented through inter-task communications.

The compute and communication resources required to complete the application are provided by a NoC. On-chip processing elements (PEs) for a heterogeneous NoC could include CPUs, embedded memory blocks, DSPs, video processors, FPGAs and other IP cores. Application mapping is becoming increasingly crucial in enhancing system efficiency and lowering energy usage as the number of integrated cores on various core architectures grows, as does the complexity of parallel processing computing.

This chapter provides an efficient core mapping technique that involves standard topology selection by providing a customized NoC platform. The proposed algorithm was applied to various application benchmarks [48] (such as AV, VOPD, MPEG-4, MWD) and SPLASH-2 Benchmarks as case studies compared with previous algorithms. The main contribution of this research paper is to improve the performance of the NoC architecture and had a great improvement in reliability, delay, area and power where minimal objective function is achieved even for larger complex networks in 2D topologies along with the evaluation of benchmark synthesis. Both the random and distributed core graphs are applicable in this research work.

3.1 BMAP Algorithm

In this research work, we focused on an efficient mapping algorithm named as BMAP, which exhibits the topology selection. Each and every NoC benchmark could be interpreted through the core graph, which is termed as an Application Core Graph (ACG). The core graphs for various benchmarks of NoC are illustrated in Figure 3.1. BMAP considers ACG and the NoC platform as its inputs. The application core graph comprises the number of core elements with its traffic characteristics, respectively. A directed graph that is derived from the ACG contains vertices and the communication rate between two individual vertices, i.e., expressed in MB/s. In our research work, various NoC topologies are presented in Figure 1.6. BMAP algorithm applied to all architectures and selected the most favorable NoC platform according to the functional metrics. Algorithm 1 clearly explains the mapping of ACG on the NoC platform.

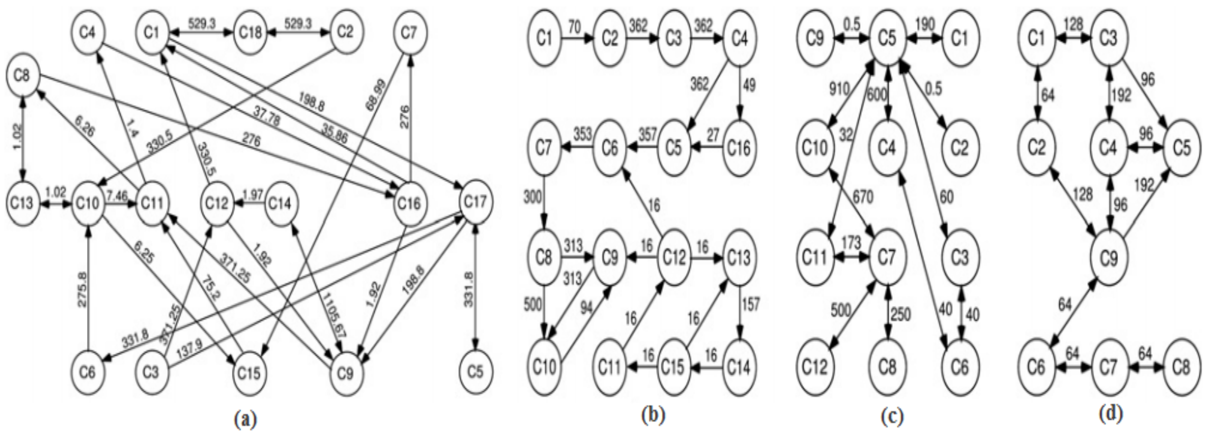

Figure 3.1: Core graphs for various NoC benchmarks [48] (a) AV, (b) VOPD, (c) MPEG-4, (d) MWD.

BMAP provides a propitious platform for NoC by considering reliability, delay, power, and area metrics. The proposed BMAP algorithm is applied to all the mentioned NoC architectures, and with the result obtained from the functional metrics, a propitious platform for NoC was determined. Initially, the functional metrics, i.e., Reliability, Delay, Power, and Area, are calculated independently. Later, to represent the objective function, all the acquired metric values (such as Area, Power, Delay) are multiplied and divided by the reliability, thereby providing an objective function from the obtained calculated values [48]. If the obtained objective function is minimal, then that core mapping is resulted as the best mapping.

Algorithm 1 BMAP Algorithm

```
Input: Let C be the order set (descending order of communication rate) of
         vertices;
       Let IP be the set of cores;
       Let TG be the given NoC platform;
Output: Core mapping (CM);

foreach $c \in \{ C \}$ do
    Find the core $ip \in IP$ such that $ip$ has the maximum number of neighboring
      free cores in the TG;
    if multiple cores are available then
        Find the maximum neighboring free core $ip \in IP$ by considering all
          neighboring cores (free, busy, and failed cores) of the TG;
    else
        Map the vertex $c$ onto $ip$ in the TG;
    end
    Update $C$ by eliminating $c$;
    if min > objective function F then
        min←objective function F ;
        BestCM← CM;
    end
end
```

3.2 Demonstration of BMAP through a Case study

The proposed BMAP algorithm extended as the chromosomes for all architectures mentioned in Figure 3.1, the approach considered these architectures to be integer vectors as illustrated in Figure 3.2. The input for the proposed algorithm comprises of vertices and cores in the NoC platform. The vertices are selected based on the maximum communication energy with their neighbors and the cores are selected that are free and interact with neighboring free cores.

In Figure 3.2, the first vertex is selected as C_9 as it contains the highest communication energy when compared to its neighbors and was mapped to the fifth position (second row, second column) because it communicated with more number of free cores in the NoC platform. From Figure 3.2(a), we can see that C_9 is communicated with four neighboring cores such as C_2, C_4, C_5, C_6 and are placed in an order according to their maximum communication energy. The other vertices are also selected in the same procedure according to their next maximum communication with the neighbors which is in the order (C_5, C_4, C_2, C_6, C_3, C_1, C_7, C_8) and these were mapped to the next maximum communicating free cores. If the communicating free

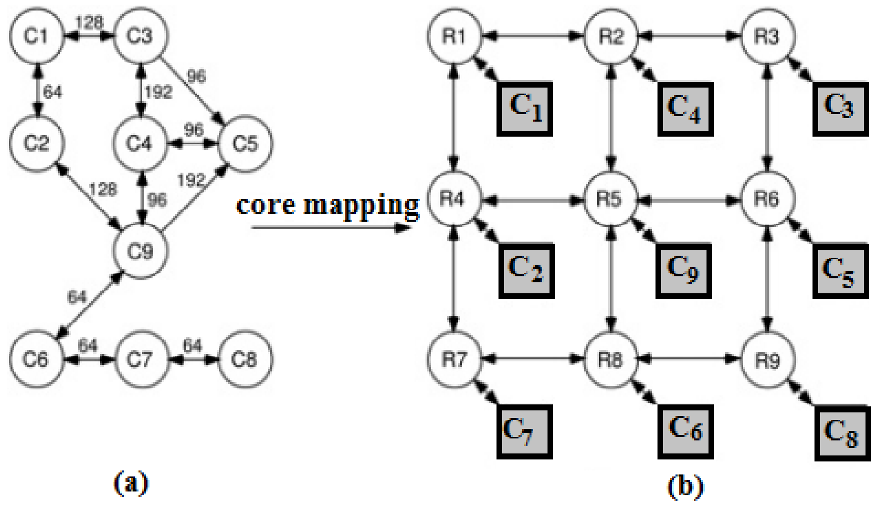

FIGURE 3.2: Example of core mapping. (a) MWD benchmark. (b) 3 x 3 2D-Mesh architecture.

cores are multiple then we select the maximum free core by considering the status of all the neighboring free cores i.e. free, busy or failed cores and map accordingly.

The proposed algorithm maps the remaining vertices of neighboring cores by implementing same iterative process. Eventually, core mapping of minimal objective function i.e. ‘F’was accomplished on NoC platform termed as the best core mapping.

3.3 Experimental Results

The proposed BMAP algorithm is applied on the four benchmarks presented in Figure 3.1. The benchmarks mentioned above are intended for four real applications with distinct core numbers: Audio-Video (AV) application consists of 18 cores, 16 cores for the VOPD application, 12 cores for the MPEG-4 decoder and 9 cores for the MWD application. The efficient core mapping pattern was determined through the C++ program, and the simulation results performed on a Noxim simulator [79].

3.3.1 Customization of NoC Platforms

The primary goal of the proposed mapping algorithm is to determine the foremost suitable norm topology of NoC to any specific application. The eight topologies discussed in Figure 1.6, were evaluated according to their metrics to assess the best of them. The functional metrics that were used in the evaluation are Reliability, Delay, Power and Area. Table 3.1 provides the result of four metrics for the eight topologies included in the research work. This is acheived by taking the average of all the respective topology values to its corresponding functional metric in terms of percentage. The topology present in the first row of every metric is considered as the best topology for that corresponding benchmark. As mentioned above, to obtain minimal objective function the reliability metric should be high when compared to other metrics which is clearly depicted in Table 3.1. On basis of the calculation, the obtained best topology along with the other topologies are described individually in percentage terms.

Let's examine each and every metric and provide the best topology for a specific application as illustrated below.

3.3.1.1 Reliability

The term reliability is calculated when the data packet gets transmitted from source to the destination even in the midst of noise. The faster the data gets transmitted in such network, the more reliable it becomes. Table 3.1 that represents the reliability factor for all the topologies to the mentioned NoC applications. For the reliability metric, the 'T' topology demonstrates the most desirable performance on reliability.

3.3.1.2 Delay

The term delay is defined as the traffic time in transferring the data packets from the source to the destination is evaluated in terms of delay. Mostly delay is categorized into two types: a) arbitrary delay and b) propagation delays, where the propagation delays are caused due to the links and de-serializing/serializing with the routers via NI. The lesser the delay, faster the data transmission. Table 3.1 that represents the delay factor for all the topologies to the mentioned NoC applications. The 'BFT22' topology demonstrates the most minimal delay that resulted in Delay metric.

3.3.1.3 Power

The term power is distinguished as the longer the time required for communicating between the cores requires more power to be consumed. In NoC, mostly the links between the cores and the routers consumes more power. Table 3.1 that represents the power factor for all the topologies to the mentioned NoC applications. The

Table 3.1: Comparison among various topologies for all metrics for the four application benchmarks.

Metric	Benchmark							
	AV		VOPD		MPEG4		MWD	
	Topology	%	Topology	%	Topology	%	Topology	%
Reliability	CLOS	100	CLOS	100	T	100	T	100
	BFT23	85	T	90	M	80	CLOS	90
	BFT22	65	P	80	P	65	M	80
	T	60	M	78	CLOS	50	BFT22	80
	M	55	BFT22	70	BFT22	35	BFT23	75
	P	40	R	60	R	30	R	70
	BT	35	BFT23	55	BFT23	25	BT	65
	R	30	BT	50	BT	20	P	65
Delay	BFT23	100	BFT22	100	M	100	M	100
	BFT22	105	R	105	BFT22	103	BFT22	102.6
	BT	109	M	109	P	106	T	105
	M	114	P	115	R	107	CLOS	107
	R	118	T	119	T	108	BFT23	108
	P	120	BFT23	123	BFT23	110	R	109
	T	121	BT	127	CLOS	112	BT	110
	CLOS	126	CLOS	129	BT	120	P	112
Area	BT	100	BT	100	BT	100	BT	100
	R	108	R	109	R	111	R	118
	BFT22	110	BFT22	116	BFT22	119	BFT22	127
	M	114	M	129	M	132	M	138
	P	123	P	138	P	143	P	149
	T	146	T	149	T	151	T	158
	BFT23	158	BFT23	165	BFT23	169	BFT23	172
	CLOS	176	CLOS	186	CLOS	189	CLOS	192
Power	BFT22	100	R	100	R	100	R	100
	BT	101	BFT22	105	BFT22	108	BFT22	115
	R	102	M	108	M	112	M	119
	BFT23	105	P	116	P	123	P	129
	M	119	BT	125	BT	134	BT	142
	P	147	BFT23	149	BFT23	159	BFT23	162
	T	152	T	174	T	194	T	215
	CLOS	208	CLOS	214	CLOS	245	CLOS	234

Note: The % is calculated for each metric with regards to that of the best topology.

minimal consumption of the power was achieved by 'R' topology which was resulted in power metric.

3.3.1.4 Area

As the number of components are increasing exponentially for On Chip Networks, the area (A) is getting increased. Mostly, the routers and the links in the network consumes more area same as the power. The efficient usage of the components leads to lesser area consumption. Table 3.1 that represents the area (A_{NoC}) for all the topologies to the mentioned NoC applications. The minimal consumption of the area was achieved by 'BT' topology which was resulted in area metric. Eventually, compared to the other topologies, 'CLOS' topology accomplished a most desirable performance and cost respectively.

3.3.2 Outcome of SPLASH-2 Benchmark Synthesis

The SPLASH-2 suite consists of a mixture of complete applications and computational kernels [80]. The SPLASH-2 benchmarks are FFT , radix, radiosity, ocean (contiguous), cholesky, lu(contiguous), water-nsquared, raytrace, ocean, water-spatial and barnes. In this research for the SPLASH-2 Benchmarks, we considered 11 workloads; out of these, 7 workloads comes under high performance computing namely Ocean-c, Cholesky, Lu, Water-N, Ocean, Water-S, and Barnes. Radiosity and Ray-Trace workloads comes under Graphics, FFT workload is of signal processing and Radix is general applications. Noxim Simulator [79] is used for performing these simulation outcomes, where each workload is evaluated against the other compared algorithms. Therefore, proposed BMAP algorithm and different algorithms listed in Table 3.2 illustrating about Characteristics and optimization of the various algorithms as follows:

Table 3.2: Characteristics of various algorithms and optimization algorithm

	Algorithm	Optimization
NMAP [50]	Initialization & Iteration	Shortest path
MMAP [51]	bandwidth requirements & priority of communication flows	Energy saving
EMAP [52]	Mapping and Swapping	Low communication Energy
Proposed Algorithm	Efficient Mapping	Improvement in Performance

The proposed BMAP algorithm is compared with NMAP [50], MMAP[51] and EMAP [52] over the NMAP baseline. During the comparison with other algorithms regarding synthesis of SPLASH-2 benchmarks, below are the following metrics obtained while executing individual workloads as well as its average workloads using BMAP algorithm which outperformed when compared to other mapping algorithms.

3.3.2.1 Speed-up Execution Time

It is defined as an improvement in the speed while executing various tasks in an application. The average improvement observed when compared with other algorithms are explained in Table 3.3, where the Speed-up Execution Time in BMAP has improved up to 40% when compared to NMAP, 30% when compared to MMAP and 20% when compared with EMAP. The simulated results of Speed-up Execution Time is shown in Figure 3.3.

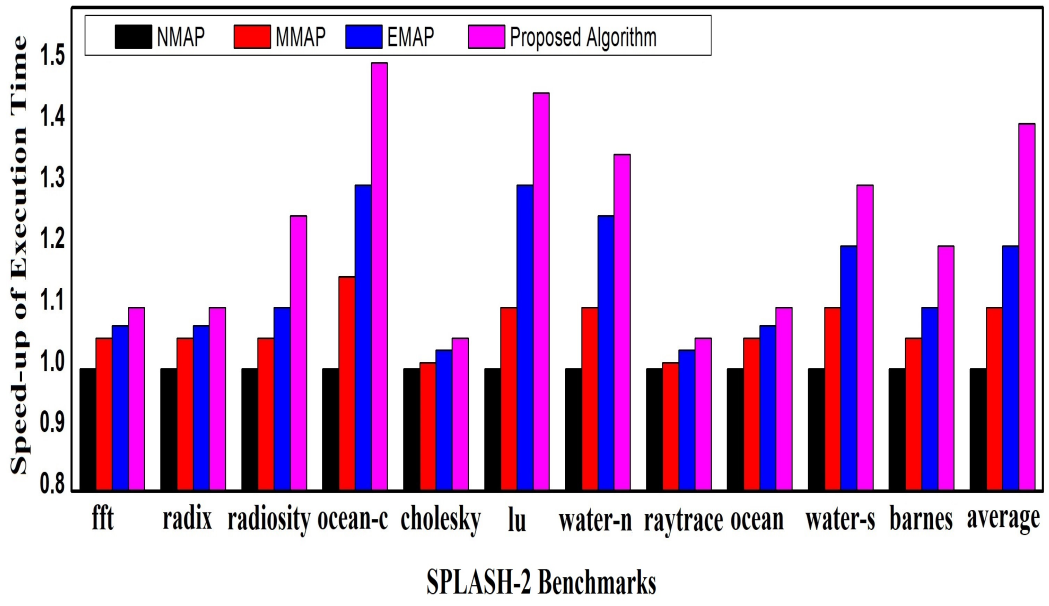

FIGURE 3.3: Comparison Results of Speed-up Execution Time using SPLASH-2 benchmarks.

3.3.2.2 Latency

It is defined as the time required for the packets to get transferred from one to other. The average reduction of time required observed when compared with other algorithms are explained in Table 3.3, where the Latency was reduced by 42% when compared to NMAP, 34% when compared to MMAP and 28% when compared with EMAP. The simulated results of Latency are shown in Figure 3.4.

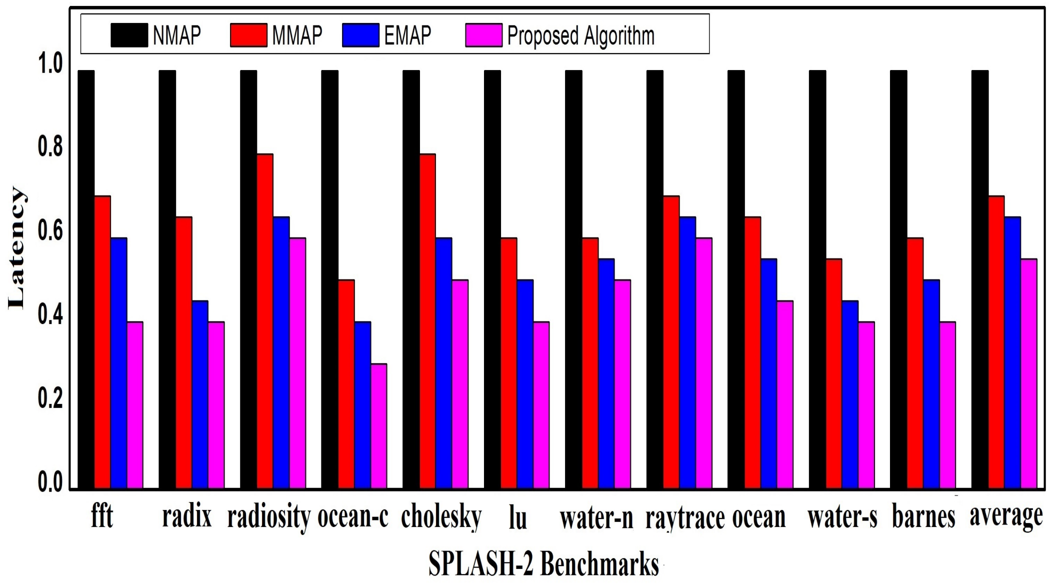

FIGURE 3.4: Comparison Results of Latency using SPLASH-2 benchmarks.

3.3.2.3 Energy efficiency

It is defined as utilizing minimal amount of energy during a task execution, which results in energy saving. The average improvement observed when compared with other algorithms are explained in Table 3.3, where the Energy efficiency was improved by 36% when compared to NMAP, 30% when compared to MMAP and 26% when compared with EMAP. The simulated results of Energy efficiency are shown in Figure 3.5.

3.3.2.4 Power consumption

It is defined as in any On-chip networks, the power utilized between the cores and routers. Therefore, reducing the power consumption provides better outcomes. The average reduction observed when compared with other algorithms are explained in Table 3.3, where the Power consumption was reduced by 32.6% when compared to NMAP, 28.2% when compared to MMAP and 26.4% when compared with EMAP. The simulated results of Power consumption are shown in Figure 3.6.

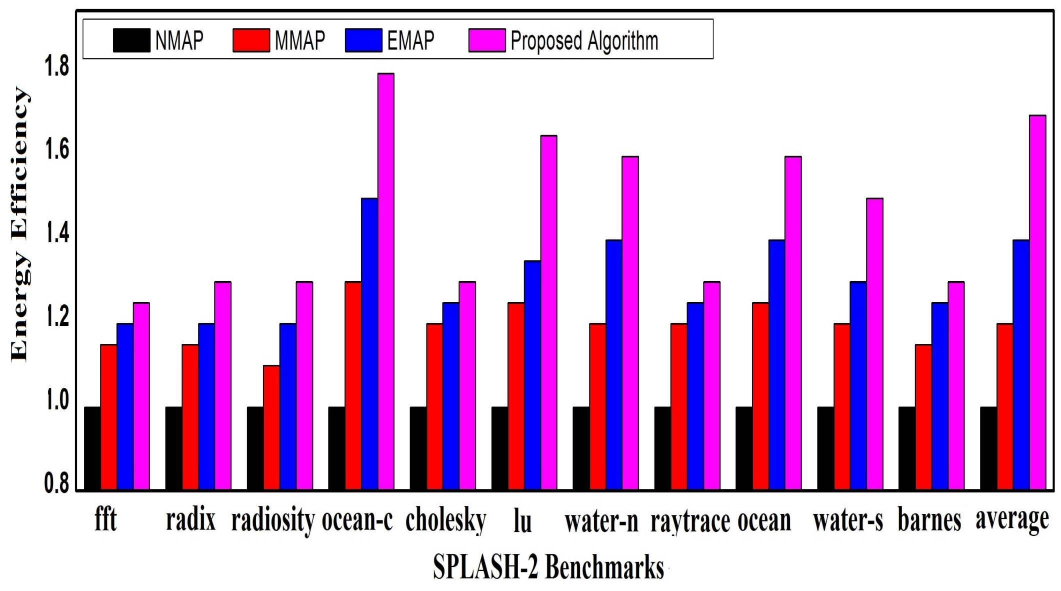

FIGURE 3.5: Comparison Results of Energy efficiency using SPLASH-2 benchmarks.

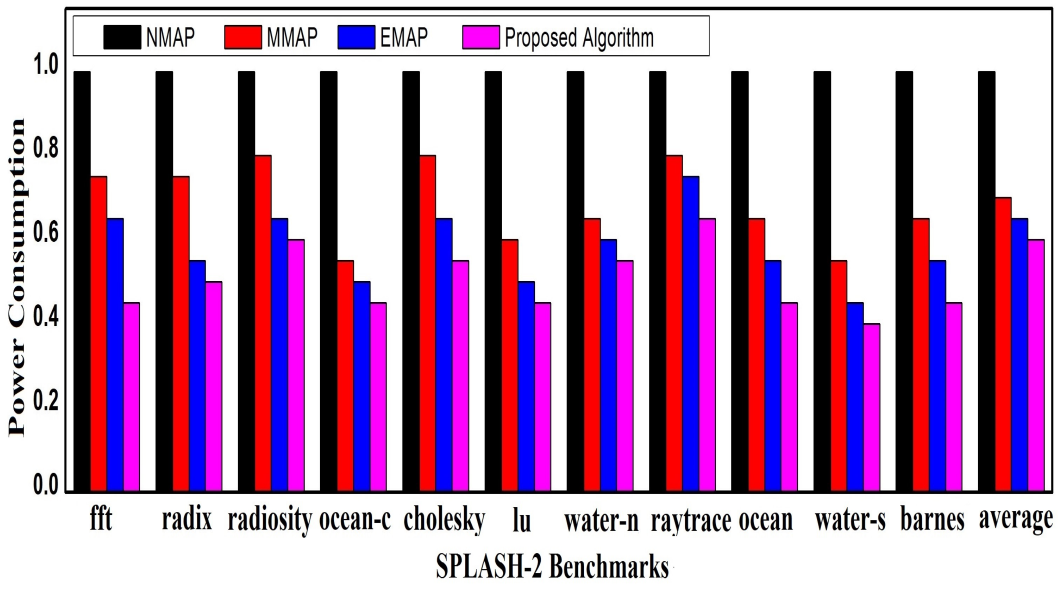

FIGURE 3.6: Comparison Results of Power consumption using SPLASH-2 benchmarks.

Table 3.3 represents the evaluation of Speed-up Execution Time, Latency, Energy efficiency, and Power consumption of the proposed BMAP algorithm against NMAP, MMAP, and EMAP over the NMAP baseline for SPLASH-2 benchmarks.

Table 3.3: Comparative analysis of BMAP against NMAP, MMAP and EMAP over the NMAP baseline for SPLASH-2 benchmarks.

	BMAP against NMAP[50]	BMAP against MMAP[51]	BMAP against EMAP[52]
Speed-up Execution Time	40%	30%	20%
Latency	42%	34%	28%
Energy efficiency	36%	30%	26%
Power Consumption	32.6%	28.2%	26.4%

3.4 Summary

In this chapter, an efficient core mapping technique and customization of NoC platforms are presented. The proposed algorithm incorporates standard topology selection and efficient core mapping. This algorithm was evaluated and compared with previous algorithms, and a great improvement in reliability, delay, area and power was observed where minimal objective function is achieved even for larger complex networks in 2D topologies. During the SPLASH-2 benchmark synthesis, the simulated outcomes exhibit that BMAP effectively improves performance by reducing the execution time with low Latency against NMAP, MMAP, and EMAP algorithms.

Chapter 4

Adaptive Core Mapping and its Hardware verification

Reliable core mapping on NoC depends on the functional metrics. We propose that an application must be mapped to the NoC platform. Each application has numerous processes running in the background. By using static analysis, the data traded among cores can be probably be regulated. The data indicate how frequently a packet is transmitted between the source and destination in the network. However, our problem is to map the cores onto the mesh-based NoC platform by using functional metrics and optimizing formulas.

This chapter provides an implementation of an Adaptive core mapping (ACM) technique, with the inputs as a core graph (CG) and NoC topology. The mapping as well as scheduling of the core graph onto the NoC architecture is handled by the core graph unit. Through mapping the application core graph to the free and non-faulty processor cores, this unit keeps the core graph information of the application model updated. In the mapping algorithm, vertices are mapped to tiles on NoC based on preliminaries like Weighted Communication Energy (WCE), Nodes Average Distance (NAD) and Placing unmapped Vertices Region (PVR). It tries to decrease conflicts between running applications on NoC by determining a mapping region for cores. If there is any probability of failures occurred in the system, the ACM technique migrates the tasks of the failed cores to free core through spare core replacement, which improves the system reliability. This technique exhibited greater throughput, lesser delay, and energy consumption when compared to other related algorithms. The latency and system performance is outperformed when applied on PARSEC benchmarks and also the hardware verification of the proposed ACM is carried out on FPGA Kintex-7 KC705 board.

4.1 Functional Metrics used for Core Mapping

The three main functional metrics used for the core mapping are: a) Weighted Communication Energy (WCE), b) Node Average Distance (NAD) and finally, c) Placing unmapped Vertices Region (PVR).

Each of these metrics have a significant place in performing the mapping of cores in NoC. NAD and PVR plays an important role for selecting the region of mapping in a NoC platform and whereas, WCE is utilized for the communication cost. So, based on the calculation of WCE for each mapped region, the minimum WCE is considered as the best and the final core mapping. Let us examine each of the functional metrics mentioned:

4.1.1 Weighted Communication Energy (WCE)

The WCE is used for calculating the communication energy of any two nodes or vertices present in a NoC application [78].

The Energy is directly proportion to the Distance. So let us indicate it as

$$E \propto D \tag{4.1}$$

Hence, the Communication Energy is also directly proportion to the distance between the nodes. It is indicated as E(e_{ij}).

The distance is calculated from one point to the other point while following the square grid-path. The Distance between any two nodes or vertices is given as (V_i, V_j)
where V_i parameters are (x_1,y_1), V_j parameters are (x_2,y_2)
So now the distance between the two nodes or vertices are calculated as

$$Distance = |(x_1 - x_2)| + |(y_1 - y_2)| \tag{4.2}$$

As per Eq. (4.1), the communication energy is directly proportional to the distance calculated between two nodes is given as follows:

$$E(e_{ij}) \propto |(x_1 - x_2)| + |(y_1 - y_2)| \tag{4.3}$$

$$E(e_{ij}) = W(e_{ij}) \times \{|(x_1 - x_2)| + |(y_1 - y_2)|\} \tag{4.4}$$

Where W(e$_{ij}$) is given as a constant, which is also termed as the Communication weight between any two vertices. Therefore, this is the calculation of the Weighted Communication Energy(WCE) metric which is used for calculation of the communication energy between the nodes or vertices.

4.1.2 Node Average Distance (NAD)

The metric NAD is defined as the Longest shortest path for any of the two selected nodes of a network. It states that path between the two selected nodes in a network is considered to be minimal [78].

The average distance for two selected nodes (NAD) in NoC, having X*Y size is computed as illustrated below Eq. (4.5)

NAD = $\frac{1}{3}$ [(X - 1/X) + (Y - 1/Y)]

$$NAD = \frac{X+Y}{3} \,.\, (1 - \frac{1}{XY}) \tag{4.5}$$

Therefore, this is the calculation of the Node Average Distance metric which is used for calculating the smallest region in the network.

4.1.3 Placing unmapped Vertices Region (PVR)

Placing unmapped vertices in a region is dependent upon the situation and was derived in Eq. (4.6)[78]. The cores that are unmapped are indicated as 'C_{um}'and the unmapped Vertices were indicated as 'V_{um}'

$$C_{um} > V_{um} \tag{4.6}$$

$C_{um} = V_{um}$ + S (S ∈ [1, n])

The total number of the free cores present in ACG should be more when compared to the total number of vertices, which results in successfully mapping the ACG for every NoC size. If the above mentioned constraint is not satisfied then it results to a unsuccessful core mapping. To map the cores in ACG successfully for any NoC sizes, the prerequisite is the total number of free cores need to be more when compared with the total number of the vertices present in ACG. Therefore, this is the calculation of the Placing unmapped Vertices Region metric which is used for calculating the mapping region for a network [78].

4.2 Proposed Adaptive Core Mapping

There should be a mechanism to prevent faulty cores from being used in the adaptive core mapping performance evaluation on NoC. The proposed ACM comprises two processes; one is mapping the cores based on application, whereas the other is spare core placement. After mapping the core, if a fault occurs at any core, follow the fault diagnosis method, which determines the location of the damaged resource and correct the error using error detection and correction mechanism. If the faults occur even after applying the fault diagnosis method, perform task migration using spare core placement.

4.2.1 Core Mapping

To perform an application core mapping, let us consider the NoC platform of mesh topology and a Core graph (CG) as inputs. The number of free cores present in the NoC platform should be more than the number of vertices in CG. The mapping region is obtained from the calculation of NAD and PVR functional metrics. Once mapping region is finalized, the cores are mapped on to the selected NoC topology in the sequence of the lowest communication energy. Algorithm 2 clearly explains the core mapping technique of proposed ACM.

4.2.2 Spare Core Placement

After mapping the cores, if fault occurs at any core, follow fault diagnosis method which determines the location of the damaged resource and error is being corrected through error detection and correction mechanism (it means transient/ intermittent faults). In case, if the faults occur even after applying the fault diagnosis method

Algorithm 2 Adaptive Core Mapping

```
Input:  Let V be the order set (ascending order of communication with vertices)
        Let T be the set of tiles
        Let M be the core mapping region ;
Output: Core mapping (ACM);

foreach $m \in \{ M \}$ do
    Calculate NAD and PVR region ;
end
foreach $v \in \{ V \}$ do
    if Selected tile $= v_0$ (First tile of lowest communication with neighbors) then
        Map the tile at the corner side of NAD region;
    else
        Select the neighboring position of the previous mapped core ;
        if multiple neighboring tiles are present then
            Select the tile $t \in T$ i.e. corner of NAD region which is free, non-busy,
              and non-failed cores of M ;
        else
            Map the vertex $v$ onto $t$ in M at the corner position of NAD and
              change the status of tile as BUSY ;
        end
    end
    Update $V$ by eliminating $v$;
    Calculate the WCE for the mapped region
    if minimum $WCE > WCE$ then
        minimum WCE ← WCE
        BestCM← CM
    end
end
```

(it means permanent faults), perform task migration using spare core placement as illustrated in Figure 4.1.

Spare core placement technique has five steps.

1. Initial communication energy is assumed to be zero.
2. Spare core is located close to the faulty core and calculate communication energy using Eq. (4.4).
3. Consequential communication energy is added to the initial communication energy.
4. This process is repeated for all cores in mapping region.
5. At last position of spare core is selected. Which has a lowest communication energy surrounded by all the obtainable free cores.

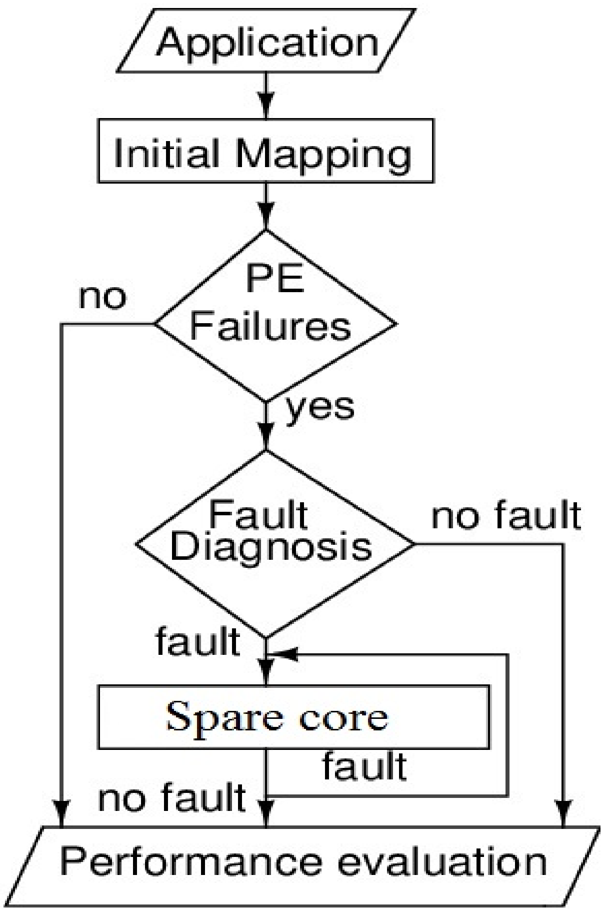

FIGURE 4.1: Block diagram for the proposed methodology

4.3 Demonstration of ACM through a Case study

Figure 4.2 elucidates the process of mapping the spare core placement of ACM. A simple application core graph illustrated in Figure 4.2(a); this core graph is represented onto a 6 x 6 NoC platform, as illustrated in Figure 4.2(b). By considering the above inputs, let us elaborate the mapping technique i.e. illustrated in Algorithm 2. This mapping is applied on the mesh topology of specific NoC size. By the values obtained from the calculation of Eq. (4.5), construct the NAD region in the respective mapping region. Here, the mapping region depicted in blue, preferably 3 x 3 region sized. In the 3 x 3 region, seven free unmapped cores, one fault core, and one busy core are present. This 3 x 3 region perfectly satisfies the unmapped cores for efficient mapping. The vertices which are having least communication energy are been set in an ascending order (V_2, V_0, V_3, V_4, V_1) respectively.

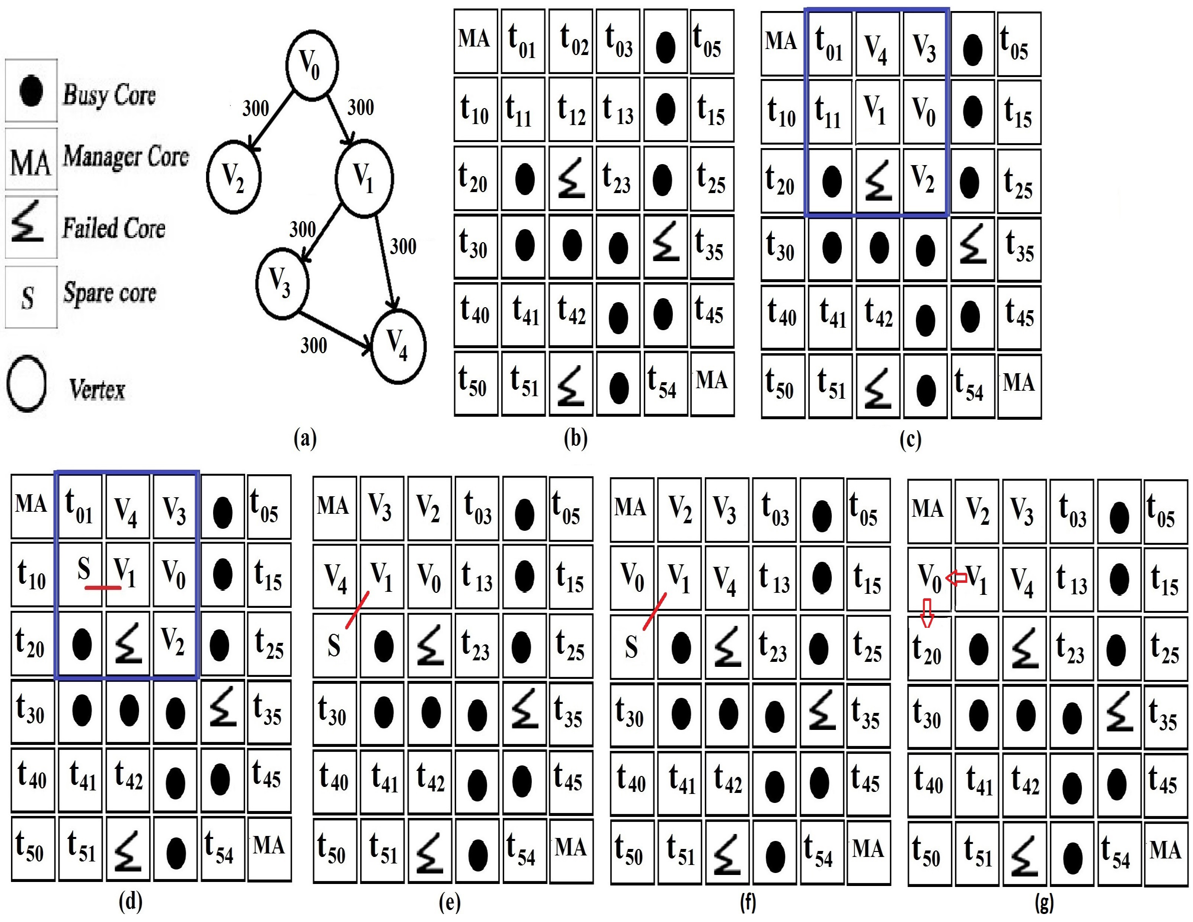

FIGURE 4.2: Spare core placement: (a) An example CG, (b) 6 x 6 mesh NoC, (c) ACM Mapping algorithm, (d) ACM Spare core placement, (e) FASA algorithm, (f) FARM algorithm, and (g) NMAP algorithm.

Once the NAD region in obtained, map the vertex V_2 (i.e. having the least communication energy, so considered as the first vertex) at the corner of the NAD region as per the proposed algorithm which improves the performance comparatively illustrated in Figure 4.2(c). After mapping the first vertex to the corner of NAD, the other remaining vertices are mapped at the neighbor position of the previously mapped vertex or tile which is free, non-busy and non-failed. Once the tile is mapped, it is marked as BUSY core. If there are multiple free neighbor positions, then consider the corner position of the NAD and map it accordingly. The same iterative process continuous until all the vertices are mapped. Finally, the total communication energy of this core mapping is minimum which resulting as the best core mapping in terms of performance characteristics of NoC.

Here, after post completion of mapping the cores, every free core that exists acts as spare core, but a spare core can be assigned only based on the faulty cores. N number of spare cores are required when there are N faulty cores in NoC platform. After performing the core mapping as illustrated in Algorithm 2, there are two more tiles or vertices (t_{01}, t_{11}) as free cores. These cores can be utilized as spare cores if any of the failure cores occur. The result of the ACM spare placement shown in Figure 4.2(d). Figure 4.2(e–g) represents FASA [62], FARM [63], and NMAP [50] algorithms.

4.4 Experimental Analysis of various routing types

Noxim simulator is used to calculate the performance of proposed Adaptive Core Mapping (ACM). It is a cyclical-accurate NoC simulator written in System C and determines the delay, throughput and energy consumption. Permutation and combination of different inputs in 6 x 6 NoC platform is tabulated and for each set, a systematic experiment is realized on Noxim Simulator [79]. For this evaluation, 1-flit packet and 1-flit buffer was considered. The channel is 128 bit and for every 128 cycles, it is cleared.

4.4.1 Delay

Delay is one of the finest performance metrics in systems. Generally, a Delay is termed as the total amount of time required for a message or a packet to be transmitted from the source to its respective destination. It is a deterministic function of the transmission rate. The delays achieved in different routing algorithms namely XY, WEST-FIRST, NORTH-LAST and ODD-EVEN routing were shown in Figure 4.3. Significant improvement was observed in the proposed ACM when compared to FASA [62], FARM[63] and NMAP [50].

4.4.2 Throughput

Throughput is considered as the main metric in system performance. It is the maximum amount of data conveyed per unit time. The experimental data indicates a relation of throughput with different parameters of the system as mentioned in the Eq. (4.7).

$$Throughput = \frac{(number\ of\ completed\ messages\ X\ packet\ length)}{(number\ of\ cores\ X\ total\ times)} \tag{4.7}$$

As it can be distinguished from Eq. (4.7), the throughput is directly proportional to the total number messages of completed and length of the message or packet while,

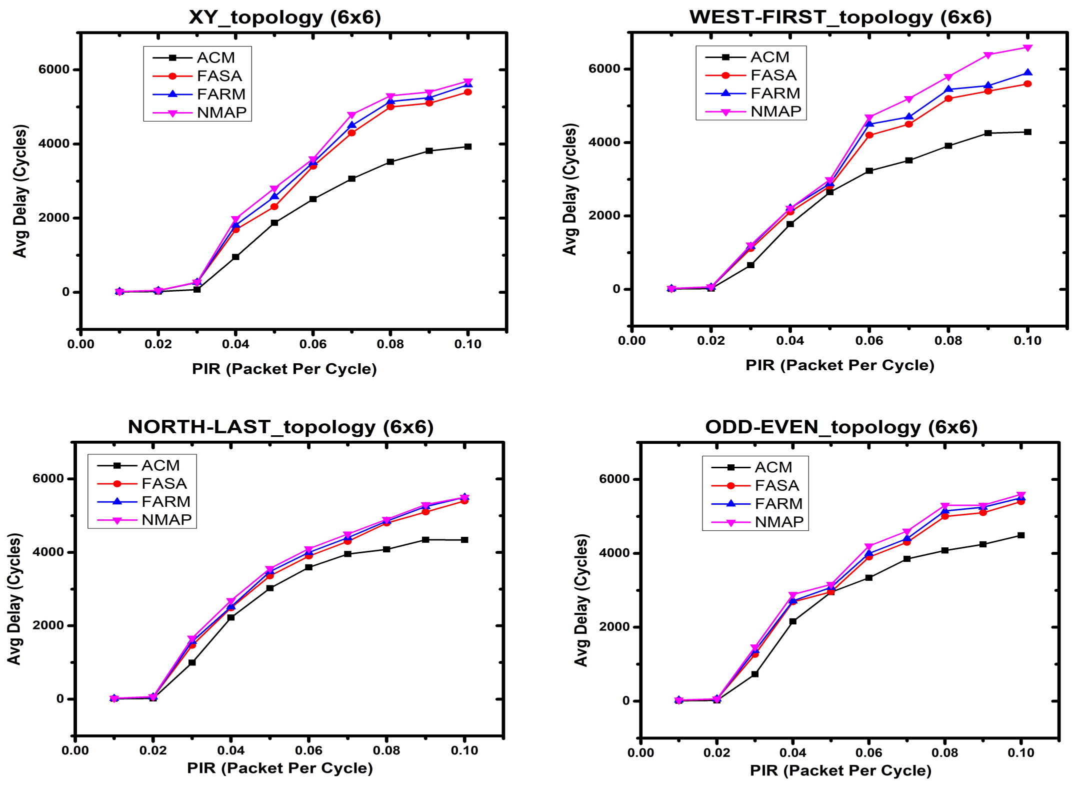

FIGURE 4.3: PIR vs Average Delay considering different routing types in 6 x 6 mesh network.

its relation is in inverse with the total number of clock cycles and total number of cores present in the application.

The throughput achieved in different routing algorithms such as XY, WEST- FIRST, NORTH-LAST and ODD-EVEN routing is shown in the Figure 4.4. ACM spare core placement exhibited better improvement when compared to FASA [62], FARM [63] and NMAP [50].

4.4.3 Energy Consumption

The total energy is determined as the energy consumed between cores and routers. The total energy $\varepsilon(t)$ is calculated as:

$$\varepsilon(t) = \sum_{i=1}^{N} \alpha(t) E_i + \sum_{j=1}^{N_f} \alpha(t) E_j \tag{4.8}$$

where N determines number of cores, $\alpha(t)$ is termed as number of bits that reaches till time t. E is energy consumed while transferring bits, which is dependent on NoC

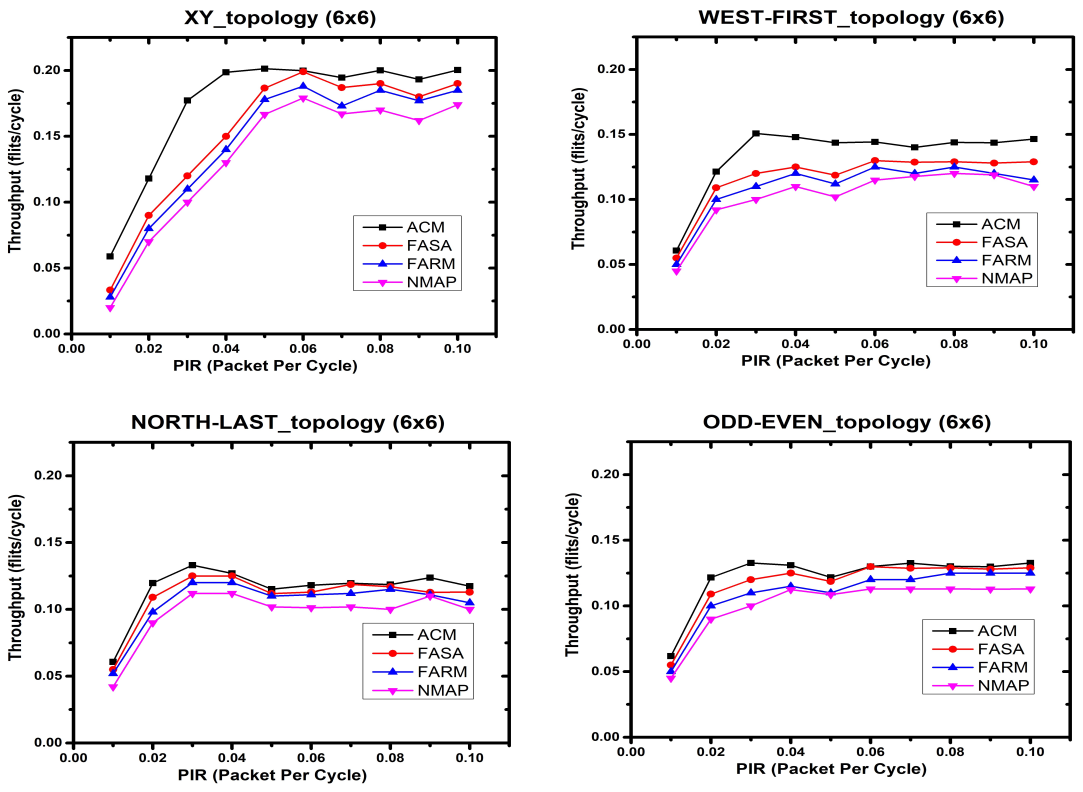

FIGURE 4.4: PIR vs Throughput considering different routing types in 6 x 6 mesh network.

platform. N_f is number of failed cores.

In XY, WEST-FIRST, NORTH-LAST and ODD-EVEN routing shown in the Figure 4.5, energy consumption decreases under ACM spare core placement when compared to FASA [62], FARM [63] and NMAP [50]. This significance could be clarified through large number of packets produced when there is an increase in the injection rate. ACM has less delay, energy consumption and high throughput values when compared to other related algorithms. The above mentioned figures clearly explain the overall routing patterns in which the proposed ACM technique shows greater throughput, lesser delay and reduced energy consumption than FASA [62], FARM [63] and NMAP [50].

The efficiency of the ACM technique is evaluated through the experimental result that demonstrates the performance on both real and simulated applications is discussed independently. The following are the Real applications used in our technique: MWD, Video object plane decoder(VOPD) and MPEG4 decoder. The number of vertices are taken in the range of 4 to 20 in our application core graph. In this

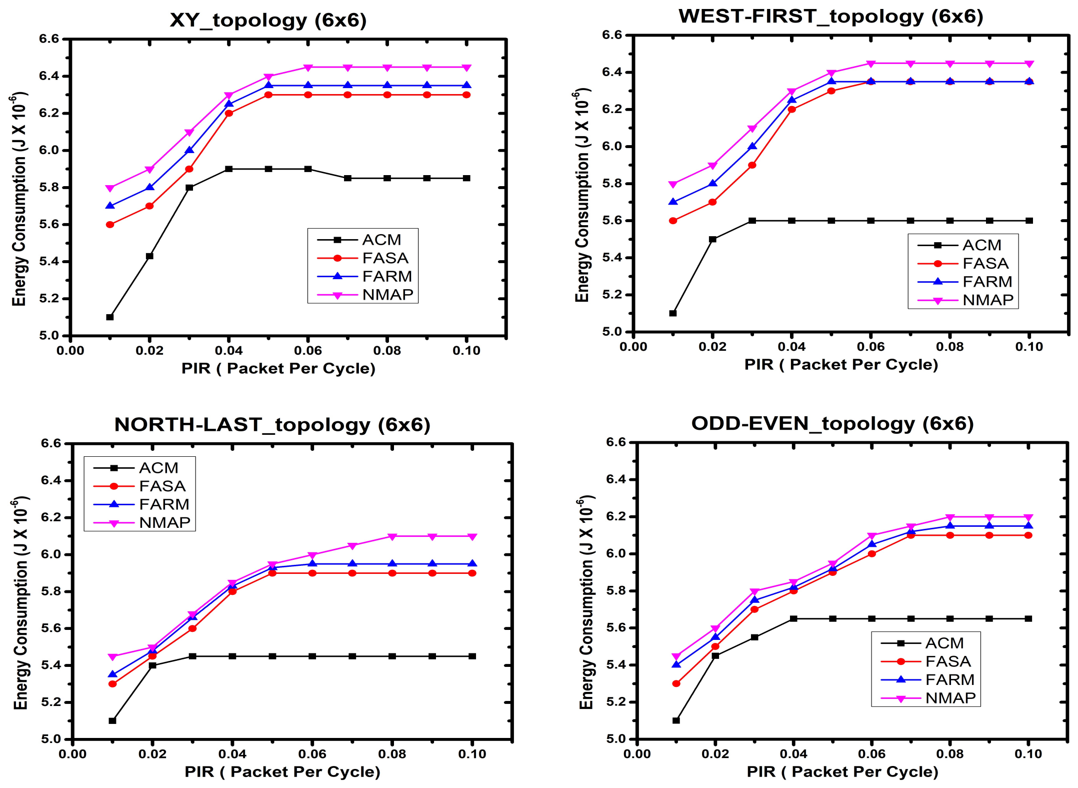

FIGURE 4.5: PIR vs Energy Consumption considering different routing types in 6 x 6 mesh network.

paper, a 6 x 6 sized network is considered and delay, throughput and energy consumption were calculated under two conditions (1 failed core and 2 failed cores) for ACM against FASA, FARM and NMAP as demonstrated in Table 4.1, Table 4.2 and Table 4.3. Significant improvement was observed in ACM compared to FASA [62], FARM [63] and NMAP [50].

TABLE 4.1: Performance Improvement of ACM and FASA

		ACM against FASA					
		1 Failed Core			2 Failed Cores		
Test case (No of Cores)	Arch (M x N)	Delay (%)	Throughput (%)	Energy Consumption (%)	Delay (%)	Throughput (%)	Energy Consumption (%)
MPEG4 Decoder (12)	6 x 6	12.9	13.2	13.5	14.7	15.1	14.9
MWD (12)	6 x 6	12.6	12.9	13	13.9	14.4	14.6
VOPD (16)	6 x 6	13.6	14.5	13.9	15.2	15.9	15.5

The effect of the core failure in the performance of ACM in the above example (Refer Figure 4.2) is illustrated in Figure 4.6. The throughput is shown in 'Y' axis and packet injection rate is shown in 'X' axis. Here when the faulty core prospects increase, then the throughput decrease. This is due to the reason that when cores are

Table 4.2: Performance Improvement of ACM and FARM

		ACM against FARM					
		1 Failed Core			2 Failed Cores		
Test case (No of Cores)	Arch (M x N)	Delay (%)	Throughput (%)	Energy Consumption (%)	Delay (%)	Throughput (%)	Energy Consumption (%)
MPEG4 Decoder (12)	6 x 6	14.1	14.6	14.9	16	16.4	16.2
MWD (12)	6 x 6	13.4	13.8	13.6	14.8	15.2	14.9
VOPD (16)	6 x 6	14.7	15.2	14.9	16.4	17.2	16.8

Table 4.3: Performance Improvement of ACM and NMAP

		ACM against NMAP					
		1 Failed Core			2 Failed Cores		
Test case (No of Cores)	Arch (M x N)	Delay (%)	Throughput (%)	Energy Consumption (%)	Delay (%)	Throughput (%)	Energy Consumption (%)
MPEG4 Decoder (12)	6 x 6	15.4	16	16.4	18.2	18.6	18.4
MWD (12)	6 x 6	14.2	15.2	14.8	15.9	16.6	16.2
VOPD (16)	6 x 6	15.8	16.4	16.6	18.2	19.4	18.6

more, spare cores are highly exploited compared to before, as in the above discussed manner, spare core can be placed near to the application core graph which results in increased throughput and ultimately increased performance.

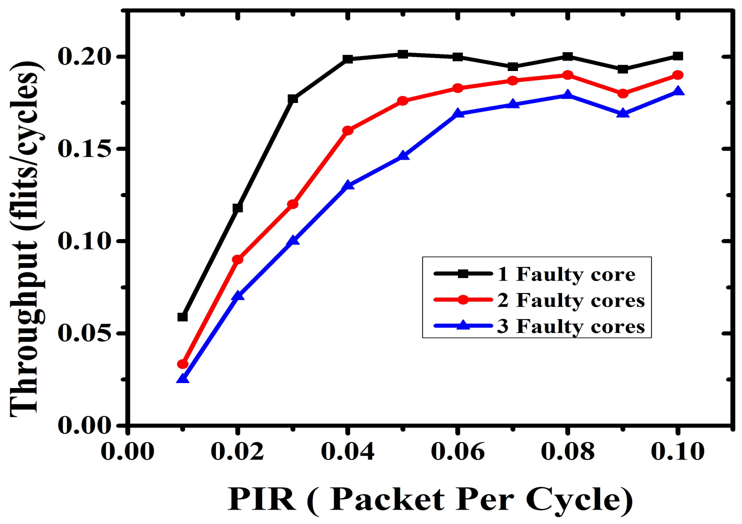

Figure 4.6: Throughput for different faulty cores in 6 x 6 mesh network

4.5 Benchmark Evaluation

For benchmark applications, the experimental evaluation was constructed with respect to PARSEC benchmark suit (Blackscholes, Facesim, Vips and Swaptions) [81].

A reasonable NoC platform using GEM5 [82] simulator was fabricated to assess the proposed approach. It had 36 processing elements, on which the application mapping was modelled. For mapping and scheduling the application on NoC platform, different injection rate was used. Same ACG and platform were used for other related algorithms.

On account of core mapping algorithm, the communication latency between the entire system performance and mapping cores has a drastic change. Figure 4.7, clearly shown on-chip communication latency for PARSEC benchmark applications. The proposed ACM approach expressively improves the communication efficiency when compared to FASA [62], FARM [63] and NMAP [50].

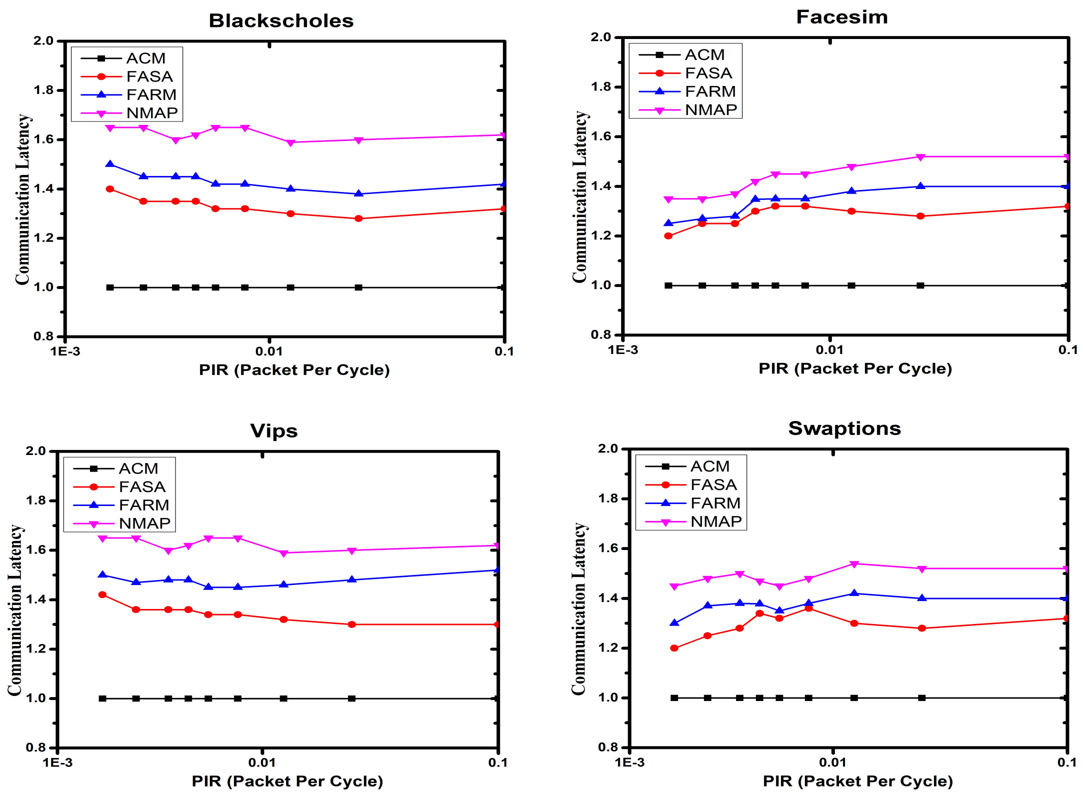

FIGURE 4.7: Comparison of communication latency with different packet injection rates considering PARSEC benchmark applications.

With respect to the application performance as shown in Figure 4.8, proposed approach in ACM can expressively improve system performance compared to FASA [62], FARM [63] and NMAP [50].

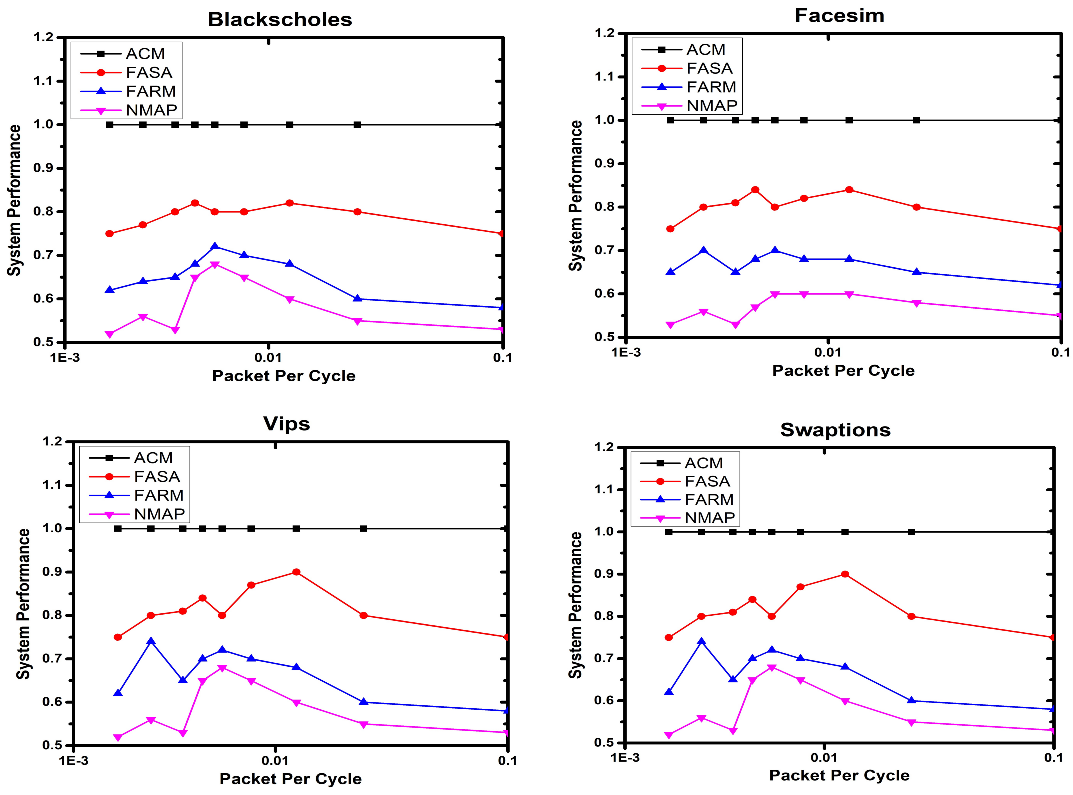

Figure 4.8: Comparison of system performance with different packet injection rates considering PARSEC benchmark applications.

4.6 Hardware Verification

Hardware verification is also as important as the implementation. Everything that is designed must be tested and verified because otherwise there is no certainty that the design behaves as specified. Verification can take the time of the whole design process. This section describes the verification of Adaptive core mapping on NoC and comparing with previous techniques.

The proposed Adaptive Core Mapping (ACM) is coded in Verilog HDL, synthesized and simulated in Vivado Design Suite 2018.3 [83]. As shown in Figure 4.9, FPGA board Kintex 7 (KC705) board is the target device which is used for synthesis [84]. An FPGA switch acts as input whereas the LEDs act as output. In the current research, vertices are considered as switches and NoC platform cores are considered as LEDs. The faults present in the mapping core are represented through the switch on board (where '1' denotes as no fault and '0' denotes a fault). The proposed ACG comprises of 5 vertices only, so this requires five input switches for Kintex 7 FPGA board, in order to represent those 5 vertices described above (V_4, V_3, V_2, V_1, and

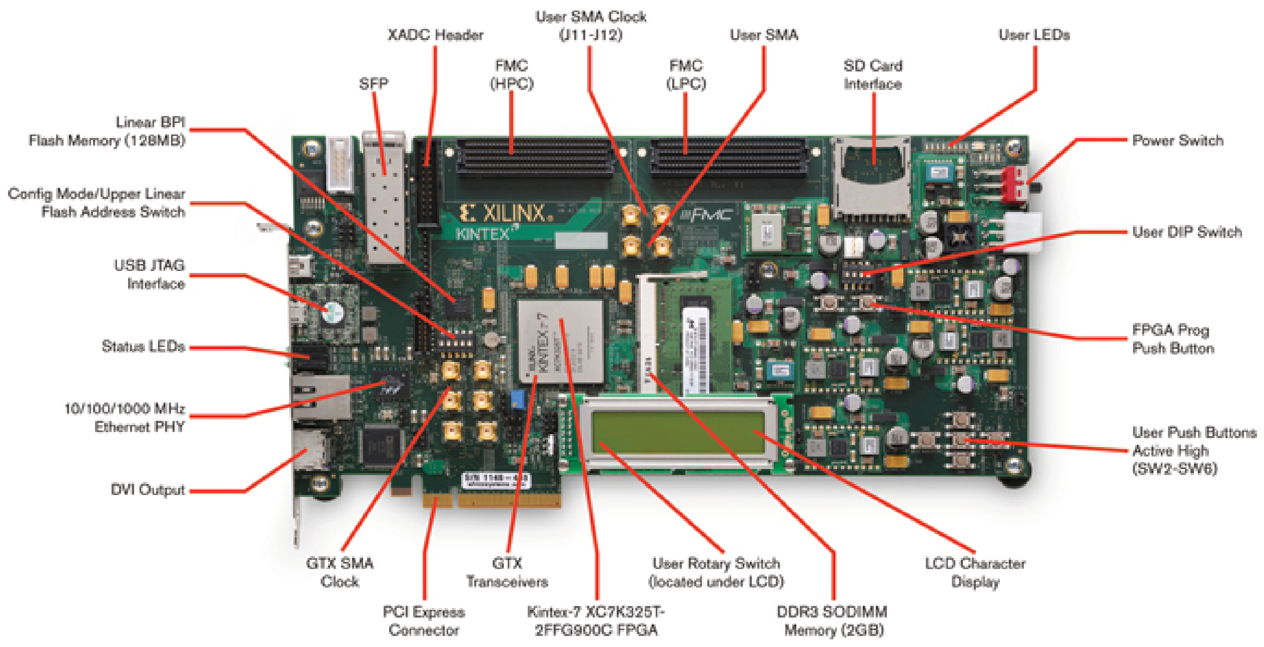

FIGURE 4.9: FPGA based Verification Platform.

V_0) and also the output LEDs as (t_{01}, t_{11}, t_{02}, t_{03}, t_{23}, t_{12} and t_{13}) indicating NoC core. As per the output LEDs listed above, the first 2 (i.e., t_{01}, t_{11}) are considered spare cores. If any core gets failed even after recovery, the faulty core gets replaced with the nearest spare core. A red LED glow which is designated as '1' is used indicating the successful transfer of data (i.e., there is no fault in the presented core) and the LED off is designated as '0' which indicates that the faults occurred at the core. If any faults occur, recovery cores get executed for verifying and to correct the code using error correction code (Hamming code). Even if the fault happens after the detection and correction mechanism (using Hamming code), then the failed core tasks gets migrated to the available spare core. Spare core distinguishes efficiently for each incoming application in terms of number and their positions.

Experimental data is tabulated in Table 4.4. When analyzing the acquired results, if the NoC platform contains no faults, the data gets transferred as per the ACG shown in row 1. In row 2, V_0 core fails due to which the recovery core verifies and corrects the core using error correction code (hamming code). V_2 core fails even after applying the hamming code which results in migration of V_2 core tasks to the spare core position (t_{11}) which is clearly shown in row 3. In row 4, faults can recover alike row 2. Using spare cores, the row 5 faults can be avoided.

TABLE 4.4: Experimental tabulation of data after FPGA based verification for different cores.

Input Switches (V_4, V_3, V_2, V_1, V_0)	Output LED (t_{01}, t_{11}, t_{02}, t_{03}, t_{23}, t_{12}, t_{13})	Faulty Cores	Recovery Cores	Spare Cores
11111	0011111	None	None	None
11110	0011111	t_{13}	t_{13}	None
11011	0111011	t_{23}	None	t_{11}
10101	0011111	t_{03} & t_{12}	t_{03} & t_{12}	None
01011	1101011	t_{02} & t_{23}	None	t_{01} & t_{11}

4.6.1 Comparative Results

In order to precisely calculate latency and mapping time of proposed ACM and FASA [62], FARM [63] & NMAP [50] Vivado Design Suite 2018.3 is utilized.

Table 4.5 indicates the total latency, longest path latency represented through clock cycles and the mapping time represented in terms of milliseconds (ms). These experimental results reveal that the ACM algorithm is better than FASA [62], FARM [63] & NMAP [50].

TABLE 4.5: Experimental results on the 6 x 6 Mesh NoC.

	ACM	FASA	FARM	NMAP
Total latency (clock cycles)	74016	79987	82356	85879
Longest path latency (clock cycles)	24167	31011	34269	37301
Mapping time (ms)	99986	100143	102789	108149

In order to accurately estimate the metrics of the ACM such as area, performance and the power and also the other related algorithms, Vivado Design Suite 2018.3 was utilized. The evaluated area, performance and power of the proposed ACM were compared with the other algorithms such as FASA [62], FARM [63] and NMAP [50], as tabulated in Table 4.6. One of the main advantage of the proposed ACM over the other related algorithms is the area efficiency, area is computed in the form of number of logic blocks. Regarding power analysis, ACM utilizes XY routing algorithm, where packets get distributed along the minimum hop paths in a network. The performance that is obtained is evaluated as throughput of a network. The results that are obtained for the area and power consumption of ACM algorithm decreased by an average of 7.2% and 9.75%, 11.36% and 10.54%, and 12.4% and 11.11% when compared to FASA [62], FARM [63], and NMAP [50]. The improvement of performance by an average of 12.5%, 14.7%, and 18% when ACM algorithm is compared to FASA [62], FARM [63], and NMAP [50]. It is visible that ACM algorithm is an efficient core mapping outperformed with related algorithms. Finally, the system results reveal that the simulation and hardware verification is absolutely reliable.

Table 4.6: Evaluation of area, power consumption and throughput of ACM against FASA,FARM and NMAP.

	ACM against FASA	ACM against FARM	ACM against NMAP
Area	7.2%	11.36%	12.4%
Power Consumption (W)	9.75%	10.54%	11.11%
Throughput (Gbps)	12.5%	14.7%	18%

By analyzing the test results, if faults are not present in the platform, data is directly transmitted as per the ACG. Otherwise the fault diagnosis mechanism is applied which tries to recover faulty cores through error detection and correction mechanism. Even if the faulty core exists the spare core replacement occurs. The test data results reveal that hardware verification and simulation results are completely consistent.

4.7 Summary

In this chapter, we implemented an Adaptive Core Mapping (ACM) technique comprising the mapping cores and spare core replacement on NoC. The ACM technique showed an increased performance for various sizes of NoC cores. The experiments conducted on 6 x 6 mesh NoCs revealed that the adaptive core mapping technique exhibited greater throughput, lesser delay, and energy consumption than other related algorithms. The current research also addressed the spare core placement issue, which replaces the faulty core with the available free core, therefore enhancing the reliability of the processor.

Apart from this evaluation, the sequence for core mapping is initiated from the lowest communication energy and applied to the PARSEC benchmark using GEM5 Simulator. The obtained results show a significant improvement in latency and system performance. An experimentation environment verified on the Kintex-7 FPGA KC705 board, which elucidates the faulty cores, recovery cores and spare cores in the network. The results implicate a dramatic decrease in area, power consumption, and an increase in throughput, which illustrates the efficiency of the proposed ACM algorithm. Any core mapping techniques of the network can utilize the ACM methodology to enhance the system performance.

Chapter 5

Fault-Tolerant Core Mapping

Expanding the number of transistors as well as wires packed onto a single chip allows for the development of more complex electronic devices as semiconductor fabrication technology advances. Scaling effects, such as the reduced transistor dimensions, the decrease of critical charge, the increase of clock frequency and the increase of the power density, intensify the frailty of electronic devices to environmental variations, which also has a negative impact on long-term chip lifetime and consequently accelerates the occurrence of faults of the circuit. Fault-tolerance becomes an essential design objective for critical digital systems, especially those in highly specialized fields. One of the key factor for the implementation of the best application mapping strategy is, it should be fault tolerant, which finally results in the improvement of overall performance of a system.

This chapter presents the Fault-tolerant core mapping (FTMAP) which is responsible for tracking cores in the mapped NoC platform and storing the information collected from the mapped NoC platform in the memory unit. This technique effectively maps the vertices on the network and finalize the mapping based on the minimum communication energy.The obtained output is considered as the best core mapping by assessing the communication as well as the execution time of the network by employing it on various multimedia benchmarks. Based on the failure probability, the faults in a network can be identified priory during the mapping process. Based on the obtained failure probabilities, the core that has the highest probability will be moved to the nearest free core respectively. The experimentation was carried out for NFT, 1FT, 2FT where the communication energy and execution time were outperformed.

5.1 Faults classification

A defect is the failure of a component in one layer of a computer network, which could be separated into numerous layers. It could be a computer defect within applications or software platforms, or even a hardware defect caused by radiation or wear-out defects in the silicon chip, causing the chip to malfunction. Faults relate to the failure of integrated circuits in this thesis. While performing the mapping process, many faults could emerge onto cores. Faults within integrated circuits are categorised into two groups based on its duration time: 1) Transient faults and 2) Permanent faults; intermittent faults are sometimes included as a third category. If the faults are occurred repeatedly it is termed as transient and if the faults last for longer period it is termed as permanent

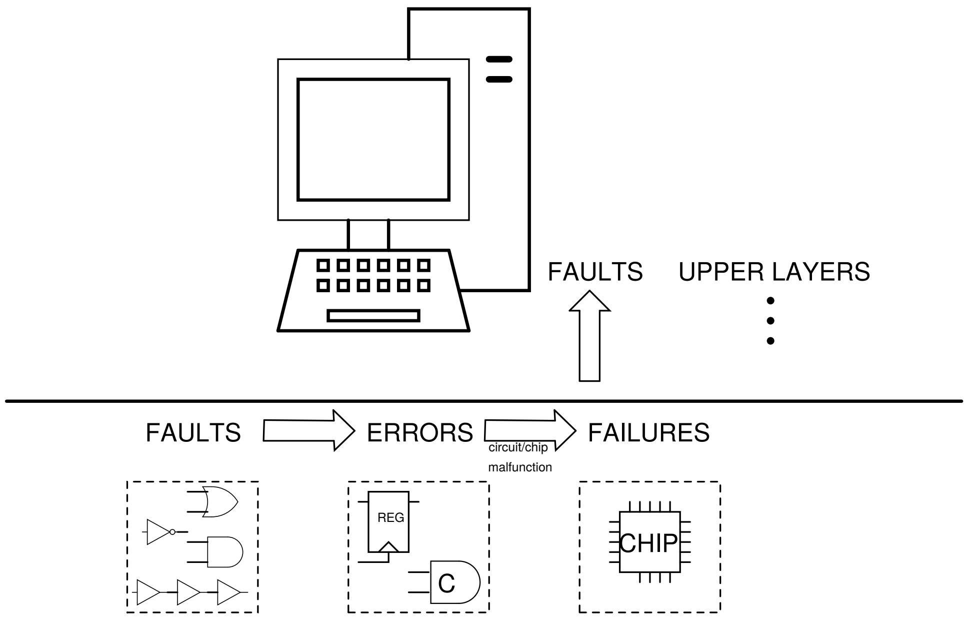

FIGURE 5.1: Relationship between faults, error and failure.

The relation between three commonly utilized terms in fault-tolerance literature: fault, error, and failure is depicted in Figure 5.1. Errors are manifestations of faults. Errors in integrated circuits are described as failures collected by memory components (like C-elements within asynchronous circuits), which further divided into two types: transient and permanent. Faults were required for mistakes to occur, although not all faults result in errors because many are hidden during its propagation. If mistakes are not addressed, they might produce circuit or chip output malfunctions, or even a circuit failure, which can lead to problems in higher layers of a computer network (like the operating system or application software level), where detection and correction can be much more difficult and expensive. To handle with various fault conditions, a multi-layer, fault-tolerant architecture shielding a computer network from the bottom circuit to the high software level is necessary. The thesis focuses on fault tolerance at the semiconductor level to minimize chip defects as well as avoid chip failures.

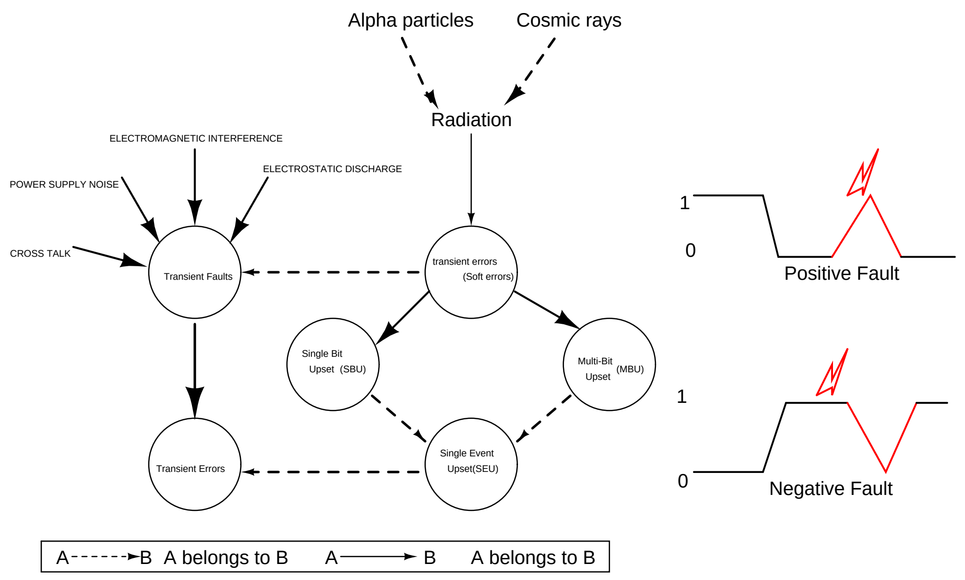

Figure 5.2: Sources of transient faults.

5.1.1 Transient faults

The susceptibility of electronic devices to environmental fluctuations increases as NoC size dimensions, clock frequency, integrated circuit density, and also critical charge decrease, greatly increasing the probability of transient failures. As Figure 5.2 describes, transient faults could be caused by a variety of factors. A bit-flip, often described as glitch, is a common transitory defect symptom that is either favorable or unfavorable. When memory components collect them, they normally last a short time creating soft errors which are non-permanent as well as non-recurring.

5.1.2 Permanent faults

Permanent faults (also known as hard faults) are divided into two categories based on its occurrence time: 1) manufacturing faults and 2) operational hard defects.

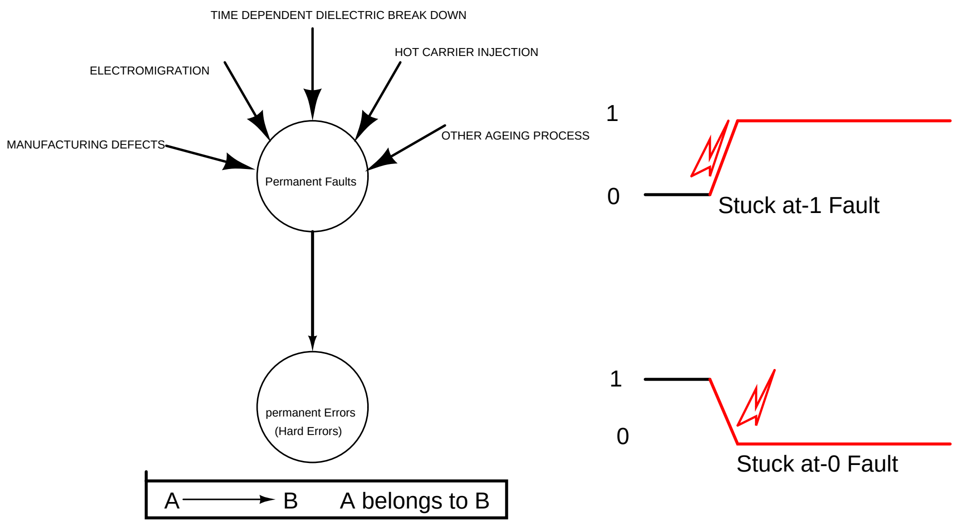

FIGURE 5.3: Sources of permanent faults.

Production faults can occur throughout the chip manufacturing, resulting in a reduction of chip yield. To improve chip yield, defect-tolerance methods are applied. This type of permanent problem is becoming less common as chip manufacturing techniques evolve which is clearly depicted in Figure 5.3.

5.1.3 Intermittent faults

The intermittent faults are defined as the a faults that appear, then vanishes, again reappears, and vanishes back. These are the most aggravating component defects. This type of faults is exemplified by a fragile connection.

5.2 Probability of Faults

The rate of failure for the processing core is modelled using FTMAP. The failure probability is evaluated same as in Fault Aware Resource Management (FARM) & Fault Aware Spare Allocation (FASA) which is illustrated in Eq. (5.1) [85].

$$F_{xy}(t) = 1 - e^{-(\lambda_{xy} t)} \tag{5.1}$$

Where F$_{xy}$(t) is the failure probability of the processing core, which is located in the x^{th} row and y^{th} column and 't' is the life time of NoC. λ_{xy} is the failure rate, which is a measure of failure per unit of time. As the failure rate increases, it reduces the lifetime, and it becomes constant until breakdown.

The failure rate is computed by multiplying the total number of failures with the complete operating time. Failure rate (λ_{xy}) is inversely proportional to the total core hours as well as the acceleration factor.

$$\lambda_{xy} \propto (\frac{1}{TCH.AF}) \tag{5.2}$$

Where the total core hours (TCH) are calculated by multiplying the number of units by the total time. AF represents for acceleration factor, which is the Arrhenius equation's test time multiplier. When a device is operated at a high temperature, the AF value is obtained which can be calculated through the given Eq. (5.3) [86].

$$AF = e^{(E/KT_{xy})} \tag{5.3}$$

E = Activation Energy (eV) of the failure mode.
K(Boltzmann Constant) = 8.617 x 10^{-5} eV/^{O} K
T$_{xy}$ = Temperature of the core, which is located at the xth row and yth column.

$$\lambda_{xy} = \frac{1}{TCH} \ . \ e^{(-\frac{E}{KT_{xy}})} \tag{5.4}$$

Where $\frac{1}{TCH}$ is a constant, denoted by 'Z'

$$\lambda_{xy} = Z \ . \ e^{(-\frac{E}{KT_{xy}})} \tag{5.5}$$

Constant 'Z'is calculated as the failure rate per cycle for each processing core operating at a useful life of 10^{-9} under a typical core temperature i.e., 55 °C.

5.3 FTMAP Algorithm

This section incorporates the proposed fault-tolerant core mapping strategy(FTMAP) in order to improve the performance of the system. In this FTMAP algorithm, we calculate the communication energy and execution time. From the obtained communication energy's of the respective core mapping, we find the minimal communication energy which is formulated as below:

Problem 1 : To find out the minimum communication energy (CE_{min}).

Let's consider task graph G(T, E), and the topology N(P, D); perform the mapping function MF : T → P , where $t_i \in \{$ T $\}$ and $p_{ij} \in$ P, and calculate the total communication energy and minimum communication energy which can be obtained from Eq. (5.6) and Eq. (5.7).

$$CE_{Total} = \sum_{\forall t_i \in \{T\}} W(E_{ij}) \times X_{ij} \tag{5.6}$$

The mapping function and total communication energy is explained in Problem 1, whereas, the minimum communication energy is illustrated in Eq. (5.7).

$$CE_{min} = min\{CE : CE \in CE_{Total}\} \tag{5.7}$$

The proposed FTMAP Algorithm is clearly depicted in Algorithm 3. The inputs considered as part of this algorithm are i)Task graph and ii)NoC topology, where the task graph contains various nodes representing the tasks, which can be mapped and scheduled on NoC topology based on minimum communication energy. The output will be considered as the best core mapping.

5.4 Demonstration of FTMAP through a Case study

This section provides the elucidation of proposed Fault-Tolerant Core Mapping (FTMAP) depicted in Algorithm 3. The cores are mapped on the NoC topology based on its minimum communication energy. During the task mapping and scheduling, if any of the core or processing element is failed, those tasks can be migrated to another free core or PE according to their communication energy. Figure 5.4(a) illustrates the application task graph which contains six tasks, Figure 5.4(b) provides the structure of 6 x 6 mesh NoC topology. On the basis of Algorithm 3, initially, the tasks present in the Figure 5.4(a) were mapped in different ways illustrated in Figure 5.4(c) to Figure 5.4(f), and perform the calculation of communication energy for each set of mapping. Finally, from the obtained total communication energy of various mappings, select the mapping which has the minimum communication energy as represented in Figure 5.4(g). If any of the core gets failed, the tasks of the specific failed core were migrated to the nearest available free or manager cores in the NoC platform. The high probability of failure cores were given in the order as V_1, V_4, V_0, V_3, V_2 and V_5.

Algorithm 3 FTMAP Algorithm

```
Input:  Task Graph G = (T,E) ;
        NoC Topology N = (P,D) ;
Output: N = (P,D) Best FT_CoreMapping(FTMAP) and Scheduling NoC
        Topology ;
foreach mapping graph do
    Calculate Communication Energy ;
end
for $t_i \in \{ T \}$ do
    Task Type = get_category_of_task($t_i$) ;
    Allocate $t_i$ to PE ;
    Update Communication Energy ;
    if Task Allocation PE = busy then
        Allocate $t_i$ to another PE (which is neighboring position of previous PE) ;
        Assign Execution time of $t_i$ on $PE_{sel}$ ;
        Update Communication Energy ;
    end
    else if Task Allocation PE = failed then
        Migrate $t_i$ to nearest free PE ;
        Calculate migration time and execution time ;
        Update Communication Energy ;
    end
    Compute Total Communication Energy($E(e_{ij})$) and Processor Execution time
     ;
    if $CE_{min} > E(e_{ij})$ then
        $CE_{min} \leftarrow E(e_{ij})$ ;
        Best FT_CoreMapping(FTMAP) $\leftarrow$ FT_CoreMapping ;
    end
end
return Best FT_CoreMapping(FTMAP) ;
```

5.4.1 Calculation of Communication Energy

The calculation for the total communication energy of a network is performed by using Eq. (4.4) [78].

For NFT, the communication energy is calculated as follows:

$CE_{Total} = CE_{01} + CE_{02} + CE_{13} + CE_{14} + CE_{25} + CE_{34}$

$CE_{01} = 200 \text{ X } (|(1-0)| + |(1-1)|) = 200$

(Here, E_{01} weight is 200, From Fig. 4(g), V_0 parameters are (0, 1), simlarly V_1 parameters are (1, 1))

CE_{02}, CE_{13}, CE_{14}, CE_{25}, and CE_{34} calculated like CE_{01}

$CE_{02} = 200 \text{ X } (|(0- 0)|+ |(2- 1)|) = 200$

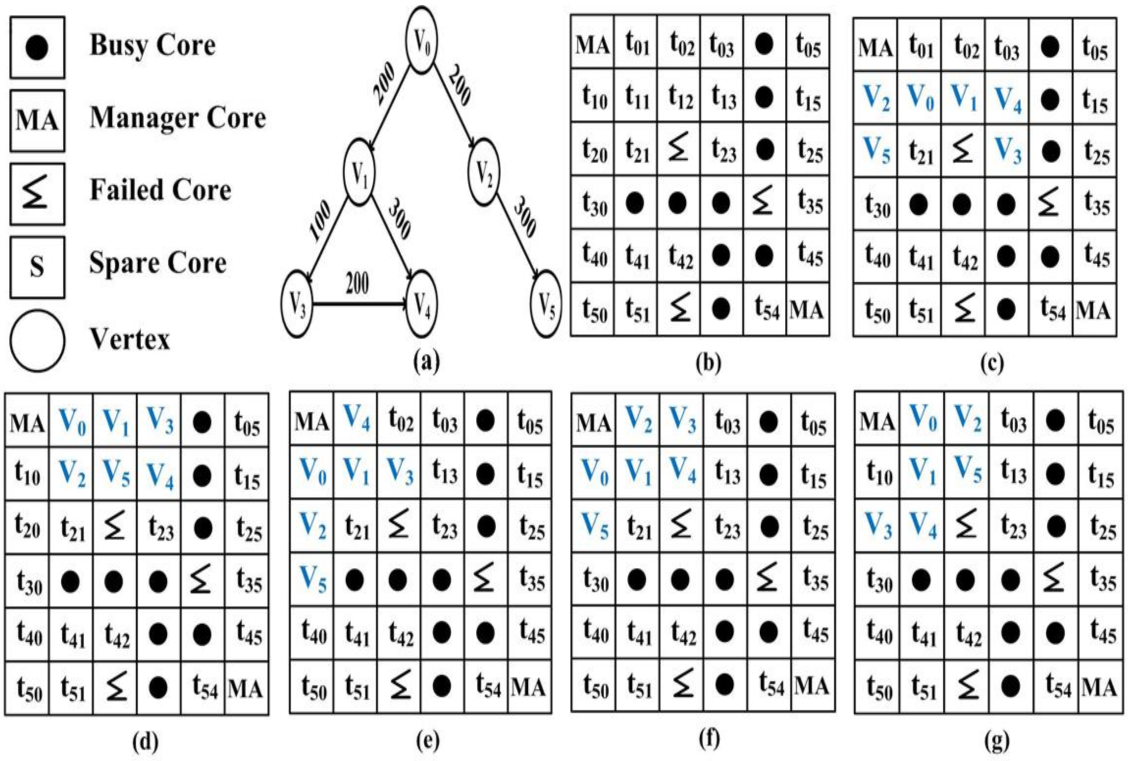

FIGURE 5.4: FTMAP Algorithm: (a) An example of Application Task Graph, (b) 6 x 6 mesh NoC, (c - f) Various mapping ways obtained through proposed algorithm and (g) Best FT Core Mapping.

CE_{13} = 100 X (|(2- 1)|+ |(0- 1)|) = 200
CE_{14} = 300 X (|(2- 1)|+ |(1- 1)|) = 300
CE_{25} = 300 X (|(1- 0)|+ |(2- 2)|) = 300
CE_{34} = 200 X (|(2- 2)|+ |(1- 0)|) = 200

CE_{Total}= 200 + 200 +200 +300 +300 +200 = 1400.

Similarly, the CE_{Total} for 1FT and 2FT are calculated same as NFT by considering the failure probabilities. For 1FT, V_1 is reallocated to the nearest free core, where for 2FT, V_4 is reallocated to next neighboring free core. Communication energy (µJ) of proposed Fault tolerance mapping algorithm with respect to NFT, 1FT, and 2FT denote the results of non-fault tolerance, one-fault tolerance, and two-fault tolerance tabulated in Table 5.1.

Table 5.1: Communication Energy (µJ) of NFT, 1FT, and 2FT on NoC.

Proposed FTMAP algorithm	NFT	1FT	2FT
Communication Energy	1400	1800	2000

5.5 Experimental Results

In this section, we evaluate the proposed fault tolerance mapping algorithm by comparing it with the other related works (FTTG [76], K-FTTG [77]). The proposed methodology has also been examined through simulating the task graphs on different sizes of 5 x 5, 6 x 6, 10 x 10 and 20 x 20 mesh NoC platforms. Utilizing TGFF [87], numerous configurations of synthetic applications have been developed. The real multimedia benchmarks used as part of this research are namely, 1. MPEG4, 2. VOPD, 3. MWD, 4. 263dec, 5. 263enc, and 6. Mp3dec. Table 5.2 represents the characteristics view of multimedia application models [88]. Noxim simulator [79] is used for simulation, which is a cyclical accurate System C simulator for NoC systems. We have simulated the proposed FTMAP algorithm and other related works (FTTG [76], K-FTTG [77]) by taking into consideration of specified multimedia benchmarks i.e. referred in Table 5.2.

Table 5.2: Multimedia Benchmark Properties

Benchmark	Vertices	Edges	Application Domain
MPEG4	12	13	MPEG4 Decoder
VOPD	16	20	Video Object Plane Decoder
MWD	12	12	Multi Window Display
263dec	14	15	H.263 Decoder
263enc	12	12	H.263 Encoder
Mp3dec	13	13	Mp3 Decoder

Table 5.3: Multimedia Benchmarks Communication Energy (µJ) of Proposed algorithm when compared to FTTG and K-FTTG with respect to NFT, 1FT, and 2FT.

Benchmarks	FTTG [76]			K-FTTG [77]			Proposed Algorithm		
	NFT	1FT	2FT	NFT	1FT	2FT	NFT	1FT	2FT
MPEG4	5013	5874	6906	4804	5136	5564	4608	4926	5206
VOPD	4955	5485	6695	4725	5028	5288	4488	4766	5010
MWD	1568	1834	2486	1426	1682	2086	1362	1524	1869
263Dec	28.5	41.5	54.5	25.5	38	46.5	23.5	32	38.5
263Enc	376	512	764	328	488	698	298	424	586
Mp3dec	25.4	38.8	49.6	23.2	34.2	44.6	21	29.8	36

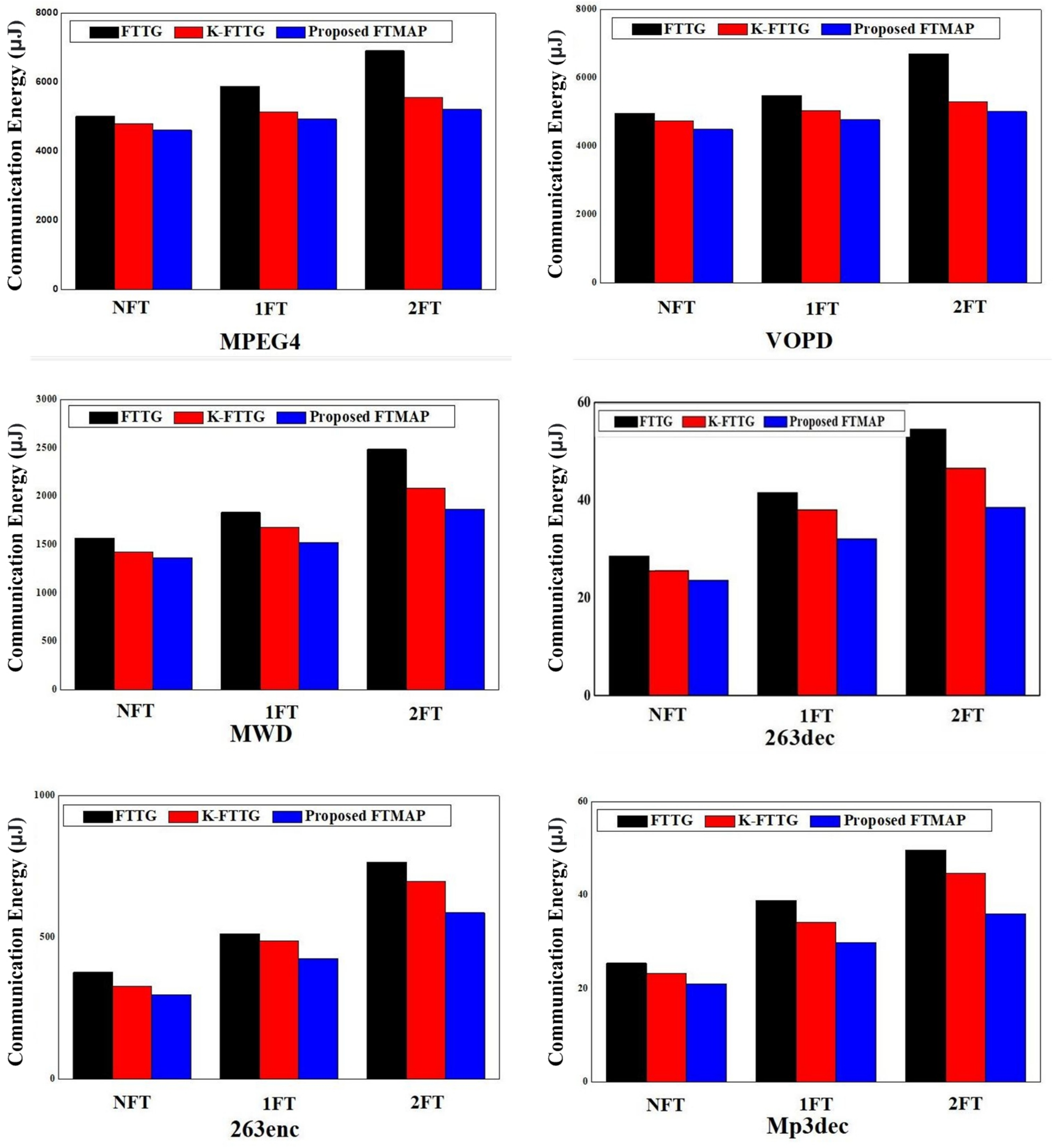

Figure 5.5: Comparison of Communication energy for different failed cores by considering Multimedia benchmarks among FTTG, K-FTTG, and Proposed FTMAP.

Table 5.3 clearly explains the communication energy with respect to NFT, 1FT, and 2FT that denotes the results of non-fault tolerance, one-fault tolerance, and two-fault tolerance respectively. The proposed algorithm signifies the reduction of communication energy efficiency by an average of 8%, 6% when compared to FTTG [76], K-FTTG [77] with respect to NFT, 12%, 9% reduction of communication energy efficiency when compared to FTTG [76], K-FTTG [77] with respect to 1FT,and 14%,

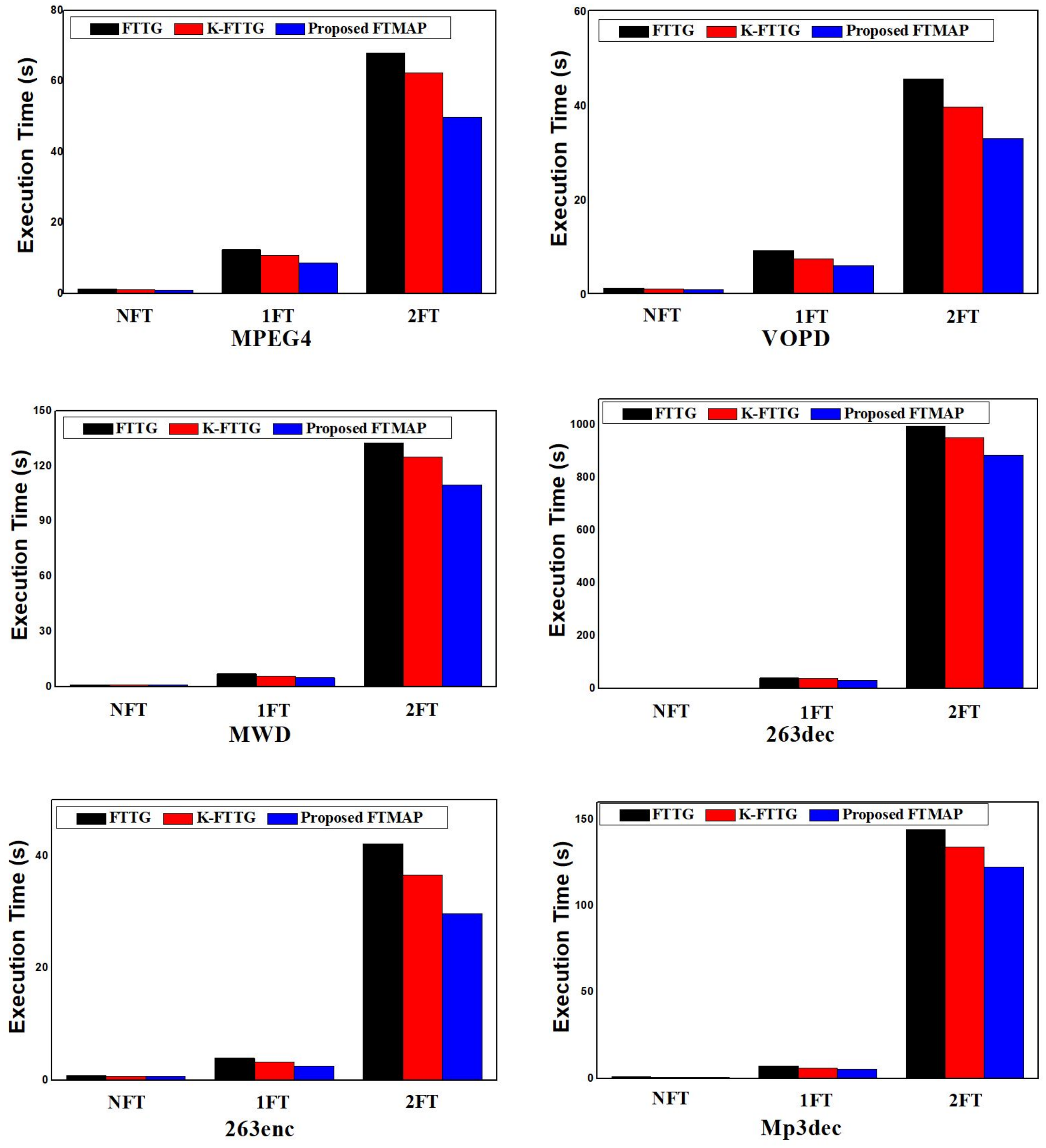

Figure 5.6: Comparison of Execution time for different failed cores by considering Multimedia benchmarks among FTTG, K-FTTG, and Proposed FTMAP.

10% reduction of communication energy efficiency when compared to FTTG [76], K-FTTG [77] with respect to 2FT as depicted in Figure 5.5.

Performance is evaluated as the entire processors execution time, which contains executing task time, waiting time and migration time (when faults occur). Execution time of proposed algorithm is compared with the other two algorithms(FTTG [76], K-FTTG [77]) with respect to NFT, 1FT, and 2FT tabulated in Table 5.4. The proposed algorithm shows the reduction of execution time by an average of 18%,

Table 5.4: Multimedia Benchmarks Execution Time (s) of Proposed algorithm when compared to FTTG and K-FTTG with respect to NFT, 1FT, and 2FT.

Benchmarks	FTTG [76]			K-FTTG [77]			Proposed Algorithm		
	NFT	1FT	2FT	NFT	1FT	2FT	NFT	1FT	2FT
MPEG4	1.38	12.5	68	1.222	10.92	62.402	1.06	8.63	49.8
VOPD	1.2	9.2	45.6	1.005	7.355	39.522	0.95	5.98	32.86
MWD	1.28	7.13	132.8	1.102	5.754	124.97	1.05	4.969	109.998
263Dec	1.3	40.2	996.3	1.216	38.389	952.4	1.09	31.64	886.8
263Enc	0.92	3.98	42.24	0.785	3.226	36.62	0.704	2.59	29.8
Mp3dec	1.08	7.34	144.14	0.810	6.235	134.18	0.746	5.4	122.6

Table 5.5: Evaluation of Communication Energy and Execution time of FTMAP against FTTG and K-FTTG Algorithms with respect to NFT, 1FT, and 2FT.

Performance Metric	FTMAP against FTTG [76]			FTMAP against K-FTTG [77]		
	NFT	1FT	2FT	NFT	1FT	2FT
Communication Energy (µJ)	8%	12%	14%	6%	9%	10%
Execution time (s)	18%	24%	26%	13%	19%	21%

13% when compared to FTTG [76], K-FTTG [77] with respect to NFT, 24%, 19% reduction of the execution time when compared to FTTG [76], K-FTTG [77] with respect to 1FT, 26%, 21% reduction of execution time when compared to FTTG [76], K-FTTG [77] with respect to 2FT as shown in Figure 5.6. These results clearly show that proposed fault tolerance methodology(FTMAP) effectively reduces the communication energy and execution time of the network illustrated in Table 5.5.

5.6 Summary

In this chapter, provided the implementation of FTMAP algorithm, which performs the effective mapping that is fault tolerant in order to reduce the overall communication energy and the execution time. This algorithm emphasizes the mapping of the cores on the basis of selected task graph and mainly focuses on the replacement of faulty cores in a network. The simulation outcomes outperform the reduction of communication energy by an average of 8%, 12%, 14% with respect to NFT, 1FT, 2FT when compared to FTTG and 6%, 9%, 10% with respect to NFT, 1FT, 2FT when compared to K-FTTG and execution time by an average of 18%, 24%, 26% with respect to NFT, 1FT, 2FT when compared to FTTG and 13%, 19%, 21% with respect to NFT, 1FT, 2FT when compared to K-FTTG.

Chapter 6

Efficient Real-Time Embedded Application Mapping

The primary responsibility of any mapping technique is to map the tasks to the cores available in the chosen topology. Then, the mapping of an application allows to perform the tasks as mapped accordingly and provide the suitable output. Application mapping is the strategy used for efficiently mapping the cores. Each core present in the core graph is mapped to each vertex on the NoC platform sequentially. The method of applying the core mapping differs from one to another. As the number of cores is increasing drastically, many mapping techniques came into existence to provide a reliable result. So, it is essential to follow certain rules by considering the critical shortcomings in the present NoC methodologies to design a efficient application mapping. The mapping of cores performed in two ways, namely static mapping and dynamic mapping. Based on the requirement, the designers can choose their respective mapping for their NoC architecture.

In this chapter, implemented an Efficient Real-Time Embedded Application Mapping i.e. (ERTEAM), which initially identifies the minimum Node Average Distance (NAD) and maps the vertices in this region on the basis of minimum communication energy. This algorithm is evaluated a set of real-time embedded applications [87] such as H264 encoder (H264_enc), MP3 decoder (MP3_dec), Network processing (NP), MPEG2 encoder (MPEG2), Multimedia Systems (MMS), Video object plan decoder (VOPD). This algorithm outperforms the latency, throughput, simulation time and communication energy when compared with the BBPCR [60] and SBMAP [61].

6.1 ERTEAM Algorithm

The proposed ERTEAM Algorithm is clearly depicted in Algorithm 4. This mapping algorithm mainly requires Application core graph which resembles any NoC application and a mesh based NoC topology. Consider the ACG and NoC topology as an input and perform the core mapping. Initially, before performing the core mapping, calculate the NAD for the mapping region. After the mapping region is finalized, arrange the vertices based on the minimum communication energy. Now map the vertices on the PE and calculate the communication energy. For every mapping, update the communication energy and if the total communication energy is less than the minimum communication energy, it is considered as best core mapping.

Algorithm 4 ERTEAM Algorithm

```
Input: Network Core Graph (NCG) G = (P,A) ;
       NoC Architecture Graph (NAG) A = (C,D) ;
Output: NoC Architecture Mapping Graph (NMG) = M(C,D) ;
       M: Mapping Region ;
foreach mapping region do
    Calculate Effective Region ;
    {
    Select Effective Region corresponding to the min Node Average Distance
      (NAD) ;
    }
end
Initialize Mapping ;
min cost = $\infty$ ;
do
    Calculate Core Bandwidth (BW) ;
    Calculate Communication Distance (CD) ;
    Calculate Communication Energy (CE) ;
    {
    CE = BW $\times CD$
    }
    if min CE = Total CE ;
     then Total Communication Energy < min Communication Energy
    end
    Core Mapping = min CE mapping ;
    Total CE (WCE) = $\sum BW_{(P_i,P_j)} \times CD_{(C_i,C_j)}$
while Next Mapping;
return Best Communicative Mapping with lowest Communication Energy ;
Calculate Core Mapping Execution Time ;
Calculate Latency and Throughput ;
```

6.2 Demonstration of ERTEAM through a Case study

Proposed core mapping algorithm explained in Algorithm 4. Network core graph (NCG) and NoC architecture graphs (NAG) are acts as input, NoC Architecture Mapping Graph (NMG) as output. Initially, select the efficient mapping region using minimum Node average distance (NAD), reducing the mapping area. Then, Processing Element (PE's) in NCG mapped on efficient mapping region in NoC according to the minimum communication energy. A simple example clearly explained in Figure 6.1. A simple network core graph has shown in Figure 6.1(a) and 5 x 5 NoC Architecture Graph shown in Figure 6.1(b). As the number of vertices is 7 in the NCG, the efficient mapping region is selected based on NAD, preferably a size 3 x 3 region shown in Figure 6.1(c). Finally, NCG vertices mapped on 3 x 3 region according to the minimum communication energy of the network shown in Figure 6.1(d).

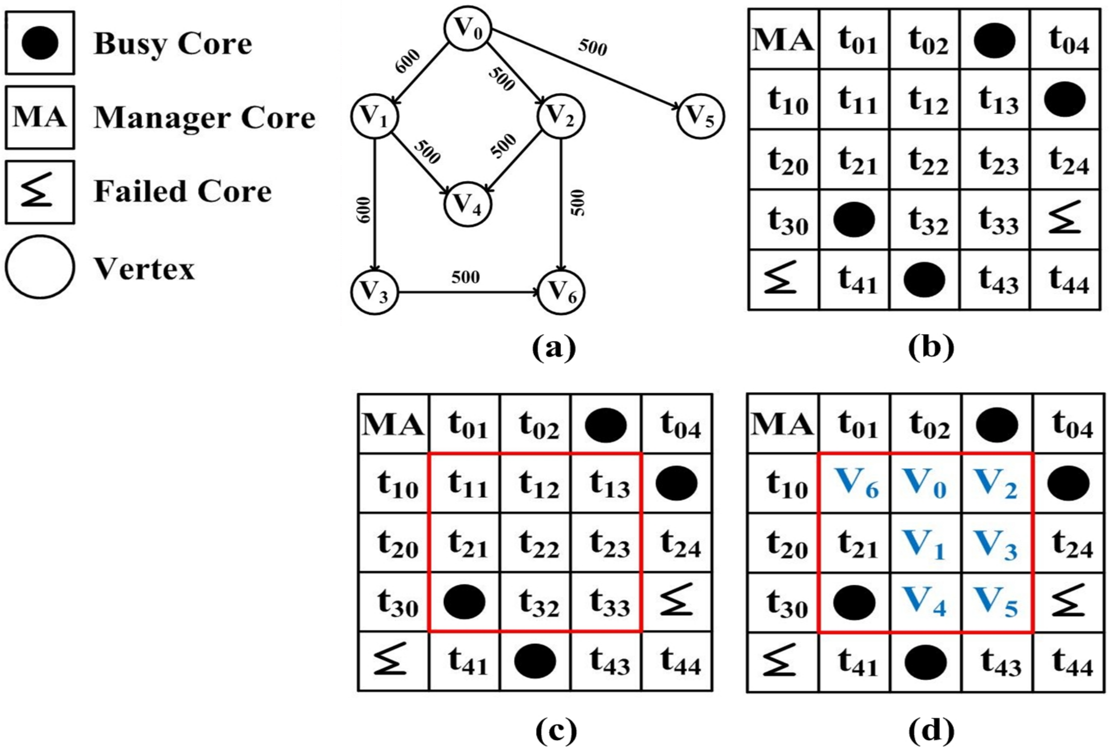

FIGURE 6.1: ERTEAM Algorithm: (a) An example of Application Task Graph, (b) 5 x 5 mesh NoC, (c) Mapping region obtained through minimum NAD, and (g) Efficient ERTEAM Core Mapping.

6.3 Experimental Results

In this section, we conduct sets of comprehensive experiments to evaluate the effectiveness of the ERTEAM algorithm, mapping performance and communication energy. The mentioned metrics are compared with state-of-the-art approaches on embedded applications. A set of embedded applications exploited for evaluation. Application names and their numbers of cores are shown in Table 6.1 [87]. The best mapping pattern found using a C++ program, the simulations carried out on Noxim simulator [79].

Table 6.1: Specifications of Embedded Applications.

Application	No. Cores	Network Size
H264 encoder (H264_enc)	36	6 x 6
MP3 decoder (MP3_dec)	16	4 x 4
Network processing (NP)	16	4 x 4
MPEG2 encoder (MPEG2)	16	4 x 4
Multimedia Systems (MMS)	16	4 x 4
Video object plan decoder (VOPD)	16	4 x 4

For all of the following simulations, Network Core Graph and NoC Architecture Graph are identical. This research methodology evaluates performance metrics such as Latency, Simulation Time, Throughput, and Communication Energy.

6.3.1 Latency

The time taken by the packet's header flit to migrate between any source to destination in the network referred to as latency. According to network congestion, latency frequently involves a packet's waiting time between any source to the destination node, illustrated in Eq. (6.1).

$$Latency = \frac{1}{K}\sum_{n=1}^{K}(L_n) \tag{6.1}$$

K = Total number of packets reaching their destination cores. L_n = The clock cycle latency for the n^{th} node.

Table 6.2 explains the obtained latency of the proposed algorithm ERTEAM (in terms of cycles) compared to the BBPCR [60] and SBMAP [61]. Therefore, the graphical representation of the latency depicted in Figure 6.2.

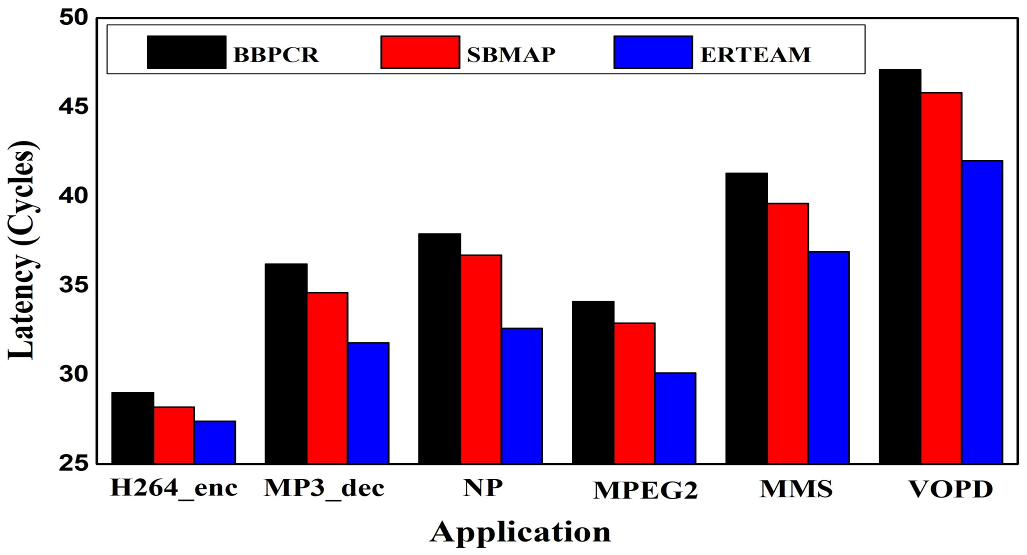

FIGURE 6.2: Latency of the proposed algorithm ERTEAM (in terms of cycles) compared to the BBPCR and SBMAP.

TABLE 6.2: Latency of the proposed algorithm for various embedded applications.

Latency (Cycles)			
Application	BBPCR	SBMAP	ERTEAM
H264 encoder (H264_enc)	29	28.2	27.4
MP3 decoder (MP3_dec)	36.2	34.6	31.8
Network processing (NP)	37.9	36.7	32.6
MPEG2 encoder (MPEG2)	34.1	32.9	30.1
Multimedia Systems (MMS)	41.3	39.6	36.9
Video object plan decoder (VOPD)	47.1	45.8	42

6.3.2 Simulation Time

The term simulation time is defined as the overall time required by the system to execute the tasks during the mapping of cores, known as the simulation time or the execution time. Thus, lesser simulation time provides an increase in the performance of the system. Table 6.3 illustrates the obtained simulation time of the proposed algorithm ERTEAM (in terms of seconds) compared to the BBPCR [60] and SBMAP [61]. Therefore, the graphical representation of the simulation time depicted in Figure 6.3.

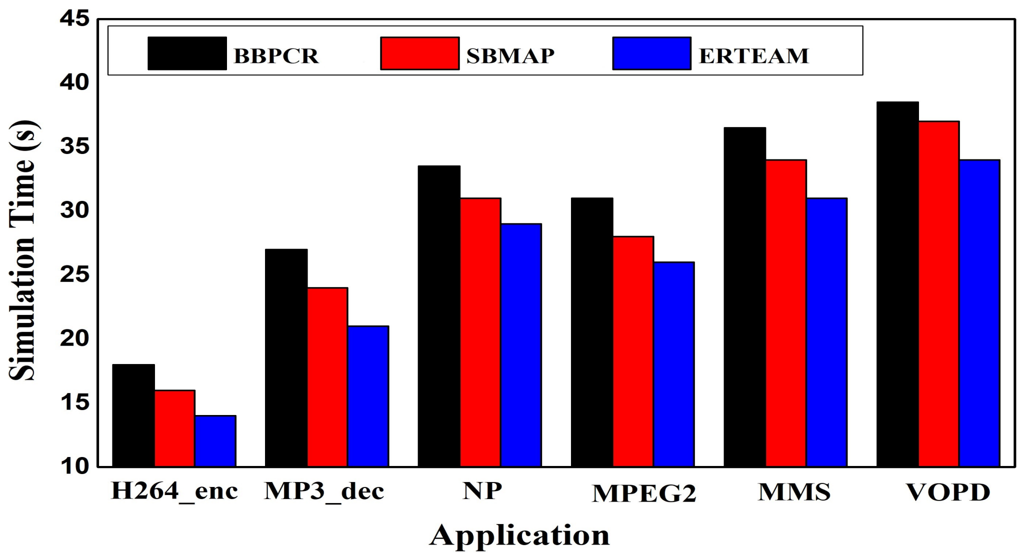

FIGURE 6.3: Simulation Time of the proposed algorithm ERTEAM (in terms of seconds) compared to the BBPCR and SBMAP.

TABLE 6.3: Simulation time of the proposed algorithm for various embedded applications.

Simulation Time (s)			
Application	BBPCR	SBMAP	ERTEAM
H264 encoder (H264_enc)	18	16	14
MP3 decoder (MP3_dec)	27	24	21
Network processing (NP)	33.5	31	29
MPEG2 encoder (MPEG2)	31	28	26
Multimedia Systems (MMS)	36.5	34	31
Video object plan decoder (VOPD)	38.5	37	34

6.3.3 Throughput

Throughput considered as one of the important parameters regarding the performance of the system. It represents the maximum amount of information that transferred in a given amount of time. Therefore, the mathematical formulation for throughput illustrated in Eq. (6.2).

$$Throughput = \frac{R_p}{N \times N_p} \tag{6.2}$$

Where, R_p = total number of received packets, N = the total number of cores, N_p = number of clocks cycles lapsed from the first generated packet to the last received packet.

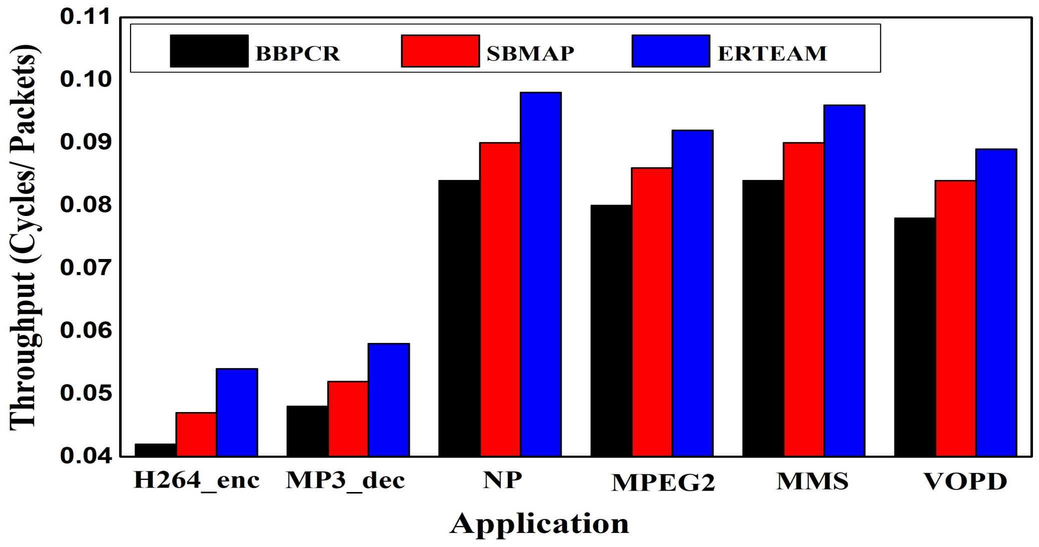

FIGURE 6.4: Throughput of the proposed algorithm ERTEAM (in terms of cycles/packets) compared to the BBPCR and SBMAP.

Table 6.4 describes the resultant throughput of the proposed algorithm ERTEAM (in terms of cycles/packets) compared to the BBPCR [60] and SBMAP [61], whereas the graphical representation of throughput depicted in Figure 6.4.

TABLE 6.4: Throughput of the proposed algorithm for various embedded applications.

Throughput (Cycles/ Packets)			
Application	BBPCR	SBMAP	ERTEAM
H264 encoder (H264_enc)	0.042	0.047	0.054
MP3 decoder (MP3_dec)	0.048	0.052	0.058
Network processing (NP)	0.084	0.09	0.098
MPEG2 encoder (MPEG2)	0.08	0.086	0.092
Multimedia Systems (MMS)	0.084	0.09	0.096
Video object plan decoder (VOPD)	0.078	0.084	0.089

6.3.4 Communication Energy

The term Communication Energy defined as the sum of differences between their respective modules establishes the distance between any two nodes in a chosen topology of a network. Table 6.5 illustrates the communication energy of the proposed algorithm ERTEAM (in terms of µJ) compared to the BBPCR [60] and SBMAP [61]. Therefore, the graphical representation of the communication energy depicted in Figure 6.5.

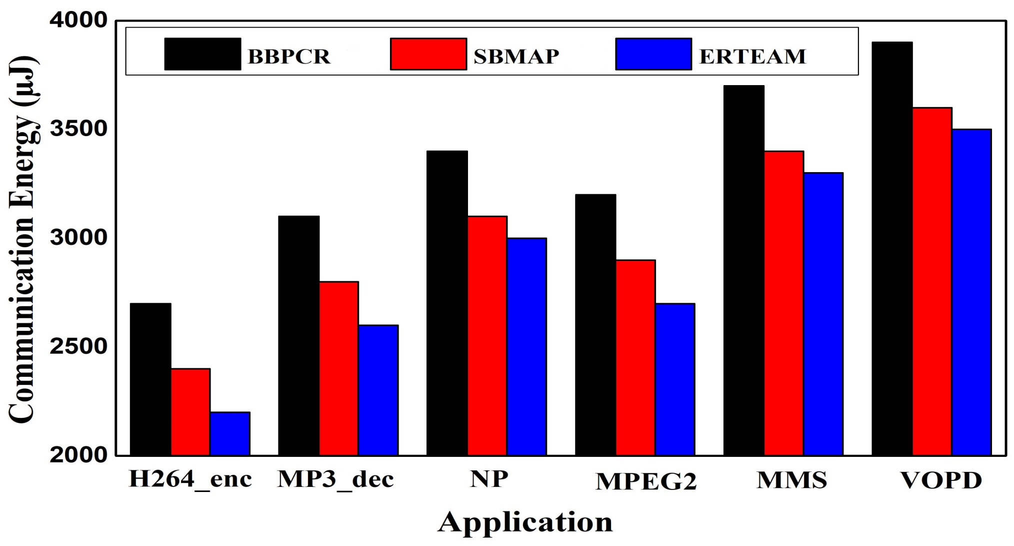

FIGURE 6.5: Communication Energy of the proposed algorithm ERTEAM (in terms of µJ) compared to the BBPCR and SBMAP.

TABLE 6.5: Communication Energy of the proposed algorithm for various embedded applications.

Communication Energy (µJ)			
Application	BBPCR	SBMAP	ERTEAM
H264 encoder (H264_enc)	2700	2400	2200
MP3 decoder (MP3_dec)	3100	2800	2600
Network processing (NP)	3400	3100	3000
MPEG2 encoder (MPEG2)	3200	2900	2700
Multimedia Systems (MMS)	3700	3400	3300
Video object plan decoder (VOPD)	3900	3600	3500

Table 6.6 demonstrates the evaluation of the metrics for the proposed ERTEAM algorithm against the BBPCR [60] and SBMAP [61]. The reduction of latency improved by an average of 12.3% and 8.4% against BBPCR [60] and SBMAP [61], the overall simulation time reduced to 19%, 9.6% compared to BBPCR [60] and SBMAP [61]. Furthermore, the throughput of ERTEAM improved by an average of 14.5%, 7.8% compared to BBPCR [60] and SBMAP [61] and the communication energy reduced to 15.6%, 5.2% against BBPCR [60] and SBMAP [61].

Table 6.6: Evaluation of latency, simulation time, throughput and communication energy of ERTEAM against BBPCR and SBMAP.

	ERTEAM against BBPCR [60]	ERTEAM against SBMAP [61]
Latency (Cycles)	12.3%	8.4%
Simulation Time (s)	19%	9.6%
Throughput (Cycles/ Packets)	14.5%	7.8%
Communication Energy (µJ)	15.6%	5.2%

6.4 Summary

In this chapter, implemented a mapping strategy entitled Efficient Real-Time Embedded Application Mapping (ERTEAM) that is applied to real-time embedded applications to improve the system performance. This implementation chooses the mapping region based on the minimum Node Average Distance (NAD). After providing the mapping area, the PE's are embedded in the arrangement of minimum communication energy between the cores. The execution time is calculated after obtaining the best core mapping as an output. The resultant outcome of the proposed mapping technique provides low latency, less simulation time, less communication energy and the overall throughput increased compared to BBPCR and SBMAP when applied to the mentioned embedded real-time applications.

Chapter 7

Conclusion and Future work

The customization of on-chip network architectures lies at the heart of reducing the communication energy and enhancing the performance of NoC-based systems. In this dissertation, we proposed core mapping, spare core replacement and fault tolerance methods. The method was proposed at the system-level to allow for a best mapping. Comparative analysis were carried out as a proof of concept to explain the significance of our contributions. We believe that our contributions could help reducing the communication energy and improving the performance of the underlying architectures for modern communication-intensive SoC applications.

7.1 Conclusion

In this thesis, we concentrated on the efficient core mapping technique named as BMAP which mainly deals with the standard topology selection and the customization of NoC platforms. This proposed technique is applied on various benchmarks applications that has shown a great improvement reliability, delay, area and power whereas a minimal objective function is achieved for complex networks in 2D topologies. The performance evaluation is carried out through the synthesis of SPLASH-2 Benchmark using Noxim simulator which outperformed its metrics such as Speed-up Execution Time increased by 40%, 30% and 20%, Latency reduced by 42%, 34% and 28%, Energy efficiency improved by 36%, 30%, 26% and Power consumption reduced by 32.6%, 28.2%, and 26.4% when compared with NMAP, MMAP, and EMAP algorithms respectively.

The proposed ACM technique involves in mapping the cores and also the spare core replacement. To perform core mapping, initially we need to figure out the mapping region using the NAD and PVR functional metrics. After the mapping region is finalized, the core mapping is performed. If any faults occurred at a core, fault diagnosis method is applied which provides the damaged core location and correct it using the error detection and correction mechanism. Even after applying the method, if faults occur then the spare core replacement is performed where the tasks of the failed core are migrated to the free spare core improving the system reliability. The experiments conducted on 6 x 6 mesh NoCs revealed that the adaptive core mapping technique exhibited greater throughput, lesser delay, and energy consumption than other related algorithms. The proposed algorithm is applied to the PARSEC benchmark using GEM5 Simulator, where the obtained results show a significant improvement in latency and system performance. An experimentation environment verified on the Kintex-7 FPGA KC705 board, which elucidates the faulty cores, recovery cores and spare cores in the network. The results implicate a dramatic decrease in area, power consumption, and an increase in throughput, which illustrates the efficiency of the proposed ACM algorithm.

The fault-tolerant mapping technique known as FTMAP mainly focuses on finding out the failure probability in a core mapping network and provide the solution through migrating the faulty core to the nearest free core. This provides the evaluation of communication energy and the execution time of the network by employing it on various multimedia benchmarks. The experimental outcomes reveal that it reduces the communication energy by 8%, 12%, 14% with respect to NFT, 1FT, 2FT compared to FTTG and 6%, 9%, 10% with respect to NFT, 1FT, 2FT when compared to K-FTTG. The reduction of the execution time has also outperformed by 18%, 24%, 26% with respect to NFT, 1FT, 2FT compared to FTTG and 13%, 19%, 21% with respect to NFT, 1FT, 2FT when compared to K-FTTG.

The proposed ERTEAM entitled as Efficient Real-Time Embedded Application Mapping provides a core mapping technique that initially identifies the minimum Node Average Distance (NAD) and maps the vertices in this region on the basis of minimum communication energy for NoC applications. This algorithm is evaluated a set of real time embedded applications which outperforms the latency, throughput, simulation time and communication energy. The experimental outcomes reveals a reduction in latency at 12.3% against BBPCR and 8.4% against SBMAP. The simulation time reduces at an average of 19% against BBPCR and 9.6% against SBMAP. The throughput increases at an average of 14.5% against BBPCR and 7.8% against SBMAP and reduces the communication energy by 15.6% against BBPCR and 5.2%

against SBMAP. The limitation that was observed during the execution of the proposed techniques is the area overhead. Due to the performance improvements, this factor can be ignored.

7.2 Future Work

This research work could be extended in two ways. The first is an expansion of this research into 3D NoCs to attain reliability. The 3D design adds a whole new level of complexity towards the equation of factors to be taken into account to attain reliability. Because of the natural stacking in 3D, hotspots become a serious difficulty which must be addressed. Additionally, because the baseline system is developed in three dimensions, thorough analysis of the core mapping of the protection system which is incorporated to accomplish fault tolerance is required. The complete evaluation of a 2D designed protection system against a 3D designed protection system as a result, reliability improvement obtained and also communication energy expense should be included in the analysis of fault tolerance. The current work is primarily focused on achieving fault tolerance core mapping for NoC based system. The second possible extension to this work is apply fault tolerance mechanism to entire NoC (core, router and network interface).

Appendix A

Evaluation

A.1 Reliability Assessment

The reliability of a system is determined in 2 stages: 1) Determining the best mapping technique in terms of effectiveness as well as weighted communication energy, and 2) determining overall reliability for core mapping with a faulty core. The reliability for the i^{th} condition of all M possibilities when n links are faulty is given by

$$R_{i,n} = \sum_{SD} R_i^{SD} F_i^{SD} \tag{A.1}$$

where R_i^{SD} is the reliability of the source (S) to destination (D) under the i_{th} fault condition, and F_i^{SD} indicates communication between S and D. Which is defined by

$$F_i^{SD} = \begin{Bmatrix} 1 & a_{SD} & \in A \\ 0 & a_{SD} & \notin A \end{Bmatrix} \tag{A.2}$$

Since the reliability (R_{NoC}) varies when the faulty condition changes, it is important to consider all faulty conditions. Let $P_{I,n}$ denotes the faulty probability when the i^{th} faulty core occurs, which is indicated below, and I indicates a set of 'n'faulty cores for the i^{th} faulty condition.

$$P_{I,n} = \prod_{j=1,j\in I}^{N} (p_j) \prod_{j=1,j\notin I}^{N} (1-p_j) \tag{A.3}$$

Where P_j represents the faulty probability of the j^{th} core.

The reliability of fault core mapping of the i^{th} faulty core among the number of failed cores ‘n’ for a particular application is represented by $R^A_{i,n}$. The reliability of core mapping for an application is as follows:

$$R^A = \sum_{n=0}^{N} \sum_{i=1}^{M} R^A_{i,n} P_{I,n} \tag{A.4}$$

The reliability can then be normalized by a normalization factor N_R, that is calculated using the equation above while addressing the network with the most defective links. The normalized reliability is defined as

$$\text{NR} = R/N_R \tag{A.5}$$

A.2 Evaluation of Latency

Latency gets categorized into 3 elements throughout this work: 1) communication time between sender to receiver, 2) fault-related waiting time, as well as 3) congestion-related waiting time.

The basic communication time refers to the time it takes for the head flit to travel from source towards destination.

$$LC_{i,n} = \sum_{SD} [t_w d_i^{SD} + t_r(d_i^{SD} + 1)] F_i^{SD} \tag{A.6}$$

where t_r and t_w represent the time consumption of transporting a flit through a router and a link, respectively, and F_i^{SD} is defined in reliability concept. It is anticipated that once the head flit encounters a broken link, the head flit would be transferred to next cycle again. The transmission of such head flit would be attempted for each subsequent cycle till the connection fault is cleared. As a result, the average wait time was utilized to calculate the waiting time incurred by a bad (faulty) connection j.

$$LF_j = lim_{T\to\infty}((p_j) + 2(p_j^2) + 3(p_j^3) + ... + T(p_j^T)) = \frac{p_j}{(1-p_j)^2} \tag{A.7}$$

where p_j is the failure probability of link j and T means the cycles the head flit must wait. The first-in-first-out (FIFO) queue is used to alleviate congestion, so every router is treated like a server inside the queue. Whenever the transmitter and receiver are known, data packet is sent to the certain router using predetermined routing methods. There will be only 1 server at every queue inside this situation. Adaptive routing methods, on the other hand, allow the data packet to choose whether router can be used based network's current state.

This indicates here that packet is ready to be served by several hosts. As a result, the G/G/m-FIFO priority queue is being used to calculate the waiting time of congestion, because the inter-arrival time as well as processing times are both considered independent general probabilities. Using the Allen–Cunneen formula, the waiting time of the u^{th} source point to the v^{th} destination point is can be calculated

$$WT_{u\to v} = \frac{WT_0}{(1-\sum_{x=u}^{U}\rho_{x\to v})(1-\sum_{x=u+1}^{U}\rho_{x\to v})} \tag{A.8}$$

Then the latency for the i^{th} faulty condition of n^{th} core being faulty is calculated by

$$L_{i,n} = LC_{i,n} + \sum_{SD}\left[\sum_{k=1}^{d_i^{SD}} WT_{U(K)\to V(K)}^{R(K)} + \sum_{j=1}^{d_i^{SD}} LF_{L(j)}\right]F_i^{SD} \tag{A.9}$$

where R(K) is the function to obtain the index of the K^{th} core on the communication path of S and D, U(K) and V(K)are the functions to obtain the index of the source core and destination core, and L(j) is the function to obtain the number of the j^{th} link. When all the faulty conditions are taken into account, the total latency is then calculated by

$$\mathrm{L} = \sum_{n=0}^{N}\sum_{i=1}^{M} L_{i,n}P_i \tag{A.10}$$

Appendix B

NoC Simulators

There seem to be a variety of assessment tools and approaches available to help in NoC research. Every tool strives to address one or more areas of NoC architecture space exploration, such as vertex configuration, topology configuration, routing algorithms, virtual channels, Data Transmission needs, benchmarks, and results evaluation.

Several NoC emulators were developed for NoC assessment as well as design space exploration. A collection of NoC emulators as well as tools accessible to model and analyze various kinds of NoC is provided by Cristinel Ababei et al., [89] and Achballah et al., [90]. We'll look at something and examine a few of the source code NoC emulators throughout this part. Every simulator does have its own set of capabilities, as well as certain limits. One of the most typical challenges in choosing the correct NoC emulator is that existing tools were usually powerful in some areas while lacking with others. The NoC simulator can indeed be categorized into two groups:

(1) General network simulators that can be used for NoC simulations (e.g. NS2, NS3, Omnet++, Wattch, Hotspot, Netsim, Gem5 Simulator, Graphite, Hornet, Opnet, Fusionsim).

(2) Specific NoC simulators, which are explicitly designed for NoC simulation (e.g. BookSim, HNOCS, WormSim, Ocsim, Vnoc, Matrics, SICOSY, Tpzsimul, Garnet, SUNMAP, Ocintsim, Noxim, Nostrum, Nirgam, Occn, Nocsim, NoCTweak, Atlas, Gpnocsim, Xmulator, NONMAP, ReliableNoC, MapoNoC, Phoenixsim, Access Noxim and ORION).

The NoC simulator may have the following input parameters:

(1) Configuration options: The type of application traffic simulated by the NoC tool is defined by configuration settings. Synthetic traffic patterns or integrated software traces might be used. It may additionally have a simulation seed value, a log file for simulation results, a warm-up period for the network to consolidate, and a simulation program execution decision.

(2) Synthetic options: The size and kind of topology for traffic, such as 2D mesh, as well as the type of synthetic traffic pattern, such as random, transpose, bit-complement, bit-reverse, bit-shuffle, bit-rotate, and hotspot routers, are defined by synthetic options. Hotspots are network routers that accept packetized data at a faster pace than they can process.

(3) Embedded application traces: Embedded applications, such as a VOPD, multimedia system, multi window display, MPEG4 decoder, and E3S benchmarks, are genuine application task graphs employed in the simulation.

(4) Mapping option: Mapping option such as near-optimal mapping (NMAP), simulating annealing (SA), branch and bound (BB) should also be included for obtaining optimised latency, throughput and energy consumption.

(5) Traffic options: The amount of flits injected by each core every cycle (flit injection rate), the probability distribution of the interval between two injected packets, packet length, and flits per packet selection are all available traffic settings.

(6) Router settings: The type of router, such as wormhole router, virtual channel router, shared queues router, bufferless router, and circuit-switched router, is determined by the router settings. It also specifies the pipeline type, number of pipeline stages, and buffer depth, among other things.

(7) Routing options: Options for routing XY dimension-ordered routing, west-first, north-last, and odd-even (OE) minimum adaptive routing are examples of routing algorithms. It may also feature output port selections such as X dimension first, dimension closest to the destination first, dimension farthest from the destination first, round-robin among output ports, output port with greatest credit first, switching arbitration policy, and inter router link length.

(8) Technology settings: It includes CMOS technology process (e.g. 90, 65, 45, 32, 22 nm), clock frequency and supply voltage selection.

(9) Measurement options: Throughput, power, delay, latency, and energy consumption are among the output measuring metrics. Before hardware implementation, the output performance parameters forecast the characteristics of the NoC multicore system.

B.1 Noxim Simulator

SystemC was used to create Noxim [91], a NoC simulator. It offers a command-line interface (CLI) that allows you to parametrize different NoC modules. Network size, routing algorithm, buffer size, injection rate, as well as traffic pattern could all be customized in Noxim. During synthesized traffic conditions, Noxim exclusively permits mesh topology including wormhole routers. The outcomes of the NoC assessment are measured in units of throughput, latency, as well as energy consumption. Maximum data packets transmitted or received, average global throughput, maximum and minimum global delay, power consumption, as well as other complete assessment parameters can be examined [79].

B.2 GEM5 Simulator

Gem5 is a tool [82] to simulate hardware architecture with different components such as cores, memories and buses. Gem5 supports the most commercial ISA: ARM, ALPHA, MIPS, PowerPC, SPARC and x86 (64 bits) and it is written primarily in C++ and Python. When starting a simulation, it is possible to modify various parameters such as core type, core number, memory type and size, cache size, associativity, disk image, and kernel. With this configuration, gem5 will simulate our architecture allowing us to execute applications and give us various output files containing the exact simulated configuration as well as statistics like the numbers of access to the memories, the execution times, and the transactions during the simulation, etc.

Appendix C

KC705 Evaluation Board

C.1 Overview

The Kintex®-7 FPGA KC705 evaluation board offers a hardware environment for creating and testing architectures for the Kintex-7 XC7K325T-2FFG900C FPGA. The KC705 board has DDR3 SODIMM memory, an 8-lane PCI Express® interface, a tri-mode Ethernet PHY, general purpose I/O, and a UART interface, which are all standard features in embedded computing systems. FPGA Mezzanine Cards (FMCs) can be connected to either of the two VITA-57 FPGA mezzanine connections on the board to offer additional functionalities. FMCs with a high pin count (HPC) and a low pin count (LPC) are available. For a comprehensive list of features, see KC705 Board Features [84]. Each feature's specifics are shown below:

C.2 KC705 Board Features

- Kintex-7 XC7K325T-2FFG900C FPGA
- 1 GB DDR3 memory SODIMM
- 128 MB Linear Byte Peripheral Interface (BPI) flash memory
- 128 Mb Quad Serial Peripheral Interface (SPI) flash memory
- USB JTAG via Digilent module
- Clock generation
    1. Fixed 200 MHz LVDS oscillator (differential)
    2. Inter-integrated circuit (I2C) programmable LVDS oscillator (differential)
    3. SMA connectors (differential)

  4. SMA connectors for GTX transceiver clocking

- GTX transceivers
  1. FMC HPC connector (four GTX transceivers)
  2. FMC LPC connector (one GTX transceiver)
  3. SMA connectors (one pair each for TX, RX, and REFCLK)
  4. PCI Express (eight lanes)
  5. Ethernet PHY SGMII interface (RJ-45 connector)
- PCI Express endpoint connectivity
- SFP+ Connector
- 10/100/1000 tri-speed Ethernet PHY
- USB-to-UART bridge
- High-Definition Multimedia Interface ^{TM} (HDMI) technology codec
- I2C bus
  1. I2C mux
  2. I2C EEPROM (1 KB)
  3. HDMI codec
  4. FMC HPC connector
  5. FMC LPC connector
  6. SFP+ connector
  7. I2C programmable jitter-attenuating precision clock multiplier
- Status LEDs
- User I/O
  1. USER LEDs (eight GPIO)
  2. User pushbuttons (five directional)
  3. CPU reset pushbutton
  4. User DIP switch (4-pole GPIO)
  5. LCD character display (16 characters x 2 lines)
- Switches
  1. Power on/off slide switch
  2. FPGA _ PROG _ B pushbutton switch
- VITA 57.1 FMC HPC Connector
- VITA 57.1 FMC LPC Connector

- Power management
- XADC header
- Configuration options
  1. Linear BPI flash memory
  2. Quad SPI flash memory
  3. USB JTAG configuration port
  4. Platform cable header JTAG configuration port

C.3 KC705 Board Component Descriptions

KC705 board components as shown in Figure C.1 and component description clearly mentioned in Table C.1.

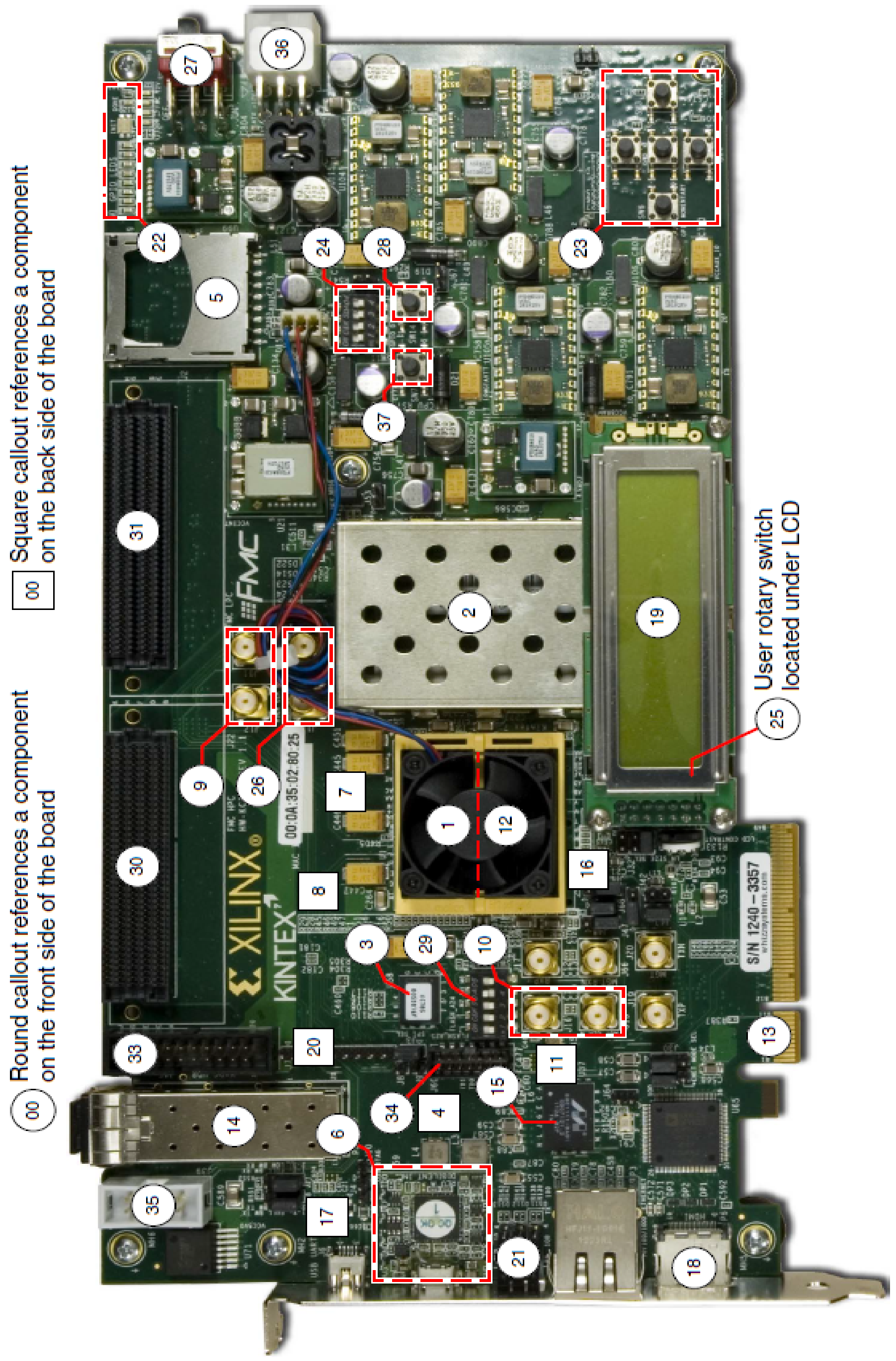
00 Round callout references a component on the front side of the board
00 Square callout references a component on the back side of the board
25 User rotary switch located under LCD
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
26
27
28
29
30
31
33
34
35
36
37

FIGURE C.1: KC705 Board Components

Table C.1: KC705 Board Component Descriptions

Callout	Reference Designator	Component Description	Notes
1	U1	Kintex-7 FPGA (Located under fansink)	XC7K325T-2FFG900C, Radian INC3001-7_1.5BU_LI98
2	J1	DDR3 Memory Module, under EMI shield	Micron MT8JTF12864HZ-1G6G1
3	U58	LinearBPI Flash Memory	Micron PC28F00AP30TF
4	U7	Quad SPI Flash Memory	Micron N25Q128A13BSF40F
5	U9	SD Card Interface	Molex 67840-8001
6		USB JTAG Module	Digilent USB JTAG Module (with micro-B receptacle)
7	U6	System Clock Source (back side of board)	SiTime SIT9102-243N25E200.0000
8	U45	Programmable User Clock Source (back side of board)	Silicon Labs SI570BAB0000544DG
9	J11, J12	User SMA Clock Input	Rosenberger 32K10K-400L5
10	J15, J16	GTX SMA Clock Input	Rosenberger 32K10K-400L5
11	U70	Jitter Attenuated Clock	Silicon Labs SI5324C-C-GM
12		GTX Transceivers	Embedded within FPGA U1
13	P1	PCI Express Edge Connector	8-lane card edge connector
14	P5	SFP/SFP+ Connector	Molex 74441-0010
15	U37	10/100/1000 Tri-Speed Ethernet PHY	Marvell M88E1111-BAB1C000
16	U2	SGMII GTX Transceiver Clock Generator	ICS ICS84402IAGI-01LF
17	J6, U12	USB-to-UART Bridge	Silicon Labs CP2103GM bridge (back side of board) and min-B receptacle (front side of board)

Callout	Reference Designator	Component Description	Notes
18	P6, U65	HDMI Video Output	Molex 500254-1927, Analog Devices ADV7511KSTZ-P
19	J31	LCD Character Display	2 x 7 0.1 in male pin header
20	U49	I2C Bus Switch	TI PCA9548ARGER
21	DS11 - DS13	Ethernet PHY Status LEDs	EPHY status LED, dual green
22	DS1 - DS4,Ds10,DS25 - DS27	UserGPIO LEDs	GPIO LEDs, green
23	SW2 – SW6	User Push buttons	E-Switch TL3301EP100QG
24	SW11	GPIO DIP Switch	C and K 4-pole, SDA05H1SBD
25	SW8	Rotary Switch	Panasonic EVQ-WK4001
26	J13, J14	GPIO SMA Connectors	Rosenberger 32K10K-400L5
27	SW15	Power On/Off Slide Switch SW15	C and K 1201M2S3AQE2
28	SW14	FPGA_PROG_B Pushbutton SW14 (Active-Low)	E-Switch TL3301EP100QG
29	SW13	Configuration Mode and Upper Linear Flash Address Switch (SW13)	5-pole C and K SDA05H1IBD
30	J22	HPC Connector J22	Samtec ASP_134486_01
31	J2	LPC Connector J2	Samtec ASP_134603_01
32	U55, U21, U103, U17, U56, U104, U105, U89, U106, U99, U71, U62, U17, U18, U33	Power Management (voltage regulators front side of board, controllers back side of board)	TI UCD9248PFC controllers in conjunction with various regulators
33	J46	XADC Header	2X10 0.1” male header
34	J60	2 x 7 2 mm shrouded JTAG cable connector	Molex 87832-1420
35	J39	2 x 5 shrouded PMBus connector	Assman HW10G-0202
36	J49	12V power input 2 x 3 connector	Molex 39-30-1060
37	SW7	CPU Reset Pushbutton	E-Switch TL3301EP100QG

Bibliography

[1] ITRS, International technology roadmap for semiconductor 2.0, Executive report (2015).

[2] Luca Benini and Micheli G.D., "Networks on Chips: A New SoC Paradigm", *IEEE Computer*, Vol. 35, No. 1, pp. 70–78, Jan. 2002.

[3] Tobias Bjerregaard and Shankar Mahadevan,"A Survey of Research and Practices of Network-on-Chip", *ACM Computing Surveys*, Vol. 38, 2006.

[4] N. E. Jerger and L. S. Peh, "On-chip Networks", *Morgan*, New York, 2009.

[5] Y. Wu, G. Min, M. Ould Khaoua, H. Yin, and L. Wang, "Analytical modelling of networks in multicomputer systems under bursty and batch arrival traffic", *The Journal of Super computing*, vol. 51, No. 2, pp. 115–130, 2010.

[6] Robert R. Schaller, "Moore's law: Past, present, and future", *IEEE Spectrum*, Vol. 34, No. 6, pp. 52–59, 1997.

[7] Tilera tile-gx processor. `http://www.tilera.com`.

[8] Dominic DiTomaso, Randy Morris, Avinash Karanth Kodi, Ashwini Sarathy and Ahmed Louri.,"Extending the Energy Efficiency and Performance With Channel Buffers, Crossbars, and Topology Analysis for Network-on-Chips", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 21, No. 11, pp. 2141–2154, 2013.

[9] Jongman Kim, Chrysostomos Nicopoulos, Dongkook Park, Vijaykrishnan Narayanan, Mazin S. Yousif and Chita R. Das., "A Gracefully Degrading and Energy-Efficient Modular Router Architecture for On-Chip Networks", *in Proceedings of the 33rd International Symposium on Computer Architecture*, 2006.

[10] Kanchan Manna, Santanu Chattopadhyay and Indranil Sen Gupta.,"Energy and Performance Evaluation of a Dimension Order Routing Algorithm for Mesh

of Tree based Network-on-Chip Architecture", *Annual IEEE India Conference (INDICON)*, pp. 1–4, 2010.

[11] D. Bertozzi and L. Benini, "Xpipes: A Network-on-Chip Architecture for Gigascale System on Chip", *in IEEE Circuits and Systems*, Vol. 4, No. 2, pp. 18–31, 2004.

[12] Jorg Henkel, Wayne Wolf and Srimat T. Chakradhar, "On-chip networks: a scalable, communication centric embedded system design paradigm", *in Proceedings of the 17th IEEE International Conference on VLSI Design*, pp. 845–852, 2004.

[13] S. Borkar, "Thousand core chips-A technology perspective", *in Proceedings of* 44th *ACM/IEEE Design Automation Conference*, pp. 746–749, 2007.

[14] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks", *in Proceedings of ACM/IEEE Design Automation Conference*, pp. 18–22, 2001.

[15] S. Bertozzi, A. Acquaviva, D. Bertozzi and A. Poggiali, "Supporting task migration in multi-processor systems-on-chip: a feasible study", *in Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, 2006.

[16] On network-on-chip comparison. `https://distrinet.cs.kuleuven.be/projects/martes/public/papers/Salminen_-_On_Network-on-chip_compar.pdf`.

[17] D. Fick, A. DeOrio, Jin Hu, V. Bertacco, D. Blaauw, and D Sylvester, "Vicis: A reliable network for unreliable silicon", 46th *ACM/IEEE Design Automation Conference*, pp. 812–817, 2009.

[18] William Dally and Brian Towles,"Principles and Practices of Interconnection Networks", *Morgan Kaufmann*, 2003.

[19] Y. Thonnart, P. Vivet, and F. Clermidy,"A fully-asynchronous low-power framework for gals NoC integration", *in Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 33–38, 2010.

[20] A. Jantsch and H. Tenhunen, "Networks on Chip", *Kluwer Academic Publisher*, 2003.

[21] B. N. K. Reddy, M. H. Vasantha, Y. B. Nithin Kumar and D. Sharma, "A fine grained position for modular core on NoC", *2015 International Conference on Computer, Communication and Control (IC4)*, Indore, pp. 1–4, 2015.

[22] F. Khalili and H. R. Zarandi, "A Reliability-Aware Multi-application Mapping Technique in Networks-on-Chip",*2013* 21st *Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pp. 478–485, 2013.

[23] Fatemeh Khalili and Hamid R. Zarandi, "A Fault-Aware Low-Energy Spare Core Allocation in Networks-on-Chip", *NORCHIP*, pp. 1–4, 2012.

[24] Shamshiri, S., Cheng, K.T., "Modeling yield, cost, and quality of a spare-enhanced multicore chip", *IEEE Transactions on Computers*, Vol. 60, No. 9, pp. 1246–1259, 2011.

[25] Leandro Fiorin and Mariagiovanna Sami,"Fault-Tolerant Network Interfaces for Networks-on-Chip", *IEEE Transactions on Dependable and Secure Computing*, Vol. 11, No. 1, 2014.

[26] A. Ferrante, S. Medardoni, and D. Bertozzi,"Network Interface Sharing Techniques for Area Optimized NoC Architectures", *in Proceedings of* 11th *EUROMICRO Conference Digital System Design Architectures, Methods and Tools (DSD)*, pp. 10–17, 2008.

[27] A. T. Tran and B. M. Baas, "Achieving High-Performance On-Chip Networks With Shared-Buffer Routers", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 22, No. 6, pp. 1391–1403, June 2014.

[28] E. Lakshmi Prasad, M.N. Giri Prasad, A.R. Reddy,"High-Speed Virtual Logic Network on Chip Router Architecture for Various Topologies", *Computers and Electrical Engineering*, Vol. 67, pp. 536–550, 2018.

[29] L.S. Peh, S. Keckler, and S. Vangal,"On-chip networks for multicore systems", *Multicore Processors and Systems: Integrated Circuits and Systems*, pp. 35–71, 2009.

[30] Pradip Kumar Sahu, Santanu Chattopadhyay,"A survey on application mapping strategies for Network-on-Chip design", *Journal of Systems Architecture*, Vol. 59, Issue 1, Pages 60–76, 2013.

[31] M. Ali, M. Welzl, S. Hessler, and S. Hellebrand,"An efficient fault tolerant mechanism to deal with permanent and transient failures in a network on chip", *International Journal of High Performance System Architecture*, Vol. 1, No. 2, pp. 113–123, 2007.

[32] D. Bertozzi, L. Benini, and G. De Micheli,"Error control schemes for on-chip communication links: The energy-reliability trade off", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 24, No. 6, pp. 818–831, 2005.

[33] Ali, M., Welzl, M., Zwicknagl, M. and Hellebrand,"Considerations for fault-tolerant Network on chips", *in Proceedings of* 17th *International Conference on Microelectronics*, pp. 177–181, 2005.

[34] Ali, M., Welzl, M. and Zwicknagl, M. "Networks on chips:scalable interconnects for future systems on chips", *in Proceedings of the* 3rd *IEEE International Conference on Circuits and Systems for Communications (ICCSC'06)*, 2006.

[35] Coskun Celk and Cuneyt F. Baziamacci,"Effect of Application Mapping on Network-on-Chip Performance", *in Proceedings of* 20th *Euromicro International Conference on Parallel, Distributed and Network-based Processing*, pp. 465–472, 2012.

[36] Bharath Phanibhushana, Kunal Ganeshpure and Sandip Kundu,"Task Model for on-Chip communication infrastructure design for multicore systems", *IEEE 29th International Conference on Computer Design(ICCD)*, pp. 360–365, 2011.

[37] Gianluca Palermo, Giovanni Mariani, Cristina Silvano, Riccardo Locatelli and Marcello Coppola, "Mapping and Topology Customization Approaches for Application-Specific STNoC Designs", *IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2007.

[38] D. Bertozzi, A. Jalabert, Srinivasan Murali, R. Tamhankar, S. Stergiou, L. Benini and G. De Micheli, "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 16, No. 2, 2005.

[39] Reetuparna Das, Rachata Ausavarungnirun, Onur Mutlu, Akhilesh Kumar, and Mani Azimi, "Application-to-core mapping policies to reduce memory interference in multi-core systems", 21st *International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2012.

[40] Chen Wu, Chenchen Deng, Leibo Liu, Jie Han, Jiqiang Chen, Shouyi Yin and Shaojun Wei,"A Multi-Objective Model Oriented Mapping Approach for NoC-based Computing Systems", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 28, No. 3, 2017.

[41] Omar Adil Mahdi, Ainuddin Wahid Abdul Wahab, Mohd Yamani Idna Idris, Ammar Abu Znaid, Yusor Rafid Bahar Al-Mayouf, and Suleman Khan,"WDARS: A Weighted Data Aggregation Routing Strategy with Minimum Link Cost in Event-Driven WSNs", *Journal of Sensors*, 2016.

[42] Bo Yang, "Towards Optimal Application Mapping for Energy-Efficient Many-Core Platforms", *Turku Centre for Computer Science*, No. 167, 2013.

[43] Chen Wu, Chenchen Deng, Leibo Liu, Jie Han, Jiqiang Chen, Shouyi Yin, and Shaojun Wei,"An Efficient Application Mapping Approach for the Co-Optimization of Reliability, Energy, and Performance in Reconfigurable NoC Architectures", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 34, No. 8, 2015.

[44] P. C. Chang, I. W. Wu, J. J. Shann, and C. P. Chung, "ETAHM: An energy-aware task allocation algorithm for heterogeneous multiprocessor", 45^{th} *ACM/IEEE Design Automation Conference*, pp. 776–779, 2008.

[45] Wooyoung Jang and David Z. Pan, "A3MAP: Architecture Aware Analytic Mapping for Networks on Chip", *The* 15^{th} *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2010.

[46] Ou He, Sheqin Dong, Wooyoung Jang, Jinian Bian, and David Z. Pan, "UNISM: Unified Scheduling and Mapping for General Networks on Chip", *IEEE Transactions on Very Large Scale Integration systems*, Vol. 20, No. 8, Aug 2012.

[47] Xinyu Wang, Tsan-Ming Choi, Xiaohang Yue, Mengji Zhang, and Wanyu Du, "An Effective Optimization Algorithm for Application Mapping in Network-on-Chip Designs", *IEEE Transactions on Industrial Electronics*, Vol. 67, No. 7, pp. 5798–5809, July 2020.

[48] Ahmed A. Morgan, Haytham Elmiligi, Mohamed Watheq El-Kharashi, and Fayez Gebali, "Unified multi-objective mapping and architecture customisation of networks-on-chip", *IET Computers & Digital Techniques*, Vol. 7, Iss. 6, pp. 282–293, 2013.

[49] Pradeep Kumar Sharma, Santosh Biswas, Pinaki Mitra, “Energy-efficient heuristic application mapping for 2-D mesh-based network-on-chip”, *Microprocessors and Microsystems*, Vol. 64, pp. 88–100, 2019.

[50] Srinivasan Murali and Micheli, G.D., “Bandwidth-Constrained Mapping of Cores onto NoC Architectures”, *in Proceedings of Design, Automation and Test in Europe Conference and Exhibition*, pp. 896–901, 2004.

[51] A. Habibi, M. Arjomand, H. Sarbazi-Azad, “Multicast-Aware Mapping Algorithm for On-chip Networks”, *in 2011* 19th *International Euromicro Conference on Parallel, Distributed and Network-Based Processing*, pp. 455–462, 2011.

[52] B. Reddy, Sireesha, “An Energy-Efficient Core Mapping Algorithm on Network on Chip (NoC)”, *in* 22nd *International Symposium VLSI Design and Test(VDAT)*, 2018.

[53] Chen Wu et al., “An Efficient Application Mapping Approach for the Co-Optimization of Reliability, Energy, and Performance in Reconfigurable NoC Architectures”, *in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 34, No. 8, pp. 1264–1277, Aug. 2015.

[54] Sahu, P. K., Shah T., Manna, K. and Chattopadhyay S., “Application Mapping Onto Mesh-Based Network-on-Chip Using Discrete Particle Swarm Optimization”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 22, No. 2, pp. 300–312, 2014.

[55] Upadhyay, M., Shah, M., Bhanu, P. V.,Soumya, J. and Cenkeramaddi,L. R. “Multi-application Based Network-on-Chip Design for Mesh-of-Tree Topology Using Global Mapping and Reconfigurable Architecture”, *2019* 32nd *International Conference on VLSI Design and 2019* 18th *International Conference on Embedded Systems (VLSID),India*, pp. 527–528, 2019.

[56] B. Li, X. Wang, A. K. Singh and T. Mak, “On Runtime Communication and Thermal-Aware Application Mapping and Defragmentation in 3D NoC Systems”, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 30, No. 12, pp. 2775–2789, 2019.

[57] G. Li, J. Wu and G. Ma, “Mapping of irregular IP onto NoC architecture with optimal energy consumption”, *in Tsinghua Science and Technology*, Vol. 12, No. S1, pp. 146–149, July 2007.

[58] W. Liu et al., "Thermal-Aware Task Mapping on Dynamically Reconfigurable Network-on-Chip Based Multiprocessor System-on-Chip", *IEEE Transactions on Computers*, Vol. 67, No. 12, pp. 1818–1834, 2018.

[59] G. Jiang, Z. Li, F. Wang and S. Wei, "Mapping of Embedded Applications on Hybrid Networks-on-Chip with Multiple Switching Mechanisms", *IEEE Embedded Systems Letters*, Vol. 7, No. 2, pp. 59–62, June 2015.

[60] L. Liu et al., "A Flexible Energy- and Reliability-Aware Application Mapping for NoC-Based Reconfigurable Architectures", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 23, No. 11, pp. 2566–2580, 2015.

[61] S. Khan, S. Anjum, U. A. Gulzari, T. Umer and B. Kim, "Bandwidth-Constrained Multi-Objective Segmented Brute-Force Algorithm for Efficient Mapping of Embedded Applications on NoC Architecture", *in IEEE Access*, Vol. 6, pp. 11242–11254, 2018.

[62] Fatemeh Khalili and Hamid R.Zarandi, "A fault tolerant core mapping technique in networks-on-chip", *IET Computers & Digital Techniques*, Vol. 7, Iss.6, pp. 238–245, 2013.

[63] Chen-Ling Chou and Radu Marculescu, "FARM: Fault Aware Resource Management in NoC based Multiprocessor Platforms", *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, 2011.

[64] P. V. Bhanu, R. Govindan, P. Kattamuri, J. Soumya and L. R. Cenkeramaddi, "Flexible Spare Core Placement in Torus Topology Based NoCs and Its Validation on an FPGA", *in IEEE Access*, Vol. 9, pp. 45935–45954, 2021.

[65] Naresh Kumar Reddy Beechu, Vasantha Moodabettu Harishchandra, and Nithin Kumar Yernad Balachandra., "High-performance and energy-efficient fault-tolerance core mapping in NoC", *Sustainable Computing: Informatics and Systems*, Vol 16, pp. 1-10, 2017.

[66] N. Genko, D. Atienza, G. De Micheli, J.M. Mendias, R. Hermida, and F. Catthoor,"A complete network-on-chip emulation framework", *in Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2005.

[67] Y.E. Krasteva, F. Criado, E. de la Torre, and T. Riesgo.,"A Fast Emulation-Based NoC Prototyping Framework", *in Proceedings International Conference on Reconfigurable Computing and FPGAs*, 2008.

[68] G. Schelle and D. Grunwald.,“Onchip Interconnect Exploration for Multicore Processors Utilizing FPGAs”, *in* 2^{nd} *Workshop on Architecture Research using FPGA Platforms*, 2006.

[69] P.T. Wolkotte, P.K.F. Holzenspies, and G.J.M. Smit.,“Fast, Accurate and Detailed NoC Simulations”, *in First Proceedings International Symposium on Networks-on-Chip (NOCS’07)*, pp. 323–332, 2007.

[70] Naresh Kumar Reddy Becchu, Vasantha Moodabettu Harishchandra, Nithin Kumar Yernad Balachandra, “System level fault-tolerance core mapping and FPGA-based verification of NoC”, *Microelectronics Journal*, Vol. 70, pp. 16–26, 2017.

[71] L. Zhang, J. Yang, C. Xue, Y. Ma and S. Cao, “A two-stage variation-aware task mapping scheme for fault-tolerant multi-core Network-on-Chips”, *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, Baltimore, pp. 1–4, 2017.

[72] Z. Wang, A. Littarru, E. I. Ugwu, S. Kanwal and A. Chattopadhyay, “Reliable Many-Core System-on-Chip Design Using K-Node Fault Tolerant Graphs”, *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, USA, pp. 619–624, 2016.

[73] C. Bonney, P. Campos, N. Dahir and G. Tempesti, “Fault tolerant task mapping on many-core arrays”, *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, Athens, pp. 1–8, 2016.

[74] Navonil Chatterjee, Suraj Paul, Santanu Chattopadhyay, “Task mapping and scheduling for network-on-chip based multi-core platform with transient faults”, *Journal of Systems Architecture*, Vol. 83, Pages 34–56, 2018.

[75] F. Khalili and H. R. Zarandi, “A Fault-Tolerant Low-Energy Multi-Application Mapping onto NoC-based Multiprocessors”, *2012 IEEE* 15^{th} *International Conference on Computational Science and Engineering*, Cyprus, pp. 421–428, 2012.

[76] S. Tosun, V. B. Ajabshir, O. Mercanoglu and O. Ozturk, “Fault-Tolerant Topology Generation Method for Application-Specific Network-on-Chips”, *in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 34, No. 9, pp. 1495–1508, Sept. 2015.

[77] S. Chen, M. Ge, Z. Li, J. Huang, Q. Xu and F. Wu, "Generalized Fault-Tolerance Topology Generation for Application-Specific Network-on-Chips", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 39, No. 6, pp. 1191–1204, June 2020.

[78] Naresh Kumar Reddy Beechu, Vasantha Moodabettu Harishchandra, Nithin Kumar Yernad Balachandra, "An energy-efficient fault-aware core mapping in mesh-based network on chip systems", *Journal of Network and Computer Applications*, Vol. 105, pp. 79–87, 2017.

[79] Vincenzo Catania, Andrea Mineo, Salvatore Monteleone, Maurizio Palesi, and Davide Patti., "Noxim: An open, extensible and cycle-accurate network on chip simulator", *in Proceedings of the 2015 IEEE 26th International Conference on Application-specific Systems, Architectures and Processors (ASAP15)*, pp. 162–163, 2015.

[80] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh and A. Gupta, "The SPLASH-2 programs: characterization and methodological considerations", *in Proceedings of 22nd Annual International Symposium on Computer Architecture*, pp. 24–36, 1995.

[81] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications", *in Proceedings of the 17th International conference on Parallel architectures and compilation techniques*, pp. 72–81, 2008.

[82] N. Binkert, et al., "The gem5 simulator", *ACM SIGARCH Computer Architecture News*, Vol. 39, No. 2, pp. 1-–7, 2011.

[83] Haoyuan Ying, Thomas Hollstein and Klaus Hofmann,"A Hardware/Software Co-Design Reconfigurable Network-on-Chip FPGA Emulation Method", 9th*International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, pp. 1–8, 2014.

[84] Xilinx kintex-7 fpga kc705 evaluation kit. `http://www.xilinx.com/products/boards-and-kits/ek-k7-kc705-g.html`, 2015.

[85] Failure rates, mtbfs, and all that. `http://www.mathpages.com/home/kmath498/kmath498.htm`.

[86] Reliability and mtbf overview. `http://www.vicorpower.com/documents/quality/Rel_MTBF.pdf`, 2015.

[87] David Rhodes and Robert Dick. Task graphs for free (tgff). `http://ziyang.eecs.umich.edu/~dickrp/tgff/`.

[88] Y. Xue and P. Bogdan, "Scalable and realistic benchmark synthesis for efficient NoC performance evaluation: A complex network analysis approach", *2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), USA*, pp. 1–10, 2016.

[89] Partha Pratim Pande S. P. Cristinel Ababei., `http://networkonchip.wordpress.com/2011/02/22/simulators/`, 2011.

[90] A. B. Achballah and S. B. Saoud., "A survey of network-on-chip tools",*International Journal of Advanced Computer Science & Applications*, Vol. 4, No. 9, 2013.

[91] davidepatti. Noxim – the noc simulator. `https://github.com/davidepatti/noxim`, 2015.

Publications

Papers Published/Accepted in Journals:

1. **Authors: Aruru Sai Kumar** and T.V.K. Hanumantha Rao,
   Title: Scalable benchmark synthesis for performance evaluation of NoC core mapping
   Journal: Microprocessors and Microsystems, 2020. **(Elsevier, SCI Indexed, Impact Factor= 1.525)**.**(Published)**
2. **Authors: Aruru Sai Kumar** and T.V.K. Hanumantha Rao,
   Title: An Adaptive Core Mapping Algorithm on NoC for Future Heterogeneous System-on-Chip
   Journal: Computers and Electrical Engineering, 2021. **(Elsevier, SCI Indexed, Impact Factor=3.818) (Published)**
3. **Authors: Aruru Sai Kumar** and T.V.K. Hanumantha Rao,
   Title: Performance and Communication Energy constrained Embedded Benchmark for Fault Tolerant Core Mapping onto NoC architectures
   Journal: International Journal of Ad Hoc and Ubiquitous Computing, 2021. **(Inderscience Publishers, SCI Indexed, Impact Factor=0.654) (Accepted)**
4. **Authors: Aruru Sai Kumar** and T.V.K. Hanumantha Rao,
   Title: Performance Assessment of Adaptive Core Mapping for NoC-based architectures
   Journal: International Journal of Embedded Systems, 2021. **(Inderscience Publishers, ESCI and Scopus Indexed) (Accepted)**

Papers Communicated in Journals:

5. **Authors: Aruru Sai Kumar** and T.V.K. Hanumantha Rao,
   Title: An Efficient Real -Time Embedded Application Mapping For NoC Based Multiprocessor System on Chip
   Journal: Wireless Personal Communications, 2021.**(Springer, SCI Indexed) (Under Review)**

Papers Accepted/Published in Conferences:

6. **Authors: Aruru Sai Kumar** and T.V.K. Hanumantha Rao,
   Title: An efficient low latency router architecture for mesh based NoC
   Conference: International Conference on Communications, Signal Processing and VLSI (**IC2SV-2019**), NIT Warangal, India, pp. 241-248, 2019.**(Springer-Published)**

7. **Authors: Aruru Sai Kumar** and T.V.K. Hanumantha Rao,
   Title: Efficient Core Mapping on Customization of NoC Platforms
   Conference: IEEE International Symposium on Smart Electronic Systems (**iSES-2019**), NIT Rourkela, India, pp.57-62, 2019. **(IEEE-Published)**

8. **Authors: Aruru Sai Kumar**, T.V.K. Hanumantha Rao and B.N. Kumar Reddy,
   Title: Exact Formulas for Fault Aware Core Mapping on NoC Reliability
   Conference: IEEE 17th India Council International Conference (**INDICON-2020**), New Delhi, India, pp. 1-5, 2020. **(IEEE-Published)**