EFFICIENT DUAL PARTITION MULTICAST ROUTING WITH LOAD BALANCING FOR ON CHIP NETWORKS

by

CHINMAYA KUMAR PADHI 2011103555
VIGNESH T 2011103038
KAVIN M 2011103051

A project report submitted to the

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANNA UNIVERSITY, CHENNAI – 25

MAY 2015

BONAFIDE CERTIFICATE

Certified that this project report titled **EFFICIENT DUAL PARTITION MULTICAST ROUTING WITH LOAD BALANCING FOR ON CHIP NETWORKS** is the *bonafide* work of **CHINMAYA KUMAR PADHI (2011103555)**, **VIGNESH T (2011103038)** and **KAVIN M (2011103051)** who carried out the project work under my supervision, for the fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering. Certified further that to the best of my knowledge, the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or an award was conferred on an earlier occasion on these or any other candidates.

Place: Chennai
Date:

Dr.S.Sudha
Sr.Assistant Professor
Department of Computer Science and Engineering
Anna University, Chennai – 25

COUNTERSIGNED

Head of the Department,
Department of Computer Science and Engineering,
Anna University Chennai,
Chennai – 600025

ACKNOWLEDGEMENT

We owe our thanks to **Dr.S.Sudha**, our project guide, for her continuous and valuable guidance, support and encouragement during every stage of the project leading to its successful completion.

We express our sincere gratitude to **Dr.A.P.Shanthi, Dr.Rajeswari Sridhar, Dr.D.Shiloah Elizabeth**, the members of project review committee for their valuable suggestions and guidance.

We would like to convey our gratitude to **Dr.D.Manjula**, Head of the Department, Department of Computer Science and Engineering, for providing us the opportunity to carry out this project.

We like to thank all our friends and family members for providing moral support to us during the entire course of project.

Chinmaya Kumar Padhi **Vignesh T** **Kavin M**

ABSTRACT

Network on Chip (NoC) is making a new era in interconnection networks (System on Chip), and is depicting a lucid future with its more effective architectural models and adaptive routing algorithms. System on Chip ascribes its basic fundamental building blocks on the efficient NoC switches, which connects SoC's memories and processing elements. NoC, when built to its best level can result in an infrastructure that is robust, scalable, high-performance and minimizing area and power consumption. Increasingly NoC's performance will in turn augment SoC's performance considerably.

Without efficient multicasting support, traditional unicasting on-chip networks will be less efficient in tackling such multicast communication. In this project, we propose Dual Partition Multicast (DPM) Routing algorithm with load balancing to increase the throughput, minimize power consumption and balance network resource utilization. Specifically, DPM scheme adaptively makes routing decisions based on the network load-balance level characterized by the distribution of the multicasting destinations. The algorithm is also extended to stand against faulty links and aid uniform traffic distribution.

ABSTRACT

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

DPM	Dual Partition Multicast
NoC	Network on Chip
SoC	System on Chip
VC	Virtual Channels
IC	Input Channel Controller
OC	Output Channel Controller
VCA	Virtual Channel Allocator

CHAPTER 1

INTRODUCTION

1.1 OBJECTIVE

Without efficient multicasting support, traditional unicasting on-chip networks will be low efficiency in tackling such multicast communication. In this project, we propose Dual Partition Multicast (DPM) Routing with load balancing to increase the throughput, minimize power consumption and balance network resource utilization.

Specifically, DPM scheme adaptively makes routing decisions based on the network load-balance level characterized by the distribution of the multicasting destinations. The algorithm is also extended to stand against faulty links and aid uniform traffic distribution.

1.2 DEFINITIONS

The development in silicon technology is moving at such a rapid pace that before the end of this decade we will witness chips capable of accommodating billions of transistors. The advancements will surely help chip designers and engineers to build complex SoC's hence bringing a whole new revolution in chip technology. This revolution is, nevertheless, barricaded by buses - traditional interconnects of SoC's. Studies have revealed that buses cannot scale beyond a certain number of partners on-chip. Due to this very reason buses have become a bottleneck in the advancement and growth of future SoC's.

After realising the inefficiency of traditional buses in SoC's, in conjunction with so many other factors, the designers of SoC's have come to a cross-road where they meet the computer architecture designers who are always interested in finding dynamic and scalable architectures for building microprocessors. The scalability and wide success of the Internet has attracted the attention of computer architecture as well as SoC's designers and influenced them to borrow the idea of using packet based switching networks for the design of future SoC communication infrastructure.

It is an understood fact that the actual reason behind success of the Internet and its scalabililty lies in a well defined protocol stack; the idea was to decouple communication from computation. Packet switched communication not only provides high scalability, but also facilitates reuse of the communication architecture. The two major problems faced by SoC designers are re-usability and scalability, therefore will be addressed by the adoption of packet switched communication infrastructure for SoC interconnects.

Also, from a business point of view, it is important to reduce the design time by adopting reuse not only at the computational level but also reuse of the communication structure. This will in turn lower the time to market new products with ease. Keeping in view this idea of Internet, many researchers have proposed communication architectures based upon packets switched on chip networks for connecting components in the future SoC's.

1.3 CHALLENGES

The rudimentary state of progress of the Networks-on-Chip (NoC) paradigm is a great boon which in turn triggers many to indulge in the process of innovation. The optimal Networks-on-Chip (NoC) paradigm depends on multiple parameters like topology, routing, load distribution, fault tolerance, additional and apt usage of virtual channels, buffer allocation, etc. Concentrating on each of these individual parameters and making them to work at its best will drastically improve the performance far from all the existing paradigms. For this, we design Dual Partition Multicast (DPM) [1] Routing which handles efficient routing, load balancing and fault tolerance. After making prior designing and tuning of all these parameters at its individual level, we eventually establish our improved performance over traditional XY Routing algorithms through our simulation results.

1.4 PROBLEM STATEMENT

As the number of cores integrated onto a single chip increases, power dissipation and network latency become ever-increasingly stringent. On-chip network provides an efficient and scalable interconnection paradigm for chip multiprocessors (CMPs), wherein one-to-many (multicast) communication is universal for such platforms.In this project, a study is made at architectural design, switching technologies, flow control and routing algorithm to overcome the problem of uneven load distribution and extend our study to fault tolerance. At architectural level, Mesh NoC is chosen to maximize the best access to every recess node which prevents deadlock and channel dependencies.

1.5 OVERVIEW OF THE PROJECT

We propose a new approach ”Dual Partition Multicast Routing” which optimally routes the packets by balancing the load at each tile. The algorithm even has the flexibility to route in case of faulty links and nodes by calculating new alternate paths. Using this concept, we maintain a uniform load distribution. Our routing algorithm routes via the shortest path in case of no failures. If one or more nodes or links fail, then it can still route to destinations via any path available while maintaining uniform load distribution.

The NoC methodology will likely be the best solution to counter the increasing complexity of future SoC’s. From the above discussion it can also be concluded that future SoC’s will be platform based because of short-time-to-market constraints. Development of NoC’s will be a huge effort as it involves reuse at all levels; reuse of architecture, hardware and software. Also, it includes reuse of different languages, methods, tools and practices during development. Although the potential of NoC’s is tremendous, it would be rather unlikely to fulfill all its promises before the development of some of its key components like a reliable NoC architecture, assurance of quality of service, and a viable NoC software model. Clearly, it can be concluded that there is a lot of need and scope for research in this area.

The NoCs have the following advantages:

- Links can be shared
- High level of parallelism
- High Efficiency
- Scalability
- Modularity
- High Level of Abstraction

CHAPTER 2

LITERATURE SURVEY

The non-adaptive routing routes only through the shortest path from source to destination. Te best example of this routing is XY-routing. This algorithm routes first via the x-axis followed by y-axis. But in the case of partially-adaptive routing and fully-adaptive routing, the virtual channels are used to make it adaptive. But in this non-adaptive, partially-adaptive and fully-adaptive there is no assurance that the packets are forwarded in case of any faulty links.

Ming Zong et al.[2] proposed a new distributed multicast routing algorithm using the shortest path to each destination, while focusing on minimizing the link usage. By building a tree of fully-used shared path and appending other destinations to the tree, they obtained the minimal link usage multicast tree. However, the major drawback is that tree based routing incurs high congestion in wormhole network and decreases the throughput.

Lei Wang et al.[4] proposed Recursive Partitioning Multicast (RPM) routing for NOCs.RPM allows routers to intermediate replication nodes based on global distribution of destination nodes. Simulation results showed that RPM saved 25% of link power and 33% of link utilization. But RPM failed to improve the performance significantly as it gave higher priority to flits routed towards North or South port restricting the direct West or East routing.

Jianhua Li et al.[1] proposed Dual partitioning multicasting scheme which adaptively makes routing decisions based on the network load-balance level as well as the link sharing patterns. However, implementing link sharing pattern algorithm degrades the performance.Further, it does not provide fault tolerance and load balancing. In this project, we follow a new mechanism which balances the load without compromising the performance.

CHAPTER 3

DESIGN

3.1 SIMULATOR DESIGN

NIRGAM is a discrete event, cycle accurate simulator targeted at Network on Chip (NoC) research. It provides substantial support to experiment with NoC design in terms of routing algorithms and applications on various topologies. The simulator is written in systemC.It allows to experiment with various options available at every stage of NoC design: topology, switching technique, virtual channels, buffer parameters, routing mechanism and applications.

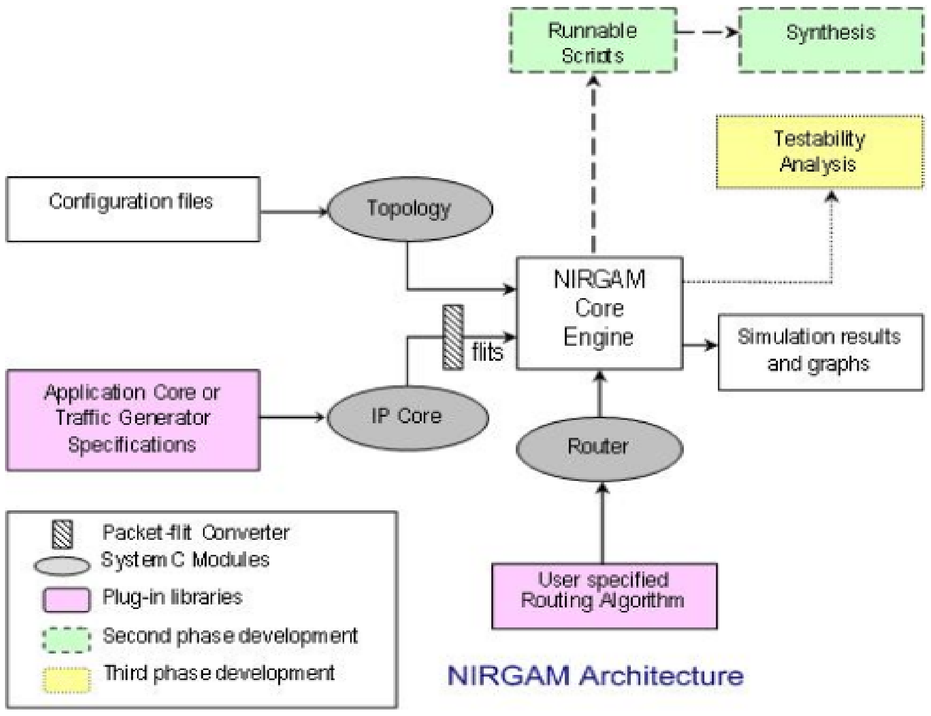


Figure 3.1 NIRGAM Architecture

3.2 HIGH-LEVEL BLOCK DIAGRAM

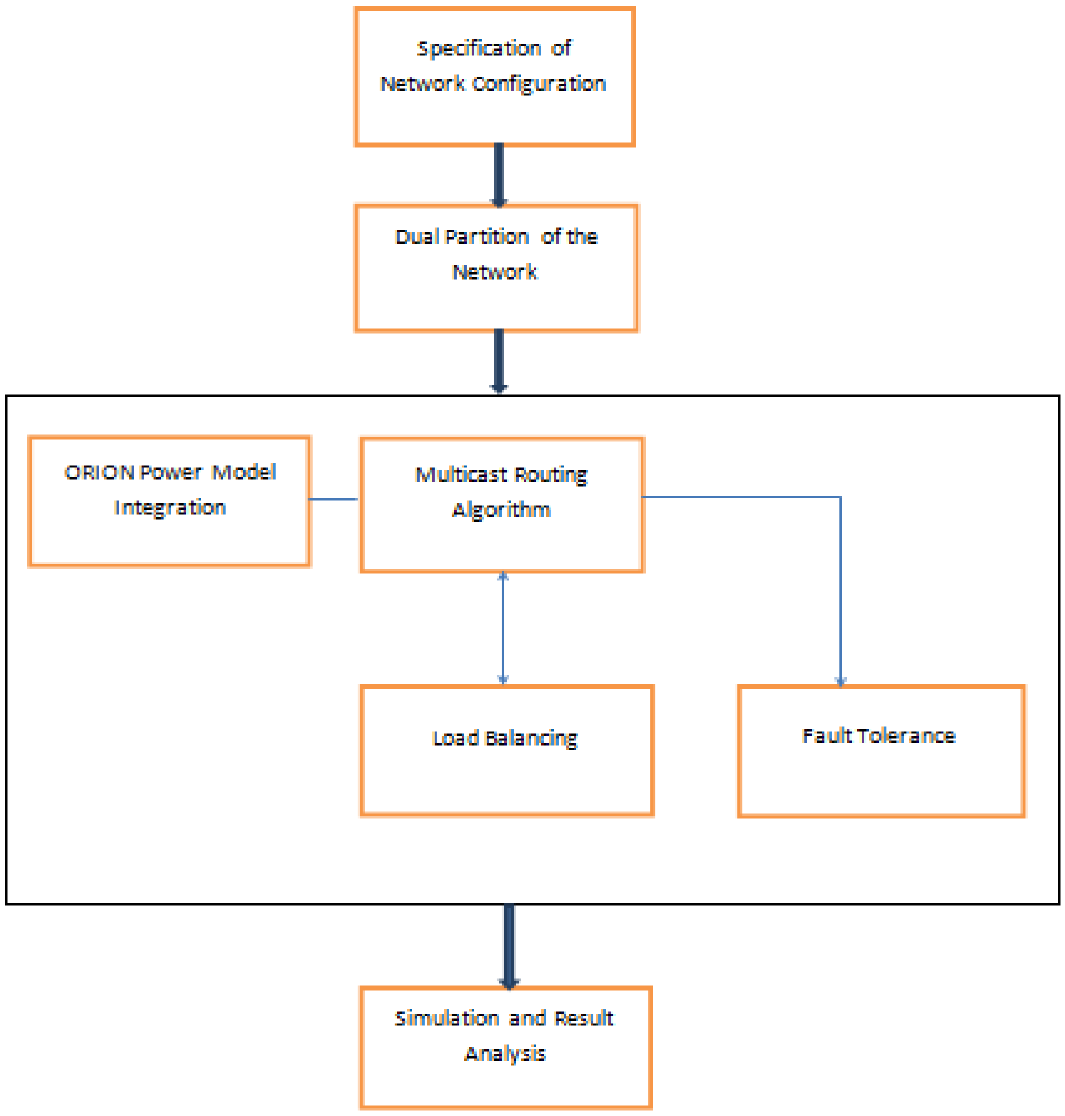

Specification of Network Configuration
Dual Partition of the Network
ORION Power Model Integration
Multicast Routing Algorithm
Load Balancing
Fault Tolerance
Simulation and Result Analysis

Figure 3.2 Block Diagram

The Network Configuration of NoC is specified in the files nirgam.config and application.config. The configuration parameters in nirgam.config are number of rows and columns, clock cycle, number of packets to be sent across the network, failure links, topology, routing algorithm. The mesh topology for a 3x4 matrix is shown below.

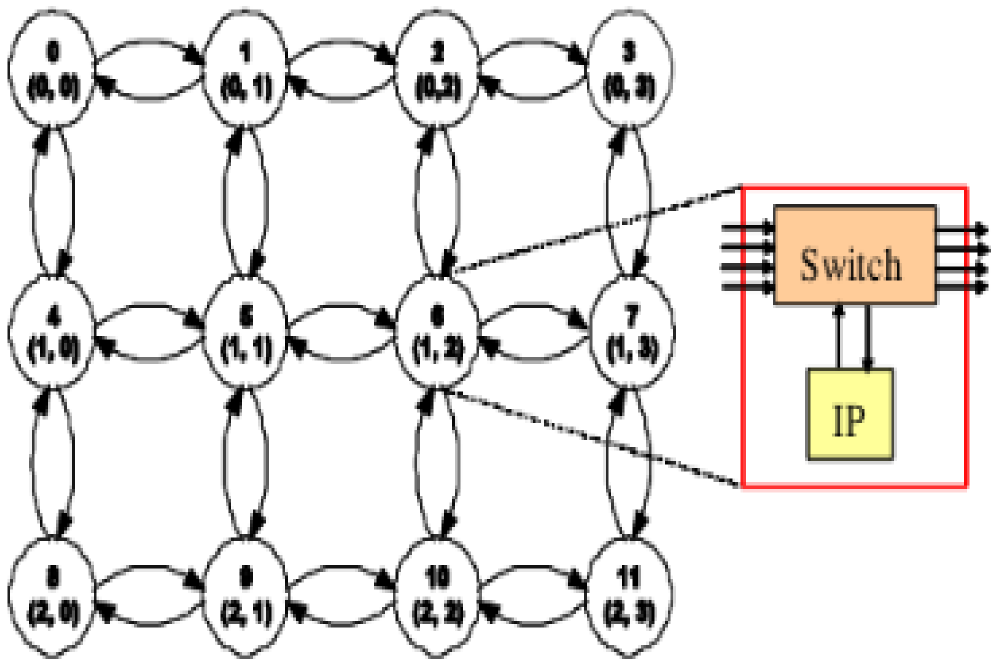


Figure 3.3 3x4 Mesh Topology

The source and destination tiles as well as the application that is to be attached are specified in application.config. Each tile can be attached with a specific application to send and receive packets in the network.

3.3 DUAL PARTITION OF NETWORK

Based on the position of source, we divide the network into eight different regions - North, South, East, West, North-East, North-West, South-East and South-West. We use North-Last and West-Last routing algorithms to route the packets from the source tile to all the destination tiles. The packets in the network is divided into flow control digits (flits). These flits are routed one after another from the source to all the multicast destinations.

The first flit of the packet is the header flit. The header flit contains routing information like the destination address. All flits of a packet take the path to the receiver that was established by the header flit.

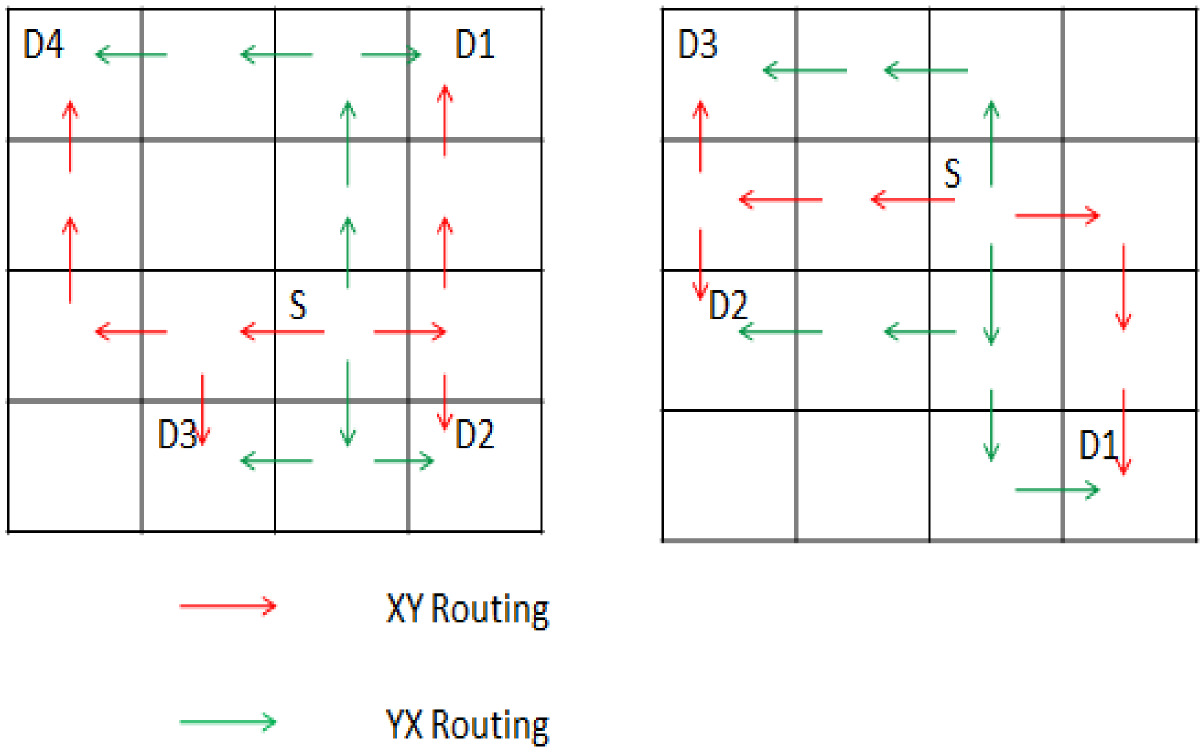


Figure 3.4 Dual Partitioned Network

3.4 LOAD BALANCING

Routing to multicast destinations can cause congestion which leads to network imbalance. We use 8-bit counter to indicate the number of packets entering a particular sub-network. Every time a packet is sent, increment or decrement the counter till it reaches a particular value, we use either XY-routing or YX-routing to route the packets to the destinations. Therefore, the packets are not routed in the same direction the entire time and gets routed in other possible directions thus balancing the load across the network.

Methodology

- A counter is used to indicate the number of packets that are injected into a particular sub-network.
- Initially the counter to set to some value k .
- When the packet is injected into the sub-network N0 and if the value of the counter is lesser than 2k, increase counter by 1.
- When the packet is injected into the sub-network N1 and if the value of the counter is larger than 0, decrease counter by 1.
- If the counter value is within the range (0,2k) then, the packets are routed normally.
- If the counter value is less than 0, packet is injected into N0 , otherwise the packet is injected into N1.

3.5 FAULT TOLERANCE

Routing the packets through the XY-routing algorithm does not work whenever there is node failure or link failure in the routing path. To make the DPM routing fault tolerant, we monitor the failed links and nodes and route in any other possible direction. The failed links are specified in the configuration file and are stored in a fail matrix. On routing we consider the links in the fail matrix and route the packets through other possible directions.

Methodology

1. A n x 2 matrix called fail_matrix is constructed to store the tile ids that are connected by failed links, where n is the number of failed links

2. The current tile id is looked up into this table

3. If the current tile id is not present in the table, routing happens normally

4. If tile id is present in the table, then the id of neighboring tile (next_id) in the determined routing direction is looked up.

5. If next_id is also present in the table in the same row as the current tile_id , the routing towards the specified direction is invalid and should be routed in any other possible direction.

3.6 ORION POWER MODEL

On-chip networks consume a significant fraction of power in many-core chips. It is a suite of dynamic and leakage power models developed for on-chip networks to calculate the power consumed by various components. On-chip networks consume a significant fraction of power in many-core chips. It is a suite of dynamic and leakage power models developed for on-chip networks to calculate the power consumed by various components. The purpose of integrating this model is to show the power efficiency of multicast over unicast routing. It calculates the power consumed by each tile and each link in the Network. The total power consumed by the network in unicast is compared with that of multicast.

Tile Power Calculation

The power consumed by each tile depends upon the following factors : (per clock cycle)

1. Avg. num of ip buffer writes(buf_rl)
2. Avg. num of ip buffer reads(buf_wl)
3. Avg. num of flits passing through switch(c_tl)
4. Avg. num of arbitrations(n_arb)
5. Avg. num of reads during VC allocation(nvc_read)
6. Avg. num of writes during VC allocation (nvc_write)

Power Calculation

Link power depends on the following factors:

1. No. of times the link traversed (load)
2. Clock frequency ()
3. Link Length (link_len)
4. Data Width (Flitsize 8)
5. Operating Voltage (Vdd)

The following formulae are used to compute the average power consumption.

$$\mathrm{P}_{tile} = v * (P_{buf\_wl} + P_{buf\_rl} + P_{c\_tl} + P_{n\_arb} + P_{nvc\_read} + P_{nvc\_write}) \ldots (1)$$

This equation determines the power consumed by each tile. Depending on the number of links, buffers and virtual channels of tile used, the power consumed by each tile varies.

$$\mathrm{P}_{leakage} = V_{dd}^2 * link\_len * data\_width \qquad ...(2)$$

Equation 2 refers to the power expended due to losses in the form of heat, overload etc.

$$\mathrm{P}_{dynamic} = \frac{1}{2}(load * P_{leakage} * v) \qquad ...(3)$$

Equation 3 calculates the power absorbed by each link when packets are routed across them.

$$\mathrm{P}_{link} = P_{dynamic} + P_{leakage} \qquad ...(4)$$

Equation 4 gives the total power consumed by each link. It includes the power loss incurred in a link in addition to power consumed by the link.

$$\mathrm{P}_{avg} = n * P_{link} + \sum_1^m P_{tile} \qquad ...(5)$$

The above equation gives the overall power consumed by the network, that is, the power exhausted by both the links as well as tiles in the whole run.

where

n - number of links traversed

$v - numberofclockcycles$

m - number of tiles in network

CHAPTER 4

IMPLEMENTATION

4.1 TOOLS DESCRIPTION

NIRGAM is a discrete event, cycle accurate simulator targeted at Network on Chip (NoC) research. It provides substantial support to experiment with NoC design in terms of routing algorithms and applications on various topologies. The simulator is written in systemC.

NIRGAM allows to experiment with various options available at every stage og NoC design like

- topology,
- switching technique,
- virtual channels,
- buffer parameters,
- routing mechanisms and
- applications.

Besides built-in capabilities, it can be easily extended to include new applications and routing algorithms. The simulator can output performance metrics (power, latency, traffic load and throughput) for a given set of choices.

We can configure the following NoC parameters:

1. Topology
2. Switching Mechanism
3. Virtual channels
4. Buffer
5. Clock Frequency
6. Routing Algorithms
7. Applications
8. Performance Analysis

4.1.1 Code Structure

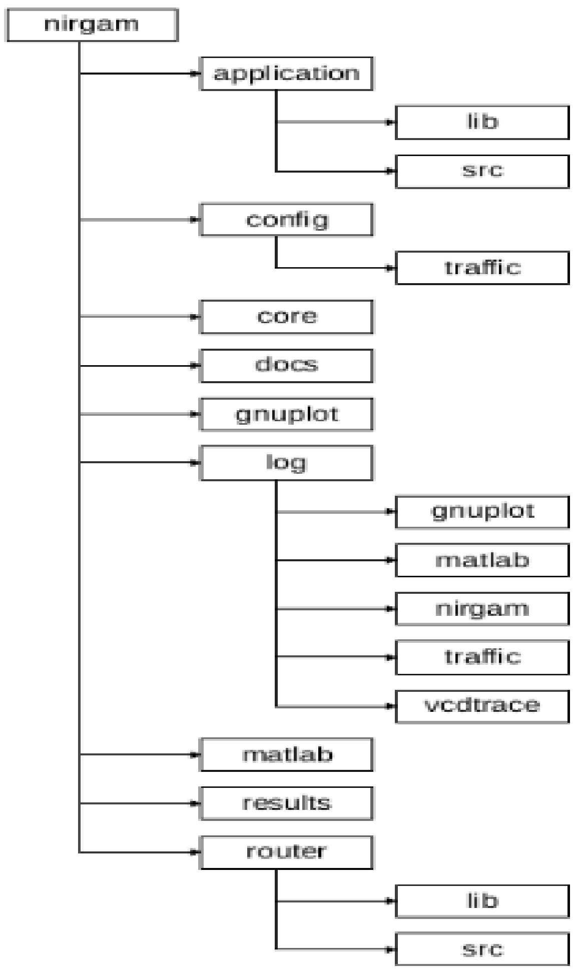


Figure 4.1 NIRGAM code structure

Figure 4.1 shows the directory structure of the simulator code. The top level directory 'nirgam' contains code subdirectories, makefile and the executable.
A brief description of the contents of each directory follows.

1. application

   This folder contains the source code of applications and application libraries that can be plugged in to an ipcore.
2. config

   This folder contains all the configuration files for the simulator as well as traffic. Users can experiment with different design choices by changing parameters in these files.
3. core

   This folder is the core engine that implements NoC. It contains code for implementation of NoC topology and architecture.
4. docs

   This folder contains the documentation.
5. gnuplot

   This folder contains gnuplot script for generating graphs.
6. log

   This folder contains log files. The subfolders are as follows:

   gnuplot - contains input data files for gnuplot script.
   matlab - contains input data files for matlab script.
   nirgam - contains event log file for the complete simulation.
   traffic - contains log of traffic generated by each tile.
   vcdtrace - contains any vcdtrace generated using systemC.
7. matlab

   This folder contains matlab script to generate graphs.

8. results

   This folder contains result statistics and graphs for the simulations.

9. router

   This folder contains the source code of routing algorithms and respective libraries that can be attached to the tiles in the network.

4.2 IMPLEMENTATION PLATFORM

4.2.1 Software Requirements

- Platform

  ubuntu 12.04 LTS

  gcc 4.6.3

- Simulator

  Nirgam 2.1 using systemC 2.3.0

4.2.2 Hardware Requirements

A PC (with specified software) for simulation.

4.3 SIMULATION USING NIRGAM

NIRGAM core engine implements Network-on-Chip. This section describes the NoC architecture as implemented in NIRGAM. NoC is modelled as a 2-dimensional network of tiles and is represented by module NoC. Each tile is represented by module NWTile.The major components of each tile and the modules that implement them are as follows:

- **Input Channel Controller (IC):** represented by module InputChannel

  Each tile consists of one IC for each neighbor, and an IC for ipcore. For example, a tile having 4 neighbors (one in each of the directions North(N), South(S), East(E) and West(W)) will have 5 ICs: N IC, S IC, E IC, W IC and core IC. Each IC consists of one or more virtual channels (VCs). Each VC consists of a fifo buffer.
- **Controller:** represented by module Controller

  Each tile consists of one Controller which implements router to service routing requests from all ICs.
- **Virtual Channel Allocator(VCA):** represented by module VCAllocator

  Each tile consists of one VCA that services requests for virtual channel allocation from all ICs.
- **Output Channel Allocator(VCA):** represented by module OutputChannel

  Each tile consists of one OC for each neighbor, and an OC for ipcore. Each OC consists of an array of registers r in[], one for each input port. Each OC consists of an array of registers r vc[], one for each virtual channel in IC of neighbor tile.
- **ipcore:** represented by module ipcore

  Each tile consists of an ipcore (IP element) to which an application or traffic generator can be attached.

4.4 IMPLEMENTING ROUTING ALGORITHM

The complexity of a router can vary depending upon its intelligence and route logic.You may wish to hack Controller and InputChannel modules to implement any routing algorithm. However a routing algorithm which is a function of source and destination address can be easily implemented as a class derived from class router. Next we will explain steps to add such a router. As an example let us consider that we need to add a routing algorithm identified by name R and library name myrouter.so.

1. Run make clean.
2. Create header file myrouter.h in $NIRGAM/router/src. A template file is included in source code for your reference.
3. Create implementation file myrouter.cpp in $NIRGAM/router/src. A template file is included in source code for your reference.
4. Edit calc_next() function in myrouter.cpp to insert your route logic. Following variables are passed as parameters to the router from Controller and may be used in this function:

   ip dir: input direction from which flit entered the tile. Possible values: N, S, E, W, C.

   source id: tile id of source tile.

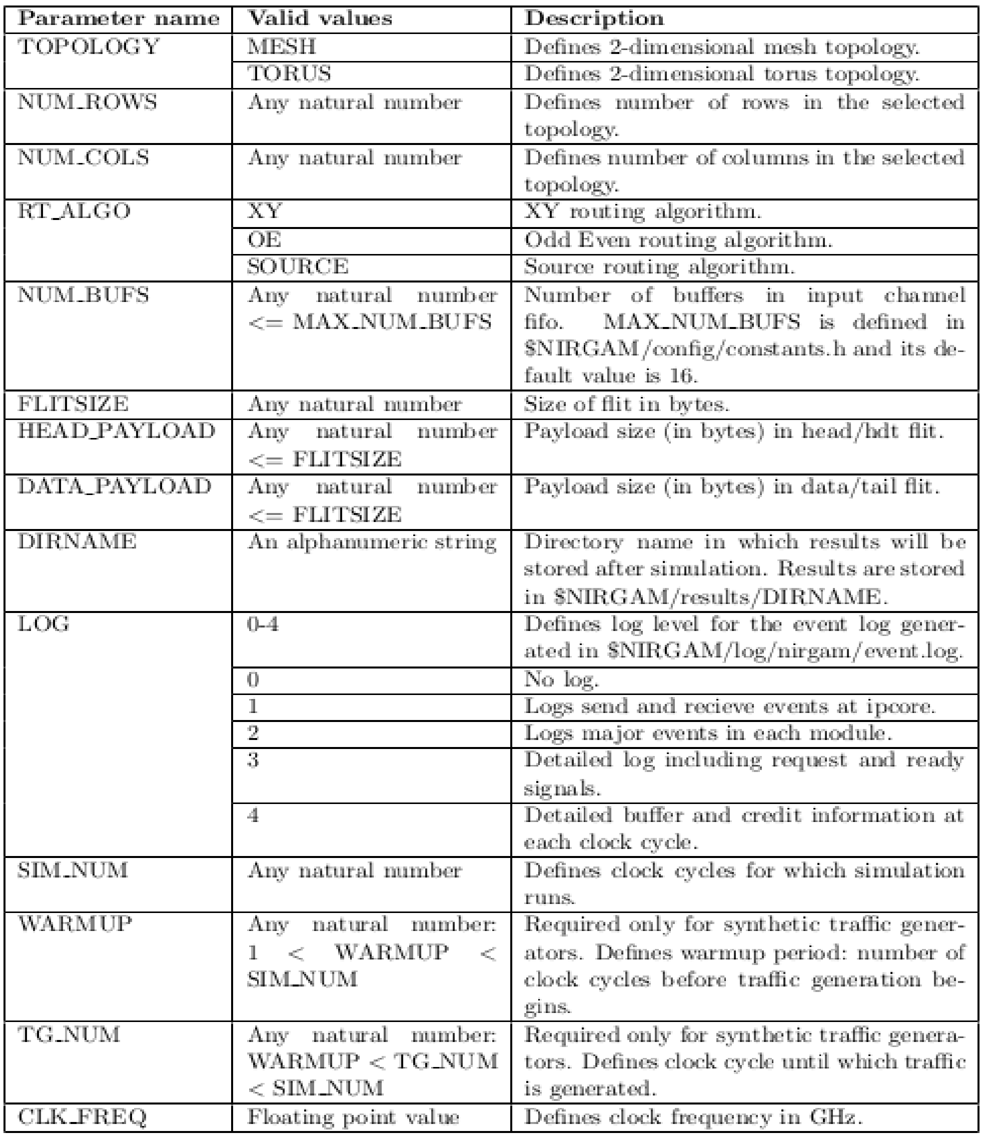
   dest id: tile id of destination tile. The function should return next hop direction as determined by route logic. Hence the return value should be one of: N, S, E, W, C.
5. Edit $NIRGAM/config/constants.h to include name of your routing algorithm(R)
6. Edit Constructor in $NIRGAM/core/Controller.cpp to enable the controller to attach router library
7. Edit Makefile to include name of source file in ROUTER SRCS
8. Run make. This should create library myrouter.so in $NIRGAM/router/lib.
9. New routing algorithm can now be used in NIRGAM. Configure parameter RT_ALGO in nirgam.config to use your routing algorithm.

4.5 MODULE DESCRIPTION

4.5.1 Specification of Network Configuration

- nirgam.config

This is the main configuration file that contains parameters specific to NoC design and simulation.



Parameter name	Valid values	Description
TOPOLOGY	MESH	Defines 2-dimensional mesh topology.
	TORUS	Defines 2-dimensional torus topology.
NUM_ROWS	Any natural number	Defines number of rows in the selected topology.
NUM_COLS	Any natural number	Defines number of columns in the selected topology.
RT_ALGO	XY	XY routing algorithm.
	OE	Odd Even routing algorithm.
	SOURCE	Source routing algorithm.
NUM_BUFS	Any natural number <= MAX_NUM_BUFS	Number of buffers in input channel fifo. MAX_NUM_BUFS is defined in $NIRGAM/config/constants.h and its default value is 16.
FLITSIZE	Any natural number	Size of flit in bytes.
HEAD_PAYLOAD	Any natural number <= FLITSIZE	Payload size (in bytes) in head/hdt flit.
DATA_PAYLOAD	Any natural number <= FLITSIZE	Payload size (in bytes) in data/tail flit.
DIRNAME	An alphanumeric string	Directory name in which results will be stored after simulation. Results are stored in $NIRGAM/results/DIRNAME.
LOG	0-4	Defines log level for the event log generated in $NIRGAM/log/nirgam/event.log.
	0	No log.
	1	Logs send and recieve events at ipcore.
	2	Logs major events in each module.
	3	Detailed log including request and ready signals.
	4	Detailed buffer and credit information at each clock cycle.
SIM_NUM	Any natural number	Defines clock cycles for which simulation runs.
WARMUP	Any natural number: 1 < WARMUP < SIM_NUM	Required only for synthetic traffic generators. Defines warmup period: number of clock cycles before traffic generation begins.
TG_NUM	Any natural number: WARMUP < TG_NUM < SIM_NUM	Required only for synthetic traffic generators. Defines clock cycle until which traffic is generated.
CLK_FREQ	Floating point value	Defines clock frequency in GHz.

Figure 4.2 nirgam.config

- application.config

  This configuration file lets you specify application mapping. It accepts input as pairs of tileID and name of application library to be attached to the tile. The applications that can be attached are stored in $NIRGAM/application/lib. Tiles not specified in this file have no application attached to them and behave as only routers.

4.5.2 Dual Partition of the Network

We propose four routing algorithms :

- North-Last (or) South-Last
- West-Last (or) East-Last

North-Last Routing

- If there exists multicasting destinations in the NE part, transmit the packet to the East output port.
- If there exists multicasting destinations in the NW part, transmit the packet to the West output port.
- Packet to multicasting destinations in the SE part is transmitted through the South output port if there are no destinations in the NE and DE parts and at least one destination locates in the SW or DS part. Otherwise, a packet to destinations in the SE part is transmitted through the East output port.
- Packet to multicasting destinations in the SW part is transmitted through the South output port if there are no destinations in the NW and DW parts and at least one destination locates in the SE or DS part. Otherwise, a packet to destinations in the SW part is transmitted through the West output port.

West-Last Routing

- If there exists multicasting destinations in the NW part, transmit the packet to the North output port.
- If there exists multicasting destinations in the SW part, transmit the packet to the South output port.
- Packet to multicasting destinations in the NE part is transmitted through the East output port if there are no destinations in the NW and DN parts and at least one destination locates in the SE or DE part. Otherwise, a packet to destinations in the NE part is transmitted through the North output port.
- Packet to multicasting destinations in the SE part is transmitted through the East output port if there are no destinations in the SW and DS parts and at least one destination locates in the NE or DE part. Otherwise, a packet to destinations in the SE part is transmitted through the South output port.

4.5.3 Multicast Routing Algorithm

The multicast routing algorithm comprises of two applications. The application can be attached to any tile in the NoC.

- App_send.cpp

  This application is attached to the source tile which sends the packets to the multicasting destination tiles. Each packet is divided into three flits - head flit, data flit and tail flit.

- App_recv.cpp

  This application is attached to all the destination tile which recieves the packets from the source tile.

4.5.4 Load Balancing

Heterogeneous multicast traffic can lead to imbalanced network traffic between the Dual partitioned networks. We propose to balance the network load by distributing the packets that are injected into the sub-networks N0 and N1.

The figure 4.3 shows the distribution of packets in a 3x4 Mesh Topology. The source tile is 3 and the remaining tiles are the destinations. The packets are routed using XY Routing algorithm without load balance.

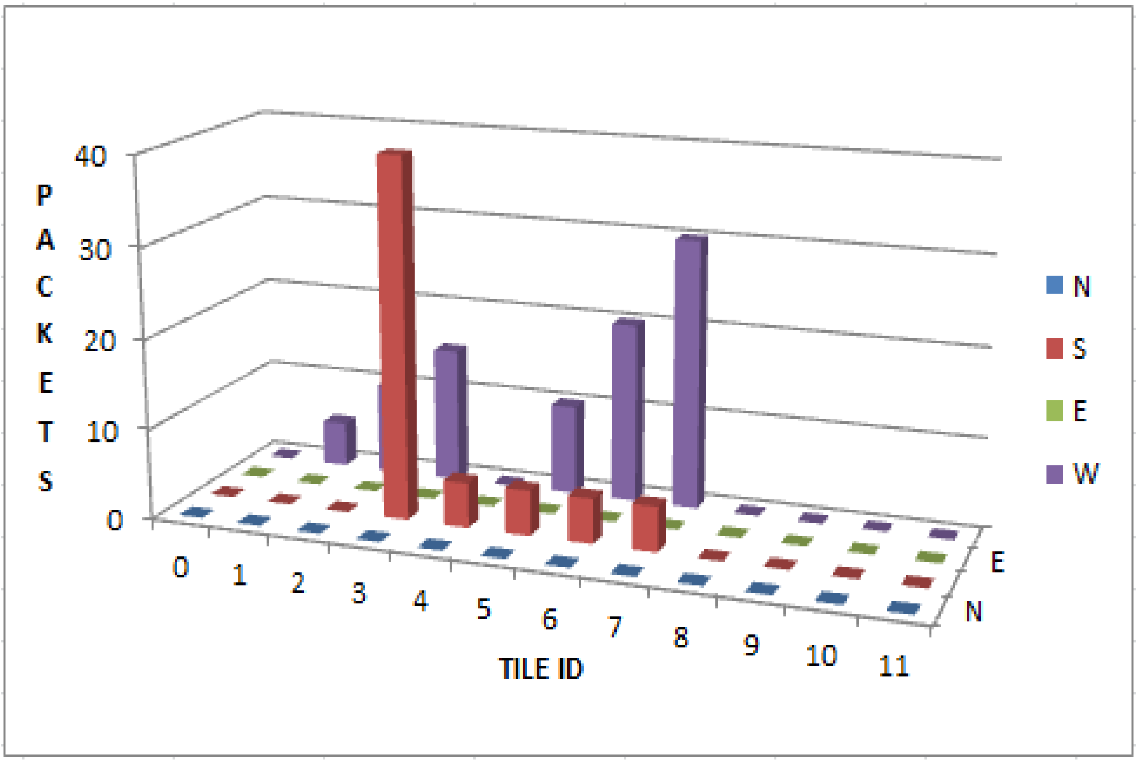


Figure 4.3 NoC without Load Balance

The figure 4.4 shows the distribution of packets in a 3*4 Mesh Topology. The source tile is 3 and the remaining tiles are the destinations. The packets are routed using DPM Routing algorithm with load balance. On comparing figure 4.3 and figure 4.4 , in figure 4.4 the packets are distributed across the network thus utilizing the resources efficiently.

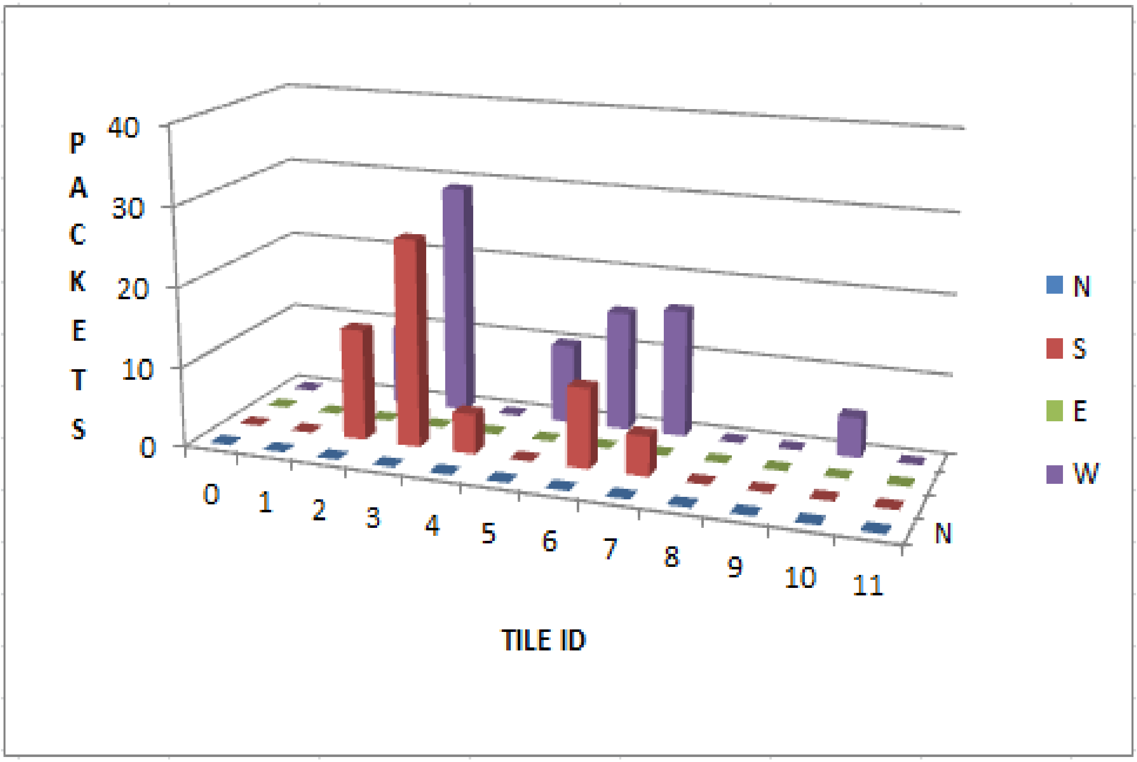


Figure 4.4 NoC with Load Balance

Pseudo code

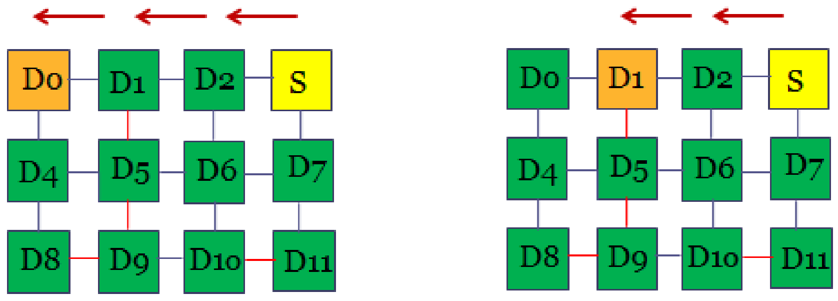
```
Input: NoC topology information
Output: Distribution of traffic across the network
initialize counter to k
begin loop
if ( counter $\prec 2k$)
    inc counter
    DPM
else if ( counter $\succ 0$)
    dec counter
    DPM
else {
if (counter $\preceq 0$)
    XY Routing
else
    YX Routing
}
end loop
```

4.5.5 Fault Tolerance

A fault happens whenever a link or tile fails in the NoC. Fault Tolerance is defined as the ability of the NoC to the route the packets to correct destination even in case of faults.

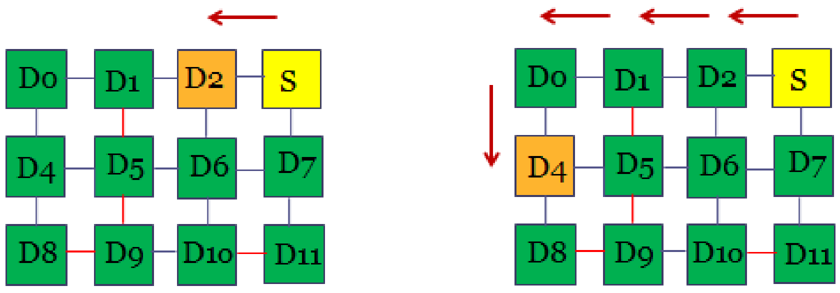
- Link fault - When a link between two tiles fail
- Tile fault - When all the four links of a tile fail

We modify the XY-Routing algorithm to provide fault tolerance. The failed links can be specified in the configuration file. Fault Tolerance algorithm uses this information to route the packets.



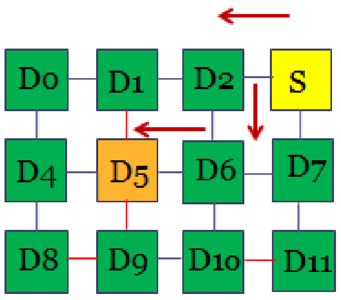
[1] [2]

Figure 4.5 Fault Tolerance

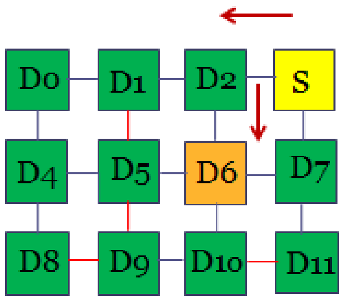


[3] [4]

Figure 4.6 Fault Tolerance

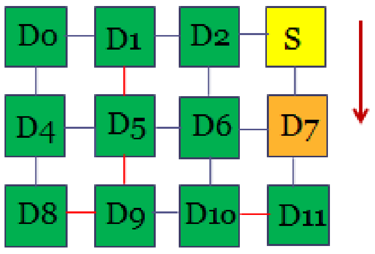


[5]

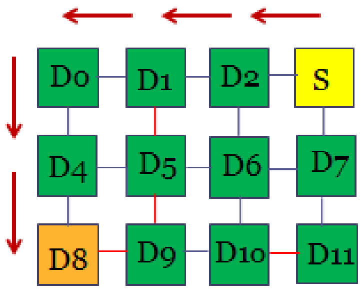


[6]

Figure 4.7 Fault Tolerance

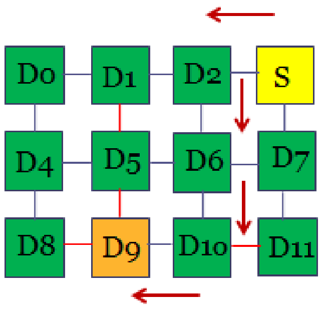


[7]

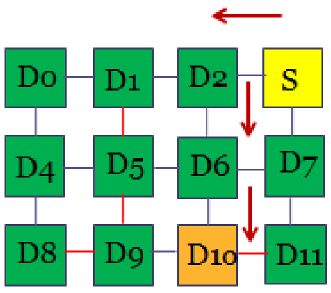


[8]

Figure 4.8 Fault Tolerance

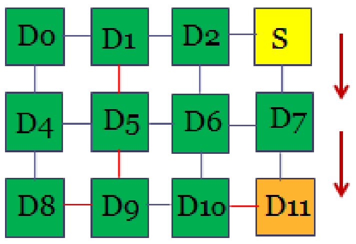


[9]



[10]

Figure 4.9 Fault Tolerance



[11]

Figure 4.10 Fault Tolerance

Pseudo code

Input: NoC configuration

Output : Routing decisions without any faults

```
begin

initialize fail_matrix

set dir_x and dir_y

for i in 1 to failno

begin loop
    if (id = fail_matrix[i][0])
    check if (next_id in fail_matrix[i][1])
    decide the next possible direction
end loop

return routing_decision

end
```

CHAPTER 5

PERFORMANCE ANALYSIS

5.1 RESULTS

In this section, we present a detailed evaluation of the DPM scheme with respect to the average throughput, power consumption and network load-balance. We compare the performance of the proposed DPM scheme with traditional multicast XY-routing algorithm. The simulation methodology is first involves measuring the performance metrics for different configurations of network. Then, the basic evaluation results will be presented and analyzed by plotting charts. We further show how our schemes are scalable different network sizes.

5.1.1 Evaluation

DPM algorithm's performance is compared with traditional XY multicast routing algorithm for metrics like throughput, packet latency, load distribution and power consumption.

- Throughput

  The following table shows 7 different NoC configurations and the corresponding chart compares the average throughput (in packets/clock cycle) between DPM and XY scheme for each NoC configuration.

Testcase	Size	Packets	Source	Destination	Linkfails
1	3x5	4	0	2,10	0
2	7x6	20	0	10,14,19	0
3	3x5	8	10	0,1,2,3,4,5,6,7,8,9,11	0
4	5x4	5	3	7,9,8	2
5	6x6	5	4	13,31,32	5
6	6x6	15	0	8,19,21,31,33	10
7	3x4	5	3	0,1,2,4,5,6,7,8,9,10,11	4

Table 5.1 Test cases

Table 5.1 represent the various NoC configurations that have been used to analyze the performance of the DPM algorithm.

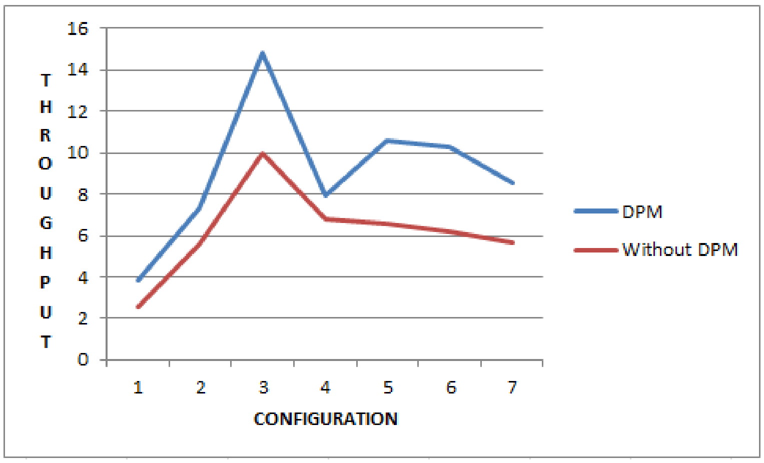


Figure 5.1 Average throughput for different NoC configurations

Figure 5.1 represents comparison between the average throughput obtained in traditional multicast algorithm and DPM algorithm. Figure 5.2 shows the throughput at each tile when traditional and DPM algorithms are run for testcase 7.

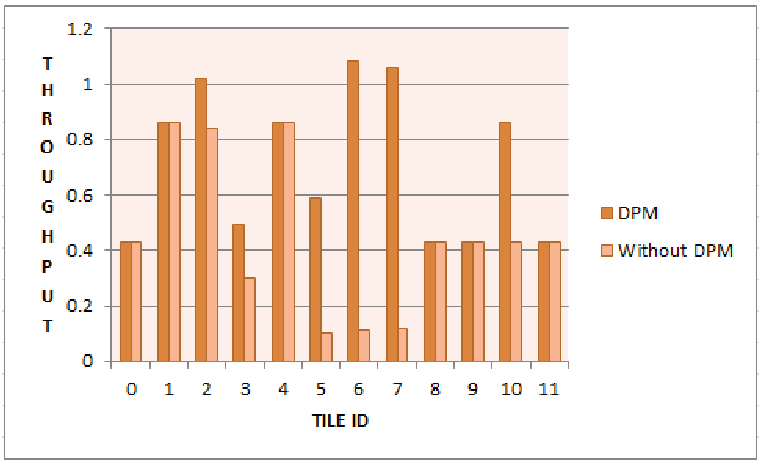


Figure 5.2 Throughput recorded per tile for testcase 7 (in table 5.1)

- Power consumption

  Figure 5.3 depicts the average power consumption of each testcase in Table 5.1 while Figure 5.4 represents gives the power consumption of each tile for testcase 6.

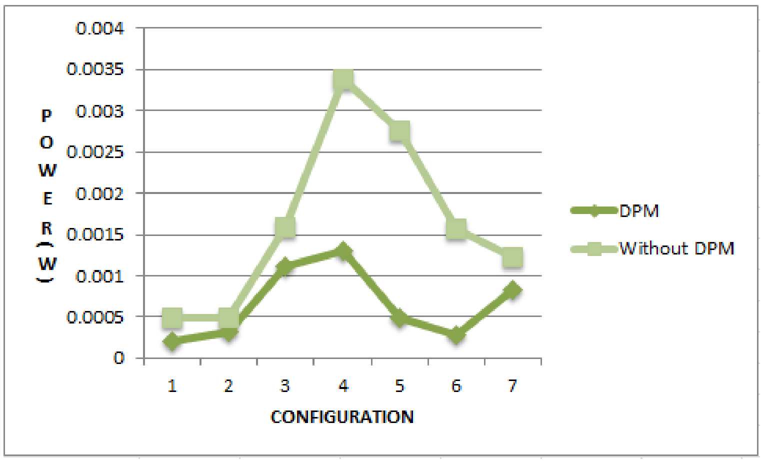


Figure 5.3 Average power consumption for different NoC configurations

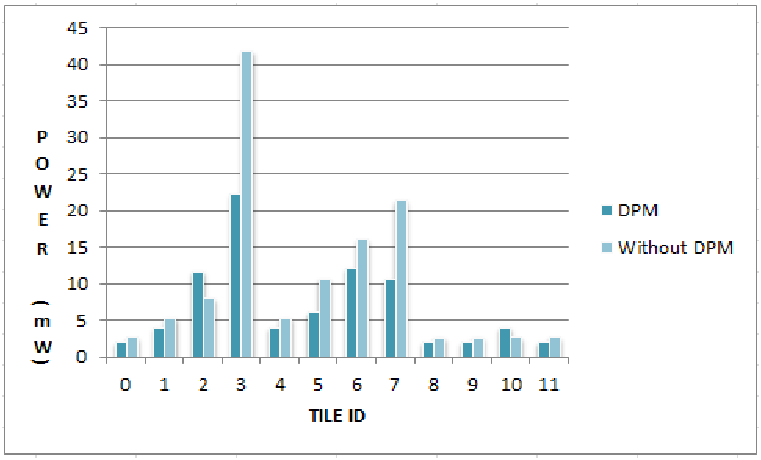


Figure 5.4 Power consumed per tile for configuration 7 (in table 6.1)

- Average Packet Latency

  Average packet latency is defined as the time required for a packet to traverse from the source to destination.

$$avg.\ latency = \frac{1}{k}\ \Sigma_{i=1}^{k}\ lat_i$$

where k - number of packets

By varying the packet injection rates for each configuration, the average packet latency is computed and plotted. Figure 5.5 depicts this symmetric graph.

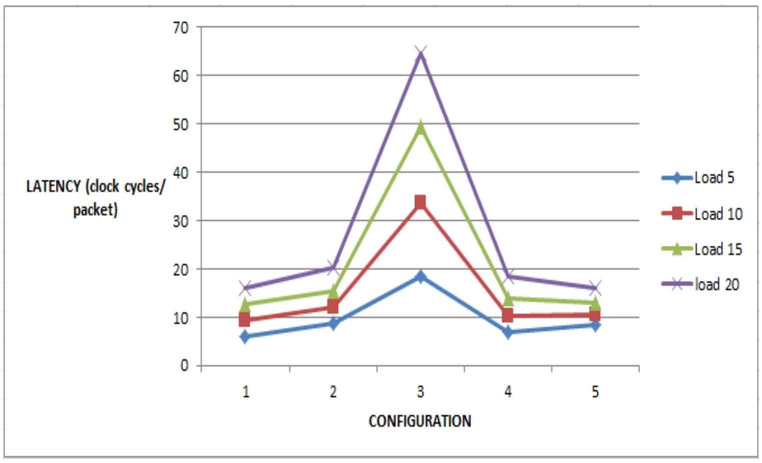


Figure 5.5 Average Latency vs Traffic Load

- Load Balancing

  The load distribution achieved by DPM scheme is far more superior to the traditional XY routing algorithm used. Figures 5.6 and 5.7 illuminates the difference.

  Figure 5.6 shows the load distribution for testcase 3 in traditional multicast algorithm. The high spikes shows the over utilization of a few particular tiles.

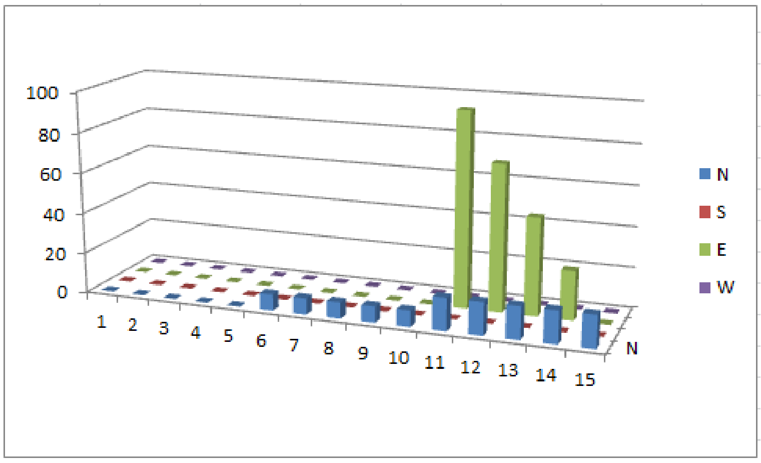


Figure 5.6 Testcase 3 : Without DPM

Figure 5.7 shows the even distribution of load after the application of DPM algorithm. The number of peaks have been eliminated significantly.

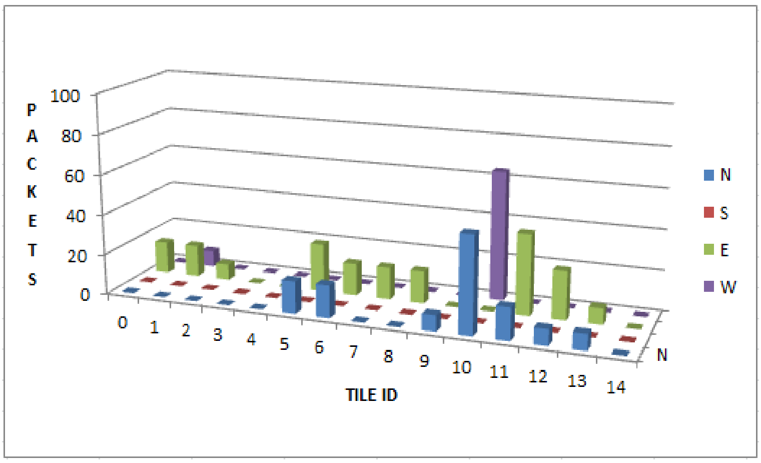


Figure 5.7 Testcase 3 : DPM

When we check the efficiency of DPM algorithm by increasing the network size, it still performs consistently. This is shown in Figures 5.8 and 5.9.

Figure 5.8 shows multicasting for configuration 6 which is a 6x6 NoC. Uneven load distribution with steep peaks are seen.

Figure 5.9 shows traffic distribution by applying DPM algorithm. It easily shows the even distribution of traffic by better utilization of the network resources.

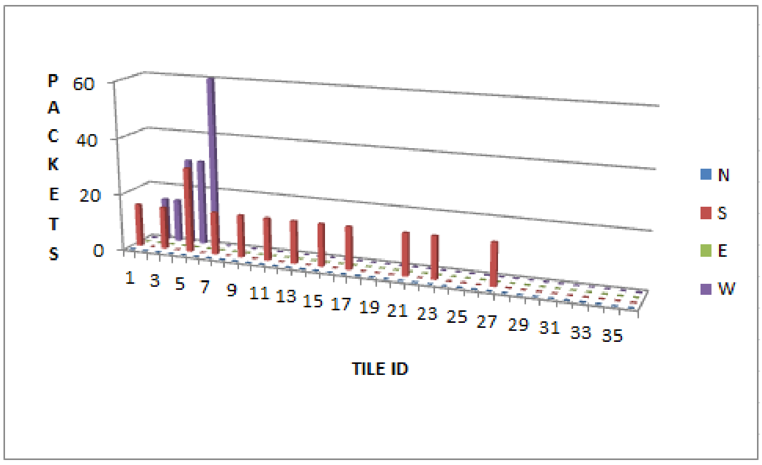


Figure 5.8 Configuration 6 : Without DPM

From these four figures 5.6-5.9, it is clear that DPM algorithm performs consistently well with networks of different sizes and packet injection rates. This shows the scalability of DPM algorithm across multiple networks.

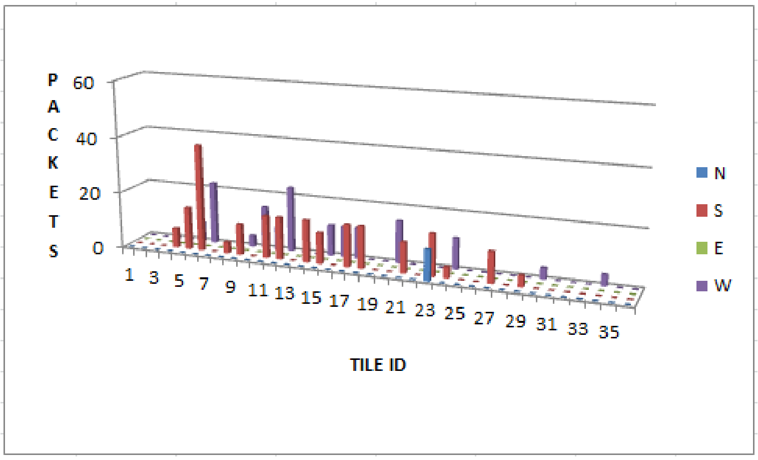


Figure 5.9 Configuration 6 : DPM

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

From the simulation results, it has been found that average latency throughput for routing packets in NoC is considerably reduced when Dual Partition Multicast (DPM) Routing Algorithm is used in place of other existing routing algorithms. DPM routing algorithm also distributes the traffic across the entire network. Apart form traffic distribution it can also handle faulty links and faulty nodes. Repeating the simulation over number of trials by varying the number of tiles and traffic load, it is found that DPM routing algorithm increases the average throughput and balances the load significantly.

6.2 FUTURE WORK

In future, we can make this Dual Partition Routing Algorithm to work on any of the other efficient topologies and can get better performance. This algorithm basically works on the principle of efficient distribution of traffic load along the path between source to destination and is topology independent.So, whatever maybe the underlying topology, the routing algorithm works smooth.

The tuning of the power consumption and energy used for this entire setup by adjusting the number of virtual channels and traffic load leading to increased throughput could be a great optimizing study. The application mapping of any of the real-time traffic scenario could be a great work to test the robustness of this project.

Eventually, the software project can be extended and implemented in hardware using Field Programming Gate Array (FPGA) board. Using the software, we can write the Dual Partition Multicast Routing Algorithm in the memory processing elements of the individual tile and configure to work the tile according to our required functionality. By this method, we can check the performance through hardware analysis.

REFERENCES

[1] J. Li, L. Shi, C. J. Xue, Y. Xu, Dual portioning multicasting for high performance on-chip networks, in: Journal of Parallel and Distributed Computing, Elsevier 74 (2014) pp. 1858-1871.

[2] M. Zhong, Z. Wang, H. Gu, An Improved Minimal Multicast Routing Algorithm for Mesh-based Networks-on-Chip, in: Proceedings of the 2014 IEEE 20th International Symposium on High- Performance Computer Architecture, HPCA14, 2014, pp. 25-42.

[3] S. Ma, N.E. Jerger, Z. Wang, Supporting efficient collective communication in NoCs, in: Proceedings of the 2012 IEEE 18th International Symposium on High- Performance Computer Architecture, HPCA12, 2012, pp. 112.

[4] L. Wang, Y. Jin, H. Kim, E.J. Kim, Recursive partitioning multicast: a bandwidth efficient routing for networks-on-chip, in: The 3rd ACM/IEEE International Symposium on Networks-on-Chip, NOCS09, 2009, pp. 6473.

[5] A. Kahng, B. Lin, K. Samadi, R. Ramanujam, Trace-driven optimization ofnetworks-on-chip configurations, in: 47th ACM/IEEE Design Automation Conference, DAC10, 2010, pp. 437442.

[6] W. Zuo, S. Feng, Z. Qi, J. Weixing, L. Jiaxin, D. Ning, X. Licheng, T. Yuan Q. Baojun, Group-caching for NoC based multicore cache coherent systems in: Proceedings of the Conference on Design,

Automation and Test in Europe,DATE09, 2009, pp. 755760.

[7] S. Rodrigo, J. Flich, J. Duato, M. Hummel, Efficient unicast and multicast support for cmps, in: Proceedings of the 41st Annual IEEE/ACM International Symposium on Micro architecture, MICRO08, 2008, pp. 364375.

[8] NIRGAM, A simulator for noc interconnect routing and application modeling,2007. http://nirgam.ecs.soton.ac.uk/.

[9] W. Zuo, S. Feng, Z. Qi, J. Weixing, L. Jiaxin, D. Ning, X. Licheng, T. Yuan,Q. Baojun, Group-caching for NoC based multicore cache coherent systems,in: Proceedings of the Conference on Design, Automation and Test in Europe, DATE09, 2009, pp. 755760.

[10] N. Agarwal, T. Krishna, L.-S. Peh, N. Jha, GARNET: a detailed on-chip network model inside a full-system simulator, in: IEEE International Symposium on Performance Analysis of Systems and Software, 2009, ISPASS 2009, 2009, pp. 3342.

[11] P. Abad, V. Puente, J.-A. Gregorio, MRR: Enabling fully adaptive multicast routing for CMP interconnection networks, in: Proceedings of the 15th International Symposium on High Performance Computer Architecture, HPCA09, 2009, pp. 355366.